RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
LEHR- UND FORSCHUNGSGEBIET THEORETISCHE INFORMATIK
UNIV.-PROF DR. PETER ROSSMANITH

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
LEHRSTUHL FÜR INFORMATIK VI
UNIV.-PROF DR. HERMANN NEY

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
LEHRSTUHL FÜR INFORMATIK VIII
UNIV.-PROF DR. LEIF KOBBELT

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
LEHRSTUHL FÜR INFORMATIK IX
UNIV.-PROF DR. THOMAS SEIDL

RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
MEDIZINISCHE FAKULTÄT
INSTITUT FÜR MEDIZINISCHE INFORMATIK
UNIV.-PROF DR. DR. KLAUS KABINO

Hauptseminar im Wintersemester 2013/14

Medizinische Bildverarbeitung

Volume 9, Band 2
ISSN 1860-8906
ISBN 978-3-9813213-7-1

Aachener Schriften zur Medizinischen Informatik

Aachener Schriften zur Medizinischen Informatik
ISSN 1860-8906
ISBN 978-3-9813213-7-1

Herausgeber: Institut für Medizinische Informatik der RWTH Aachen
Pauwelsstr. 30
D-52074 Aachen
Geschäftsführender Direktor: Universitätsprofessor Dr. Dr. Klaus Kabino

Preface

In the winter semester 2013/2014 twelfe students of the Medical Image Processinglecture took part in this seminar. They had the opportunity to intensify their newfound knowledge of medical imaging by self-applying new learnt methods on state-of-the-art research topics.

For two years, the seminar has been completely performed in English, making tribute to the diversity of the student population. The event was jointly organized by the department of computer science, chairs of computer science 6, 8 and theoretical informatics, and the de-partment of medical informatics. This meeting of different research areas promised a broad base of interesting problems demanding interdisciplinary solutions for presentations and dis-cussions.

The seminar was the occasion for the students to apply scientific methodology: literature re-search, critical thinking, reimplementation and presentation to the peers. Placing the articles in a broader context led to fruitful and constructive discussions, whose synthesis are summarised in these proceedings.

We would particularly like to thank Professors Thomas Deserno, Leif Kobbelt, Hermann Ney and Peter Rossmanith, as well as the involved supervisors, without whom this interdiscipli-nary seminar would not be possible. Last but not least, we would like to thank all participating students for their engagement and passion in research for the field of medical imaging.

Aachen, February 2014

Daniel Haak, Antoine Serrurier, Stephan Jonas

Inhalt

Hauptseminar Medizinische Bildverarbeitung

Vorträge der Blockveranstaltung im Wintersemester 13/14

theoretischer Teil/Prof. Rossmanith, 30.01.2014

13:15-14:00	Graph-Cut Segmentation in Medical Images	Referent: *Markus Joppich* Betreuer: *Stephan Jonas*
14:00-14:15	Diskussion	
14:15-15:00	Automatic segmentation of the heart	Referent: *Fatima Bushra*
15:00-15:15	Diskussion	Betreuer: *Antoine Serrurier*
15:15-16:00	MRI automatic segmentation of the bone using statistical shape models	Referent: *Eugen Beck* Betreuer: *Antoine Serrurier*
16:00-16:15	Diskussion	
16:15-17:00	Automatic Skin Lesion Segmentation Based on Texture Analysis and Supervised Learning	Referent: *Aly Mahmoud* Betreuer: *Daniel Haak*
17:00-17:15	Diskussion	

praktischer Teil/Prof. Seidl/Prof. Ney, 27.01.2014

14:00-14:45	Methoden zur Optimierung des K-Means Algorithmus für große Datenmengen	Referent: *Markus Harmsen* Betreuer: *Sergej Fries/Seidl*
14:45-15:00	Diskussion	
15:00-15:45	Efficient similarity joins with MapReduce	Referent: *Luzia Beisiegel*
15:45-16:00	Diskussion	Betreuer: *Sergej Fries/Seidl*
16:00-16:45	Video-Based Activity Recognition during Trauma Resuscitation	Referent: *Eric Marre* Betreuer: *Jens Forster/Ney*
16:45-17:00	Diskussion	
17:00-17:45	Multiple Foreground Co-segmentation for Medical Images	Referent: *Jens Böttcher* Betreuer: *Yannick Gweth/Ney*
17:45-18:00	Diskussion	
18:00-18:45	Graph-Based Multi-Object Tracking	Referent: *Dominic Chmiel*
18:45-19:00	Diskussion	Betreuer: *C. Oberdörfer/Ney*

praktischer Teil/Prof. Kobbelt/Prof. Deserno, 31.01.2014

09:00-09:45	Fused DTI/HARDI Visualization	Referent: *Sathvik Parekodi*
09:45-10:00	Diskussion	Betreuer: *D. Sibbing/Kobbelt*
10:00-10:45	Oberflächenrekonstruktion medizinischer 3D-Daten	Referent: *Robert Schwieger*
10:45-11:00	Diskussion	Betreuer: *Wagenknecht/Deserno*

Optimizing K-Means for BigData

Markus Harmsen

Zusammenfassung

Data clustering has drawn attention in many applications like data mining, pattern classification or image segmentation. Although more than half a century old, K-Means is often used due to its simplicity and well proved results, but the steady growing volumes of information and decreasing costs of hardware are demanding for parallel execution, for what most K-Means implementations are not capable of by default. This paper will introduce the basic algorithm and show how it can be extended to run in parallel on the MapReduce framework. Afterwards we iteratively discuss how to improve the algorithm further and how to parallelize the extended version.

Keywords: K-Means{∅, ++, ‖}, Clustering, MapReduce, Parallelization

1 Introduction

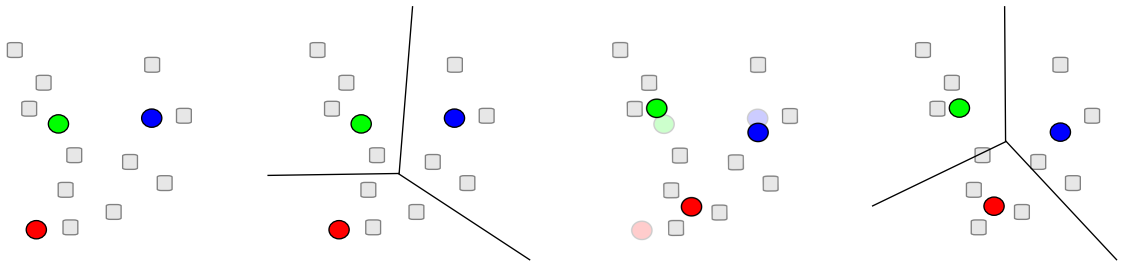
Data clustering is often the key in data management and spanning a wide field of applications like data mining, image segmentation, machine learning, artificial intelligence or pattern classification. Although many approaches exist which allow clustering, a more than half a century old algorithm, K-Means is by far the most popular clustering tool used in scientific and industrial applications [1]. Usages in the medical domain are for example medical data mining, e.g. knowledge induction from medical databases [2], genome analysis [3] or image processing, e.g. contrast enhancement and clustering segmentation of gray level images [4] which are very common in the medical area. Reasons for the popularity of K-Means could be its well proven results or its simplicity: select k random cluster means, assign each object to the nearest cluster mean, for each cluster compute a new mean by averaging the objects in it and repeat the last two steps until no cluster mean changed. Over the years the steady growing volumes of information and decreasing costs of hardware are demanding for parallel execution in many areas as well as for clustering and K-Means. Although the basic K-Means algorithm is not parallelized by default it can be easily extended to run in a parallelized way. As an example already in 1999 I. Dhillon and D. Modha presented a parallel implementation of K-Means using the message passing model [5]. Since then, lots of parallel implementations have been introduced [6, 7, 8] but often implementations assume that all objects can reside in memory or they use restricted and custom programming models to achieve parallelization, which can therefore depend on custom computation cluster setups and prevent generalization or portability.

The MapReduce [9] framework is a programming model and an associated implementation for processing large distributed datasets that has been developed by Google in 2004. It has the advantage of providing an higher abstraction layer for parallelizing applications, where the developer only has to supply defined methods. It has drawn lot of attention over the years since it has been published and many custom implementations, besides the proprietary one, have been developed for various languages and often released under a public license. A common used one is the Apache Hadoop project [10], which even provides a built-in K-Means implementation via the Apache Mahout library [11].

In this paper we will firstly introduce the basic K-Means algorithm and after providing more details on the MapReduce framework, we will see how to apply K-Means on MapReduce. Afterwards we iteratively discuss how to improve the algorithm further and how to parallelize the extended version again. Experiments with our own implementation are then going to proof the enhancements and will be critically compared to previous published results.

2 Related work

This paper is mainly based on three previous papers that have been published before. At first Zhao et al. [12], who presented in 2009 their approach to parallelize the K-Means algorithm using MapReduce. We will not directly follow the idea presented by them, since they mainly used string processing to pass



(a) initial (b) assignment (c) update (d) assignment

Abb. 1.1: 4 Steps of the algorithm. 1.1(a) random observations selected as means, 1.1(b) assign observations to means, 1.1(c) update means, 1.1(d) newly assign observations to means. Notice that the observation on the "cluster-line" in 1.1(d) has been assigned to the bottom mean.

the numeric data. To improve the initialization phase of K-Means, we then utilize the approach by D. Arthur and S. Vassilvitskii called K-Means++ [13]. Here they did not only present their smarter selection routine, they also proved its competitive properties. The last paper was published in 2012 by Bahmani et al. and presents a way to apply the ideas behind K-Means++ on the MapReduce framework, called K-Means‖. We mainly refer to this paper, because we will compare our experimental result on all three approaches with their published results which also cover the basic K-Means as well as K-Means++ and of course K-Means‖.

3 Methods

In this section the basic K-Means algorithm is presented and the MapReduce framework is introduced at first. Afterwards, we will iteratively enhance the basic algorithm to be parallelized based on the MapReduce framework and apply further improvements to the initialization step.

3.1 K-Means Algorithm

The K-Means "standard" algorithm was already proposed by Lloyd in 1957 but in another context and not published before 1982 [14]. In 1965 Forgy published essentially the same algorithm [15], therefore the "standard" K-Means is often referred as "Lloyd-Forgy-Algorithm", whereas the term K-Means was introduced by MacQueen in 1967 [16]. The algorithm proceeds as follows:

1. **Initial step**: choose equal distributed randomly k different observations from the dataset. These observations represent the initial cluster means.
2. **Assignment step**: assign each observation $x \in X$, where X is the set of all observations, to the cluster set S_i for which the intra-cluster variance is increased at least:

$$S_i = \{x | d(x, m_i) \leq d(x, m_j) \forall 0 \leq j < k\} \tag{1.1}$$

   where m_i donates the mean observation for the cluster set S_i and $d(y, z)$ the distance between y and z. In most cases the squared Euclidean distance is used:

$$d(y, z) = \sqrt{(y_1 - z_1)^2 + \ldots + (y_n - z_n)^2}^2 \tag{1.2}$$

   where y, z are points in the n-dimensional space, which results intuitively into assigning each observation to the "nearest" mean.
3. **Update step**: calculate new cluster mean m_i to be the mean of the observations in the according cluster S_i:

$$m_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j \tag{1.3}$$

(a) local optimum (b) global optimum

Abb. 1.2: Local optimum example. Assuming the situation in 1.2(a), K-Means stops here since the means will not change in the next iteration. Nevertheless the solution in 1.2(b) is the global optimum.

The algorithms stops when the means no longer change, i.e a local optimum has been found. An example run can be seen in figure 1.1. Notice that the K-Mean algorithm does not necessary result in a global minimum as seen in figure 1.2. To overcome this issue in, the algorithm could be executed more than once and simply returning the most optimal result. In the *assignment step* the distance function is not limited to the Euclidean one, e.g. the Manhattan distance could be used too. For an interesting view on the effects on different distance functions, see [17]. The most intensive calculations occur in the second step during the distance calculation d: if n is the total number of observations, the distance function d is called $n \cdot k$ times. Although some approaches exist, which for example use more efficient data structures like trees to reduce the number of computations needed [18], we will only use parallelizing to tackle this point.

3.2 MapReduce Framework

A classical attempt to develop a distributed application is often to use the Message Passing Interface (MPI) [19]. However, the abstraction level of the MPI is quite low and adds complexity to the desired computation.

In contrast to that, the MapReduce framework [9] allows a higher level of abstraction by providing a simple programming model, job management, cluster management and even a distributed file system [20]. Although experiencing some kind of "hype" since MapReduce has been published 2004 by Google, some criticize its lack of novelty [21, 22]. Nevertheless it has gained enormous attention and due to its widely usage and standardized design, vendors even provide MapReduce clusters on demand [23, 24], which increases its availability even further. There exist several implementations of the framework besides Google's proprietary implementation (C++) [9] like the widely used open source implementation Apache Hadoop (Java) [10] which also offers a distributed file system HDFS [25]. The framework is built on the following assumptions: the input records are stored on a distributed file system in a key/value fashioned way. The input records are divided into subsets and applied to the worker instances which call the *mapper* for each input record from the subset and emit the results via network in a key/value fashion to the shuffle phase. Here all results from the mappers are sorted by their key and delivered aggregated key-wise to the *reducers*. Notice that the mapping and the reduce phases are running in parallel and isolated, so that there is no knowledge shared between the mappers or the reducer instances besides the shuffle phase. An overview is shown in figure 1.3.

Under the MapReduce programming model, one needs only to provide implementations of the *map* and *reduce* parts - the execution and distribution is done by the framework afterwards. In addition to the *map* and *reduce* methods there are optional *configure* and *close* methods which are called before and after each map and reduce call to allow further processing.

3.3 Parallel K-Means based on MapReduce

As noticed before, most computation during the K-Means algorithm is done during the distance computation. The map function will now be assigning the observations to the nearest mean, whereas the reducer will update the new cluster means. After one iteration, the updated cluster means will be used by the workers again. The pseudo-code is shown in algorithm 1.

Although Algorithm 1 is a parallelized version of K-Means, it would perform poorly since the whole dataset, i.e. each point, is emitted by the mapper and therefore transferred via network. Here the "combiner" paradigm can be applied to lower the transferred data volume: since the reducer only depends on the sum and number of points in a cluster, the mapper could already sum them up before emitting the aggregated data - see Algorithm 2.

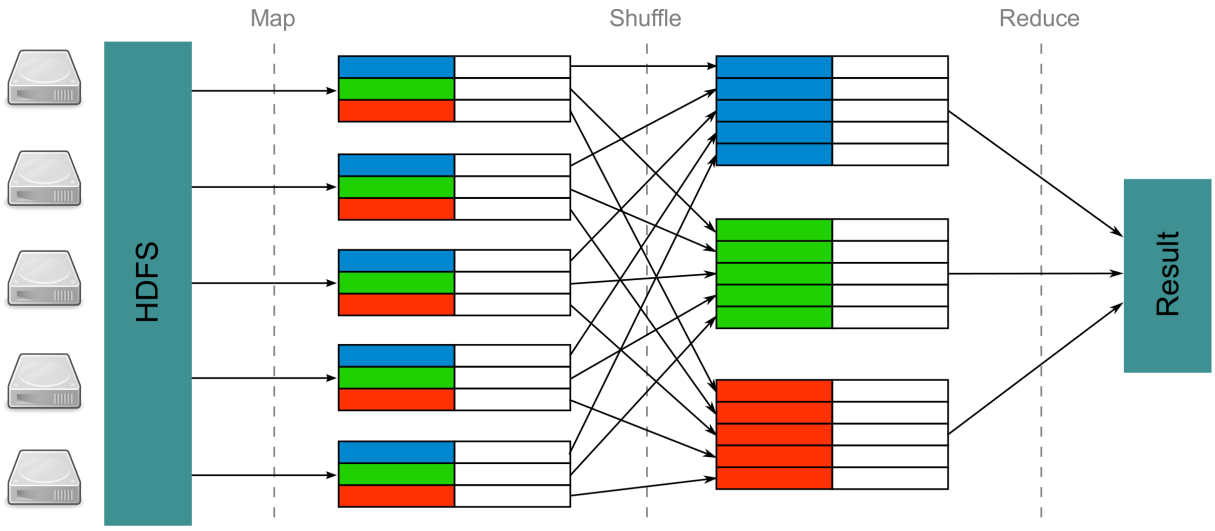


Abb. 1.3: Illustration of MapReduce. The distributed input-data is applied to the worker instances which emit the results to the shuffle phase. In the shuffle phase all emitted results are collected and aggregated by key. Afterwards the aggregated key/values are given to the reducers - one reducer instance per key.

Given that the reducer only accepts a list of points, the *PushLast* is used to extend a point by another dimension initialized by 1. When summing up, the last dimension now denotes the number of points in the current cluster-id *nc*. The length can now be received in the reducer by removing the last dimension from the transferred sum-point via *PopLast*. A similar approach has already been presented by Zhao et al. [12], although they used string concatenation and string parsing as "passing the length - workaround".

3.4 K-Means++: careful seeding

In the last section we have seen how to parallelize the basic algorithm of K-Means. Nevertheless the initial selected cluster means are still uniformly random, which may result in a worse local optimum and/or more iterations until convergence - see figure 1.4.

Hence, in this section, we describe a method proposed in 2007 by D. Arthur and S. Vassilvitskii called K-Means++ [13], which chooses random starting means with specific probabilities, i.e "a observation x is chosen as a cluster mean with probability to x's contribution to the overall potential". Their method does not only improve the basic K-Means in terms of accuracy, but should also often result in fewer needed iterations, is proposed as follows:

1. Choose an initial cluster mean c_1 uniformly random from all observations X.
2. Choose the next cluster mean c_i, $c_i = x \in X$ with probability:

$$P(x) = \frac{D^2(x, C)}{\sum_{x' \in X} D^2(x', C)} \tag{1.4}$$

   where C is the set of selected cluster means so far and $D^2(x, C)$ the minimal distance from x to the nearest cluster mean in C, i.e:

$$D^2(x, C) = d(x, c_n) \iff d(x, c_n) \leq d(x, c_o) \forall c_o \in C \tag{1.5}$$

   and $c_n \in C$ donates the "nearest" cluster mean for x.
3. Repeat step 2 until k cluster means have been selected.
4. Continue with the basic K-Means algorithm, i.e. the *assignment* and *update* step.

The authors of the K-Means++ algorithm also proved that their K-Means++ is $O(\log k)$ competitive, i.e. for any set of observations the total error of a clustering by K-Means++ is below $O(\log k)$ times the optimal clustering error. For a proof, we refer to the corresponding paper[13]. They furthermore stated that for very well formed datasets the algorithm is even $O(1)$ competitive, although these datasets are not explained further.

Algorithm 1 K-Means on MapReduce Naive

```
class Mapper
  method Configure()
    c ← LoadClusters()
  method Map(id i, point p)
    nc ← NearestClusterID(c, p)
    Emit(nc, p)

class Reducer
  method Reduce(clusterid n, points s)
    sum ← NewZeroPoint(s.first.length)
    for point p in s
      sum ← sum + p
    m ← ComputeCenter(sum, s.length)
    Emit(n, m)
```

Algorithm 2 K-Means on MapReduce

```
class Mapper
  method Configure()
    c ← LoadClusters()
    h ← AssociativeArray()
  method Map(id i, point p)
    nc ← NearestClusterID(c, p)
    pc ← PushLast(p, 1)
    h[nc] ← h[nc] + pc
  method close()
    for clusterid n in h
      Emit(n, h)

class Reducer
  method Reduce(clusterid n, points s)
    sum ← NewZeroPoint(s.first.length)
    for point p in s
      sum ← sum + p
    length ← PopLast(sum)
    m ← ComputeCenter(sum, length)
    Emit(n, m)
```

Alg. 1,2: K-Means Mapper implementations. The left algorithm shows the "naive" version, where each point is directly emitted and therefore causes high network traffic. The right variant uses a associative array the cache the cluster sums before emitting them. The *NewZeroPoint* function simply creates a vector of given size which values are all zero. Note that in this case, the length information, i.e. how many points have been summed up for each cluster mean, has be preserved somehow. Here an additional dimension is added (*PushLast*) by the Mapper and received (*PopLast*) by the Reducer later on.

3.5 K-Means‖: Scalable K-Means++

In the last section we have seen a major improvement to the initialization sequence of the K-Means algorithm. Although this enhances the performance in terms of a more optimal clustering as well as an often lower runtime of the basic algorithm, it has the downside of being highly sequential due to the probability computation: the next cluster mean is selected *dependently* on the previous selected cluster means so far. This sequential nature limits its applicability of being parallelized, e.g. as seen in equation 1.4, and can not be useful distributed on more than one computational unit. Although some approaches have been developed to improve the K-Means++ runtime like the stream based StreamKM++ by M. Ackermann et. al [26], which seems to perform well while making only a single pass over the observations, we will introduce a paralyzed attempt using MapReduce again.

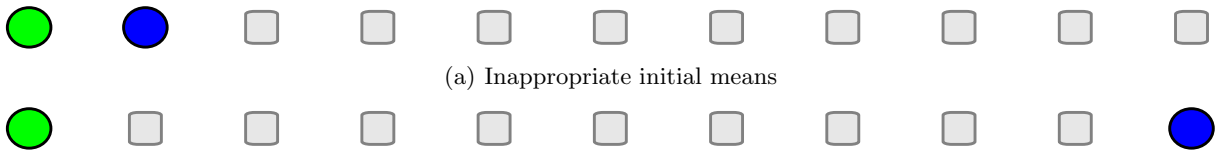
In this paper we have seen two approaches so far:

- K-Means: Select the initial cluster means completely random in one iteration, which can be done very fast and is easy to be parallelized since the next cluster mean selection does not depend on the previous ones.
- K-Means++: Select the initial cluster means in a specific way in k iterations so that they will properly result in a good clustering. Therefore choose each cluster mean by its potential to the final clustering, dependently on the previous selected ones and is therefore not easy to parallelize.

Hence in this section we describe a method proposed in 2012 by B. Bahmani et al. called K-Means‖ [27], which tries to use the best of both worlds:

- K-Means‖: Select more that one initial cluster means per iteration with respect to their possible potential to the overall clustering.

The K-Means‖ algorithm is proposed as follows:



(a) Inappropriate initial means

(b) Favorable initial means

Abb. 1.4: Effect of different initial selected means. 1.4(a) would result in more iterations, since the cluster means will only slowly move to the local optimum, whereas in 1.4(b) only one iteration is needed until the optimum is reached.

1. Choose an initial cluster mean c_1 uniformly random from all observations X.
2. Compute the *current* costs ψ of the selected initial mean:

$$\psi = \sum_{x' \in X} D^2(x', C) \tag{1.6}$$

   where C denotes the current cluster mean set and $D^2(x, C)$ again the minimal distance from x to the nearest cluster mean in C. We have seen this costs equation already in the denominator of the equation 1.4 to derive the probabilities used by K-Means++.
3. In $O(\log \psi)$ rounds do
   (a) Choose a set of next cluster means C', each observation $x \in X$ is sampled *independently* with probability:

$$P(x) = \frac{I \cdot D^2(x, C)}{\sum_{x' \in X} D^2(x', C)} \tag{1.7}$$

   where I donates a given *oversampling-factor*. This results in roughly I selected cluster means in the set C' per iteration.
   (b) Let C be $C \cup C'$

   Depending on the initial costs ψ and the defined oversampling-factor I, the initial cluster set C will now contain roughly $O(\log \psi) \cdot I$ observations selected as cluster means.
4. Weight each $c_i \in C$ to be the number of observations in X closer to c_i than any other observation in C.
5. Recluster the weighted cluster means C into k cluster means.
6. Continue with the basic K-Means algorithm, i.e. the *assignment* and *update* step.

The authors proved that if an $O(\log k)$ competitive algorithm, like K-Means++, is used during the reclustering (step 5), K-Means‖ is also $O(\log k)$ competitive - for the details, we refer to [27].

3.6 Parallel K-Means‖ based on MapReduce

In the last section we have introduced K-Means‖ for which we will now discuss a parallel implementation in this section. For this, we will refer to the algorithm steps presented before and focus only on the steps 2-5, since step 1 is trivial and we have already seen how to implement the basic algorithm used in step 6 in section 3.3. Step 2, computing the costs ψ - again used in the denominator of equation 1.7 in step 3a - can be implemented as follows: each Mapper works on a subset $X' \subseteq X$ and can compute the costs for X' with given cluster means C. Afterwards the Reducer simply sums up all emitted costs by the Mappers. The selection step 3a could be implemented as follows: each Mapper works independently on a subset $X' \subseteq X$ and samples each $x' \in X'$ according to equation 1.7. The current cluster means, costs as well as the oversampling-factor I are given to each Mapper during the initialization phase. In this step the Reducer phase is omitted since the Mappers already emit chosen observations directly. An example Mapper implementation is given in algorithm 3. The weighting step 4 is analogous to the costs computation: each Mapper works on a subset $X' \subseteq X$ and the cluster means C and weights for each

Algorithm 3 K-Means‖ cluster means selection Mapper

```
class Mapper
  method Configure()
    c ← LoadClusters()
    l ← LoadOversampling()
    psi ← LoadCosts()
  method Map(id i, point p)
    r ← l · NearestClusterDistance(c, p)/psi
    if r >= 1
      Emit(NULL, p)
    else if Rand(Math.ceil(1/r)) == 0
      Emit(NULL, p)
```

Alg. 3: The K-Means‖ cluster means selection Mapper computes the probability r of sampling the point p. Afterwards p is emitted if $r \geq 1$ or a random integer in range $0...\frac{1}{r}$ is exactly 0 - which is the case with probability r. Note that if no Reducer is given, the MapReduce framework uses the results from the Mapper phase and skips the Reducer phase. When no *key* is emitted, i.e the *key* is $NULL$, the keys are omitted in the resulting dataset and therefore here only contain the sampled points.

$c_i \in C$ the number of observations in X' closer to c_i than to any other cluster mean in C. The Reducer afterwards sums up the weights for each $c_i \in C$. For the recustering step 5 the authors simply assume that the number of cluster means in C is reasonable small, so it can be achieved with K-Means++ on a single machine.

4 Experiments

In this section we are going to present experiments for each of the approaches introduced before and compare them among themselves as well as to the published experimental results by B. Bahmani et al. [27]. We have implemented all previous K-Means variants in Java using the MapReduce framework in a sequential as well as in a parallel way, resulting in 6 applications: a) K-Means sequential, b) K-Means on MapReduce naive, c) K-Means on MapReduce, d) K-Means++ sequential, e) K-Means‖ sequential and f) K-Means‖ on MapReduce. Although the sequential versions, i.e. running on a single computational unit, could be implemented without the help of the MapReduce framework, we decided to use it here too - where applicable - to minimize the effects of different optimizations. As an example, data reading and writing is always been done using the IO functions of MapReduce. Furthermore no extra caching has been used, although all data would fit into memory, i.e. for each iteration the input is read again from the disk and the results are written onto disk if needed and are potentially too big to be kept in memory on huge datasets. Our implementation has been published on GitHub: https://github.com/MarkusHarmsen/KMeansOnMapReduce.

Our experiments are evaluated on a single laptop with a quad-core 2.7GHz processor, 8GB of memory and Solid-State-Drive installed. When running the sequential versions, the process was limited to one CPU only. During the MapReduce tests, we ran Hadoop in pseudo distributed mode and used all four available CPU cores. The reference results by B. Bahmani et al. have been evaluated by them on a quite huge Hadoop cluster of 1968 nodes, each with two quad-core 2.5GHz processors and 16GB of memory.

B. Bahmani et al. did also compare their method with a streaming algorithm for K-means clustering *partition* published in 2009 [28] which itself is based on K-Means++. For a detailed comparison we refer to their paper. Furthermore they stated that: a) K-Means‖ should be on par regarding the clustering costs with K-Means++ and b) K-Means‖ runs in fewer rounds when compared to K-Means++, resulting in a lower overall runtime. To prove their theorems they used - in their words - "massive, real-world datasets" which we will discuss in the next section.

4.1 Datasets

B. Bahmani et al. used three different datasets in their experiments: a synthetic spherical *GAUSSIAN* dataset which they generated, the *SPAM* dataset for "classifying email as spam or non-spam" having 4601

observations with 57 dimensions each [29] and the *KDDCup1999* dataset for "classifying network traffic as attack or non-attack" having 4898431 observations with 42 dimensions each [30]. For the sequential implementations they used the *GAUSSIAN* as well as the *SPAM* dataset, whereas the *KDDCup1999* dataset has only been used for the parallel implementations.

The *KDDCup1999* dataset does not contain only integer and real type features, but also categorical once, which are not compatible by default using K-Means, since only numerical type features are supported. Therefore some conversion has be be done in advance, which was not described by the initial paper. For that reason, and since we did not want to increase the dimensions of the dataset, we simply converted each category into an own integer representation, which of course could be misleading if we would use this dataset for experiments pointing the costs only (e.g. when non related categories are converted into a list of ascending integers, the first and last category will be seen as less related to each other by the classificator as two directly following ones, which of course is not true - each category should have the same relation among themselves). Therefore we will only use the *KDDCup1999* dataset to have a bigger set of observations and concentrate on runtime instead of costs.

Anticipating the discussion, we must at this point clarify that a) in our opinion both datasets are not "massive" in terms of a huge amount of observations or dimensions and b) at least the *KDDCup1999* has been criticized not be a "real-world dataset" [31]. In our experiments we therefore used the publicly available datasets *SPAM* and *KDDCup1999* but won't limit *KDDCup1999* dataset on parallel implementations only.

4.2 Results

Firstly we are going to present the results for the sequential implementations on the datasets introduced in the section before. For all experiments, we used 20 individual runs. Afterwards, we present the results for the parallel implementations using MapReduce, where only one run for each approach has been performed.

4.2.1 Sequential Results We present our results for the sequential implementations regarding the median costs on the *SPAM* dataset in table 1. For each cluster count $k \in \{100, 90, 80, 70, 60, 50, 40, 30, 20, 10\}$ and for each of the implementations we tested the performance in terms of costs, i.e:

$$costs = \sum_{x' \in X} D^2(x', C) \tag{1.8}$$

where X are all observations in the dataset, C the derived cluster means by the selected approach and $D^2(x', C)$ again the smallest distance between x' and any cluster mean in C. Furthermore the results by B. Bahmani et al. for $k \in \{20, 50, 100\}$ are shown here for comparison. As one can easily see, the results for K-Means and K-Means++ are nearly identical or only differ about the standard deviation we have observed during our experiments (see table 2). With respect to the K-Means‖ results, the $0.5k, r = 5$ as well as the $2k, r = 5$ parameterized setting are nearly identical to the previous published results when including the standard deviation, where $0.5k$ and $2k$ denote the oversampling factor I and $r = 5$ limits the rounds of the cluster means selection step to 5. Afterwards we also tried K-Means‖ without limiting the number of rounds with the previous used oversampling factor $I \in \{0.5k, 2k\}$ and observed that more initial rounds do not have an positive impact at least the *SPAM* dataset: for $k \geq 30$ the costs are nearly the same, for $k < 30$ lower rounds even lead to better results. Regarding the costs we could observe the anticipated lower costs when increasing the cluster count k, but it was unexpected to see the standard deviation raise, i.e. the randomness takes more into account, for all implementations except the basic K-Means one with decreasing k. For the basic K-Means the opposite seems to be true: with fewer clusters, the costs show a lower standard deviation. As an overall result we can say that K-Means++ and K-Means‖ clearly outperform the basic K-Means algorithm with respect to the costs.

The number of needed average iterations for the final basic K-Means always used in the last step are shown in table 3. When respecting the standard deviation presented in table 4, our results and the one published by B. Bahmani et al. are again nearly identical for all implementations. For the basic K-Means implementation, i.e random initial selected cluster means, we observed most iterations especially for $k \in 50, 40, 30, 20$. The extended implementations need fewer iterations due to the improved initial selection, whereas K-Means‖ with a limited number of selection rounds needs roughly about the same number of iterations as the K-Means++ one. Notice that the K-Means‖ setup with no round restriction

k		100	90	80	70	60	50	40	30	20	10
K-Means		1391	1399	1453	1417	1489	1496	1501	1505	1527	1695
		1384	-	-	-	-	1488	-	-	1528	-
K-Means++		24	28	34	40	51	66	89	132	248	849
		24	-	-	-	-	68	-	-	233	-
K-Means‖	$0.5k, r = 5$	23	28	33	41	51	66	89	133	238	865
		23	-	-	-	-	65	-	-	241	-
	$2k, r = 5$	23	28	34	40	52	67	91	132	250	849
		24	-	-	-	-	66	-	-	234	-
	$0.5k$	23	28	33	41	51	67	90	132	254	887
	$2k$	23	28	34	42	51	66	90	132	250	870

Tab. 1: *SPAM* dataset: median costs over 20 runs for the sequential variants with decreasing cluster count k scaled down by 10^5. For K-Means‖ the oversampling factor is given as well as a "round-limit" r, i.e. not more than r rounds are used to sample the initial cluster means. The second row on every variant shows the results of B. Bahmani et al. (over 11 runs) except for the K-Means‖ without limiting r where they did not provide any results.

k		100	90	80	70	60	50	40	30	20	10
K-Means		183.7	284.7	263.5	193.3	163.9	48.9	91.0	26.2	4.3	7.5
K-Means++		1.1	1.2	2.4	2.0	3.2	2.9	3.7	10.4	22.7	101.5
K-Means‖	$0.5k, r = 5$	0.8	1.1	1.2	1.6	2.2	2.7	4.3	6.2	29.1	116.0
	$2k, r = 5$	0.8	1.4	1.3	2.1	2.0	3.5	4.3	9.2	18.0	80.6
	$0.5k$	1.2	1.3	1.6	2.1	2.1	2.5	5.5	6.1	27.7	146.4
	$2k$	0.9	0.9	1.5	2.0	2.1	3.0	5.7	8.5	33.0	91.3

Tab. 2: *SPAM* dataset: Standard deviation for our sequential costs scaled down by 10^5. Since B. Bahmani et al. did not provide their standard deviation, only our results are presented here.

reaches indeed the lowest needed iterations afterwards. When taken the standard deviation into account, one can see that K-Means‖ without round restriction seems also lead to the most stable one.

We present the median runtime results for the sequential implementations in table 5. For the basic K-Means one can directly see the connection between needed iterations (table 3) and runtime: although the number of distance calculations raises when increasing k, the runtime for $k = 100$ with approx. 45 iterations and $k = 10$ with approx. 103 iterations is the same with 3.3 seconds each. The effects of the K-Means++ initial cluster selection can be seen here as well: when having a few clusters $k \leq 50$, the initial selection is fast and resulting in fewer needed iterations of the last step, so that the overall runtime (e.g 1.1 seconds for $k = 10$) is lower than the basic K-Means runtime (3.3 seconds for $k = 10$). For $k > 50$ instead, the time needed for the initial mean selection is too large to take benefit of the reduced iterations afterwards (3.3 seconds for the basic and 10.8 seconds for the K-Means++ version by having $k = 100$). For K-Means‖ with limited rounds $r = 5$ and reasonable small oversampling factor $I \in \{0.5k, 2k\}$ one can see the faster initial mean selection phase when compared with K-Means++. Since we have seen before that the number of final iterations does not differ much on K-Means++ and K-Means‖, here K-Means‖ clearly outperforms K-Means++ in terms of runtime as well as K-Means for every $k < 90$. The same does not hold for K-Means‖ without round restriction and for the same oversampling factors: here the increased number of initial selected means, and therefore long initialization phase, lead to the slowest runtime compared with all other approaches.

4.2.2 Parallel Results For the parallel implementations K-Means Naive, K-Means and K-Means‖ on MapReduce we show our results on the *SPAM* dataset in table 6 while $k = 50$. The naive K-Means implementation had an unfavorable run with 200 iterations compared to the smarter K-Means implementation with 104 iterations. Therefore the resulting runtime is very reasonable to be roughly twice as large on the naive version: 5183 versus 2651 seconds. It seems that the increased IO when using the naive approach is only evinced by comparing the system load during the run: whereas the naive one had a load of 2.3, the smarter K-Means version could use more CPU resources with a load of 2.5. Please notice that our

k		100	90	80	70	60	50	40	30	20	10
K-Means		44.6	51.4	60.7	74.1	112.5	123.3	184.1	181.2	172.6	103.1
		60.4	-	-	-	-	166.8	-	-	176.4	-
K-Means++		33.6	44.1	44.1	35.6	32.6	35.9	32.0	31.5	31.4	18.8
		36.6	-	-	-	-	42.2	-	-	38.3	-
K-Means‖	$0.5k, r = 5$	40.7	27.5	27.0	27.6	35.6	27.3	35.3	34.4	31.8	21.1
		30.2	-	-	-	-	30.8	-	-	36.9	-
	$2k, r = 5$	30.2	36.9	30.8	26.9	30.4	33.5	27.0	30.7	27.3	18.9
		29.7	-	-	-	-	28.1	-	-	23.3	-
	$0.5k$	28.1	26.5	32.9	27.5	27.2	26.6	36.5	22.3	23.0	18.9
	$2k$	10.8	11.3	13.8	10.3	15.0	16.6	18.8	16.9	16.3	11.7

Tab. 3: *SPAM* dataset: average iterations of the basic K-Means algorithm over 20 runs for the sequential variants with decreasing cluster count k. The second row on every variant shows the results of B. Bahmani et al. (over 10 runs) except for the K-Means‖ without limiting r where they did not provide any results.

k		100	90	80	70	60	50	40	30	20	10
K-Means		17.0	27.1	25.7	33.2	54.8	40.1	42.9	65.0	26.1	10.9
K-Means++		14.5	21.0	26.5	12.7	12.5	14.5	17.6	9.6	10.9	7.6
K-Means‖	$0.5k, r = 5$	18.4	11.4	13.2	11.4	13.5	12.3	12.9	19.8	18.1	11.6
	$2k, r = 5$	20.7	21.8	14.2	13.4	16.4	19.0	12.6	15.0	16.4	9.2
	$0.5k$	17.9	11.7	19.8	12.5	12.1	9.2	17.4	7.9	12.0	10.5
	$2k$	10.4	7.4	12.7	5.8	9.4	11.1	13.0	12.0	9.6	6.1

Tab. 4: *SPAM* dataset: Standard deviation for our sequential iterations of the basic K-Means algorithm. Since B. Bahmani et al. did not provide their standard deviation, only our results are presented here.

k		100	90	80	70	60	50	40	30	20	10
K-Means		3.3	3.1	3.8	4.5	6.1	6.5	8.6	6.9	5.5	3.3
K-Means++		10.8	10.1	8.6	7.5	6.4	5.4	4.0	3.1	2.2	1.1
K-Means‖	$0.5k, r = 5$	4.0	3.1	2.6	2.6	2.5	2.0	2.1	1.7	1.5	1.0
	$2k, r = 5$	6.2	5.9	5.0	4.0	3.9	3.2	2.5	2.4	1.7	1.2
	$0.5k$	12.0	10.6	10.4	8.3	7.6	6.5	6.0	4.6	3.7	2.8
	$2k$	54.9	47.9	40.2	29.9	23.8	17.9	13.7	10.7	7.5	4.5

Tab. 5: *SPAM* dataset: median runtime in seconds over 20 runs for our sequential methods. B. Bahmani et al. did not provide any runtime data for the sequential versions.

		Runtime	Iterations	System load
K-Means Naive		5182	200	2.3
K-Means		2651	104	2.5
K-Means‖	$0.5k, r = 5$	824	29	2.5

Tab. 6: *SPAM* dataset: MapReduce versions runtime in seconds, iterations and average system load on one single run with $k = 50$ clusters. For MapReduce only the $0.5k, r = 5$ parameter has been tested for K-Means‖. Notice the lower average system load on the naive implementation could indicate that the limiting factor seemed not to be the CPU.

	$k = 500$		$k = 1000$	
Approach	Runtime	Costs	Runtime	Costs
Random	300	$6.8 \cdot 10^7$	489.4	$6.4 \cdot 10^7$
Partition	420.2	7.3	1021.7	1.9
K-Means‖ $0.5k$ MapReduce	69	19	42	5.2
K-Means‖ k MapReduce	75.6	7.7	89.1	2.0
K-Means‖ $10k$ MapReduce	75.7	5.8	101.0	1.6
K-Means‖ $0.5k, r = 5$ Sequential	700	3.6	-	-
K-Means++	-	-	-	-

Tab. 7: *KDDCup1999* dataset: Random, Partition and K-Means‖ MapReduce results taken from B. Bahmani et al. The costs have been scaled down by 10^{10}. Notice that the K-Means++ run has been canceled after 12 hours without leaving the initial cluster mean selection phase.

own experiments have been done on a single machine and therefore the network IO had not such a huge impact. For the K-Means‖ approach we can again clearly see the enhancement in the initial cluster mean selection, resulting in fewer iterations afterwards and therefore being the fastest attempt with only 824 seconds runtime.

The *KDDCup1999* dataset has been used by B. Bahmani et al. to test their approach on a larger set of observations and their results are presented in table 7. One can clearly see, that K-Means‖ outperforms K-Means in terms of runtime and costs, as well as Partition, which was the slowest at all, in terms of runtime. As we have already seen in the sequential implementations, the oversampling factor influences the runtime in a negative way, whereas it directly improves results in terms of costs. By setting an appropriate oversampling factor $I = 10k$, the authors showed that K-Means‖ now is not only faster than K-Means and Partition, it is also the best approach regarding the costs. The sequential K-Means‖ has an expected higher runtime when compared to the MapReduce version: 700 versus 69 minutes for $k = 500$. Due to its long runtime, we did not test it for $k = 1000$ clusters. As already stated in section 4.1, one should be critical when comparing the costs as the datasets could have been differently converted. We also tried the sequential K-Means++ approach, but had to cancel it after 12 hour runtime and not even leaving the initial cluster mean selection phase, which did only take roughly 20 minutes for our K-Means‖ having $k = 500$ clusters.

5 Conclusions

In this paper we have seen an evolution of the basic K-Means algorithm to be at first applied on MapReduce, enhanced its initial cluster selection phase, thereby improved the accuracy but make it hard to parallelize again and at least found a useful trade-off to have, on the one side, a smarter initial cluster mean selection as well as on the other side being able to run in parallel. We implemented all the different approaches and could proof that, at least on the tested datasets, indeed K-Means‖ is on par in terms of accuracy with K-Means++ as predicted by B. Bahmani et al., whereas it nearly always outperforms K-Means++ in terms of runtime due to the faster initialization phase. But there where some more outcomes: the basic K-Means algorithm is surprisingly stable regarding the standard deviation of the accuracy, at least on the *SPAM* dataset, when having only a few clusters, in contrast to all other approaches (see table 2). This could be due to the law of large numbers or the characteristics of the *SPAM* dataset.

While testing the K-Means‖ approach we found an interesting "bug" in our first implementation: we missed to perform the last basic K-Means phase and therefore directly have taken the reclustered means as final result, leading to an extremely fast runtime even for the sequential experiment on the *KDDCup1999* dataset. Here for $k = 500$ clusters only 23 minutes (48 minutes for $k = 1000$) where needed - and so outperforming the huge cluster with a single laptop - although the overall costs did not increase that much ($7.3 \cdot 10^{10}$ versus $3.6 \cdot 10^{10}$ on the correct run keeping in mind the overall variation). The same holds for the *SPAM* dataset with parameters $0.5k, r = 5$ and $k = 100$ clusters: 1.2 seconds runtime having median costs of $27 * 10^5$ for the "buggy" implementation versus 18.4 seconds runtime having median costs of $23 * 10^5$ for the corrected version. When runtime is a primary concern, this would be an interesting point to investigate further.

When talking about the runtime performance is seems that the MapReduce framework might be prone to the highly iteration based K-Means algorithms, i.e. after each iteration the results from the previous iteration must be distributed on all nodes and the next iteration job has to be set up. This was something we noticed especially on small datasets like *SPAM*: the parallelized basic K-Means approach takes 2651 seconds (see table 6), whereas the sequential one only needs 6.5 seconds (median) for $k = 50$ clusters (see table 5). Even when comparing the larger *KDDCup1999* dataset on $k = 500$ clusters, it takes 1968 nodes, i.e. roughly 1968 times the resources of our test machine - see section 4, to get a result 10 times faster than on the sequential implementation: 69 minutes runtime on the cluster versus 700 minutes runtime on the single machine using only one CPU - see table 7. This is also a point we have already addressed in section 4.1: although B. Bahmani et al. used a bigger dataset - or as they stated "massive", it is not large enough to prove the real advantage of using MapReduce as the parallelizing method of choice.

Literaturverzeichnis

[1] Berkhin P. A Survey of Clustering Data Mining Techniques. Grouping Multidimensional Data. 2006;p. 25–71.

[2] Kerdprasop N, Kerdprasop K. Knowledge Induction from Medical Databases with Higher-order Programming. WSEAS Trans Info Sci and App. 2009 Oct;6(10):1719–1728.

[3] Othman F, Abdullah R, Rashid NA, Salam RA. Parallel K-means Clustering Algorithm on DNA Dataset. In: Proceedings of the 5th International Conference on Parallel and Distributed Computing: Applications and Technologies. PDCAT'04. Berlin, Heidelberg: Springer-Verlag; 2004. p. 248–251.

[4] Ye Z, Mohamadian H, Pang SS, Iyengar S. Contrast Enhancement and Clustering Segmentation of Gray Level Images with Quantitative Information Evaluation. WSEAS Trans Info Sci and App. 2008 Feb;5(2):181–188.

[5] Dhillon IS, Modha DS. A Data-Clustering Algorithm on Distributed Memory Multiprocessors. In: Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD. London, UK, UK: Springer-Verlag; 2000. p. 245–260.

[6] Mackey P, Feo J, Wong PC, Chen Y. A Highly Parallel Implementation of K-means for Multithreaded Architecture. In: Proceedings of the 19th High Performance Computing Symposia. HPC '11. San Diego, CA, USA: Society for Computer Simulation International; 2011. p. 33–39.

[7] Marzouk YM, Ghoniem AF. K-means Clustering for Optimal Partitioning and Dynamic Load Balancing of Parallel Hierarchical N-body Simulations. J Comput Phys. 2005 Aug;207(2):493–528.

[8] Esteves RM, Hacker T, Rong C. Cluster Analysis for the Cloud: Parallel Competitive Fitness and Parallel K-means++ for Large Dataset Analysis. In: Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom). CLOUDCOM '12. Washington, DC, USA: IEEE Computer Society; 2012. p. 177–184.

[9] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. In: Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6. OSDI'04. Berkeley, CA, USA: USENIX Association; 2004. p. 10–10.

[10] Hadoop: Open source implementation of MapReduce;. [Online; accessed 20.01.2014]. http://lucene.apache.org/hadoop/.

[11] Apache Mahout: machine learning library;. [Online; accessed 20.01.2014]. https://mahout.apache.org/.

[12] Zhao W, Ma H, He Q. Parallel K-Means Clustering Based on MapReduce. In: Proceedings of the 1st International Conference on Cloud Computing. CloudCom '09. Berlin, Heidelberg: Springer-Verlag; 2009. p. 674–679.

[13] Arthur D, Vassilvitskii S. K-means++: The Advantages of Careful Seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics; 2007. p. 1027–1035.

[14] Lloyd S. Least squares quantization in PCM. Information Theory, IEEE Transactions on. 1982 Mar;28(2):129–137.

[15] Forgy E. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics. 1965;21:768–780.

[16] MacQueen J. Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics. Berkeley, Calif.: University of California Press; 1967. p. 281–297.

[17] Loohach R, Garg K. Article: Effect of Distance Functions on Simple K-means Clustering Algorithm. International Journal of Computer Applications. 2012 July;49(6):7–9. Published by Foundation of Computer Science, New York, USA.

[18] Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY. An efficient k-means clustering algorithm: analysis and implementation. Pattern Analysis and Machine Intelligence, IEEE Transactions on. 2002;24(7):881–892.

[19] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard, Version 3.0; 2012. [Online; accessed 20.01.2014]. http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf.

[20] Ghemawat S, Gobioff H, Leung ST. The Google File System. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. SOSP '03. New York, NY, USA: ACM; 2003. p. 29–43.

[21] DeWitt S. MapReduce: A major step backwards; 2009. [Online; accessed 20.01.2014]. http://craig-henderson.blogspot.de/2009/11/dewitt-and-stonebrakers-mapreduce-major.html.

[22] More patent nonsense - Google MapReduce; 2010. [Online; accessed 20.01.2014]. http://www.dbms2.com/2010/02/11/google-mapreduce-patent.

[23] Amazon MapRecude Cluster; 2014. [Online; accessed 20.01.2014]. http://aws.amazon.com/de/elasticmapreduce.

[24] Intel MapRecude Cluster; 2014. [Online; accessed 20.01.2014]. http://www.intel.com/content/www/us/en/big-data/big-data-analytics-turning-big-data-into-intelligence-cmpg.html.

[25] Shvachko K, Kuang H, Radia S, Chansler R. The Hadoop Distributed File System. In: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). MSST '10. Washington, DC, USA: IEEE Computer Society; 2010. p. 1–10.

[26] Ackermann MR, Märtens M, Raupach C, Swierkot K, Lammersen C, Sohler C. StreamKM++: A Clustering Algorithm for Data Streams. J Exp Algorithmics. 2012 May;17:2.4:2.1–2.4:2.30.

[27] Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S. Scalable K-means++. Proc VLDB Endow. 2012 Mar;5(7):622–633.

[28] Ailon N, Jaiswal R, Monteleoni C. Streaming k-means approximation. In: NIPS; 2009. p. 10–18.

[29] UCI Machine Learning Repository: Spambase Data Set;. [Online; accessed 20.01.2014]. http://archive.ics.uci.edu/ml/datasets/Spambase.

[30] UCI Machine Learning Repository: UCI KDD Archive;. [Online; accessed 20.01.2014]. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[31] Terry Brugger: KDD Cup '99 dataset (Network Intrusion) considered harmful;. [Online; accessed 20.01.2014]. http://www.kdnuggets.com/news/2007/n18/4i.html.

Efficient Similarity Joins using MapReduce

Luzia Beisiegel

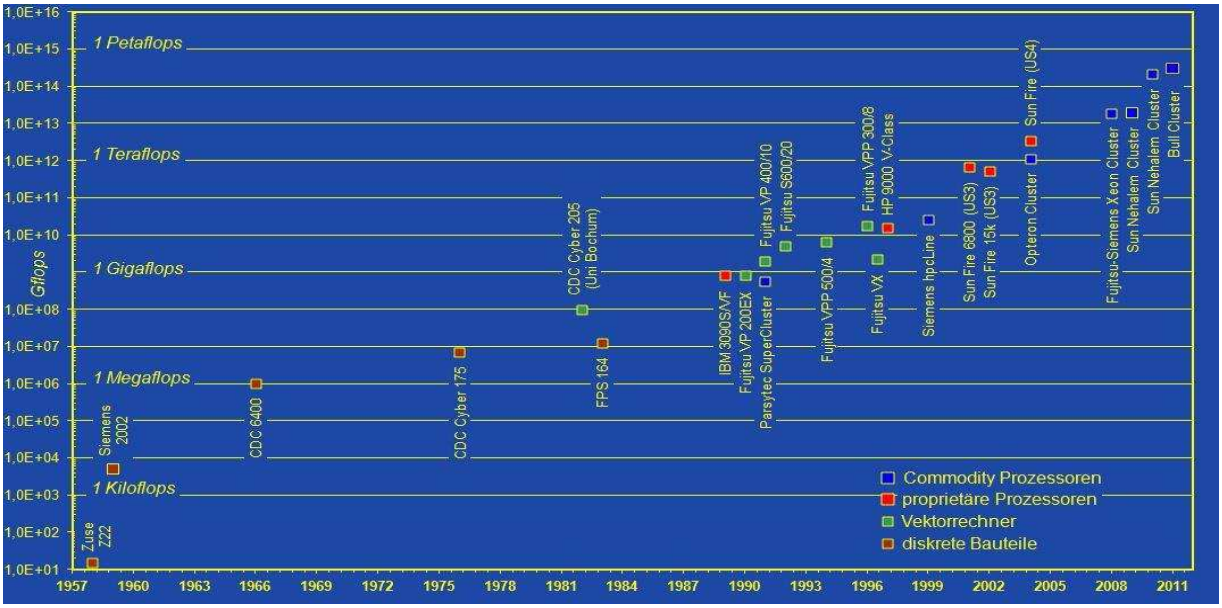
Zusammenfassung

This seminar paper presents and discusses 'Similarity Joins' focusing upon the 'k Nearest Neighbor Joins' by using MapReduce. By constructing a bounding area in order to derive a pruning rule for distance filtering, and by using the MapReduce framework, it is possible to make the expensive k Nearest Neighbor Joins algorithm more efficient, such that it is possible to handle huge amounts of data. This is important with respect to data mining in general as well as for classification problems in (medical) image processing, since the amount of data to be processed is increasing very fast.

Keywords: Similarity Joins, kNN Joins, MapReduce, efficiency, data processing

1 Introduction

As the amount of data to be processed is increasing very fast over the whole world, there is a steady demand for more efficient data processing methods. Fig. 1 gives us an idea about how fast the processing methods have been improved. In the year 1954, at the University RWTH Aachen, one was able to process only about 1 kiloflops of data. In the year 2011 one was already able to process about 1 petaflops. Hence, over a period of 57 years, the amount of data which one was able to process at the RWTH increased by a factor of about 10^{12}. This development is shown in Fig. 1 below.



Source: www.rz.rwth-achen.de/aw/cms/rz/Themen/hochleistungsrechnen/rechnersysteme/~oef/hpc_historie_im_rz/?lang=de

Measurements:

Kiloflops: 10^3 (1.000)
Megaflops : 10^6 (1.000.000)
Gigaflops : 10^9 (1.000.000.000)
Teraflops : 10^{12} (1.000.000.000.000)
Petaflops : 10^{15} (1.000.000.000.000.000)

Abb. 2.1: An example of how fast processing methods are improved

The case is similar in medical image processing. Within this context, Elsayed [2], for example, wrote in his PhD-Thesis that *'the quantity of medical image data that is being generated is fast becoming*

overwhelming, while at the same time there is a requirement for ever more sophisticated tools to assist in the analysis of this imagery'. Alongside this argument it was pointed out by Aji [3], that a typical hole-slide image, for examples, contains 1 Mio. x 1 Mio. x 1 Mio. number of pixels.

Concerning the problem of handling this overwhelming amount of data in general, Lu et al. [1] made an important contribution, by developing the *PBJ* (a partition based join method) and especially the *PGBJ* (a partition based join method with a grouping algorithm on top), which are the result by performing the *k Nearest Neighbor Joins* (kNN Joins) using MapReduce. Their methods can be applied for example for pixel classification in (medical) image processing.

Let consider a Magnet Resonance Image (MRI) for which we want to perform (pixel) classification. One possibility is to use the *k Nearest Neighbor Classification* method (see [2, 5, 6]) based on the kNN algorithm. The Idea is to assign each pixel, represented by a property vector, to a class label by a majority vote of its neighbors from a training data set, objects for which the class membership is already known. In the example below, Fig. 2, we have to classify the green point. We can see, that in the case of $k = 3$, the majority vote of the green point's neighbors holds for the red class label, therefore the green point is assigned to this class. In the case that $k = 9$, the majority vote of the green point's neighbors holds for the blue class label, therefore it is assigned to the class with the blue class label. Considering this example, we can see, that the result of this classification method is sensitive to the choice of k (see also [2, 5, 6]).

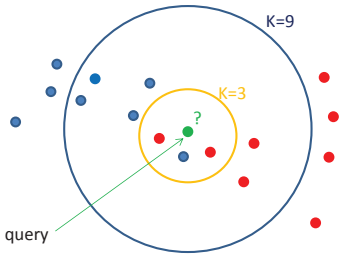


Abb. 2.2: Example for kNN classification with $k = 3$ or $k = 9$

Taking this example as a starting point, this seminar paper is structured as follows: In the second chapter I'm going to give a short introduction to MapReduce and *Similarity Joins* using MapReduce. In the third chapter I will show which problems arise when performing kNN Joins using MapReduce which I will solve by introducing the quite complex but efficient methods *PBJ* and *PGBJ* proposed by Lu et al. [1]. Here, according to Lu et al. [1], I will construct a distance bounding area in order to derive a pruning rule for reducing the size of the data set. In the fourth chapter I will show several experimental evaluations according to Lu et al. [1], in order to see, for example what is the impact of dimensionality on the running time. In the fifth chapter I will give an overview of how the *PGBJ* can be applied in medical image processing. In the sixth chapter I discuss two more problems. Firstly the problem with the right choice of a distance function when applying the *PBJ* or the *PGBJ* method for (pixel) classification and secondly the empty space phenomenon when dimensionality is increasing. Finally, chapter seven concludes this seminar paper by pointing out the most important results and given an outline of my future work within my studies of mathematics.

2 Introduction to MapReduce and Similarity Joins using MapReduce

Similarity Joins are a popular tool in data mining. In this work I'm only interested in two, the *Theta Joins* with a similarity join condition (see Okcan and Riedewald [4]) and the *k Nearest Neighbor Joins* (see [1], [7]). Within the context of my interest, the similarity is measured by a distance function d. Thus the more similar objects are, the smaller is the distance d between the two objects considered.

Lately, after the MapReduce framework was introduced in 2004, there has been a considerable effort on researches to modify the existing methods in order that they can be performed by using the popular MapReduce framework (see [1, 7, 8]), such as the following two Similarity Joins: The *Theta Join* with a similarity join condition using MapReduce introduced by Okcan and Riedewald [4]) and the kNN Join by Lu et al. [1]. Before considering these two *Similarity Joins*, I would like to give a short introduction to MapReduce.

2.1 MapReduce

The MapReduce framework has been introduced in the year 2004 by two Google Fellows J. Dean and S. Ghemawat and published under the title *MapReduce: Simplified Data Processing on Large Clusters.* The main idea is to parallelize automatically the computation across large-scale clusters of machines or across multiple cores of the same machine [9]. It is a simple and revolutionary concept for data mining. Moreover MapReduce can be implemented for example in the free open-source Hadoop platform http://hadoop.apache.org which makes this framework very popular. Within this context Dean and Ghemawat [9], for example, wrote that over the four years before 2008 '*ten thousand distinct MapReduce programs have been implemented internally at Google ... and an average of one hundred thousand MapReduce jobs are executed on Google's cluster every day, processing a total of more than twenty petabytes of data per day.*'.

In more detail, MapReduce works as follows: Users can specify the computation in terms of a *map*- and a *reduce*-function, whereas the *map*- function is defined as

$$map(k1, v1) \rightarrow list(k2, v2) \tag{2.1}$$

and the *reduce*-function as

$$reduce(k2, list(v2)) \rightarrow list(v2) \tag{2.2}$$

$\forall k1, k2 \in K$ and $\forall v1, v2 \in V$ whereas, K is the set of all keys and V of all values [9] .

Then MapReduce run-time system groups and sorts all the intermediate values assigned with the same key and maps them to the *reduce*-function, which accepts an intermediate key and its corresponding value. The final result is typically a list of values [1, 9]. This work flow is shown in Fig. 3 below.

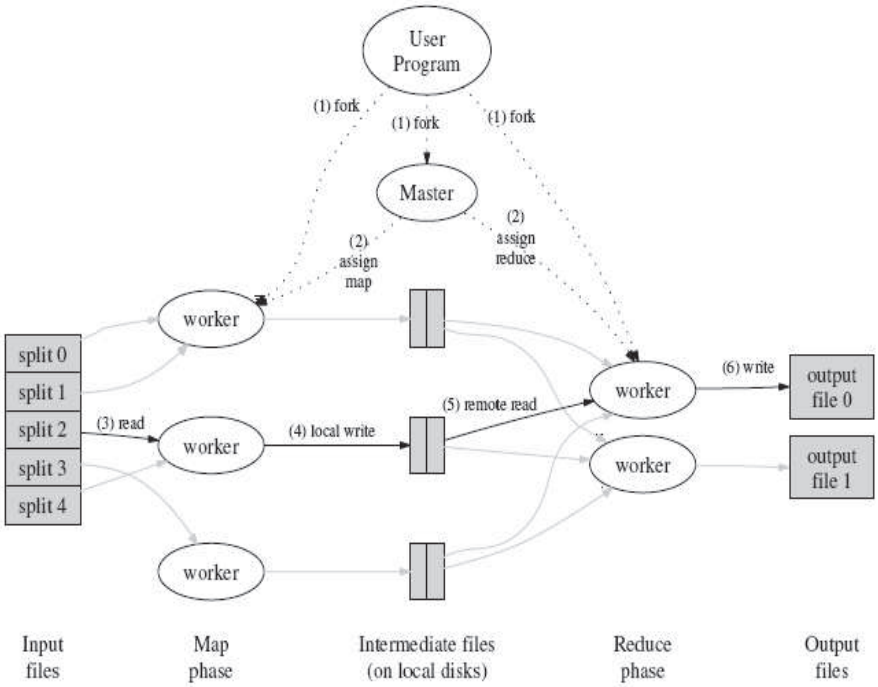


Abb. 2.3: Execution overview ([9], fig. 1)

After having been introduced to the MapReduce framework, I will show in short, how Okcan and Riedewald used the MapReduce framework to perform *Theta Joins.*

2.2 How to perform Theta Joins using MapReduce

The basic idea of *Theta Joins* is very simple: compare every object with every other object. In *Theta Joins* users can define a join condition – in the context of *Similarity Joins* the join condition will become a similarity condition defined via a distance function. Then a Cartesian product that's filtered by this condition, compares values from data set S with values from data set R (it is also possible, that $R = S$) and according to this condition the objects are joined (see [4]).

In [4] the authors deliver the fundamental work to perform any *Theta Joins* using MapReduce. There is no need of modifications to the MapReduce Framework, there is required only one MapReduce job and it is required only minimal statistics, the input cardinality [4]. The join between two data sets R and S is modeled by a join matrix M. Each matrix entry $M(i, j)$ - row i and column j of the matrix M - is set to true, if for the i-th entry from data set R and for the j-th entry from S the join condition is satisfied, and false otherwise [4]. Each matrix cells with value $M(i, j) = true$ are mapped to the reducers. In order to avoid duplications or replications, all entries with the same join attribute value are mapped to the same reducer. In order to reduce total job completion time, the authors Okcan and Riedewald propose 1- and M-Bucket-Theta algorithm [4]. For a simplified graphical representation of this work flow I would like to refer to Fig. 4.

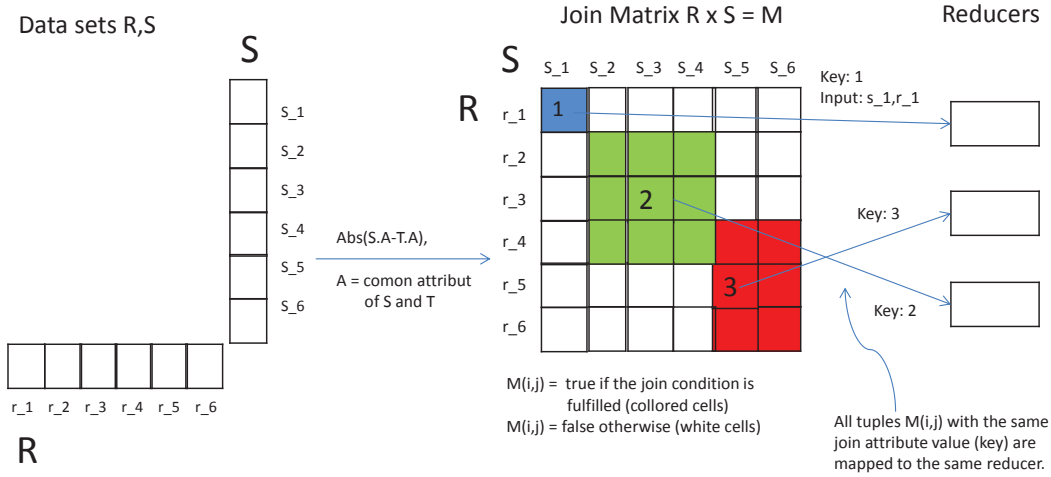


Abb. 2.4: Execution overview of *Similarity (Theta) Joins* using MapReduce (compare to [4], fig. 3, 4)

With the similarity condition for this *Similarity (Theta) Joins*, we construct a distance bound in which all objects of interest are laying. This can be used for example for finding all Facebook accounts similar - measured by a distance function - to a referential account, in order to see if there is may be someone with more than one account.

Considering the classification problem mentioned in the introduction, this kind of *Similarity Joins*, without any modification is not adapted, because we consider all objects laying in a distance bounding of an object to classify, i.e. there can be laying only one object or about 50 objects. Having only very few, may be only one nearest neighbor, we have the problem, that our kNN classification algorithm is becoming very sensitive to outliers[5, 2]. Results of the classification problems are logically better, if we consider a steady number k of objects as neighbors - k should be far smaller than the minimum number of objects in each class of the training data set, but at the same time, it should not be too small. This is the reason way we have to get introduced to another *Similarity Joins*, the *k Nearest Neighbor Joins*. Lu et al. introduced in [1] a method how to make kNN Joins more efficient by constructing a pruning rule and by using MapReduce (see also [7]).

2.3 How to perform *k*NN Joins using MapReduce

The kNN Joins is a combination of the kNN query and a join operation, i.e. its operation associates each data object in one data set with it's k nearest neighbors from the same or a different data set. Later we will see, that the main task of each mapper and reducer are the following: cluster objects into groups (mappers) and perform the kNN Joins on each group of objects separately (reducers) [1].

Definition 1 *(kNN Joins: $R \propto S$, see Böhm and Krebs [10]) $R \propto S$ is the smallest subset of RxS that contains for each point of R at least k points of S and for which the following condition holds:*

$$\forall(r,s) \in R \propto S, \forall(r,s') \in S - R \propto S; d(r,s) \leq d(r,s'), \tag{2.3}$$

for a distance function d and data sets R, S.

In general the kNN Joins is an expensive operation. Having two data sets S, R (it is also possible, that $S = R$), the naive implementation of kNN Joins requires scanning S once for each object in R leading to a complexity of $O(|R|x|S|)$, [1]. Therefore, given the fast increasing volume of data to be processed, it makes sense to perform it using MapReduce and to construct pruning rules, in order to reduce both, the shuffling and the computational, by not having to scan S once for each object in R anymore.

The basic idea to implement the kNN Joins in MapReduce is similar to the *Hash Join* algorithm [1, 11]. The data sets are loaded into the main memory of the mappers whereas each object from R and S is assigned with a key, then all the objects with the same key are distributed to the same reducer in the shuffling process. Finally, the reducers perform the kNN Joins over the objects that have been shuffled to it [1].

Lu et al. describe in their paper [1] a kind of evolution. They start with the naive basic idea and the *H-BRJ* method, then they arrive at the PBJ method till they arrive at their finial and the most efficient method, the $PBGJ$.

First, the *Basic Idea* and the *H-BRJ* method. In the *Basic Idea* we split data set R into N disjoint subsets $R_i, i = 1, ..., N$ by the *map*()-function. Each R_i is then distributed to one reducer and the entire set S has to be sent to each reducer to be joined (by the *reduce*()-function) with R_i, for $i = 1, ..., N$. Obviously, sending the entire set S to each reducer and join the entire S with each $R_i, i = 1, ..., N$ produces a lot of replications in the shuffling phase, see Fig. 5 (a). In this method only one MapReduce job is required and the shuffling costs are $|R| + N|S|$. If the data set S is to large, it may go beyond the capability of the reducer, then the alternative method can be applied, the *H-BRJ*. Here both, R and S are divided into $\sqrt{N}$ disjoint subsets $R_i, S_j, i, j = 1, ..., \sqrt{N}$. Then, each $R_i, i = 1, ..., \sqrt{N}$ is sent to exactly one of the N reducers and each S_j is sent to each of the $\sqrt{N}$ reducers for all $j = 1, ..., \sqrt{N}$. In this method there are required a second MapReduce job to combine the $R_i \propto S_j, \forall j = 1, ..., \sqrt{N}$. The shuffling costs are $\sqrt{N}(|R| + |S|) + \sum_i \sum_j |R_i \propto S_j|^2$ [1]. Obviously there are also in this methods a lot of replications which lead to inefficiency, since every $S_j, j = 1, ..., \sqrt{N}$ is sent to each reducer, see Fig. 5(b).

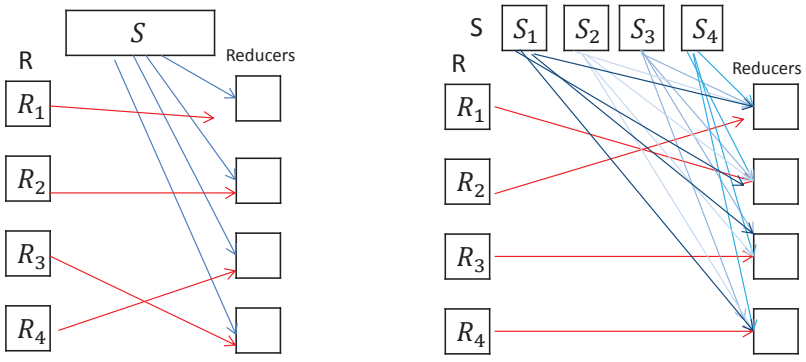


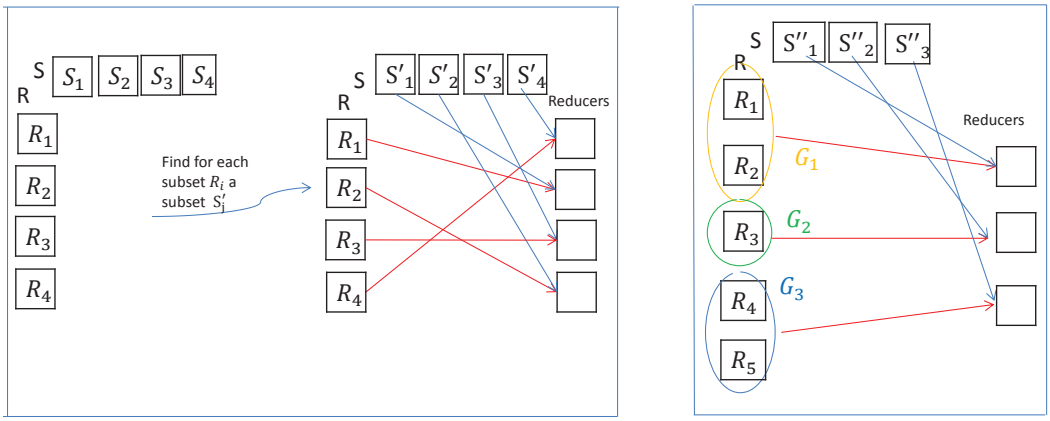
Abb. 2.5: The *Basic Idea* (a) and the *H-BRJ* method (b) – same elements of set S are sent to several different reducers in each method, which provokes replication.

The idea is now, to partition R into N disjoint subset $R_i, i = 1, ..., N$ and to find for each subset $R_i, i = 1, ..., N$ the right subset of S denoted by S'_i, such that the join has to be performed only between R_i and $S'_i, i = 1, ..., N$. This idea leads to the PBJ method.

Second,the PBJ method. After having partitioned R into N disjoint subsets $R_i, i = 1, ..., N$ and found for each R_i, its corresponding $S'_i \subset S$, such that $R_i \propto S = R_i \propto S'_i = \bigcup_{1 \leq i \leq N}$, each R_i can be sent together with its corresponding S'_i to one reducer, where the join will be performed, for each $i = 1, ..., N$, see Fig. 6 (a), a very simplified illustration of this work-flow. Obviously in this way we are able to reduce

the size of S and to reduce the number of replications on objects in S. In this method we need two MapReduce jobs, as we will see later.

Third, the $PGBJ$ method. This method is quite similar to the PBJ method. The only difference is, that, on top of the PBJ, we apply a grouping algorithm on data set R, see Fig. 6 (b). Later we will see how this works and that in this way, we are able to minimize the number of replication, which leads to lower shuffling and computational costs.



(a) (b)

Abb. 2.6: The PBJ (a) and the $PGBJ$ (b) method

Remark 1 .

- *Obviously Fig. 5 (a), (b) and 6 (a), (b) are a very simplified illustration of the work-flow.*
- *It is important to be aware of the fact, that it is possible, that $S'_i = S_i$ but usually this is not the case.*

As the aim is to get an efficient method, I'm focusing only upon last two methods. Furthermore I focus firstly upon the PBJ and secondly upon the $PGBJ$, since $PGNJ$ is just a modification of PBJ.

3 The *PBJ* and *PGBJ* Method

PBJ and $PGBJ$ are two resulting methods of performing kNN Joins using MapReduce proposed by Lu et al. [1]. In these two methods Lu et al. [1] solved the following four problems arising, when performing kNN Joins using MapReduce:

1. How to find a good partition of R?
2. How to reduce the size of S?
3. How to join each R_i only with its corresponding S'_i?
4. How to minimize the number of replications on S?

The first Question can be answered by applying the Voronoi Diagram partitioning, the second and the third by constructing a distance bounding area and finally, the last question can be answered by applying on top a grouping algorithm, which leads from the PBJ to the $PGBJ$ method.

Focusing on the PBJ, when performing kNN Joins using MapReduce, the tree main steps are: The pre-processing step, the First MapReduce job and the second MapReduce job. In the pre-processing step we have to find pivots on R and to create a Voronoi Diagram. In the First MapReduce job we have to partition on R and S and calculate all the distances. In the second MapReduce job we have to find S'_i for each R_i and to perform the kNN Joins between S'_i and R_i. Befor discussing each of the three steps in more detail separately, I would like to refer to Fig. 7 below for the graphical overview of this work-flow.

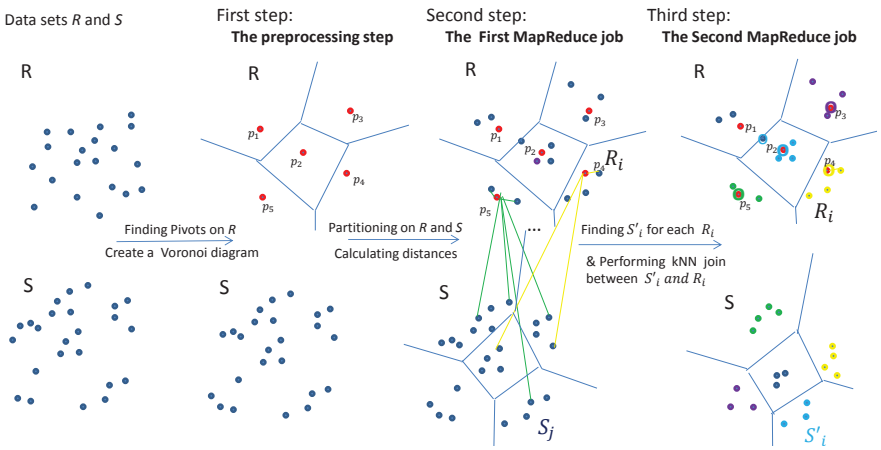


Abb. 2.7: The work-flow of performing kNN Joins using MapReduce focusing upon the PBJ

3.1 The Pre-processing Step

In order to perform the Voronoi Diagram partitioning, its formal definition is required.

Definition 2 *(Voronoi Diagram based Partitioning, see Lu et al. [1]) Given a set O we select M objects (which may not belong to O) as pivots and then we split objects of O into M disjoint partitions where each object is assigned to the partition with its closest pivot.*

Later we will require the following notions:

P_o^O *is the partition of set O that corresponds to the pivot p_o*

$U(P_i^O)$ *and* $L(P_i^O)$ *are the maximum and minimum distance from pivot p_i to the objects of P_i^O respectively*

$P_i^R = R_i$ *and* $P_i^S = S_i$ *for data sets R and S!*

But before performing the Voronoi Diagramm partitioning, we have to select a set of pivot from data set R. There exist several strategies for selecting a set of pivot as follows (compare to Lu et al. [1]):

1. Random selection:

   Firstly, T random sets of objects are selected from R. Then we have to compute the total sum distance between every $t_i, t_j \in T$. Finally, the objects from the set with the maximum total sum of distances are selected as pivots for data partitioning.

2. k-means selection:

   First we have to sample on the set R, then apply a traditional k-means clustering method on the sample. Finally, the center point of each cluster is chosen as pivot.

3. Farthest selection (iteratively):

   Firstly we have to select randomly an object for the first pivot p_1. Then the object with the largest distance to p_1 is selected as the second pivot p_2. Finally, in the i-th iteration, the object that maximizes the sum of its distance to the first $i-1$ pivots is chosen as the i^{th} pivot.

The Voronoi Diagram based partitioning applied on set R is required in order to perform the second step, the first MapReduce job.

3.2 The First MapReduce Job

Given the set of pivots selected in the pre-processing step, we launch a MapReduce job for performing data partitioning and collecting some statistics for each partition. These statistics are then selected in two summary tables T_R and T_S for data set R and S respectively containing the following information:

T_R	T_S
Partition id	Partition id
Number of objects	Number of objects
$L(R_i)$ and $U(R_i)$	$L(S_j)$ and $U(S_j)$
	$d(p_i, kNN(p_i, S_j))$

These two memory tables will be used to guide how to generate S_i' for each $R_i, i = 1, ..., N$ and to speed up the computation of R_i joined S_i'.

3.3 The Second MapReduce Job

This third step is the most complicate. Here we finally construct the already several times mentioned distance bounding area to derive a pruning rule, which helps us to reduce the size of S. In this step, we use the two summary tables T_R and T_S received in the previous step in order to fasten the algorithm. Because of its complexity, I would like to discuss the second MapReduce job in the following three steps:

(i) How to construct a distance bound of kNN Joins based on the partition of R in order to reduce the size of S_i' for all $i = 1, ..., N$?

(ii) How to find S_i' for each $R_i, i = 1, ..., N$?

(iii) How to perform the kNN Joins between R_i and S_i' for $i = 1, ..., N$?

(i) Construction of Distance Bounding Area. .

In order to construct the distance bounding area, we need to be in a metric space (see Definition 3 below) and the concepts of the hyperplane, on which the Theorem 1 will be based as well as the concept of the range query (also known as range selection), see Definition 4 below.

Definition 3 (Hyperplane) .
Let (M,d) be a metric space with distance metric d, i.e. $M \neq \emptyset$, $M \subset \mathbb{K}^n$, $\mathbb{K}^n = \mathbb{R}^n$, $\mathbb{C}^n$, *and d such that the following holds:*

(1) $d(x, y) = 0 \Rightarrow x = y \ \forall x, y \in M$,

(2) $d(x, y) = d(y, x) \ \forall x, y \in M$,

(3) $d(x, z) \leq d(x, y) + d(y, z) \ \forall x, y, z \in M$.

Then a Hyperplane denoted by HP *is a subspace of* M *with dimension* $n - 1$.

Theorem 1 *(A generalized version of Theorem 1 by Lu et al. [1])*
Given two pivots p_i, p_j, *let* $HP(p_i, p_j)$ *be the generalized hyperplane, where any object o lying on* $HP(p_i, p_j)$ *has the equal distance to* p_i *and* p_j. $\forall o \in P_j^O$, *the distance of o to* $HP(p_i, p_j)$ *is*

$$d(o, HP(p_i, p_j)) = inf_{x \in HP(p_i, p_j)} d(o, x). \tag{2.4}$$

Proof See standard books about functional analysis.

□

Definition 4 (Range query) *(compare to Lu et al. [1])*
Given a data set O, an object q, and a distance threshold θ, *range query of q from O is to find all objects (denoted as* $\tilde{O}$*) of O, such that* $\forall o \in \tilde{O}$, $d(q, o) \leq \theta$.

For a graphical understanding of the concept of the hyperplane and Definition 4, I would like to refer to Fig. 8 below. In Fig.8 (a) we can see the concept of the hyperplane for dividing two partition, i. e. all objects lying on the left side of hyperplane $H(p_i, p_j)$ belong to partition P_i^O and all objects lying on the right side of the hyperplane $H(p_i, p_j)$ belong to partition P_j^O. In Fig. 8. (b) is illustrated the concept of the range query. All objects lying in the dark circular disk around the pivot p_t are of interest, all the other objects are discarded.

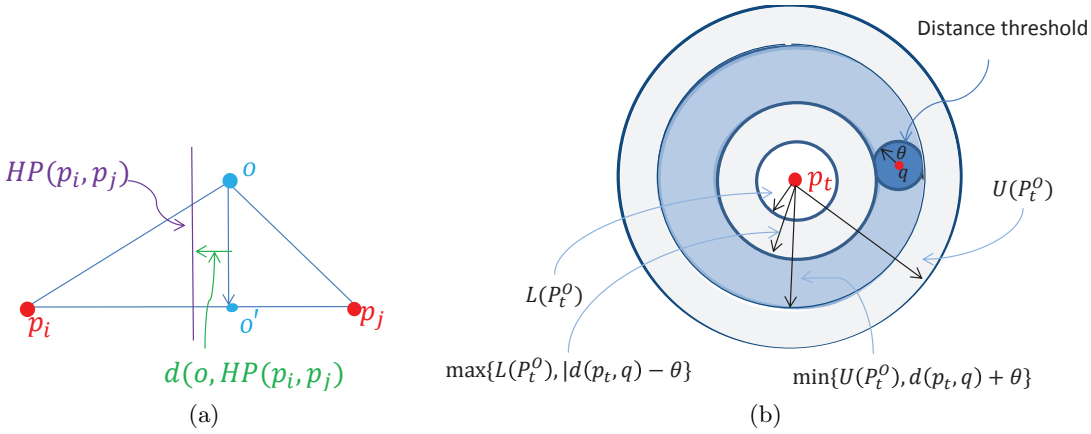


Abb. 2.8: (a) Hyperplane for dividing partitions. (b) Threshold θ for range query (compare to Lu et al. [1])

Remark 2 *It is important to assume that we are in a metric space assigned with a distance metric d for which the triangle inequality holds, otherwise this concept doesn't work. To proof each of the following theorems or corollaries, the triangle inequality is the basic tool.*

Based on the concept of the hyperplane and Definition 4 we can now derive the following corollary:

Corollary 1 *Given a partition P_i^O and P_q^O such that $P_j^O \neq P_q^O$. If the condition:*

$$d(q, H(p_q, p_i)) > \theta \tag{2.5}$$

then $\forall o \in P_i^O, d(q, o) > \theta$.

Proof See Lu et al. [1], Corollary 1.

□

For a graphical view of Corollary 1, I would like to refer to the following figure below. In Fig.9 (a) we are in the case, that $d(o, HP, (p_i, p_j) > \theta$ and therefore there are no objects of interest laying in partitions O_q or O_i. In (b) we have the case, that $d(o, HP, (p_i, p_j) \leq \theta$, therefore there is still one object laying in partition O_q which we still have to check, in order to find the kNN of object o.

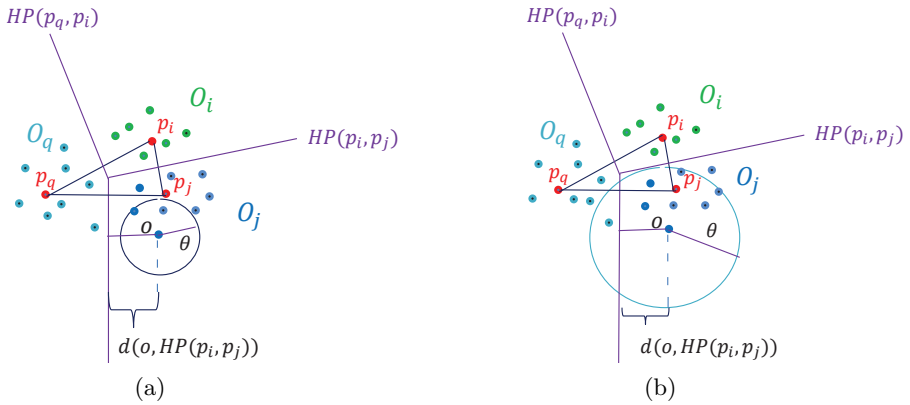


Abb. 2.9: (a) $d(o, HP(p_i, p_j)) \leq \theta$. (b) $d(o, HP(p_i, p_j)) > \theta$

In summery, this corollary says, that if condition (5) holds, then we can discard all objects of P_i^O, otherwise we check partial objects of P_i^O based on the following theorem:

Theorem 2 *Given a partition P_i^O, $\forall o \in P_i^O$ the necessary condition that $d(q,o) \leq \theta$ is:*

$$max\left\{L(P_i^O), d(p_i,q) - \theta\right\} \leq d(p_i,o) \leq min\left\{L(P_i^O), d(p_i,q) - \theta\right\}. \tag{2.6}$$

Proof See Lu et al. [1], Theorem 2.

□

This bounding concept can be applied to bound the kNN $\in S$ for all objects $r \in R_i$, $\forall i = 1, ..., N$, i.e. finding S_i' for R_i $\forall i = 1, ..., N$.

Theorem 3 *Given a partition $R_i \subset R$, an object $s \in S_j \subset S$, the upper bound distance from s to $\forall r \in R_i$, denoted as $ub(s, R_i)$, is:*

$$ub(s,R_i) = U(R_i) + d(p_i,p_j) + d(p_j,s). \tag{2.7}$$

Proof See Lu et al. [1], Theorem 3.

□

Based on this Theorem and the considerations in the Definition 4, we now can derive a bound denoted as θ_i that corresponds to $R_i, i = 1, ..., N$ of the kNN distance for all objects in R_i as follows:

$$\theta_i = max_{\forall s \in kNN(R_i,S)} |ub(s,R_i)|, i = 1, ..., N. \tag{2.8}$$

i.e. $\forall r \in R_i$ the distance from r to any object of kNN(r,S) is less or equal to θ_i. This holds for each $i = 1, ..., N$. In order to find this θ_i in equation (8), Lu et al. [1] propose an algorithm (see Appendix, Algorithm 1). Firstly we have to create a priority queue PQ with size k in order to sort the kNN$\in S$ for each object $r \in R$ in an ascending order. Then we have to compute for partition S_j the upper-bound $ub(s, R_i)$ for each $s \in k$NN(p_j, S_j) based on the information given in T_S. Finally, when $ub(s, R_i) >$ PQ.top, we can guarantee, that no remaining objects in kNN(p_j, S_j) helps to refine θ_i, and we can return to the top of the PQ which is taken as θ_i. For a graphical point of view of these steps, I would refer to Fig. 10.

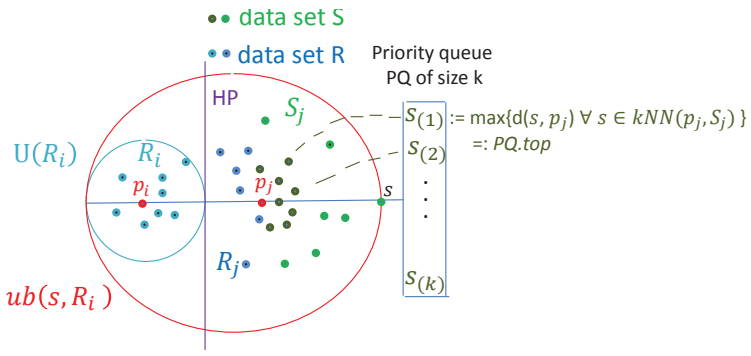


Abb. 2.10: A graphical view of how to find θ_i: First, create priority queue PQ. Second, compute the upper-bound $ub(s, R_i), \forall i = 1, ..., N$. Third, if $ub(s, R_i) > PQ.top$, return to the top of the PQ wich is taken as θ_i.

Similar to the previous Theorem, we can also derive the lower bound distance from an object $s \in S_j$ to any object of R_i as follows:

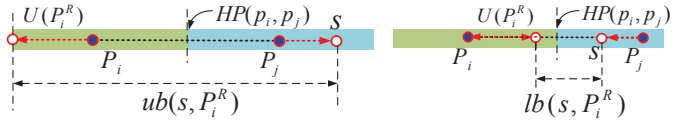
Theorem 4 *Given a partition $R_i \subset R$, an object $s \in S_j \subset S$, the lower bound distance from s to $\forall r \in R_i$, denoted as $lb(s, R_i)$, is:*

$$lb(s,R_i) = max\left\{0, d(p_i,p_j) - U(R_i) - d(p_j,s)\right\} \tag{2.9}$$

Proof See Lu et al. [1], Theorem 4.

□

The bounding area looks now as depicted in Fig.11. We still have the hyperplane between two pivots for dividing its corresponding partitions. The only difference to the concept shown in Fig. 8 (a), (b) is that we have a second data set which we have to bound. Therefore we are not anymore considering the distance between an object o and the hyperplane, but just the lower-bound lb which bounds the elements of set S corresponding to partition R_i, for all $i = 1, ..., N$.



(a) upper bound (b) lower bound

Abb. 2.11: Bounding k nearest neighbors (Lu et al. [1], Fig. 5)

(ii) Finding S'_i for each R_i .

The aim is to find for all objects $r \in R_i$ its kNN$\in S$ leading to our desired S'_i, i.e. S'_i contains only all the kNN of all objects $r \in R_i$, $\forall i = 1, ..., N$. To find our desired S'_i we use the information given in the two summary tables T_R and T_S which we have as a byproduct of the first MapReduce job.

The bounding concept introduced above can now be applied to bound the kNN $\in S$ for all objects $r \in R_i$, $\forall i = 1, ..., N$. i.e. finding S'_i for R_i $\forall i = 1, ..., N$. Considering the concept of Fig. 9 and Fig. 11 above, we can see, that all $s \in S$ for which the inequality $lb(s, R_i) > \theta_i$ is fulfilled, cannot be objects in $k\text{NN}(r, S), \forall r \in R_i$, i.e. s is safe to be pruned. This concept holds for each $i = 1, ..., N$. Based on this concept, the following Theorem can be derived:

Theorem 5 *Given a partition R_i and an object $s \in S$, the necessary condition that s is assigned to S_i is that:*

$$lb(s, R_i) \leq \theta_i \tag{2.10}$$

Proof See Lu et al. [1], Theorem 5.

□

In the case that the number of partitions for R is large, Lu et al. [1] propose the following corollary:

Given a partition R_i and a partition S_j, $\forall s \in S_j$, the necessary condition that s is assigned to S_i is that:

$$d(s, p_j) \geq LB(S_j, R_i) \tag{2.11}$$

where $\text{LB}(\text{S}_j, R_i) = d(p_i, p_j) - U(R_i) - \theta_i$.
Proof See Lu et al. [1], Corollary 2.

□

Hence, S'_i is constructed by all the objects lying in the region $[LB(S_j, R_i), U(Sj)]$.

(iii) Joining S'_i with R_i .

In (ii) we have found for each R_i its corresponding $S_i, i = 1, ..., N$, whereas the partition id defines the input key. Hence, all objects r of partition R_i and all objects s of partition S'_i can be sent to on reducer for $i = 1, ..., N$. As a consequence of the previous considerations, we can perform the join by each reducer as follows:

1. pars R_i and S'_i, such that $S'_i = \bigcap_{l=1}^{M} S'_{i_1} ... S'_{i_M}$.
2. sort $S'_{i_1} ... S'_{i_M}$ in ascending order with respect to $d(p_i, p_{j_l}), l = 1...M$.

3. derive a distance bounding $\theta_{i_l}, l = 1...M$ of kNN for every object $r_{i_l} \in R_i, i = 1, ..., N$.

4. derive a distance bounding θ for all objects of R_i via the third step and by checking with Corollary 1 or Theorem 2 (depending on the fact weather the object $s \in S'_{i_l}, l = 1...M$ is pruned or not).

Finally each of the reducer outputs a list of all kNN for each object $r \in R_i$ $i = 1, ..., N$ of the form $output(r, kNN(r, S))$.

As a summery of the second MapReduce job I would like to refer to Algorithm 3 of Lu et al. [1] in the Appendix and as a summery of the three steps (pre-processing step, first MapReduce job and second MapReduce job) discussed above, I would like to refer to the following figure below from Lu et al [1] which depicts again the work-flow of kNN Joins using MapReduce.

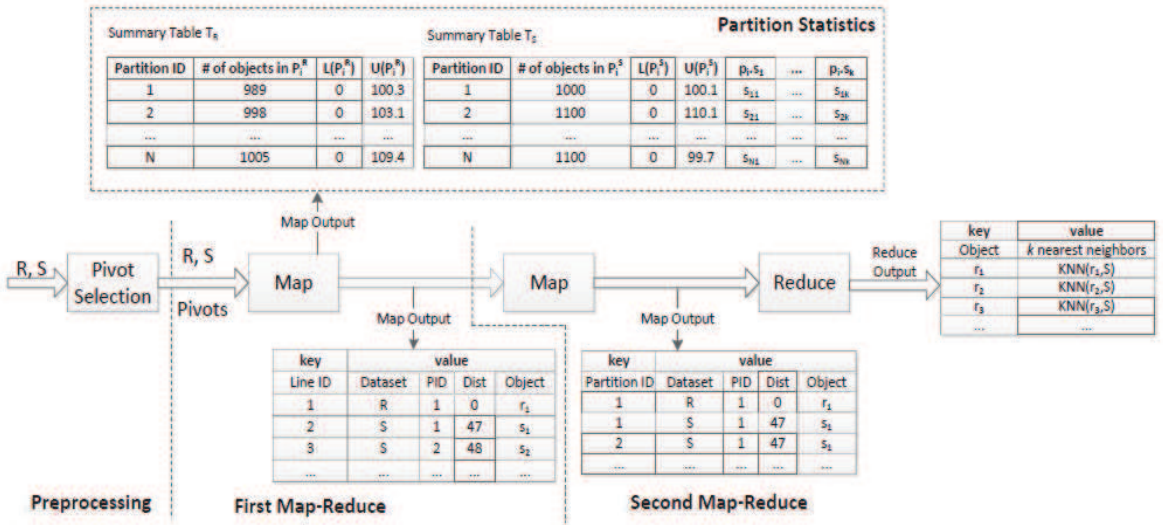


Abb. 2.12: The work-flow of performing kNN Joins using MapReduce (Lu et al. [1], Fig. 3)

3.4 PBJ with a Grouping Algorithm on Top

In the previous section we answered the question of *How to find a good partition on R?* by introducing the Voronoi Diagram based partitioning method; the questions of *How to reduce the size of S?* and *How to join each R_i only with its according S'_i?* by introducing via the distance bounding area concept a pruning rule for several objects of $s \in S$. Now the last question remains to be answered: *How to minimize replications?*

Replications on S means, that there is an object $s_1 \in S$, such that s_1 is one of the kNN of for example $r_2 \in R_2$ and of $r_1 \in R_4$, i.e. s_1 is sent to two different reducers. This is viewed by the following figure below:

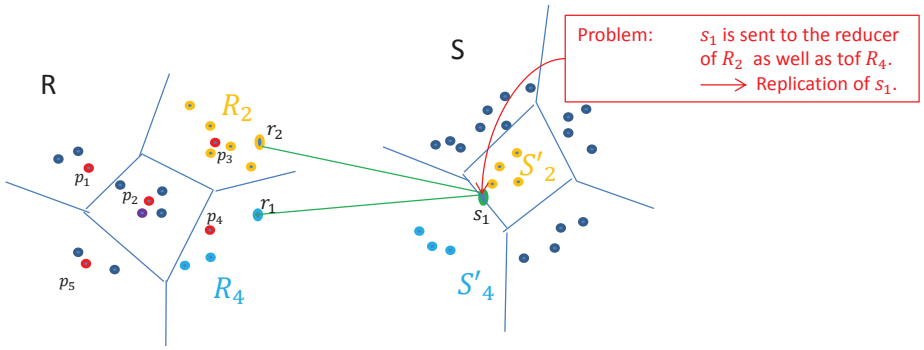


Abb. 2.13: Replication on S: s_1 is sent to two different reducers

On the one hand, due to the fact, that we are in a metric space, where the triangle inequality is fulfilled, it is clear that if there is a replication on S, in our example above the object s_1 is member of the bounding area of two different object's (In the example above $r_2 \in R_2$, $r_1 \in R_4$) k nearest neighbors. In this case the distance between the two pivotsp_1, p_4 to which the partitions belong to, i.e. partitions R_2 and R_4 in our example, cannot be too large. On the other hand, we know from the considerations of our distance bounding area that if we increase the number of pivots, the distances between the pivots decrease. The idea provided by Lu et al. [1] is now the following: Firstly we increased the number of pivots, secondly we group together the partitions of R, to which the distances of their corresponding pivots are closest (For the concrete algorithm I would like to refer to Algorithm 4 in the Appendix). It is important to note, that groups $G_1, ..., G_m$ are disjoint sets. In our example above, for example, if we increase the number of pivot by one, we can group together partitions R'_3 and R'_6, as it is shown in Fig. 14. One possible algorithm is the geometric grouping method, an iterative algorithm. We first chose the pivot p_i with the farthest distance to all the other pivots and assign its corresponding partition to group G_1. In the i-th step we have to compute the pivot p_l which maximizes the sum of its distances between all the remaining N-i pivots and assign R_l to group G_i, for $i = 1, ..., N$. After having assigned the first partition to each group, we have to select the group G_i with the smallest number of objects and compute the pivot p_l by minimizing the sum of its distances between the pivots of $G_i, i = 1, ..., N$. With this last step it is possible to achieve that the number of objects in each group is almost the same, i.e. as a result we have balanced workload for each of the N reducers. For more details and the concrete algorithm I would like to refer to Lu et al. [1]. For the grouping problem Lu et al. [1] present two different algorithms, the geometric and the greedy grouping. However within this context I focus only on the geometric grouping algorithm, because the experimental evaluations by Lu et al. [1] - which I show later - are based on this.

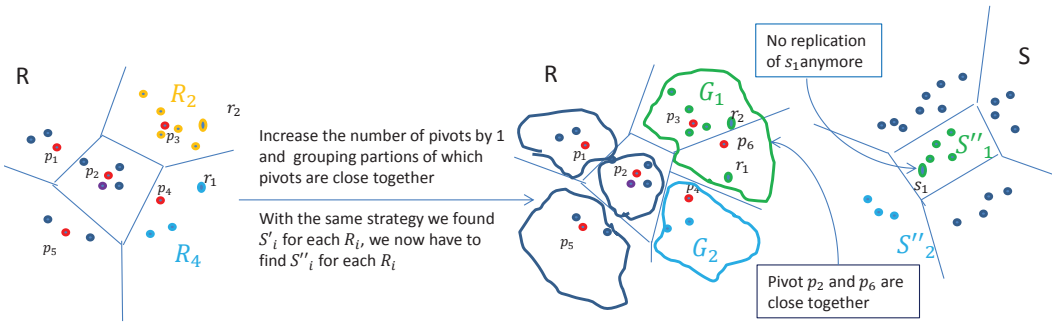


Abb. 2.14: Removing replication on S by grouping together partition, which pivots are closest

After having grouped together certain partitions, we can find our S''_i for each $R_i, i = 1, ..., N$ analogously to the way we found S'_i. The only difference is, that we have to substitute each R_i by $G_i, i = 1, ..., N$. Hence, we can rewrite Corollary 2 as follows:

Theorem 6 *Given a partition S_j and group $G_i, \forall s \in S_j$, the necessary condition that s is assigned to S''_i is that:*

$$d(s, p_j) \geq LB(S_j, G_i) \tag{2.12}$$

where $LB(S_j, G_i) = min_{\forall R_i \in G_i} LB(S_j, R_i)$

Proof See Lu et al. [1], Theorem 6.

□

Finally, the last question -*How to minimize replications on S?*- is answered by applying a grouping strategy.

4 Experimental Evaluations

In this chapter I present some experimental evaluations done by Lu et al. [1], in order to see what is the effect of pivot selecting, of k, of dimensionality and of the scalability on running time, computation selectivity and on the shuffling costs.

The authors Lu et al. in [1] use for evaluating the performance of their algorithm their in-house cluster Awan. The cluster includes 72 computing n-nodes, each of which has an Intel X3430 2.4GHz processor, 8GB of memory two 500GB SATA hard discs and gigabyte Ethernet. On each node, they install CentOS 5.5 operating system, Java 1.6.0 with a 64-bit server VM, and Hadoop 0.20.2. All the nodes are connected via three high-speed switches. To adapt the Hadoop environment to their application, they make the following changes: First, the replication factor is set to 1; second, each node is configured to run one map and one reduce task; third, the size of virtual memory for each map and reduce task is set to 4GB. The evaluation of the $H - BRJ$, PBJ and $PGBJ$ is conducted using real data sets which they expanded for their experimental evaluations to to 580x10K objects, each with 45 attributes. For more detail I would like to refer to Lu et al. [1].

4.1 The effect of Pivot Selecting Strategies

In Fig. 15 and 16 below, the effect of pivot selection strategies is showed. One can see, that there is no big difference between the random and the k means selection, in each of them the standard deviation and the partition size drops rapidly when increasing the numbers of pivots. Considering the farthest selection, we can identify a significant variation of partition size. This is due to the fact, that in the farthest selection, outliers are always selected as pivots, therefore it might happens, that in one partition, the partition with the outlier as pivot, are only a very few objects, whereas in others are plenty of objects [1]. This effect will lead to an unbalanced workload, because there is one reducer to which only a few objects are sent, and others, to which plenty of objects are sent [1]. The case looks similar when using the geometric grouping approach. Using the random and the k means selection are almost the same, whereas, when using the farthest selection, the number in each groups varies again significantly which provokes a quite unbalanced workload again.

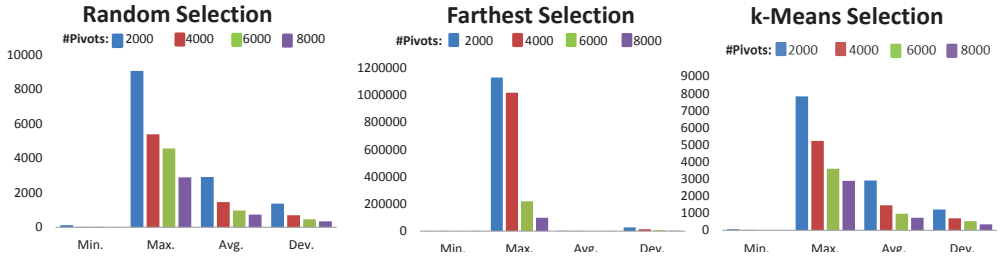


Abb. 2.15: Statistics of partition size (compare to Lu et al. [1], Table 2)

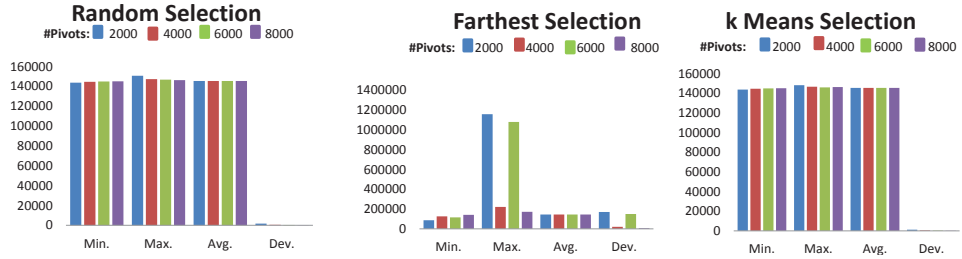
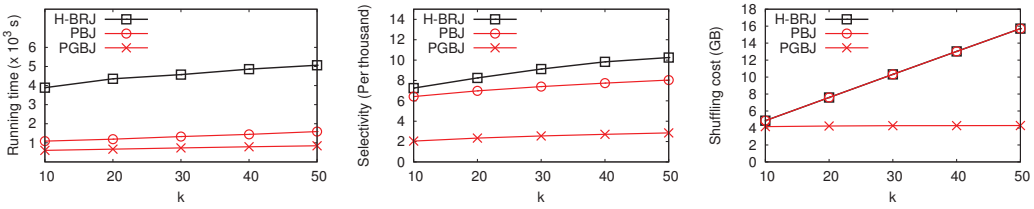


Abb. 2.16: Statistics of group size (compare to Lu et al. [1], Table 3)

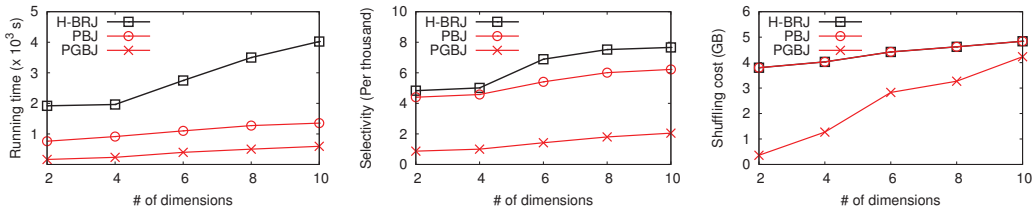
4.2 The Effect of k, Dimensionality and Scalability

Considering Fig. 17, 18, 19, we can see, that when increasing k, the dimensionality or the scalability, the $PGBJ$ performs after the PBJ strategy always the best results for the running time, computational selectivity and for the shuffling costs. This is due to the fact, that the pruning rule proposed by Lu et al. [1] helps to reduce computational costs. The $PGBJ$ has lower running times than the PBJ, which is a result by minimizing replications of set S, which reduces both, computational and shuffling costs. These arguments underlines the fact that results for computation selectivity and shuffling costs are similar. For more detail I would like to refer to Lu et al. [1].



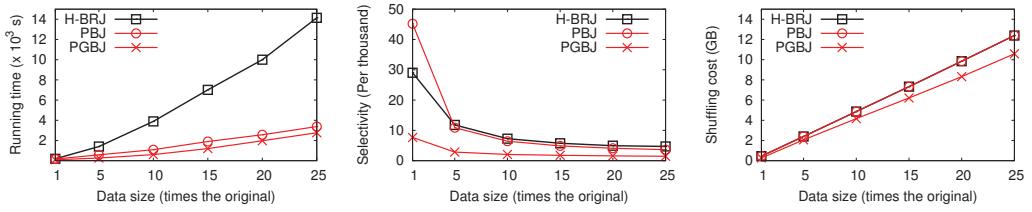
(a) running time (b) computation selectivity (c) shuffling cost

Abb. 2.17: The Effect of k on running time, computation selectivity and shuffling costs, Lu et al. [1], Fig. 8, (a), (b), (c)



(a) running time (b) computation selectivity (c) shuffling cost

Abb. 2.18: The Effect of Dimensionality on running time, computation selectivity and shuffling costs, Lu et al. [1], Fig. 8, (a), (b), (c)



(a) running time (b) computation selectivity (c) shuffling cost

Abb. 2.19: The Effect of Dimensionality on running time, computation selectivity and shuffling costs, Lu et al. [1], Fig. 8, (a), (b), (c)

5 Outline of How to Apply *PGBJ* to our Introductory Classification Example

In medical image processing, (pixel) classification works in four phases: The pre-processing, the feature extraction, the classification and the post-processing phase. The pre-processing phase includes amongst others noise removal. In the feature extraction phase, we have to find our training data set depending on which kind of classification we are interested in. In the classification phase we should run our above introduced *PGBJ* with some slight modifications. Post processing includes for example detection of abnormal region from the input image. [6, 2]. However, in our context we are especially interested in the third phase, the classification. Applying the simple kNN classification method, for each object to classify the whole training data set has to be taken into consideration, which seems obviously not quite efficient. Via the bounding area concept proposed by Lu et al. [1], we need to take into consideration that if a pivot p_j is near to another pivot p_i, then the partition R_j which corresponds to p_j has higher probability of containing objects that are closer to $r_i \in R_i \subset R$ for each object $r_i \in R_i$ to classify.

After having extracted a good training data set, we have to define:

- The training data set $S := (s_1, s_2, ...s_n)$,

- the data to classify $R := (r_1, r_2, ...r_m)$,
- choose k in order that k is far smaller than the number of objects laying in each class of the training data set and at the same time not too small,
- a function which identifies the majority vote of each object's k nearest neighbors (this should be performed in the reduce phase).

The main tasks of the Mappers will be the following:

- Load all the data (data set R and S) into the main memory of the mappers, each object assigned with a $key - value$-pair, and partition R, S into N disjoint subsets as proposed by Lu et al. [1] in the $PGBJ$ method.
- Find for each $R_i \in R$ its corresponding S_i' via constructing a pruning rule by the above mentioned bounding area from Lu et al. [1].
- create two summary tables T_R, T_S, in which several statistics about number of objects and distances are contained

Finally R_i and S_i' can be sent to one reducer for all $i = 1, ...N$ for which the main task is the following:

- to performe the join between each object r_i with its kNN $\in S$ according to the Algorithm 3 proposed by Lu et al. [1].
- to identify the majority vote of the k nearest neighbors of each object r_i to classify.
- output a list of values of the form $(r_i, classlabel)$.

In the next to last step, one has to make some modifications. In the reducing phase, after we have found for each $r_i, i = 1, ..., N$ its kNN we have to identify their majority vote. This works iteratively. We start with r_1. After we have identified its neighbor's majority vote, we can join r_1 to its class $(output(r_1, classlabel))$. Now, the size of one class of the training data set has increased by one object. Then we have to do the same for r_2 and so on till every object $r_i, i = 1, ..., N$ is joined to its class.

6 Discussion

As already mentioned, Lu et al. [1] made an important contribution to handle huge amounts of data for certain problems by developing the two methods PBJ and $PGBJ$. Within the context of PBJ and $PGBJ$ their four most important contributions are the following:

(i) They present an implementation of the kNN Joins operator using MapReduce.

(ii) They construct a distance pruning rule in order to filter partitions of the data, which are divided by using the Voronoi Diagram partitioning method. Within this procedure all objects are clustered into groups only if they are not discarded by the pruning rule.

(iii) They derive a cost model for computing the number of replicas generated in the shuffling process. Based on this they propose two grouping strategies in order to reduce the number of replicas;

(iv) They conduct extensive experiments to study the effects of dimensionality, scalability and increasing k by using two real and some synthetic data sets.

However, there still has to be discussed two important points, which are not sufficiently pointed out by the authors:

First, the problem of finding the right distance metric.

Second, the problem arising when dimensionality increases – the empty space phenomenon.

6.1 Why we cannot just use the *Euclidean distance*?

Throughout their paper Lu et al. [1] use the *Euclidean distance*. This might be because of the fact that it is the most popular. But there exist various metric spaces with different distance-metrics d, such as the *Manhattan-distance*, the *Mahanalobis distance*, the *Minkovsky distance*. From a theoretical point of view, the *Euclidean distance* might be sufficient for the considerations made by Lu et al. [1], but considering applications of their algorithm, i.e. for classification problems, a discussion of which distance metric is the most adaptive is unavoidable. Let us consider the *Euclidean* and the *Manhattan distance* which are defined respectively as follows:

$$d_{eucl.} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad d_{eucl.}(x,y) := \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \tag{2.13}$$

$$d_{manh.} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad d_{manh}(x,y) := \sum_{i=1}^{n}|x_i - y_i|. \tag{2.14}$$

Let $n = 2$, $x, y \in \mathbb{R}^2$, $x = (1,3)^T$, and $y = (2,5)^T$ then the $d_{eucl.}(x,y) = \sqrt{5}$ but $d_{manh.}(x,y) = 3$. Obviously $\sqrt{5} \neq 3$. For the the graphical view of this example I would like to refer to Fig. 20 below.

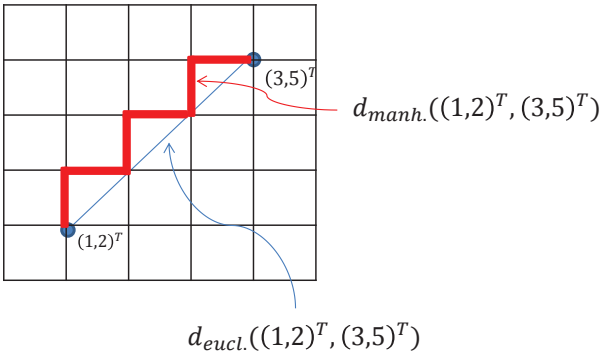


Abb. 2.20: The blue line denotes the *Euclidean* and the red line one possible *Manhattan distance*

But what does this mean for our classification example in the metric space $(M,d), M \subset \mathbb{R}^2$, $d = d_{eucl.}$ shown in Fig. 2 at the beginning? The answer is based on basic analysis lectures. The *Euclidean distance* denotes a circle, the *Manhattan distance* a cuboid. Therefore it might happens, that we have complete different results between using these two metrics. In our classification problem of Fig. 2 we see, that if $k = 3$ then there is no difference, the green point is still assigned to the red class, but in the case that $k = 9$, the green point is not classified anymore to the blue class label, but to the red class label.

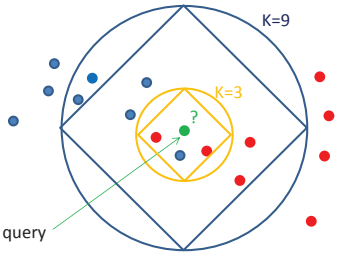


Abb. 2.21: The circle denotes the Euclidean and the cuboid the Manhattan distance

The case is similar if we make the same consideration for the *Mahanalobis distance*, which denotes an ellipsoid. Hence we can see, that the performance of the *kNN Classification* depends crucially on the distance metric used to identify nearest neighbors. Obviously, the optimal distance metric is amongst others dependent on the problem, we have to solve.

One possible method how to solve the distance problem in the application of the *kNN Classification* is proposed by Weinberger et al. [12]. The authors show how to learn a *Mahanalobis distance metric* for *kNN Classification* in order get an optimized *Mahalanobis distance function*.

The *Euclidean distance* has another problem: In high dimensional spaces, all points tend to be equally distant from each others, with respect to the *Euclidean distance*. This problem and the phenomenon of the empty space have been discussed among others by Verleysen et al. [13].

6.2 High dimensional data and the empty space phenomenon

Verleysen et al. [13], based on researches of Scott and Thompson [14], show weird facts about high-dimensional spaces. One of the fact is, that the volume of a hyper-sphere of unit radius (defined by $V(d) = \frac{\pi^{d/2}}{\Gamma(d/2+1)} r^d$) goes to zero as dimension growths. This observation implies that high-dimensional spaces are mostly empty.

Moreover, Verleysen et al. [13] discuss the fact that the standard deviation of the norm of random vectors converges to a constant as dimension increases though the expectation of their norm growth as the square root of the dimension. As metrics are induced by norms, this observation holds also for the *Euclidean distance metric* (considering random vectors).

The authors draw from these observations the conclusion, that in high dimensional spaces, all points tend to be equally distant from each others, with respect to the *Euclidean norm/metric*. If this is the case, then the algorithm proposed by Lu et al. [1] doesn´t work anymore – we are unable to find any k nearest neighbors for an arbitrary point since all distances have the same value. This is a problem of each similarity search.

Verleysen et al. [13] propose to consider new similarity measures between data or to reduce the dimension through projection on (non-linear) sub-manifolds. In case of high dimensional MRI data, there can be performed dimensionality reduction, for example, via region of interest based measures as input features [15] or also by non-linear methods [16].

7 Conclusion

In this seminar paper I discussed the PBJ and $PGBJ$ methods, two resulting methods by performing kNN Joins using MapReduce, proposed by Lu et al. [1]. Via Voronoi Diagram based partitioning method they derive a pruning rule in order to answer the *k Nearest Neighbor Joins* by only checking object pairs within each partition as well as to *reduce* the shuffling and the computational costs. Especially the $PGBJ$ is a new and very efficient algorithm, which uses on top of the PBJ a grouping algorithm in order to group partitions to which corresponding pivots are closest, together. With this method replications can be minimized, i.e. the shuffling and computational costs.

Their proposed methods can be applied in various areas and it might be important also for medical image processing, such as for solving classification problems as shown in Section 5, since the amount of data of a MRI is becoming overwhelming as well. Despite of their very interesting, especially theoretical approach, there still have to be done researches on the choice of an optimized distance metric and on the empty space phenomenon when using high dimensional data sets. These two problems are recently becoming important, since not only the amount of data is increasing very fast over the whole world but the dimensions considered are growing higher.

As already mentioned, one possible application of the PBJ or $PGBJ$ method proposed by Lu et al. [1] is pixel classification amongst others in the medical image processing area. However there still have to be some slight modifications on Lu et al. [1] proposed algorithm. My future task will be (may be within the scope of another seminar paper) to elaborate the modification of the algorithm, to implement it and to do some experimental evaluation.

Literatur

[1] Lu W, Shen Y, Chen S, Ooi BC: Efficient Processing of k Nearest Neighbor Joins using MapReduce. PVLDB, 2012; 5(1): 1016–1027.

[2] Elsayed A: Region of Interest Based Image Classification: A Study in MRI Brain Scan Categorization [PhD-Thesis]. University of Liverpool; 2011.

[3] Aji A: High Performance Spatial Query Processing for Large Scale Scientific Data Proceeding [PhD-Thesis]. SIGMOD/PODS 2012; PhD Symposium: Pages 9–14.

[4] Okcan A, Riedewald M: Processing theta-joins using MapReduce. ACM SIGMOD 2011: 949 – 960.

[5] Tönnis KD. Grundlagen der Bildverarbeitung (Hrsg.): Physikalisches Taschenbuch. Pearson, München, 2005.

[6] Ramteke RJ, Yashawant KM: Automatic Medical Image Classification and Abnormality Detection Using K Nearest Neighbour. ISSN 2012; 2(4): 190 – 196.

[7] Zhang C, Li F, Jestes J: Efficient parallel kNN Joins for large data in MapReduce. EDBT 2012: 38 – 49.

[8] Gupta H, Chawda B, Negi S, Faruquie TA, Subramaniam LV: Processing Multi-Way Spatial Joins on Map-Reduce. EDBT 2013: 113–124.

[9] Dean J, Ghemawat S.: MapReduce: Simplified data processing on large clusters. OSDI, 2004: 137 – 150.

[10] C. Böhm and F. Krebs: The k-Nearest Neighbor Join: Turbo Charging the KDD Process. In Knowl. Inf. Syst., 2004; 6: 728–749.

[11] Zeller H, Gray J: An Adaptive Hash Join Algorithm for Multiuser Environments VLDB, 1990: 186–197.

[12] Weinberger KQ, Blitzer J, Saul LK: Distance Metric Learning for Large Margin Nearest Neighbor Classification. JMLR, 2009; 10: 207–244.

[13] Verleysen M, François D, Simon G, Wertz V. Artificial Neural Nets Problem Solving Methods. Heidelberg, Springer; 2003. Book chapter, On the effects of dimensionality on data analysis with neural networks: p.105–112.

[14] Scott DW, Thompson JR: Probability Density Estimation in Higher Dimensions. In: Douglas SR, editor. Computer Science and Statistics: Proceedings of the Fifteenth Symposium on the Interface. Amsterdam; 1983. P. 173–179.

[15] Magnin B, Mesrob L, Kinkingnehun S, Pelegrini-Issac L, Colliot O, Sarazin M, Dubois B, Lehericy S, Benali H: Support vector machine-based classification of Alzheimer´s disease from whole-brain anatomical MRI. Neuroradiology 2009; 51: 73–83.

[16] Akhbardeh A, Jacobs MA. Comparative analysis of nonlinear dimensionality reduction techniques for breast MRI segmentation. Med Phys. 2012; 39(4): 2275–89.

8 Appendix

Algorithm 1 (Lu et al. [1]): bounding$kNN(P_i^R)$

1 create a priority queue PQ;

2 foreach P_j^S do

3 foreach $s \in KNN(p_j, P_j^S)$ do

4 $ub(s, P_i^R) \leftarrow U(P_i^R + d(p_i, p_j) + d(s, p_j))$;

5 if $PQ.size < k$ then $PQ.add(ub(s, P_i^R))$;

6 else if $PQ.top > d$ then

7 $PQ.remove(); PQ.add(ub(s, P_i^R))$;

8 else $break$;

9 return $PQ.top$

.

Algorithm 2 (Lu et al. [1]): compLBOReplica()

1 foreach P_i^R do
2 $\quad\theta_i \leftarrow boundingkNN(P_i^R)$;
3 foreach P_j^S do
4 $\quad$foreach P_i^R do
5 $\quad\quad LB(P_j^S, P_i^R) \leftarrow d(p_i, p_j) - U(P_i^R) - \theta_i$;

.

Algorithm 3 (Lu et al. [1]): kNN join

1 map-setup
2 $\quad$compLBOReplicas();
3 $map(k_1, v_1)$
4 $\quad$if $k_1.dataset = R$ then
5 $\quad\quad pid \leftarrow getPartitionID(k_1.partition)$;
6 $\quad\quad output(pid, (k_1, v_1))$;
7 $\quad$else
8 $\quad\quad p_j^s \leftarrow k_1.partition$;
9 $\quad\quad$foreach P_i^R do
10 $\quad\quad\quad$if $LB(P_j^S, P_i^R) \leq k_1.dist$ then
11 $\quad\quad\quad\quad output(i, (k_1, v_1))$;
12 $reduce(k_2, v_2)$
13 parse P_i^R and $S_i^S(P_{j1}^S, ..., P_{jM}^S)$ from (k_2, v_2);
14 sort $P_{j1}^S, ..., P_{jM}^S$ based on the ascending order of $d(p_i, p_j)$;
15 compute $\theta_i \leftarrow max_{\forall s \in kNN(P_i^R, S)} |ub(s, P_i^R)|$;
16 for $r \in P_i^R$ do
17 $\quad\theta \leftarrow \theta_i; kNN(r, S) \leftarrow \theta$;
18 $\quad$for $j \leftarrow j_1$ to j_M do
19 $\quad\quad$if P_j^S can be pruned by Corollary 1 then
20 $\quad\quad\quad$continue;
21 $\quad\quad$foreach $s \in P_j^S$ do
22 $\quad\quad\quad$if s *is not pruned by Theorem* 2 *then*
23 $\quad\quad\quad\quad refine\ kNN(r, S)\ by\ s$;
24 $\quad\quad\quad\quad \theta \leftarrow max_{\forall o \in kNN(r,S) d(o,r)}$;

25 $output(r, kNN(r, s))$;

.

Algorithm 4 (Lu et al. [1]): geoGrouping()

1 select p_i such that $\sum_{p_i \in \mathbb{P}} d(p_i, p_j)$ is maximized;

2 $\tau \leftarrow \{p_i\}; G_1 \leftarrow \{P_i^R\}; \mathbb{P} \leftarrow \mathbb{P} - \{p_i\};$

3 for $i \leftarrow 2$ to N do

4 select $p_l \in \mathbb{P}$ such that $\sum_{p_j \in \tau} d(p_l, p_j)$ is maximized;

5 $G_i \leftarrow G_i \bigcup \{P_i^R\}; \tau \leftarrow \tau - \{p_l\};$

6 While $\mathbb{P} \neq \emptyset$ do

7 select group G_i with the smallest number of objects;

8 select p_l $in \mathbb{P}$ such that $\sum_{\forall P_i^R \subset G_i} d(p_l, p_j)$ is minimized;

9 $G_i \leftarrow G_i \bigcup \{P_l^R\}; \mathbb{P} \leftarrow \mathbb{P} - \{p_l\};$

10 return $\{G_1, G_2, ..., G_N\}$

Video-Based Activity Recognition during Trauma Resuscitation

Seminar Medical Image Processing

Eric Marre

Zusammenfassung

Trauma resuscitation is a first aid procedure for severely injured people and at the same time a very challenging situation for all involved clinicians. Time critical decisions have to be made and parallel procedures have to be performed. Computer-assisted decision support can be used to avoid errors in this situation. The recognition of activities is an important component to realize these systems. But the trauma resuscitation situation brings high demands to an activity recognition system because it has to detect fast and simultaneous activities performed by multiple agents.

The objective of this work is to analyze an activity recognition framework which is able to handle this difficult scenario. The difficulties in realizing such a framework are illustrated and described. After this the components of the framework and their functionality are shown. This work focuses on the core components which are Human Pose Estimation, Hand Tracking and a component for logical inference. In a final step the evaluation of this framework is analyzed and discussed.

Keywords: video-based, non-intrusive, activity, recognition, trauma, resuscitation

1 Introduction

Trauma resuscitation is a first aid procedure for severely injured people. It is a very susceptible situation for medical errors because the medical team is often confronted with unstable patients, time critical decisions and has to manage parallel activities. Moreover the involvement of many medical disciplines leads to the involvement of many participants. The situation can be described as frantic and is also critical due to the danger of the patient's life. It occurs that personnel has to work over hours in this challenging situation. Because of these factors typical errors like delays to treatment or care errors arise. [16]

To avoid these typical errors applicable management protocols like "Advanced Trauma Life Support" (ATLS) are established for this setting. Although these protocols are very effective, human error factors still exist even in experienced trauma teams [16]. Computer-assisted decision support can be used to avoid the omission of steps and to aid coordination [8]. For example such a system can warn the medical team when a procedure is interleaved or performed at the wrong time. This can be realized by detecting activities in a temporal context to check whether procedures are performed at the right time. An example for an activity recognition in this situation is given by Figure 3.1 where typical activities are labeled

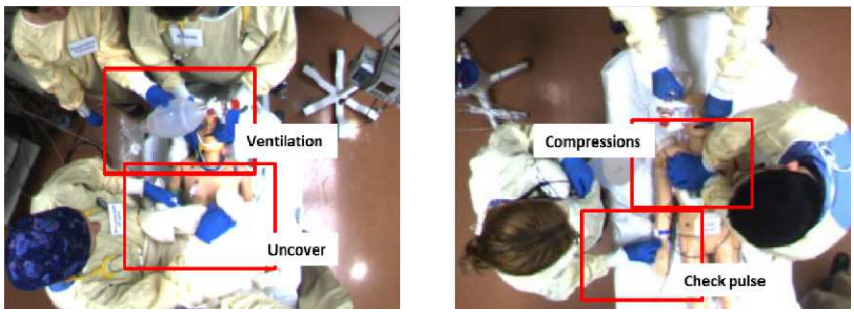


Abb. 3.1: Example of recognized activities in a ceiling view. Images from [8]

Tab. 1: Different approaches of activity recognition

Method	**Non-intrusive**	**Human interpretable**	**Multi-agent**	**Complex activities**
Sensory Cues [9]	No	No	Yes	Yes
Action Recognition [15]	Yes	Yes	No	No
Human object interactions [3]	Yes	Yes	No	No
Multi-person activities [13]	Yes	Yes	Yes	No
Presented approach [8]	Yes	Yes	Yes	Yes

in the images. It is important that the used system does not restrict the options of the medical team to achieve the best treatment for the patient. But many existing systems require active user interaction or manual input of data [8]. By that most of these systems have limited usability in this time-sensible setting. Another important point is that the system has to be reliable and accurate because a failure of a decision support system or wrong warnings can have fatal consequences.

The remainder of this work is organized as follows: Section 2 of this work gives a short overview over the common methods to recognize activities. Afterwards a framework which is designed to detect and transcribe activities during trauma resuscitation is presented. A short introduction into the general structure of this framework is given in Section 3. Next the different core components of the framework are described. The topics Human Pose Estimation (Section 3.1), Hand Tracking (Section 3.3) and the logical inference method (Section 3.6 and Section 3.5) are treated in more detail. At the end the evaluation of the activity recognition framework and its included core components is presented in Section 4 and discussed in Section 5.

2 Related Work

There are multiple methods to detect activities. Table 1 gives an overview over some of these approaches which are described in this section in more detail.

The first approach is given by sensory cues which can deal with multiple agents and can also handle complex activities [8]. For example a RFID-chip based approach is presented in [9] in which the sensor provides object centric signals. As a disadvantage they are intrusive because most of them assume a modification of devices or a special user preparation. As described in the introduction this can lead to an inhibition of the medical treatment and risks the patient's life. Further, it is challenging for humans to interpret the generated data. But it can be useful in future for a post-analysis or a reconstruction of the process by humans.

Representatives of video-based approaches are for example action recognition [15], multi-person activities [13], human-object Interaction [3] but also the framework [8] which is described in this work. Video-based approaches do not restrict the options of the medical team because they are usually non-intrusive. This makes video-based approaches more feasible for trauma resuscitation. Besides these approaches are easily human interpretable and do allow a post-analysis. In the remainder the above named approaches are described briefly.

In action recognition, single activities like running or walking are recognized. Older methods of this category used bag-of-word methods but concurrent work takes into account spatial and temporal relations of features. Action recognition methods can be realized by time series and different inference approaches like Dynamic Time Warping or Hidden Markov Models. [15]

Human-object interaction focuses on the object and hand motions to determine an activity [3]. The method is for example suitable to detect an activity when object manipulations can occur [3].

But a disadvantage of action recognition methods and human-object interaction is that the methods are not designed to track and transcribe activities by multiple agents. In trauma resuscitation a team consisting of several agents performs activities and therefore both methods are currently not suitable.

Another video-based approach which is admittedly able to detect multiple-agent activities is given by multi-person activities [13]. Information about events can be extracted based on a graph of spatio-temporal interactions in this approach. Inference in this approach is similar to the presented framework [8] because both use Markov Logic Networks. This approach is mainly investigated in the sports domain and for example used to track an one-on-one basketball game [13]. But in contrast to the medical domain

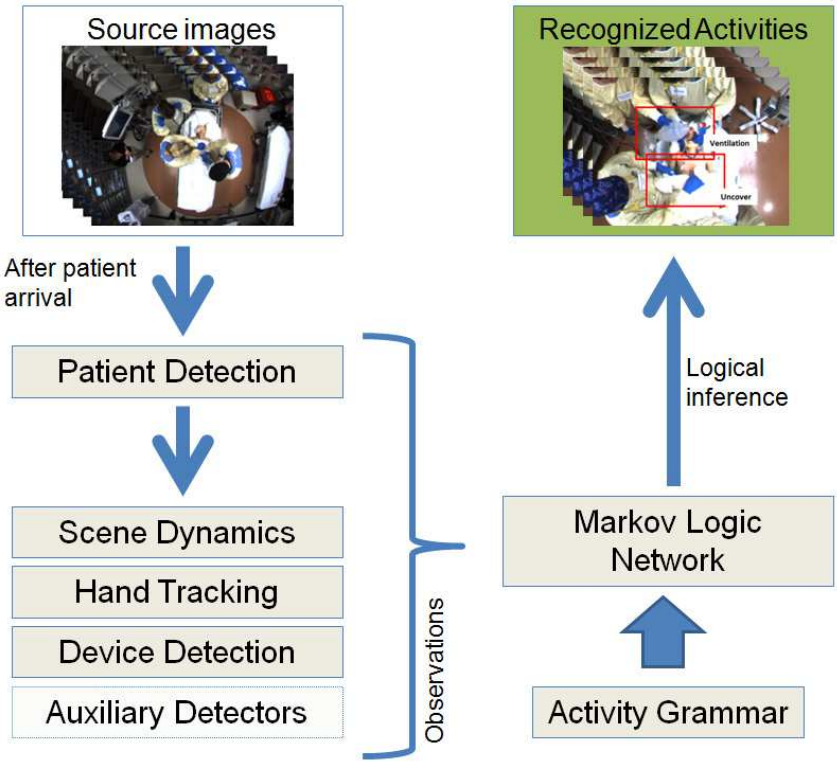


Abb. 3.2: Simplified structure of the activity recognition framework in lean to [8]

activities in the sport domain consist only of a small set of repeated actions [8]. Furthermore they usually include just interactions between a single person and a single object [8].

As one can see it is challenging to find an approach that can deal with complex trauma resuscitation activities and is human-interpretable and non-intrusive at the same time. Therefore, this paper focuses on a system which complies all of these properties. The system is using a ceiling-mounted, wide-angle camera and is thus fully automated and non-intrusive system. More specific, the camera is positioned directly over the patient in a resuscitation room as one can see in Figure 3.1. The system is described and evaluated by Ishani Chakraborty et al. [8]. This system the first one which is designed to make activity recognition possible even in this trauma resuscitation situation [8]. It tracks and transcribes the performed procedures with their temporal alignment. The key trauma procedures and their prioritization are based on ATLS protocol. As described above the trauma resuscitation scenario is very dynamic and has many similarly attired participants. The need of time critical decisions create rapidly moving objects which are difficult to recognize. A lot of activities can occur simultaneously or are similar to other activities. All these general conditions pose strong requirements on a video based system. [8]

3 Components of the Activity Recognition Framework

The basic assumption for the design of the focused framework is that complex activities are a composition of fine and simple activities. For example an intubation can be split into hand motions and the use of the appropriate device in the head area. These subactivities are less complex to detect than the entire procedure. For recognizing such subactivities the framework uses state-of-the-art methods integrated into individual components. The results are combined by a logical component to make an inference that leads to the performed medical procedure.

Figure 3.2 shows a short overview of the structure of the framework. The system detects the patient using human pose estimation at the beginning. This is described and explained in Section 3.1 of this work. Then it analyzes scene dynamics, detects the occurring devices and tracks hand positions. The hand tracking is especially challenging due the given situation so it is treated more in detail in this paper. All the given information is transferred into action attributes like “clinician approaching location x“. The last but essential step is executed by two logical components presuming a performed procedure. On the

one hand an Activity Grammar (AG) provides a set of logical statements. Each logical statement describes a medical procedure by using the action attributes. On the other hand it is made by a Markov Logic Network (MLN) in which the probabilities of the action attributes and the logical semantics given by the AG are probabilistically combined. Due to their important function both components are explained in this paper in more detail. [8]

3.1 Human Pose Estimation

As mentioned before, the framework detects the patient pose at the beginning of the trauma resuscitation phase. In this case a pose consists of the position and orientation of the patients head, chest and the upper/lower limb. After the recognition of these parts the framework assumes that the patient remains at the same position until trauma resuscitation is finished. [8]

Through pose estimation it is possible to make a body dependent distinction of activities and to increase accuracy [8]. For example an activity which could be artificial respiration on the wrong position seems to be less probable. Another point is that the pose estimation allows to detect scene dynamics in a general way [8] which is described in the next section.

For detection the framework uses a 2D pose estimation algorithm which was described by Vittorio Ferrari et al. in [20]. One of the greatest benefits of this algorithm is that it is fully automatic and self-initializing [20]. Another benefit is the robustness which lets the algorithm also work on cluttered images with dark lighting, low contrast and overlapping body parts [20]. Moreover, it does not rely on skin color segmentation [20]. These properties makes this algorithm suitable for the trauma resuscitation scenario.

The following subsections describe the 2D pose estimation algorithm. First the algorithm determines a rough area which contains the person. Then the area is reduced in a further step so the area can be used as a search area for body parts. Afterwards the parsing step for body parts begins. Finally a spatio-temporal parsing method is described which can improve pose estimation results further.

3.1.1 Patient Detection The algorithm starts with a weak model of a person which is detected by an upper-body detector. This weak model contains only the area where the torso and the head should lie. Later it is used as an anchor for the pose search area. Furthermore, it reduces the search space for body parts to a part of the image. This makes the algorithm faster and avoids dispensable search operations. [20]

Whereas the algorithm in [20] uses Histogram of Oriented Gradients features with a sliding-window mechanism and a Support Vector Machine the activity recognition framework uses “spatial prior and motion cues“ to detect a weak body model [8]. In detail the framework interprets a high action as the patients arrival and a low activity in the central zone of the image as a fixed patient [8]. This low action area is initially taken as a rough search area for body parts [8]. Figure 3.3 shows the full process in an example situation. On the left side the action measure of a pixel of each zone is visualized in dependence to the time. The central and peripheral zones are shown in the middle of the figure where they are represented by a white grid. Body parts are calibrated when the action in the central zone is low. The resulting, estimated pose can be seen in the right image of the figure.

In [20] the upper body detection is further improved by temporal association. This means that the bounding-boxes of the body in each frame are associated to the same person. This can increase detection rate in a sequence of images.

The result of the upper-body detection is a rough search area for body parts in form of a bounding-box. Moreover the area gives an approximate size for body parts which is normally proportional to the detection’s scale. Nevertheless, arms can lie outside this search area so the area is extended by a circle centred between the shoulders. But this extension and the rough estimation of the search area creates a lot of background clutter. It can impede a correct recognition of body parts. In connection to that the next subsection deals with a solution for this problem.

3.1.2 Foreground Highlighting The aim of foreground highlighting is to reduce the search area for body parts by removing excessive background. This is possible by using the rough search area from the patient detection [20]. It is known that the head must lie in the middle upper-half of the detected area and the torso below it [20]. Unfortunately, the arms can be located at an arbitrary position. This knowledge of the search area structure is used to initialize a segmentation algorithm named Grabcut [20]. This algorithm identifies the background which makes it possible to remove it [20]. Furthermore it does

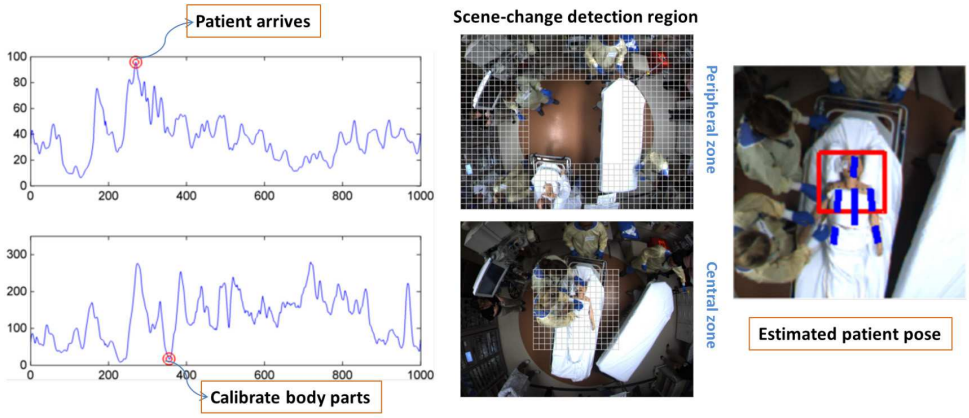


Abb. 3.3: On the left side one can see the motion in central zone and in peripheral zone displayed in a graph. Patient arrival and the time of body part calibration are marked in this graph. In the middle the central zone and the peripheral zone are visualized. On the right side the patient pose is estimated. Figure taken from [8]

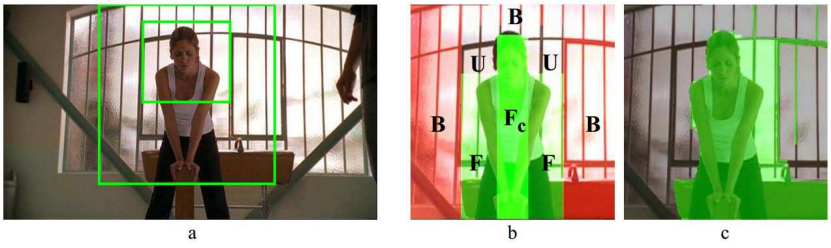


Abb. 3.4: The left image shows the rough search area for the torso and the head. In the middle the alignment of the different areas for Grabcut is shown. The image on the right side represents the output from the foreground highlighting. The green area is identified as foreground. Figure taken from [20]

not need to know the background a priori [5]. Another advantage in this scenario is that it can deal with complex environments where the background cannot subtracted trivially [5]. It is not the primarily aim to achieve a perfect segmentation of foreground and background. It is more important that none of the body parts is wrongly excluded from foreground. So the used algorithm is working conservatively to avoid losing relevant information. An example result of the process can be seen in Figure 3.4.c where the green region is the assumed foreground. [20]

More precisely the body part search area (green outer box in Figure 3.4.a) is divided into four different classes of subregions. Figure 3.4.b shows the exact alignment of the subregions. Furthermore each class is labeled by a character. The class F_c contains mostly parts of the person because it is centred and therefore congruent with the head and the torso of the person. F contains mostly foreground but can also contain parts of the background. This is the area where arms can be enclosed when they are reached out. Class B contains the known background which is far away from the assumed person's location. The last subregion class is given by U which represents neutral areas. These subregions are not reliably categorizable because the head position influences the appearance. In a further step Grabcut can assign a pixel of a subregion to another class except it is assigned to class F_c. This is because Grabcut learns a foreground model from F_c and F and a background model from B. Pixels which are assigned to U are ignored in learning process. After the learning process pixel labels are reassigned. Background can be identified and removed by this new assignment. [20]

3.1.3 Single-Frame Parsing After the reduction of the search area the image parsing technique by Deva Ramanan [17] is used to detect individual body parts. Thereby the iterative algorithm learns a region model for each body part using edge information. This defines several body part regions. These known regions can be improved further by further iterations which are based on the edge information and the body part regions of the previous step. Through this technique, region models can be learnt from the

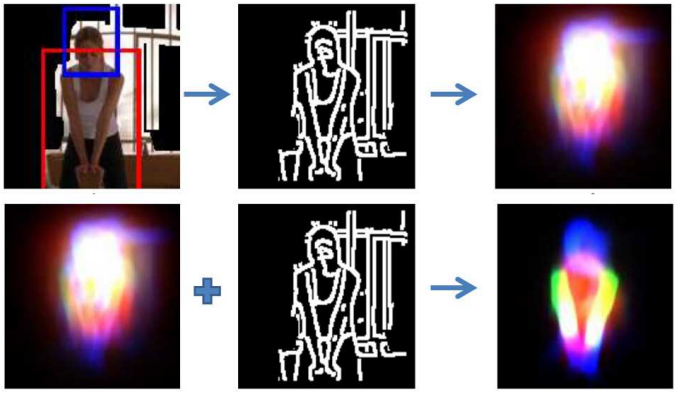


Abb. 3.5: The first line shows extraction of edge information and a first iteration based on this information. The second line shows a second iteration based on the previous result. Included images taken from [20]

image itself. In addition the foundation on edge information makes this technique robust because edge information is not dependent on a specific color. Figure 3.5 shows two iterations of the parsing technique. In the first line one can see a first, initial iteration and in the second line a second iteration based on the previous region model and the edge information. The region model is visualized in the figure as different colored areas where each color complies to a body part in this figure.

Each body part l_i is located in a patch. Each patch can be parameterized by image position x_i, y_i, a scale factor s_i and the orientation θ_i where the position is at the left top corner of the patch. So a configuration of parts is written as $L = (l_1, ..., l_K)$. It is assumed that the structure of edges E is a tree and each patch is connected to at most one parent . Then an edge in E is a connection between two body parts i and j and is written as (i, j). A posterior of a configuration of parts in a given image I can be written as a log linear model

$$P(L|I) \propto exp(\sum_{(i,j)\in E} \Psi(l_i, l_j) + \sum_i \Phi(l_i)) \tag{3.1}$$

as shown in [17].

$\Psi(l_i, l_j)$ corresponds to a spatial prior on the relative arrangement of the body parts l_i and l_j [20]. In other words $\Psi(l_i, l_j)$ is a measure how good two body parts get together. Thereby it represents the kinematic constraints like the attachments of arms to the torso [20]. These constraints are extended in the activity recognition framework by a symmetry constraint on limb position [8]. With this symmetry it is assumed that the patient is fixed in a specific position in trauma resuscitation.

$\Phi(l_i)$ corresponds to the local image evidence for a part. On the basis of the assumption that model structure E is a tree the inference can be performed by sum-product Belief Propagation in a exact and efficient way. [20]

Nevertheless this parsing technique can fail entirely on cluttered images. This happens particularly in images with a complex background and a structure which is similar to body parts. In this case the given edge information is not sufficient to build a reliable region model. Through this initial distortion the results of all following iterations are also distorted. [20]

It can be a solution to reduce the parser search space using information from previous information extraction. So the patient detection gives a rough search area which location and scale information can be used. This reduces the detection of false positives and also avoids ambiguity. Further the foreground highlighting can be used to improve parser result by removing all edges in the background region. As a third improvement it is possible to add head and torso constraints. In detail the location where the torso and the head lies can be estimated as well as in Section 3.1.2. By setting $\Phi(l_{head})$ and $\Phi(l_{torso})$ to $-\infty$ for all l_i outside the head and the torso search area it is possible to search only in promising areas for this two body parts. All these improvements lets the parser deal with highly cluttered images and enables a fast and robust 2D pose estimation. [20]

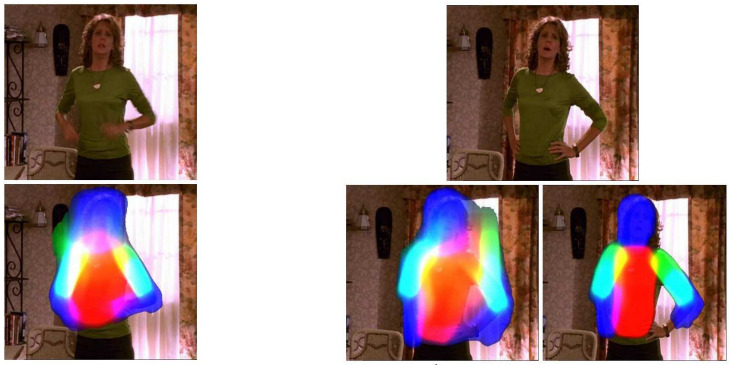


Abb. 3.6: On the left side a frame with a low TPE and the result (a). In the middle (b) a frame with a high TPE. On the right side one can see the result of spatio-temporal parsing (c). Included images taken from [20]

3.1.4 Spatio-temporal parsing The main idea of spatio-temporal parsing is to take several frames as a source. Thereby more information is available and human pose estimation can be improved further. This method can be an enrichment especially for frames which are difficult to parse. [20]

A possible approach to realize this is described in [20]. The basic idea is to choose a series of frames where the system detects the pose presumable correctly. The information is used to parse the frames again with this new and richer information. More extensively the measure of confidence which is used in the approach is the accumulated entropy of the posteriors of all part positions. The entropy of the posterior for one part l_i is given by $H(p(l_i, I))$. The total pose entropy for all parts is defined as

$$TPE = -\sum_i H(p(l_i, I)) \tag{3.2}$$

Learning is achieved by integrating the appearance models of body parts over several frames. But only frames with a low and similar TPE are chosen. The person model of [17] gives also the opportunity to represent dependencies between body parts over a range of frames. In dependence on the time t the posterior of all configurations of parts $\{L^t\} = \{l_i^t\}$ under requirement of the images $\{I^t\}$ can be written as

$$P(\{L^t\}|\{I^t\}) \propto exp(\sum_{t,k}(\sum_{j|(i,j)\in E} \Psi(l_i^t, l_j^t) + \Phi(l_i^t) + \Omega(l_k^{t-1}, l_k^t))) \tag{3.3}$$

As one can see the formula is supplemented by Ω which represents the temporal dependencies of a body part in different frames. As a temporal prior $\Omega(l_k^{t-1}, l_k^t)$ a box-shaped distribution can be used to limit the difference of a body part between frames [20]. As a result final pose estimations are more accurate with a spatio-temporal model in some cases. Figure 3.6 shows an example for such a case. The right image in the first line of the figure shows a source image. The first image (Figure 3.6.b) of the figure below this image shows an region model without spatio-temporal parsing. The second image (Figure 3.6.c) the region model with spatio-temporal parsing. One can see that the region model for body parts (visualized by colored regions) fits much better to the pose in the source image than without spatio-temporal parsing.

3.2 Scene Dynamics

For a better separation of activities it is useful to recognize scene changes which occur when moving the patient or the equipment. In order to realize these global and non-local procedures the detection of environment changes can be used. The activity recognition framework distinguishes between environment changes in the patient area and the area around the patient. The localization of the patient described in Section 3.1 makes this possible.[8]

An approach which is suitable to realize the detection of environment changes is proposed by Grimson et. al. [19]. In this approach all pixels are partitioned into foreground and background pixels. A memory-based online background model tracks intensity changes of pixels and determines whether they belong to the foreground or background. Pixel intensities are represented as Mixtures of Gaussians. When a pixel

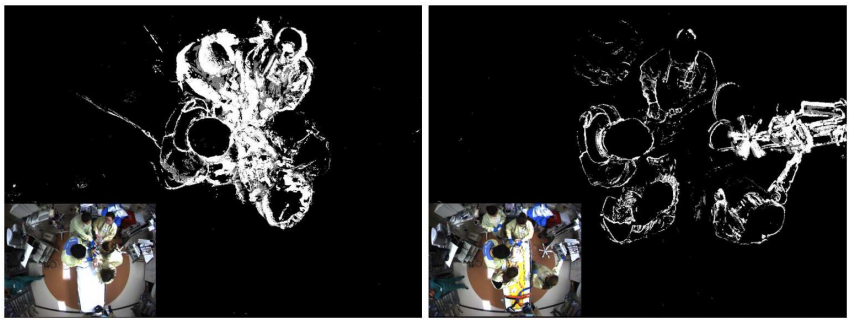


Abb. 3.7: Both images show rapid pixel intensities changes which are visualized as white pixels. In the image on the left side patient pose is changing. In the image on the right side new equipment arrives. Figure taken from [8]

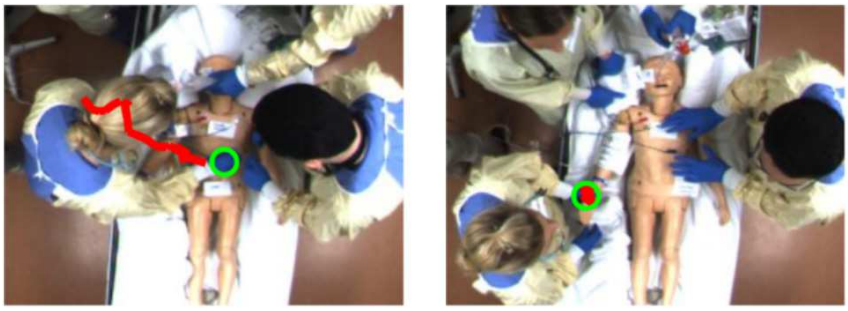


Abb. 3.8: On the left side hand is approaching towards the patient. On the right side the hand is paused to check pulse. Images taken from [8]

changes its intensity rapidly it is added to the foreground otherwise it is part of the background. All in all each pixel in the following image is assigned either to foreground or to background by a Gaussian distribution based on previous image information. [8]

The foreground is the partition which contains the relevant information for detecting a scene change. Appearance changes lead to an expansion of the foreground partition. Therefore the mean response of pixel intensities in the patient area or the peripheral area rises [8]. If the mean response exceeds a certain threshold a scene change action attribute is triggered [8]. Figure 3.7 shows this process. One can see the change of the patients pose and the arrival of new equipment. Foreground pixels in this figure are drawn white and the background pixels are black.

3.3 Hand Tracking

A lot of activities are performed by humans in the medical domain and especially in the trauma resuscitation situation. Two examples for such activities are given by Figure 3.8 where the hand tracks are visualized by red pixels. So tracking of hand positions is one of the most important components in the activity recognition framework. But the time critical situation leads to fast moving hands. In addition multiple hands have to be tracked and can be overlapped by other objects. So there are high demands to a hand detection system which is used in this situation. [8].

A popular tracking approach in general is Bayesian Sequential Estimation [10]. But the approach requires approximation of complex likelihood models which often leads to an intractable inference in practical [10]. Another approach is given by Sequential Monte Carlo methods [10]. These methods are characterized by simplicity and flexibility and are also parallelizable [1]. A deficit of this method is the poor performance in tracking multiple targets [10]. But this can be achieved by maintaining multi-modality which is described in the remainder of this section.

The approach used by hand tracking is also based on the Monte Carlo method but is able to maintain multi-modality [10]. This is achieved by mixture models [10]. So a target distribution is formed by a non-parametric mixture of filtering distributions [10]. The tracking targets are represented by the multiple modi of this distribution. The distribution can be computed by a prediction step based on previous

measurements and an update step when the new data is available [10]. The approach was presented by Vermaak et. al. in [10]. As a measurement the color of gloves is modeled which are worn by all clinicians. In the medical domain this is compulsive and usually only one color is used. To filter out luminance to make this method robust against lighting intensity changes the YUV color model is used. In this model the U and the V components represent the color and the Y component the luminance. Furthermore it is not important that a target is always identified as the same target. So it is feasible to detect the target as a anonymous object and therefore to detect the motion itself not the agent. Later the hand tracks are computed into action attributes for logical inference. In the framework this are "Approach towards", "Recede from" and "Pause at". These short motion descriptors are body part dependent using the results of the human pose estimation in Section 3.1. The mixture tracking approach is explained in more detail in the next subsections. [8]

3.3.1 Mixture Tracking In this subsection x_t is defined as the state of an object in dependence to the time t and the observations up to time t are represented by $y_1^t = (y_1, ..., y_t)$. The searched distribution is the filtering distribution $p(x_t|y_1^t)$. Using Bayesian sequential estimation this filtering distribution can be computed by the two step recursion

$$predict : p(x_t|y_1^{t-1}) = \int D(x_t|x_{t-1})p(x_{t-1}|y_1^{t-1})dx_{t-1} \tag{3.4}$$

$$update \ : p(x_t|y_1^t) = \frac{L(y_t|x_t)p(x_t|y_1^{t-1})}{\int L(y_t|s_t)p(s_t|y_1^{t-1})ds_t} \tag{3.5}$$

as shown in [10]. A model which describes the state evolution is given by $D(x_t|x_{t-1})$. The likelihood of any state is given by $L(y_t|x_t)$ [10]. At this point it is worthwhile to write the filtering distribution as a M-component mixture model. This can be written as

$$p(x_t|y_1^t) = \sum_{m=1}^{M} \pi_{m,t} p_m(x_t|y_1^t) \tag{3.6}$$

with the condition that

$$\sum_{m=1}^{M} \pi_{m,t} = 1 \tag{3.7}$$

as shown in [10]. Thereby $p_m(x_t|y_1^t)$ is the filtering distribution for the m-th component and $\pi_{m,t}$ is the associated component weight depending on time t. This mixture representation can be written as a two-step approach. This leads to the property that it can be updated recursively as the two step approach presented in Equation 3.4 and Equation 3.5. To derive such a two-step approach it is possible to substitute $p_m(x_t|y_1^t)$ from Equation 3.6 with $\int D(x_t|x_{t-1})p(dx_{t-1}|y_1^{t-1})$ from Equation 3.4. As shown in [10] this can be written as

$$\begin{aligned} p(x_t|y_1^{t-1}) &= \sum_{m=1}^{M} \pi_{m,t-1} \int D(x_t|x_{t-1}) p_m(x_{t-1}|y_1^{t-1}) dx_{t-1} \\ &= \sum_{m=1}^{M} \pi_{m,t-1} p_m(x_t|y_1^{t-1}) \end{aligned} \tag{3.8}$$

which is the new prediction distribution. Furthermore, $p_m(x_t|y_1^{t-1})$ is the prediction distribution for the m-th component. [10]

The next aim is to obtain a formula for the update step. This new filtering distribution can be derived by substituting the prediction distribution into Equation 3.5. This step is shown in [10] and is resulting in the following:

$$\begin{aligned} p(x_t|y_1^t) &= \frac{\sum_{m=1}^{M} \pi_{m,t-1} L(y_t|x_t) p_m(x_t|y_1^{t-1})}{\sum_{n=1}^{M} \pi_{n,t-1} \int L(y_t|s_t) p_n(s_t|y_1^{t-1}) ds_t} \\ &= \sum_{m=1}^{M} \left[\frac{\pi_{m,t-1} \int L(y_t|s_t) p_m(s_t|y_1^{t-1}) ds_t}{\sum_{n=1}^{M} \pi_{n,t-1} \int L(y_t|s_t) p_n(s_t|y_1^{t-1}) ds_t} \right] \left[\frac{L(y_t|x_t) p_m(x_t|y_1^{t-1})}{\int L(y_t|s_t) p_m(s_t|y_1^{t-1}) ds_t} \right] \end{aligned} \tag{3.9}$$

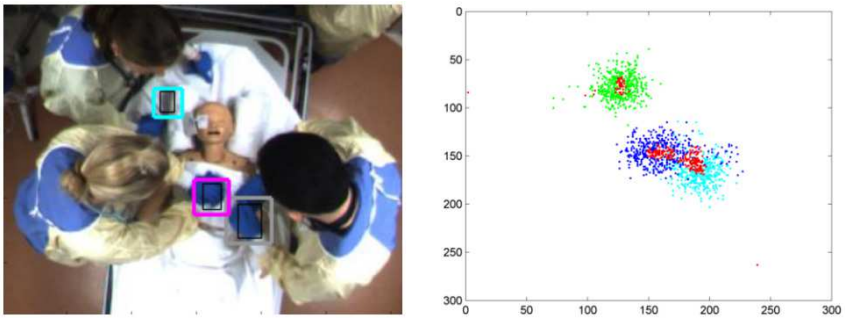


Abb. 3.9: In the left image the detected hands are shown. In the right image one can see the corresponding particles. The red particles are resampled. Images taken from [8]

So after [10] the new filtering distribution for the m-th component is given by

$$p_m(x_t|y_1^t) = \frac{L(y_t|x_t)p_m(x_t|y_1^{t-1})}{\int L(y_t|s_t)p_m(s_t|y_1^{t-1})ds_t} \tag{3.10}$$

which complies to the second term in brackets in Equation 3.9. Through the independence to x_t the first term in brackets of Equation 3.9 can be taken to be the new weight with

$$\pi_{m,t} = \frac{\pi_{m,t-1}\int L(y_t|s_t)p_m(s_t|y_1^{t-1})ds_t}{\sum_{n=1}^{M}\pi_{n,t-1}\int L(y_t|s_t)p_n(s_t|y_1^{t-1})ds_t} \tag{3.11}$$

$$= \frac{\pi_{m,t-1}p_m(y_t|y_1^{t-1})}{\sum_{n=1}^{M}\pi_{n,t-1}p_n(y_t|y_1^{t-1})} \tag{3.12}$$

as shown in [10].

All in all the result is a mixture of individual component filtering distributions like the distribution in Equation 3.6. So the filtering recursion can be computed for each component individually. The only part where components interact is the new component weight. This is an elegant result because the independence in computation makes the algorithm parallelizable. [10]

3.3.2 Particle Approximation Nevertheless the previous described mixture tracking approach can only deal with a small number of use cases. This is caused by the assumption that the dynamic model and the likelihoods have to be known. In practice they are unknown and so approximation techniques are required [10]. In connection to this, a sequential monte carlo method for approximation is presented in this subsection. In the remainder of this section the notation from [10] is adopted.

To approximate the target distribution it can be represented by a particle filter with a weighted set of samples. These weighted samples create an approximation of the target distribution in the next timestep. A particle representation of the mixture filtering distribution for time t is given by $\mathcal{P}_t = \{N, M, \Pi_t, \mathcal{X}_t, \mathcal{W}_t, \mathcal{C}_t\}$. M is the number of mixture components like in Equation 3.6 and N is the number of particles. Moreover a weight $\pi_{m,t}$ is assigned to each mixture component m so $\Pi_t = \{\pi_{m,t}\}_{m=1}^{M}$. In addition the particles are given by $\mathcal{X}_t = \{x_t^{(i)}\}_{i=1}^{N}$ and each of them has a weight $w_t^{(i)}$ and $\mathcal{W}_t = \{w_t^{(i)}\}_{i=1}^{N}$. At last a component is identified by a number $c_t^{(i)} \in \{1, ..., M\}$ to assign a particle i to a specific mixture component m so that $\mathcal{C}_t = \{c_t^{(i)}\}_{i=1}^{N}$. As an extension $\mathcal{I}_m = \{i \in \{1, ..., M\} : c_t^{(i)} = m\}$ is defined as a set of indices of the particles which belong to the component m.

As shown in [10] the particle representation of the mixture filtering distribution $\overline{p}$ can be written in the form

$$\overline{p}(x_t|y^t) = \sum_{m=1}^{M}\pi_{m,t}\sum_{i\in\mathcal{I}_m} w_t^{(i)}\delta_{x_t^{(i)}}(x_t) \tag{3.13}$$

where $\delta_{x_t^{(i)}}$ comply with the Dirac delta measure with mass at $x_t^{(i)}$. A precondition in this formula is that

$$\sum_{m=1}^{M}\pi_{m,t} = 1, \sum_{i\in\mathcal{I}_m} w_t^{(i)} = 1 \tag{3.14}$$

is applied to the weights. Figure 3.9 shows this in a visualization with three components for three hands. Each of the components is approximated by multiple particles. All particles which belong to a certain component are drawn in the same color. The red particles are described at a later position of this Section.

The next goal is to compute a new particle set $\mathcal{P}_t$ that is a sample set of $p(x_t|y_1^t)$. The set $\mathcal{P}_{t-1}$ which is approximately distributed to $p(x_{t-1}|y_1^{t-1})$ is already given. Due to the fact that the mixture components in the general mixture tracking recursion interact only in the weight (see Section 3.3.1) all particles evolve independently. In connection to this, all mixture components in this case evolve also independently. A weighted sample set from $p_m(x_{t-1}|y_1^{t-1})$ is given by $\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\}$. New samples are generated from a proposal distribution q. This proposal distribution can for example be dependent on the old state x_{t-1} and the new measurement y_t. To create a properly weighted sample set for the new particles the weights are set to

$$w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j \in \mathcal{I}_m} \tilde{w}_t^{(j)}} \tag{3.15}$$

with

$$\tilde{w}_t^{(i)} = \frac{w_t^{(i)} L(y_t|x_t^{(i)}) D(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|w_{t-1}^{(i)}, y_t)} \tag{3.16}$$

The resulting new sample set is distributed to $p_m(x_t|y_1^t)$. [10]

In a last step new mixture weights are obtained. This can be achieved by using Monte Carlo estimation. In detail, the likelihood for the m-th component is given by $p_m(y_t|y_1^{t-1}) = \sum_{i \in \mathcal{I}_m} \tilde{w}_t^{(i)}$ as in full length shown in [10]. Substituting this into Equation 3.12 leads to

$$\pi_{m,t} \approx \frac{\pi_{m,t-1} \tilde{w}_{m,t}}{\sum_{n=1}^{M} \pi_{n,t-1} \tilde{w}_{n,t}} \tag{3.17}$$

with

$$\tilde{w}_{m,t} = \sum_{i \in \mathcal{I}_m} \tilde{w}_t^{(i)} \tag{3.18}$$

A disadvantage of the approach is that the forming of incorrect particle cluster pairings appears [8]. This effect leads to an incorrect tracking of the hand targets. So it is recommended to resample particles in continuous time intervals to avoid this problem [8]. In Figure 3.9 the red particles are resampled. As one can see the clusters have a higher density than clusters without the resampling step. An advantage of the given approach is that it is possible to resample each mixture component independently due to the independency of the components [10].

3.3.3 Mixture Computation From the last subsection results a mixture representation for the filtering distribution. In practice and especially in trauma resuscitation targets disappear and appear and the exact number is rarely known before. This is caused due to the variable number of persons in the medical team. So the number of modes in the target distribution is also rarely known. A solution can be to recompute the mixture representations periodically. For example components which overlap are merged and components which are diffuse are splitted. An approach to realize this was shown by [10] and is presented in the following.

First a spatial reclustering function is given by $(\mathcal{C}_t', \mathcal{M}') = \mathcal{F}(\mathcal{X}_t, \mathcal{C}_t, \mathcal{M})$. This function takes the particles, the identifier of each particle and the number of components. As mentioned before a particle is assigned to a certain component by the particle identifier. The result is a new mixture representation in which components are merged or splitted if this was necessary. The function $\mathcal{F}$ can be realized with the K-means algorithm by executing this algorithm iteratively. But after the reclustering step mixture weights and particle weights change. So it is required to compute new mixture and new particle weights. More formally Π_t and $\mathcal{W}_t$ have to be recomputed so that a new mixture approximation is given by $\mathcal{P}_t' = \{N, M', \Pi_t', \mathcal{X}_t, \mathcal{W}_t', \mathcal{C}_t'\}$. The resulting mixture filtering distribution can be written as

$$\overline{p}(x_t|y^t) = \sum_{m=1}^{M'} \pi_{m,t}' \sum_{i \in \mathcal{I}_m} w_t'^{(i)} \delta_{x_t^{(i)}}(x_t) \tag{3.19}$$

where the new weights are given by

$$\pi_{m,t}' = \sum_{i \in \mathcal{I}_m'} \pi_{c_t^{(i)},t} w_t^{(i)} \tag{3.20}$$

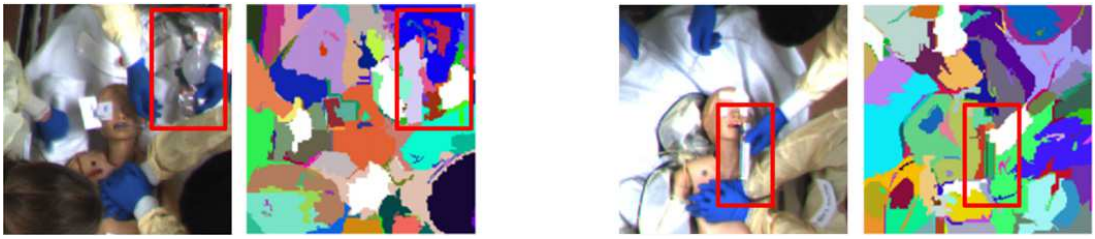


Abb. 3.10: On the left side an oxygen mask is detected respectively on the right side a laryngoscope. The right image of the pairs shows the belonging segmentation. Images in figure taken from [8]

and

$$w_t'^{(i)} = \frac{\pi_{c_t^{(i)},t} w_t^{(i)}}{\pi'_{c_t^{(i)},t}} \tag{3.21}$$

as shown in [10]. By that $\mathcal{P}_t$ can be substituted by $\mathcal{P}'_t$ because it represents the same distribution. It is notable that the particles $\mathcal{X}_t$ are not changed by computing the new representation. [10]

3.4 Device Detection

Using hand tracking, activities which are based on hand motions can be recognized. But some activities include similar hand motions for example a ventilation and an intubation. Both are performed in the head region and both are performed by hands. The only way to distinguish these activities is to detect the involved device. In trauma resuscitation these are devices like an oxygen mask, a laryngoscope or a stethoscope. An important advantage of the activity recognition framework is that it uses a fixed wide angle camera. Thereby all devices have a fixed scale and can only appear rotated or tilted. Nevertheless the appearance can vary depending on the device form. In an activity recognition framework it can be beneficial to use different approaches. So two approaches are used, one for orientation-dependent forms and another one for tube-based forms. [8]

Orientation-dependent forms are for example the form of oxygen mask or of the laryngoscope [8]. These devices have a great appearance variance depending on their orientation. In order to achieve a correct recognition the image is segmented into similar regions based on color and texture information [8]. The algorithm which is used by the framework is described by Felzenszwalb et. al. in [6]. In this algorithm the evidence for separating two regions is measured by a graph-based representation of an image. The result of this algorithm is a segmentation into small homogeneous regions [6]. The right images of the pairs in Figure 3.10 shows an example for these segmentations. To assign such a region to a certain device Scale Invariant Feature Transform (SIFT) features are used [8]. After extracting SIFT features it is possible to compare them to a training set to get a reliable matching [11].

The second approach is used for tube-based forms like the form of the stethoscope tube, the electrocardiograph leads or the intravenous access tubing. The algorithm which is used by the device detection was described by Frangi et. al. in [2]. The basic idea is that the filter assumes pixel intensities to be distributed as a Gaussian on tubular structures. Then one can compute the ratio of the principle axes after Eigen decomposition which can be interpreted as the strength of tubularness. [8]

Another improvement can be to involve the hand position from Section 3.3. This can be helpful when computing an action attribute. For example active devices have to be in the grasping radius of a hand. A combination of both information can lead to a higher precision. [8]

3.5 Activity Grammar

At this point several subactivities are detected and transferred into action attributes. The last step is a logical combination of all observations so that an inference can be made. Therefore the logical part of the activity recognition framework is especially important because it leads to the performed medical procedure. But before logical inference is possible all procedures have to be modeled in an appropriate construct. This construct is found in the Activity Grammar (AG) which is a set of declarative semantics. This section describes the AG presented in [8] and its properties. The logical inference is performed by a Markov Logic Network (MLN) which is described in the next section.

Tab. 2: An example of a set of rules. p, t, a, o denote patient's body part, time, procedure identity and medical device. B is the beginning state, D and E the during and end state. Figure taken from [8]

#	Formula	Weight
1	$Approach(p,t) \wedge Located(p,a) \Rightarrow Act(a,t,B)$	$W/2$
2	$Recede(p,t) \wedge Located(p,a) \Rightarrow Act(a,t,E)$	$W/4$
3	$Observe(o,t) \wedge Use(o,a) \Rightarrow Act(a,t,D)$	W
4	$Act(a,t_1,B) \wedge Act(a,t_2,E) \wedge (t_1 < t_2) \wedge (t_2 > t > t_1) \Rightarrow Act(a,t,D)$	$W/2$
5	$Act(a,t_1,B) \wedge Act(a,t_2,E) \wedge (t_1 > t_2) \wedge (t_1 > t > t_2) \Rightarrow !Act(a,t,D)$	W
6	$Act(a,t_1,D) \Rightarrow Act(a,succ(t_1),D)$	$W/4$
7	$Change(p,t_1) \wedge Located(p,a) \wedge Follows(t_1,t_2) \Rightarrow Act(a,t_2,B) \vee Act(a,t_2,E)$	$W/4$
8	$Act(a_1,t,D) \wedge Located(p,a_1) \Rightarrow (a_1 = a_2) \vee !(Act(a_2,t,D) \wedge Located(p,a_2))$	W
9	$PauseBegin(Chest,t) \Rightarrow Act(Compression,t,B)$	$W/2$
10	$PauseEnd(Chest,t) \Rightarrow Act(Compression,t,E)$	$W/2$
11	$PauseBegin(p,t) \wedge !(p = Chest) \Rightarrow Act(CheckPulse,t,B)$	$W/2$
12	$PauseEnd(p,t) \wedge !(p = Chest) \Rightarrow Act(CheckPulse,t,E)$	$W/2$
13	$Change(Head,t) \wedge Change(Chest,t) \Rightarrow Act(Roll,t,B) \vee Act(Roll,t,E)$	$W/4$
14	$Earpiece(t_1) \wedge Approach(Chest,t_2) \wedge Follows(t_1,t_2) \Rightarrow Act(Stet,t_2,B)$	$W/4$
15	$Skin(t_1) \wedge {}^{!}Skin(t_2) \wedge Follows(t_1,t_2) \Rightarrow Act(Cover,t_1,D)$	W
16	$\ni\ t\ Act(Roll,t,D) \Rightarrow Phase(t,3)$	W
17	$\ni\ t\ Act(Uncover,t,D) \Rightarrow Phase(t,1)$	W
18	$Follows(t,t)$	∞
19	$Follows(t,succ(t))$	∞

Declarative semantics of the AG can also be called rules. Each of these rules represents an activity in an axiomatical way and is noted in First-Order (FO) logic. This allows for example negations or existential quantifiers and gives a powerful opportunity to describe things. A rule can be dependent on a temporal attribute in the AG. Moreover each rule has an associated weight to soften the rule. So rules which apply often but have a low confidence can be handled more carefully to prevent wrong inferences. [8]

Table 2 shows an example of an AG. The weight is manually set relative to a number W. The action attributes are represented by functions like *Approach* for hand motions or Use for usage of devices. These functions are dependent on objects which are body parts p, devices o, procedure identities a or the time t. It is notable that there is only a finite set of attributes for objects. A special case is the *Act* attribute which is used to make a query for procedure identities [8].

To get most beneficial results rules should be accurate, comprehensible and be influenced by medical domain knowledge. It is recommended to limit the number of rules to a minimum and to set up rules modular. A small set of variables can reduce complexity. A last point is the consistency of rules which is important because a contradiction can make an inference impossible. The rules which are used by the framework can be categorized with several principal features which are described as follows. [8]

The first principal feature is the generalizability of rules. On the one hand a rule can be generally applicable on the other hand it can be application-specific. Generic rules can make the AG generalizable and also customizable for extensions. Generalized rules can make it possible to use the rules as some kind of a module to describe other activities. Nonetheless application-specific rules are sometimes indispensable like the rule "Check Pulse" in the AG of the framework. [8]

Another principal feature is the temporal dynamic of a procedure which can be modeled by the ruleset. It allows the detection of interleaved and simultaneous activities which is a valuable property in the trauma resuscitation scenario. Nevertheless, the implementation can lead to a problem because there are $O(T^2)$ possible time intervals for each activity for a video sequence of T frames. It is given by the number of the possible beginning times multiplied with the number of the end times. To reduce this effort for the framework, time intervals are modeled as time states for each procedure. In detail, latent variables dependent to action attributes are introduced to model these time states. This idea reduces the inference time of complex procedures and makes it possible to render a model linear in time complexity. [8]

A further principal feature is given by the type of the activity. Some activities are static and occur only once like an intubation. Others are dynamic and are based on spatially and temporally associations like listening to breath sounds. A last type is represented by casual activities which are based on scene level changes. This appears for example when the patient is covered or uncovered. [8]

Another improvement for the AG is the adaption of rules to ATLS [8]. As described in the introduction in Section 1 ATLS is an efficient way to structure the sequence of steps in trauma resuscitation. ATLS

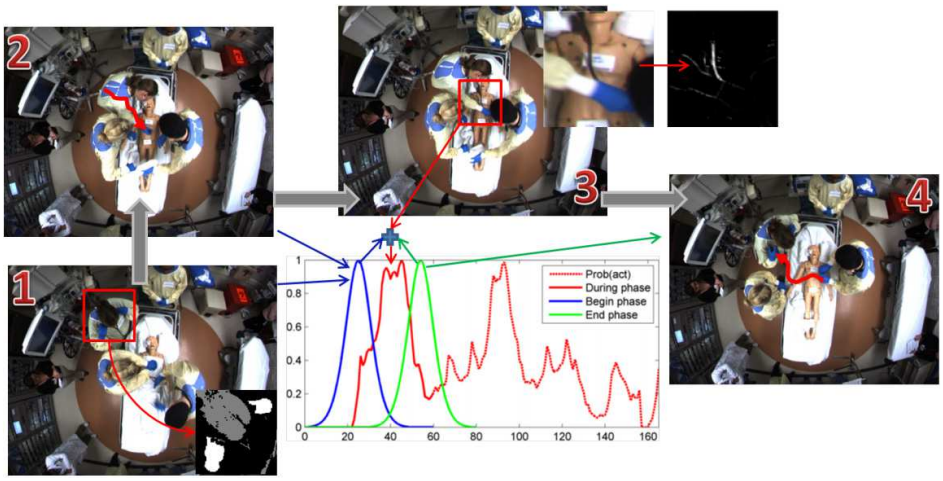


Abb. 3.11: Inference of a medical procedure with temporal associations. In this case "Listen to breath sounds" is detected. Figure taken from [8]

describes an approximate sequence of procedures which should be executed during trauma resuscitation. Based on this sequence, the entire process is splitted up into three phases. In the first phase life-threatening injuries are treated [4]. In the second phase the patient is analyzed in a physical examination and in the third phase missed injuries and related problems are treated [4]. Each procedure gets an assigned prior probability which represents the certainty that it belongs to a specific phase [8]. So the knowledge that the process is in a certain phase makes it feasible to make distinguished inferences even when the detected activities are similar to other procedures.

Figure 3.11 shows a visualization of an inference to explain the process. In this example it is assumed that the activity detection is not used in real time. The graph in the middle of the image shows the probability for a beginning (blue), end (green) and during phase (red) and the probability for an action (red dotted) in respect to the time. Image one and two in this figure show the use of a medical device and a hand motion approaching towards the patient. These two actions are identified as a begin phase so the probability value is close to one in the graph. The receding hand motion in image four in this figure is identified as an end phase. In this case the probability value for the end phase is close to one. As one can see by the dotted red line there are several candidates for a during phase due to several high probability values for an action. By using the information of the begin and the end phase only one during phase is identified as a correct during phase. The next section deals with the computation of these visualized probabilities.

3.6 Markov Logic Networks

As described in the previous subsection a ruleset is given by the AG. Based on the information of this ruleset and the observations, logical inference is made by computing the probability of an action. By that the performed medical procedures can be determined. A problem is that inference in FO-logic is semi-decidable. This means that an inference can be made when a formula holds but an inference is undecidable if it holds not. This leads to the fact that when an set of observations violates one formula the assigned activity becomes impossible. By that, knowledge bases are often restricted to special, decidable formulas. Moreover in the real world observations have a different degree of confidence which is unrepresentable in pure FO-logic. By that pure FO-logic has a limited practical applicability. [18]

A possible solution for this problem is given by Markov Logic Networks (MLN). MLN's can handle uncertainties and can make logical inferences. In detail a MLN is a statistical relational framework which converts the given FO-logic into a graphical model. This model can be evaluated in a probabilistic way for parameter estimation and inference. So a MLN is comparable to a template for constructing a Markov Network (MN). [8]

In FO-logic each formula connects different variables and constants, predicates and functions by different logical symbols. Variables and constants are called *objects*. Further *predicates* are attributes of objects and *functions* are connections between objects. All objects across the formulas are connected to each other so that the complete MLN can be viewed as a single formula. [8]

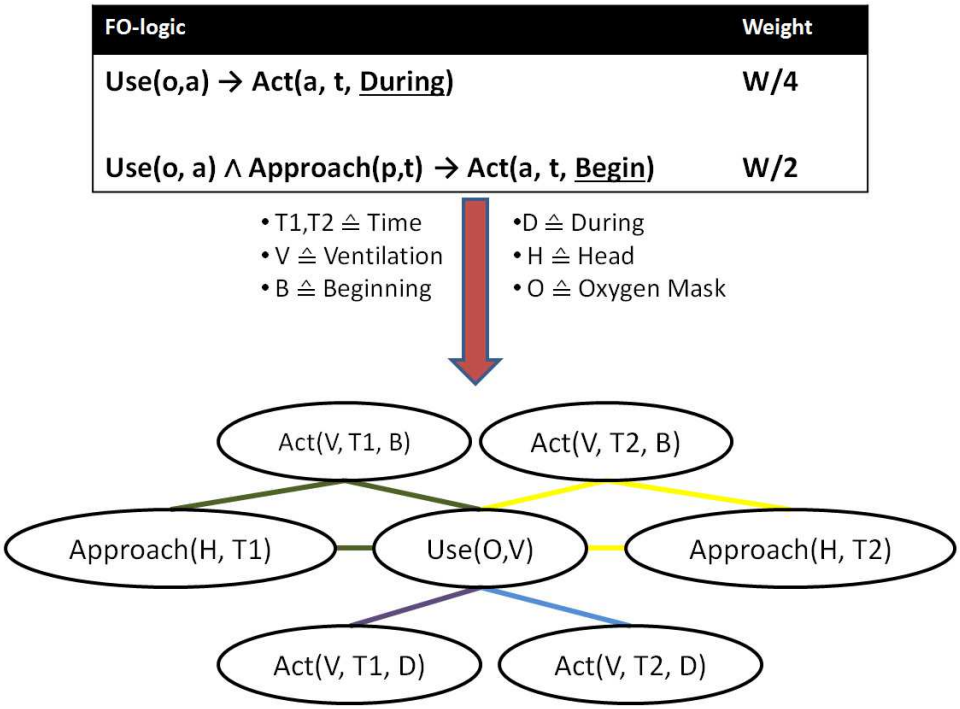


Abb. 3.12: Simplified example of a transformation from AG to MN. At the top one can see an AG represented as a table. Attributes are substituted into formulas and afterwards a MN is created. Each clique has its own edge color in the resulting network.

By these definitions a MN can be constructed which is also visualized in Figure 3.12. All object variables in predicates are replaced by constants which are based on the observations. A node is created for each possible grounding. If two ground predicates appear in one of these formula groundings together an edge is added between the nodes. The representation of each formula forms a clique in the resulting MN. The given weight w_i for each formula is therefore also assigned to each clique. [8]

After the construction of the graph inference can be made. In the following L is an MLN and C is a set of constants. $M_{L,C}$ is the MN for a given L and a given C. The probability for a formula F_1 and a given formula F_2 which holds can be computed by

$$P(F_1|F_2, L, C) = P(F_1|F_2, M_{L,C}) = \frac{\sum_{x \in \mathfrak{X}_{F_1} \cap \mathfrak{X}_{F_2}} P(X = x|M_{L,C})}{\sum_{x \in \mathfrak{X}_{F_2}} P(X = x|M_{L,C})} \tag{3.22}$$

as shown in [18]. $\mathfrak{X}_F$ are multiple sets of nodes common to formula F. By that the enumerator x is a set of nodes. In the following $n_i(x)$ is the number of true groundings of a formula F_i in x. Furthermore n is the overall number of formulas. So the probability distribution of x used in Equation 3.22 is given by

$$P(X = x|M_{L,C}) = \frac{1}{Z} exp(\sum_{i=1}^{n} w_i n_i(x)) \tag{3.23}$$

also shown in [18]. In addition w_i is the weight which is specified by the AG before. Z is used for normalization and is set to the total weight of all involved formulas. A possible computation method to make inference is given by Gibbs sampling which is also used by the framework [8].

4 Results

The intention of this section is to summarize and to compare the results of the presented activity recognition framework and of the involved core components. A final discussion of these results is given in the next section. First the general evaluation conditions are stated. Then the results are described and in the end a comparison is drawn.

Tab. 3: Relative amount of false detections in percent of hand motion activities taken from [8]

	Approach	Recede	Pause
Head	6	11	3
Chest	19	29	27
Limbs	n.a.	n.a.	37

Tab. 4: The first line shows the relative amount of wrong device detections or missed devices per frame in percent. In the second line one can see the absolute amount of wrong detections (first number) per device usage (second number). In the last line one can see the relative amount per usage. Data is taken from [8]

	OxyMask	L.scope	Steth. Tube
Per frame	11	24	37
Per usage (absolute)	1 / 24	3 / 13	0 / 39
Per usage (relative)	4	23	0

For training and evaluation a set of videos is used. These videos are created in a trauma center in the US and show a trauma resuscitation with a complete team of clinicians. The frame-rate, camera type and resolution of video source material are not specified in the evaluation. The patient is simulated by a manikin and the camera is ceiling mounted over the manikin. Feature information is aggregated over twenty frames to reduce errors in extraction. The aggregation is not described in detail in the evaluation paper. To evaluate the results all videos have been manually analyzed to determine the beginning and the ending of each procedure. Further the construction of the framework requires a training phase due to being data-driven. Learning models for hand color and object texture are build. These are especially used by hand tracking and the patient detection. More specific image information from five different videos are taken to train the framework. Assuming the maximum number of simultaneously involved clinicians is eight the number of targets for the mixture particle filter is set to 16. Furthermore, 150 particles are used for each hand blob. The maximum weight of a formula is set to $W = 4.0$ and the other weights are scaled accordingly. The following tests except the human pose estimation are based on ten video simulations which are an average of 12 minutes long. The device detection evaluation is based on 1000 randomly chosen frames containing the device. [8]

Since it was not successful to find results of the patient detection evaluation in the medical domain this work shows an evaluation of the human pose estimation framework in general. The pose estimation technique is applied to different shots from different episodes of "Buffy the vampire slayer". In detail 243 frames of a video are analyzed where the upper body is found correctly with an upper body detector. This video material is suitable because it contains a lot of clutter, persons of different sizes, low contrast and persons with all kinds of clothing. Moreover the appearance of a person is unknown a priori. It is likely that in this video material it is more complicated to detect a correct pose than in the trauma resuscitation scenario. This is partly because the viewpoint which is not frontal in the used video material but it is in the trauma resuscitation scenario. In addition poses in which arms are folded over the body are uncommon. Despite all these difficulties the human pose estimation framework correctly estimates 56% of the $243 * 6 = 1458$ body part positions in this material. [20]

The hand tracking experiment is performed using the activity recognition framework. Nevertheless the detailed evaluation method is not explained in the literature. According to [8], the overall detection precision of hand tracking is about 98%. In the remaining two percent missed targets or identity exchanges are leading to false detections. This leads to an error rate of 19% of hand motion activity detection. Table 3 shows that false detection rates vary depending on the body location. Reasons for this dissimilar distribution of false detections is for example the extent how much a region is frequented by other objects. For example activities in the head area are rare and are influenced by less occlusion. So activities are detected more accurately in the head area as one can see in Table 3. The detection of receding movements is disturbed by identity switches close to the patient's body. In according to the fact that limbs are often covered by inactive hands the detection rate of false detections is higher. [8]

Table 4 shows the results of the device detection experiment. For evaluation material 1000 randomly chosen frames per device are used. All of the chosen frames contain the used device. As one can see by the table the devices are not detected correctly in several frames. One of the reasons for this is occlusion or

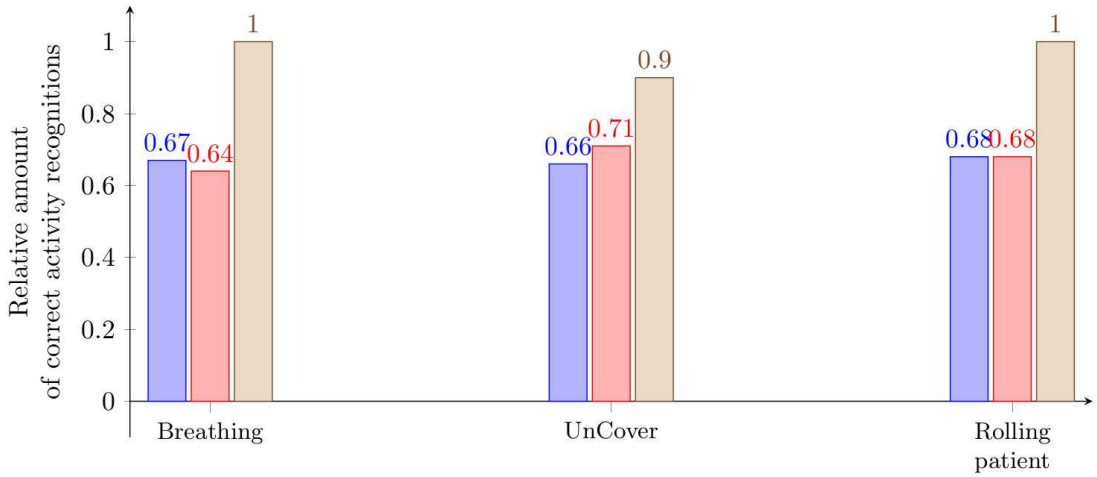


Abb. 3.13: Relative amount of correct activity recognitions with the BoW-1 method (blue), the BoW-2 method (red) and the presented activity recognition framework (brown). Data given by [8]

low contrast. Nevertheless the device usage activity itself is detected in most of the cases over a complete video. One can see this in the second line of Table 4 which shows the number of usages at the right side of each cell and the number of incorrect detections at the left side. For example the stethoscope is detected correctly in every case in the testing material. [8]

The last part of this section shows a comparison to other activity detection approaches. In this work the framework is compared to two bag-of-words approaches. The experiments to generate the data are applied by [8]. Bag-of-words based detection is a commonly used method for event detection in videos. In this method general image features are extracted and then quantized into a vocabulary of words. Afterwards words are encoded into a histogram vector. Vectors of several frames and different activities are trained by a Support Vector Machine (SVM) in a learning phase. Activity identity is labeled manually for each vector in the training set. After the learning phase is finished a classifier can be computed which assigns a arbitrary vector of the vector space to a class label. By this method the procedure identity can be predicted. [7]

In the executed evaluation the SVM is trained on image keyframes where procedure identities are labeled. SIFT features are extracted and quantized into a vocabulary of 500 words. After that each keyframe is encoded into a 500-word histogram vector. The used histogram intersection kernel of the SVM has a penalty (C) of 1. To cover different types of approaches two training settings are used. The first is the one-versus-all binary approach (later called BoW-1). In this connection a frame of a class is used as a positive example and all other as negative examples to build several binary classifiers [7]. The second type is an multi-class SVM (later called BoW-2). For testing the leave-one-out mechanism is used where all videos are used to train the SVM except one. Furthermore the chosen recall value which is used as an operation point is 75%.

Figure 3.13 shows a comparison of the described methods including the approach with the MLN based framework. First the diagram is restricted to rough and simple activities which influence a big region of the scene. It is notable that the total number of occurrences is not provided by the data source.

As one can see the advantages of the MLN-based approach in comparison to the BoW-based approaches are already visible. This is because the shown activities consist only of a small number of subactivities. For example uncovering the patient is just a single step activity with a big appearance change. But in the medical use case activities are often more complex. Figure 3.14 shows more complex and fine grained activities and the corresponding detection results.

The difference between the approaches is greater when fine grained activities are tested. Possible reasons for this are that the compared bag-of-words approach do not take into account temporal associations. Another weak point are simultaneous activities which are hard to distinguish and also affect the training of the SVM. This is because vectors are not assigned to a specific activity when two activities occur at the same time. So the used BoW-based approaches can be described as static in making inference. In contrast to this the presented activity framework uses a modular structure. Different activities are independent

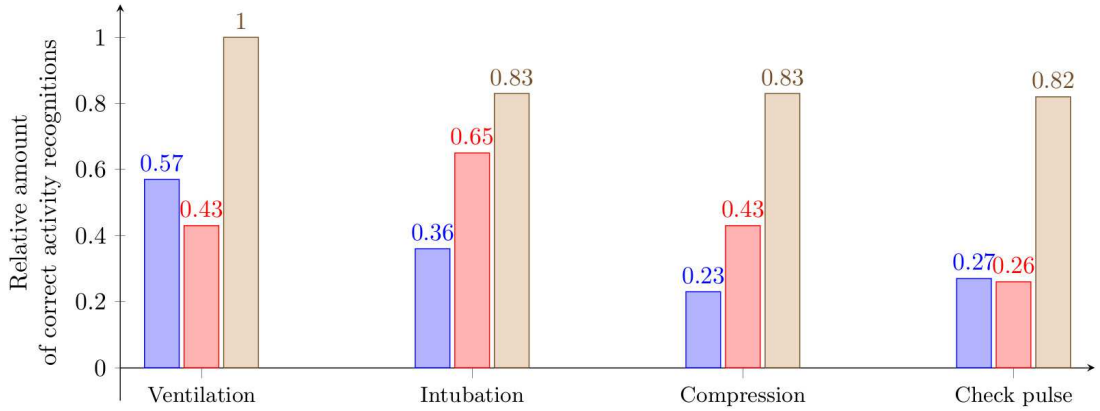


Abb. 3.14: Relative amount of correct activity recognitions with the BoW-1 method (blue), the BoW-2 method (red) and the presented activity recognition framework (brown). Activities are particularly fine grained. Data given by [8]

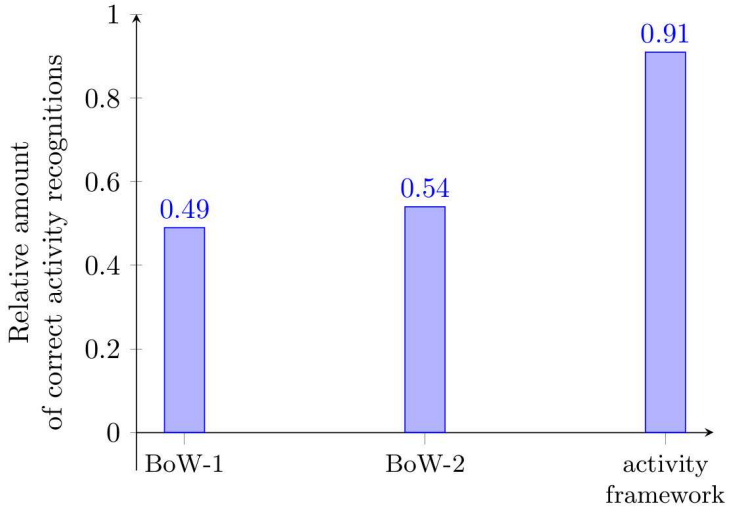


Abb. 3.15: Relative amount of total correct activity recognitions in average. Procedures: Ventilation, Intubation, Breathing, Compression, Checkpulse, UnCover, Rolling patient. Number of occurrences of each procedure is unknown. Data given by [8]

from each other and by this the dynamic of an activity cannot strongly influence inference result. Besides temporal reasoning is possible to make a decision.

All in all the overall precision of correctly detected activities is 91% (Figure 3.15) in the used testing material. In addition the processing time of each video is less than two minutes [8]. These readings are made with MATLAB [12] for feature computation and with ALCHEMY [14] for inference [8]. The performance experiments are executed without any code optimization [8]. It is supposed that real time inference is theoretically possible because the computation is six times faster than real time. It is conceivable that the performance can be improved in further implementations because of the missing optimization. Nevertheless the authors of the evaluation give no precise performance measurements. This means that the exact computation time, the variance of computation time and the setup of the system including the hardware are unknown. Therefore a detailed analysis is only possible in a restricted manner.

5 Discussion

In general the presented framework is better suited for fine-grained activities which was shown in the result section. This is accomplished by an inclusion of multiple features, involvement of temporal dynamics and an inference based on multiple observations. The framework has a modular structure which allows making

a distinguished detection of subactivities by specialized components. For example the hand tracking component can handle also a multi-agent situation and the human pose estimation detects the patient also in highly cluttered and noisy images. This gives rise to the ability of splitting up complex activities into several less complex subactivities. Another advantage of this structure is that components can be improved or replaced by other components. Moreover the modularity makes the framework expandable to auxiliary detectors and further components. So an extension by RFID, audio or other sensors is conceivable. The logical inference performed by a MLN is flexible because it is based on a FO-logic knowledge base and can handle uncertainties by weights. These properties also make the framework generalizable and open for other domains. So for example it can be possible to track and transcribe activities on a production line with a domain specific ruleset. Furthermore the presented framework is the first which implements the approach to the trauma resuscitation scenario [8].

But anyhow, there are also some weak points. A first point is that there is deficient information about the evaluation environment in the appropriate paper [8]. It would be helpful to know more details about the testing material (frame-rate, number of activity occurrences) to draw a comparison to other approaches. Especially, it would be interesting to draw a comparison to Multi-Person activities because they are able to deal with multiple agents and maybe they perform better than the used bag-of-words approaches. In addition the testing material uses only a manikin as a patient which is not able to change its pose. The framework assumes that the pose of the patient does not change after pose estimation step [8]. So there is a possible impreciseness in the results due to the simplified testing material.

But there are also disadvantages in the structure of the framework. One of them is that activities can be detected only as well as the knowledge base is defined. Nevertheless a possible solution can be to learn rules but this requires a suitable amount of training data. In some cases it may be not possible to generate this data. Another point is the complexity of the system and the error propagation. The framework uses many different components where some of them have to interact or have to be executed at the same time. The failure of initial components lets all other components fail which are initialized with their result. In case of the presented framework setup the whole system is dependent on the success of the patient detection. There is no redundancy or fallback method which makes the system unsusceptible for failures.

Furthermore, a video-based approach brings some limitations with it. So it is possibly not robust due to abrupt color changes. For example it is possible that a bleeding occurs and the gloves are covered with blood. This can especially occur in the medical domain and leads to a change of the gloves color. This can lead to a failure of the hand detection. To avoid this point of failure it is possible to improve the hand detection by additional sensors like depth sensor or multiple cameras to handle abrupt color changes. But such a solution increases the system complexity further due to the appending of components.

At the end it is interesting to discuss the precision and the performance of the system because this is an important factor when it comes to applicability in the medical domain. The overall rate of correctly detected activities is indeed higher than the rate of other presented approaches. Nonetheless this presented accuracy is too low for a decision support system which is used productively. In the medical domain reliability and robustness are important factors. Also low error rates are problematical because the special situation does not allow wrong detections or incorrect decisions. The next point is the performance of the presented system. Although the performance measurement in the given evaluation [8] is not described in detail (unknown hardware specification, no details about testing material) it is reasonable that processing time is not exorbitantly high. This is because the processing time of a video is six times higher than real time without an optimized system. This leads to the conclusion that real time recognition is presumably possible.

All in all there is the opportunity to use the framework to realize or improve computer assisted-decision support in the medical domain. This system can be applied without causing any additional user interaction and can be integrated in a non-intrusive way. But a prerequisite is that the precision is higher than presented and the performance allows real time recognition.

6 Conclusion

This work dealt with activity recognition and its realization for the trauma resuscitation domain. The given situation was described and the general difficulties and challenges for a recognition system in this domain were discussed. Moreover an activity recognition framework was shown which achieves better results in the trauma resuscitation scenario than the two compare BoW approaches. This framework and all of the components used in the framework were described. Especially the core components and the functionalities were explained in more detail. Moreover the benefits and disadvantages of this video-based

system were reconsidered. The applicability for the medical domain was analyzed. Besides other areas of application were mentioned. As a result it was shown that the framework performs already well in the trauma resuscitation scenario but is not applicable for a productive usage at this state.

Literaturverzeichnis

[1] N. D. Freitas A. Doucet and N. Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[2] K. L. Vincken A. F. Frangi, W. J. Niessen and M. A. Viergever. Multiscale vessel enhancement filtering. In *Medical Image Computing and Computer-Assisted Interventation*, pages 130–137. Springer, 1998.

[3] A. Farhadi A. Fathi and J. M. Rehg. Understanding egocentric activities. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 407–414. IEEE, 2011.

[4] C.K. Lackner W. Mutschler B. Bouillon, K.G. Kanz and J. Sturm. Die bedeutung des advanced trauma life support(atls) im schockraum. *Der Unfallchirurg*, 107(10):844–850, 2004.

[5] V. Kolmogorov C. Rother and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. 23(3):309–314, 2004.

[6] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[7] C. Hsu and C. Lin. A comparison of methods for multiclass support vector machines. *Neural Networks, IEEE Transactions on*, 13(2):415–425, 2002.

[8] A. Elgammal I. Chakraborty and R. S. Burd. Video based activity recognition in trauma resuscitation. pages 1–8, 2013.

[9] B. Jiang A. Mamishev M. Philipose A. D. Rea S. Roy J. R. Smith, K. P. Fishkin and K. Sundara-Rajan. Rfid-based techniques for human-activity detection, sep 2005.

[10] A. Doucet J. Vermaak and P. Pérez. Maintaining multimodality through mixture tracking. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1110–1116. IEEE, 2003.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[12] MathWorks. Matlab.

[13] V. I. Morariu and L.S. Davis. Multi-agent event recognition in structured scenarios. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3289–3296. IEEE, 2011.

[14] University of Washington. Alchemy: Open source ai.

[15] R. Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[16] L. K. McIntyre H. M. Foy R. L. Gruen, G. J. Jurkovich and R. V. Maier. Patterns of errors contributing to trauma mortality, December 2006.

[17] D. Ramanan. Learning to parse images of articulated bodies. pages 1129–1136, 2006.

[18] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[19] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.

[20] M.l Marin-Jimenez V. Ferrari and A. Zisserman. Progressive search space reduction for human pose estimation. pages 1–8, 2008.

Multiple Foreground Co-segmentation for Medical Images

Jens Böttcher

Zusammenfassung

In this report we will give an insight into what cosegmentation is and how this problem is addressed by two different approaches. Afterwards we will express critique on those and discuss if they are suitable for a medical application.

Keywords: medical, cosegmentation, multiple foreground cosegmentation, semi-supervised learning, connectivity constraints

1 Introduction

As the cost of memory decreases the amount of data increases. The cost for storing large amount of data (e.g. images) became less and less over the last decades. And because of the quantity of images it is harder to analyse them manually, but, on the other hand, the data and information found in those images could lead to more accuracy in the analysis.

See Abb. 4.1 as examples for two small image sets. The first row is a set of images displaying children and an apple bucket. With only three images in a set it would not take much effort to locate each person and object, but if we increase the number of images per set, this task becomes cumbersome.

Image segmentation plays an important role in medical image processing with applications like quantification of tissue volumes, diagnosis, localization of pathology and more [1], and therefore optimizing the task of segmentation is important. The second row in Abb. 4.1 are samples for x-rays of a human chest, which are taken from the same person. A pathologist could analyse this small set of images for diseases easily in foreseeable time. But again, if we increase the number of images per set, this task becomes cumbersome.

Basics

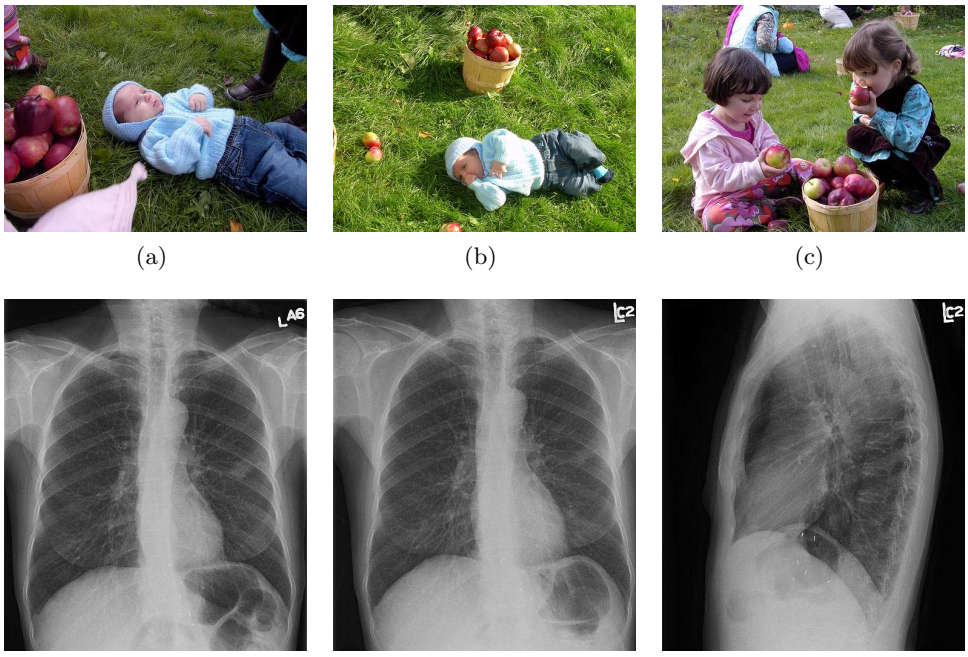
An image I is a set of n pixels. We're interested in K objects, which are denoted as set of foregrounds $\mathcal{F} = \{\mathcal{F}^1, \cdots, \mathcal{F}^K\}$. The task is to locate each given foreground in an image or a set of M images $\mathcal{I} = \{I_1, \cdots, I_M\}$. After locating the foreground(s), one can examine the results for further analysis manually or by apply other image processing applications.

Segmentation is the task of partitioning an image into subsets of pixels which are disjoint and every subset is connected. We do call those subsets segments. More formally:

Definition 1 (Segmentation)
We denote S as a set of all segments of an image I, with for all segments $s_i \in S : s_i \subseteq I$, $I = \bigcup_{i=1}^{m} s_i$, when $|s_i| = m$, and also $s_i \cap s_j = \emptyset$, with $i \neq j$. Lastly, each s_i is connected.

An example for a segmented image can be seen in Abb. 4.2(c). A segmentation technique segmented this image according to predefined settings (e.g. color, texture). The result is a set of non-overlapping segments (displayed with different color-overlays).

In practice, the goal of segmentation is to find segments which do represent all foregrounds in an images. [1, 2, 3, 4]



(a) (b) (c)

(d) (e) (f)

Abb. 4.1: Two examples for small image sets. Images (a) to (c) are a subset of a apple picking image set and images taken from the FlickrMFC dataset [2], (d) to (e) is a set of x-rays of a human chest taken from the The Cancer Imaging Archive

If we have a set of images at hand we could use information we collected on segmenting one image to segment another, because we assume that there is re-occurring content within this set. Even further, we could jointly segment multiple images and share information among the processes. This approach is called cosegmenation.

Most existing approaches for cosegmentation assume the simple but less realistic case where each image of the image set contains the same subset of foregrounds. [3, 4, 5, 6, 7, 8].

This setting might be appropriate for some use cases, but is clearly limiting the approaches to a certain degree because a set of images with a different subset of foregrounds on each image has first to be divided into subsets of images by a human.

For example, in the first row of Abb. 4.1 we see at least four foregrounds: a baby, an applebucket and two different girls. With the assumption given above we would now need to create subsets of image which are, on one hand, images 4.1(a) and 4.1(b) and, on the other, image 4.1(c). Then, we would have to do cosegmentation on each subset individually, resulting in more preparatory work, which is increasing with the quantity of images in the image set. This is cumbersome and, due to the human error, error-prone.

Here we will discuss a more realistic but also more challenging task where it is not assumed that each image has the same subset of foregrounds. Particularly, each image of the image set contains a different unknown subset of foregrounds.

Definition 2 (Multiple foreground cosegmentation [2])
The multiple foreground cosegmentation (MFC) refers to the task of jointly segmenting K different foregrounds $F = \{F^1, ..., F^K\}$ from M input images, each of which contains a different unknown subset of K foregrounds.

(Un)supervised learning

Before going deeper into the topic of cosegmentetion, we first need to clarify what supervised and unsupervised, in respect to cosegmentation, is.

With unsupersived learning the user only inputs the image set which have to be segmented and the number of foregrounds $\mathcal{F}$. The models are then estimated over distinctive features that dominate in the images.

If the feature set contains such a distinctive feature, which is not supposed to be a foreground, this feature is selected nevertheless. To demonstrate this, see Abb. 4.1, where unsupervised learning would select the meadow as foreground model for the first row of the pictures because of the dominance but this most certainly would be not in the interest of the user.

With supervised learning, on the other hand, the user inputs the image set and, additionally, pixel-wise annotations on a subset of pictures (20% of the images set was used by [2] and [9]). These annotations outline parts of the image of each foreground within this subset and are used as initial models.

See Abb. 4.2(a) and Abb. 4.2(b) for examples of annotated images. The user annotated four different foregrounds (an applebucket, two different girls and a baby). These annotated segments can then be used as initial models for these four foregrounds. Due to more reliability we will only consider the supervised scenario.

In this report we given overview over two recent cosegmentation approaches: *Multiple Foreground Cosegmentation* by Kim and Xing [2] and *Graph Transduction Learning with Connectivity Constraints* by Ma and Latecki [9]. Both are iterative graph-based proceedings using initial user input to create and improve knowledge about the segments of the image set.

The approach proposed by Kim and Xing [2] consists of two iterative steps, *foreground modelling* and *region assignment*. Foregrounds are represented by heuristic functions and assess how fitting a segment is to a given foreground.

The foreground modelling step is for updating the foreground models, which means that new segments are assigned to the foreground models to make these models more accurate. In the region assignment step, which is the important part of this approach, segments are bundled, based on their adjacency, to create bigger segments which do more sense in the terms of the foreground models. The bundles created in the region assignment step are then used in next iteration for the foreground modelling.

The approach made by Ma and Latecki [9] is based on [2] and shows therefore some similarity. Here, foregrounds are represented by *labels*. These labels are assigned to segments, which are part of the corresponding foreground. Instead of creating bundles of segments, the approach here is to propagate labels from labelled segments to unlabelled segments considering the similarity and adjacency between segments.

This report is constructed in the following way: in sections 2 and 3 we present give an overview on how the proposed approaches made by [2] and [9] are formulated. At the end of either sections we'll discuss the advantages and shortcomings of each approach. Then, in section 4, we explain how the approaches differ and give aspects of advantages for each approach over the other. Finally, we will talk about current medical approaches in cosegmentation and the possibility of using [2] and [9] in section 5.

2 Kim & Xing

In the proposed approach by Kim and Xing [2] cosegmentation is archieved by using two components: foreground models and region assignment. Those components run alternating until a suitable solution is found. We will discuss the two components in the following two subsections.

2.1 Foreground Models

The user inputs, additionally to the set of images, annotations for a fraction of images (20%). These annotations are outlines for each foreground found on those images, see Abb. 4.2(a) and Abb. 4.2(b) as examples.

These annotations are then used as initial data for the foreground model: Each foreground is represented by a model. The model of the k-th foreground is defined as a parametric function $v^k : \mathcal{S} \to \mathbb{R}$ which maps any segment or region of an image to its fitness regarding the k-th foreground.

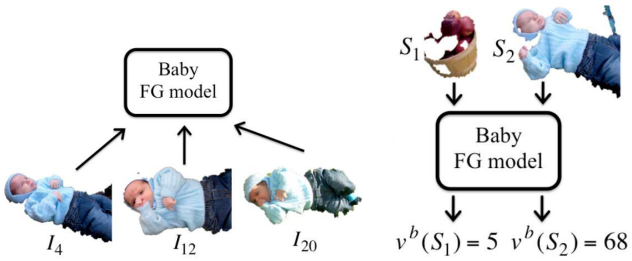
(a) (b) (c)

Abb. 4.2: (a) and (b) are examples for user annotated images from the FlickrMFC dataset [2], (c) is an oversegmented image taken from [2]

An example for a foreground model can be seen in Abb. 4.3(a) which is the baby's foreground model. The three data on the bottom of the figure are annotations provided by the user. In further steps, segments are assessed (depending on the chosen classifier) in respect to similarity to those three annotations.

It is claimed, that any classifier can be used as foreground model as long as it can assess the fitness of a region and can be updated. Used in [2] was a combination of the Gaussian mixture model and spatial pyramid matching.

In Abb. 4.3(b) a sample assessment of two segments by a foreground model is shown. The given baby foreground model assesses how fitting those segments are to the corresponding foreground is, using the heuristic function. As we can see, the left segment (which is part of the applebucket) gets lower rating than the segment representing the baby.



(a) (b)

Abb. 4.3: (a) is an example assignment of segments to a foreground model, (b) displays the assessment of segments by a foreground model. Both images are taken from [2]

Naively we could simply assess the fitness of each segment with each foreground model and then assign each segment to the most fitting foreground. But this would not lead to an optimal solution, considering the following scenario:

We have two foregrounds: a person and a cow. A segment, which to a human eye would obviously be brown human hair, is assessed by every foreground model. But due to the similarity to the cow's fur, which is brown as well, this segment is assigned to the cow's model. Then we assess a segment which represents the body of the person, of course this segment is assigned to the person's model.

The person's hair was assigned to the cow's models due to the similarity between the hair and the cow's fur. This is not a desirable solution.

A better notion would be to combine this two segments and then assess them, because then we assess the person as a whole and therefore this new segment is more fitting to the person's model than to the cow's. This will give us a correct assignment for the person's hair.

Hence, to solve the cosegmentation problem, we need to find disjoint partitions $\mathcal{S}_i = \bigcup_{k=1}^{K+1} \mathcal{F}_i^k$ where $\mathcal{F}_i^k \cap \mathcal{F}_i^l = \emptyset$, with $k \neq l$ to maximize $\sum_{k=1}^{K+1} v^k(\mathcal{F}_i^k)$. More formally, it corresponds to the following integer program:

$$\begin{aligned} max & \sum_{k=1}^{K+1} \sum_{S \subseteq \mathcal{S}_i} v^k(S) x^k(S) \\ s.t. & \sum_{k=1}^{K+1} \sum_{s \in S, S \subseteq \mathcal{S}_i} x^k(S) \leq 1, \forall s \in \mathcal{S}_i \\ & x^k(S) \in \{0,1\} \end{aligned} \tag{4.1}$$

In equation (4.1), $x^k(S)$ is a binary function which describes the allocation for a bundle of segments S to the foreground $\mathcal{F}^k$.

The first line of equation (4.1) refers to the maximisation of the fitness of every bundle of segments for every foreground, with the condition (second line), that every segment of an image is assigned to exactly one foreground. This requires us to check all subsets $S \subseteq \mathcal{S}_i$ for the fitness to every foreground, resulting in $2^{|\mathcal{S}_i|}$ possible checks.

Equation (4.1) is proven to be identical to the weighted set packing problem and is therefore NP-complete and inapproximable. [10]

Given the complexity of equation 4.1, the next step is to use the natural spatial properties of images to find a segmentation which satisfies equation 4.1.

2.2 Region Assignment

The region assignment is done for each image individually, every step described in this section is done for all $I_i \in \mathcal{I}$. The aim is to combine segments, which belong to the same foreground, into regions, which are then denoted as new segments. Each foreground usually occupies a connected region in an image and therefore consists of adjacent segments. Considering this, we built a graph over all neighbouring segments to find regions/candidates which could be occurrences of a foreground.

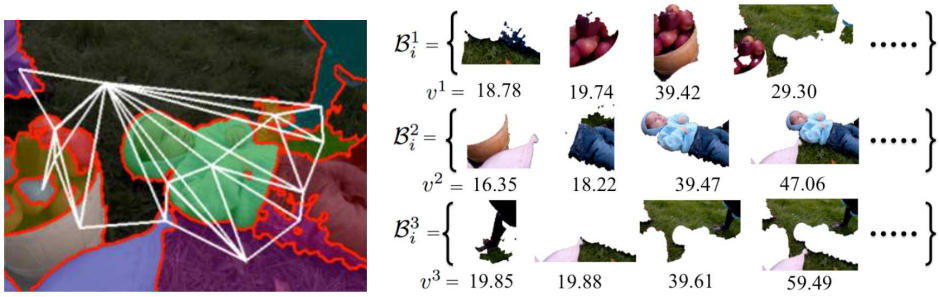
Before we select the candidates, we must map the adjacency with a adjacency graph. We create a graph $G_i = (S_i, \mathcal{E}_i)$ over an image I_i with the segments as it's nodes. Given $\rho > 0$, $(s_l, s_m) \in \mathcal{E}_i$ if the distance between s_l and s_m is at most ρ for all $s_l, s_m \in S_i$. ρ is then the maximum pixel distance between two adjacent segments (e.g. five pixel).

The resulting graph represents the alignment of the segments, and therefore any connected regions can be displayed as subtrees of G_i and the final region assignment would then be a forest of these subtrees. If we view the subtrees as candidate sets we now need to find those candidates with the highest value.

An example for a adjacency graph is displayed in Abb. 4.4(a). The underlying oversegmented picture is the same as Abb. 4.2(c) but here we see the edges of the adjacency graph over the segments as white lines.

A candidate is a triple $B_j^k = \langle k_j, C_j, w_j \rangle$ where k_j is the index of the foreground submitting the candidate, $C_j \subseteq S_i$ is a bundle of adjacent segments and w_j is the fitness of C_j to the k-th foreground. For an image I_i and a foreground $\mathcal{F}^k$ we denote a candidate set $\mathcal{B}_i^k = \{B_1^k, ..., B_n^k\}$, and $\mathcal{B}_i$ is then the set of all candidate sets of image I_i.

We generate each candidate set using beam search. Beam search is a heuristic search algorithm to explore a graph but only using the most promising results regarding a specification. The specification used here is the function v^k, which gives us the fitness of a subtree to the foreground $\mathcal{F}^k$ and therefore a ranking. With this ranking we will only select the most fitting subtrees regarding the foreground $\mathcal{F}^k$ and $D \in \mathbb{N}$ as number of maximum candidates per round. Without D, we would waste time on candidates which have a low rank, but since equation 4.1 requires us to maximize the value, we can ignore those to



(a) (b)

Abb. 4.4: (a) Adjacency graph G_i, (b) Resulting candidate set $\mathcal{B}_i^k$. Both images are taken from [2]

optimize time efficiency.

In algorithm 4 it is shown how the beam search is implemented here. The beam search receives the adjacency graph G_i, the function v^k and beam width $D \in \mathbb{N}$ as input and the output is then the candidate set $\mathcal{B}_i^k$.

First we add all segments $s \in \mathcal{S}_i$ to the candidate set $\mathcal{B}_i^k$ and denote O as the working set, which we also add all $s \in \mathcal{S}_i$ to. We than enumerate all subgraphs of G_i that can be obtained by adding an edge to any $o \in O$, assess the fitness using v^k and only keep the top D highest fitting subgraphs. The resulting subgraphs denote the new set set O and are also added to our candidate set $\mathcal{B}_i^k$. With this new O we go back to the enumeration step.

The search repeats at most $(|\mathcal{S}_i| - 1)$-times, to consider the complete G_i as a foreground candidate, or until O becomes empty.

Algorithm 4 : Build candidates $\mathcal{B}_i^k$ from G_i by beam search

---

Input: (1) Adjecency graph $G_i = (\mathcal{S}_i, \mathcal{E}_i)$, (2) Value function v^k of the k-th foreground model, (3) D : Beam width.

Output: k-th foreground candidates $\mathcal{B}_i^k$

1: Set the initial open set to be $O \leftarrow \forall s \in \mathcal{S}$, $\mathcal{B}_i^k \leftarrow \forall s \in \mathcal{S}_i$.
2: **for** $i = 1$ **to** $|\mathcal{S}_i| - 1$ **do**
3: **for all** $o \in O$ **do**
4: Enumerate all subgraphs O_o that can be obtained by adding an edge to o. $O \leftarrow O_o$ and $O \leftarrow O \backslash o$.
5: **end for**
6: Compute values $v_o \leftarrow v^k(o)$ for all $o \in O$ and remove o from O if it is not top D highly valued, $\mathcal{B}_i^k \leftarrow O$
7: **end for**

In Abb. 4.4(b) an example is shown for the candidate set $\mathcal{B}_i$ of Abb. 4.4(a). Each row represents a candidate set of a foreground, consisting of candidates. These candidates are bundles of segments (in the figure, those are already combined) and their score regarding the fitness to the foreground.

It is noted that the computation time for the beam search for each $\mathcal{B}_i^k$ would be at most $\mathcal{O}(D|\mathcal{S}_i|^2)$ and the number of foreground candidates $|\mathcal{B}_i|$ at most $\mathcal{O}(D|\mathcal{S}_i|)$.

While computation time might be correct, the maximum number of foregrounds candidates is not. On one hand, this algorithm runs for every foreground individually but the authors refer to all candidate sets for a picture (i.e. $\mathcal{B}_i^k$ should be used instead of $\mathcal{B}_i$). On the other hand, $\mathcal{O}(D|\mathcal{S}_i|)$ is true for the loop but since we also add all $s \in \mathcal{S}_i$ to $\mathcal{B}_i^k$ this calculation is not correct. The correct maximum number of foreground candidates $|\mathcal{B}_i^k|$ is $\mathcal{O}(D|\mathcal{S}_i| + |\mathcal{S}_i|)$.

Theorem 1

Dynamic programming can solve region assignment in $\mathcal{O}(|\mathcal{B}_i||\mathcal{S}_i|)$ worst time if every candidate in $\mathcal{B}_i$ can be represented by a connected subgraph of a tree $\mathcal{T}_i^*$. [2]

In theorem 1, it is suggested to create a tree $\mathcal{T}_i^*$ which can represent $\mathcal{B}_i$. Given $\mathcal{T}_i^*$, we will then be able to select feasible candidates and solve equation 4.1 with dynamic programming.

Each candidate $B_i^k \in \mathcal{B}_i$ is subtree in G_i, but the combination of all candidate sets might not. Therefore we need to prune our candidate set of some candidates which cause cycles but are not highly valued. This pruned candidate set, as a subset of $\mathcal{B}_i$, is denoted as $\mathcal{B}_i^*$.

$$T_i^* = \arg\max_{\mathcal{T} \in \mathcal{T}(G_i)} P(\mathcal{B}_i|\mathcal{T}) \tag{4.2}$$

In equation (4.2) $P(\mathcal{B}_i|\mathcal{T})$ is denoted as the data likelihood for the trees of $\mathcal{B}_i$ being part of the tree $\mathcal{T}$ and $\mathcal{T}(G_i)$ is the set of all possible spanning trees over G_i.

$\mathcal{T}_i^*$ is then the spanning tree over G_i with the highest overall value.

Once we obtained the spanning tree $\mathcal{T}_i^*$ over G_i, we consequently obtain $\mathcal{B}_i^*$. As stated in theorem 1, a dynamic programming based search algorithm is implemented, by modifying the CABOB algorithm, and equation 4.1 is solved in $\mathcal{O}(|\mathcal{B}_i||\mathcal{S}_i)$ worst time.

2.3 Experiments

Looking at the experiment results in [2] the first thing to notice is, that they used two different image datasets for benchmarking different aspects, which are accuracy and scalability.

The FlickrMFC dataset is a set of 14 image sets, consisting of 10 to 20 images each. FlickrMFC was used to benchmark the cosegmentation aspects of the approach. They claim that to have reached 48.2% accuracy in average, which was measured the standard metric for PASCAL challenges $\frac{GT_i \cap R_i}{GT_i \cup R_i}$.

The ImageNet dataset [11] contains 50 million images, each is labelled and displaying only one foreground. Also, each image in this dataset is provided with the correct segmentation, which is here used for calculating the accuracy. In their experiments, 1000 randomly selected images were used. This dataset was used for testing the scalability and the performance of single foreground cosegmentation (i.e., $|\mathcal{F}| = 1$). They claim to get a accuracy of around 70% and it took about 20 minutes to segment 1000 images on a single machine (no specifications given).

2.4 Critique

Their advantage is, that they create a database with no a priori information (except the user annotations). But if you already have a good database of foregrounds models, that you would like to segment off an image set, this advantage is not relevant. Nevertheless, we think a priori data could easily be implemented into this approach, since the foreground models are not strict but exchangeable.

Regarding their experiments, Kim and Xing should have used at least one image set which would have tested all of the aspects of their tests at once. Instead, they used different image datasets to benchmark different aspects.

Therefore, it is questionable if a image set covering all aspects would have reached the accuracy level which was reached with the separate test sets.

Further, looking at the FlickrMFC dataset we see general user images, which were the main target of this paper, however, more exotic images (e.g. medical images) would have been interesting, because the images used in the experiments do often have a high contrast between the foregrounds. Using image sets with lower contrast between the foregrounds would have given a better statement for the use case of this approach.

Lastly, the paper contains some mistakes in writing which do confuse the reader and slows the understanding of the argumentation. An example was given in section 2.2. For this report, those mistakes we're corrected with the best knowledge of the author of this report.

3 Ma & Latecki

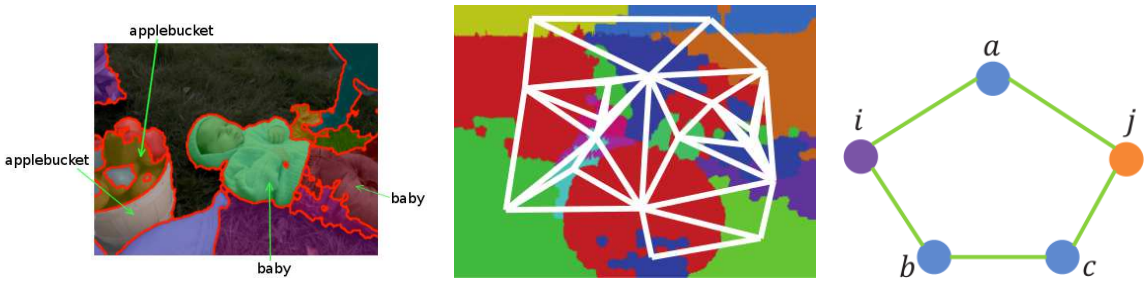
The approach provided by Ma and Latecki [9] is based on the ideas provided by Kim and Xing [2]. They adopt properties like the segment adjacency, combine it with semi-supervised learning and result in a graph-based semi-supervised learning method with connectivity constraints.

Here, it is not the goal to combine segments to bigger segments, but to label each segment with a label that represents the fitting foreground.

Semi-supervised learning (SSL) is considered as in between supervised and unsupervised learning. [9] Rather than letting the user input pixel-wise annotations, the user is provided with some of the already oversegmented images of the image set and the user labels (i.e. annotates) the segments, according to a foreground. The goal is then to predict labels for the unlabelled segments. With regards to [2], segments sharing the same label are assigned to the same foreground.

Looking at Abb. 4.5(a), we see that a user has already labelled some of the segments in this picture. Before submitting the input to the proceeding, all segments should be labelled in this picture.

The advantages of SSL are, (1) the user does not do pixel-wise annotations (reducing time consumption) and (2) the labelled areas are in fact segments (elimination of incorrect pixel-wise annotation). Summarizing, semi-supervised learning requires less human interaction and reduces human error.



(a) (b) (c)

Abb. 4.5: (a) Oversegmented and partly labelled picture [2], (b) Subgraph of the binary adjacency graph G_a [9], (c) is an example for a vertex separator $\{a, b, c\}$ regarding the connected nodes i and j [9]

Coming from the semi-supervised learning step, we now have labelled and unlabelled segments. V represents the set of segments with $N = |V|$ and L represents the set of labels with $C = |L|$. For each segment we calculate the colorSIFT descriptor [12] and quantize the result with a codebook. The representation of segment $s_i \in V$ is then a bag-of-words histogram x_i.

The next step is to build two graphs over the whole image set: a weighted similarity graph and a binary adjacency graph. For both graphs the set of nodes is equivalent to V, but the edges are defined differently.

3.1 Segment similarity

The weighted graph $G_w = (V, W)$ represents the similarity between two connected segments, where W is a non-negative matrix representing the pairwise similarity between the segments. Each entry $w_{ij} \in W$, representing the weight/similarity between the nodes $i, j \in V$, $i \neq j$, is computed using a radial basis function kernel:

$$w_{ij} = exp(-\frac{d(x_i, x_j}{2\sigma^2}) \tag{4.3}$$

where $d(x_i, x_j)$ is the χ^2 distance between the histograms x_i and x_j, and σ is the kernel bandwidth, denoted as $\frac{dist_k}{3}$, where $dist_k$ is the average distance between each sample and the k-th nearest neighbour. $w_{ij} = 0$ if $i \notin kNN(j)$, where $kNN(j)$ is the set of the k nearest neighbours to segment s_j.

$$\begin{aligned} &D = diag([d_1, \cdots, d_N]), \text{ with } d_i = \sum_{j=1}^{N} w_{ij} \\ &Y \in \{0,1\}^{N \times C}, \text{ with } y_{ij} = 1 \text{ iff node } s_i \text{ has label } l \in L \text{ and } \sum\nolimits_l y_{il} \leq 1 \\ &P = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} \end{aligned} \tag{4.4}$$

Looking at the definitions in (4.4) we see, that D represents a diagonal matrix where each d_i is the sum of all weights of edges leaving s_i, and Y maps the labelling of the segments to the labels, where an entry $y_{il} = 1$ if the node has label $l \in L$ and $y_{ij} = 0$ otherwise, and we also assume that each node has at most one label. Lastly, P is the normalized Laplacian matrix of G_w.

Furthermore, we denote the matrix $F \in \mathbb{R}^{N \times C}$ as a variable matrix estimated on the graph to minimize the cost function the following cost function:

$$\mathcal{Q}(F) = tr\{F^T P F + \mu(F - RY^T)(F - RY^T)\} \tag{4.5}$$

which is called the regularized local and global consistency method, where μ is a constant and R is a matrix to balance the leverage of the different labels, with

$$\begin{aligned} &R = diag([r_1, \cdots, r_N]), \text{ where} \\ &r_i = \begin{cases} \dfrac{1}{C} \cdot \dfrac{d_i}{\sum_k \; y_{kl} \; d_k} & \text{if } \exists_{l \in L} \; y_{il} = 1 \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{4.6}$$

which is very important for problems with highly unbalanced labelled nodes.

Then, due to the convexity and by zeroing the partial derivative $\frac{\delta \mathcal{Q}}{\delta F}$ [9], we arrive at a closed form solution

$$F^* = (\frac{L}{\mu} + I)^{-1} RY \tag{4.7}$$

where each $F^*_{il} \in F^*$ expresses the probability for the node $i \in V$ being labelled with label $l \in L$. This closed form solution will help us

$$\begin{aligned} &\mathcal{Q}(F) = tr\{F^T P F + \mu(F - RY^T)(F - RY^T)\} \\ &\quad s.t. \; tr(MF) \leq 1, \; \forall M \in C \end{aligned} \tag{4.8}$$

is the same equation as equation 4.5 but with a linear constraint set C as a set of matrices $M \in \{-1, 0, 1\}^{C \times N}$ which later will influence F to fulfil the connectivity constraints.

Therefore, we result with a convex quadratic programming problem (equation 4.8 and a closed form solution F (equation 4.7). Although F can not be derived, the quadratic programming problem can be solved efficiently by many existing solvers (e.g. IBM CPLEX). [9]

Since we're now able the calculate to probability for a node i to have a label l, we next need to take the connectivity into account. Like [2], here we consider as well, that each occurrence of a foreground does consists of adjacent segments.

3.2 Connectivity constraint

The second graph is a binary adjacency graph $G_a = (V, A)$, where two nodes are connected if the corresponding segments are part of the same image and adjacent to each other. A sample subgraph of G_a is shown in Abb. 4.5(b), where the colored fields represent segments and the white lines are the edges representing the adjacency between the segments.

Given this undirected graph G_a and with every two connected, but not adjacent, nodes $i, j \in V$ with both sharing the same label $l \in L$, we denote a vertex-separator as a set $S \subseteq V \setminus \{i, j\}$, where removing S from V would result in the disconnection of i and j. See Abb. 4.5(c), where the set of nodes $\{a, b, c\}$ is the vertex-separator of the nodes i and j and removing this set would disconnect i and j.

Further, we denote an essential vertex-separator $\bar{S}$ as a vertex-separator, where every node $v \in \bar{S}$ is essential for disconnecting i and j. In other words, the set $\bar{S}$ is an essential vertex-separator if any strict subset of $\bar{S}$ is not a vertex-separator. Finally, $\mathcal{S}(i, j)$ is defined as the set of all essential vertex-separator regarding i and j. See Abb. 4.5(c), where the sets $\{a, b\}$ and $\{a, c\}$ are essential vertex-separators and therefore $\mathcal{S}(i, j) = \{\{a, b\}, \{a, c\}\}$.

We denote $\mathcal{H}$ as a set of all triples (i, j, l), where $i, j \in V$ share the same label l and there exists a non-empty set $\mathcal{S}(i, j)$. Then, the set $\mathcal{H}$ is called check condition set because we only need to check the triples in $\mathcal{H}$ regarding the connectivity condition.

This proceeding, which is called graph transduction with connectivity constraints, operates in iterative steps. In each iteration t the first step is to obtain F^t, which is a matrix mapping the probability for all segments $s_i \in V$ having the label $l \in L$ at entry F_{il}, by solving equation 4.8.

We then need to check whether the connectivity constraint is violated, and if it is, how to fix it.

$$F_{il}^t + F_{jl}^t - \sum_{k \in \bar{S}} F_{kl}^t - 1 \leq 0, \ \forall \bar{S} \in \mathcal{S}(i, j) \tag{4.9}$$

describes the linear connectivity constraint at iteration t, whereas two connected but not adjacent segments i and j sharing the same label l should not be separated by an essential vertex-separator in which no segment is not labelled with l.

Hence, for each $(i, j, l) \in \mathcal{H}$ we can obtain the essential vertex-separator with the the most violated inequality regarding (4.9):

$$\begin{aligned} \bar{S}^*(i, j, l) &= \underset{\bar{S} \in \mathcal{S}_{(i,j)}}{\arg\max} \sum_{k \in S} F_{kl}^t \\ (i^*, j^*, l^*) &= \underset{(i,j,l) \in H}{\arg\max} \sum_{k \in \bar{S}^*(i^*, j^*, l^*)} F_{kl}^t \end{aligned} \tag{4.10}$$

$\bar{S}^*(i, j, l)$ expresses the essential vertex-separator of the set of all essential vertex-separators of node i and j where the sum of the probabilities for each segment to have label l is the lowest. Whereas the triple $(i^*, j^*, l^*) \in \mathcal{H}$ is the most violated inequality among all $(i, j, l) \in \mathcal{H}$ regarding equation 4.9.

In [9], the acquirement of $\bar{S}^*(i, j, l)$ is efficiently solved by computing max-flow on an auxiliary directed graph.

Finally, we obtained $\bar{S}^*(i^*, j^*, l^*)$ in equation 4.10 as the essential vertex-separator which yields the maximum value in (4.10) and therefore is the most violated inequality regarding (4.9).

$$F_{il}^t + F_{jl}^t - \sum_{k \in \bar{S}^*(i^*, j^*, l^*)} F_{kl}^t - 1 \leq 0 \tag{4.11}$$

is a stopping condition, which indicates, that even $\bar{S}^*(i^*, j^*, l^*)$ does not violate the connectivity constraints in (4.9), hence there is no essential vertex-separator in V which has a low probability to have label l.

Otherwise there is a constraint violated and which can be represented by the l^*-th column in matrix $M \in \{-1, 0, 1\}^{C \times N}$, with $M_{i^* l^*}, M_{j^* l^*} = 1$, and $M_{k l^*} = -1$ if $k \in \bar{S}^*(i^*, l^*, k^*)$ and $M_{k l^*} = 0$ otherwise. This new M is added to the set of linear constraints C, which is used as condition in equation 4.8 to fulfil all connectivity constraints. A new iteration begins with the updated C.

The iteration stops either, if (4.10) is fulfilled or the change between F^t and F^{t+1} is smaller than a given threshold σ. The whole procedure is summarized in algorithm 5.

Algorithm 5 : Graph Transduction with Connectivity Constraints

Input: P, A, μ, σ
Output: $F^* = F^t$

1: $t = 1$, initial C^t as an empty set
2: **repeat**
3: Obtain F^t by solving equation 4.8
4: Find the most violated constraints $\bar{S}^*(i^*, j^*, l^*)$ using equation 4.10
5: **if** Equation 4.11 holds for $\bar{S}^*(i^*, j^*, l^*)$ **then**
6: break
7: **end if**
8: Derive linear equality constraint M from $\bar{S}^*(i^*, j^*, l^*)$
9: $C^{t+1} \leftarrow C^t \cup M$
10: **until** $|F^t - F^{t-1}| < \sigma$

Since $F^* \in \mathbb{R}^{N \times C}$ is not binary, the final step is to binarize F^* to the label indicator Y^*:

$$\forall i \in V : l^* = \arg\max_{l \in L} F^*_{il}, \text{ then } Y^*_{il} = \begin{cases} 1 & l = l^* \\ 0 & l \neq l^* \end{cases} \tag{4.12}$$

Y^* is the final label distribution of labels, where each segment is labelled with the label of highest probability.

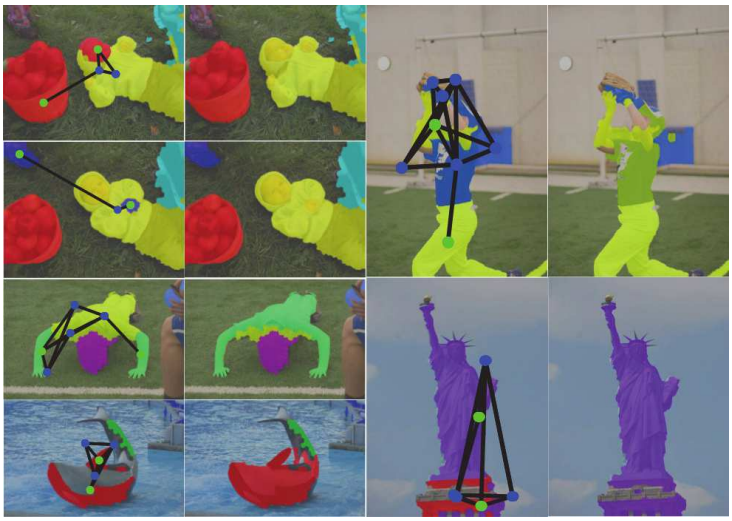


Abb. 4.6: Most violated connectivity constraints [9]

In Abb. 4.6 an example for resolving the connectivity constraint is shown. For every pair of images: on the left image, the green dots are the nodes which have the same label and are not connected, the blue dots are essential vertex-separators regarding the green nodes and the black lines represent the connectivity between the nodes, and on the right images are results with fulfilled connectivity constraints.

3.3 Critique

Some aspects pointed out in section 2.4 also apply here. They do not use any a priori information, except the user input. We think, other than with MFC, a priori information could not be implemented in this approach.

The experiments presented are fairly limited, as they were reduced to tests on the FlickrMFC, and therefore did not test the runtime and scalability. The critique about this dataset, as stated in section 2.4,

apply here too. GTC reached a accuracy of 62.6%, measured with the standard metric for PASCAL challenges. Although it is more accurate than MFC, it is still fairly low for high contrast foregrounds.

Overall, this paper is well written but still could have been a bit more detailed (especially section 3.2 seems a bit too compressed).

4 Comparison

The presented approaches [2, 9] where different approaches on how to cosegment image sets, where the subset of foregrounds for each image is not know and differs from image to image. Since GTC [9] is based on MFC [2] it is not surprising, that GTC has a higher accuracy.

Looking on the approaches in detail we see:
On one hand, Multiple Foreground Cosegmentation, where our foregrounds are represented by models. An adjacency graph was created to take the natural property of spatiality into account. Then we bundled segments with regard of the fitness of these bundles to the foregrounds. Afterwards, we had to select bundles which would give us an overall highest value with respect to the fitness to the foregrounds.

On the other hand, Graph Transduction with Connectivity Constraints, where a similar but different approach was taken. It was not supervised, but semi-supervised learning used. Instead of pixel-wise annotations the user had to label segments. This resulted into labelled and unlabelled segments. Moreover, instead of using models, here a second graph was introduced which represented the pairwise similarity between all segments. The task was then to propagate the labels from the labelled segments to the unlabelled with respect to the similarity and connectivity constraint.

Since GTC presented around 14% higher accuracy than MFC on the same image set, GTC has an advantage in this regard. However, [9] did not test for scalability and runtime. Limited to accuracy of segmentation, GTC is the superior proceeding of those two.

5 Medical application

As stated in the introduction, the amount of data is increasing, even in the medical field. The tools used to create images in the medical field (e.g. for diagnosis, study and treatment planning) get better and create more data (that is, less noise and higher resolution).
These imaging tools (e.g. magnetic resonance imaging, computed tomography) do create the images but do not analyse them. And since the amount of images is increasing, it gets harder to analyse them data by hand. Therefore, we need to support specialists with computer based image processing, which is already done in real life with well tested and accurate proceedings. [1]
With reference to new proceedings, we have to provide reliability, because without reliability it is very risky to use proceedings for a medical application due to the fact, that a false result could lead to a false diagnosis, which, in return, could then lead into false positive and/or false negative errors.

As stated in section 2.4, the dataset used for benchmarking both MFC and GTC has contrast-rich foregrounds. This is usually not given with medical images where, in most cases, they consist of different levels of grey and therefore do not have a high contrast between foregrounds. It is questionable whether the presented approaches, in the current state, would be applicable medical images. And referring to the experiments, those approaches had fairly low accuracy, thus the reliability needed for a medical application is not given.

6 Conclusion

We presented two current state-of-the-art cosegmentation proceedings, both having fairly low accuracy on contrast rich foregrounds.
Further research and modification on both MFC and GTC would give a notion whether they are suited for medical application or not.

Literaturverzeichnis

[1] Pham DL, Xu C, Prince JL. Current Methods in Medical Image Segmentation. Annual Review of Biomedical Engineering. 2000;2(1):315–337. PMID: 11701515.

[2] Kim G, Xing EP. On multiple foreground cosegmentation. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on; 2012. p. 837–844.

[3] Hochbaum DS, Singh V. An efficient algorithm for Co-segmentation. In: Computer Vision, 2009 IEEE 12th International Conference on; 2009. p. 269–276.

[4] Vicente S, Rother C, Kolmogorov V. Object cosegmentation. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on; 2011. p. 2217–2224.

[5] Rother C, Minka T, Blake A, Kolmogorov V. Cosegmentation of Image Pairs by Histogram Matching - Incorporating a Global Constraint into MRFs. In: Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. vol. 1; 2006. p. 993–1000.

[6] Joulin A, Bach F, Ponce J. Discriminative clustering for image co-segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on; 2010. p. 1943–1950.

[7] Kim G, Xing EP, Fei-Fei L, Kanade T. Distributed cosegmentation via submodular optimization on anisotropic diffusion. In: Computer Vision (ICCV), 2011 IEEE International Conference on; 2011. p. 169–176.

[8] Joulin A, Bach F, Ponce J. Discriminative clustering for image co-segmentation. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on; 2010. p. 1943–1950.

[9] Ma T, Latecki LJ. Graph Transduction Learning with Connectivity Constraints with Application to Multiple Foreground Cosegmentation. In: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on; 2013. p. 1955–1962.

[10] Lehmann D, Müller R, Sandholm T. The Winner Determination Problem. In: Combinatorial auctions. MIT Press; 2006. p. 555–596.

[11] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on; 2009. p. 248–255.

[12] van de Sande K, Gevers T, Snoek C. Evaluating Color Descriptors for Object and Scene Recognition. IEEE Trans Pattern Anal Mach Intell. 2010 Sep;32(9):1582–1596.

Title

Seminar Medical Image Processing
Cell Tracking in Video Microscopy using Graph based Algorithms

Dominik Chmiel

Zusammenfassung

The areas, in which tracking of multiple-objects can be applied, is very diverse. Many different approaches to construct trajectories for single objects through an image-sequence have been proposed. A subset of those proposed methods utilize graphs to formulate part of the multi-frame multi-object tracking. In this paper three approaches will be discussed in detail. The first two presented algorithms utilize minimum weight bipartite matching and min-cost network flow to formulate the trajectory-reconstruction problem. Finally, an algorithm that can be applied in a video surveillance system will be presented, which integrates graph partitioning and graph matching for trajectory analysis. Finally, the advantages and disadvantages of each approach will be highlighted.

Keywords: Multi-frame multi-object tracking, min-cost network flow, bipartite matching, trajectory analysis, video surveillance, cell tracking, Lagrangian relaxation, entropy thresholding

1 Introduction

Tracking real-world objects in a given video sequence is a fundamental problem in computer vision and image processing. It involves two main issues: Extracting objects of interest and segmenting them from the image-background and finding correspondences of objects over multiple frames. Analysing and recreating the trajectory of multiple targets is a complex task which is a critical part of recently-arising intelligence systems. From robotics [1] to video surveillance systems [2], the possible applications are diverse. Developing multi-frame algorithms yielding good approximations to the NP-hard problem of multi-frame assignment[3] with polynomial running time is a topic of high interest. In this paper, the focus will lie on graph-based algorithms. These algorithms reformulate parts of the tracking problem using graphs and basic graph-algorithms to approximate the tracking-problem.

Early approaches in this field utilized greedy bipartite matching on a frame-by-frame basis to recreate object trajectories piece by piece[4]. Taking the objects from two subsequent frames and creating an assignment utilizing a weight function including e.g. the Euclidean distance is a problem that can be solved in polynomial time using the Hungarian algorithm. However, this kind of algorithm yields poor results when objects are occluding each other or interact with each other.

More recent work is trying to calculate a global optimal solution by using a multi-frame-window rather than matching only pairs of frames. Using a multi-frame window enables the inclusion of higher-order smoothness constraints to adapt to more diverse movement patterns of tracking targets. This type of algorithm can be further divided into two groups: Algorithms creating network flow graphs over the entire sequence (e.g. [5]) and algorithms utilizing iterative hierarchical methods to link track-pieces (e.g. [6]).

In this paper, three different approaches will be discussed in detail. First, some basic algorithms utilized will be presented in Section 2.2. Then, the individual algorithms trying to solve the multi-frame multi-object tracking problem will be presented. One approach utilizing bipartite matching algorithms, but instead of matching pairs of frames a three frame window is observed. The approach presented in Section 3.2 utilizes min-cost network flow with Lagrangian relaxation. Lastly, an algorithm applicable in a surveillance system is presented in Section 3.3. For each algorithm, advantages and disadvantages are highlighted and discussed in Section 4.

2 Basics

The three algorithms for graph based multi-object-tracking that will be presented later in this paper are making use of a number of different algorithms. Bipartite graph matching, min-cost network flow and algorithms for extracting cells from a live image feed are used. A selection will be outlined and explained in the following sections.

2.1 Graph

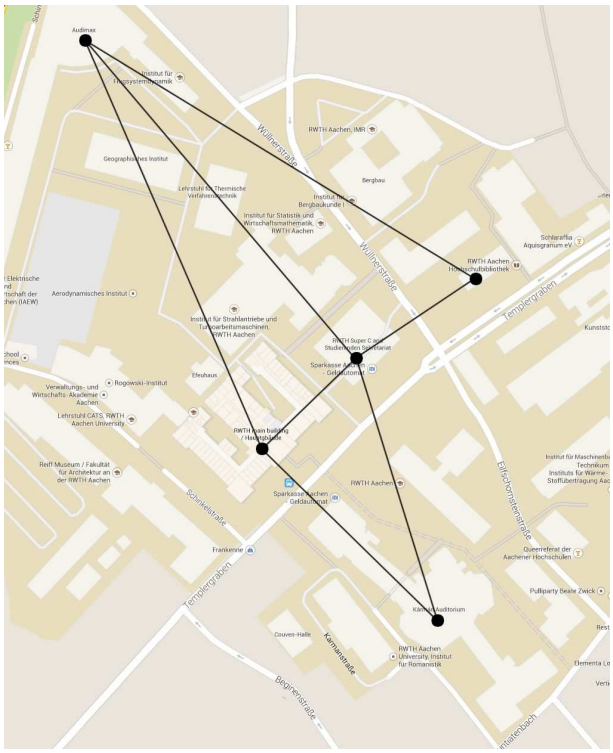


Abb. 5.1: An example for a graph representing a small navigation-network of the RWTH-campus.

Each presented algorithm uses graphs to model parts of the given problem. Graphs are a widely used and highly flexible data structure that can be used to model different problems ranging from route-planning in a navigation system (as seen in Figure 5.1) to trajectory-tracking. Graphs were introduced a long time ago, the first mention going as far back as 1878[7].

In its most basic form, a graph $G = (V, E)$ consists of a set of nodes V and a set of edges E, where each edge $(u, v) \in E$ is a two element subset of V, representing links or relationships between the nodes u and v. Each edge and each node can be labelled with additional properties of this specific entity describing it further. An example would be in a graph where nodes represent different points of interest to label the edges with the geometric distances between each pair of points (seen in Figure 5.1). In some cases it is useful to assign a direction to each edge. A directed edge between two nodes u and v can be used to represent e.g. a hierarchy (u before v) or a capacity of the connection.

Another example for a graph based problem would be the travelling salesman problem. The idea is that a salesman wants to visit a number of stores in different cities to promote his new product. Of course he only wants to visit each city once, and he wants to be back home after his tour. Again, each city can be represented as node, and each possible route as an edge, with the time taken for the route as edge-label.

In the following section, a few basic graph-algorithms are presented.

2.2 Algorithms on Graphs

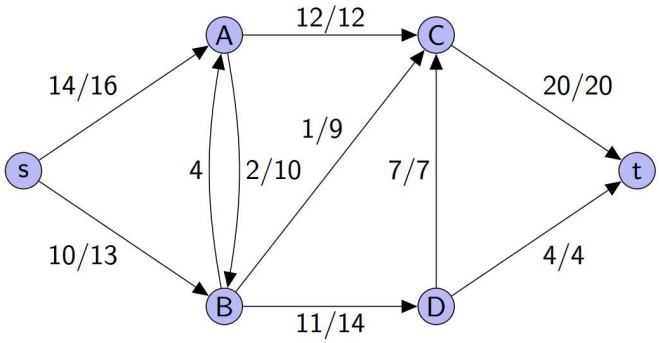


Abb. 5.2: Example for a maximal flow in a network graph. Each edge is labelled with f/c where f is the current flow and c is the capacity of this edge. [8]

2.2.1 Min-Cost Network Flow A flow network is a directed graph, where each edge has a certain capacity and receives a flow. Additionally, a flow network needs to have at least one source-node s and sink-node t. A source node only has outgoing flow, while a sink node has only incoming flow.

A flow is defined as a function $f(u, v)$, which assigns a flow value to each link (u, v), while not exceeding the capacity of this link. Additionally, $f(u, v)$ needs to fulfil the following constraints:

$$f(u, v) = -f(v, u) \tag{5.1}$$

$$\sum_{(u,v)\in E} f(u, v) = 0 \,\forall\, v \in V \backslash \{s, t\} \tag{5.2}$$

Finding the maximum possible flow in a network graph can be accomplished using e.g. the Ford-Fulkerson-Algorithm: Given a network graph with only one source and one sink, apply the following steps:

1. Search a path ρ from source to sink
2. Increase the flow for each edge in ρ by the smallest capacity in ρ
3. Search a path ρ' from source to sink, using only edges where the current flow is lower than the capacity
4. Supplement the flow of all edges in ρ' again by the smallest capacity of ρ'
5. Repeat steps 3. and 4. until no path with free capacities can be found

The min-cost network flow algorithm adds a cost-factor to each flow. The goal of the algorithm is to not only find a flow that is maximal, but also to keep the cost necessary for that flow as low as possible.

For calculating a min-cost flow in a flow network, each edge (u, v) needs to be additionally labelled with a cost $c(u, v)$. The problem that needs to be solved now is minimizing the total flow-cost defined by

$$min \sum_{(u,v)\in E} c(u, v)f(u, v) \tag{5.3}$$

Since all constraints of this problem as well as the problem itself are linear, it can be formulated as linear program. This can be solved using for example the simplex method.

2.2.2 Bipartite Matching Given a set of objects V and associated labels for each object, it is often necessary to find pairs that are similar to each other or can be associated with each other. The problem of finding such an association is called finding a matching, and the set of resulting pairs is called a matching. In most cases, each object can only be part of a single match. Here, this is the only approach discussed.

In many cases, the graph that should be matched does not have an arbitrary form, but rather consists of two independent sets of vertices V_1 and V_2. These two object-sets can represent different classes of objects - for example V_1 representing soccer players, V_2 representing soccer clubs - or the same set of objects at different time points - for example the objects visible in two subsequent frames from a video feed as shown in Figure 5.3. Each edge now represents a possible association from two objects from the two object-sets.

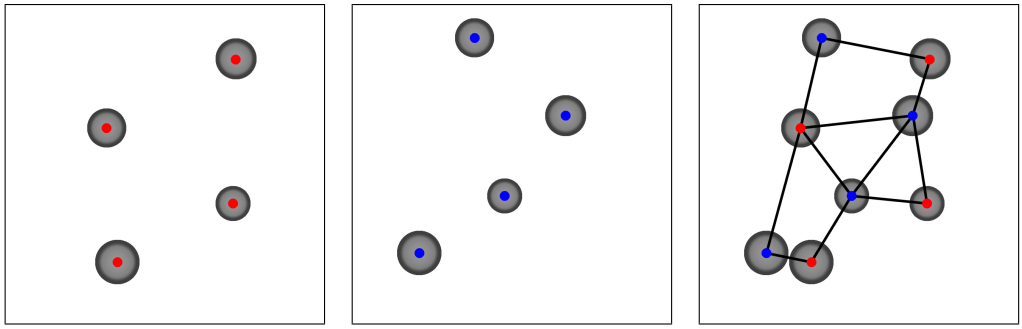


Abb. 5.3: Bipartite Matching applied to a set of observed objects. Left: First frame. Middle: Second frame. Right: Bipartite graph spanning both frames.

When creating a matching for a graph, different aspects of the objects can be used to rate a matching. In the example of cell-tracking, creating a matching that is including as many cells as possible would be most reasonable. In most cases, only a few cells enter or exit the frame in between two frames, so only very few cells will stay unmatched. This type of matching is called a maximal cardinality match and can be calculated using for example the Hopcroft-Karp algorithm[9].

A different approach would be to minimize a given weight function, for example the average distance between all matches. This minimum weight matching works best in a scenario where objects do not move much in-between observations. It can be computed using the Hungarian Algorithm, which is outlined in Algorithm 6.

An example for both these variations of bipartite matching is shown in Figure 5.4. On the right, maximum cardinality matching has been applied. This leads to a total of 3 matches. On the left a maximum weight matching is shown. In the case that multiple matches are possible for a single node - for example (1,x) and (1,y) - the one with the highest weight is chosen. This leads to a higher average weight of the matching, but leaves two nodes unmatched. The minimum weight matching is analogue, but with the lower weight match being chosen over the higher one.

By using different parameters to determine the weight, multiple parameters can be factored into the matching. Taking the cell-tracking example, the edges could be labelled with

$$d_{ex}(u, v) = ||u - v||_2 + |r(u) - r(v)| \tag{5.4}$$

where u and v are the cell positions and $r(u)$ is radius of cell u. A minimum weight matching on this graph would match cells more likely that are similar in their radius while being as close as possible to each other. An example for this type of matching is shown in Figure 5.3.

Another approach to solve the maximum cardinality bipartite matching problem is to formulate it as a maximal flow problem by adding a source s connected to all vertices in set $V1$, and a sink t connected to all vertices in set $V2$. Every edge will be labelled with a capacity of 1. If an edge gets assigned a flow of 1 by the maximal flow on this network graph, it is also part of the maximum cardinality matching.

3 The Presented Algorithms

The problem of tracking multiple objects over multiple frames can be divided into two main parts. First, features of the objects that should be tracked need to be extracted. In the first two presented algorithms, the only used feature is the position for each object, while the surveillance system presented in Section 3.3 uses additional object features. In the second step, these features need to be linked into trajectories. In the next Sections, the different approaches to those two steps will be explained.

Algorithm 6 Pseudocode-outline of the Hungarian algorithm.[10]

Require: Bipartite Graph $G = (V_1, V_2, E)$ (where $|V_1| = m, |V_2| = n$), C: mxn matrix of edge costs
for $i = 0$ **to** m **do**
colmin = $min(C.column(i))$
for $j = 0$ **to** n **do**
$C[i, j] = C[i, j] -$ colmin
end for
end for
for $j = 0$ **to** n **do**
rowmin = $min(C.row(i))$
for $i = 0$ **to** m **do**
$C[i, j] = C[i, j] -$ rowmin
end for
end for
loop
Search for combination of zeros in C, so that only one zero per row and column is selected
if no combination possible **then**
Find minimal number of horizontal and vertical lines so that all zeros inside C are crossed out
Find smallest coefficient in C not covered by lines, store in mincoeff
for $i = 0$ **to** m **do**
for $j = 0$ **to** n **do**
if $C[i, j]$ was crossed by exactly one line **then**
$C[i, j] = C[i, j]$
else if $C[i, j]$ was not crossed **then**
$C[i, j] = C[i, j] -$ mincoeff
else
$C]i, j] = C[i, j] +$ mincoeff
end if
end for
end for
else
Return indices of matrix-cells containing zeros
end if
end loop

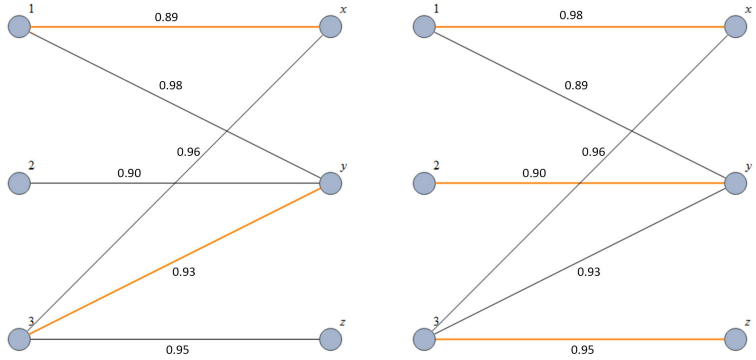


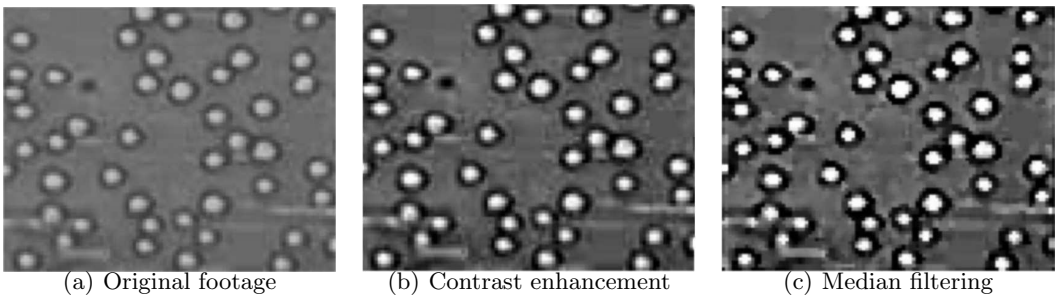
Abb. 5.4: An example showcasing maximum weight (left) versus maximum cardinality bipartite matching (right) [11]

3.1 Cell Tracking in Video Microscopy using Bipartite Graph Matching

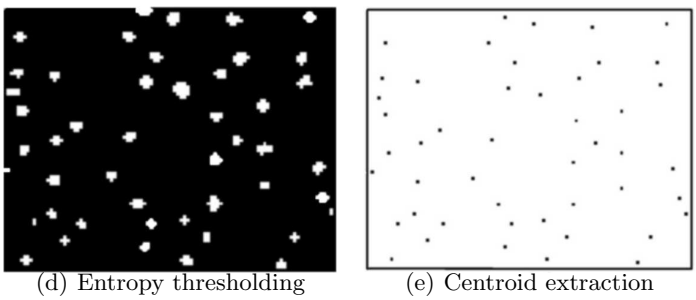
3.1.1 Overview The Method presented in [12] is the only method presented that was specifically designed for tracking cells in a video microscopy image. It is also the most recent work presented and claims to outperform earlier algorithms designed for cell tracking.

The algorithm can be divided into three main steps. At first, the images are pre-processed and the current positions of cells in a given frame are determined. Next, these results are matched between succeeding frames taking their past movement into consideration. Finally, trajectories are linked for each cell visible in the observed time window and possible errors are corrected. Each step will be explained in detail in the next Sections.

3.1.2 Pre-Processing The input is a grayscale image sequence of human monocyte cells obtained using microscopic video. No assumption is made for the movement patterns of cells. Additionally, the density of cells is quite high as can be observed in Figure 5.6.



(a) Original footage (b) Contrast enhancement (c) Median filtering



(d) Entropy thresholding (e) Centroid extraction

Abb. 5.5: A single frame taken from the captured images going through the stages of pre-processing. [12]

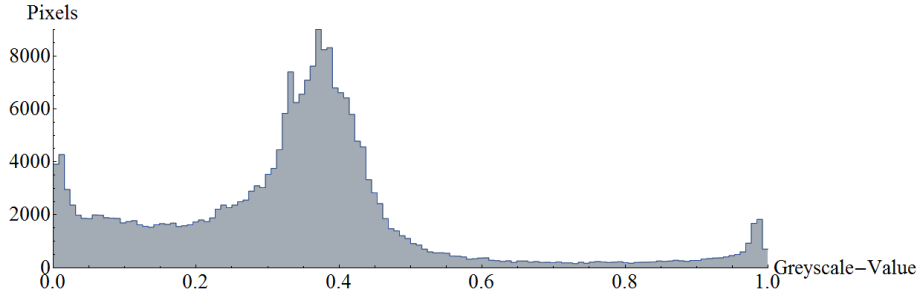


Abb. 5.6: A greyscale-histogram of the image in Figure 5.5 showing the amount of pixels at each discrete greyscale-value

In-vitro imaging tends to be susceptible to a lot of noise. The images have a very low contrast as seen in Figure 5.5 (a). To better distinguish between cells and background, the contrast is enhanced (as seen in Figure 5.5(b)). Since the images are mainly different shades of grey - no part of the original image is white or black - this can be done by applying the following function to the image:

$$\mathrm{f_{enhance}(x)} = \begin{cases} 1 & \text{if } x \geq t_1 \\ 0 & \text{if } x \leq t_2 \\ \dfrac{(x - t_1)}{t_2 - t_1} & \text{if } t_2 < x < t_2 \end{cases} \tag{5.5}$$

This function maps all values below a certain threshold t_1 to black, all values above a second threshold t_2 to white, and stretches all values in between to cover the whole gamut. To reduce the noise-levels, median-filtering is applied to the images. This leads to an image where the cells are clearly distinguishable from the background. Even in regions where multiple cells are intersecting and are occluded by other cells, as for example in the bottom right of the example image, the cells are now clearly distinguishable.

Entropy thresholding is used to determine whether a given pixel belongs to a cell or not. An example for a simple thresholding function is given in Equation 5.6. If the pixel has a brightness higher than given by t, it is marked as 1, otherwise 0. Pixels that get assigned the value 1 are considered as belonging to a cell [13].

$$\mathrm{f_{thres}(x)} = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases} \tag{5.6}$$

In Figure 5.6 a histogram of Figure 5.5 (c) is shown. It can be observed that a large amount of pixels has a brightness seated in the bottom half of the gamut. So even though the cells are composed of black and white with the background being mostly shades of grey, it would not be feasible to mark both black and white as cells, since a black pixel could also occur in the background. Hence we apply a threshold-function with a threshold of roughly $t = 2/3$. The resulting image can be seen in (d).

As a final pre-processing step, neighboring white pixels are combined to a single cell. The center of this cell can be extracted by calculating the average of all pixel-positions belonging to a single cell. The resulting estimated cell positions are marked by black dots in image (e).

3.1.3 Bipartite Matching If the source video sequence has a length of T frames, a set of positions F_k is acquired for each frame k.

$$F_k = \{pos_{k,i} \mid i = 1, ..., n\}, k \leq T$$

These single positions need to be linked into trajectories of single cells to extract the movement of individual cells over time. A bipartite graph can be constructed for two subsequent frames and the corresponding cell-positions F_k and F_{k+1}, with every feasible match being connected by an edge labelled with the likeliness of that match.

To compute the likeliness it is proposed to incorporate the mean motion of all cells as well as the past motion for each individual cell. Computing the mean motion is done by constructing a graph

$$G_{mean} = (F_k \cup F_{k+1}, E_{k,k+1}) \tag{5.7}$$

where

$$E_{k,k+1} = \{(u, v) \mid u \in F_k \wedge v \in F_{k+1}\} \tag{5.8}$$

Each edge gets labelled using the Euclidean distance

$$d(u, v) = ||u^2 + v^2||_2 \tag{5.9}$$

Where $u \in F_k$ and $v \in F_{k+1}$ are the positions of two cells.

Now, a maximal cardinality matching is computed on this graph. The maximal cardinality assures that we match as many cells as possible between two frames, even though it can be assumed that due to entry and exit of cells some may be left unmatched. If the mean displacement of cells would be computed using this match, single fast moving cells could distort the final value heavily, leading to inconsistencies. To account for this, it is proposed to sort the matched cells based on the likeliness function - in this case the Euclidean distance. Now, the mean value of the top $q\%$ is computed, leading to a stabilized average for the mean movement component. If $M = \{(u, v)|u \in F_k, v \in F_{k+1}\}$ is the computed matching sorted ascending by the Euclidean distance of the matched cells and $m = \lfloor \frac{q}{100}|M| \rfloor$ the mean displacement can be computed using

$$\overline{d} = \frac{1}{m} \sum_{i=1}^{m} d(M_i) \tag{5.10}$$

Since we do not have any prior knowledge about movement patterns of the observed cells, simply using this bipartite match could potentially lead to a very low amount of correct matches. To improve this, a value describing the past motion of each cell is computed. Assuming cells have already been tracked over the last frames and the position of cell i in frame k is $u_{k,i}$, the past movement term can be expressed as

$$pm(k, i) = (u_{k-1,i} - u_{k,i}) + \lambda_2 \, pm(k - 1, i) \tag{5.11}$$

Assuming that the past matched cells preserve the same index through each frame, so the cells described by $u_{k,i}$ are the same as $u_{k-1,i}$. The value of $\lambda_2 \in [0, 1]$ now changes how heavily the last frames are incorporated.

If cell i has not was not found in frame k, $pm(k, i)$ is set to zero. This model has the advantage that it is very general. Since all additional constraints are calculated component-wise and without making any assumption on for example how the cell is supposed to move, it can adapt to many different situations, with the only λ_2 impacting the result.

The computed values now need to be taken into account when computing a new matching for two succeeding frames. This is done by computing a prediction of the cell position for frame $k+1$ and matching those with the real positions in frame $k + 1$ instead of using the positions in frame k as done previously. A prediction for the position of a cell in frame $k + 1$ can be computed by

$$u_{new} = u + \lambda_1 \, \overline{d} + \lambda_2 \, pm(k, i) \tag{5.12}$$

being subject to the constraints

$$\lambda_1, \lambda_2 \in [0, 1] \tag{5.13}$$

$$\lambda_1 + \lambda_2 = 1 \tag{5.14}$$

The second constraint is necessary because both the mean displacement and the past motion term are predictions for the movement in between two consecutive frames, and λ_1 and λ_2 are controlling the weighting of both components.

This predicted position is a good estimate, since it factors in the mean motion of the majority of all cells as well as adapting to the movement behaviour of every individual cell.

To compute the final matching between frames F_t and F_{t+1} we can now construct a second bipartite graph $G_{adapted} = (F_t \cup F_{t+1}, E_{t,t+1})$, but instead of using the Euclidean distance of each cell between two frames to label the edges, we use the weight function

$$d_{new}(u, v) = ||u^2_{new} + v^2||_2 \tag{5.15}$$

After applying a minimum cost maximum cardinality matching to $G_{adapted}$ the resulting match will match every cell that was not exiting or leaving the frame in the observed frames very dependable.

3.1.4 Trajectory Linking While it would be possible to only compute matches for subsequent frames without regarding what happened before or after those frames would be possible, the accuracy of the algorithm can be improved greatly if multiple matches are calculated. For a good trade-off between accuracy and performance, three subsequent frames are observed.

For those three frames, three matches are calculated: F_{k-1} and F_k, F_k and F_{k+1}, and finally F_{k-1} and F_{k+1} are matched. This results in two paths from F_{k-1} to F_{k+1} for each cell.

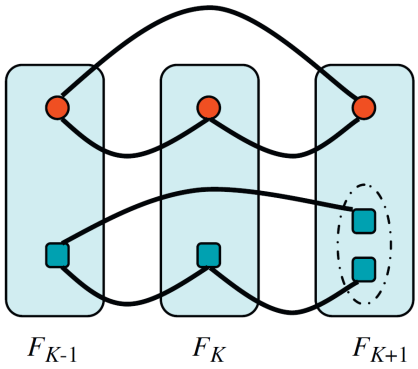


Abb. 5.7: Two scenarios for trajectories of cells tracked over three consecutive frames [12]

There are two possible scenarios for those paths:

1. Both paths lead to the same cell in F_{k+1}
2. The paths end in different frames in F_{k+1}

In the first case, indicated by red circles in Figure 5.7, we can assume that the resulting trajectory is correct, so there is no further need for investigation. In the other case, indicated by the green squares, the two cells inside the dotted ellipse are both possible resulting matches for the cell in Frame F_{k-1}. To resolve this conflict, a match-score $m(u, v)$ is introduced, which can be based on their Euclidean distance, past-movement behavior or other parameters. The exact method used to calculate this match-score is not further defined in the referenced paper. This match-score is now used to rate the two different trajectories for each cell.

$$\beta' = m(F_{k-1}, F_{k+1}) \tag{5.16}$$

$$\beta'' = m(F_{k-1}, F_k) + m(F_k, F_{k+1}) \tag{5.17}$$

According to the authors, if $\beta' < \beta''$, the trajectory resulting from matching all three frames pairwise is assumed to be correct and labelled as the proper matching. If this is not the case, the direct matching between F_{k-1} and F_{k+1} is labelled as correct.

By applying this algorithm to each tripled of consecutive frames, a trajectory is built step by step for each frame.

3.1.5 Experimental Results The authors compare the proposed method with 5 different algorithms: CG[14], SK[15], CV[16], SS[17] and JA[18]. CG is a model-based approach, which is used to show that without any prior knowledge about the motion model this type of algorithm yields unsatisfying results. The other four methods fall under the category detection-based association-algorithms.

While all of these are also cell-tracking algorithms, it should be noted that three algorithms (CG, CV and SS) were presented before 2000 and the newest algorithm is JA, presented in the year 2008.

All algorithms are applied on a not further defined set of 11 video sequences. For the proposed algorithm, the values $q = 90$, $\lambda_1 = 2/3$ and $\lambda_2 = 1/3$ are set.

As a performance measurement, cell tracking accuracy in percent is used. This cell tracking accuracy is defined by Chatterjee et al. as

$$acc = 100\frac{L_{corr}}{L_{total}} \tag{5.18}$$

where L_{corr} is the number of correct links between objects and L_{total} is the number of total links returned by the proposed method. The mean cell tracking accuracy over the eleven different videos as well as the standard deviation is shown in Table 1.

	Proposed	JA	CG	SK	CV	SS
Mean	97.97	98.20	95.66	94.42	81.22	78.31
SD	0.94	1.22	2.39	2.08	5.75	4.70

Tab. 1: Mean cell tracking accuracy of the six algorithms for 11 video sequences

Using only these values, the presented algorithm seems to perform reasonably well. It outperforms CV and SS by over 15% and performs similar to JA. But since these values were acquired using only 11 videos and no further information on those videos were given, it is questionable how much of an improvement this algorithm is. Further experiments using a larger dataset and more recent algorithms are necessary.

3.2 Multi-Target Tracking by Lagrangian Relaxation to Min-Cost Network Flow

Similar to the approach presented in Section 3.1, the algorithm presented in [3] also tries to address the problem that tracking objects by only applying the bipartite matching algorithm to pairs of subsequent frames is a fast, but very inaccurate solution of the problem of tracking multiple targets. But instead of incorporating the past movement over many frames while only directly observing a three frame window like the previous approach, Butt et al. formulate the problem as a min-cost network flow problem.

This algorithm was designed to track pedestrians in a surveillance-feed, but it can be adapted to nearly every imaginable tracking problem since only the object positions are used.

3.2.1 Overview Instead of using bipartite matching between frames to reconstruct trajectories, the problem is formulated using a network flow graph. Then, a min-cost network flow algorithm as presented in 2.2.1 can be used to reconstruct the trajectories of each object. For this to work, five steps are necessary:

1. Extract the positions of each object
2. Construct a flow network describing the tracking problem
3. Add necessary constraints to ensure integrity of the solution
4. After calculating the min-cost flow: Resolve any conflicts that might have occurred
5. Use Lagrangian relaxation to decrease the amount of unmatched objects.

The first step can be done in the same way described in Section 3.1, which enables the algorithm to be used for cell-tracking, even though it was designed with pedestrian tracking on surveillance-camera-videos in mind.

The other four steps will be described in greater detail in the following Sections.

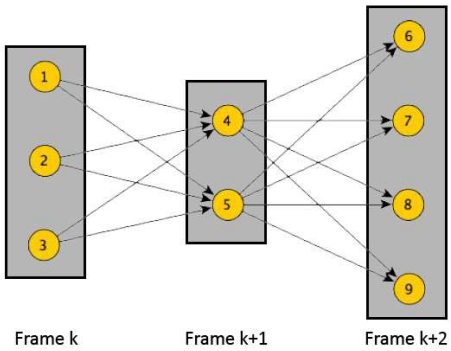


Abb. 5.8: The original scene-graph. [3]

3.2.2 Building the Graph The most basic way to represent the tracking problem over a three frame window using a graph can be seen in Figure 5.8. The grey boxes represent single frames, and all objects found in that particular frame are represented as yellow nodes. Each object from frame k is connected to each object from frame $k+1$.

Each edge in this graph can be represented using a Boolean variable $x_{u,v}$ with $u \in F_k$ and $v \in F_{k+1}$. The goal is to assign these variables so that if $x_{u,v}$ is set to true if the link (u, v) is part of a trajectory.

A new graph can now be constructed. The edges which previously represented a link of objects between two subsequent frames now are represented as nodes, and an edge between two nodes is set if the nodes share an object. Using the graph from Figure 5.8 as an example, the nodes $x_{1,4}$ and $x_{4,8}$ are connected in the newly constructed graph, since 1-4-8 is a coherent path from frame k to frame $k+2$.

The resulting graph is shown in Figure 5.9.

Without any additional constraints, a matching would not result in correct trajectories for every object. For example, if $x14$ and $x47$ are matched, $x24$ and $x34$ are no longer valid matching candidates, since this would lead to object 4 from frame $k+1$ being matched to two objects from frame k, resulting in invalid trajectories. These additional constraints are represented using colored hyperedges to the left and right of the graph. For every set of nodes connected to the same hyperedge, only one can be assigned to a match.

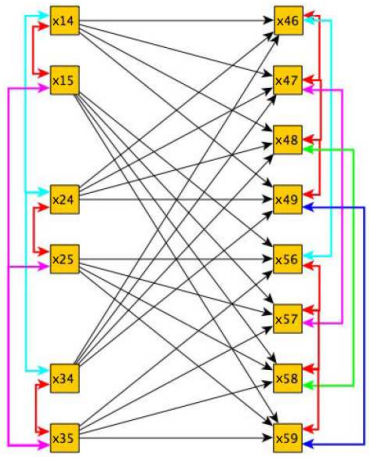


Abb. 5.9: The reformulated graph. [3]

3.2.3 Formulation as a Network Flow Problem Since the goal is to reformulate the tracking problem as a network flow problem, we need to adjust the graph further. All possible matches between two subsequent frames are given by

$$P_k = \{(u, v) \mid u \in F_k, v \in F_{k+1}\} \tag{5.19}$$

We now construct the graph shown in Figure 5.10. Since the problem should be formulated as a min-cost network flow problem, a source and a sink-node are added. Each potential match will be represented using two instead of one node. One node with only incoming connections (so either from the source or from previous frames), one with outgoing edges (to the sink or subsequent frames). The incoming and the outgoing nodes representing a single match are connected. The capacity of this connection is set to 1. Again, we create an edge between the outgoing node of a match in P_k to the incoming node a match in P_{k+1} if they share an observed object in frame $k + 1$.

Finally, edges from the source to all incoming nodes and from all outgoing nodes to the sink are added. Adding edges from the source to the matches in P_{k+1} allows cells that are occluded for a few frames or enter/exit the observation to still get a trajectory assigned.

The resulting graph is a network flow graph, to which a maximum-flow algorithm can be applied to retrieve all potential matches.

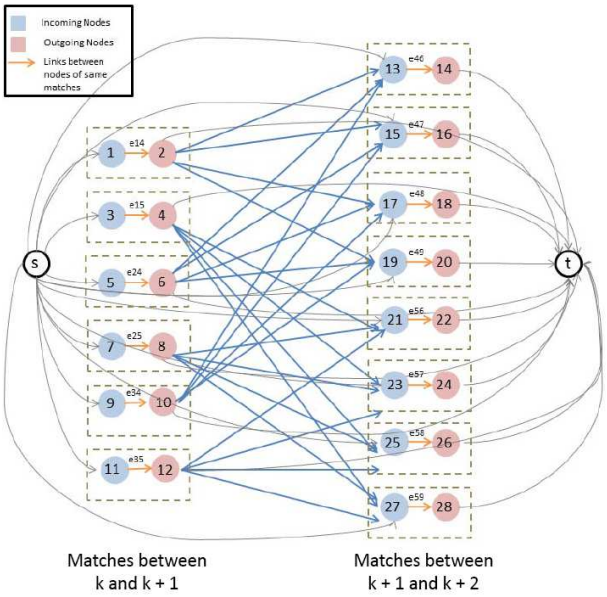


Abb. 5.10: The final constructed flow-graph including additional constraints. [3]

3.2.4 Resolving Conflicting Matches The reformulation of the problem as a network flow problem did however not solve the problem that matches conflicting each other can still occur and need to be resolved. To detect these conflicts, we retrieve all matches that a single object is part of. This conflict set for a single object u from frame k can be described as follows:

$$\mathrm{EC}_k(u) = \{(u, *) \in P_k\} \tag{5.20}$$

Since unique trajectories should be created, only one match from $\mathrm{EC}_k(u)$ can be selected. To decide, which match to select, a cost-function $a_{u,v}$ is formulated based on e.g. spatial proximity and past velocity. Additionally, visual appearance similarity could be incorporated, but when observing cells the visual differences are only minor, making this option infeasible. A simple example for such a cost-function would be e.g. the distance function presented earlier in Equation 5.9.

With the addition of this cost-function, the problem can be rewritten as a linear program. Since the most likely matches should be utilized, the target is to minimize

$$min \sum_{(u,v)\in E} a_{u,v} x_{u,v} \tag{5.21}$$

Additional constraints are that $x_{u,v}$ is a binary variable, incoming and outgoing flow for each node is the same, and only one edge of the conflict set can be activated. So the target Equation 5.21 is subject to:

$$x_{u,v} \in \{0, 1\} \tag{5.22}$$

$$\sum_{(u,w)\in E} x_{u,v} = \sum_{(w,v)\in E} x_{w,v} \qquad \forall w \in V \setminus \{s, t\} \tag{5.23}$$

$$\sum_{(u,v)\in EC_k(u)} x_{u,v} \leq 1 \qquad \forall u \in F(k) \tag{5.24}$$

Since this linear program returns values for all $x_{u,v}$, trajectories can be reconstructed by creating a set of all selected edges:

$$\text{TS} = \{(u, v) \in E | x_{u,v} = 1\} \tag{5.25}$$

Linking up all edges in TS sharing a node results in the trajectories determined by the linear program to be optimal.

3.2.5 Lagrangian Relaxation While this linear program is equal to the original multi-frame tracking-problem and resolves conflicting matches, it is very inefficient to solve with e.g. the simplex-algorithm due to the high amount of constraints. There is one constraint for every object currently observed, leading to potentially hundreds of constraints.

To create a simpler linear-program that is still equal to the problem but with less constraints, the constraint from Equation 5.24 can be factored into the cost function. This leads to a more complex target-function, but removes nearly all constraints, leading to a problem that is easier to solve.

To achieve this, a set of multipliers $\lambda = \{\lambda_1, ..., \lambda_n\}$ is introduced, where n is the amount of objects observed in frame k. The new target function becomes

$$min \sum_{(u,v)\in E} c_{u,v} x_{u,v} + \sum_{u\in F_k} \lambda_u \left(\left(\sum_{(u,v)\in EC_k(s)} x_{u,v} \right) - 1 \right) \tag{5.26}$$

The term $\sum_{(u,v)\in EC_k(u)} x_{u,v}$ describes how many trajectories the object u is part of in the current solution, which should not be greater than 1. If an object is now assigned multiple times, the factor for this object will increase, leading to a higher cost solution based on the corresponding multiplier λ_u.

Using this new target function, the constraint in Equation 5.24 can be removed from the linear program. But without choosing the right values for λ, this program might lead to invalid solutions, since it only punishes usage of single objects in multiple trajectories, and does not circumvent it in every case.

To still achieve the best solution possible in an efficient matter, the linear program is solved multiple times, while updating λ after every iteration in the following manner:

If object u is not part of more than one track, leave λ_u as it is, otherwise: increase λ_u, leading to a higher cost for this solution. After a certain number of iterations, or if λ did not change, we assume that the current solution is the best available.

Since solutions with conflicting trajectories have only been punished, not prevented, it may still happen that the final solution includes paths sharing a single observed object. To resolve those, the cost of all paths passing through an object are compared, and all but the lowest cost one are discarded.

This is a very greedy way of selection trajectories, but since solutions with conflicts are punished more and more, this algorithm should be able to find optimal solutions with no collisions in most cases, and select the best solution in all cases where conflicts are present.

This algorithm now gets applied to subsequent three frame windows to construct the trajectories for each object step by step.

3.2.6 Experimental Results The authors apply their algorithm to two different datasets, comparing the results to other algorithms for which accuracy measurements on the same videos is available. First, two videos made public by the authors of [19] are used. One image sequence with an average of only 5 tracking targets per frame, called "sparse sequence", and a video with over 20 observed objects per frame, called "dense sequence". Both sequences are 15 minutes long, but neither the frame rate nor the image resolution is further defined.

As a measurement of accuracy, a mismatch percentage

$$mmep = 100 \frac{L_{mme}}{L_{gtotal}} \tag{5.27}$$

is computed. L_{mme} is the amount of incorrect matchings, L_{gtotal} is the total amount of observed matchings. L_{gtotal} was acquired by labelling ground truth matchings manually. To observe the effect of lower frame rates on the mismatch percentage, the video-sequence has been down-sampled to frame rates of 1, 2 and 3 frames per second.

The proposed algorithm is compared to three algorithms: Projection"[20], which projects the graph with hyperedges shown in Figure 5.9 on a simple graph, "Greedy"[19] which utilizes sequential filtering, and Block-ICM[19], which is an ICM-like method. All three are state of the art algorithms, with Projection[20] having been presented as late as 2012. The resulting mismatch percentages are shown in table 2.

	Sparse Sequence			Dense Sequence		
Algorithm	1 fps	2 fps	3 fps	1 fps	2 fps	3 fps
Projection	0.05	0.12	1.40	1.05	2.55	15.51
Greedy	0.00	0.06	1.36	0.12	0.34	6.83
Block-ICM	0.00	0.12	0.80	0.13	0.25	4.13
Proposed	0.00	0.00	0.41	0.10	0.17	1.46

Tab. 2: Mismatch error percentages of the different algorithms on two source videos sampled at different frame rates

Additionally, the authors compared their algorithm to a method using dynamic programming (DP[21]) and a similar approach also utilizing min-cost network flow (MCNF[5]) on two other publicly available video sequences. As a measure of accuracy, the number of mismatches per total number of output observations was used. This can be adapted to the same measurement as shown in Equation 5.18 for better comparison using the following formula:

$$acc = 100 \frac{L_{total} - L_{mme}}{L_{total}} \tag{5.28}$$

where L_{total} is the number of observations returned by the algorithm, and L_{mme} is the number of mismatches. The resulting accuracy measurements are shown in Table 3. While the proposed algorithm outperforms the projection, greedy and block-ICM method on the used image-sequence, the improvement over the DP and MCNF method is very low. According to the authors this is due to a higher rate of false positives of the MCNF method, but additional tests by independent people are necessary to validate the better performance of the proposed method.

Algorithm	TUD	ETHMS
DP	95.83	97.33
MCNF	97.92	98.96
Proposed	98.29	98.48

Tab. 3: Mean tracking accuraccy using Equation 5.18 as measurement

3.3 Integrating Graph Partitioning and Matching for Trajectory Analysis in Video Surveillance

3.3.1 Overview In contrast to the other two presented algorithms, Lin et al. present a tracking algorithm that uses object features to more accurately distinguish between different targets.

Given an input image-sequence, the algorithm can be divided into the following steps:

1. Partition the image into foreground and background using a background modelling algorithm
2. Extract visual features using SURF and MSER feature detection
3. Create composite features consisting of one MSER region and multiple associated SURF points
4. Create a spatial-graph for every image and use graph partitioning to join multiple composite features to single tracking targets
5. Use a temporal graph using these targets as vertices and edges between vertices of subsequent frames based on their visual similarity and proximity
6. For all found trajectories, determine the most likely one by incorporating target-size prediction and geometric information

Abb. 5.11: The original video-footage (left), the extracted foreground-regions (middle), SURF + MSER features found on the silver car marked in the other two images (right).

3.3.2 Feature Extraction Since the input is an arbitrary image sequence without any information about the objects that should be tracked and only very little geometrical information is known about the objects, the first problem that has to be solved is extracting tracking targets and corresponding properties. To achieve this, the first step is to determine whether a pixel in our source image belongs to an object or not. For this purpose, a background-modelling-algorithm is deployed. This background-modelling is based on the assumption that objects of interest are moving, so a pixel that has changed in the last few frames is most likely part of a moving object. Along with the analysis of neighboring pixels to smooth the foreground-regions, a result similar to Figure 5.11 may be computed.

After foreground and background are separated, the SURF and MSER algorithms are applied to only the foreground regions, already filtering out many features that are not associated with any tracking target. The SURF-algorithm (Speeded Up Robust Features) searches for local changes in contrast, which occur for example on corners of objects, where the background has a very different color from the actual object. As an example, the back window of the silver car shown on the right in Figure 5.11 has a dark grey, while the car itself is painted silver, leading to the window corners being detected as important features, indicated by the red crosses. In contrast, the MSER-algorithm (Maximally stable extremal regions) searches for regions with a coherent appearance. Taking the back window as an example again, the whole window is nearly the same color. And nearly independent from the orientation of the car and the lighting situation, the window will always have one coherent color, only the shape of the region and the color may change. This leads to it being detected as a region of interest, as indicated by the blue circles.

3.3.3 Graph Representation Since one object can consist of multiple SURF and MSER features (the car for example consists of 10 MSER-regions), a way to accumulate multiple features to a single target needs to be found.

The first step towards traceable targets is to create an association between the two different used features. These so called composite features are composed of one MSER-region and all SURF-points, which are located inside this region.

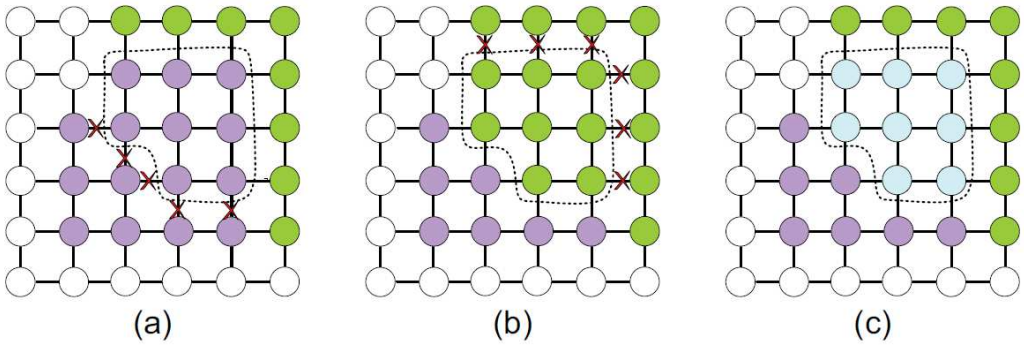


Abb. 5.12: Three different solutions for building single objects in the spatial-graph. Each color indicates one traceable target.

3.3.4 Spatial Graph Partitioning As illustrated by the example of the silver car in Figure 5.11, a single target can consist of a number of composite features. To determine which composite features can be summarized into a single target, a spatial graph is used. Each node corresponds to a single pixel in the source-image and is labelled with an ID determining which composite feature it belongs to. This is determined by testing if a pixel is inside the MSER-Region of all composite features. If its inside none, it is labelled as "background". 4 edges are created for each vertex, connecting it to the 4 neighbouring image-points.

For each edge connecting two different regions a Turn-On-Probability $p_e(u, v)$ is computed

$$p_e(u,v) = \ exp\left(-\frac{K(h(u)\ ,\ h(v)) + K(h(v)\ ,\ h(u))}{C}\right) \tag{5.29}$$

Where $h(u)$ is the histogram of image-features at the pixel corresponding to u including features such as color, optical flow and the orientation gradient. $K(h(u)\ ,\ h(v))$ is the Kullback-Leibler divergence between the two histograms $h(u)$ and $h(v)$ and C is a scaling constant.

All vertices which share the same label form a connected cluster (called CC for simplicity). For each CC all clusters connected by an edge are searched. Every CC that shares an edge with another CC is possibly part of the same tracking target. To determine, which CC's should be connected to a single target, each edge connecting two CC's is either "turned on" or "turned off". To determine which state is assigned to an edge $e = (u, v)$, the turn-on probability from Equation 5.29 is sampled following the Bernoulli probability.

One CC that has a neighbouring CC is now selected at random. For this CC, there are 2 possible scenarios:

- It corresponds to a tracking target
- It is only part of a tracking target

Using Figure 5.12 as an example, there are three possibilities for the CC marked blue in Image (c). Either it is part of the object indicated by green vertex-labelling, or it is part of the violet object, or it is its own object. If none of the edges connecting blue to green or violet are assigned a "turn-on"-label, it will continue to stand alone. The transition from state A to state B, where blue is labelled as belonging to green, is assigned to a probability α if at least one connecting edge is turned on.

$$\alpha(A \to B) = min\left(1, \frac{Q(B \to A)}{Q(A \to B)}\right) \tag{5.30}$$

where

$$\frac{Q(B \to A)}{Q(A \to B)} = \frac{q(CC^B|B)q(L(CC^B)}{q(CC^A|A)q(L(CC^A)} \tag{5.31}$$

and

$$\frac{q(CC^B|B)}{q(CC^A|A)} = \frac{\sum_{(u,v)\in\mathbb{C}_B}(1-p(u,v))}{\sum_{(u,v)\in\mathbb{C}_A}(1-p(u,v))} \tag{5.32}$$

where $\mathbb{C}_A$ are the edges turned off in state A, CC^A and CC^B are the blue and green CC's, $L(CC^A)$ is the function determining the label of CC^A. The return value of $q(L(CC^A))$ is not further defined by the authors. This probability is calculated for all possible new labelling states, and the one maximizing $\alpha(A \to B)$ is picked. The cluster differentiating states A and B is relabelled to transition into state b. This algorithm is run iteratively a number of times to ensure that all CC's are properly labelled.

3.3.5 Temporal Graph Matching Now that the individual tracking-targets in each frame have been extracted, the trajectories for each object need to be linked through all available frames. For this purpose, a second graph is constructed.

In this temporal graph, each node depicts a single target in a single frame, and an edge between two targets always links two different objects from subsequent frames and is labelled with their matching likeliness. If an object has no match in the previous frame, it is assumed that it has just entered the frame, and if an object has no match in the following frame, it is assumed to have exited the frame.

The final matches are now selected using Bayesian analysis. Bayesian interference derives posterior probability for an event based on prior probability and a likelihood-function, which is derived from a probabilistic model for the data that is being observed. The new probability is calculated using Bayes' rule, which states the following:

$$\mathrm{P(X|Y)} = \frac{\mathrm{P(Y|X)} * \mathrm{P(X)}}{\mathrm{P(Y)}}$$

Where | means given, X is the hypothesis whose probability can be affected by data, $P(X)$ is the prior probability, $P(X|Y)$ is the posterior probability, which is the probability of X after Y has been observed. $P(Y|X)$ is the likelihood-function, depicting the compatibility of the evidence with the given hypothesis. $P(Y)$ is depicting the model evidence. The hypothesis X is not included, since it is constant for all hypothesis being tested[22].

In this case, the posterior probability is defined as

$$\mathrm{W}^*_{[0,\tau]} = \mathrm{max_w\ p(I_{[0,\tau]}|W_{[0,\tau]};\beta) * p(W_{[0,\tau]}|\theta)}$$

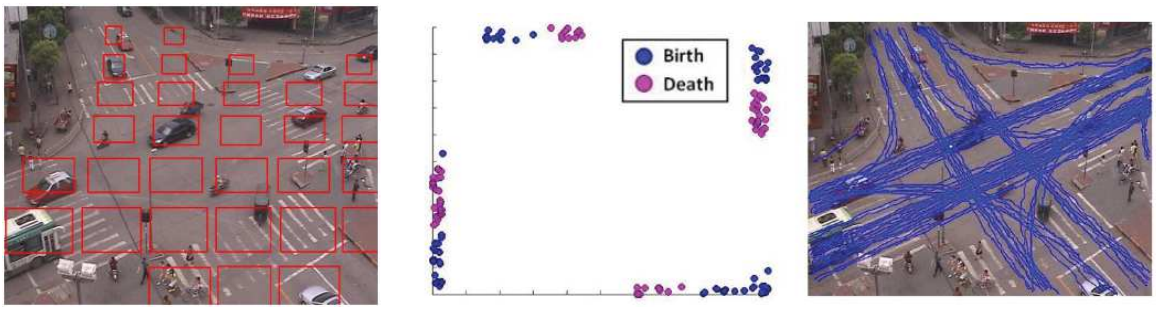


Abb. 5.13: The reference information included in the bayesian model. The anticipated target-size as indicated by the size of the red squares (left), most likely spawn/death-points (middle) and the used reference-trajectories (right). [23]

Where β is a parameter for the likelihood-model, and θ is a parameter for the prior-model. The prior probability $p(W_{[0,\tau]}|\theta)$ is calculated based on scene-parameters such as the camera-orientation and its correlation to target-object-size (as shown in Figure 5.13 on the left), and the most likely entry/exit-points (shown on the right). The scene-information necessary needs to be provided by the user as reference-data.

The likelihood-function $p(I_{[0,\tau]}|W_{[0,\tau]};\beta)$ is including two aspects: the region appearances in accordance with the background-model and the consistency of objects along its associated trajectory.

3.3.6 Experimental Results The performance of the presented system is evaluated using three different publicly available video databases (TRECVID, PETS, LHI). The TRECVID and PETS database shows pedestrians in a pedestrian zone, while the LHI database is showing cars at an intersection. A total of 24 different videos were selected from those databases. A total of 31 thousand video frames at a resolution of 352 x 288 are processed, including more than 900 individual targets throughout the sequences. To measure the precision of each algorithm, four different measures are introduced. "Recall", which is the ratio of correctly matched targets and total ground-truth targets on a per frame basis, "Precision", which is the ratio of matched targets and total output targets of the algorithm in each frame, "FA/Frm" which is the number of falsely detected targets per frame, and "SwitchIDs", which is the amount of times that a track gets assigned to a different object in between frames.

The results of the proposed method are compared to three algorithms: Zhao et al.[24], where a model-based approach is used to interpret the image observations by multiple partially occluded human hypotheses. Since it was designed to track pedestrians only, it could not be applied to the LHI database. The other two algorithms are Huang et al.[25] and Leibe et al.[26] which were both designed for arbitrary surveillance situations. The resulting measurements are showin in Tables 4, 5 and 6. The proposed method outperforms the competing algorithms in all three databases and using all four accuracy measurements.

Method	Recall	Precision	FA/Frm	SwitchIDs
Zhao et al.[24]	73.2%	72.6%	1.27	14
Leibe et al.[26]	79.1%	73.4%	1.51	10
Proposed	91.3%	86.1%	0.84	7

Tab. 4: Accuraccy measurements of the different algorithms on the LHI-database

Method	Recall	Precision	FA/Frm	SwitchIDs
Zhao et al.[24]	76.2%	72.7%	1.31	12
Huang et al.[25]	69.1%	63.1%	1.82	13
Leibe et al.[26]	78.9%	69.4%	2.01	9
Proposed	83.3%	79.4%	0.72	7

Tab. 5: Accuraccy measurements of the different algorithms on the TRECVID-database

Method	Recall	Precision	FA/Frm	SwitchIDs
Zhao et al.[24]	82.4%	79.7%	0.92	18
Huang et al.[25]	71.1%	68.5%	1.98	14
Leibe et al.[26]	79.1%	73.1%	1.38	16
Proposed	87.7%	82.9%	0.82	8

Tab. 6: Accuraccy measurements of the different algorithms on the PETS-database

4 Comparison and Discussion

All three presented algorithms are solving the same problem - tracking multiple objects in a multi-frame-video-sequence - but approach the problem differently.

The first algorithm presented in Section 3.1 is the only algorithm specifically designed to be applied in the medical field. Because of this it focuses heavily on using only little data (only the position) from each object while still maintaining a high accuracy. Due to this, the algorithm can be adapted to any scenario where tracking of certain objects is necessary, the only part that needs to be adapted is the position-extraction of the observed objects. While no performance measurements are given, the execution time still is polynomial due to bipartite matching being solvable efficiently. Chatterjee et al. claim that the algorithm is interactive for images with less than a hundred cells in frame. While it outperforms the algorithms it was compared to, the newest algorithm (Jaqaman et. al[18]) performs very similar on the video sequences selected by Chaterjee et al.. Additionally, there are no further accuracy measurements available. It was only applied to 11 different videos, which are not further defined. Because of this, it is unclear how well the algorithm will perform on arbitrary video-sequences.

An advantage of this algorithm is the low amount of parameters that need to be user-defined. Only the factors q, λ_1 and λ_2, controlling the influence of the average motion-term and the past-motion term respectably can be adapted. In case of the video sequences selected by the authors, values of$q = 80$, $\lambda_1 = 2/3$ and $\lambda_2 = 1/3$ lead to the most accurate results. If these values perform similar well on arbitrary videos, no user-interaction would be required.

The second algorithm - described in Section 3.2 - was not designed with cell-tracking in mind. But since it is also only using the object-positions and no visual features, it could also be applied for nearly any tracking-situation. Also, by adapting the cost-function used to label different matches, visual similarity could be included to track more complex objects than cells. According to the comparisons given by the authors, the algorithm seems to outperform other state-of-the-art methods. But since it was only applied to 2 different video sequences, it is unclear how well it would perform on arbitrary video sequences. Concerning usability, this algorithm is on par with the first algorithm. No real input parameters are necessary to produce convincing results. Additionally, since it uses linear programs to solve the trajectory-linking problem, the implementation is comparably simple. Linear programs find application in many different fields and are very well researched. A 200 frame sequence with roughly 10-15 pedestrians in frame takes about 1.43 seconds to analyse using an implementation in MATLAB, but an optimized implementation for example in C++ could reach even faster times, meaning it could be applied in real-time scenarios.

The last presented algorithm can only be applied in areas where complex objects need to be tracked since enough SURF and MSER features need to be found to define a traceable object. The additional incorporation of scene-information like expected object-size and exit-/entry-points make it a very inflexible algorithm where a lot of effort is necessary to configure it right. Due to the high amount of data that is included, the computation-times are comparably slow as well. The feed of a surveillance-camera using a resolution of only 352 x 288 pixels can be processed with 15 frames per second on a 3 GHz Dual-Core. It can be assumed that with further optimization, application in a real-time scenario would be achievable.

From the presented methods, this one had the most diverse set of accuracy measurements. Four different measurements were included, the algorithm was compared to three other algorithms with comparable complexity, and was applied to 24 different video-sequences. Due to this, the accuracy improvement over comparable methods claimed by the authors seems well reasoned.

4.0.7 Problems of Understanding The presented algorithms make use of many different techniques, and while the motivation behind using certain techniques is often quite clear, in some cases vital information is missing, which is necessary to create a implementation of these algorithms.

1. In the algorithm using bipartite graph matching, a match score $m(u, v)$ is introduced to resolve conflicting matches after linking trajectories through three frames. But it is never explained what exactly this match-score was based on, and in what scope it should operate. It would be consistent if the match-score is higher when the traced object and its predicted position for the same frame are closer together, but it is never stated how it is calculated and why the absolute value of the calculated selection parameters needs to be computed. Providing a concrete example for a proper match-score-function would have helped to understand the trajectory-selection part of this algorithm.

   Additionally, the authors compare β' and β'' directly, even though β'' is the sum of two match-scores. If the match-score should be independent of the time between frames, comparing β' and $0.5\beta''$ would be the better approach.

2. The algorithm using a min-cost network flow has a similar flaw. A cost function is used to rate every possible edge based on “appearance similarity and smoothness constraints”[3], but it is never stated how these can be obtained or what kind of smoothness constraints were used. Again, a concrete function would have helped to clear this up.

3. The same algorithm also uses a total of three different graph-representations to finally construct a linear program to solve the problem. While their approach gets easier understandable this way, it is not clear why all these sidesteps are taken, since only little of it is used in the final program used to calculate a solution.

4. When introducing the Lagrangian relaxation to the program in the same algorithm, parameters λ are introduced. It is stated how these values are updated every iteration of the algorithm, and how they are used as stopping criteria for the algorithm, but how these values are initialized is never

explained. Since the initialization can have a major impact on the amount of iterations needed to find the best solution, this is an important step and should have been briefly mentioned how to find the right initialization values.

5 Conclusion

While all algorithms have different advantages and the scenarios in which they can be applied differ a lot, they all manage to solve the problem of multi-object-tracking in a video-feed with their chosen scenario constraints.

Although it would be nearly impossible to create an implementation of the algorithms due to the missing functions discussed earlier, a quite detailed idea of how the algorithms work can be obtained. Especially the first two algorithms are also very user friendly, since they require very little user-configuration to be used. Due to this, they are also usable in areas where no technical assistance is available while producing data that is easily understandable. Also it performs well enough to analyze and track hundreds of cells simultaneously. If new methods to track individual cells in live organisms are developed, these methods could be used to gain a better understanding of many diseases and fundamental biological processes, which may be able to help us cure diseases like metastatic cancer, or at least making the creation of an antidote easier.

With the rapid increase in computational power available, more and more improvements to the already very accurate tracking-algorithms become feasible. The algorithm Lin et al. propose is making use of this power by creating models for the scene geometry to further improve their results, showing one possibility of how to use this power while still staying close to a real-time algorithm. With the increase in precision of microscopic imaging techniques, a possible improvement to the algorithms would be to incorporate a 3-dimensional model of the cells. This could yield additional information about the cell-orientation, which could be used as additional object-feature.

So while the current algorithms are already very accurate (with detection rates of over 98% for the full trajectory), there is still a bit of room for possible improvements.

A Appendix

A.1 Symbol Table

Symbol	Meaning
k	Frame index
$F(k)$	Feature-vectors of objects in Frame k
T	Number of frames of input sequence
$E_{k,k+1}$	All edges between F_k and F_{k+1}
$G = (V, E)$	Graph with nodes V and edges E
u, v	Nodes $\in V$
s	Source node of a network graph
t	Sink node of a network graph
$c(u, v)$	Capacity of edge (u, v)
ρ, ρ'	Path through flow network
$d(u, v)$	Euclidean distance between u and v
$d_{ex}(u, v)$	Euclidean distance between u and v including cell-radius
$\overline{d}$	Average movement factor
M	Minimum weight maximum bipartite matching of F_k and F_{k+1} based on Euclidean distance
$pm(k, i)$	Past movement term for cell i in Frame k
u_{new}	Projected position of u incorporating past and average movement
$d_{new}(u, v)$	Weight of match u and v including past and average movement
$f_{enhance}(x)$	Contrast enhancement function
$f_{thresh}(x)$	Thresholding function
$a(u, v)$	Cost associated with the edge (u, v)
$f(u, v)$	Flow of edge (u, v)
P_k	All possible matches between Frames k and $k + 1$
$EC_k(u)$	Conflicting matches for object u in frame k
$m(u, v)$	Match-score between u and v, determining the quality of the match
λ_1, λ_2	Weighting variables for the past-motion and average movement term
q	Percentage of matches that are averaged for the average movement term
$x_{u,v}$	Boolean variable determining if u and v are matched
$c_{u,v}$	Cost of matching u and v
TS	All trajectory-pieces
λ_s	Lagrangian multiplier for object s
L_{corr}	Number of correct matched trajectory-pieces
L_{mme}	Number of mismatched trajectory-pieces
L_{total}	Total number of trajectory-pieces returned by algorithm
L_{gtotal}	Total number of trajectory-pieces gained by manual labelling
$h(x)$	Image-feature histogram of image-point x
$K(x, y)$	Kullback-Leibler divergence between x and y
$p_e(u, v)$	Turn-on probability of edge (u, v)
β, θ	Weight parameters for likelihood-model and prior-model
CC_A	Connected cluster
$\alpha(A \to B)$	Probability of transitioning from CC-state A to B
$Q(A \to B)$	Proposal probability of state transition $A \Rightarrow B$
$\mathbb{C}_A$	All edges turned of in state A

Literaturverzeichnis

[1] Nicola Bellotto and Huosheng Hu. Multisensor-based human detection and tracking for mobile service robots. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):167–181, 2009.

[2] Pan Pan and Dan Schonfeld. Video tracking based on sequential particle filtering on graphs. *Image Processing, IEEE Transactions on*, 20(6):1641–1651, 2011.

[3] Asad A Butt and Robert T Collins. Multi-target tracking by lagrangian relaxation to min-cost network flow. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1846–1853. IEEE, 2013.

[4] Cor J Veenman, Marcel JT Reinders, and Eric Backer. Resolving motion correspondence for densely moving points. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(1):54–72, 2001.

[5] Li Zhang, Yuan Li, and Ramakant Nevatia. Global data association for multi-object tracking using network flows. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[6] Bo Yang, Chang Huang, and Ram Nevatia. Learning affinities and dependencies for multi-target tracking using a crf model. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1233–1240. IEEE, 2011.

[7] Jonathan L Gross and Jay Yellen. *Handbook of graph theory*. CRC press, 2003.

[8] Joost-Pieter Katoen. Datenstrukturen und algorithmen vorlesung 19, 2012.

[9] John E Hopcroft and Richard M Karp. An n^5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[10] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[11] Y Cheng, Victor Wu, Robert Collins, A Hanson, and E Riseman. Maximum-weight bipartite matching technique and its application in image feature matching. In *SPIE Conference on Visual Communication and Image Processing*, pages 1358–1379, 1996.

[12] Rohit Chatterjee, Mayukh Ghosh, Ananda S. Chowdhury, and Nilanjan Ray. Cell tracking in microscopic video using matching and linking of bipartite graphs. *Computer Methods and Programs in Biomedicine*, 112(3):422 – 431, 2013.

[13] Prasanna K Sahoo, S Soltani, and AKC Wong. A survey of thresholding techniques. *Computer vision, graphics, and image processing*, 41(2):233–260, 1988.

[14] John C Crocker and David G Grier. Methods of digital video microscopy for colloidal studies. *Journal of colloid and interface science*, 179(1):298–310, 1996.

[15] Ivo F Sbalzarini and Petros Koumoutsakos. Feature point tracking and trajectory analysis for video imaging in cell biology. *Journal of structural biology*, 151(2):182–195, 2005.

[16] Dmitry Chetverikov and Judit Verestói. Feature point tracking for incomplete trajectories. *Computing*, 62(4):321–338, 1999.

[17] V Salari and Ishwar K. Sethi. Feature point correspondence in the presence of occlusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(1):87–91, 1990.

[18] Khuloud Jaqaman, Dinah Loerke, Marcel Mettlen, Hirotaka Kuwata, Sergio Grinstein, Sandra L Schmid, and Gaudenz Danuser. Robust single-particle tracking in live-cell time-lapse sequences. *Nature methods*, 5(8):695–702, 2008.

[19] Robert T Collins. Multitarget data association with higher-order motion models. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1744–1751. IEEE, 2012.

[20] Peter Ochs and Thomas Brox. Higher order motion models and spectral clustering. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 614–621. IEEE, 2012.

[21] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1201–1208. IEEE, 2011.

[22] John K Kruschke. Bayesian data analysis. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(5):658–676, 2010.

[23] Liang Lin, Yongyi Lu, Yan Pan, and Xiaowu Chen. Integrating graph partitioning and matching for trajectory analysis in video surveillance. *IEEE Transactions on Image Processing*, 21(12), 2012.

[24] Tao Zhao, Ramakant Nevatia, and Bo Wu. Segmentation and tracking of multiple humans in crowded environments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1198–1211, 2008.

[25] Chang Huang, Bo Wu, and Ramakant Nevatia. Robust object tracking by hierarchical association of detection responses. In *Computer Vision–ECCV 2008*, pages 788–801. Springer, 2008.

[26] Bastian Leibe, Konrad Schindler, and Luc Van Gool. Coupled detection and trajectory estimation for multi-object tracking. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

Graph-Cut Segmentation in Medical Images for the Seminar Medical Image Processing in winter semester 2013/14

Markus Joppich

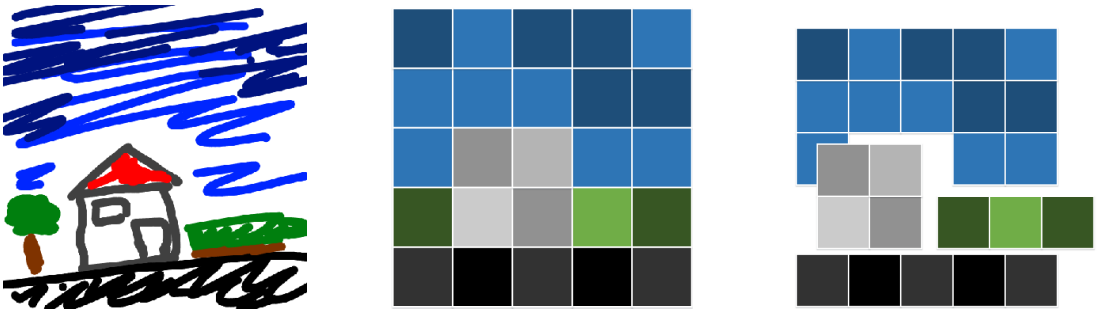
Zusammenfassung

Two papers making use of the graph cut for image segmentation using regions instead of pixels are presented. One implementation introduces automatic seed generation for non-Ground-Glass Opacity (GGO) nodules using a shape index method, the other one applies a hierarchical graph cut for GGO nodules. The theoretic foundations, for instance, random fields theory, statistical region merging, mean-shift clustering and quadratic pseudo-boolean optimization, for both approaches are presented. A self-developed extension for automatic seed generation working for GGO nodules is introduced as well as an adapted hierarchical graph cut method.

Keywords: Graph-Cut, Clustering, Seed, Automatic

1 Introduction

For computer-aided diagnosis systems in medical environments an accurate and fast operation is needed. While semi-automatic solutions are available, the strive is to fully automated frameworks. A frequent task in imaging processes is the division of an image into groups of semantically meaningful regions, for example, to label a part of an image into background and foreground – or even to automatically recognize bones, hearts, etc.. Taking, for example, a painted image of a house standing at a road surrounded by some trees (Figure 6.1(a)). Often only a lower quality, rasterized, version is available (Figure 6.1(b)). Still the optimally wanted segmentation would, for example, be a division into regions for the sky, the road, the house and the plants.



(a) Example image. showing a house, sky, road and some plants.

(b) Rasterized image.

(c) Optimal, multi-label segmentation for the objects in the image.

Abb. 6.1: Exemplary segmentation of the original image 6.1(a) after rasterization 6.1(b). The optimal segmentation 6.1(c) creates segments for the road, house and plants.

There are many algorithms for solving the image segmentation problem but each having different drawbacks. Two classes of these algorithms are extension algorithms and higher order learning schemes. A representative of the first class is the *Random Walker* algorithm which often estimates the target segment too large (over-estimation). The other class, for instance, including the application of the Graph-Cut (GC) on the pixels directly, often has problems with finding the correct boundaries [1].

Recently, more algorithms use a pre-clustering step creating regions of homogeneous properties, using a GC to solve the segmentation problem. Two of such algorithms are presented in the base papers by Tsou

et al. [1] and Ye et al. [2]. Outgoing from an original image, both algorithms use a different framework to finally use the GC for image segmentation. While Tsou et. al. [1] use Statistical Region Merging for finding hierarchical regions of similar properties , Ye et. al. [2] first extract Joint-Spatial Intensity Shape (JSIS) features which are then clustered by an adapted Mean-Shift Clustering (MSC). For both frameworks the segmentation is performed using a GC. In contrast to Tsou, where seeds are manually supplied, Ye automatically retrieves seeds for foreground and background segmentation.

2 Theoretical Foundations

2.1 Segmentation Problem

The segmentation problem often is a problem of finding regions with similar properties. This can also be formulated as labeling problem (Definition 2.1).

Definition 2.1 (Labeling Problem) *Assign a label from the label set $\mathcal{L} = \{x_1, \ldots, x_n\}$ to each of the sites $F_i\mathcal{S}$.*

In Figure 6.2, for instance, three different sites are given, described by their shape. For each of these sites a color-label can be assigned (red or green). The two outer cases represent a proper labeling, while in the middle no label can be assigned to the circle. Therefore, a solution for approximated labeling will be presented in 2.4. For the remainder of this paper, the Labeling Problem reflects the binary labeling problem where $\mathcal{L} = \{0, 1\}$.

Abb. 6.2: Possible labelings for $\mathcal{S} = \{pentagon, rectangle, circle\}$ with $\mathcal{L} = \{red, green\}$. In the two border cases a full labeling for all objects was found. In the middle case, a label for each the pentagon and rectangle could be assigned, but the circle remains unlabeled - only a partial labeling was found.

2.2 Random Fields

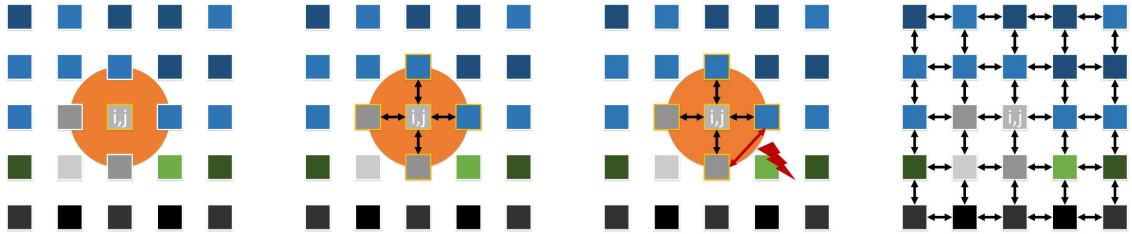
Random fields (Definition 2.2) can be used as an image representation for the segmentation problem. More intuitively, there is a one-to-one mapping between sites in the random field and pixels in an image (Figure 6.3). Each pixel represents a site in $F_i \in \mathcal{S}$. The probability $P(F_i = f_i)$ that the pixel F_i takes label f_i can be seen as the probability that F_i belongs to segment f_i. An assignment of a label to each site is also called a configuration or labeling.

Definition 2.2 (Random Field [3]) *Let $F = \{F_1, \ldots F_m\}$ be a family of random variables defined on the set $\mathcal{S}$ in which F_i takes a value $f_i \in \mathcal{L}$. The family F is called a* random field. *For the discrete label set $\mathcal{L}$, the probability that random variable F_i takes label f_i is denoted as $P(F_i = f_i) = P(f_i)$.*

The set of pixels in an image defines a regular lattice. Therefore, a neighborhood, restricted to four neighbors, is defined as in Definition 2.3 [3]. In most cases, the euclidean distance is used as distance function *dist*.

Definition 2.3 (4-Neighborhood [3]) *The set $\mathcal{N}_i = \{j \in \mathcal{S} | (dist(i,j)^2) \leq r, i \neq j, i \in \mathcal{S}\}$ for a site $i \in \mathcal{S}$ is called the* 4-Neighborhood *of i (with radius $\sqrt{r}$). A site is not neighboring to itself ($i \notin \mathcal{N}_i$) and the relationship is mutual ($i \in \mathcal{N}_j \Leftrightarrow j \in \mathcal{N}_i$).*

From this it follows, that for the selected site (i, j) (Figure 6.3(a)), the four nearest neighbors lying in the orange circle belong to its neighborhood. This relation can be expressed by bi-directional edges (mutuality) (Figure 6.3(b)). Thus each neighborhood also consists of a set of cliques, which are restricted to 1-cliques and 2-cliques. n-cliques with $n > 2$ are not existing, because no diagonal relations exist (Figure 6.3(c)). Finally the Neighborhood System $\mathcal{N} = \{\mathcal{N}_i | i \in \mathcal{S}\}$ for the image is created (Figure 6.3(d)). It becomes clear that $(\mathcal{S}, \mathcal{N}) \stackrel{\triangle}{=} G(V, E)$ constitutes a graph with $V = \mathcal{S}$ and $(i, j) \to (i', j') \in E$ if there exists such a 2-clique in $\mathcal{N}$.



(a) Pixel (i, j) is selected and in orange the iso-surface of elements with a distance less than 1 from (i, j) is shown.

(b) Yellow borders denote the pixels in the neighborhood of (i, j). Their relation is expressed using arrows.

(c) Since there are no diagonal relations, no 3-cliques can be constructed.

(d) Neighborhood-System $\mathcal{N}$ for the image from 6.1(b).

Abb. 6.3: Determination of the Neighborhood system (d) for the original image (Figure 6.1(b)).

It has been shown that a random field is useful for image segmentation. The neighborhood system defined on the random field can be used to model adjacency of image pixels. Therefore, the Markov Random Field (MRF) is used (Definition 2.4).

The optimal configuration for the random field maximizes the probability $P(F = f)$. Therefore, a way for maximizing the *a posteriori* probability $p(f|d)$ needs to be found. As an intermediate step first the *a posteriori* probability needs to be derived.

Definition 2.4 (Markov Random Field (MRF)) *A random field F is called a MRF on $\mathcal{S}$ with respect to neighborhood system $\mathcal{N}$ if and only if*

$$P(f) > 0 \quad \forall f \in F \qquad \textit{positivity}$$
$$P(f_i | f_{\mathcal{S} \setminus \{i\}}) = P(f_i | f_{\mathcal{N}_i}) \qquad \textit{Markovianity}$$

where $f_{\mathcal{N}_i} = \{f_j | j \in \mathcal{N}_i\}$.

Definition 2.5 (Gibbs Distribution) *A Gibbs distribution has the form $P(f) = Z^{-1} \cdot e^{-\frac{1}{T}U(f)}$ with the* partition function $Z = \sum_{f \in \mathbb{F}} e^{-\frac{1}{T}U(f)}$, temperature $T(= 1)$ *and* energy function $U(f) = \sum_{c \in \mathcal{C}} V_c(f)$.

Definition 2.6 (Gibbs Random Field (GRF)) *A set of random variables F is called a GRF on $\mathcal{S}$ with respect to $\mathcal{N}$ if and only if its configurations obey a Gibbs distribution.*

For the intermediate step, the *Hammersley-Clifford-Theorem* shows the equivalence of MRF and GRF [3]. The consequence from this equivalence is, that a local property used for expressing adjacency, the *Markovianity*, and a global property, the *Gibbs distribution*, define the same *a posteriori* probability for the random field. Furthermore, the Gibbs distribution can be used to encode *a priori* probabilities for the sites in the random field.

Theorem 2.1 (Hammersley-Clifford-Theorem) *Markov Random Field and Gibbs Random Field are equivalent.*

The GRF is equivalent to a MRF.

Let $P(f)$ be a Gibbs distribution on $\mathcal{S}$ with respect to the neighborhood system $\mathcal{N}$. We assume that $P(f) \geq 0$. The conditional probability is given by

$$P(f_i|f_{\mathcal{S}\setminus\{i\}}) = \frac{P(f_i, f_{\mathcal{S}\setminus\{i\}})}{f_{\mathcal{S}\setminus\{i\}})} = \frac{P(f)}{\sum_{f_i'\in\mathcal{L}} P(f')} = \frac{e^{-\sum_{c\in\mathcal{C}} V_c(f)}}{\sum_{f_i'} e^{-\sum_{c\in\mathcal{C}} V_c(f')}}$$

with $f' = f \setminus \{f_i\} \cup \{f_i'\}$ for the GRF. Dividing $\mathcal{C}$ into sets of cliques containing i ($\mathcal{A}$) and not ()$\mathcal{B}$):

$$P(f_i|f_{\mathcal{S}\setminus\{i\}}) = \frac{e^{-\sum_{c\in\mathcal{A}} V_c(f)} \cdot e^{-\sum_{c\in\mathcal{B}} V_c(f)}}{\sum_{f_i'}[e^{-\sum_{c\in\mathcal{A}} V_c(f')}][e^{-\sum_{c\in\mathcal{B}} V_c(f')}]} = \frac{e^{-\sum_{c\in\mathcal{A}} V_c(f)}}{\sum_{f_i'} e^{-\sum_{c\in\mathcal{A}} V_c(f')}}$$

because $V_c(f) = V_c(f')$ for $c \in \mathcal{B}$. It can be seen that $P(f_i|f_{\mathcal{S}\setminus\{i\}})$ only depends on labels which are in the neighborhood of i. This is the *Markovianity* of the MRF. For the proof that a MRF is also a GRF see [4].

From the MRF-GRF equivalence the Maximum a-posteriori (MAP) probability can be derived. To derive the conditional probability the Bayesian Inference in the discrete case is used. Then, the MAP is determined by

$$\begin{aligned}
\hat{f} = \arg\max_f P(f|d) &= \arg\max_f P(d|f) \cdot P(f) && (6.1)\\
&= \arg\max_f \sum_{i=1}^{m} \log P(d_i|f_i) + \log P(f) && (6.2)\\
&= \arg\max_f \sum_{i=1}^{m} -T \cdot U(d_i|f_i) + (-T) \cdot U(f) && (6.3)\\
&= \arg\min_f \sum_{i\in\mathcal{S}} \lambda_1 \cdot V_1(i) + \sum_{c\in\mathcal{C}} \lambda_2 \cdot V_2(c) && (6.4)
\end{aligned}$$

where usually $T = 1$ and λ_1, λ_2 depend on the Gibbs distribution. If V_2 is chosen such that $V_2 = 0$ for 1-cliques, the conditional probability of the Gibbs energy function determines *unary* energies per site and the Gibbs distribution defines *pairwise* energies between adjacent sites [3].

For maximizing the *a posteriori* energy of the segmentation problem, an energy of the form $\sum_{i\in\mathcal{S}} V_1(d_i|x_i) + \sum_{i\in\mathcal{S}} \sum_{j\in\mathcal{N}_i} V_2(x_i, x_j)$ needs to be minimized. This then defines the optimal labeling and a solution to the segmentation problem.

In 2001 the Conditional Random Field was introduced by Lafferty, McCallum and Pareira [5]. The CRF directly encodes the *a posteriori* probability and therefore the derivation via the MRF-GRF equivalence is not needed. Also the CRF uses the observation data for the pairwise terms, and the unary terms get the full observation data. While the first can be modeled in the GRF, the latter is unique to the CRF [3].

2.3 Solution to Maximum a-posteriori-Markov Random Field

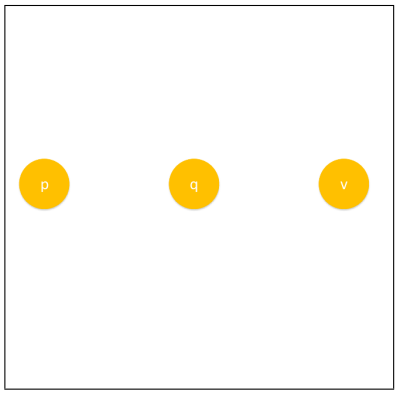
The MAP-MRF can be solved via the minimum graph cut. While in Section 2.2 it was shown that the set of sites and the neighborhood system constitutes a graph, this graph can be extended to a *s-t-network*.

Definition 2.7 (s-t Network) *Let $G = (V, E)$ be a directed graph where there exist a source $s \in V$ and a sink $t \in V$ ($s \neq t$) and each edge $(i, j) \in E$ has a capacity $c(i, j) \geq 0$. G is called a* s-t network.

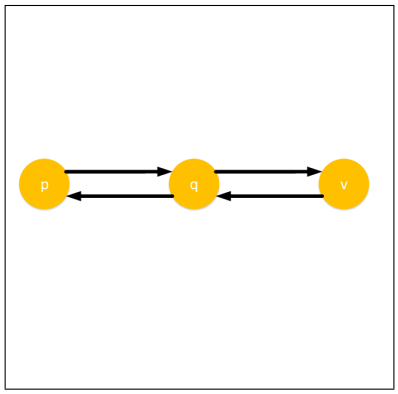
The graph $G'(V, E) = (\mathcal{S}, \mathcal{N})$, which is constituted by the MRF via the Neighborhood System $\mathcal{N}$, can be extended to a s-t-network $G = (V, E)$ by choosing $V = \mathcal{S} \cup \{s, t\}$ and $E = \mathcal{N} \cup \{(s, i), (i, t)|i \in \mathcal{S}\}$. The construction is presented in Figure 6.4.

Picard and Ratliff [6] have shown that the GC minimizes functions of the form $\sum_{j=1}^{n} p_j \cdot x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} x_i x_j$ where x_i is the label $\in \{0, 1\}$ of node i and p_j, q_{ij} are real valued constants and $q_{ij} \geq 0$. Greig, Porteous and Seheult proved that the GC is a solution to the MAP-MRF in the binary case [7].

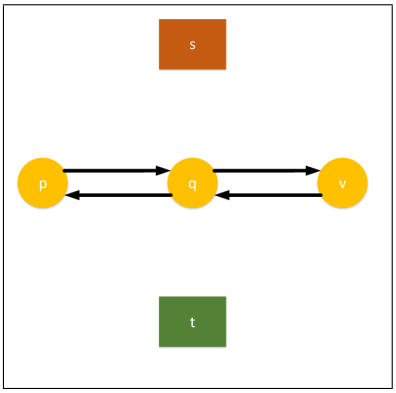
The minimum GC can be found using the Ford-Fulkerson theorem stating that for calculating the minimum-cut it is sufficient to calculate the maximal flow of the MRF-s-t-network. To find the minimum cut, the zero-edges in the residual network then can be cut.



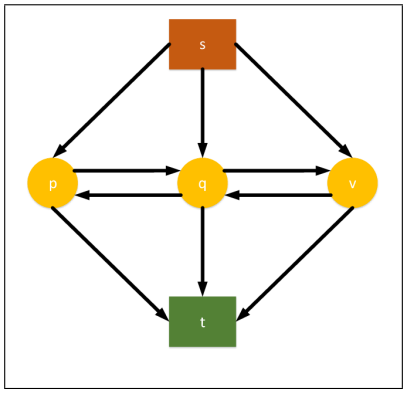
(a) Set of nodes p, q and v representing the random variables in the MRF.



(b) Neighborhoud system for the random variables p, q and v.



(c) Addition of nodes for the source s and sink t.



(d) Edges connecting the source with all nodes representing the random variables, and the random variable nodes connected with the sink.

Abb. 6.4: Construction of a s-t-network from a MRF by adding source s, sink t and directed edges.

Theorem 2.2 (Ford-Fulkerson Theorem) *Given a s-t-network G(V,E), then the value of the maximal flow is equal to the capacity of the minimal cut.*

The Ford-Fulkerson algorithm can be used for calculating the minimal flow with runtime complexity $O(|E| \cdot |f_{max}|)$ (f_{max} is maximal flow) in the integral case. For real-valued edge capacities this algorithm may not terminate. Other algorithms are the Edmonds-Karp algorithm ($O(V \cdot E^2)$) and the Push-Relabel (Goldberg-Tarjan) algorithm ($O(V^2 \cdot E)$) which is commonly used nowadays.

On the way to more generalism, the class $\mathcal{F}^2$ of *pseudo-boolean* energy functions is introduced.

Definition 2.8 (Class $\mathcal{F}^2$ of pseudo-boolean energy functions) *Pseudo-boolean functions* $\mathbb{B}^n \to \mathbb{R} \cup \{+\infty\}$ *of the form* $E(\mathbf{x}) = E(x_1, \cdots, x_n) = \theta_{const} + \sum_{p \in \mathcal{S}} \theta_p(x_p) + \sum_{(p,q) \in \mathcal{N}} \theta_{pq}(x_p, x_q)$ *belong into the* class $\mathcal{F}^2$. *These classes can be represented as a single vector:* $\theta = \{\theta_\alpha | \alpha \in \mathcal{I}\}$ *where* $\mathcal{I} = \{const\} \cup \{(p,i)\} \cup \{(pq,ij)\}$ *for* $p, q \in V$ *and* $i, j \in \mathcal{L}$.

E *is in normal-form if:* $\min\{\theta_{p,0}, \theta_{p,1}\} = 0$ *and* $\min\{\theta_{pq,0j}, \theta_{pq,1j}\} = 0 \ \forall (p,q) \in E$.

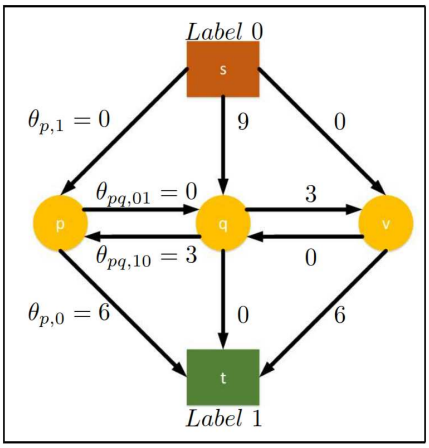
Definition 2.9 (Reparameterization of E) *If two parameter vectors* θ *and* θ' *define the same energy function (i.e.* $E(\mathbf{x}|\theta) = E(\mathbf{x}|\theta')$ *for all configurations* x*, then* θ *is called a* reparameterization *of* θ'*, and the relation is denoted by* $\theta \equiv \theta'$.

Definition 2.10 (Submodularity) *A function* $E(\cdot)$ *of class* $\mathcal{F}^2$ *is* sub-modular*, if and only if every pairwise term* θ_{pq} *satisfies* $\theta_{pq}(0,0) + \theta_{pq}(1,1) \leq \theta_{pq}(1,0) + \theta_{pq}(0,1)$.

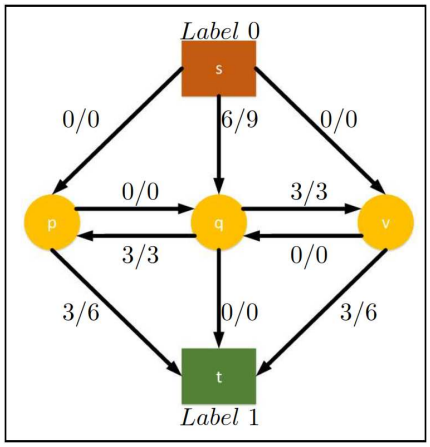
Definition 2.11 ($\mathcal{F}^2$ Theorem: Graph-Representability [8]) *Let* $E(\mathbf{x})$ *be a function of* n *binary variables from the class* $\mathcal{F}^2$. E *is* graph-representable *if and only if each term* θ_{pq} *is* submodular.

Every energy function E can be brought into normal-form using the concept of reparameterization (Definition 2.9). Submodularity (Definition 2.10) ensures that only non-negative edges exist in the graph constructed for E. If E is not submodular, it still can be represented as a graph, however the energy minimization using GC will fail.

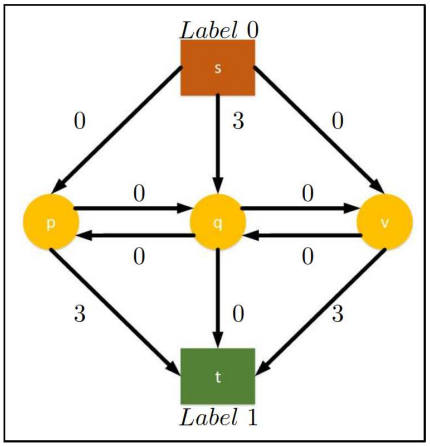
The binary segmentation of a 3 pixel image is performed in Figure 6.5. The used energy θ is already in normal-form and the final segmentation can be derived by cutting the 0 edges in the residual flow network.



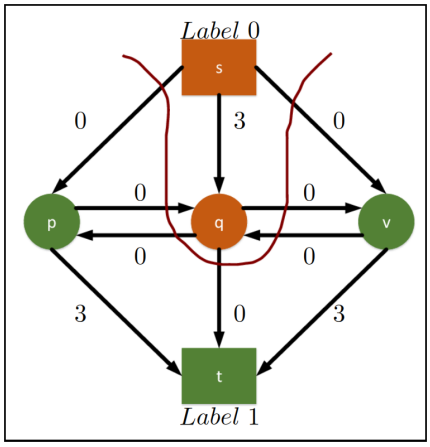
(a) s-t-network corresponding to the MRF-s-t-network creation as in Figure 6.4(d). On the edges the θ for the corresponding energy function are annotated. Notice that E is in normal form.



(b) The maximal flow for the s-t-network is calculated. The flow from p, v to the sink is restricted by the flow from s to q.



(c) The residual network for the maximal flow.



(d) Resulting segmentation from graph-cut. q belongs to label 0 and p, v belong to label 1.

Abb. 6.5: Example for calculating a segmentation using the minimal graph cut derived from the maximal flow.

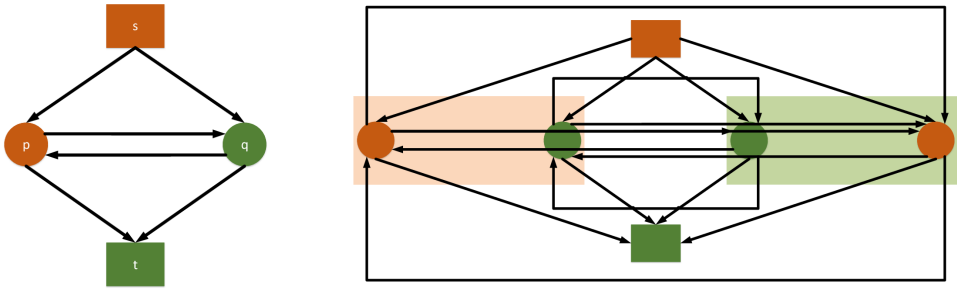
In cases where θ is not submodular, the current approach does not work and an approximate solution is needed.

2.4 Finding Approximate Solutions

The graph cut methods works for minimizing energies when they are regular [8]. In general the minimization problem for the segmentation can be reduced to the maximal independent set problem which is known to be NP hard. The proof is omitted here and can be found in [8].

Instead, the principles of an approximation to the optimal segmentation using Quadratic Pseudo-Boolean Optimization (QPBO) will be presented. This approximation can be used whenever the energy to be minimized is only partly sub-modular, and some parts are super-modular.

This QPBO algorithm is based on the so-called roof duality. Therefore, the existing s-t-network $G = (V, E)$ for the graph cut is extended to $G' = (V', E')$ with $V' = V \cup \{\overline{p} | p \in \mathcal{S}\}$ and $E' = E \cup \{(s, \overline{p}), (\overline{p}, t), (\overline{p}, \overline{q}), (\overline{q}, \overline{p}), (p, \overline{q}), (q, \overline{p}), (\overline{p}, q), (\overline{q}, p) | p, q \in \mathcal{S}\}$.



(a) Original graph $G = (V, E)$ representing a MRF in an s-t-network.

(b) Extending G to $G' = (V', E')$ for roof duality.

Abb. 6.6: Extension of an s-t-network for a segmentation of two regions to roof duality. For each node $p \in V$ of the original graph $G(V, E)$ two nodes $p, \overline{p} \in V'$ are present in the roof duality graph $G' = (V', E')$.

For the roof duality construction it must be noted that if the energy for x_p is submodular, then the energy of variables $\{x_p, x_{\overline{p}}\}$ is also submodular [9]. Furthermore, the idea of *involution*, as $x_{\overline{p}}$ is the negation of x_p, is introduced: $x_{\overline{p}} = 1 - x_p$. If this property holds globally, an optimal solution can be found. If it does not hold, parts of the energy function are not submodular.

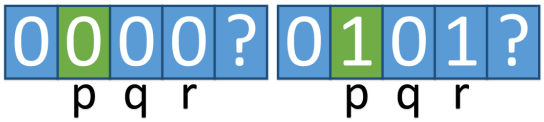
In fact, this already presents the idea of the QPBO approach: the whole s-t-network can be partitioned into a submodular and a supermodular part. For the submodular network a solution can be calculated, as already presented. This partial solution, or partial labeling, can then be copied to the supermodular network, where still some variables remain unlabeled.

More formally, it can also be observed, that two different partitions exist in the residual s-t-network: first a network U^0 exists where p and $\overline{p}$ are in different strongly connected regions. By the Ford-Fulkerson theorem, a flow between the two variables exists, and thus an optimal labeling exists. Secondly, if p and $\overline{p}$ are in the same strongly connected region, no cut for labeling both regions exists. Furthermore, there might be multiple cases where the negation of a variable is in the same strongly connected region. These partitions will be named U^k. It can be seen that for the roof duality network $G = (V, E)$ $V = U^0 \cup \bigcup_k U^k$ holds. If U^k is small, exhaustive search can be used to find an optimal labeling.

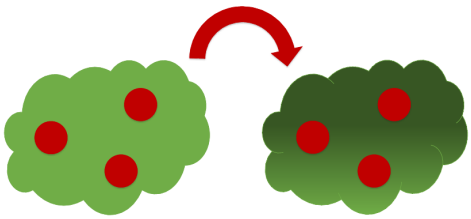
As extension to roof duality, QPBO probing and QPBO improve are introduced [10]. In QPBO probing, a global optimum is searched. Once a complete labeling is found, QPBO can be used to improve this labeling. For QPBO probing, two partial labelings $x^0 = QPBO(E[p \leftarrow 0])$ and $x^1 = QPBO(E[p \leftarrow 1])$ are calculated, where p denotes an unlabeled element. Assuming that q and r are also unlabeled, two cases can arise (Figure 6.7). In the first case, for the label x_q of element q, no matter the choice of x_p, $x_q = 0$. Thus x_q can be fixed. In the second case, either $x_q^0 = x_q^1 = j \Rightarrow x_q^* = j$ or $x_q^0 \neq x_q^1 \Rightarrow x_q^0 = 0 \wedge x_q^1 = 1$ or $x_q^0 = 1 \wedge x_q^1 = 0$. If $x_q^0 = 0 \wedge x_q^1 = 1$, p and q can be contracted as they have the same behavior. In the other case, by involution, set $x_p^* = 1 - x_q^*$, flip x_q and then contract p and q with the new labels. The QPBO improve step works by fixing nodes in $S \subset V$ to labels given by a complete labeling x. Calculate $y = QPBO(E[S \leftarrow x])$ and $y_p = x_p$ for nodes in $p \in S$. The combination z of x and y can be calculated using the $fuse$ operation. It can be proven, that $E(z) \leq E(x)$ and therefore at most the solution remains as bad as before. Finally, $x = z$ is set and the improve step is repeated for a different subset of S.

2.5 Graph Extensions

So far only a binary labeling problem on a flat MRF has been considered. Important other cases are the extension to a multi-label problem as well as the hierarchical case.



(a) Exemplary output for a QPBO probing step at region p.



(b) Visualization of the idea of QPBO improve by fixing the labels for the variables in the red set and calculating a (partially) new labeling for the rest.

Abb. 6.7: (a) shows an exemplary output QPBO probing with $x^0 = QPBO(E[p \leftarrow 0])$ (left) and $x^1 = QPBO(E[p \leftarrow 1])$ (right) where p, q, r are initially unlabeled. Subfigure 6.7(b) shows the principle of QPBO improve. The labels for the red dots are fixed, and the new output on the right side might be partially different.

2.5.1 Multi-Label Graph Cut is an extension to the graph representation of the MRF via s-t-network [11]. For each additional label, a new node is inserted. Formally each random variable nodes X_p is substituted by a set of nodes for each label $\{Z_p^0, Z_p^1, \ldots, Z_p^i\}$, where the edge $w_p^{i,i-1} = \infty$ ensures that only one of the vertical edges $w_p^{i-1,i} = \theta_{p,i}$ is cut. Additional horizontal edges are created $w_{pq}^{i,j} = \theta_{pq,(i+1)j} + \theta_{pq,i(j+1)} - \theta_{pq,ij} - \theta_{pq,(i+1)(j+1)}$, and again submodularity needs to be ensured to have non-negative edge weights. This process is also presented in Figure 6.8.

2.5.2 Hierarchical Graph Cut is another extension of the graph representation [12]. For each hierarchy nodes are created, and edges of different kinds are created (Figure 6.8(d)). Edge weights are chosen as follow:

Unary cost (blue):	$E_p^t(1,0) = \theta_{p,t}$ and $E_p^t(0,1) = +\infty$ to prevent switches from label 1 to 0.
Boundary Terms (green):	$E_{pq}^{BND}(0,1) = E_{pq}^{BND}(1,0) = \theta_{pq,01}$
Exclusion (red):	$E_p^{EXC}(1,1) = +\infty$ because only one label per site might be labeled.
Completeness (magenta):	$E_p^{CPL}(0,0) = +\infty$ because one label per site must be labeled.

The unary costs have the same responsibility as in the general GC approach. The boundary terms correspond to the pairwise terms. Exclusion and completeness terms are added for this extension to ensure correct labeling. Also $E_p^t(0,1) = +\infty$ is needed to be introduced for the hierarchy. The resulting graph can be solved by the known algorithms. The extension to a multi-label-hierarchical GC however is more sophisticated. Additional considerations for maintaining submodularity are needed.

2.6 Cluster Analysis

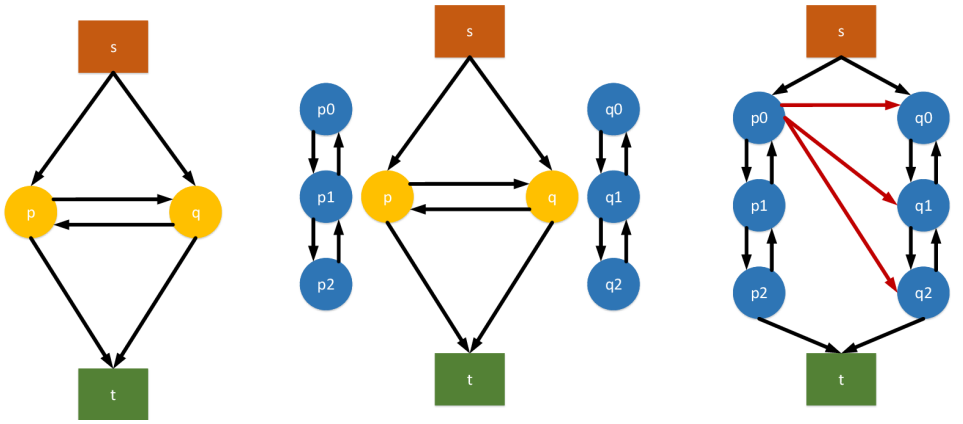
Cluster analysis is an interesting topic of its own research. However, for the graph cut it can help to reduce the amount of objects such that less variables need to be labeled by the graph cut.

Definition 2.12 (Cluster Analysis [13]) *The process by which observed data are organized, or clustered, into meaningful groups, on the basis of common properties.*

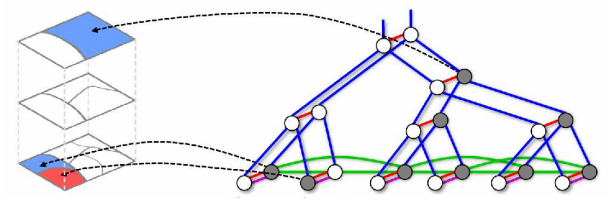
In the clustering process, many objects are grouped into meaningful meta-objects sharing, for instance, the shape of the objects. Then a labeling for the meta-object representing the small objects can be found.

In the process of clustering, it is important to note two facts:

1. if the clustering is performed such that the meta-objects do not match the interesting features, the clustering will yield misleading results.



(a) Original s-t-network. (b) Extension of s-t-network to chain. (c) Additional horizontal edges



(d) Hierarchical Graph Construction (image from [12])

Abb. 6.8: Creation of multi-label graph ((a)-(c)) and hierarchical graph construction ((d)).

2. if the difference in meta-objects is not assessed in accordance to the interesting features, the segmentation result will suffer.

Finally, the region-based segmentation relies on proper results from the clustering. As presented here, it is not possible to add more details after the clustering is finished.

In the two base papers two clustering techniques are used: Mean-Shift Clustering (MSC) and Statistical Region Merging (SRM). In the remainder of this section both ideas will be presented.

2.6.1 Mean-Shift Clustering Mean-Shift Clustering (MSC) [14] is an iterative process which pushes each point into the direction of the maximum increase of intensity. Thus, by applying MSC, in the specific case of pixel intensities, pixels are pushed towards levels of the same intensity. These level then form regions for the GC. Therefore, the gradient of the multivariate kernel density estimator $\hat{\nabla} f_{h,K}(x)$ is used to calculate the mean shift:

$$\mathbf{m}_{h,G} = \frac{1}{2} h^2 c \frac{\hat{\nabla} f_{h,K}(x)}{\hat{f}_{h,G}(x)} = \frac{\sum_{i=1}^{n} x_i g\left(\|x - x_i\|^2\right)}{\sum_{i=1}^{n} g\left(\|x - x_i\|^2\right)} - x \tag{6.5}$$

In the mean shift procedure, each iteration consists of the computation of the mean shift $\mathbf{m}_{h,G}$, and the translation of each point (pixel): $y_{j+1} = \mathbf{m}_{h,G}(y_j) - y(j)$. This method converges, when the gradient of the density function becomes zero, however in most cases it is sufficient to abort the process once the disturbance becomes less than a threshold.

More informally, MSC tries to push each point towards a certain level. Thus, it is a nonparametric clustering technique which does not require any prior information. Also the shape of the clusters is not influenced by this method. Furthermore, using multivariate kernel densities, it is also easy to extend as done in [2].

2.6.2 Statistical Region Merging Statistical Region Merging (SRM) [15] is a clustering technique also used directly for image segmentation. A region R is a set of pixels and the cardinality $|R|$ determines

how many pixels are in a region. Starting with a sorted set of connected regions (w. r. t. some distance function f), two regions R and R' are merged if the qualification criteria $|\overline{R'} - \overline{R}| \leq \sqrt{b^2(R) + b^2(R')}$ with $b(R) = g \cdot \sqrt{\frac{\ln \frac{\mathcal{R}_{|R|}}{\delta}}{2Q|R|}}$ is fulfilled. Therefore, $\mathcal{R}_{|R|}$ is the set of regions with $|R|$ pixels. Typically Q is chosen as $Q \in [256, 1]$ and $\delta = \frac{1}{|I|^2}$.

The Q parameter mainly influences the merging process. An example is given in Figure 6.9. Choosing lower values for Q, the regions are becoming more coarse. Using a union-find structure, the segmentation does not need to be recalculated for each Q level. For the step from q to $\frac{q}{2}$, simply the qualification criteria needs to be applied to the regions from the q result. A MATLAB implementation can be found in [16].

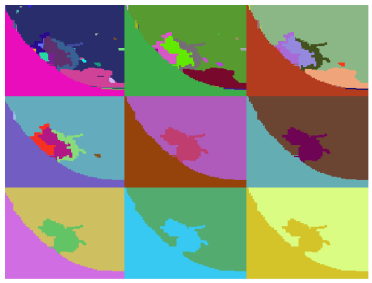


Abb. 6.9: SRM segmentation result for the test image from [1]. The upper left segmentation is received from $Q = 256$ and spread from left up to right down Q is bisected.

3 Base Paper Implementations

Figure 6.11 summarizes the ideas of both papers. Both papers start with an input image. For the pipeline presented in [1] first SRM is applied. The so created hierarchical regions are transformed into a hierarchical tree and then given to the graph cut which will be used for segmenting the image. In [2] first Joint-Spatial Intensity Shape (JSIS) features are extracted which are then the feature space used by MSC to create regions. The formed regions are transformed into a graph structure on which a graph cut is applied, which results in a segmented image.

3.1 Automatic Foreground/Background Segmentation

The method for the *Automatic Graph Cut Segmentation of Lesions in CT Using Mean Shift Superpixels* [2] consists of two steps before applying the graph cut.

Unfortunately, the paper is not clear on whether it operates fully on 3D images or not, as the used terminology sometimes switches from voxel to pixel and vice-versa. From the presented examples it can be assumed that the presented method was performed on each slice of a 3D image.

At first Joint-Spatial Intensity Shape (JSIS) features are extracted. For each voxel the 5D JSIS feature is calculated which consist of spatial information (X, Y, Z position), intensity and shape index. The extracted shape index is the novelty.

Shape Index The shape index is calculated as presented in [17] and is derived from the principal curvatures κ_1, κ_2 at each pixel: $SI(v) \equiv \frac{1}{2} - \frac{1}{\pi} \arctan \frac{\kappa_1(v)+\kappa_2(v)}{\kappa_1(v)-\kappa_2(v)}$. A shape index value of 0 describes a cup, while a value of 1 would describe rather a cap. Intermediate values resemble different other forms, for instance ridges. However the shape index calculation underlies the restriction that objects should be separated. This will be discussed in Section 4.1.

The shape index is used to automatically determine foreground and background seeds using thresholds. Selecting the foreground seeds works as follows:

1. all voxels in a region must have shape index $\geq T_l$;
2. at least one voxel in a region must have shape index $\geq T_h$.

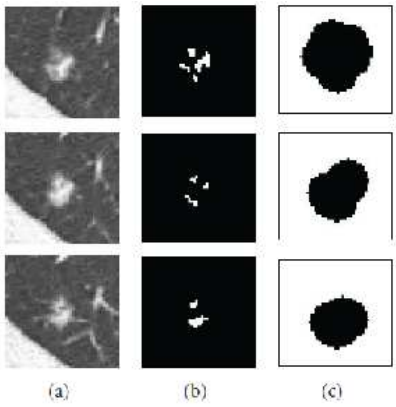


Abb. 6.10: Automatic foreground/background seed generation [2]. (a) shows the original image, in (b) foreground seeds are marked white. In (c) the enlarged foreground section is black, the selected background is white.

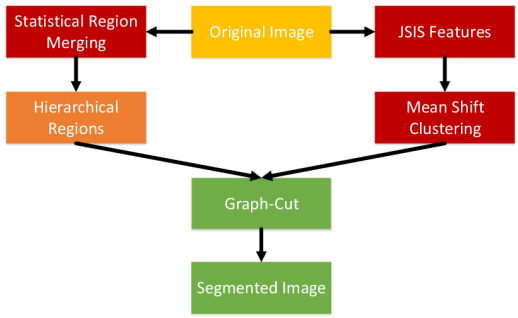


Abb. 6.11: Image segmentation pipelines for the two base papers [1] (left) and [2] (right).

For the background seeds, first the foreground area is extended to ensure to not include any foreground. Then background is considered to be not extended foreground (Figure 6.10). While this approach is the base of [2], in Section 4.1 it is explored as an extension to [1].

The shape index-based foreground/background determination seems to work under many circumstances. In general, the paper states, that the addition of the shape index kernel in the mean shift clustering is beneficial. This is underlined by several exemplary pictures shown for several different conditions. Additionally, useful comparisons to other methods, such as pixel-based GCs or region-based GCs without shape index are made. Finally, the overall quality of the segmentation results are graded by a comparison of Dice's coefficient for about 100 different segmentations. As reference the four dimensional MSC approach without the shape index is chosen. The proposed method increases the mean Dice coefficient from 0.71 to 0.79 compared to the reference implementation while decreasing the standard deviation from 0.1 to 0.06. Therefore, the method really improves segmentation results.

However, the results for Ground-Glass Opacity (GGO) nodules are, as expected by the authors, not as good as for the solid nodules. This can be explained by the low contrast and irregular shape forms of GGO nodules as well as the often attached vessels. Several ideas for improving the segmentation are proposed in the paper, such as grey level co-occurrence and a more robust foreground pixel selection process not only considering the shape index but also pixel intensities.

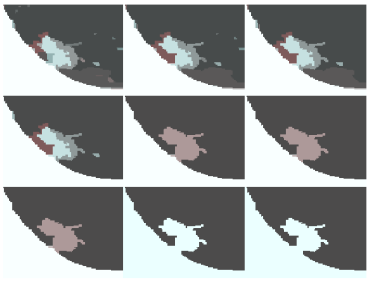
3.2 Hierarchical Graph-Cut

The paper *Region-based graph cut using hierarchical structure with application to ground-glass opacity pulmonary nodules segmentation* [1] also bases on the creation of regions which are then translated into a graph structure. On these regions a graph cut is performed for the segmentation. The test image is taken from the paper [1] with parts parts of the right border missing due to cropping from the literature. The following parts are segmented (color coding): lung wall (*blue*), background (*green*), GGO area and vessels (*yellow*) and the nodule (*red*).

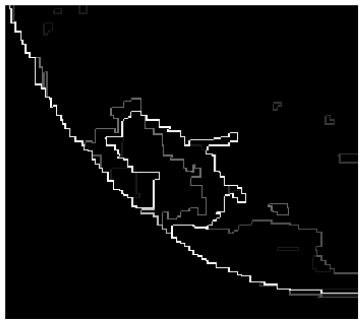
For the creation of regions, statistical region merging as presented in Section 2.6.2 is used. Regarding the creation of the hierarchical edge map, the paper says that the parameter Q range is set up through the values $1, \ldots, 256$ and that the hierarchical edge map is obtained by using union-find algorithm [1]. While it is true that the parameter Q should be set up in this range, one has to start with high values of Q. By this, the edge map can be constructed by printing the edges at each Q value in a different strength, starting at a low strength for high values of Q and maximal strength for $Q = 1$ (Figure 6.12(a)-(c)). If two regions are merged in the step from Q to $\frac{Q}{2}$, the outer edges of the merged regions remain the same. Furthermore, using the union-find structure, it can be kept track which regions are merged.

In the second step, the hierarchical graph cut is performed. Therefore, the edge-map is transformed into a hierarchical tree [1]. The paper does not describe this process in detail, it is assumed that the original implementation from [12] is used. Here first the finest regions are recreated, and then neighborhoods are

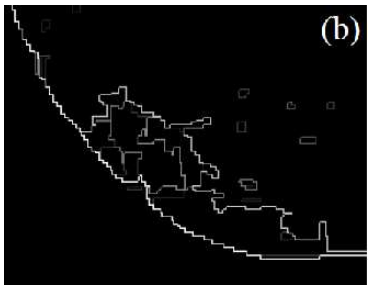
created, whenever the same edge-strength divides two adjacent regions. For the algorithm, the original image, the (inverted) edge map and a seed image is needed. The seed image is mentioned in [1], but it is not addressed how to generate it. Therefore, it must be assumed, that it needs to be generated manually. All needed input images are shown in Figure 6.12(d)-(e).



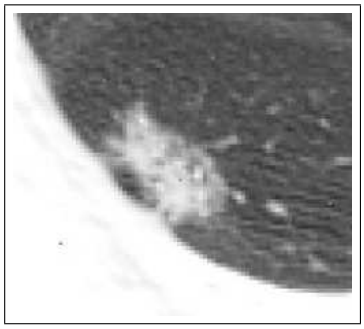
(a) Segmentation result from SRM for $Q = 256$ (upper left) to $Q = 1$ (lower right).



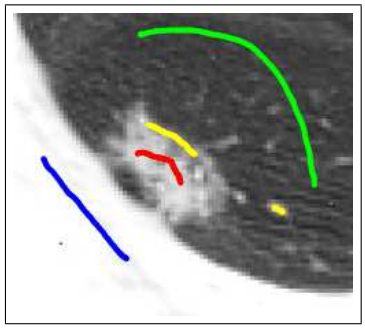
(b) Inverted hierarchical edge map for the SRM segmentation.



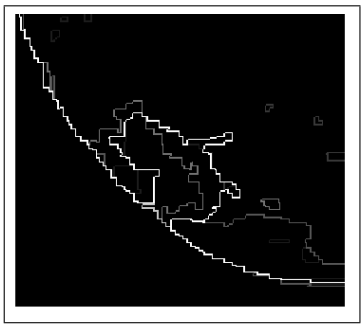
(c) Hierarchical edge map for the same input image [1].



(d) Input image for GC.



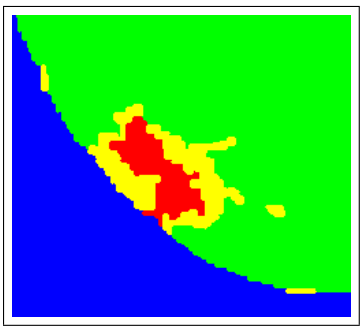
(e) Manually created seeds for segmentation.



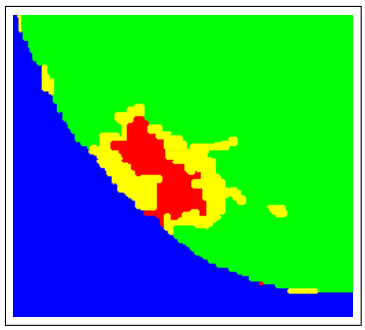
(f) Inverted hierarchical edge map from SRM.

Abb. 6.12: (a)-(c): SRM results for the test image from [1]; (d)-(e): input images for the hierarchical GC.

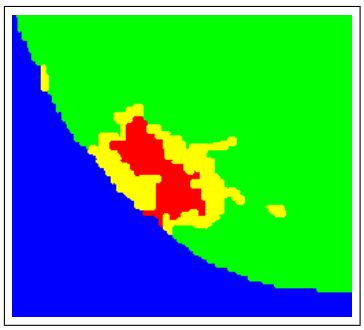
The segmentation is calculated on the hierarchical graph structure using the QPBO algorithm which was introduced in Section 2.4. Finally, the algorithm returns three segmentations, where one operates on a CRF, another one on the hierarchical graph without pairwise terms and finally a segmentation using the hierarchical graph with pairwise terms. The segmentations for the input (Figure 6.12) are shown in Figure 6.13.



(a) Segmentation result using flat CRF segmentation.



(b) Segmentation result using hierarchical GC without pairwise terms.



(c) Segmentation result using hierarchical GC with pairwise terms.

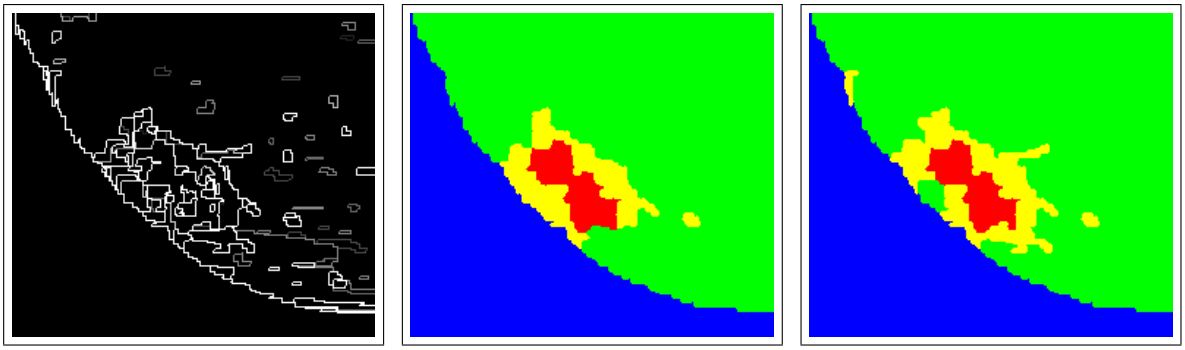
Abb. 6.13: Segmentation results of the test image from [1] using the original pylon model implementation.

However, some problems with this approach have been determined, some of them as general remarks regarding region-based segmentation, others related to the presented work flow. First, only existing regions

can be labeled and no regions can be added adaptively after the clustering process. The adaptive adding of regions might become handy, where a region is close to all of the possible categories.
Furthermore, it can be seen, that in this specific case, the SRM creates too coarse regions. This can be observed at the green separated background between the nodule and the lung wall.
Also neither unary nor pairwise energies are given by Tsou et al. [1]. Their choice of energy can only be guessed. This is particularly regretful since those energies are *the* important input parameters for the GC method, besides the seeds. However, it has been observed that the original MATLAB implementation results of the GC [12] comes close to the segmentations presented by Tsou et al..
The used MATLAB implementation applies a χ^2 probability distribution for the labels around manual seeds. This might be misleading where there are sharp and small borders. A possible solution would be to favor the pairwise energy over unary energies or use to no transitioning.

Finally, the original MATLAB implementations [12, 16] have been adapted such that the SRM creates more regions. Further, the pairwise energy has been adapted to use the maximum of mean edge strength and median. Also the exponential pairwise term is scaled differently, such that large edge strength differences are more important. The unary term influences the segmentation less. Using this adapted version of the image segmentation pipeline, the results are presented in Figure 6.14 and come close to the presented segmentation.

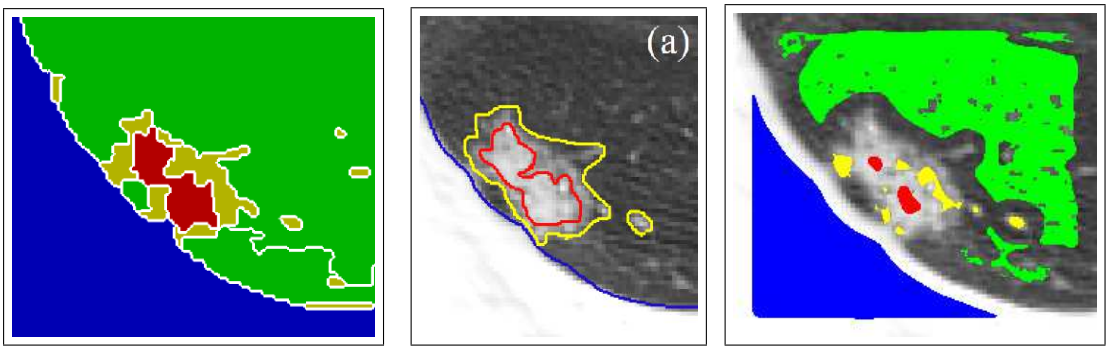
Besides missing details of their implementation, the evaluation process chosen by the authors is not optimal. The ground truth was determined by four doctoral students. A consultation of a proven expert on the area would have been useful. While 77 CT studies have been evaluated, it would have been interesting to have more than one image discussed. After all, the proposed method seems to improve segmentation results over the compared Grow Cut algorithm, especially as it is said to be robust to problems of heterogeneous textures as GGO areas. However no measure suggesting the significance of the improvement is given.



(a) Regions created from SRM clustering using the adapted version.

(b) Flat CRF segmentation result for the adapted hierarchical version.

(c) Hierarchical segmentation result for the adapted version with pairwise terms.



(d) Optimal computer generated solution presented in [1].

(e) Optimal manual solution presented in [1].

(f) Automatically generated seeds using the method presented in Section 4.1.

Abb. 6.14: Results for the adapted hierarchical segmentation using the adapted SRM method compared to the automatic and manual segmentation result presented in [1]. In (c) comparable results to the results from the paper ((d)) with slight differences in the lower part of the GGO nodule have been achieved. The adapted version performs better regarding separated GGO areas.

4 Extension: Automatic Seed Generation

4.1 Shape Index Results

Ye et al. ([2]) introduce an automatic image segmentation pipeline, which bases its seed generation on the shape index [17]. However it is stated, that the approach has problems with GGO nodules. [1] however presents an semi-automatic image segmentation for GGO nodules, where the user has to apply seeds. A plausible extension is to combine the automatic seed segmentation, with the segmentation working for GGO nodules to receive a fully automated pipeline.

It soon became obvious that when calculating the shape index for the example GGO nodule from [1], the results are disappointing (Figure 6.15). It can be seen, that for the background lung wall an index of 0 (blue) is reached, however, for the interesting area around the nodule, no relevant features can be extracted. As a shape index around 1.0 (red) is expected around the nodule, only values larger than 0.8 are regarded. Still no interesting feature can be recognized. Even after smoothing, high levels of noise remain throughout the image making the shape index a not suitable method for automatically calculating seeds.

4.2 Smoothness Index

For position prediction in virtual environments, jitter or noise in the output signal is not wanted while often present. Since discovering smooth areas is a similar problem to jitter detection, a simple method for determining jitter can be used to measure non-jitter, smoothness [18]. It is assumed that jitter-free areas of a position signal do not differ in velocity.

Smooth areas don't differ in intensity, and therefore only low changes in velocity (intensity change) can be recorded. For the reduction of noise, this operation is performed on the smoothed input image. Then the smoothness s of a pixel p can be determined as:

$$s(p) = \sum_{p' \in \mathcal{N}_k} \nabla(p') / \arg\max_p s(p) \tag{6.6}$$

Using a threshold method for selecting different areas as given by $TS_l \leq s(p) \leq TS_h \wedge TI_l \leq I \leq TI_h$, the results for empirically found thresholds are given in Figure 6.17.

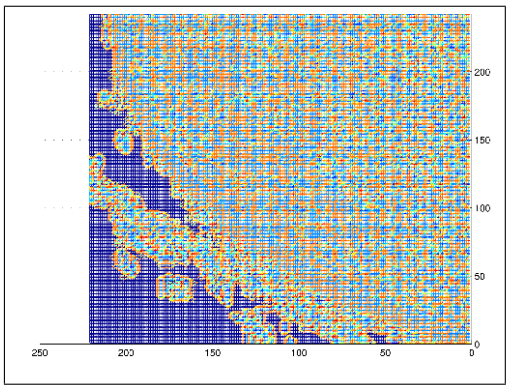
For the empirical choice of thresholds it can be argued that these are hand tailored to the specific case. While this surely is true to a certain extent, a randomly chosen region of interest from [2] is used to verify the algorithm. In general more test cases must be chosen, yet only one test case is considered as this is a seminar thesis. The final seeds are shown in Figure 6.16 with segmentation results in Figure 6.16. It is interesting to note, that the hierarchical pairwise segmentation seems to have more problems along the border to the background as the flat CRF segmentation. This surely depends on the selected energy function, it must also be questioned, whether the more complex hierarchical approach is always useful.

5 Conclusion

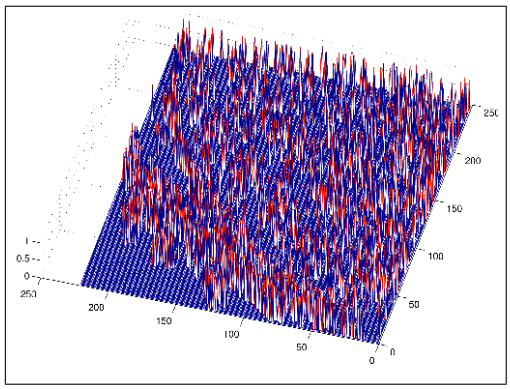
It has been presented, that the pixel-level Graph-Cut (GC), in contrast to other methods, has problems with erroneous boundaries, while other algorithms suffer from over-segmentation. Region-based GCs try to reduce these problems [1, 2]. Furthermore, it has been shown, that image segmentation is a special form of the labeling problem which can be represented using random field theories and finally is solved by the minimum graph cut. Efficient algorithms for solving the minimum graph cut base on finding the maximum flow. Whenever a problem is not restricted to sub-modular energy functions, only approximations can be calculated as the labeling problem is NP-hard in general. For solving this, the QPBO algorithm has been presented.

For the automatic foreground/background determination, [2] proposes a method which is based on the shape index of the pixels. As noted in the paper, the seed generation approach has problems on GGO nodules. This constraint has also been verified in the course of this thesis. However possible improvements are already given in the paper: gray level co-occurrence and a more robust shape index are, while not in detail, proposed.

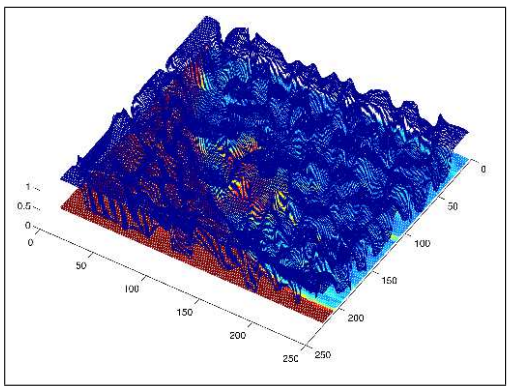
In the hierarchical segmentation paper [1], a rough description of a pipeline using SRM for region creation, and a hierarchical graph cut, is given. Using existing implementations, the methods used in this



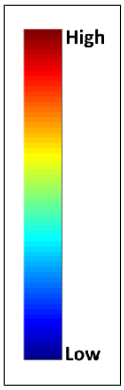
(a) Plot of the shape index for the test image from [1]. No clear features can be seen.



(b) Plot of the shape index (≥ 0.8) for the test image from [1]. No clear features can be seen.

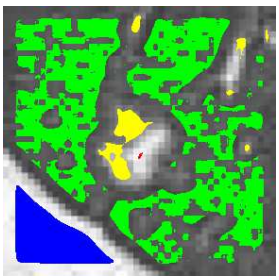


(c) Plot of the shape index for the highly smoothed test image from [1]. No clear features can be seen.

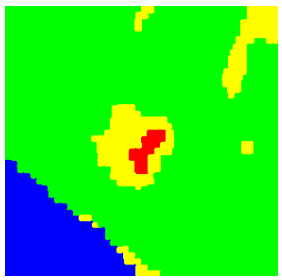


(d) Colormap (jet) for shape index and smoothness index plots.

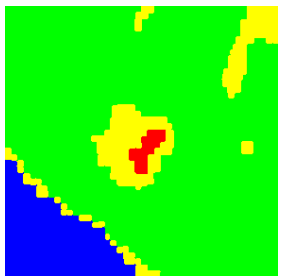
Abb. 6.15: Shape index calculation for the test image from [1]. On the raw image no clear feature around the GGO nodule can be seen. Even after smoothing, which emphasizes the cap structure of the nodules, no feature can be detected.



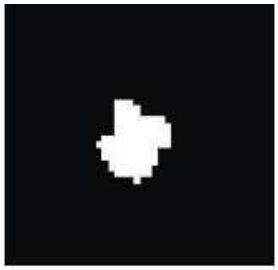
(a) Automatic seed generation result for the test image from [2].



(b) Segmentation result using CRF only.

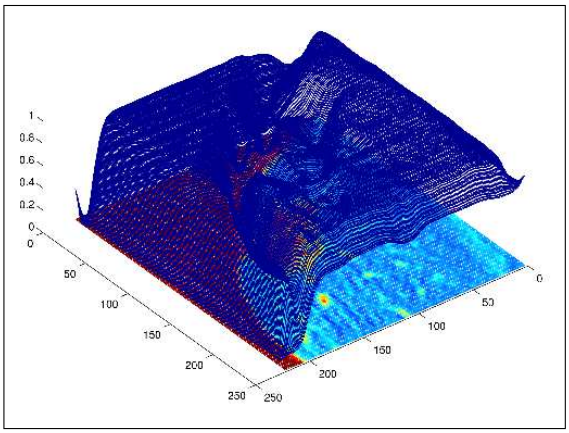


(c) Segmentation result using hierarchical GC with pairwise energy.

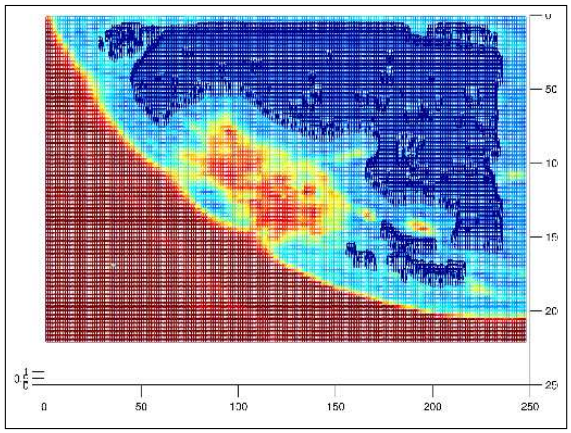


(d) Segmentation result from [2].

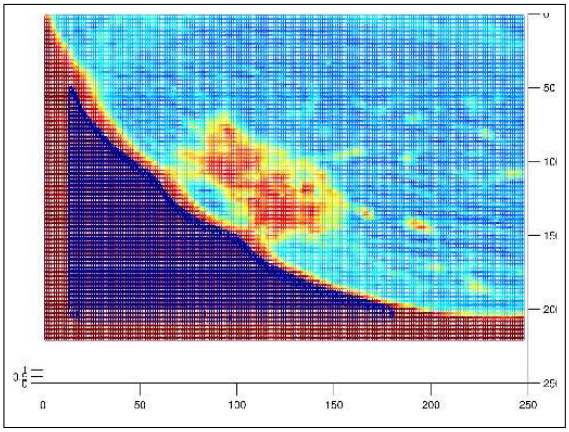
Abb. 6.16: Seeds automatically determined by the presented automatic seed generation using the smoothness index. In *blue* the tissue walls are marked, *green* for background, *yellow* for GGO fragments and *red* the nodules. Segmentation results for the test image from [2] using the hierarchical approach compared to the result from the original paper (d).



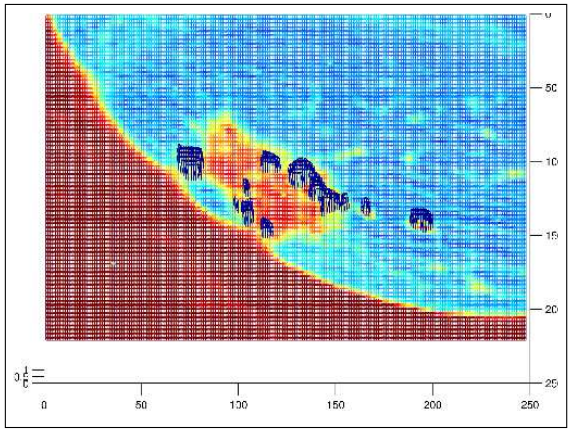
(a) Smoothness index for the whole test image.



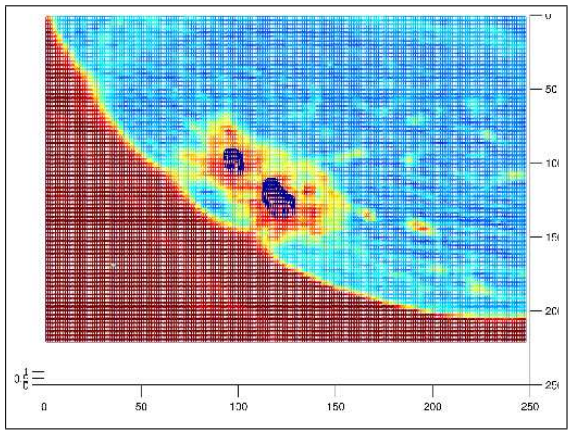
(b) Background selected with $s(p) > 0.96$ and $I < 85$.



(c) Tissue wall selected with $s(p) > 0.85$ and $sI < 235$.



(d) GGO area selected with $0.8 < s(p) < 0.95$ and $125 < sI < 175$.



(e) Nodule area selected with $s(p) > 0.9$ and $200 < sI < 230$.

Abb. 6.17: Selected regions from smoothness index where $s(p)$ is calculated on the smoothed image sI of the test image I from [1].

approach could be clarified, while *important* details such as the used energy function remain hidden. Also regarding the creation of the hierarchical tree, no substantial details are given. Problems of the described pipeline could be detected. For instance, a bad initial clustering will lead to a bad segmentation. Also poor manual seeds influence the outcome of the segmentation dramatically, especially along edges, due to the possibly applied distribution along the seeds.

Regarding the limitation with manual seeds, it has been tried to merge the automatic seed generation from [2] with the working GGO nodule segmentation from [1]. However, it showed that the current shape

index implementation delivers highly noisy output for GGO nodules making it infeasible to use. Instead, a new smoothness index is proposed. This smoothness index makes use of the observation that the change in intensity in smooth areas is minimal. The seeds are selected on the basis of thresholding. Finally, a more robust seed generation algorithm is given.

Possible future work regarding the seed generation could be an adaptive retrieval of the thresholds for intensity and smoothness. For this, the pixel intensity histogram could be analyzed using *k-means* clustering to find the correct intervals for each image label. Similarly, the pixel smoothness index histogram could be used for finding corresponding values.

Literaturverzeichnis

[1] Tsou CH, Lor KL, Chang YC, Chen CM. Region-based graph cut using hierarchical structure with application to ground-glass opacity pulmonary nodules segmentation. Proc SPIE. 2013;8669:866906–6.

[2] Ye X, Beddoe G, Slabaugh G. Automatic graph cut segmentation of lesions in CT using mean shift superpixels. Int J Biomed Imaging. 2010 Jan;2010:19:1–19:14.

[3] Li SZ. Markov random field modeling in image analysis. Advances in pattern recognition. New York, London: Springer; 2009.

[4] Griffeath D. Introduction to random fields. In: Denumerable markov chains. vol. 40 of Grad. Texts in Math. New York: Springer; 1976. p. 425–58.

[5] Lafferty JD, McCallum A, Pereira FCN. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth ICML. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2001. p. 282–9.

[6] Picard JC, Ratliff HD. Minimum cuts and related problems. Networks. 1975;5(4):357–70.

[7] Greig DM, Porteous BT, Seheult AH. Exact maximum a posteriori estimation for binary images. J R Stat Soc, Series B (Methodological). 1989;51(2):271–9.

[8] Kolmogorov V, Zabin R. What energy functions can be minimized via graph cuts? IEEE Transactions on PAMI. 2004;26(2):147–59.

[9] Kolmogorov V, Rother C. Minimizing nonsubmodular functions with graph cuts-a review. IEEE Transactions on PAMI. 2007;29(7):1274–9.

[10] Rother C, Kolmogorov V, Lempitsky V, Szummer M. Optimizing binary MRFs via extended roof duality. In: Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on; 2007. p. 1–8.

[11] Carr P, Hartley R. Solving multilabel graph cut problems with multilabel swap. In: Proceedings of the 2009 Digital Image Computing: Techniques and Applications. DICTA '09. Washington, DC, USA: IEEE Computer Society; 2009. p. 532–9.

[12] Lempitsky VS, Vedaldi A, Zisserman A. Pylon model for semantic segmentation. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira FCN, Weinberger KQ, editors. NIPS; 2011. p. 1485–1493.

[13] Zvelebil M, Baum JO. Understanding bioinformatics. New York: Garland Science Taylor & Francis Group; 2008.

[14] Comaniciu D, Meer P. Mean shift: a robust approach toward feature space analysis. IEEE Transactions on PAMI. 2002;24(5):603–19.

[15] Nock R, Nielsen F. Statistical region merging. IEEE Transactions on PAMI. 2004;26:1452–8.

[16] Boltz S. Statistical region merging matlab implementation; 2014. Available from: http://www.mathworks.com/matlabcentral/fileexchange/25619-image-segmentation-using-statistical-region-merging. Accessed 12 Dec 2013.

[17] Yoshida H, Nappi J. Three-dimensional computer-aided diagnosis scheme for detection of colonic polyps. IEEE Transactions on MI. 2001;20(12):1261–74.

[18] Joppich M, Rausch D, Kuhlen T. Adaptive human motion prediction using multiple model approaches. In: Virtuelle und Erweiterte Realität, 10. Workshop der GI-Fachgruppe VR/AR. Shaker Verlag; 2013. p. 169–80.

Automatic Segmentation of the Heart

Bushra Fatima

1 Introduction

According to World Health Organization report an estimate of 17.8 million people died of cardiovascular diseases (CVDs) in 2008. About 80% of CVDs related deaths happen in low and middle income countries. By 2030 more than 23 million deaths will occur annually because of heart diseases. It is thus imperative to reduce this death toll by adopting a healthy lifestyle; however advancement in medical technology is also crucial for early diagnosis and treatment to improve chances of survival for the patients. Today's imaging devices such as CT and MR scanners provide a vast amount of high-quality images with high resolution in space and time. A wealth of patient-specific information can be obtained from these images which are both diagnostically and therapeutically relevant. [**?**]

1.1 Brief Description: Anatomy of Heart

Heart is a muscular organ in human body that is responsible for circulating blood in the body. It collects impure or deoxygenated blood through the veins and sends it to the lungs for oxygenation which is then transferred back to the heart and is pumped throughout the body via various arteries. The heart is located in the thoracic cavity which is in the middle of the lungs and behind the sternum. On its upper end and the base the heart is attached to aorta pulmonary arteries and the veins and the vena cava. The heart wall is made of 3 layers: epicardium (outermost wall), myocardium (middle layer) and endocardium (internal heart wall). The heart contains 4 chambers: the right atrium, left atrium, right ventricle, and left ventricle. The heart functions by pumping blood both to the lungs and to the systems of the body. To prevent blood from flowing back into the heart, a system of one-way valves is present in the heart (See Figure 1). At any given time the chambers of the heart may found in one of two states:

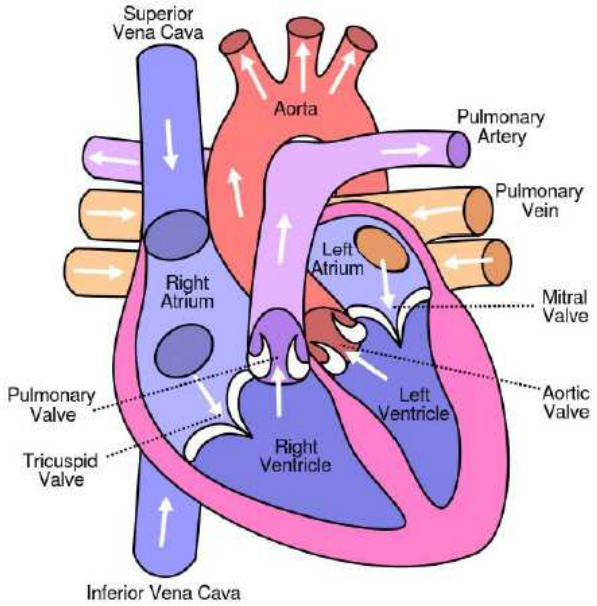


Abb. 7.1: Heart Anatomy

1. Systole: During systole, cardiac muscle tissue is contracting to push blood out of the chamber.

2. Diastole: During diastole, the cardiac muscle cells relax to allow the chamber to fill with blood The cardiac cycle includes all of the events that take place during one heartbeat. [2]

1.2 Medical Imaging Technology for Cardiac Pathology Diagnosis

For a patient with ischemia and infarction induced by coronary-artery stenosis, information about the heart function, myocardial perfusion and scar tissue can be derived from imaging studies. Both computed tomography (CT) and magnetic resonance imaging (MRI) techniques can be used as an alternative to invasive angiography, a procedure that uses cardiac catheterization. Along with other tools such as nuclear imaging, Doppler ultrasound and PET-CT scans, these noninvasive techniques provide fast, accurate visualization and analysis of the heart and surrounding structures.

1.2.1 Computed Tomography (CT) CT scan uses contrast dye injected into the body along with a series of X-ray images to provide highly detailed images of the heart and surrounding structures. It can provide accurate cardiac visualization and analysis in less than 30 seconds. The images are taken in multiple slices between beats for accurate imaging. [3]

1.2.2 Magnetic Resonance Imaging (MRI) MRI is a diagnostic procedure that uses a combination of a large magnet, radiofrequencies, and a computer to produce detailed images of organs and structures within the body. MRI does not use ionizing radiation as CT does. [3]

1.2.3 Limitations Associated with Raw Images Some of the details of heart anomalies are not clear from CAT scan and MRI and other imaging tools and so a need to clearly visualize is imperative. The images contain a huge amount of information; this information is not readily available and must be extracted from the images. Numerous methods have been developed for this purpose. The purpose of the paper is not to explore which imaging technology is better but rather to study methods of image segmentation which is part of the post processing of image information for clear diagnosis. Time-consuming manual approaches are a serious hurdle in routine clinical practice and the need for an efficient clinical workflow drives the development of automated methods. A suitable method of presenting the information is also needed that allows, for instance, integration of complementary information from different scans as well as the intuitive display of the information during an intervention. Numerous techniques have been proposed in the past which were mostly designed for segmentation of the left/right ventricle. These methods employed image processing based techniques such as threshold and clustering or manipulation of simple geometric models according to the obtained patient specific images or flexible deformable models that allowed local manipulations with smoothness constraint. [3]

In the following section we will first discuss the segmentation of medical images in general.

2 Automatic Segmentation Technique for Medical Images

Accurate medical image segmentation is crucial in fine tuning useful information during diagnosis and treatment planning. Computed topography (CT) and Magnetic resonance (MR) imaging are the most widely used radiographic techniques in diagnosis, clinical studies and treatment planning. This section gives and overview of automated method of segmentation in context of CT and MR. Also we will explore problems encountered in image segmentation and merits and demerits of state of the art methods for image segmentation. Due to the increased use of CT and MR imaging for diagnosis it has become imperative to use computerized (semi-automated or completely automated) techniques for clinical diagnosis and treatment planning. Hence, there is a need for reliable algorithms for delineation of anatomical structures and regions of interest (ROI). Following are the goals for automation of diagnosis:

1. Handling of large number of cases and to increase accuracy through reducing human error.
2. Speed up diagnosis process and hence speed up steps towards patient recovery.

Each technique that is available for segmentation of medical images is specific to particular application, image type and innate characteristics of organ under consideration. There is no universal algorithm for segmentation of every medical image. However the problem with both CT and MR medical images involve:

partial volume effect, organ specific image modalities such as motion artifacts and ring artifacts etc., also noise due to sensors and related electronic systems. Images are presented in 2D as well as 3D. In 2D each element in the image is called a pixel whereas in 3D domain it is called voxel. Sometimes the 3D image is achieved using series of 2D image slices. The segmentation process involves dividing the image into regions with similar properties such as gray level, color, texture, brightness and contrast. In case of medical images the aim is to study the anatomical structure, identification of region of interest e.g. location of any anomalies such as tumors, measure tissue volume to detect abnormality, and treatment planning. Automatic segmentation of medical images is a complex process as the organs do not have linear features and the output of the segmentation may be affected due to CT/ MR limitations, presence of noise due to electronic signals or movement, intensity inhomogeneity and non-distinguishable gray level of different/ adjacent tissues. Researches have proposed image segmentation which may be categorized as:

1. Methods based on gray level features
2. Methods based on texture features

2.1 Methods based on gray level features

This encompasses the following segmentation methods:

2.1.1 Amplitude segmentation based on histogram features: This includes segmentation of an image based on thresholding of histogram features and gray level thresholding is perhaps the simplest example of this technique. This is particularly suitable for an image with region or object of uniform brightness placed against a back ground of different gray level, a threshold can be applied to segment the object and background to segment the ROI. However the limitation of this method is selection of proper values of threshold is quite difficult and performance may be affected in presence of noise organ specific modalities.

2.1.2 Edge based segmentation: Edge based segmentation is the most common method based on detection of edges i.e. boundaries which separate distinct regions. Edge detection method is based on marking of discontinuities in gray level, color etc., and often these edges represent boundaries between objects. This method divides an image on the basis of boundaries. One of the examples of edge based segmentation algorithms is "Hough Transformationänd "Border detection method".

2.1.3 Region based segmentation: Region based methods are based on the principle of homogeneity - pixels with similar properties are clustered together to form a homogenous region.

2.2 Methods based on texture features

Textural features of image are important from image segmentation and classification point of view. Different researchers have used these features to achieve image segmentation, classification, and both segmentation as well as classification. The aim of texture based segmentation method is to subdivide the image into region having different texture properties, while in classification the aim is to classify the regions which have already been segmented by one or other method. [4]

2.3 Approaches in Focus

The focus of this seminar paper is to explore in detail two approaches of automated segmentation which are state of art in medical image segmentation, which are as following:

1. Model based segmentation and
2. Atlas based segmentation.

In section 3 and 4 we will explore these methods of segmentation with reference to heart. Subsequently we will discuss, compare and contrast in detail two approaches of automatic segmentation of heart, first proposed by O. Ecabert et al. in their paper SSegmentation of the heart and great vessels in CT images using a model based adaptation frameworkänd second X. Zhuang et al. proposed techniques in Ä registration based propagation framework for automatic whole heart segmentation of cardiac MRI".

3 Model based Segmentation

The basic approach is that the structure of organs has a repetitive form of geometry and can be modeled probabilistically for variation of shape and geometry. This can be used as constraint while segmenting the image and involves:

1. Registration of the training data.
2. Probabilistic representation of variation of registered data.
3. Statistical influence between model and image.

Model based methods of segmentation involve active shape and appearance model, deformable models and level-set based models. Some of the disadvantages of model based segmentations are:

1. They require manual interaction to place an initial model and choose appropriate parameters.
2. Standard deformable models can also exhibit poor convergence to concave boundaries.

Section 3.1 explores method proposed by Olivier Ecabert et al. which performs segmentation of the heart through model adaption using a framework that not just performs segmentation of the heart chambers but also performs segmentation of the tubular structures attached. [4]

3.1 Model Based Adaptation Framework

The method as proposed by Ecabert et al. first detects the heart using adapted Generalized Hough transformation (GHT) and performs segmentation of the heart by adapting a generic heart model represented by a surface mesh. Iteratively adjustments are made by scaling, and changing global orientation, also parameters of shape variability are optimized to perform deformable adaptation while keeping the heart shape consistent. A more definite heart model is achieved using boundary detection functions.

In this technique firstly a detailed heart and vessel model is achieved using gathered image data. The parametric description of the model's shape variability is important for further adaptation using multiple linear transformations which are used to bend and bow vessel diameters. Finally information for boundary detection is associated to each triangle in the shape mesh. For building the model 35 data sets (28 for whole heart and 7 specifics for construction of great vessels images) from 20 patients were used. These data sets were mapped onto variety of phases from beginning of the beat to the start of the next also known as the cardiac cycle. This will enable of get typical image data set that covers all possible shape variations that can be seen in a routine clinical exam of a patient.

3.1.1 Heart and Vessel Model Generation After collecting the dataset of cardiac CTA a geometric mesh model of the heart and the major vessels is generated. Starting with 4 chamber model consisting of 7286 vertices and 14,771 small triangles the mesh can be constructed from scratch (figure 2).

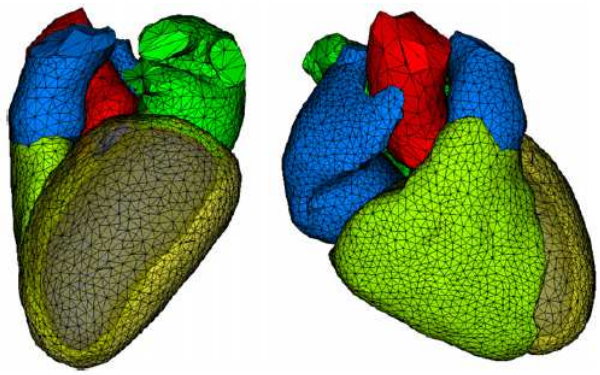


Abb. 7.2: Four chamber heart model.

The model is then further extended by generating and attaching meshes of great vessels for which the 7 CTA images were taken in which aortic arch and major portion of superior and inferior vena cava.

The centerlines of inferior and superior vena cava, the coronary sinus and the four pulmonary veins were identified manually in these data sets and represented by equidistantly placed points. Radius information was also association with these identified centerlines using a search-ray based method. This method casts 20 search rays estimating the radius along the contour of the vessels, estimating mean radius. The threshold in this method is manually adjusted in a way that eventually vessel lumen (cross sectional radius of the vessel opening) is described. Next the mean geometry of the great vessels is generated by adapting the 4 chamber heart model according to the 7 CTA data set and the orientation of the un-deformed heart model is determined by point based registration to generate the adapted heart model. Average radii and centerlines are generated by taking average of the difference vectors between two successive centerline points. Finally, meshes of the great vessels are constructed and attached to the 4 chamber heart model.

The resulting heart model has 4 chamber mesh model with attaches great vessels. This mesh contains 8506 vertices for 17,324 small triangles. 8 great vessels are also attached composed of 99 rings which also contain information about the vessel lumen (Figure 3).

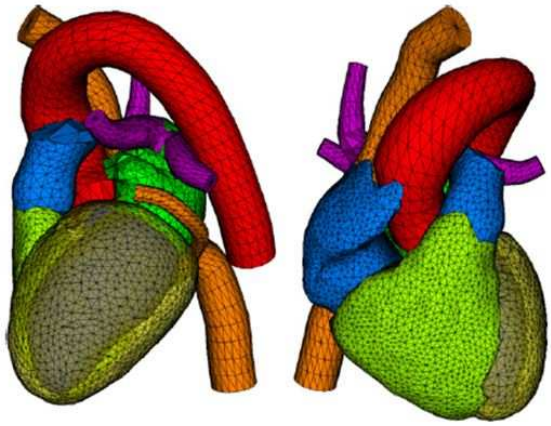


Abb. 7.3: Heart Model with great vessels in High Resolution

By removing 4819 (56.7%) of the vertices from the mesh model of the heart chambers, the resolution is reduced however the triangle size remains consistent (figure 4).

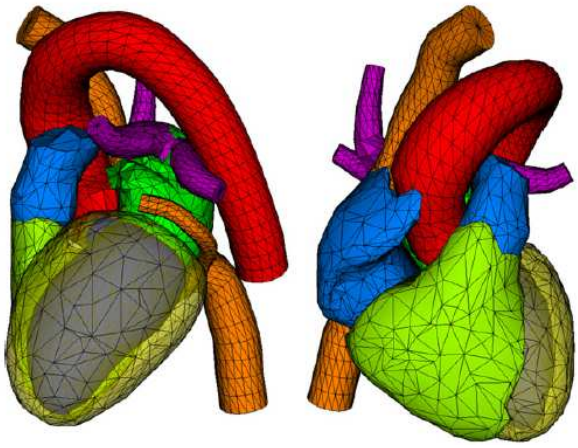


Abb. 7.4: Heart Model with great vessels in Low Resolution.

3.1.2 Associated Parametric Description of Mesh Model After the mesh model is achieved the parametric description of shape variability is generated and also the function for transformation. The method subdivides the reference model represented by mesh vertices $(m_1, ..., m_v)$ in K individual parts described by weights $w_(i, k)$, based on condition that all parts are independent, defined as:

$$w_(i,k) = \begin{cases} 1 \text{ if vertex i belongs to part k} \\ 0 \text{ otherwise} \end{cases} \tag{7.1}$$

The entire model can be transformed using a linear transformation based on a certain parameters $q = q_1, ..., q_k$, given as:

$$T_{multilinear}(q)[m_i] = \sum_{k=1}^{K} w_{i,k}.T_k(q_k)[m_i] \tag{7.2}$$

After transformation of the model certain discontinuities may result in the mesh model which can be smoothed out by introducing transition regions between individually transformed parts interpolated together. This is achieved by assigning weights $0 < w_(i,k) < 1$ in the transition regions using the following equation:

$$\sum_{k=1}^{K} w_{i,k} = 1 \forall i \tag{7.3}$$

The model shape variability is done through multilinear transformations which provide empirical parameterization and not based on statistical information. In order to make accurate adaptation of the model which is close to real approximation especially for the 4 chambers heart model eigenmodes resulting from principal component analysis are used. An affine transformation is used to further subdivide the heart chamber model into atriums and epicardium. However, for the attached vessels the 3 and 6 successive structured rings into short tubular segments, each of which is associated with individual similarity transformation (figure 5). Between two tubular segments there is a transition region for preservation of smooth mesh geometry. The associated multilinear transformations enable not just global bending of tubular structures but also local diameter variation.

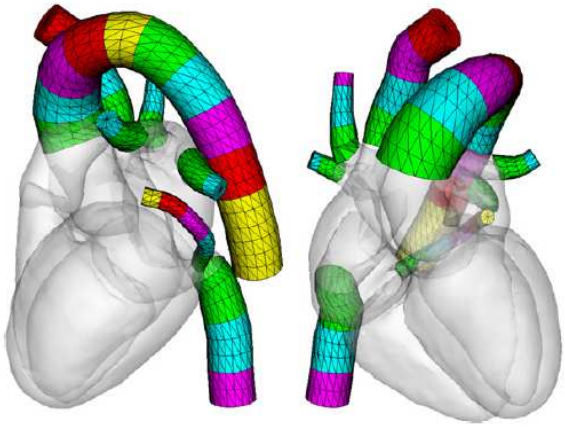


Abb. 7.5: Color coded visualization of combination of successive structured rings into short tubular segments

3.1.3 Boundary Detection Function During model adaptation, the boundaries of the target organ must be detected in the image. Boundary detection is supported by appearance information that is learned from a set of annotated reference images and corresponding meshes. During a training phase, the images and meshes are used in a first step to create a large number $(500 - 10000)$ of boundary detection function candidates which use gradient information and gray-value information on one or both sides of the mesh. For gray-value calibration, information about global gray-value statistics can also be used. Distinct gray value transitions are identified along the profiles parallel to normal of the triangle and passing through its center c. The detected boundary point is given by;

$$\widehat{x} = c + arg_{i=-1...+1} max[F(c + i\delta n) - Di^2\delta^2].\delta.n \tag{7.4}$$

x denoting the detected boundary point, $(2l+1)\delta$ is the profile length, where δ denotes the sampling step size and D biasing boundary detection towards nearby points. The boundary detection function is given as:

$$F(x) = \begin{cases} 1 \pm G \overset{limit}{proj} & \text{for } Q_k(x)[min_k, max_k]\forall k \\ 0 \text{ otherwise} \end{cases} \tag{7.5}$$

with,

$$G^{limit}_{project}(x) = (n.\nabla l(x)).\frac{g_{max}(g_{max} + ||\nabla l(x)||}{g^2_{max} + ||\nabla l(x)||^2} \tag{7.6}$$

Here,
$I(x)$ denotes the gray-value at the point x, ? is the gradient operator and g_max is the heuristic damping factor. Boundary detection functions are defined individually for each triangle using triangle specific min, max intervals. SSimulated Searchässigns an optimal boundary detection function to each triangle. The selection process works as follows for each triangle independently:

1. The pose of the triangles in the reference meshes is slightly disturbed.
2. The boundary detection process using the given boundary detection function is performed.
3. The residual error between the detected point and the reference position is recorded for all tested displacements and all function candidates.
4. The candidate with the smallest simulated residual error is finally selected.

The annotated images and corresponding meshes are created by the following approach. In the first step the mesh is adapted to very few 13 manually annotated images. The resulting reference meshes are sufficient to train a first set of boundary detection functions by SSimulated Search". This initial model is then adapted to a larger set of images and the result is thoroughly refined, e.g. by a clinical expert. With the new reference images and meshes, we then train a second set of boundary detection functions. The process of automatic adaptation, manual refinement, and training of new boundary detection functions is continued until sufficient reference images and meshes are available. After defining the boundary detection function is trained using the 35 data sets. Following are the steps:

1. Boundary detection functions of 4 chamber heat model are mapped onto the high resolution heart model with great vessels.
2. Resulting model is adapted to 5 out of 28 data sets where all the great vessels are visible and distinctly identifiable. The segmentations are finely corrected and boundary detection functions of the great vessels are trained using 7 + 5 data sets.
3. The resulting data set is the used to segment all 35 data sets and thorough corrections are made. For aorta, inferior vena cava and coronary sinus reference segmentation could only be obtained for a subset of the 35 data sets (26, 12, and 16 respectively) as these structures where not clearly visible or distinctly identifiable because of which accurate segmentation was not possible and hence separate trainings were done for them.

3.2 Model based Adaptation Framework at Work

Automatic adaptation of the generic shape model to an image is typically achieved in several steps. The heart chambers are first localized and the low resolution heart model is positioned. Orientation alignment is corrected by matching the heart chambers and vessel trunks to the image using similarity transformation. Then, the descending aorta is detected. Parametric adaptation of the heart chambers and the most distal segments of the descending aorta are achieved using multi-linear transformations. In subsequent steps the model is accurately adapted using deformable adaptation while tubular segments are successively activated. After activating the resulting tubular segments the most distal parts of the descending aorta, the high resolution heart model is initialized. Deformable adaptation is done on remaining segments with subsequent activation. The well adapted mesh regions are frozen or fixed to speed up the process.

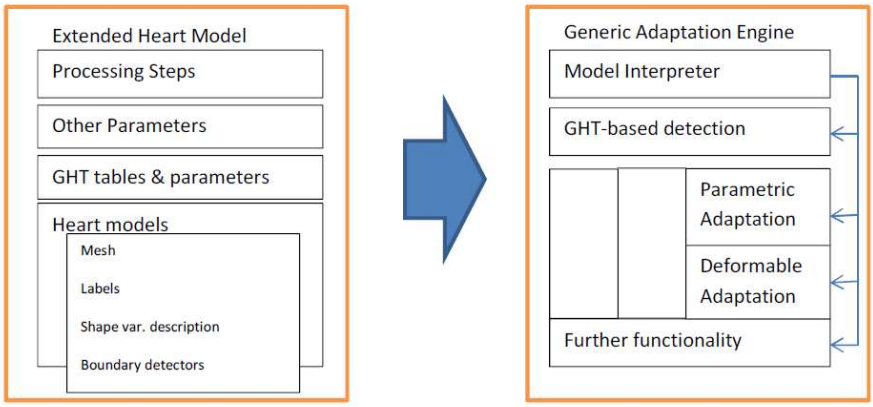


Abb. 7.6: Overall architecture of the segmentation framework

This process is actually a configurable algorithmic framework that is called the adaptation engine. The architecture of the adaption engine is illustrated as follows (figure 6):

Following are the techniques provided by the adaptation engine:

1. First, we use the generalized Hough transform (GHT) to roughly localize the organ in the image and adapt the size.
2. Secondly, location, orientation and scaling are refined. To this end, boundary detection is performed for each mesh triangle and the parameters of the similarity transformation are modified to minimize the sum of squared distances between the mesh triangles and the detected boundaries. Boundary detection and parameter refinement are iterated until no major changes are observed.
3. Thirdly, the parameters of the combined linear transformations that characterize the shape variability are adapted by iterating boundary detection and parameter refinement.
4. Finally, a deformable adaptation is performed.
5. Again boundary detection and mesh refinement are iterated until convergence. In this phase, the locations of all mesh vertices are optimized during mesh refinement until a balance is reached between the geometric constraints defined by the generic shape model and the forces attracting the mesh to the detected boundaries. (See figure 7)

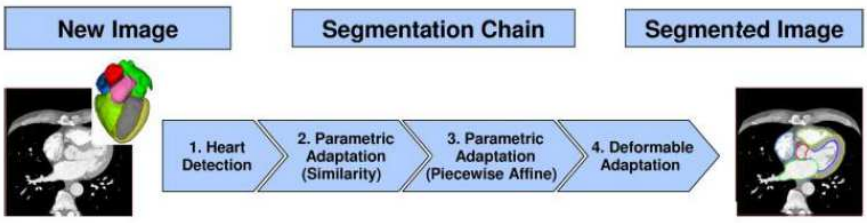


Abb. 7.7: Flowchart illustrating the chain of modules making up the coarse-to-fine algorithm for whole heart segmentation

The process of model adaptation can be modified in various ways. To speed up model adaptation, the adaptation process described above may be done with a low-resolution mesh model and a final deformable adaptation step with a high-resolution mesh model may be added. Reliable adaptation of vascular structures can, for instance, be achieved by adapting the four heart chambers first and successively activating the tubular segments representing the vascular structures. The information about the adaptation steps that are actually carried out, the mesh resolution or other parameters related to the different steps as well as information about the adaptation order of different model parts is contained in a control file of the generic organ model.[5]

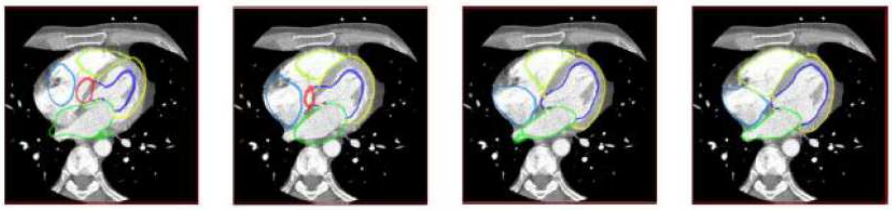


Abb. 7.8: State of the mesh adaptation after each step of the segmentation chain. (a) GHT-based heart localization. (b) Parametric adaptation (similarity transformation) (c) Parametric adaptation (piecewise affine transformation) (d) Deformable adaptation.

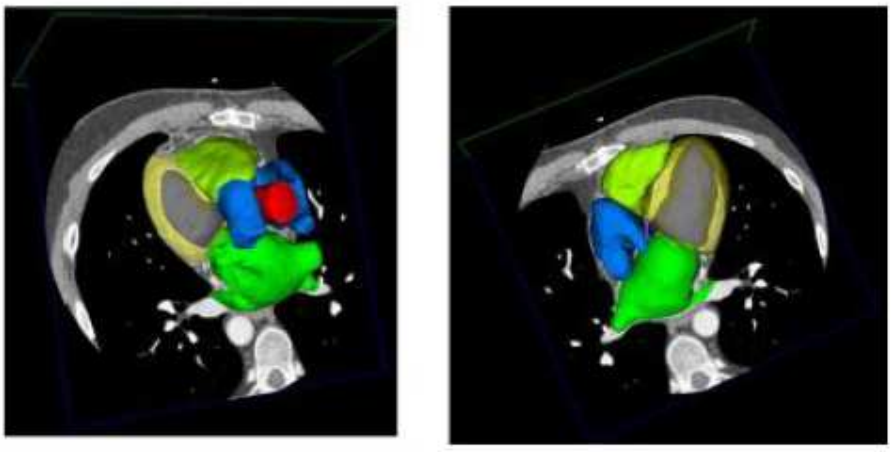


Abb. 7.9: Three-dimensional visualization of the cardiac model and CT image after automatic adaptation

4 Atlas based Segmentation

In this, information on anatomy, shape, size, and features of different, organs, soft tissues is compiled in the form of atlas or look up table (LUT). Atlas guided approaches are similar to co-relation approaches and the plus point of atlas based approaches is that it performs segmentation and classification in one go. Atlas based segmentation approaches are among the third-generation algorithms. If refined expertly, certain atlas based methods can compete with manual segmentations although atlas selection, atlas registration procedure, and the manual tracing protocol used in atlas formation are factors that can affect performance. However, they face limitations in segmenting complex structure with variable shape, size, and properties and expert knowledge is required in building the database. [**?**]

Xiahai Zhuang et al. have proposed an atlas based segmentation framework for whole heart segmentation of cardiac MRI in their paper: „A registration based propagation framework for automatic whole heart segmentation of cardiac MRI"which we will explain in the following section.

4.1 Registration based Propagation Framework

The paper proposes a registration framework that conserves the structure of the heart along with dealing with variability if shape in individual cases. The framework is composed of two algorithmic components; Locally Affine Registration Method (LARM) and a flexible registration method using free form deformations with adaptive control point status (ACPS FFDs).

LARM is used to obtain a robust initialization of various substructures of the heart such as the 4 chambers and the major vessels. Transformations are then applied to deform the atlas but at the same time the overall shape is maintained. After the image has been initialized using LARM finer local details are refined using ACPS FFDs. This scheme makes use of the atlas image for to control the deformations to avoid myocardial leaking by one to one mapping of the epicardium in atlas with epicardium in the unsegmented image. Once the segmentation is done it is propagated using inverse transformation based on Dynamic Resampling And distance Weighting interpolation (DRAW). (See figure) The following sections will describe in detail each component of the framework.

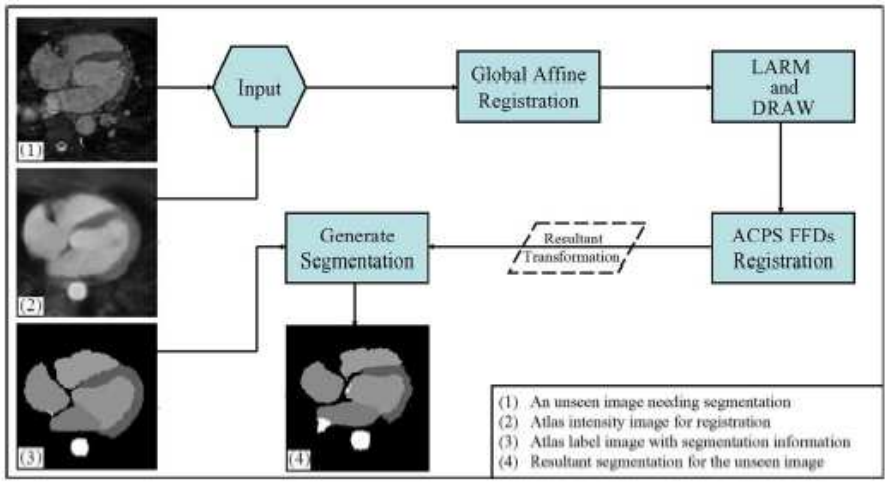


Abb. 7.10: The framework of automatic whole heart segmentation based on atlas propagation

4.1.1 Locally Affine Registration Method In contrast to global affine transformation which cannot provide accuracy a registration using locally affine transformations is recommended especially in case of heart anatomy due to large variability in shape among subjects. It also enables flexible registration of the heart model with high DOFs. LARM (Locally Affine Registration Method) assigns local transformation to each substructure specifically thus maintaining original topology across various subjects.

In this method let V_i be the set of predefined local regions which have a minimal distance between each other; let G_i bet the set of assigned local affine transformations. To achieve a global non-linear transformation T from G_i, a direct fusion based on the distance weighting interpolation is used, given as:

$$T(x) = \begin{cases} G_i(x), x \in V_i,\ i = 1...n \\ \sum_{i=1}^{n} w_i(x) G_i(x_i),\ x \notin \bigcup_{i=1}^{n} V_i \end{cases} \tag{7.7}$$

$w_i(x)$ Is a normalized weighting factor related to distance $d_i(x)$ between point x and region V_i.

$$w_i(x) = \frac{1/(d_i(x))^e}{\sum_{i=1}^{n} 1/(d_i(x))^e} \tag{7.8}$$

Where, e controls the locality of affine transformations. These transformations can also be used to for global affine transformations of boundary of ROI. However error can happen in two cases:

1. Overlapping of local regions after individual transformations
2. Due to large displacements of local regions folding may occur

The overlapping issue can be resolved by introducing a minimum distance between over lapping structures using morphology dilation (10 mm as done in implementation mentioned in paper). The folding problem is solved by regularization step monitoring the Jacobian J_T of the deformation field.

$$J_T = \sum_{i=1}^{n} \frac{\partial w_i}{\partial x}.G_i + \sum_{i=1}^{n} w_i.\frac{\partial G_i}{\partial x} = \nabla W.G^T + \nabla G.W_T \tag{7.9}$$

If the determinant of the Jacobian, $det(J_T)$ drops below a threshold ($det(J_T) < 0.5$) a new image pair is generated from the original one using the current deformation and the locally affine transformation model is reset to identify for a new optimization process. Hence, the resultant transformations of the registrations are concatenation of the series of transformations.

In the following figure pseudo code of proposed LARM is given:

For each optimization step
  Global transformation, $F(x) = T \circ G$, and
    $T = T_m \circ T_{m-1} \circ \ldots \circ T_1$
  Optimize global affine G
  Optimize local affine G_i of T_m, compute derivatives based on (8)
  Regularization step:
    For each region V_i in $\{V_i\}$
      Overlap correction using (3)
      If V_i needs correction and is corrected, then
        If $|V_i| = 0$, then END registration
        else re-compute distance transformation of V_i for (2)
    End of each region V_i in $\{V_i\}$
    If MIN($\det(J_{T_m})) < 0.5$, then
      new a T_{m+1}, $T = T_{m+1} \circ T$, and $m = m + 1$
  End of Regularization
End of Optimization

Abb. 7.11: Pseudo-code of the proposed LARM

4.1.2 Adaptive Control Point Status Free Form Deformations (ACPS FFDs) After LARM initializes the substructures fine tuning is required to get a more accurate registration of the image. Using high degree of freedom (DOFs) we can do non rigid registration. This is done by having status associated with each control point in the mesh of the non-uniform rational B-Splines. The status is computed either based on reference image measures or based on joint mage pair measures. This status remains constant during the registration process. This technique significantly improves performance computationally. There are however two challenges while applying this technique to cardiac MRI registration. Firstly the image may contain background of adjacent tissue images such as that of lunch and liver which we are not interested in for segmentation. This however should not affect the quality of segmentation of the area of interest. Secondly, if the status is present for the control points and it remains constant and some points may remain inactive; this could result in decreased degree of freedom for model adaptation. To resolve these challenges, the suggested technique is to set the status of control points adaptively for each registration iteration, hence the ACPS (adaptive control point status). The status settings based on updating information can be extended as follows:

$$Status(\varphi_i) = \begin{cases} \text{active, if } T_i(\varphi_j) \in M_s \\ \text{passive, if } T_i(\varphi_j) \in M_s \end{cases} \tag{7.10}$$

Where, ϕ_j is the coordinate of control point φ_j and M_s is a mask image generated based on prior knowledge. The image mask enabled selection of region of interest while keeping the background image from effecting by avoiding activation of control points on adjacent tissue images with indistinct boundaries.

4.1.3 Dynamic Resampling and Distance Weighting Interpolation: DRAW Using the definition already present in the floating image (the atlas) which is considered as the reference image, inverse transformation is applied to the reference image to get inverted version of general dense displacements.

In figure 13, we can see diffeomorphic transformation T mapping a coordinate t in the reference image, to reference space in the floating image (atlas) having coordinate s.

$$s = T(t) \tag{7.11}$$

To compute inverse $T^(-1)$:

$$t = T^(-1)(s) \tag{7.12}$$

For the given set of scatter points s_i given by the four green points we can inverse transformation can be computed as following:

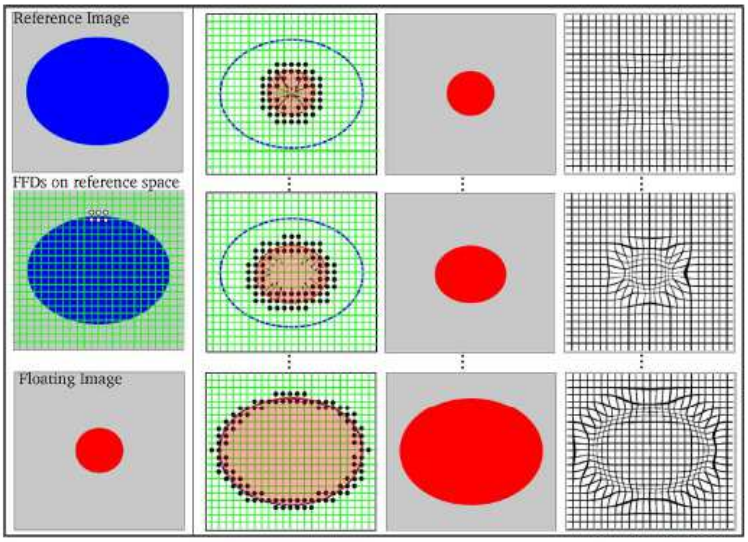


Abb. 7.12: Registration of an ellipse, the reference image and a circle, the floating image using adaptive control point status FFDs. The white dots in the image of middle (a) are special control points explained in the text. (b) FFD meshes, the contour of the ellipse, and the contour of the inverse-transformed floating image. The black dots are activated control points. The arrows demonstrate the registration driving forces and no arrow means a convergence of the registration. (c) Floating image inversely transformed into the reference space at different registration steps. (d) Deformed FFD meshes at different registration steps whose concatenation gives the resultant transformation.

$$T_I(s) = s + \sum w(s_i).(t_i - s_i) \tag{7.13}$$

Where s_i transformations by T from are set of points t_i in the reference space and $w(s_i)$ is the normalized weighting function computed from the distance between s_i and s. The accurate estimation can be achieved if these two conditions are fulfilled. Firstly, closest point s_i is found from each octant (3D) or from quadrant (2D) of s within one pixel size. The s_i should be transformed from the scatter points t_i which are within the volume if one pixel or voxel size. The diffeomorphic transformation from reference image to float image should be within the volume enclosed (red area in figure 13) and the interpolation error should not be larger than the enclosed volume. However the results may be affected because firstly, as shown in the figure 14:

The points in the reference image may be too sparse in the sample volume so the first condition for one to one interpolation may not be present. To solve this issue resampling of more points should be done and then transformation should be applied to get the floating image space as shown in figure 14. Secondly, the transformation may cause isotropic contraction and the scatter points may not be within one pixel size in the reference image space. Again to solve this issue one may have to do resampling of more points in orthogonal direction to provide a closer scatter point set which may be within the one pixel size. (See figure 15):

4.2 Registration based Propagation Framework at work

The segmentation framework includes three registration steps:

1. First a global affine registration is applied to localize the heart. This is usually challenging because of variability in field of view or orientation of the organ however in this particular case it was easy because of similar orientation field of view. It starts by using the complete global information resulting in coarse localization which will also include the periphery organ tissue images as well. Then a mask is used to cover the region within 20 mm to the heart of the atlas to focus on registering the heart structure.

2. The LARM is used to further initialize the substructures. This is a three stage process. In the first stage only two local affine transformations are used. The corresponding local regions are atrial

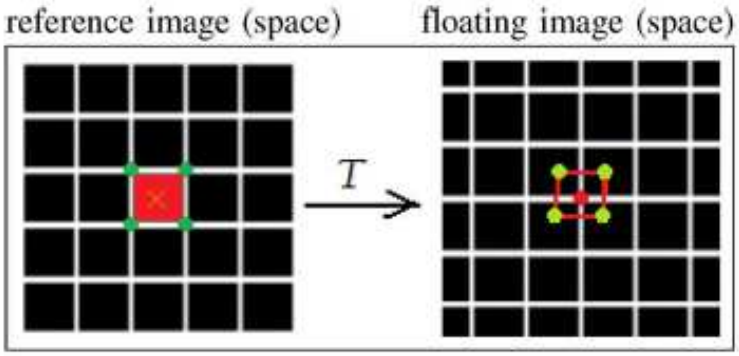


Abb. 7.13: DRAW using inverse distance interpolation from forwardly transformed scatter points

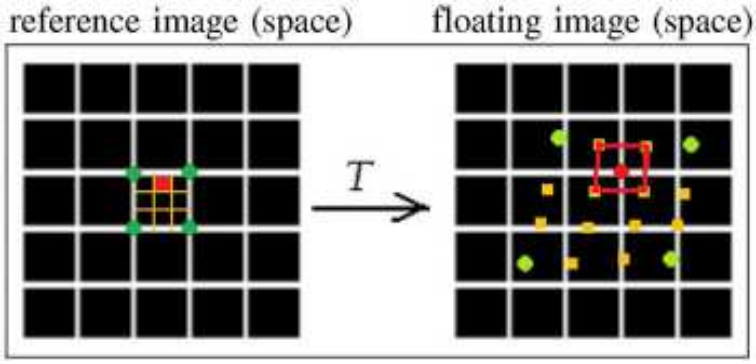


Abb. 7.14: Resampling more points within a pixel when the transformation field causes dilation

region which includes the two atria and great vessels, and ventricular region which includes the two ventricles and myocardium. In the second stage, the right ventricle and the right atrium are separated from the regions defined in the first stage to include another two affine transformations. Finally, seven local regions, including two ventricles, two atria, pulmonary artery, ascending aorta and aortic arch, and descending aorta, are used to achieve a seven-local-affine LARM. As the atlas image needs to be defined in the reference image in LARM, the resultant transformation of LARM is inverted using DRAW to get the transformation defined from the unseen MR image to the atlas.

3. Finally, registration using ACPS FFDs is employed to refine the local detail. This registration mainly contributes to the alignment of the endocardial surfaces where the boundaries are subsequently fine tunes it to 10 mm.

5 Empirical Findings

5.1 Validation Procedure and Results (Ecabert et al.)

For the validation of the model based adaptation framework CT images that reflected actual clinical variability in quality, noise and field of view were used. As previously explain a data set of 37 was reconstructed extracted from cardiac cycle gathered from 17 adult patients. The images gathered for validation were different from those gathered for model building and training so as to avoid any bias. A contrasting agent bolus was injected so as to clearly differentiate the chambers and the vessels which resulted variability of grey level distribution due to difference in arrival times of the contrasting agent. This strongly affected the contrast between the most distal parts of the vessels and the surrounding structures. Aortic arch was not covered in all images used as training data.

Accurate segmented models of the test images were obtained through a multistep process resulting in a thoroughly annotated mask which will enable detailed manipulation. These expert edited image masks are considered ground truth annotations in this case. From these ground truth meshes are generated. Several error metrics have been used to measure the quality of the segmentation results. To measure the

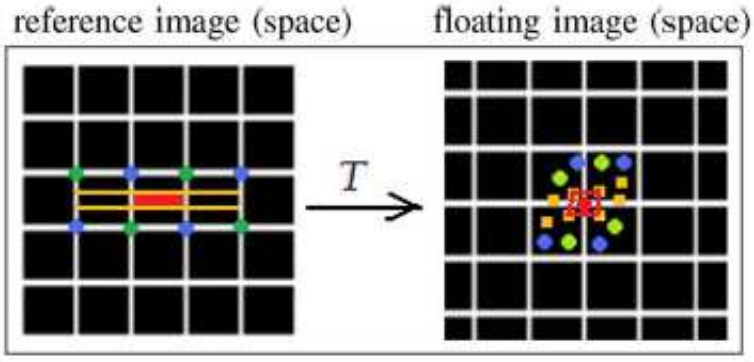


Abb. 7.15: Resampling more points in orthogonal direction when the transformation field causes anisotropic contraction in one direction

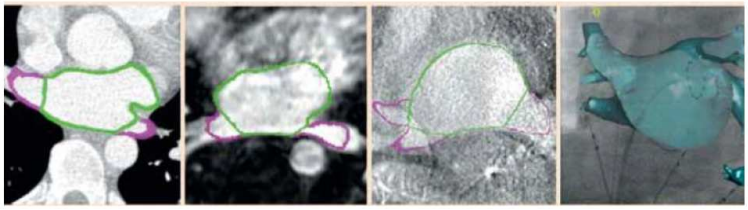


Abb. 7.16: From left to right: CTA, MRA and rotational X-ray image with the anatomy of the left atrium and pulmonary veins outlined and an overlay of a patient-specific LAPV model onto fluoroscopy data.

accuracy of surface detection the authors use the concept of constrained point-to-surface (CPS) distance. Given by the equation:

$$\epsilon_{\Pi(r)}(c_i^{adapt,j}, c_i^{ref,j}) = min_{x \in \Pi(r, c_i^{ref,j})} ||c_i^{adapt,j} - x|| \tag{7.14}$$

This calculates the smallest Euclidean distance between the triangle center of the adapted mesh in the image and a surface patch of a certain geodesic radius surrounding the corresponding triangle center in the reference mesh. As this distance is not symmetric (from one mesh to another) we define mean CPS distance as:

$$\epsilon_{mean}^{CPS} = \frac{1}{N_1.M} \sum_{j=1}^{N_1} \sum_{i=1}^{M} \frac{1}{2} (\epsilon_{\Pi(r)}(c_i^{adapt,j}, c_i^{ref,j}) + \epsilon_{\Pi(r)}(c_i^{ref,j}), c_i^{adapt,j}) \tag{7.15}$$

The inner summation results in unsigned distances or errors for the whole mesh or for specific substructures. The outer summation is for available data sets. The next measure taken into account is centerline to centerline distances which is used to measure the accuracy of the vessel course. For this constrained point-to-centerline (CPC) distance is defined as a sequence of straight line segments connecting the centers of successive vertex rings of the tubular mesh:

$$\epsilon_{\Lambda(l)}(p_i^{adapt,j}, p_i^{ref,j}) = min_{x \in \Lambda(r, p_i^{ref,j})} ||p_i^{adapt,j} - x|| \tag{7.16}$$

This gives the smallest Euclidean distance between a center point and an adapted centerline in an image and an interval of a certain length (set at 10 mm) surrounding the corresponding point in the reference centerline. Similarly we define the mean CPC:

$$\epsilon_{mean}^{CPC} = \frac{1}{N_1.N_s} \sum_{j=1}^{N_1} \sum_{i=1}^{N_s} \frac{1}{2} (\epsilon_{\lambda(l)}(p_i^{adapt,j}, p_i^{ref,j}) + \epsilon_{\lambda(l)}(p_i^{ref,j}, p_i^{adapt,j})) \tag{7.17}$$

(where N_s is the number of segments composing a centeline.)

The third measure is based on volume overlap which has important applications in clinical practice. The adapted and reference meshes are converted into annotation masks with voxel resolution of $0.2^3 m^3$.

This fine resolution will enable reduced inaccuracies. The equation given as following is the relative volume overlap between a region enclosed by adapted mesh and the corresponding region in ground truth annotation:

$$\epsilon^{overlap}(R^{adapt,j}, R^{ref,j}) = \frac{|R^{adapt,j} \bigcap R^{ref,j}|}{R^{adapt,j}} \tag{7.18}$$

The mean volume overlap is given by:

$$\epsilon^{overlap}_{mean} = \frac{1}{N_1} \sum_{j=1}^{N_I} \frac{1}{2} (\epsilon^{overlap}(R^{adapt,j}, R^{ref,j}) + \epsilon^{o} verlap(R^{ref,j}, R^{adapt,j})) \tag{7.19}$$

For vessel like structures the volume overlap since the mesh triangles should not slide along the image boundaries:

$$\epsilon^{overlap}_{e}(R^{adapt,j}, R^{ref,j}_{[i-e,i+e}) = \frac{|R^{adapt,j} \bigcap R^{ref,j}_{i-e,i+e}|}{|R^{adapt,j}|} \tag{7.20}$$

The equation for calculating mean is also changed introducing N_s which represents number of rings in the vessel structure.

$$\epsilon^{overlap}_{mean} = \frac{1}{N_I.N_s} \sum_{j=1}^{N_I} \sum_{i=1}^{N_s} \frac{1}{2} (\epsilon^{overlap}_{e}(R^{adapt,j}, R^{ref,j}_{[i-e,i+e]}) + \epsilon^{overlap}_{e}(R^{ref,j}, R^{adapt,j}_{[i-e,i+e]})) \tag{7.21}$$

5.1.1 Results and Discussion The CPS distance for the whole mesh is between 0.31 and 1.16 mm with a mean of 0.70 mm for the 37 data sets used during validation. The mean point to surface error $0.50 - 0.82$ mm for the heart chambers and $0.60 - 1.32$ mm for the visible parts of the great vessels is reported. The volume overlap for chambers is $94.0 - 96.2\%$ and $70.4 - 95.2\%$ for great vessels. Overall the results demonstrate that the proposed framework can be applied to adapt a mesh to both round (chambers) and tubular structures (vessels) by properly orienting the deformable shape models, which is also the contribution of this paper.

5.2 Validation Procedure and Results (Zhuang et al.)

The test data consisted of 37 MR volumes on the end diastolic phases is used as sample un-segmented images in the experiment. Among these 19 cases were confirmed pathologies so as to enable identification of various cardiac anomalies. The subjects were aged from 5 to 80 hence displaying wide variety of heart shapes. To avoid the advantageous bias of using an atlas with similar heart shape to segment the unseen data, a training dataset of images from 10 separate healthy volunteers was acquired for the construction of the atlas. All these data were manually segmented either done by fitting a deformable mesh model with manual corrections or using a semi-automated editing tool using a commercially available tool, this was done by either a clinician or a research associate with knowledge of heart anatomy. Two different measures were used to evaluate the accuracy of a segmentation result, the surface distance measure and the volume measure. The surface measure computes the surface to surface distance between the propagated segmentation and the standard segmentation. The surface distance is calculated using each surface point from the propagated segmentation to the triangle defined by three closest surface points in the standard segmentation. Six surfaces are considered to compute the mean surface distance, the standard deviation etc. Volume measurements are also done using dice coefficient, volume overlap both used to identify overlap of substructures (maximum being 1 and no overlap being 0) and volume difference.

The evaluation is done in three steps. 19 pathological cases are used to test the performance of the proposed approach to different heart forms or atlases. From the set three atlases are tested. First a healthy reference shape is extracted referred to as One Shape. Then mean shape of five random training data is used as reference shape referred to as Five Shape. Then the mean of all 10 training data set is used to get the mean reference shape referred to as Ten Shape. In the next step the proposed improvements as proposed by the registration algorithms is assessed against the Ten Shape atlas. The test data includes all the 37 cases for which the following computations are performed:

1. A single affine for global localization and traditional FFDs for refinement (Affine_FFDs).
2. A single affine for global localization then LARM, and then traditional FFDs for refinement (LARM_FFDs).
3. The proposed framework, a single affine for global localization, then LARM for substructure initialization, and then finally ACPS FFDs for refinement (LARM_ACPS).

For all FFDs multi resolution meshes are used. Finally using the proposed framework and the Ten Shape atlas results in segmentation output of the 37 test data.

5.2.1 Results and Discussion The following figure illustrates the root mean square surface distance errors of the 19 pathological cases:

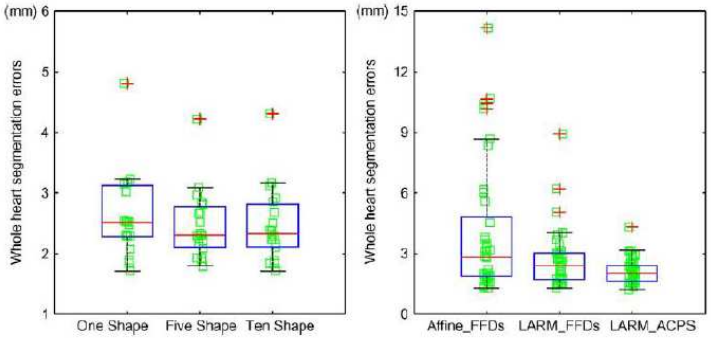


Abb. 7.17: Whole Heart segmentation errors using the rms surface-to-surface error measure, Left: the errors of the 19 pathological cases using the proposed segmentation approach combined with the three different atlases. Right: the errors of the 37 cases using the three different segmentation frameworks.

The mean and standard deviation of the errors are as following:

1. using One Shape atlas: 2.63 ± 0.70 (mm)
2. using Five Shape atlas: 2.47 ± 0.57 (mm)
3. using Ten Shape atlas: 2.47 ± 0.61 (mm)

From the above data we can see that using Five Shape atlas has the best results. However the evidence from performance test showed that segmentation of pathological data is not significantly different by using an atlas from the mean shape of a training set that has larger number of healthy volunteer data however if only one subject is used it may lead to performance bias.

The above table lists the surface to surface errors of the proposed segmentation method for each substructure category. The root mean square error, the mean error, the standard deviation and the percentage of error ranges are presented. For average whole heart segmentation more than 95 surface to surface error is within 5 mm.

Briefly nine different pathologies were used to test the performance of the proposed registration based segmentation propagation framework. There was clearly no significant difference found by using 3 atlases that were constructed from training data set extracted from healthy subjects when used to segment the pathological cases. However LARM and ACPS FFDs enabled significant improvement in accuracy by reducing errors. These enabled refinement of local topology in finer detail. The proposed segmentation framework achieved an root mean square surface-to-surface error of 2.14 ± 0.63 mm and a Dice measure of 0.84 ± 0.05 between the propagated segmentation and the standard segmentation.

5.2.2 Limitation According to the authors there are three limitations of the proposed work:

1. Using the intensity-based registration it is difficult to consider the thin regions which can be critical in separating substructures in some cases. For example, the thin membrane between epicardium and liver or the thin atrial wall between the two atria can be displayed as a thin region whose intensity significantly differs from its neighboring regions. However, the intensity-based registration may fail to incorporate this information due to their relatively small sizes.

2. Secondly, the propagation can have a big variation in the clear identification of the valves.
3. This approach takes relatively longer computation time to finish, 2 4 h per volume, compared to the deformable model- based, boundary-searching techniques.

6 Future Work

The model based adaptation framework as proposed by Ecabert et al. is fast and facilitates acceleration of the image analysis process unlike the registration based propagation framework as proposed by Zhuang. The proposed framework along with the simulated search for training the boundary detection functions can provide adaptation to new segmentation tasks with a reasonable amount of computation effort. Future work can include detailed segmentation and identification of finer anatomical structures such a brain and using this technique to process images from other sources such as MRI etc. The future work for registration based propagation framework as proposed by Zhuang et al. may include work on the limitations. Boundary searching technique can be incorporated for improving boundary detection which will have more accurate results as compared to the intensity based gradient approach.

As for computational performance improvement it is suggested use parallel processing such as using graphical processing units that may accelerate the segmentation process. Accuracy can be improved using multi classifier techniques for segmentation and propagation along with multi-atlas techniques.

Literaturverzeichnis

[1] World Health Organization *Report on Cardiovascular Diseases* 2013.

[2] Web Article *Heart*http://www.innerbody.com/image/card01.html

[3] Online Health Library of UC SanDiegoArticle *Magnetic Resonance Imaging and Computer Tomography* http://myhealth.ucsd.edu/85,P01289

[4] Neeraj Sharma and Lalit M. Aggarwal *Automated medical image segmentation techniques* 2010: Journal of Medical Physics.

[5] Olivier Ecabert et al. *Segmentation of the heart and great vessels in CT images using a model-based adaptation framework* 2011: Medical Image Analysis Conference.

[6] Xiahai Zhuang et al. *A Registration-Based Propagation Framework for Automatic Whole Heart Segmentation of Cardiac MRI* 2010: IEEE Transactions on Medical Imaging.

[7] Weese et al. *The Generation of Patient-Specific Heart Models for Diagnosis and Interventions* 2010: Statistical Atlases and Computational Models of the Heart.

[8] Oliver Ecabert et al. *Modeling shape variability for full heart segmentation in cardiac CT images* 2006: Proc. SPIE Med. Imag.

[9] Martin et al. *Semi-Automatic Feature Delineation in Medical Images* 2004: Symposium on Information Visualisation.

[10] Peters J et al. *Automatic whole heart segmentation in static magnetic resonance image volumes* 2007: International Conference on Medical Image Computing and Computer Assisted Intervention.

Robust statistical shape models for MRI bone segmentation in presence of small field of view

Eugen Beck

Zusammenfassung

In this seminar paper we discuss a method for bone segmentation from MRI images. It is based on building a parametrized statistical model from (hand)-segmented images that is later used to segment new images by first finding good parameters and then deforming the model using a physics based simulation. We will also compare the method to other current approaches found in the literature and present its achievements and shortcomings.

Keywords: MRI, bone segmentation, femur, pelvis, hip, statistical shape models

1 Introduction

The paper under consideration is "Robust statistical shape models for MRI bone segmentation in presence of small field of view" by JÃ©rÃ´me Schmid, Jinman Kim and Nadia Magnenat-Thalmann ("the authors" from now on) published 2011 in Issue 1 of Volume 15 in the journal Medical Image Analysis. It deals with the task of bone segmentation. In this task the aim is to extract a 3D model of the bone-structure that appears in an medical image. The images the paper is performing this task on are acquired using magnetic resonance imaging (MRI) using different fields of view (FOV). This acquisition method has the advantage that in contrast to computer tomography the subjects are not exposed the harmful radiation, but the resulting images provide lower contrast between bony and soft tissues.

Accurate bone models are useful in the planning/post-evaluation of operations, e.g. of Osteoarthritis [1]. When obtaining MRI images one can trade between acquisition time, field of view. The paper focuses on the effects of a small FOV and how to train SSMs under this conditions. The area where the femoral head connects to the acetabulum is especially challenging, as the bones are separated by a few millimeters of cartilage.

2 State of the Art

The major approaches to medical image segmentation can be put into one of two categories: Low level approaches and high level approaches. The first ones use information present in the pixel-values of the image directly. Common techniques in this group are pixel-based (thresholding [2]), edge-based (Livewire [3]) and region growing [4], [5]. The second ones use information gathered from other segmentations to build a model that guides the segmentation approach. These methods use techniques such as the watershed transform [6] or point distribution models [7]. In addition the generalized Hough transformation can be used to initialize some model parameters [8].

The approach of the authors belongs to the category of high level approaches. The 3 main stages of SSM based models are "(i) construction, (ii) initialization, (iii) evolution" [1]. The authors devised a process to obtain robust multi-resolution SSMs which can be deformed to segment MRI images of the pelvis region.

3 Overview

The goal of the process to be presented is to build a 3 dimensional representation of a subjects hip and femur bone from an MRI image of the relevant region. To that end the authors first construct a

[1] [9], p.157 top right column

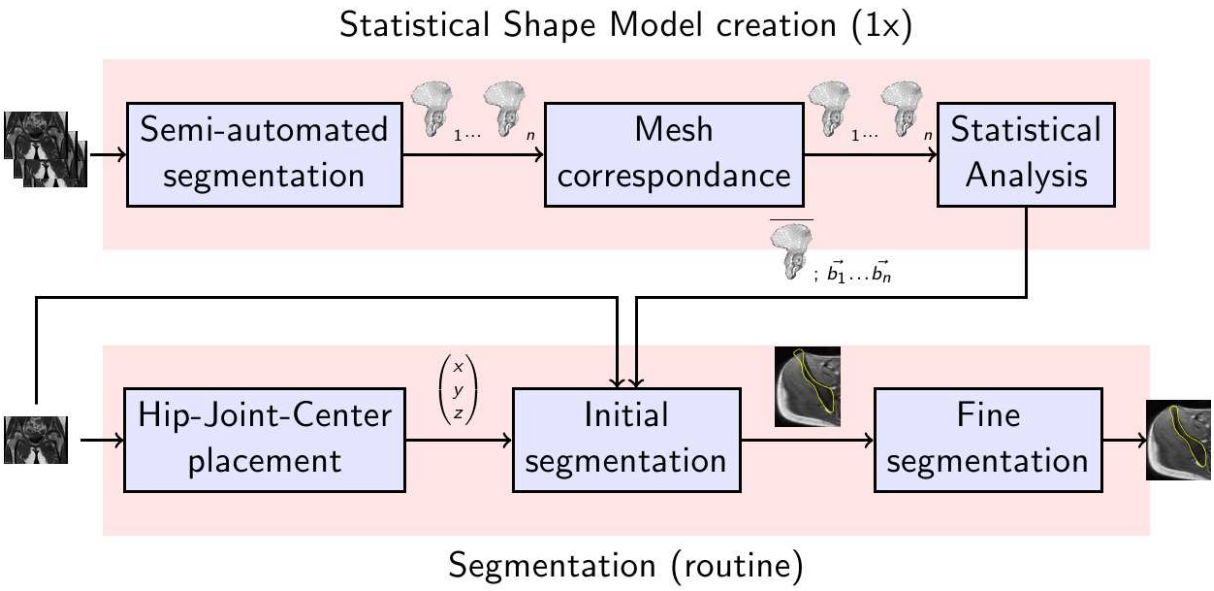


Abb. 8.1: Overview of the entire process

statistical shape model from a set of sample segmentations that captures the general structure of the bones to be segmented such that individual variability is captured with a small number of parameters. In the second step, the actual segmentation, the statistical shape model serves as a starting point to find a good segmentation by means of a physics inspired simulation.

In the first step the authors start by using a semi-automated segmentation approach to create meshes from a set of training images. These meshes are composed of vertices that do not have point-to-point correspondence. This means that two point with the same index in two different meshes are not located at the same point of the pelvis/femur. This is a necessary condition for to train a SSM. To satisfy this the authors use landmark sliding. Then a SSM is build. It consists of a mean shape and a transformation matrix. The transformation matrix captures the individual variances in bone structure using a relatively small number of parameters.

To segment a new image these parameters need to be estimated. Together with translation and rotation parameters. The translation parameters are provided by a human expert by locating the hip-joint-center in the image. The others are estimated using an evolutionary algorithm. But this only provides a rough estimate of the bone structure. In order to obtain a better segmentation the model is deformed by using a physics simulation where the forces that act on the model try to fit it to the bone structure while also enforcing smoothness and non-overlapping constrains.

A graphical overview of the process is given in figure 8.1.

4 Generating SSMs

4.1 Building Training Shapes

The shapes are represented using a 2D-simplex mesh as described in [10]. In a 2-simplex mesh each vertex P is described by a linear combination of 3 neighboring vertices P_i ($i \in 1, 2, 3$) and a parameter $h \in \mathbb{R}$. Let $\vec{p}$ be the surface normal of the plane formed by the points P_i. Then the point P is given as $P = \varepsilon_1 P_1 + \varepsilon_2 P_2 + (1 - \varepsilon_1 - \varepsilon_2) P_3 + h \cdot \vec{p}$. See figure 8.2 for a graphical representation. The authors use quasi-regular simplex meshes (hexagons).

In [9] chapter 2.1 shapes are defined as a set of points x_i plus a surface normal n_i. It is unclear to us how the simplex-mesh representation described in [10] can be transformed to such a form. To speed up computations multiple resolutions of meshes are computed. The method is based on the finding that each simplex mesh is dual to a triangulation ([12]). The dual triangle mesh for a simplex mesh can be obtained by swapping the roles of points and surfaces (see figure 8.3). To obtain a higher resolution mesh the edges of the triangles of the dual mesh are divided in half to produce 4 new triangles. These can be

converted back to the dual simplex mesh. The resulting mesh has the nice property that it reuses all of the points from the course mesh as seen in figure 8.2.

The authors generate the training shapes by a semi-automated process using techniques from previous papers of their group ([13], [11], [14]) using a template mesh of unknown origin.

4.2 Point-correspondence

In order to perform the generalized Procrustes analysis the points of the different training shapes have to be brought into correspondence, e.g. a point that is located at a specific anatomical feature in one shape should also be near/at the same anatomical feature in all other shapes[2]. The authors use the method described in [15]. The main idea is to slide a landmark on its tangent plane (constructed from nearby vertices) such that a certain distance function (TPS-energy) is minimized and then reproject it onto the surface of the shape. The landmarks that undergo this sliding process are the ones from the most coarse resolution as they "are regularly distributed and located near key anatomical features" [3]. For the reprojection they use the highest resolution mesh in order to minimize the error.

4.3 SSM construction

The goal of Procrustes analysis ([16]) is for each shape y^k to find a representation consisting of a mean shape $\overline{x}$ and transformation matrix Φ that are common to all shapes plus a set of parameters b^k and a shape transformation T^k such that

$$\|y^k - T^k(\overline{x} + \Phi b^k)\|$$

is minimized.

The first step to find these matrices and vectors is to align the shapes y^k using matrix T^k. In the alignment a transformation matrix is found for each shape that minimizes the total distance of the shapes to the mean shape. The mean shape is computed as follows:

$$\overline{x} = \frac{1}{K}\sum_{k=1}^{K} x^k$$

where $x^k = (T^k)^{-1}y^k$. These alignment transforms can have different forms. The authors use "rigid", "affine" or "similarity" alignment transforms. The latter ones use the singular-value-decomposition to minimize the distance between two shapes. It is noted that the alignment only occurs between two shapes. Thus an iterative approach is used to find the mean shape. One starts by defining an arbitrary shape as the mean shape. Then one after another the other shapes are aligned to the mean shape. After alignment a new mean shape is computed using the formula above. This causes the mean shape to change.

[2] As the correspondence is determined automatically, the points do not need to be at positions where domain experts would place anatomic landmarks

[3] [9], p. 158 center left column

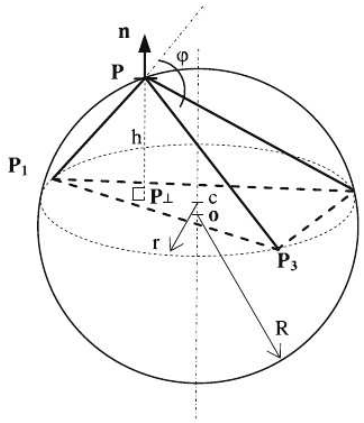


Abb. 8.2: Simplex surface local geometry, source: [11]

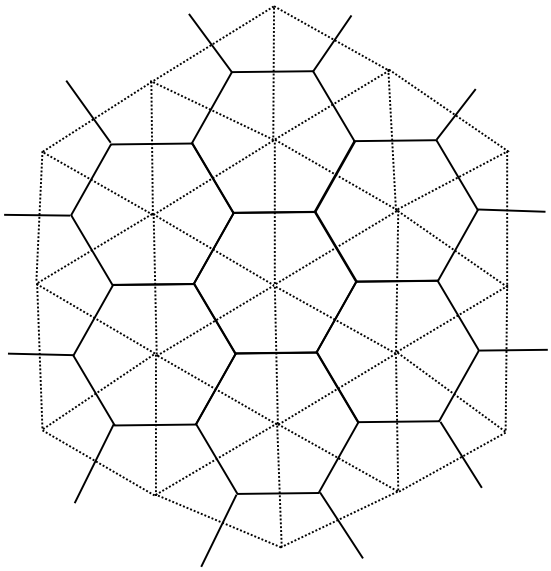


Abb. 8.3: solid lines: simplex mesh, dotted lines: its dual triangle mesh

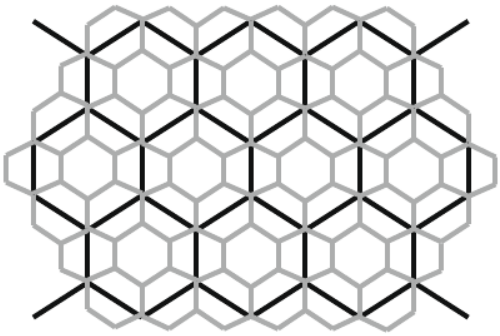


Abb. 8.4: Multi-Resolution scheme, source: [11]

Now with the new mean shape the other shapes can be aligned again. This allows to iteratively obtain an optimal mean shape. In [17] it is noted that in order to avoid the mean shape to drift "size and orientation should be fixed at each iteration".

Now x^k can be rewritten to $x^k = \overline{x} + \Delta x^k$. The next goal is to find a matrix Φ that allowed us to represent Δx^k with a small number of Parameters b^k ($\Delta x^k \approx \Phi b^k$). This will drastically reduce the number of parameters needed later to find an initial segmentation of the MRI image. To do this the authors perform principal component analysis. To do this the Δx^k are stored in a matrix ΔX (one in each row). The eigenvectors of $\Delta X^T \Delta X$, W, then form a new basis of the space of Δx^k and we can represent Δx^k in that space as $\Delta x^k = W z^k$. The important property here is that the eigenvectors with the smallest eigenvalues contribute the least to Δx^k, meaning dropping them causes the smallest change to Δx^k. Thus the reduced matrix W forms the matrix Φ and the equally reduced vector z^k form the shape parameters b^k.

The authors have chosen the number of dimensions to drop such that 99% of the total variance is preserved. In order to obtain shapes that are similar to the original shape the parameters b^k are clamped such that $-3\sqrt{\lambda_j} \leq b_j \leq 3\sqrt{\lambda_j}$ where λ_j is the j-th column of Φ. In addition the Mahalanobis distance from the mean (assuming a gaussian model) $\sum_j \frac{b_j^2}{\lambda_j}$ should be less than a parameter M, which is said to be derived from the χ_k^2 distribution without providing details. These restrictions are based on [18].

The process described above only makes sense if the shapes that are to be aligned are somewhat similar. This is not the case if the images have different FOVs. Thus the authors introduce weights w_{ij} for each component i of shape j and now compute the alignment w.r.t. these weights. For corrupted shapes (shapes with points missing) the weight of the missing points is set to 0 and thus excludes them from the alignment. To estimate such an alignment the authors use the EM-algorithm from [19].

5 Deformable Model Segmentation

The next step is to use the SSMs to segment an actual image. The segmentation consists of two stages: First a set of shape parameters b and transformation parameters T is determined using an evolutionary algorithm such that a cost function based on image energies is minimized. Then the model is deformed using forces that pull the points of the SSM towards points of minimal image energy along their normal. Forces based on the shape prior ([20]) and the shape's smoothness ([11]) and a collision detection and response technique ([11]) prevent the model from diverting too far from known shapes and into unreasonable optima.

5.1 SSM parameter estimation

For this the authors have chosen to exploit the specific structure of the hip joint. They state that the hip joint center (HJC) can usually be easily approximated on the image. Given its location in the image and in a shape, a rough global initialization can be performed by aligning the HJC of the shape with the HJC of the image. Subsequently the transformation parameter T can be restricted to a rotation around the HJC plus a small translation δ to account for errors in the placement of the HJC. The HJC is determined by the center of a least-squares fitted sphere of the vertices corresponding to the acetabulum or femoral head of the pelvis and femur models respectively. As there is a one to one point correspondence the vertices only have to be marked once on any model.

The cost function $C_f(y)$ to be minimized is a linear interpolation of two image energies $E^{ip} \in [-1, 1]$ and $E^g \in [-1, 1]$.

The first energy E^{ip} is based on the normalized cross-correlation (NCC) of two intensity profiles extracted along the normal of each point. The NCC of two vectors $a, b \in \mathbb{R}^n$ is defined as follows:

$$\mathrm{NCC}(a, b) = \frac{(a - \overline{a})^\top (b - \overline{b})}{\|a - \overline{a}\| \cdot \|b - \overline{b}\|} = \cos\theta \tag{8.1}$$

where $\overline{a}$ is the vector that contains the average of a in each component and σ_a is the standard deviation of a and θ the angle between $a - \overline{a}$ and $b - \overline{b}$. This makes the similarity measure invariant to changes in overall intensity. Then $E^{ip} = -\mathrm{NCC}$, as the energy is a cost function.

The second energy E^g is based on the normalized image gradient of [21]:

$$E^g(y_i) = \epsilon \cdot n_i \nabla_\gamma I(y_i) \tag{8.2}$$

$$\nabla_\gamma I(y) = \frac{\nabla I(y)}{\sqrt{\nabla I(y)^\top \nabla I(y) + \gamma^2}} \tag{8.3}$$

$$\gamma = \frac{\eta}{|\Omega|} \int_\Omega |\nabla I(x)| dx \tag{8.4}$$

where ϵ is 1/-1 when the expected gradient points inward/outward. η is the estimated noise level using the pseudo-residuals technique from [22].

The minimization is performed using the differential evolution algorithm of [23] where each population member is a set of parameters (T, b). In this algorithm the parameters are evolved in multiple generations. The members of a generation are obtained by using selection, mutation and crossover operations on the members of the previous generation. The restrictions for the translation δ and clamping b_i were enforced in this step aswell. The specific scheme used here is RAND/1/BIN, which means that the new population members were based on crossing over with a random member of the previous generation to which the difference of 1 population-member pair was added. The crossover points were determined using independent binomial trials.

5.2 Shape deformation

The process described above gives a rough segmentation based on the training shapes. To improve the segmentation the resulting shape is deformed based on the process described in [11][4]. In this paper the points are treated as particles with mass upon which some forces apply. A force $f_i(x_i)$ at point x_i is modeled as $f_i(x) = \alpha_i(y_i - x_i)$, where y_i is some other point to which the point x_i is pulled. α_i acts as a stiffness parameter and allows the user to control the deformation process. The two external forces f^{ip} and f^g pull towards the minimum of the energies E^{ip} and E^g. In addition two internal forces apply a force towards a more smooth surface and towards the closest representation using the SSM generated before.

The forces are then applied in a number of iterations until the point movement dropped below a small threshold (e.g. .5mm) using a stable Euler implicit scheme.

6 Experimental results

In this section we will at first describe the measures and the dataset used by the authors to evaluate their process. Then we will present the results of those experiments. Afterwards we will compare them to results found in other literature.

[4]Note that this paper has also been covered in the seminar on medical image processing in the year 2011

6.1 Evaluation Measures

The evaluation measures used are taken from [24]. They are:

- Average Symmetric Surface Distance (ASSD)
- Average Symmetric Root mean square Surface Distance (ASRSD)
- Maximum Surface Distance (MSD)
- Volumetric Overlap Error (VOE)

All this measures (at least as described in [24]) rely on a classification of voxels. This means that the mesh is presumably used to label each voxel of the image as 'bone' (inside the mesh) or 'not bone' (outside the mesh). These sets of voxels for two segmentations are referred to as A or B respectively. For the first three measures the voxels that have at lest one non-bone voxel in their 18-neighborhood (any voxel that shares a face or edge) are considered surface voxels. The set of surface voxels for a segmentation A is denoted by $S(A)$. For each of them one can compute the distance to the closest surface voxel of a surface voxel in segmentation B: $d(v, S(B)) = \min_{w \in S(B)} \|v - w\|_2$ using the euclidian distance.

The Average Symmetric Surface Distance measures the overall quality of the segmentation by averaging over the minimum distances for all surface voxels from both segmentations. It is defined as follows:

$$ASD(A,B) = \frac{\sum\limits_{v \in S(A)} d(v, S(B)) + \sum\limits_{w \in S(B)} d(w, S(A))}{|S(A)| + |S(B)|}$$

It is given in units of millimeters. For a perfect segmentation it returns 0 mm.

The Average Symmetric Root mean square Surface Distance seeks to punish large deviations stronger than small ones. This is achieved by squaring the individual distances and then taking the square root over the sum. It can be show (using the Cauchy-Schwartz inequality) that for any sequence of positive numbers x_i with a fixed sum $\sum_{i=1}^{N} x_i = c$ and a fixed length N, the term $\sum_{i=1}^{N} x_i^2$ is minimized by setting $x_i = c/N \ \forall i$. Thus of two segmentations with the same ASSD, the one with a more evenly distributed error will get a better ASRSD score. The formal definition is:

$$ASRSD(A,B) = \sqrt{\frac{\sum\limits_{v \in S(A)} d(v, S(B))^2 + \sum\limits_{w \in S(B)} d(w, S(A)}{|S(A)| + |S(B)|}}$$

Again the unit is millimeters and the perfect value is 0 mm.

The Maximum Surface Distance yields an upper bound on the segmentation error. Its definition is as follows:

$$MSD(A,B) = \max\{\max_{v \in S(A)} d(v, S(B)), \max_{w \in S(B)} d(w, S(A))\}$$

This measure uses millimeters as unit aswell and the perfect value is 0 mm.

The Volumetric Overlap Error is a popular measure that is useful for scenarios were the volume of the segmentation is important, e.g. for the segmentation of tumors. It measures the amount of voxels that are not shared by the two segmentations relative to the amount of voxels that are in either one of them. The authors only use it to determine if the global initialization was successful (VOE $< 60\%$). The formula is:

$$VOE = 1 - \frac{|A \cap B|}{|A \cup B|}$$

The unit is percent and the perfect value is 0%.

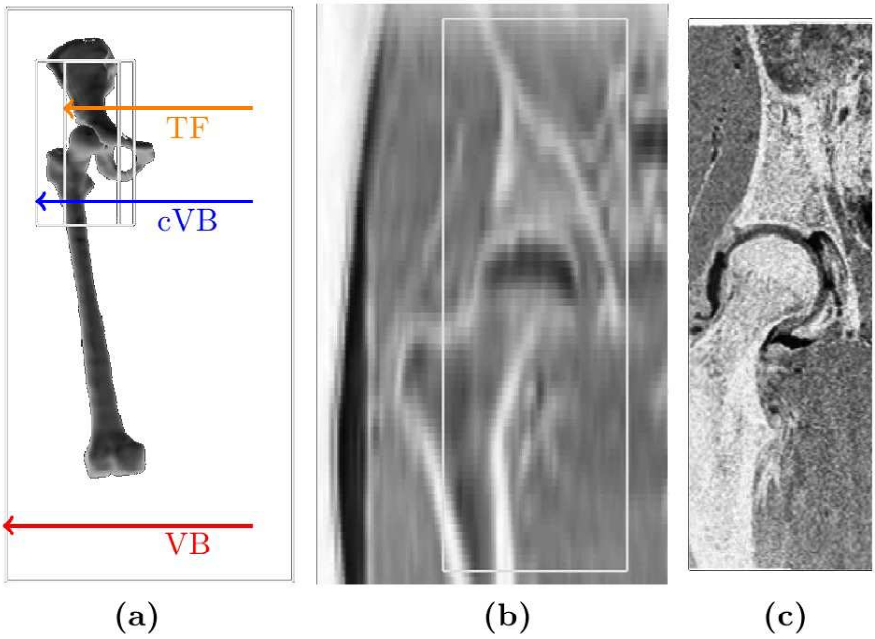


Abb. 8.5: (a) All modalities, (b) cVB, (c) TF, Source: [9]

6.2 Dataset

The authors created a dataset comprised of 2 MRI images from 43 female volunteers with an average age of 25.1 years using two protocols of the left and right hip on a 1.5T MRI device from Philips Medical Systems: The first image was acquired using the VIBE (VB) protocol[5], which yielded low resolution images with a wide FOV (covering both hip joins and a large part of the pelvis). The second one was acquired using the TRUE-FISP (TF) protocol[6] and has a low FOV, but a higher image resolution. In addition the VIBE images were cropped to produce images with a FOV that is slightly larger than that of the TF dataset. These images are referred to as the cVB dataset. An overview over the different FOVs can be found in figure 8.5.

6.3 Experiment-Settings

SSM construction: For each model four resolution-levels were generated. The SSM-scheme used were RAA (for small FOV images) and SSS (for VB) for the first three resolution levels. For the highest resolution level no SSM was generated and the evolution of parameters for this level is only guided by smoothness constrains. This allows to capture local details that are not captured by the SSM. At each vertex image intensities were captured along the normals. 5 IPs were captured at the shape interior and 20 at the shape exterior using a spacing of 0.5 mm. These samples were averaged for all vertices over all images and stored alongside the SSM.

Course initialization: For images where the full bone structure is visible the models are initialized using manually placed points at anatomical landmarks (7 for the pelvis, 8 for the femur) and then using thin plate spline (TPS) interpolation to register the SSM (for details see: [11]). According to the authors this placement of landmarks becomes too difficult for small FOVs and thus they are initialized using the evolutionary algorithm described previously. The first generation of parameters is initialized using uniformly drawn random numbers (within the range of allowed values). The evolution is stopped after 100 generations are created. It is not mentioned how large the population is.

Fine segmentation: The parameters for the model deformation are: $\beta = 0.5$ (both image energies are equally important). The contribution of external forces was set to 30% and the number of iterations for the 4 resolutions is: 300/100/20/5 (course to fine).

[5] Axial 3D T1, TR/TE=4.15/1.69ms, FOV/Matrix=35cm/256×256, resolution=1.367×1.367×5mm
[6] Sagittal 3D T2, TR/TE=10.57/3.63ms, FOV/Matrix=20cm/384×384, resolution=0.52×0.52×0.6mm

	ASSD	ASRSD	MSD	VOE
VB dataset				
initial segmentation	2.88 ± 0.68	4.12 ± 0.94	17.30 ± 4.70	38.48 ± 7.09
fine segmentation	1.21 ± 0.35	1.98 ± 0.48	9.65 ± 2.17	19.47 ± 3.77
cVB dataset				
initial segmentation	1.49 ± 0.46	2.44 ± 0.66	12.79 ± 3.49	23.99 ± 5.11
fine segmentation	1.12 ± 0.46	1.85 ± 0.61	8.98 ± 2.55	19.04 ± 5.44
TF dataset				
initial segmentation	1.80 ± 0.56	2.59 ± 0.89	14.63 ± 5.10	25.03 ± 6.32
fine segmentation	1.31 ± 0.44	1.84 ± 0.70	10.72 ± 4.09	18.89 ± 4.93

Tab. 1: Overall results for all datasets and their standard deviation. ASSD, ASRSD and MSD are in mm, VOE in %.

The cost function for the image energies used $\beta = 0.5$ and the force regulators α^{ip} and α^{g} were set to the same value.

6.4 Experiment-Results

The method was evaluated using a leaving-one-out (LOO) test procedure. The results are shown in table 1. The first interesting finding is that the segmentation on the cropped VIBE data performs better than on the complete image. This is in contradiction to the authors statement that smaller FOVs are harder. But when looking at the asymmetric error distance (see figure 8.6) one finds that the local error is very high at femur condyles which have a high individual variability and are located almost parallel to the axial plane and thus are subject to the partial volume effect. Thus the higher error rate for the VB dataset is partially due to unfavorable error computation. Especially the initialization seems to suffer from this (2.88 ± 0.68 for VB vs. 1.49 ± 0.46 for cVB), but as different methods are used this might no be a result of the characteristics of the different FOVs, but a problem with the initialization.

The TF overall results are worse than the cVB results. The authors see this as evidence that a smaller FOV is more difficult. Afterwards they also state that the higher resolution affects the minimization of the cost function (more local minima).

6.5 Existing art

In the recently published [1] the femoral head and acetabulum were segmented using a hybrid approach (preprocessing + thresholding + interactive improvement using Bayes decision rule). They used larger (110 hips) dataset of CT images that also included pathological samples. Their average subject age was 52 years. Their resolution was slightly lower (0.68mm×0.68mm×1.5mm) than that of the TF dataset (0.52×0.52×0.6mm) but also covered a larger area. They report an average ASSD of 1.22mm ± 0.98mm,

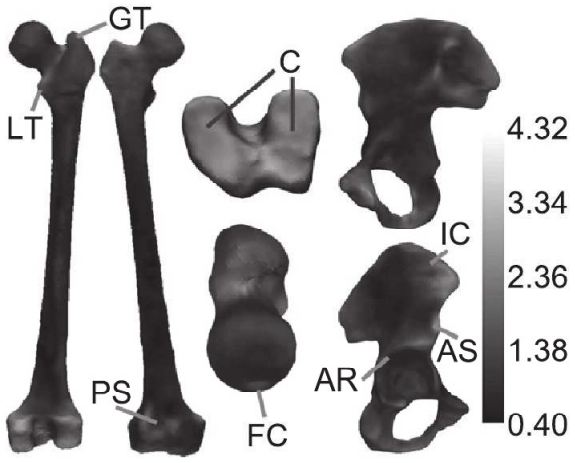


Abb. 8.6: Asymmetric error by location

which is between the results from the cVB and TF dataset. As CT images offer a better contrast between bony and non-bony structure this comparison is favoring [1].

In [25] only the pelvis bone is segmented using a statical distribution model build from CT and MRI images that is used to segment MRI images. They report an ASSD of 1.2377 ± 0.2723 and an ASRSD of 1.5968 ± 0.3686. The ASSD is slightly higher in this case than the ASSD of the hip bone in the VB dataset (1.15 ± 0.40), but the ASRSD is somewhat lower than that of the VB dataset (1.90 ± 0.52). Together with the smaller standard deviation this hints at a more regular distribution of the error.

All in all the results of the authors seem to be competitive even when comparing to more recent results, but as these results were obtained on different datasets it would be better to compare them on a common dataset.

7 Discussion

The authors do not mention in the abstract that their method uses the kinematic structure of the hip joint in order to perform initialization for small FOV images. This approach may be feasible in some other regions such as the knee, but is not applicable for non joint areas.

Another critical point is the data used: dataset of young and probably healthy (not mentioned in the paper) test subjects. Thus it is unclear whether the approach also works on pathological patients. But as the group is not based in a hospital (or similar institution) it would be difficult for them to obtain such images.

Last but not least they do not compare to existing literature. At least for the VIBE dataset such a comparison should have been possible. Thus it cannot be reliably evaluated whether their approach provides a real improvement over already existing approaches.

Literaturverzeichnis

[1] Cheng Y, Zhou S, Wang Y, Guo C, Bai J, Tamura S. Automatic segmentation technique for acetabulum and femoral head in CT images. Pattern Recognition. 2013;46(11):2969 – 2984.

[2] Dalvi R, Abugharbieh R, Wilson DC, Wilson DR. Multi-Contrast MR for Enhanced Bone Imaging and Segmentation. In: Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE; 2007. p. 5620–5623.

[3] Falcao AX, Udupa JK, Samarasekera S, Sharma S, Hirsch BE, de A Lotufo R. User-Steered Image Segmentation Paradigms: Live Wire and Live Lane. Graphical Models and Image Processing. 1998;60(4):233 – 260.

[4] Zoroofi RA, Sato Y, Nishii T, Sugano N, Yoshikawa H, Tamura S. Automated segmentation of necrotic femoral head from 3D MR data. Computerized Medical Imaging and Graphics. 2004;28(5):267 – 278.

[5] Nguyen NT, Laurendeau D, Branzan-Albu A. A new segmentation method for MRI images of the shoulder joint. In: Computer and Robot Vision, 2007. CRV '07. Fourth Canadian Conference on; 2007. p. 329–338.

[6] Grau V, Mewes AUJ, Alcaniz M, Kikinis R, Warfield SK. Improved watershed transform for medical image segmentation using prior information. Medical Imaging, IEEE Transactions on. 2004 April;23(4):447–458.

[7] Zheng G, Dong X, Rajamani KT, Zhang X, Styner M, Thoranaghatte RU, et al. Accurate and Robust Reconstruction of a Surface Model of the Proximal Femur From Sparse-Point Data and a Dense-Point Distribution Model for Surgical Navigation. Biomedical Engineering, IEEE Transactions on. 2007 Dec;54(12):2109–2122.

[8] Seim H, Kainmueller D, Heller M, Lamecker H, Zachow S, Hege HC. Automatic Segmentation of the Pelvic Bones from CT Data Based on a Statistical Shape Model. In: Proceedings of the First Eurographics Conference on Visual Computing for Biomedicine. EG VCBM'08. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association; 2008. p. 93–100.

[9] Schmid J, Kim J, Magnenat-Thalmann N. Robust statistical shape models for MRI bone segmentation in presence of small field of view. Medical Image Analysis. 2011;15(1):155 – 168.

[10] Delingette H. General Object Reconstruction Based on Simplex Meshes. International Journal of Computer Vision. 1999;32(2):111–146.

[11] Gilles B, Magnenat-Thalmann N. Musculoskeletal MRI segmentation using multi-resolution simplex meshes with medial representations. Medical Image Analysis. 2010;14(3):291 – 302.

[12] Delingette H. Simplex meshes: a general representation for 3D shape reconstruction. In: Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on; 1994. p. 856–859.

[13] Gilles B, Moccozet L, Magnenat-Thalmann N. Anatomical Modelling of the Musculoskeletal System from MRI. In: Larsen R, Nielsen M, Sporring J, editors. Medical Image Computing and Computer-Assisted Intervention - MICCAI 2006. vol. 4190 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2006. p. 289–296.

[14] Schmid J, Nijdam N, Han S, Kim J, Magnenat-Thalmann N. Interactive Segmentation of Volumetric Medical Images for Collaborative Telemedicine. In: Magnenat-Thalmann N, editor. Modelling the Physiological Human. vol. 5903 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2009. p. 13–24.

[15] Dalal P, Munsell BC, Wang S, Tang J, Oliver K, Ninomiya H, et al. A Fast 3D Correspondence Method for Statistical Shape Modeling. In: Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on; 2007. p. 1–8.

[16] Gower JC. Generalized procrustes analysis. Psychometrika. 1975;40(1):33–51.

[17] Stagmann MB, Gomez DD. A brief introduction to statistical shape analysis. Informatics and Mathematical Modeling, Technical University of Denmark, DTU. 2002;p. 15.

[18] Cootes T, Hill A, Taylor C, Haslam J. Use of active shape models for locating structures in medical images. Image and Vision Computing. 1994;12(6):355 – 365. <ce:title>Information processing in medical imaging</ce:title>.

[19] Skocaj D, Leonardis A, Bischof H. Weighted and robust learning of subspace representations. Pattern Recognition. 2007;40(5):1556 – 1569.

[20] Schmid J, Magnenat-Thalmann N. MRI Bone Segmentation Using Deformable Models and Shape Priors. In: Metaxas D, Axel L, Fichtinger G, SzÃ©kely G, editors. Medical Image Computing and Computer-Assisted Intervention - MICCAI 2008. vol. 5241 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2008. p. 119–126.

[21] Haber E, Modersitzki J. Intensity Gradient Based Registration and Fusion of Multi-modal Images. In: Larsen R, Nielsen M, Sporring J, editors. Medical Image Computing and Computer-Assisted Intervention - MICCAI 2006. vol. 4191 of Lecture Notes in Computer Science. Springer Berlin Heidelberg; 2006. p. 726–733.

[22] Coupe P, Yger P, Prima S, Hellier P, Kervrann C, Barillot C. An Optimized Blockwise Nonlocal Means Denoising Filter for 3-D Magnetic Resonance Images. Medical Imaging, IEEE Transactions on. 2008;27(4):425–441.

[23] Storn R, Price K. Differential Evolution - A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization. 1997;11(4):341–359.

[24] Heimann T, Van Ginneken B, Styner MA, Arzhaeva Y, Aurich V, Bauer C, et al. Comparison and Evaluation of Methods for Liver Segmentation From CT Datasets. Medical Imaging, IEEE Transactions on. 2009;28(8):1251–1265.

[25] Gao Q, Chang PL, Rueckert D, Ali SM, Cohen D, Pratt P, et al. Modeling of the bony pelvis from {MRI} using a multi-atlas AE-SDM for registration and tracking in image-guided robotic prostatectomy. Computerized Medical Imaging and Graphics. 2013;37(2):183 – 194. <ce:title>Special Issue on Mixed Reality Guidance of Therapy - Towards Clinical Implementation</ce:title>.

Seminar on Fused DTI/HARDI Visualization

Sathvik Parekodi

1 Abstract

Traditionally for many years, the diffusion of water molecules in the brain tissues are studied using technology like Diffusion Tensor Imaging (DTI). This technique is good at visualizing water diffusion but omits certain important and finer details, which is not accurate enough for certain applications. Recently, a new approach like High angular resolution diffusion-weighted imaging (HARDI) has been proposed. HARDI makes use of current GPU power to render the visualization faster and to give detailed information than previous methods. But, this method has complexity of its own, which does not make it so intuitive for many to use. In a recent effort to improve the performance of the rendering and to provide better features, both DTI and HARDI are fused to form a new framework. This new approach opens up many new opportunities. In this seminar report, fused DTI/HARDI is discussed in detail to understand its benefits, scope for improvements in future, user feedback on the new framework and possible improvements.

2 Introduction

Brownian motion describes the random motion of molecules in liquid or gas. Due to the changes in thermal energy at the molecular level, molecules start moving from one point to other. By tracing the movement of these water molecules in the brain, it is possible to reveal the underlying structure of brain tissue.

When water molecules encounter a cell, it moves along the cell and cannot go through it easily. Magnetic resonance diffusion imaging measures this diffusion of water molecules to show the underlying path and direction of flow. One of the main components of the central nervous system is white matter consisting of glial cells and myelinated axons which transmits signal between different parts of the brain. An application of DW-MRI is to study the white matter in human brain. Voxel represents a single value of a grid in a three dimensional space. In a single voxel, if the fibers in the brain are organized in one direction, then it can be represented by Gaussian probability density function. In the paper by Basser et al on Diffusion Tensor Imaging [1] makes an assumption that all voxels will have a Gaussian probability distribution, but this has been proved false when underlying fiber structure gets complex. Structures get complicated at interfering regions of two or more fibers. DTI techniques are inadequate in representing them in a precise manner.

Another idea of visualizing the diffusion of water molecules in the brain tissue was introduced by Tuch [2] called High Angular Resolution Diffusion Imaging (HARDI) . This method makes less assumption about the diffusion process in the tissue and rather focuses on the characteristics of the tissue like orientation of multiple fiber direction. But, this technique has disadvantage because of complex modeling, processing time and enormous amount of data provided to the user after processing. This becomes an overhead while visualizing the regions of single fiber coherence where DTI technique would have performed much better.

Fused DTI/HARDI framework uses the key features from both of the techniques and improves on it. It classifies the data into three categories. Isotropic or the areas from the gray matter, anisotropic-Gaussian are the areas with significant single fiber coherence, and non-Gaussian are the areas where the white matter bundles have more complex structures. DTI technique, which does fast rendering, is applied at single coherent fibers and HARDI techniques are used at the regions having complex structures and diffusion pattern.

In this seminar paper, individual techniques like DTI and HARDI is introduced followed by fusion of these two techniques and it's advantages over the individual techniques, visualization in fused DTI/HARDI techniques, color-coding and benefits of the new approach.

3 Data representation

3.1 Diffusion Tensor Imaging

With the introduction of Diffusion Tensor Imaging, the analysis and the imaging of the water diffusion was improved significantly compared to previous approaches like nuclear MR (NMR). DTI assumes that, in anisotropic materials, anisotropic Gaussian describes the diffusion function.

In biological applications, mainly protons of water molecules are measured and behavior of these water protons are sometimes called as spins for convenience. These spins are used for deriving diffusion equation further.

A simple diffusion tensor framework can describe only certain type of diffusion phenomenon, for example Gaussian diffusion. This framework cannot represent wide variety of things which is observed during in vivo like restriction, heterogeneity, anomalous diffusion, and finite boundary permeability. Hence a new descriptive framework was formulated and diffusion propagator was introduced.

Diffusion propagator describes the probability of a spin traveling from position $r^{'}$ to r in the diffusion time τ. Spin probability density flux $\rho(r)$ defines the flux in a unit area at position r and time τ.

The diffusion equation gives the relationship between observed diffusion propagator and underlying micro-structure. Diffusion absorption equation is derived using Fick's first law and the continuity theorem. [2]

$$[\frac{\partial}{\partial\tau} - \nabla(D(r)\nabla) + \rho(r)]P(r, r^{'}, \tau) = \delta(r - r^{'})\delta(\tau) \tag{9.1}$$

The Gaussian propagator PG is used to solve the diffusion equation.

$$P_G(R, \tau) = (|D|(4\pi\tau)^3)^{-1/2} e^{(-R^T D^{-1} R/(4\tau))} \tag{9.2}$$

whose parameter are the diffusion tensor D, and the relative displacement $R = r - r^{'}$. Parameter of anisotropic Gaussian is diffusion tensor D, and the eigensystem of it reveals the orientational structure of the underlying material.

Magnetic resonance diffusion tensor imaging can be used to measure tensor, under the assumption of Gaussian diffusion. Measured diffusion tensor has an Eigen structure, which informs about the orientation of the diffusion compartments with in a voxel. Major Eigen vector is parallel to the mean fiber orientation and minor eigenvector is normal to the mean plane of fiber dispersion. An Eigen system is represented by,

$$D = R\Lambda R^T \tag{9.3}$$

where R=(e1,e2,e3) is a coloumn matrix of the orthonormal diffusion tensor eigen vectors e_v, and $\Lambda = (\lambda_1, \lambda_2, \lambda_3)$ is the diagonal matrix of the diffusion tensor eigenvalues. e1 is the major, e2 is medium and e3 is minor eigen vectors. This is illustrated diagrammatically in Figure 9.1 .

The Eigen system for the diffusion tensor can be thought in terms of diffusion isoprobability surface. This is a surface where a spin at the origin will diffuse with equal probability. Isoprobability surface for the Gaussian diffusion is represented by an ellipsoid. The axes of the ellipsoid are oriented in the direction of eigenvectors. Axes have lengths proportional to the diffusion distance along the corresponding eigenvectors. An isoprobability surface ellipsoid is a good representation of the Eigen system but sometimes it is desirable to display the tensor in other geometric forms such as cuboid to stress on local fiber direction. Even polygonal cylinders have been used to represent major and minor eigenvectors, or a simple line to represent only major Eigen vector.

3.2 High Angular Resolution Diffusion Imaging (HARDI)

In a homogenous and isotropic medium, according to Fick's second law of diffusion, diffusion propagator $P(r)$ is given by

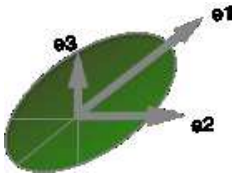


Figure 9.1: Ellipsoid with Eigen vectors determining the shape of it.

$$P(r) \propto e^{-r^2/4Dt} \tag{9.4}$$

with scalar diffusion constant D, but at the regions of complex fibers one cannot rely on the Gaussian diffusion. In DW-MRI diffusion propagator is not measured directly, instead signal S(q) is measured from the nuclear spins with diffusion sensitizing magnetic gradient.

All the functions in diffusion weighted MRI modeling techniques are rendered on a spherical surface. Spherical Harmonics(SH) provides a convenient basis for representing functions on a sphere. SH is a good linear basis for representing the angular portion of the reconstructed diffusion function. These functions are called Spherical distribution functions (SDFs) of HARDI modeling technique. It characterizes the local intra-voxel fiber structure, and can be represented by Laplace series, which is a linear combination of spherical harmonics.

$$\psi(\theta,\phi) = \sum_{l=1}^{\infty} \sum_{m=-l}^{l} a_l^m Y_l^m(\theta,\phi) \tag{9.5}$$

where, a_l^m are SH coefficients resolved from SH transformations. Y_l^m represents the Spherical harmonics.

3.2.1 Q-Ball Imaging(QBI) In Q-Space Imaging, a sampling of range of q-values is obtained according to diffusion wave vector given by equation 6, where a q-value is proportional to diffusion encoding gradient.

$$q = \frac{\gamma}{2\pi} \int_0^{\delta} G(t)\, dt \tag{9.6}$$

where, δ is the diffusion gradient duration, $G(t)$ the diffusion gradient vector, γ is gyromagnetic ratio for the particular interested nucleus. [3]

3.2.2 Diffusion Spectrum Imaging In DSI, multiple q-values are acquired along multiple diffusion encoding orientation, which leads to the estimation of 3D spin displacement probability density function(pdf). In this method no underlying model is assumed and with sufficient q sampling, it can detect any diffusion pattern, but this method has extreme sampling requirement. Even a 8x8x8 grid sampling requires 512 measurement as indicated by McNab in paper [4].

Two major disadvantages of Q-space imaging were indicated by Tuch in the paper [3]. It is shown that this technique involves sampling on three-dimensional Cartesian lattice, which is not efficient. And, this method requires large pulsed field gradients according to the Nyquist condition for diffusion in nerve tissue.

To solve the problem of high sampling requirement, a new approach was proposed based on sampling on spherical shell in diffusion wave vector space. Spherical Shell approach is mainly used in HARDI. This approach is efficient because sample is only obtained on spherical shell compared to three-dimensional Cartesian volume.

For HARDI data, a model free reconstruction approach is used. This approach is based on Funk-Radon transform(FRT), called as Funk Transform. Funk transform is an integral transform defined by integrating a function on great circles of the sphere. This method has several advantages compared to previous techniques of reconstruction. This is much simpler in computation, linear in signal and model independent. To represent the relevant directional information of the diffusion function, Diffusion orientation distribution function (ODF) is used. The ODF defines the probability of a spin displacing into

a differential solid angle in the fiber direction u. ODF $\psi(u)$ is defined as the radial projection of the diffusion function and is given by equation (9.7).

$$\psi(u) = \frac{1}{Z}\int_0^\infty P(ru)dr \tag{9.7}$$

where Z is a dimensionless normalization constant. Material science makes use of ODF to great extent to describe oriental structure of polymers, liquid crystals etc. Z normalizes the ODF to unit mass.

QBI reconstruction is based on FRT. FRT is the extension to planar Random Transform. Given unit direction vector w and a function $f(w)$ on the sphere, FRT is defined as sum over the set of points perpendicular to w. FRT for the direction u is given by

$$F[f(w)](u) = \int f(w)\delta(w^T u)dw \tag{9.8}$$

where δ is a Dirac delta function. FRT is a transformation from one sphere to another, but the requirement is to transform three dimensional cartesian space to sphere, hence Tuch in the paper [2] extended this version of FRT as follows. Given a three dimensional function $f(x)$ with x being a three dimensional vector, FRT is evaluated at particular radius $r^{'}$.

$$F[f(x)](u, r^{'}) = \int f(x)\,\delta(x^T u)\,\delta(|x| - r^{'})dx \tag{9.9}$$

This is denoted by G_r It is shown that FRT of the diffusion gives an approximation of the ODF [2] i.e.

$$\psi(u) = \frac{1}{Z}F_q[E(q)] \tag{9.10}$$

This leads to estimation of diffusion probability in a particular direction i.e. by summing the diffusion signal along the equator around that direction as shown in Figure 9.2

Figure 9.2: q-space sampling scheme is indicated by the sphere. The grey arrow gives the direction of interest. The light grey circle indicates the equator around the direction of interest. [3]

4 Visualization techniques

This section briefly describes the visualization methods used for DTI and HARDI techniques. It introduces topics like Fractional Anisotropy, Volumetric rendering, HARDI Glyphs and Tractography, effect of noise on few of these techniques.

4.1 DTI

DTI data i.e. a 3D diffusion tensor is represented by positive symmetric matrix at each voxel in the three dimensional grid.

$$D = \begin{bmatrix} D_{xx} & D_{xy} & D_{xz} \\ D_{xy} & D_{yy} & D_{yz} \\ D_{xz} & D_{yz} & D_{zz} \end{bmatrix} \tag{9.11}$$

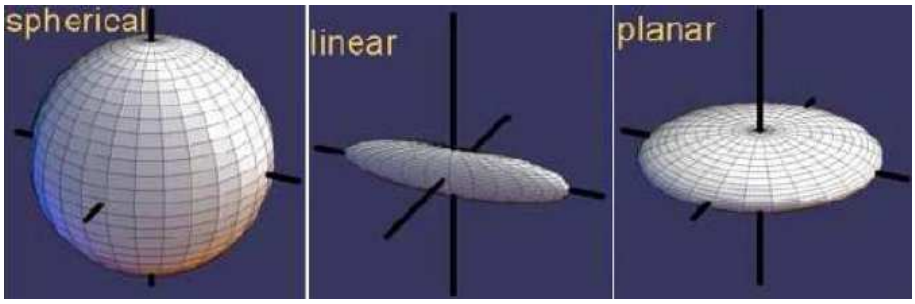


Figure 9.3: Diffusion Tensor Glyphs.

After Diagonalizing D , we get three positive eigen values λ_1, λ_2, λ_3 in decreasing order. Depending on the eigenvalue, shapes of the diffusion tensor glyphs differs.

When $\lambda_1 = \lambda_2 = \lambda_3$, it is represented by a spherical shape. When $\lambda_2 \approx \lambda_1 \approx 0$, it is represented by a linear shape. When $\lambda_1 \approx \lambda_2, \lambda_3 \approx 0$, it is represented by a planar shape as show in the Figure 9.3

To visualize globally, and to get better understanding of the fiber orientation details, methods like volumetric rendering and fiber tracking can be used.

4.1.1 Volumeteric Rendering A volumetric data is represented by a set V of samples(x,y,z,v). These are also called as voxels, representing the value v at a 3D location(x,y,z). In binary data, v can be either zero or any integer. A value of zero indicates the background and integer value indicates the presence of the object. It can even be mulit valued representing properties like colours, density, heat and pressure. It can even represent a vector for example, velocity at each location or color triples(RGB). If volume data is time varying, V becomes a 4D set of samples (x,y,z,t,v) where v represents the value at (x,yz,t).

Volume visualization is the method of obtaining meaningful data from the volumetric data. This technique requires volume data representation, modeling, manipulation and rendering. Volumetric data as shown above contains 3D entities, which is too huge and time intensive using primitive geometry like triangles for rendering. They are obtained by sampling, simulating and modeling techniques.

A sequence of 2D slices obtained from the Magnetic Resonance Imaging (MRI) are used for volume model and used for diagnostic purpose, planning for treatment or surgery. Many techniques were proposed to visualize these volumetric data, since rendering primitive geometries were well known. But, many initial approaches made an approximation of the surfaces using primitive geometry. When surface rendering techniques are used to represent volumetric data, generally a dimension of information is lost i.e surface rendering fails to render regions hidden behind other overlapping regions. To overcome this disadvantage, many new rendering techniques were suggested which can represent the three-dimensional data into a single two-dimensional image. Volume rendering conveys more information than the surface rendering but algorithm becomes complex and rendering time is increased.

Volumetric rendering can be achieved using different methods, some of the popular methods are object order, image order and domain based techniques. Object order technique makes use of forward mapping scheme where volume data is mapped to the image plane. In image ordering technique a backward mapping scheme is used where rays are projected from the pixels from the image plane through the volumetric data to calculate the final value of the pixel. In domain based technique, volume data is transformed to other domain such as frequency, wavelets and then projected from that domain.

There are different types of rendering modes like X-ray rendering, Maximum Intensity Projection, Iso- surface rendering and full volume rendering. The effects of these different methods are shown in Figure 9.4.

Above methods have some similar operations. In all the above methods, Image ray is projected from the image pixels, and these rays intersect grids at discrete points in their path. They all obtain samples through interpolation. But, they differ in a way the samples are taken along the ray. In X-ray method, interpolated values are summed up, In MIP only the interpolated sample with highest value is written to pixel. In full volume rendering interpolated samples go through further processing to simulate light properties within a volumetric medium.

4.1.2 Fractional anisotropy is a scalar value, whose value ranges from zero to one. These values describe the degree of anisotropy in a diffusion process. Zero indicates isotropic diffusion and value of one

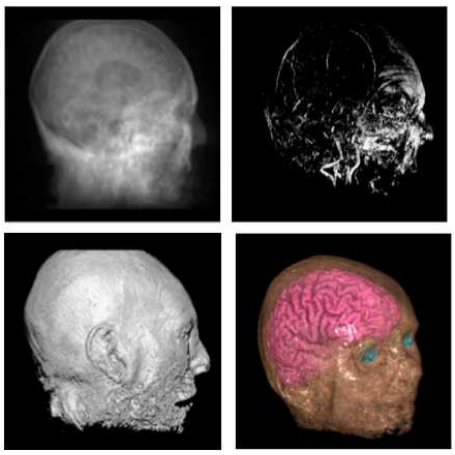


Figure 9.4: Volume rendering using different techniques From top to right, X-ray; MIP; Iso-surface; Translucent. [5]

indicates that diffusion happened only in one direction. By computing scalar anisotropy indices from the diffusion tensors, volume rendering can be used to visualize it. FA is given by,

$$FA = \frac{\sqrt{3((\lambda_1 - E[\lambda])^2 + (\lambda_2 - E[\lambda])^2 + (\lambda_3 - E[\lambda])^2)}}{\sqrt{2(\lambda_1^2 + \lambda_2^2 + \lambda_3^2)}} \tag{9.12}$$

$E(\lambda) = (\lambda_1 + \lambda_2 + \lambda_3)/3$, where λ_1, λ_2, λ_3 are the eigen vectors. λ_1 is the eigen value corresponding to major eigen vector whose magnitude is higher than other eigen vectors and parallel to the direction of axon.

DTI represents two important information about water diffusion, amount of diffusion anisotropy and orientation. It is assumed that the principal axis of diffusion tensor aligns with predominant finer orientation in the MRI voxel. A 2D and 3D vector fields can be obtained, which represents the fiber orientation at each voxel. Volume rendering along with FA value can be used to assign colors and opacity while rendering.

Transfer functions are the functions, which assigns colours and opacity according to the measured tensor fields. Thus, combining volume rendering technique and DTI involves finding and mapping parameters of transfer functions. Gradient of FA can be calculated using chain rule as a normal for surface shading. Figure 9.5 shows the 3D structure with volume rendered isosurfaces of fractional anisotropy. These images were computed using opacity step function. Opacity is 0.0 or 1.0 depending on whether FA is below or above the certain threshold.

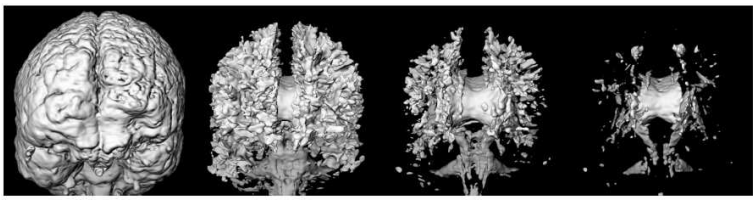


Figure 9.5: Volume-rendered isosurfaces at differnt range revealing 3D structure of underlying white matter.

4.1.3 Measurements in DTI Voxel dimensions are measured in order of 1-5mm and DTI measures average diffusion properties inside this voxel. Voxel size is generally enough to distinguish white and gray matter. White matter tracts contain several axons within them, which might contribute to high anisotropy. There is enough anatomical knowledge about human white matter structure but DTI with the size of voxel and human brain, can even reveal microscopic 3D structure and finer details about white matter tracts.

4.2 Tractography

It is a modeling technique, which is used to represent neural tracts using data that are collected by methods like DTI. Streamlines represent the abstract path of fibers in the brain but they may not be exact reproduction of the anatomical fibers. Streamlines are represented by tracking algorithms and represent the most probable pathways in the tensor field. In this technique, a seed voxel is chosen and tracking is performed in both forward and backward direction. FA measure is used as threshold for seed voxel and also for stopping of the propagation of streamline. Voxel with high FA is used as seed voxels and if FA becomes lesser than the threshold, tracking is stopped.

The direction at each step is calculated by resolving underlying Eigen system of the tensor and identifying the principle eigen vector. This is given by Euler integration [6]. But for numerical accuracy, a higher order integration scheme is applied i.e. Runge-Kutta of order four, which interpolates and calculates principal eigenvector repeatedly till the direction of streamline propagation is determined.

There are two main categories of tacking. First one is based on line propagation algorithm that uses local tensor information. Second type is based on minimizing global energy information to find paths, which are energetically more favorable between two pixels. In the next section line propagation approach is described in brief.

4.2.1 Line propagation approaches One can create a 3D trajectory from 3D vector field by propagating a line from a seed point in the direction of vector orientation. Pixels are discrete entities and by connecting these discrete entities, some of the vector information at each voxel is lost while propagating. [7]

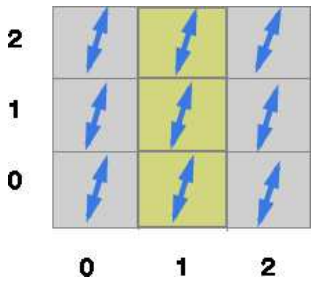


Figure 9.6: Showing the discrete field and propagation decision based on it.

In Figure 9.6 , we can see that all the pixels have vector orientation is the same direction. It is ambiguous to connect pixels and to form a tract because of similar orientation field. In the above example, seed point (1,1) is chosen and to propagate to next voxel in discrete points it is not clear if one has to choose (1,2) or (2,2). The above approach leaves out many significant information and cannot represent the tract in even simple case because of the ambiguity in choosing the voxels while propagating.

In another approach, discrete voxel information is converted to continuous tracking line by linearly propagating a line in a continuous number field. The conversion is shown schematically in Figure 9.7 , where a seed point (1.50,1.50) is chosen and line propagates in the direction of vector orientation of the pixel with discrete coordinate (1,1), In this continuous medium we can see that tracts reconstructed are much precise.

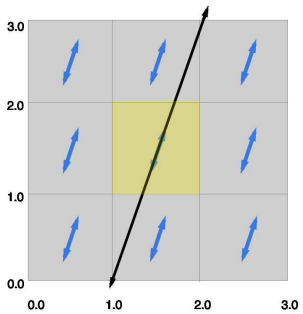


Figure 9.7: Showing the continuous field and propagation decision based on it.

There can be more improvements to the above methods to have smooth paths, which will represent the curvature of the reconstructed line if the line is steep with respect to imaging resolution. As shown in Figure 9.8 , in this method a step size is chosen. When it reaches a new point a distance-averaged vector orientation is calculated. In a minimalistic way, two pixels closest to the new coordinate are determined and then average is calculated to draw a smooth line between pixels.

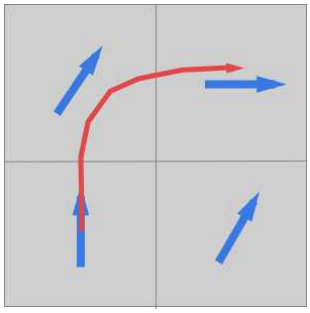


Figure 9.8: Effect of interpolation.

4.2.2 Effect of Noise on tracking Usually 3D vector fields computed from DTI data contains noise, which effects the vector direction and may deviate from the real fiber orientation. These errors are added at each step while applying line propagation methods. Effects of these errors as a function of Signal to Noise ratio are evaluated for different models. The results show that there is a dependence on the anisotropy, resolution, shape of the trajectory and the particular interpolation method used. These effects can be reduced using smoothing or interpolation techniques by averaging the vector or tensor information among neighboring pixels, which results in the reduction of effective resolution. Many other approaches have been proposed to minimize the effect of noise like approximation of tensor field based on B-spline fitting or regularization of the tensor field based on 'low curvature hypothesis'. Hence tractography is used with caution.

4.3 Limitations of DTI

The main limitation of the DTI is due to the assumption of Gaussian diffusion at every voxel. Hence, voxels with multiple fiber crossings and high curvature breaks the diffusion tensor model. It is shown that between 1/3 and 2/3 white matter with FA greater than 0.1 have a multiple fiber crossing configurations as shown in the Figure 9.9. Fiber crossings and region of high curvature is represented by greenish ellipsoids which are planar in shape. DTI ellipsoids having planar shapes stops fiber tracking as the planar shapes doesnot reveal the exact orientation of the fiber. Therefore, higher order models and new reconstruction approaches are required to describe the non-Gaussian profiles. HARDI visualization is one such technique.

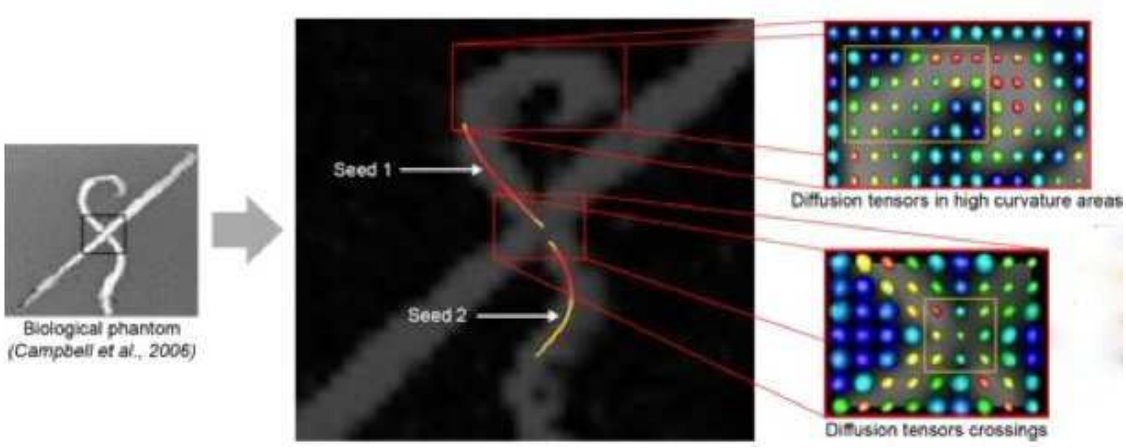


Figure 9.9: Limitations of DTI visualization in fiber crossings and at regions of high curvature.

4.4 HARDI Glyphs

Sampling of the diffusion signal is directly done on the spherical shell as shown in Figure 9.10 a. Reconstruction of ODF using spherical shell has many advantages. First, Reconstruction using spherical shell method is immune to Cartesian reconstruction bias. Second, with spherical shell approach there is a natural framework to calculate angular resolution.

Tessellation is a process of tiling the surface with certain geometrical shape like triangles or icosahedron. Figure 9.10 a, is a sampling of diffusion signal on a fivefold tessellated icosahedron. White, Yellow and red color dots and their size indicates the signal intensity. Figure 9.10 b, shows the regridding of the diffusion signal onto set of equators around vertices of dodecahedron. Figure 9.10 c, represents the ODF, which is calculated using Funk transform. Figure 9.10 d shows the color coded spherical polar plot rendering of ODF. In Figure 9.10 e, min-max normalization is applied to clearly visualize the orientational information.

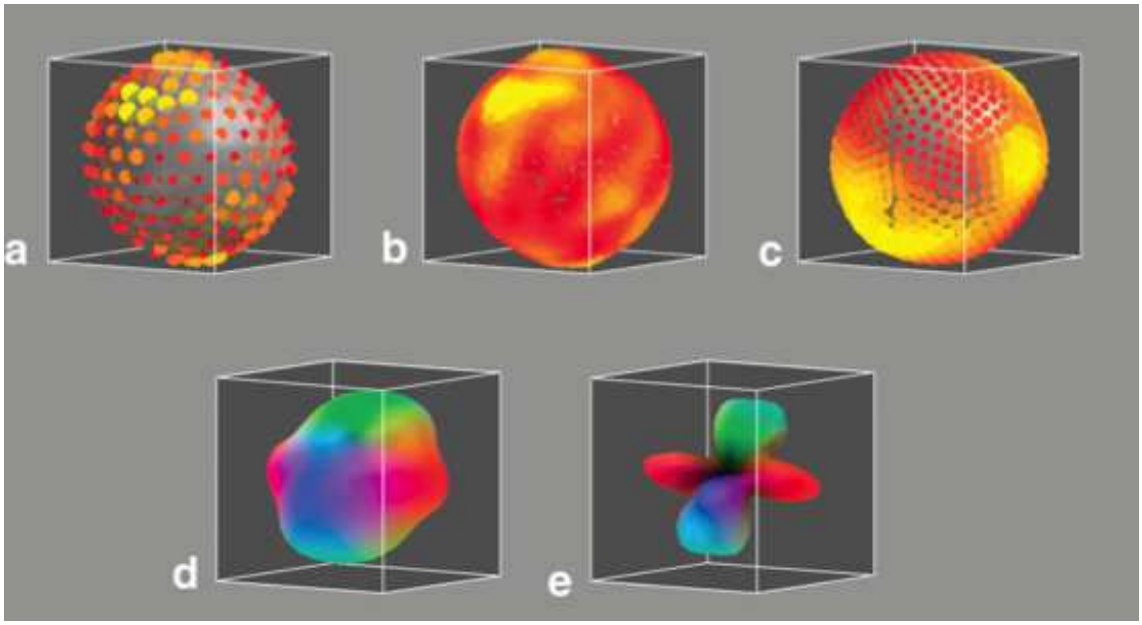


Figure 9.10: Q-Ball imaging. [3]

For a very smooth surface, the 16th order icosahedral tessellation is used, which produces 2256 points on a sphere. With so many points being generated, visualization of full brain slice for real time interaction is quite impossible to use. To gain speed and to improve performance a coarser mesh can be used, but with the disadvantage of loosing finer details and often angular maxima are displaced. Change of the shape with different mesh can be seen in Figure 9.12 . [8]

HARDI Glyphs can represent both single fiber orientation and multiple fiber orientation as shown in Figure 9.11.

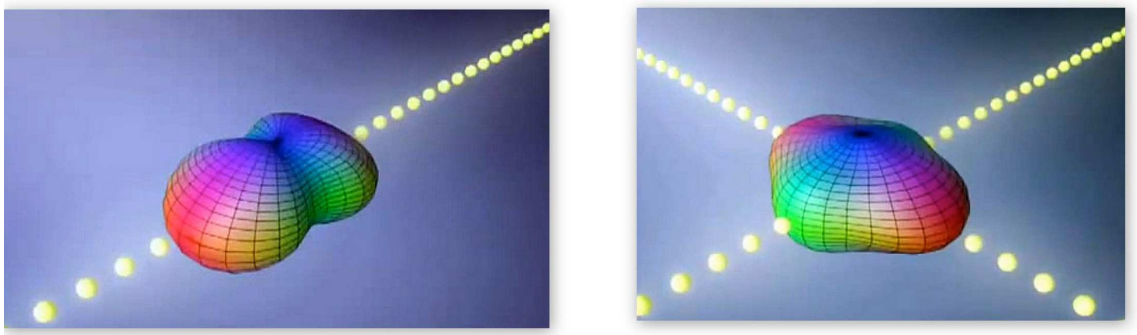


Figure 9.11: Left: Single fiber orientation, Right: Multiple fiber orientation with fiber crossings.

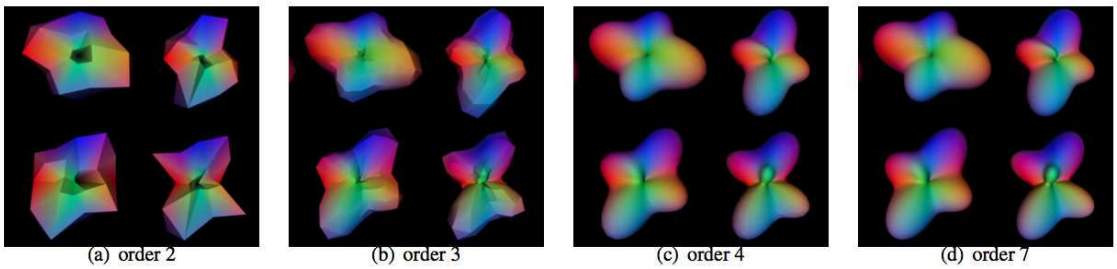


Figure 9.12: Visualizing HARDI glyphs with increasing order or icosahedral tessellation which smooths the surface.

5 Visualization combining both DTI and HARDI

Both the visualization techniques DTI and HARDI has its own advantages and disadvantages. DTI is fast to compute but makes an assumption of Gaussian diffusion in every voxel. HARDI excels in recognizing the local diffusion properties and structures with the help of higher order models but it increases the complexity and displays too much of data without revealing global context. Fused DTI/HARDI makes use of both the techniques and merges them. This technique also shows fiber tracts and GPU accelerated DT ellipsoids with HARDI SDFs.

Fused DTI/HARDI classifies the data into different categories and suitable method is applied for each category. In the areas categorized as Gaussian profile, tensor glyphs and fiber tracking is applied and in the regions where complex structure is detected, HARDI method is applied and local HARDI information is retrieved at that point. A snapshot of this merged visualization is shown in Figure 9.13 . This method has an advantage of global context by displaying DTI tensor glyphs and it also represents better local diffusion information due to HARDI method.

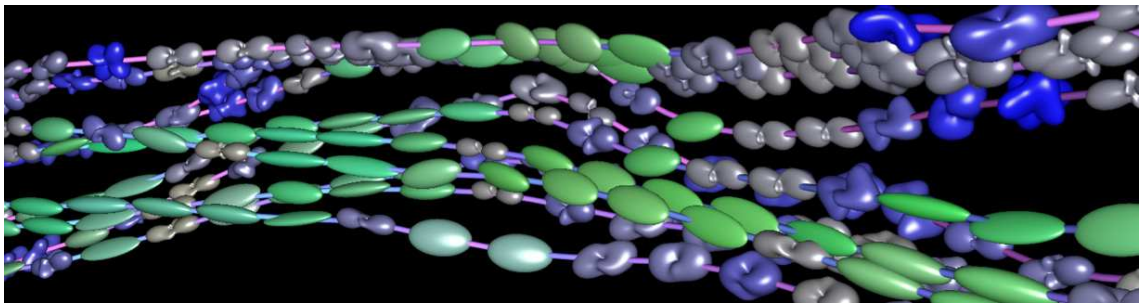


Figure 9.13: Fiber tracts showing DTI (green ellipsoids) and HARDI (QBalls, blue and grey with complex shapes) glyphs representing the local diffusion. [9]

5.1 GPU based ray casting for Tensor Glyphs

As the name suggests, a ray casting method is used to render tensor Glyphs. Glyphs therefore are embedded into bounding boxes with axes aligned to the volume image. Eigen vectors are computed using Eigen values from the corresponding diffusion tensors. These Eigen vectors can be represented as e1, e2, and e3 and are used as vertex attributes in GPU. The intersection points of the ellipsoidal glyphs with view ray of each pixel is determined at GPU level using ray casting. But the axes of ellipsoidal glyphs and view ray for ray casting are different. Therefore, the axes of the view ray are rotated to match the axes of the ellipsoidal glyphs. View ray is rotated with matrix as given by equation 13.

$$R = (e1, e2, e3)^T \tag{9.13}$$

Axes of ellipsoidal glyphs are always aligned with the Eigen vectors of the diffusion tensor at that point. After rotating the axes of view ray, a transformed view ray is obtained.

$$V' = V + \mu v \tag{9.14}$$

Ellipsoid equation is represented by $\frac{x^2}{r_x^2} + \frac{y^2}{r_y^2} + \frac{z^2}{r_z^2} = 1$, where $(r_x, r_y, r_z) = (\lambda_1, \lambda_2, \lambda_3)$ are the radii of the ellipsoid. By replacing $V^{'}$ in the $(x, y, z)^T$ of the above equation , this is reduced to quadratic equation. We can solve for the roots of this quadratic equation using the formula

$$\mu = \frac{-b \pm \sqrt{d}}{2a} \tag{9.15}$$

where $d = b^2 - 4ac$, $a = \sum_i 2r_i v_i^2$, $b = \sum_i 2r_i(V_i - Q_i)v_i$ *and* $c = \sum_i r_i(V_i - Q_i)^2$ *for* $i \in x, y, z$. Discriminant of the above method determines if intersection is possible, i.e. positive value indicates intersection, negative value indicates otherwise. Coloring of each individual ellipsoid is determined by x,y,z component of main Eigen vector e1 to RGB values. Apart from this coloring scheme, other coloring scheme is also available.

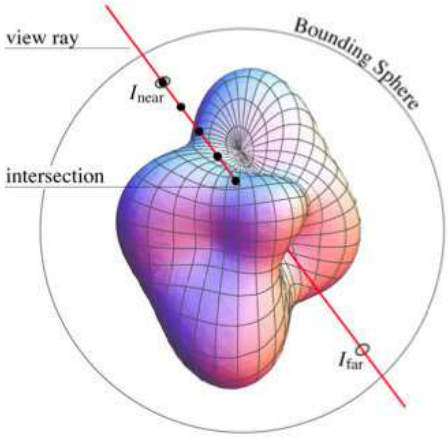


Figure 9.14: Ray casting algorithm for SH for HARDI [8]

5.2 Ray casting of spherical harmonics Glyphs

Ray casting of spherical harmonics is similar to that of ray casting of tensor glyphs. But, due to the complex structure of spherical harmonics, finding intersection point on the complex surface is difficult. The size of the bounding box cannot be readily assumed, as it cannot be calculated analytically. A maximum upper bound radius is calculated in this case or pre-computed estimated value is used. HARDI signal $\psi(\theta, \phi)$ is expanded by real valued co-efficient $\widetilde{a}_l^m$. A maximum upper bound radius is found using the following equation.

$$|\psi(\theta, \phi)| \leq \sqrt{\frac{1}{2}l_{max} + 1}\sqrt{\sum_{l=0}^{l_{max}}(\frac{2l+1}{4\pi}\sum_{m=-l}^{l}(\widetilde{a}_l^m)^2)} = r_{max} \tag{9.16}$$

After deciding the radius for the bounding box using r_{max}, the intersection of a view ray with the SH glyphs is calculated. To find intersection between SH glyphs and view ray, first a point on the view ray is calculated such that point lies inside the glyph. This is done using linear search along the view ray. This is shown in Figure 9.14 schematically. Precise location on the sphere is then calculated by binary search. The difficult part in this process is to find if the point is inside or outside the glyph.

Min-Max normalization is a process of transforming any measured unit to a value between zero and one where the lowest value is set to zero and highest value is set to 1 and all other values are transformed between zero and one depending on the range of values.

To visualize maxima i.e peaks of SDFs and for the structures, which are isotropic in nature like Q-ball, a technique called min-max normalization can be used. Using min-max normalization an alternative

method is proposed, where for every imaging voxel minimum and maximum are pre-calculated from a discrete surface mesh and passed to the GPU as vertex attributes.

Sharpened fiber ODF is implemented in this framework where user has the option to choose the areas of significant single fiber coherence as de-convolution kernel and user can see the effects of it in real time [9]. This shows the advantage of using GPU accelerated visualization. With the benefit of interactivity and ability to adjust the parameters in real time to see immediate effect, I believe that this method holds a great potential for the researchers in medical image processing, compared to previous methods.

In DW-MRI, directions are encoded using colour mapping, with RGB colour applied for SDF glyphs where red colour indicates mediolateral, green colour anteroposterior, and blue colour superior-inferior. But, this scheme stresses on axes aligned orientation and peaks are not so clear using this scheme. A new colouring scheme was proposed which stresses on maxima. An example of it shown in Figure 9.15 . Peaks are coloured with yellow and rest are interpolated with red. User is even allowed to change the threshold radius of yellow/red colouring. Furthermore, whole surface of the glyph can be coded using HARDI anisotropy measures. This gives information about the amount and types of anisotropy at each glyph. This is particularly useful for isotropic structures like Q-balls.

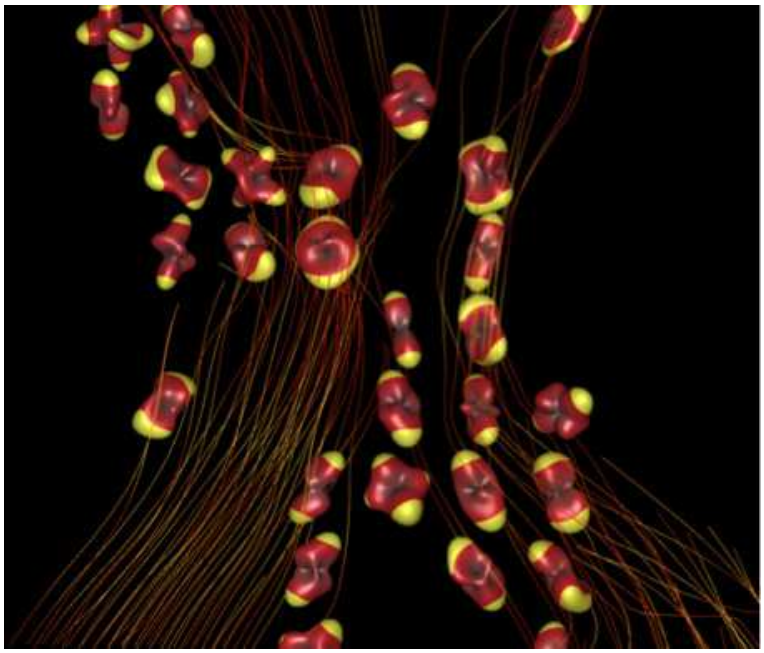


Figure 9.15: The color coding scheme which enhances the maxima.

5.3 Classification

HARDI anisotropy measure classifies each voxel into three compartments isotropic, non-Gaussian and anisotropic-Gaussian [9], but this method has a disadvantage because user has to experiment with threshold values which gives good balance between the compartments by a trial and error method. This method is highly dependent on the angular configuration of the non-Gaussian profile and acquisition parameter. These parameters are dynamic, making this method unsuitable for the purpose of classification. A new approach to classification is proposed in the paper [9] called semi automatic threshold tuning.

In the semi automatic threshold tuning algorithm, user chooses a HARDI anisotropy measure M for classification. In this classification method, regions where crossing is expected are called positive region (Pos) and linear areas are called negative (Neg) regions. The proposed algorithm then calculates the distributions M(Pos) and M(Neg) in the voxels of Pos and Neg regions. Minimum value of M(Pos) is set as lower threshold value, and upper threshold is set to maximum value of M(Pos), if M(Pos) and M(neg) have no overlaps.

If M(Pos) and M(Neg) overlaps, then average of the medians of M(Pos) and M(Neg) is set as upper threshold. User is allowed to change manually the threshold and fine-tune the selection regions. This is proved useful in several applications.

When fibers are used as seeding points for glyph visualization it helps in detecting the uncertainties and unreliable outcome from fiber tracking algorithm. A traditional method to visualize diffusion using geometric glyphs is shown in Figure 9.16.

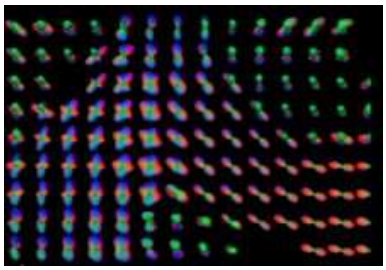


Figure 9.16: Traditional Method

In Figure 9.17 , tensors and HARDI glyphs are combined. Ellipsoid gives information about the direction much better than the HARDI linear glyphs. HARDI glyphs are color coded by the types of anisotropy, where high anisotropy is coded with yellow and low anisotropy is coded with dark blue. Depending on the threshold tuning, linear tensor glyphs may change.

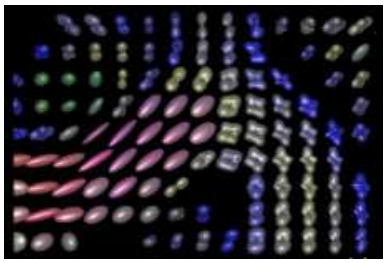


Figure 9.17: Fused Tensor

In Figure 9.17 , glyphs that are coded with dark blue colour appears as crossings but they have very low anisotropy and color coding indicates that they belong to the grey matter or the ventricles. These glyphs are masked using classification and avoids any redundant and misleading information.

In Figure 9.18, it is further simplified by showing only fibers where tensor information is available and the non-Gausssian areas are represented by HARDI glyphs.

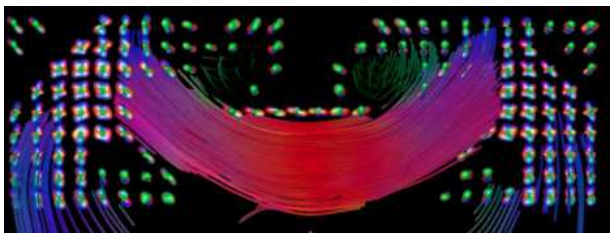


Figure 9.18: Glyphs with fiber tracking

5.4 Fused Visualization pipeline

Figure 9.19 illustrates the pipeline of fused DTI/HARDI visualization. Red circles represent user interaction, whereas green circles represent seed points, which are calculated after fiber tracking. Dashed lines represent optional actions. Blue rounded rectangles represent different input data and pink rectangles represent transformation on the data. In the beginning, tensors and SH coefficients have to be calculated representing different spherical density functions(SDF). Fiber tracking algorithm is applied where data is classified as anisotropic Gaussian and this can be rendered using different algorithms like line rendering or it can be represented as tensor glyphs. Non-Gaussian areas can be represented by SDFs using 4th order SH.

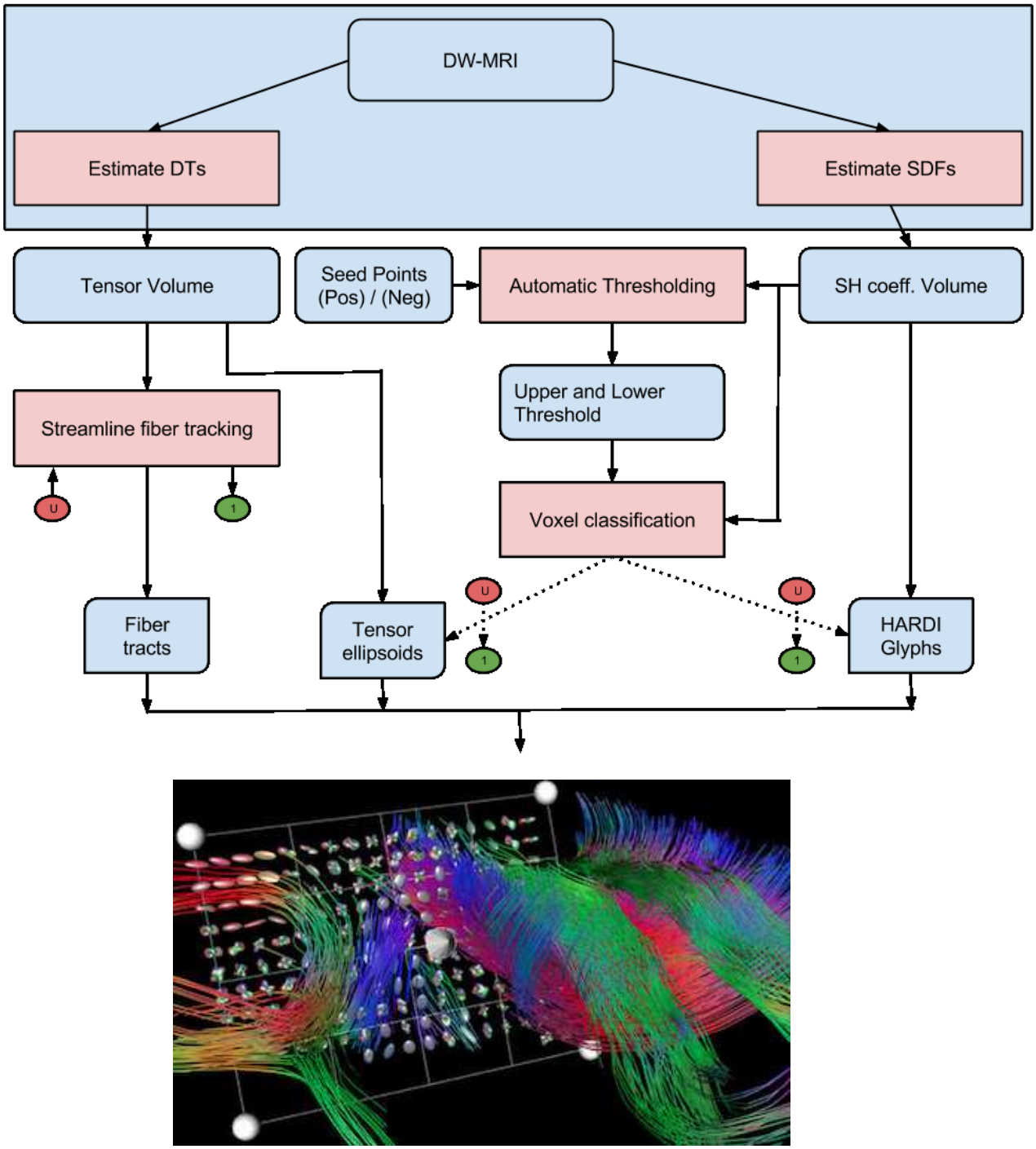


Figure 9.19: Fused DTI/HARDI pipeline.

DTI and HARDI visualization technique is fused by classifying voxel. After selecting the positive and negative data samples i.e. crossings and single fiber, thresholding algorithm is applied which divides the data into three categories namely isotropic, non-Gaussian and anisotropic-Gaussian. Later, tensor glyphs and HARDI glyphs are combined to represent data. Even streamline fibers and HARDI glyphs can be combined by taking into consideration that streamline fiber tracking depends only on diffusion tensor data. Glyph visualization can take input from fiber tracking, which can show HARDI and DTI data on the globally tracked DTI fibers as shown in Figure 9.18 . User interaction is allowed by the framework where user can change the parameters for classifying thresholds. This is mainly useful in the visualization of boundaries where the anisotropic-Gaussian profiles changes to non-Gaussian. This has been proved

important in investigating structures of white matter in the brain and to evaluate the amount of valid voxels in the given region. User can see the results in real time, where tensor glyphs diminishes and HARDI glyphs appears on the border when Gaussian profile changes to non-Gaussian.

5.5 Performance of fused HARDI / DTI

The strength of the fused HARDI/DTI method lies in the fact that user can change the thresholds and the rendering is done in real time, which gives a smooth experience without any delay in using the framework. This gives quick insight into the data. It is achieved by using ray-casting method, as mentioned previously. Traditional method of using geometric glyphs could render only 6 FPS given 10K DTI glyphs and 64 vertices per glyph. This rendering was done only after generation of vertices which took 8s as described in the paper [9]. This has significant delays between steps and not suitable for doing real time interaction. It is showed in the paper [9] that for the same amount of glyphs, by using ray-casting method they could achieve 35 FPS, which is quite a significant difference. It is quite clear on why this can be achieved, because there is no change in geometry when user changes the region of interest. In HARDI method, SH glyphs are used, which are more complex to render and geometric based method inherently suffers because of the complexity. Ray casting method performed much better in visualizing SH based glyphs. In the fused framework, user can change the region of interest in real time, and can change the seed points, which are used to place the glyphs by moving the interactive plane as shown in Figure 9.20 .

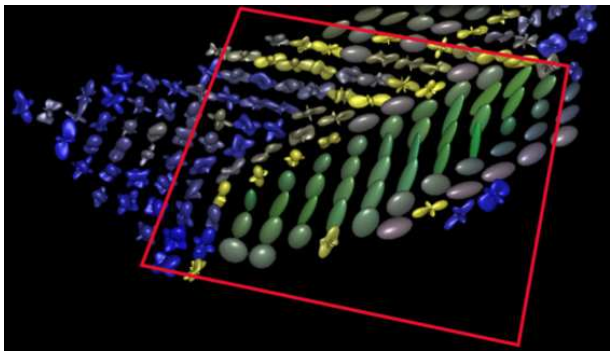


Figure 9.20: Fused Glyph visualization , red square indicates the plane which can be moved to new areas to visualize those data points.

A standard transforms like translation and rotation can be applied on this plane using the mouse. As soon as plane is rotated or translated new glyphs are immediately made visible without any delay. By using more complex SH glyph rendering algorithm only in the areas where it has benefits over the ellipsoid glyphs, rendering performance of the framework is greatly improved.

5.6 Evaluation

In the paper presented by [9] several user studies were done to evaluate the fused DTI/HARDI framework. Users of the framework were either researchers in the filed of visualization or neurosurgeons, who use various tools to analyze the DTI data in a better way. Participants were asked to score on a scale of 1 to 5, 1 being the best. They mainly considered five aspects. Usability of the framework, Learning effort, Usefulness and Interaction compared to existing tools.

One of the researchers had extensive knowledge on differential geometric models, measures and fiber tracking methods. In this area of research, it is often wished to have a real time interaction with the system. For example, to explore the outcome of fiber tracking along with the information of the underlying structure. Some users found this framework really useful for current research topics. The above framework provided both interactivity and combined fiber glyph visualization as shown in Figure 9.20. This framework has a capability to increase the productivity in everyday research. Another users work involved in estimating diffusion propagator $P(r)$. This kind of work often requires exploration of reconstructed profiles in synthetic or real data. With the available technology, these users could work only on tiny amount of data. With the traditional way of rendering, data had to be processed overnight to see the visualization. In such system real time interaction is quite impossible. This framework perfectly fits

their requirement by giving quick results. There was one minor issue when this framework was tested with different users because of non standardized DTI data formats. Different users used different DTI formats and framework was not capable of rendering all types of DTI formats. But this was rectified by converting the DTI data into the format which framework expected. This indicates that framework cannot handle all the DTI format out of the box and may involve certain initial set up time.

A feedback from the neuroscience researcher was also taken. In this field, current visualization techniques like HARDI method is quite complex and slow to render, which cannot be used for real time interaction. This framework has an obvious advantage over traditional methods with respect to speed and interactivity. One main feature of the framework is much appreciated i.e. of interactive plane, which can be moved around to observe different viewports of the data. This is beneficial in clinical research. Another major advantage of this framework is that it uses ray casting for rendering glyphs which gives minute details. And due to the fact that ray casting method is used, user can zoom in on the visualization to have a closer look into the details without having to loose much of quality in the image or zoom out to have a global perspective. This can be useful in exploring different parts of white matter.

Another user feedback from the DTI validation researcher and application programmer was gathered as mentioned in the paper[9]. Framework was found to be useful like others, and faced some issues with some data conversion. A constructive suggestion was given which stated that it will be beneficial if a user can zoom out and not loose the context of the point from where user zooms out, so that he can zoom in to the same position later on. User gave the feedback saying, learning effort was not too much and manageable and gave a score of 2.

A review session from the neurosurgeon was performed who had no technical pre knowledge about the software, but had experience with DTI data which is mentioned in the paper [9]. This review gives an interesting insight about the capability of the framework. It is observed that, without the support of anatomical data it is very difficult to use this framework effectively in clinical research. Fused DTI/HARDI with the capability of interactivity was much appreciated, but it has few drawbacks, which is avoiding this method to be used in clinical research yet. It was discovered that, for clinical purpose it is important to have minimal details in the visualization and is not too overwhelming. Experimenting with threshold value for the framework in real time for clinical research does not seem to be feasible. Rather, it is recommended that, to make framework more useful and faster, pre determined threshold can be saved and can be used as default value.

All the researchers agreed that, it is a big step in improving visualization and has clear advantage over other techniques. But, exhaustive validation is recommended before using in clinical research. Integration of this framework with anatomical data will make this framework more useful for the researchers and users in this field.

6 Conclusion and Summary

In this seminar paper initially, a basic approach to DTI and HARDI is explored and their advantages and disadvantages were discussed. Traditional visualization and tracking techniques were explained and it showed the need for the fused framework like fused DTI/HARDI. Framework's ability was demonstrated, where user had a good control over the framework. User could change the threshold values by moving the selection pane around the screen to see the changes in real time. Fused DTI/HARDI framework was beneficial to every researcher in this field. Framework's classification method distinguished certain areas of visualization, whose rendering was optimized by choosing faster and appropriate approach, thus improving the overall performance. An user evaluation of the system was also conducted to check the usefulness of the framework. This method, as indicated by many researchers has the potential to be used in clinical research with certain improvements. This framework is one of the major steps taken to improve the visualization of complex HARDI data, and an important contribution in the field of medical image processing and visualization.

References

[1] Basser PJ, Mattiello J, LeBihan D. MR diffusion tensor spectroscopy and imaging. Biophysical journal. 1994;66(1):259–267.

[2] Tuch DS. Diffusion MRI of complex tissue structure. Citeseer; 2002.

[3] Tuch DS. Q-ball imaging. Magnetic Resonance in Medicine. 2004;52(6):1358–1372.

[4] McNab JA. The Bloch-Torrey Equation & Diffusion Imaging (DWI, DTI, q-Space Imaging). 2013;.

[5] Hansen CD, Johnson CR. The visualization handbook. Access Online via Elsevier; 2005.

[6] Basser PJ, Pajevic S, Pierpaoli C, Duda J, Aldroubi A. In vivo fiber tractography using DT-MRI data. Magnetic resonance in medicine. 2000;44(4):625–632.

[7] Mori S, van Zijl P. Fiber tracking: principles and strategies–a technical review. NMR in Biomedicine. 2002;15(7-8):468–480.

[8] Peeters TH, Prckovska V, van Almsick M, Vilanova A, ter Haar Romeny BM. Fast and sleek glyph rendering for interactive HARDI data exploration. In: Visualization Symposium, 2009. PacificVis' 09. IEEE Pacific. IEEE; 2009. p. 153–160.

[9] Prckovska V, Peeters TH, van Almsick M, ter Haar Romeny B, Vilanova i Bartroli A. Fused DTI/HARDI Visualization. Visualization and Computer Graphics, IEEE Transactions on. 2011;17(10):1407–1419.

Surface Reconstruction of Medical 3D Data

Robert Schwieger

Abstract

Recently, there has been an increasing demand of 3D computer models in a variety of fields. Surface reconstruction plays a key role in visualizing medical data obtained from CT/MRI scans or microscopy data. Depending on the application, the given data can be sparse and scattered. In this case, the problem can be compared with the work of an artist inpainting parts of an image. The focus here, is on PDE based approaches, which were originally designed for image smoothing (see [9]) - namely linear diffusion, nonlinear isotropic diffusion according to Perona-Malik and edge enhancing diffusion (EED). Recently, they are used also for scattered data interpolation in the context of image compression [3]. Their feasibility in interpolation is investigated here. Experiments with dipoles are revisited to show empirically their inpainting properties. To provide more examples, the emphasis is on 2D images. The interpolation algorithm can easily be extended to a three dimensional image domain [7].

1 Introduction

1.1 Basic Definitions

A method for *scattered data interpolation* is described here. Let $\Omega \subset \mathbb{R}^2$ or $\Omega \subset \mathbb{R}^3$ denote the *image domain* and let Γ denote the positive half axis:

$$\Gamma \quad = \quad \{t \in \mathbb{R} | t \geq 0\}$$

An image u that varies over time is defined as a continuous function:

$$\begin{aligned} u : \Omega \times \Gamma &\rightarrow \mathbb{R} \\ (x,t) &\rightarrow u(x,t) \end{aligned} \tag{10.1}$$

1.2 Image Inpainting

In [2], image inpainting is defined as a set of techniques for making undetectable modifications to images. One of the best known examples from the "art" is the "making disappear" of Nikolai Iwanowitsch Jeschow - a Soviet secret police official under Joseph Stalin - in Soviet press, seen in Fig. 10.1. Image inpainting, performed in computers, is used for example in [2]:

- removal of an object from a scene
- film restoration
- reverting the deterioration in photographs
- removing red-eye

An example of image inpainting, which is used for image compression in [3] and surface reconstruction is considered here. The algorithm is PDE based. In [3], the goal of PDE-based image inpainting is described as follows:

> "The goal behind PDE-based image inpainting is to compute a reconstruction u of the original image $v : \Omega \rightarrow \mathbb{R}$ that fulfills the following two properties: First of all, u should be identical to v at those locations for which the brightness is known [...]. Secondly, u should own some kind of regularity properties. "

This is now explained in detail:

Abb. 10.1: Inpainting as performed by artists. Top, detail from Cornelia, Mother of the Gracci"by Suvee, from G. Emile-Male, The Restorer's Handbook of Easel Painting. Bottom, from D. King, The Commissar Vanishes"[2].

2 Interpolation Problem

The interpolation problem is formulated in terms of a problem of a partial differential equation. The partial differential equation considered here, has the following form:

$$\partial_t u = Lu \tag{10.2}$$
$$Lu = \operatorname{div}(D\nabla u) \tag{10.3}$$

The letter L denotes a *differential operator*. D is called a *diffusion tensor*. Different choices for D are going to be considered in the sections 3.1, 3.2 and 3.3. The diffusion tensor D is considered in the two dimensional context, where $D \in \mathbb{R}^{2\times 2}$ holds, and also in the three dimensional context with $D \in \mathbb{R}^{3\times 3}$. Important for interpolation is the steady state of the diffusion equation. That means:

$$Lu = 0 \tag{10.4}$$

When the above equation is expressed in terms of an interpolation problem (see [3]), the following equation is obtained:

$$(1 - c_K) \cdot Lu - c_K \cdot (u - f) = 0$$

The problem is considered on a domain $\Omega \subset \mathbb{R}^2$ or $\Omega \subset \mathbb{R}^3$. The set $K \subset \Omega$ is called *interpolation mask* and the function $c_K : \Omega \to \{1, 0\}$ is the *characteristic function* of K. That means:

$$c_K(x) \quad = \quad \begin{cases} 1 & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

The function $v : \Omega \to \mathbb{R}$ is the original image, which should be restored. A function f is chosen with

$$f(x) \quad = \quad \begin{cases} v(x) & \text{if } x \in K \\ 0 & \text{if } x \notin K \end{cases}$$

as initial value and boundary condition. So the above equation states that

$$u(x) \quad = \quad f(x)$$

holds for $x \in K$ and

$$Lu(x) \quad = \quad 0$$

holds for $x \in \Omega \backslash K$. Let $\partial\Omega$ denote the boundary of Ω. Combining this with homogeneous Neumann boundary conditions, the following equation is obtained:

$$\begin{aligned} (1 - c_K) \cdot Lu - c_K \cdot (u - f) &= 0 && (10.5) \\ u(x, 0) &= f(x) && (10.6) \\ \langle D\nabla u, \overrightarrow{n} \rangle &= 0 \text{ on } \partial\Omega && (10.7) \end{aligned}$$

3 Kinds of Diffusion

Several kinds of diffusion can be distinguished here. These definitions are according to [9] (p. 2):

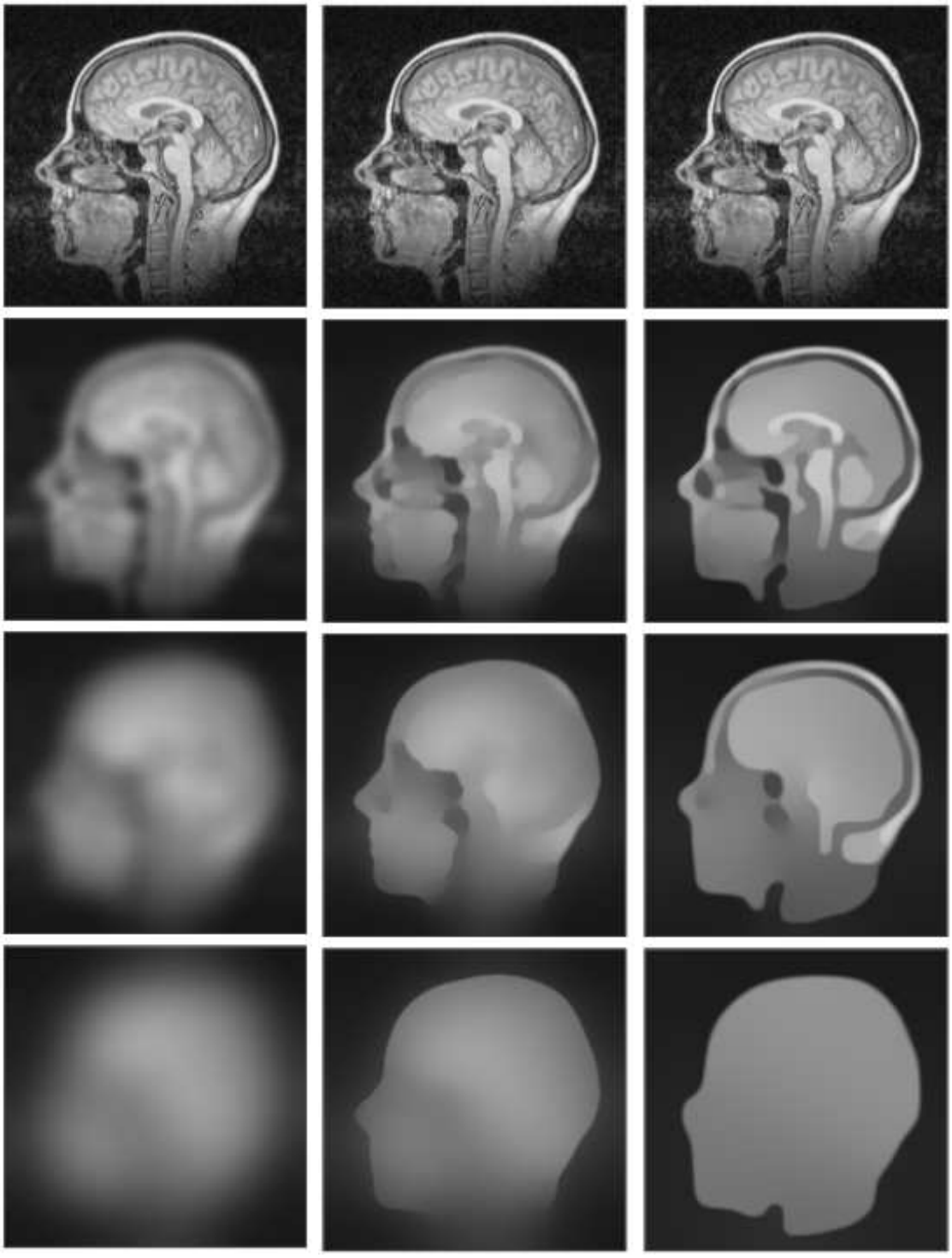
- homogeneous diffusion: D is a constant over the whole image domain.
- inhomogeneous diffusion: D is space-dependent.
- linear diffusion: D does not depend on the evolving image.
- nonlinear diffusion: D depends on the differential structure of the evolving image.
- isotropic diffusion: The eigenvalues of D are equal to each other.
- anisotropic diffusion: The eigenvalues of D differ from each other.

Three kinds of diffusion are going to be important here: Linear diffusion, nonlinear isotropic diffusion and nonlinear anisotropic diffusion. They are going to be explained in detail in the following sections. Diffusion processes were mainly used for image denoising in the past (see [9]). In Fig. 10.2 examples of the important diffusions in the context of image denoising are shown. The parameters listed there will be explained later. Only recently the value of diffusion processes in interpolation has been realized.

3.1 Linear Diffusion

Let $c \in \mathbb{R}$ denote a constant. And let $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ or $I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ denote the unit matrix in two or three dimensions. Linear diffusion is obtained, when D is chosen as follows:

$$D \quad = \quad c \cdot I \qquad (10.8)$$



(a) (b) (c)

Abb. 10.2: Evolution of an MRI slice: Up: Original image, $\Omega = (0, 236)^2$ a) Linear diffusion, top to bottom: $t = 0, 12.5, 50, 200$. b) Nonlinear isotropic diffusion with the Charbonnier diffusivity $(\lambda = 3), t = 0, 70, 150, 400$. c) Edge enhancing anisotropic diffusion $(\lambda = 3, \sigma = 1), t = 0, 250, 875, 3000$ ([9], p.120-121).

3.2 Nonlinear Isotropic Diffusion

Perona and Malik [6] proposed in 1987 the following diffusion tensor, in order to obtain a diffusion process that preserves high contrast areas longer than areas with lower contrast:

$$D = g(\|\nabla u\|^2) \cdot I \tag{10.9}$$

The obective was to smooth an image and keep relevant edges as long as possible. g denotes a scalar valued function:

$$\begin{aligned} g : \mathbb{R} &\rightarrow \mathbb{R} \\ s &\rightarrow g(s) \end{aligned}$$

with

$$g(0) = 1 \tag{10.10}$$
$$g \quad \text{is decreasing.} \tag{10.11}$$
$$g \quad \text{is continuous.} \tag{10.12}$$

The function g is called *diffusivity*. Details and examples of different diffusivities will be presented in section 4.1. In the examples shwon in [3, 4, 7], only Charbonnier diffusivity is used. Nonlinear isotropic diffusion with Charbonnier diffusivity is called Charbonnier diffusion and the resulting interpolation Charbonnier interpolation. In [1] the authors proposed to use the Gaussian-smoothed gradient $\nabla u_\sigma := \nabla(G_\sigma * u)$ instead of ∇u in g

$$D = g(\|\nabla u_\sigma\|^2) \cdot I, \tag{10.13}$$

in order to make the equation more robust against small disturbances in the initial state. This variant is called regularized Charbonnier diffusion.

3.3 Nonlinear Anisotropic Diffusion

As in [3], the focus is on *edge enhancing diffusion(EED)*. To obtain EED, the diffusion tensor D is constructed in the following way:

1. An eigenvector $v_1 \| \nabla u_\sigma$ is chosen and completed to an orthonormal basis $\{v_1, v_2, v_3\}$.
2. Eigenvalues $\lambda(v_1) = g(\|\nabla u_\sigma\|^2)$, $\lambda(v_2) = 1$ and $\lambda(v_3) = 1$ are chosen. The function g needs to fulfill the same properties as in section 3.2. Some examples are shown in section 4.1.2.

The intention for the choice of the eigenvectors is best seen in Fig.10.3 taken from [10]. The gradient ∇u points in the direction perpendicular to the edge. Since $\nabla u_\sigma \| v_1$ holds, the vector v_1 points approximately in the same direction. By choosing an appropriate diffusivity g, it is assured that diffusion is not performed across the edge, when the contrast is high. Details are described in section 4.1. The eigenvalues of the eigenvectors with orientation along the edge are chosen to be 1. This has the effect that diffusion along the edge is not inhibited.

4 Parameters of Diffusion

4.1 Diffusivity and Flux

4.1.1 Properties of the Diffusivity In section 3.2 and section 3.3 a diffusivity is used, which fulfills the following properties:

$$g(0) = 1 \tag{10.14}$$
$$g \text{ is decreasing} \tag{10.15}$$
$$g \text{ is continuous} \tag{10.16}$$

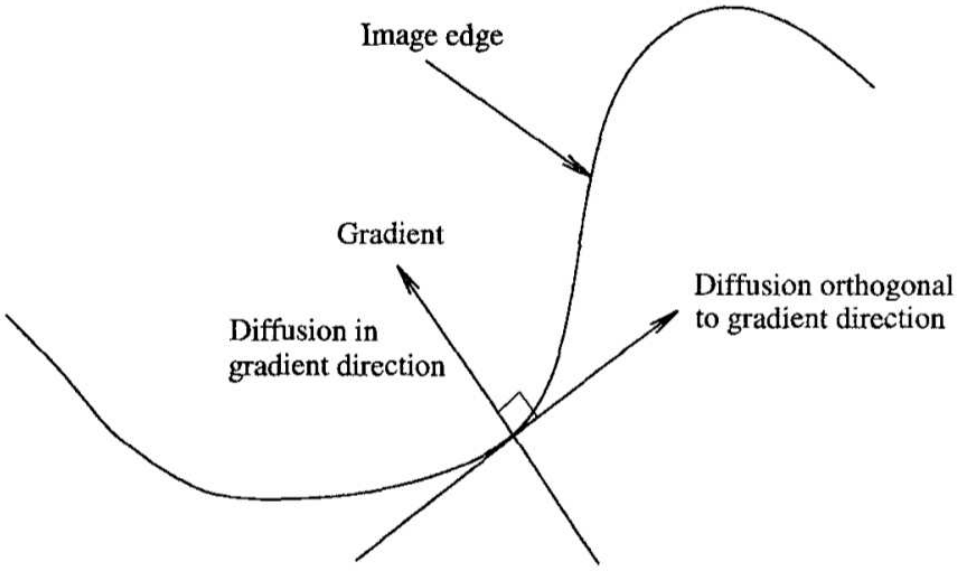


Abb. 10.3: Orthogonal decomposition of anisotropic diffusion [10].

The role of g in the nonlinear isotropic diffusion and nonlinear anisotropic diffusion is very similar. When the contrast $\|\nabla u_\sigma\|$ is approaching zero, the diffusion tensor D becomes similar to the identity matrix due to the condition $g(0) = 1$ and the continuity of g. That means D is converging towards $D = I$, which describes linear diffusion. As mentioned in section 3.1, linear diffusion smooths an image equally in all directions. In contrast, when the gradient $\|\nabla u_\sigma\|$ is big, actually less diffusion occurs in the nonlinear isotropic case, since $g(\|\nabla u_\sigma\|^2) \cdot I$ becomes very small. In the nonlinear anisotropic case, only diffusion across the edge is inhibited. The diffusion tensor becomes increasingly anisotropic. Diffusion tends to occur only along the edge. These properties get differently pronounced depending on the choice of the diffusivity.

4.1.2 Examples of Diffusivities A few examples of functions fulfilling these properties will be shown:

$$g_1(s^2) = \exp(-\frac{s^2}{\lambda^2}) \tag{10.17}$$

$$g_2(s^2) = \frac{1}{\sqrt{1+\frac{s^2}{\lambda^2}}} \tag{10.18}$$

$$g_3(s^2) = \frac{1}{1+\frac{s^2}{\lambda^2}} \tag{10.19}$$

- $g_1(s^2)$ is called here *Perona-Malik diffusivitiy 1.* (see [6])
- $g_2(s^2)$ is called *Charbonnier diffusivity.* (see [3])
- $g_3(s^2)$ is called here *Perona-Malik diffusivity 2.* (see [6])

Plotted graphs of the diffusivities are shown in Fig. 10.4 a). The green graph shows the Charbonnier diffusivity, red Perona Malik 1 diffusivity and black Perona Malik 2 diffusivity. On the x-axis, the squared gradient (that means$\|\nabla u\|^2$ or $\|\nabla u_\sigma\|^2$) is shown and on the y-axis the corresponding diffusivity. When the norm of the gradient disappears, all diffusivity functions take the value one. This is according to eq. (10.15). With growing norm of the gradient, the diffusivities are decreasing. While Perona Malik 1(red) decreases the fastest, Charbonnier diffusivity(green) decreases relatively slowly. Perona Malik 2 is between these two diffusivities. When the diffusivity is decreasing, it does not necessarily mean that also the flux is decreasing. Therefore, also a so called flux function is considered.

4.1.3 Flux Functions The function $\phi(s) := s \cdot g(s^2)$ is called the flux function. It describes approximately the flux across the "edge".

$$\phi_1(s) = s \cdot \exp(-\frac{s^2}{\lambda^2}) \tag{10.20}$$

$$\phi_2(s) = \frac{s}{\sqrt{1+\frac{s^2}{\lambda^2}}} \tag{10.21}$$

$$\phi_3(s) = \frac{s}{1+\frac{s^2}{\lambda^2}} \tag{10.22}$$

The flux functions are shown in Fig. 10.4b). On the x-axis the norm of the gradient is shown ($\|\nabla u\|$ or $\|\nabla u_\sigma\|$) and on the y-axis the corresponding flux. In order to illustrate the difference to linear diffusion, also its flux function is plotted in this figure (that is $\phi(s) = s$ in purple.)Since it is growing quickly in comparison to the other flux functions, it is not shown completely in the figure. When the norm of the gradient is small, the flux functions resemble linear flux. Therefore in areas with low contrast, linear diffusion is approximated.
In case of the Charbonnier diffusivity, although the diffusivity is decreasing, its flux is increasing. Nevertheless, it is much lower than in the linear case. The Perona-Malik 1 flux is smaller than the other flux functions. After reaching its peak, it decreases quickly until it reaches almost 0 around 0.8. This has the effect that less smoothing occurs in areas with high contrast compared with the other diffusivities. The shape of the Perona-Malik 2 flux is similar, but less accentuated.

4.1.4 Contrast Parameter The parameter λ, occurring in the diffusivity and the flux function, is called *contrast parameter*. The meaning of the contrast parameter differs. When the charbonnier function is chosen as diffusivity, λ determines the maximal flux:

$$\begin{aligned}\lim_{s\to\infty} \phi_2(s) &= \lim_{s\to\infty} \frac{s}{\sqrt{1+\frac{s^2}{\lambda^2}}} \\ &= \lim_{s\to\infty} \frac{s}{\sqrt{\frac{s^2}{\lambda^2}}} \\ &= \lambda\end{aligned}$$

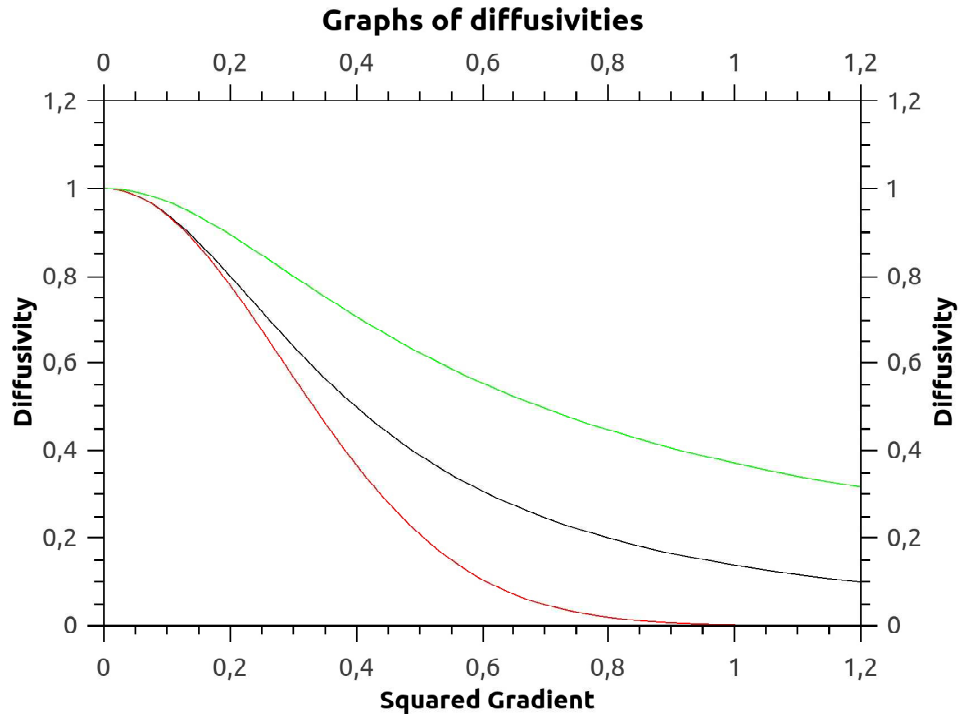
In the Perona-Malik 1 and 2 diffusivity, λ determines the position of the maximum in the flux function. (The Perona-Malik 1 has its maximum at $s = \frac{\lambda}{\sqrt{2}}$, Perona-Malik 2 at $s = \lambda$.) Since EED and nonlinear isotropic diffusion distinguish low and high contrast areas, the choice of λ decides, which contrast is getting smoothed or preserved.

4.2 Regularization Parameter

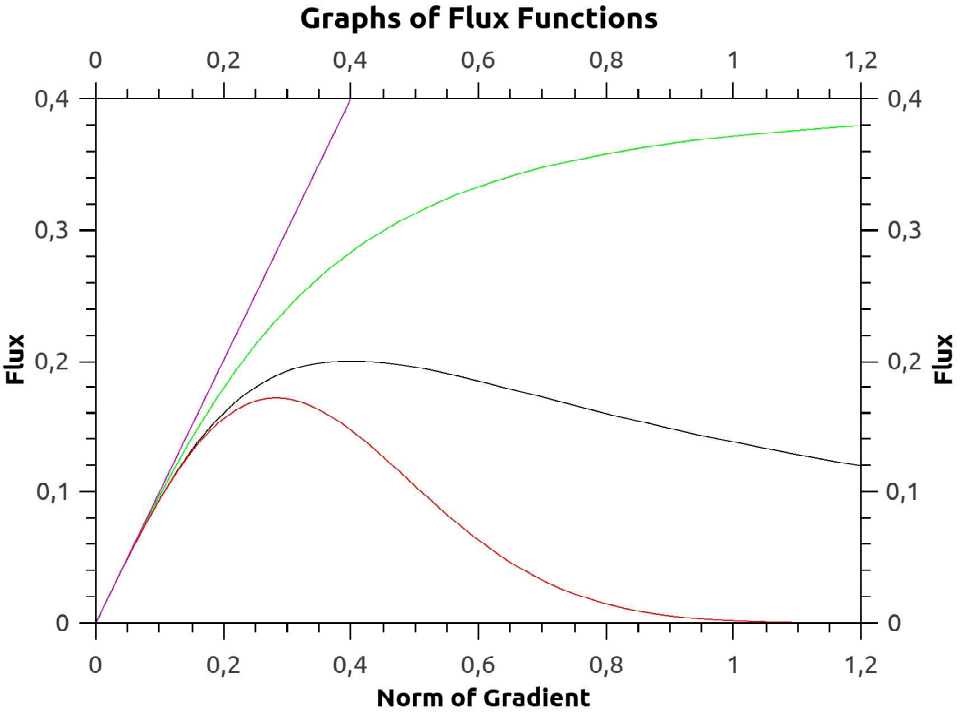
In the regularized Charbonnier interpolation and in EED a regularization parameter σ is used. It determines the amount of Gaussian smoothing in $\|\nabla u_\sigma\|$. This raises the question, how the regularization parameter is influencing the result. In [3] (p.16), it is claimed that the choice of σ does not play a significant role in the output. The authors in [3] summarize, the pure presence of σ seems to be more important than its precise amount.

5 Discretization

The focus here is on discretization with finite differences of eq. (10.2) using EED. Therefore the content of [8] is summarized shortly. The discretization of eq. (10.2) is usually done in two steps. In a first step the image $u((x,y,z),t)$ is replaced by a vector $u(t) : \Gamma \to \mathbb{R}^N$ and $f(x,y,z)$ by $f \in \mathbb{R}^N$. This is done with a grid. The size of the grid corresponds then to the number of entries N of the vectors $u(t)$ and f. Fig. 10.5 illustrates a possible enumeration of the grid. Each entry of the vector $u(t)$ represents the gray value of a pixel or voxel in the image. Therefore the entries of $u(t)$ are sometimes also written in a more convenient format: $u_{i,j,k}(t)$ denotes the gray value of the pixel at position $(i \cdot h_x, j \cdot h_y, k \cdot h_z)$. The grid



(a)



(b)

Abb. 10.4: Graphs of diffusivities and flux with $\lambda = 0.4$. Green: Charbonnier diffusivity; Black: Perona-Malik 2; Red: Perona-Malik 1; Purple: flux in linear diffusion.

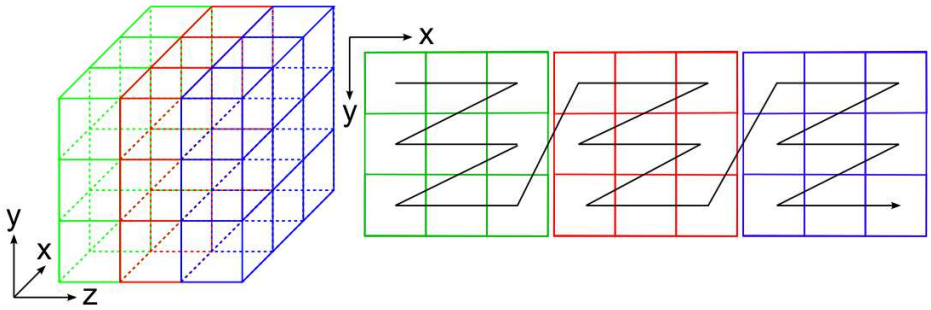


Abb. 10.5: The figure illustrates a possible ordering of three dimensional image coordinates in a vector [8].

distances h_x, h_y and h_z are set to one usually in image processing. The discretization of the occurring partial derivatives is usually done with central differences.

$$\begin{aligned}
\partial_x u_{i,j,k}(t) &= \frac{u_{i+1,j,k}(t) - u_{i-1,j,k}(t)}{2h_x} \\
\partial_y u_{i,j,k}(t) &= \frac{u_{i,j+1,k}(t) - u_{i,j-1,k}(t)}{2h_y} \\
\partial_z u_{i,j,k}(t) &= \frac{u_{i,j,k+1}(t) - u_{i,j,k-1}(t)}{2h_z}
\end{aligned}$$

These discrete partial derivatives are used in the discretization of the divergence, the gradient and the smoothed gradient. Furthermore, when using EED, the diffusion tensor D depends on ∇u_σ for each time and point on the grid. Therefore, it needs to be calculated in each point of the grid. In the end, a so called stencil for each time and coordinate (also known as convolution mask) is obtained (see [8] (p. 37) for details):

$$W(u_{i,j,k}(t)) = \{w_1, \ldots, w_n\}$$

It should not be forgetten that the stencil by itself depends on $u(t)$, since it makes use of the diffusion tensor, which depends on $u(t)$. With this stencil, it is possible to approximate $\partial_t u_{i,j,k}(t)$ as a sum of the gray values from the n-neighborhood $N(u_{i,j,k}(t))$ of $u_{i,j,k}(t)$. The neighborhood of a gray value $u_{i,j,k}(t)$ consists of n adjacent positions on the grid.

$$\partial_t u_{i,j,k}(t) \approx \sum_{l=1}^{n} w_l x_l(t) \tag{10.23}$$

$$x_l \in N(u_{i,j,k}(t)) \tag{10.24}$$

$$w_l \in W(u_{i,j,k}(t)) \tag{10.25}$$

For a complete discretization the resulting eq. (10.23) are usually written in a compact way with a matrix $A \in \mathbb{R}^{N \times N}$. Each row of the matrix A represents the eq. (10.23) for a coordinate (i, j, k). Since the stencil depends on $u(t)$, the matrix depends also on $u(t)$ and it is more precise to write $A(u(t)) \in \mathbb{R}^{N \times N}$ for a fixed $u(t)$. An ordinary differential equation is obtained:

$$\partial_t u(t) = A(u(t)) \cdot u(t) \tag{10.26}$$

$$u(0) = f \tag{10.27}$$

Since the time variable t is still left continuous in eq. (10.26), it is called semi-discrete (see [9]). For the complete discretization of eq. (10.26) it is now possible to apply methods for ordinary differential equations. τ denotes the time step used in the discretization. u^k is an abbreviation for the vector $u(k \cdot \tau)$. The most simple discretization is:

$$u^{(k+1)} = u^k + \tau \cdot A(u^{(k)}) \tag{10.28}$$

Since one is interested in image inpainting in the steady state eq. (10.5), it would be advantageous to use relatively big time steps τ in eq. (10.28). Unfortunately only time steps with

$$\tau \quad < \quad \frac{1}{\max_{i\in\{1,\dots,N\}} |a_{ii}(u^k)|} \tag{10.29}$$

guarantee that eq. (10.28) converges (see[9] (p.103)). A remedy is to use so called semi implicit schemes. They have the form:

$$u^{(k+1)} \quad = \quad u^k + \tau \cdot A(u^k) \cdot u^{(k+1)} \tag{10.30}$$

These schemes do not have any time step restrictions. Since the left side depends on u^{k+1} as well as the right side, solving this is not anymore so easy as in eq. (10.28). For each time step, a linear system of equations needs to be solved. Nevertheless, better results regarding the running time are achieved in this manner. Therefore, semi implicit schemes are used in [3].

6 Experiments

The inpainting properties of linear, nonlinear isotropic and nonlinear anisotropic diffusion were tested in [3, 4]. From the inpainting algorithms it was expected that they complete broken edges. This is called edge propagation. Experiments regarding this are presented in section 6.1. Additionally, edges should be inpainted in a way that the curvature is just big enough to ensure that the inpainted edges are smoothly integrated in the edge fragments. Details regarding this are described in section 6.2.

6.1 Edge Propagation

The inpainting properties of EED, Charbonnier diffusion and linear diffusion were tested in Fig. 10.6(see [3]). The image is similar to the so called Kanizsa's triangle. An optical illusion, where a human observer percepts a triangle. This suggests that also an inpainting algorithm should output a triangle in this case. Only EED is able to reconstruct the triangle properly. Charbonnier interpolation fails to propagate any information from the given circles. Linear diffusion and regularised Charbonnier interpolation perform very similar, but regularised Charbonnier interpolation does not propagate the white area of the triangle so evenly in the image.

The authors in [3] attribute this to the combination of anisotropic behavior and the use of the gradient of the regularized image. The regularized gradient takes the direction of the edges in its neighborhood into account. In this way, edge information from surrounding regions gets propagated.
In Fig. 10.6, it canot be noticed that Charbonnier diffusion (see 4.1.2) is not able to reconstruct the triangle. In nonlinear isotropic diffusion only the amount of contrast gets propagated, but the direction of edges is not taken into account. This negative example confirms the need of anisotropy for edge propagation. While in both cases diffusion across the edge is limited by the diffusivity, Charbonnier diffusion fails to inpaint the edge.
All in all, the success of EED is founded in the use ∇u_σ instead of ∇u and its anisotropic design, which in combination induces a way how information about the steepness and orientation of an edge can be transported to neighboring pixels or voxels.
Another simple image demonstrating that EED is capable of edge propagation, is shown in Fig. 10.7([3]). Starting with a dipole, EED is able to segment the complete image into a black and white part.

6.2 Curvature Minimizing Effect

In sample images, EED has a curvature minimizing effect (see [3], p.9). In order to demonstrate this, in [3] an experiment with four different orientated dipoles (Fig. 10.8) is presented. The result is a circle, which is the form with the least curvature, when the dipoles are assumed to be fixed. Also, the behavior of EED in image denoising is suggesting a curvature minimizing effect. Most obvious in Fig. 10.2, areas with lower contrast get smoothed before areas with high contrast. It can be noticed that parts of the contour with a lot of curvature (e.g. the nose) get straighten by time.

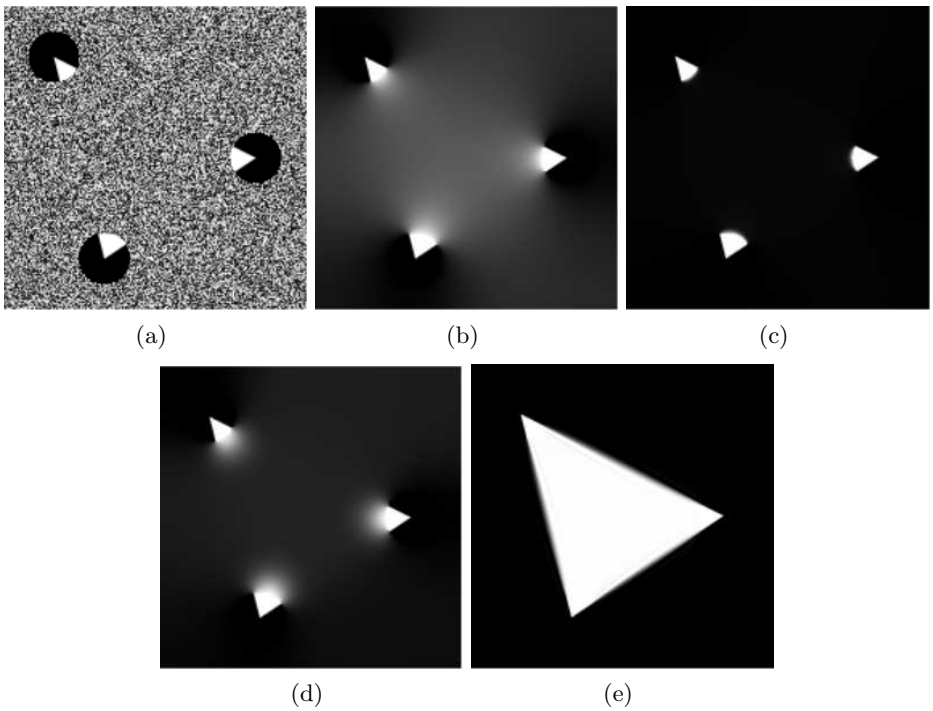


Abb. 10.6: Inpainting from the Kanizsa triangle. a) Initial image, 189 × 189 pixels. The brightness values inside the three disks are specified, while the remainder was initialised with uniform noise. b) Reconstruction result with linear diffusion interpolation. c) Charbonnier interpolation ($\lambda = 0.1$). d) regularised Charbonnier interpolation ($\lambda = 0.1, \sigma = 8$) e) EED interpolation ($\lambda = 0.01, \sigma = 4$) [3].

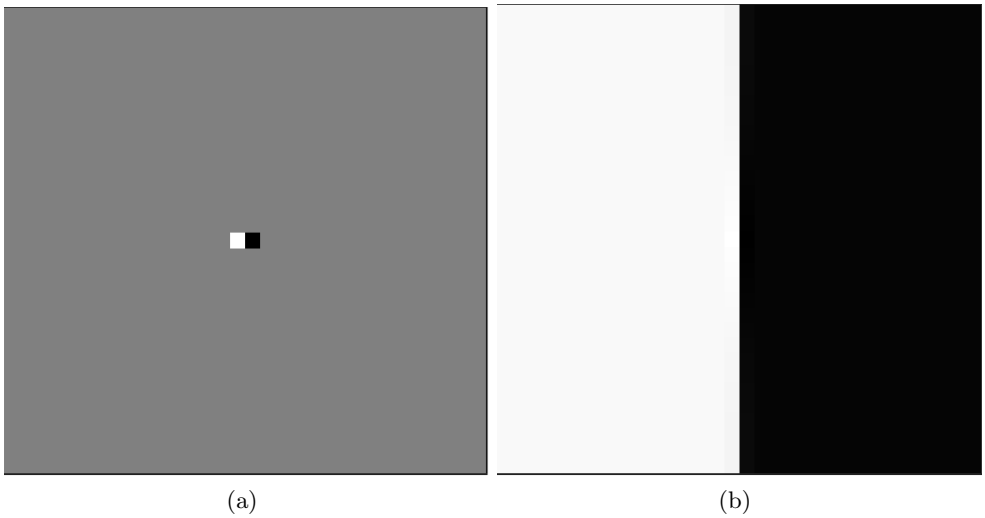
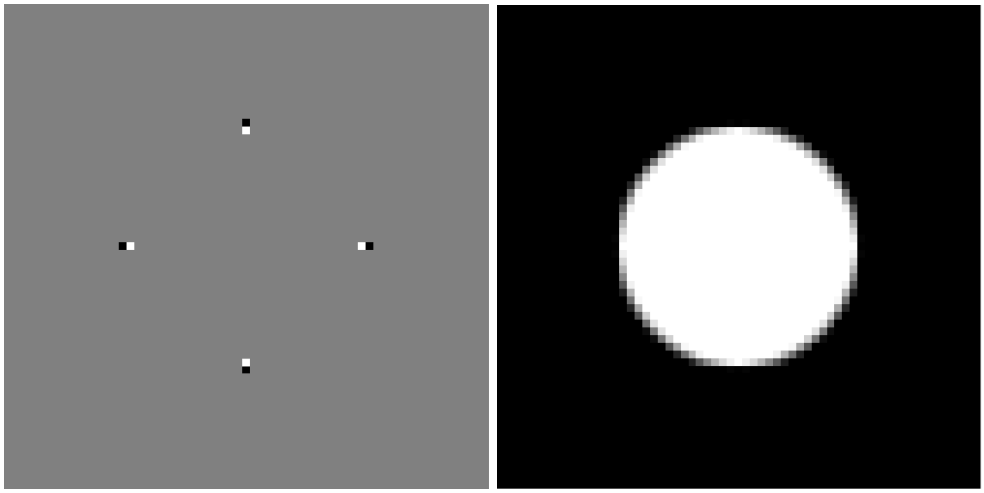


Abb. 10.7: a) input image b) EED interpolation [3].

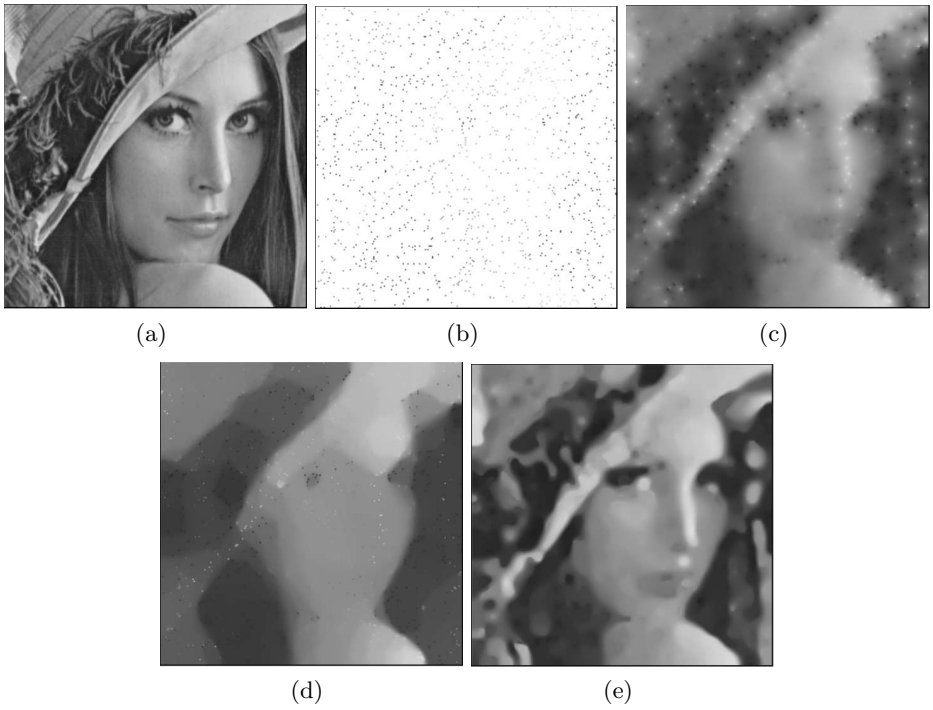


(a) (b)

Abb. 10.8: a) Original image of size 63×63 with 4 dipoles specified. b) After EED interpolation ($\lambda = 0.01, \sigma = 1$) [3]).

6.3 Evaluation

After discretization (see section 5) an $N-$dimensional vector $u \in \mathbb{R}^N$ is obtained as solution of the interpolation problem. As before, the original image is denoted with v and is represented as a vector $v \in \mathbb{R}^N$. The results of the grey-value interpolation were evaluated on example images (Fig. 10.9) (see [4, 5]). The authors used the average absolute error (AAE) and the mean squared error (MSE). In the



(a) (b) (c)

(d) (e)

Abb. 10.9: a) Zoom into the test image Lena, 256×256 pixels. b) Grey values of the scattered data interpolation points (2 percent of all pixels, chosen randomly). c) Interpolation by linear diffusion. d) Nonlinear isotropic diffusion ($\lambda = 0.1$) e) EED ($\lambda = 0.1, \sigma = 1$) [4].

two dimensional case they are defind as follows:

$$AAE(u,v) = \frac{1}{N}\sum_{i,j}|u_{i,j}-v_{i,j}| \tag{10.31}$$

$$MSE(u,v) = \frac{1}{N}\sum_{i,j}|u_{i,j}-v_{i,j}|^2 \tag{10.32}$$

The error measures express how similar the result of an interpolation is to the original image. AAE and MSE depend on the grey-value range of the image, which is $[0, 255]$. Linear diffusion, EED, and Charbonnier diffusion obey an extremum principle (see [9], p.98). That means, the grey scale of the result of the interpolation is not increasing. Therefore, also the results have a greyscale of [0,255]. AAE and MSE can be defined in the same way in three dimensions. Applying linear diffusion, Charbonnier diffusion, and EED to the test image Fig. 10.9 the following results were obtained in [5]:

PDE Method	AAE	MSE
Linear diffusion	16.98	611.5
Charbonnier diffusion	21.80	987.0
EED	14.58	591.7

EED performs the best using AAE as well as MSE. Charbonnier diffusion has the highest error measures in the experiment. Since in Fig. 10.6 already (not regularized) Charbonnier diffusion was performing badly, this is not surprising. It would be interesting to see how regularized Charbonnier diffusion performs. Linear diffusion in comparison has lower AAE and MSE. Nevertheless, the authors in [5] claim, that it is not very suitable for scattered data interpolation since it creates singularities on the interpolation mask. Such singularities can be seen in Fig. 10.9 c). (e.g. on the hat of Lena) as isolated white spots. It is not completley clear how severe this problem is and how much it affects the perception of the interpolated image.

7 Shape Interpolation and Extension to Surface Reconstruction

7.1 Extension of EED to 3D

In order to give more data suggesting the feasibility of EED in a 3D setting, some results of [7] are summarized. It was shown in section 6 that EED can reconstruct shapes in a 2D setting. It is easy to generalize the method to three dimensions. As explained in section 2, instead of a two dimensional diffusion tensor, a three dimensional tensor is chosen. Except of that, the algorithm stays the same. In [3], the inpainting capabilities of EED were used to create a 2D image compression algorithm. In [7], these techniques were generalized to compression of volumes. When using inpainting in image compression, a specified set of data of an image is saved. When decompressing, theses data are used as an interpolation mask. After inpainting, an image similar to the original image is obtained. In [3], compression using EED in 2D is called R-EED. There are two ways to extend compression with EED to the 3D setting: Either the original image is divided into several slices and R-EED compression is applies to each slice. The other approach is to apply the 3D version of EED directly to the volume. This is done in [7]. The resulting technique is called C-EED compression. The results from [7] are shown in Fig. 10.10. The task of reconstructing a volume of a given interpolation mask differs from image compression in one point. While in image reconstruction one cannot choose the interpolation mask, in image compression it is possible to select an optimal interpolation mask for later reconstruction. Therefore, one must be careful when comparing the results of scattered data interpolation of a random set with decompressed images, where the interpolation mask was well chosen. There is an interesting link between shape interpolation and image compression. When choosing an interpolation mask to obtain good compression rates, it is advantageous to choose a lot of pixels along the borders of the objects (see [4]). In Fig. 10.11 such interpolation masks are shown. The pixels, which are stored in the interpolation mask, seem to be also the pixels, which are most relevant for a human observer who wants to recognize the shape of an object. In the next section we describe how shape interpolation can be realized.

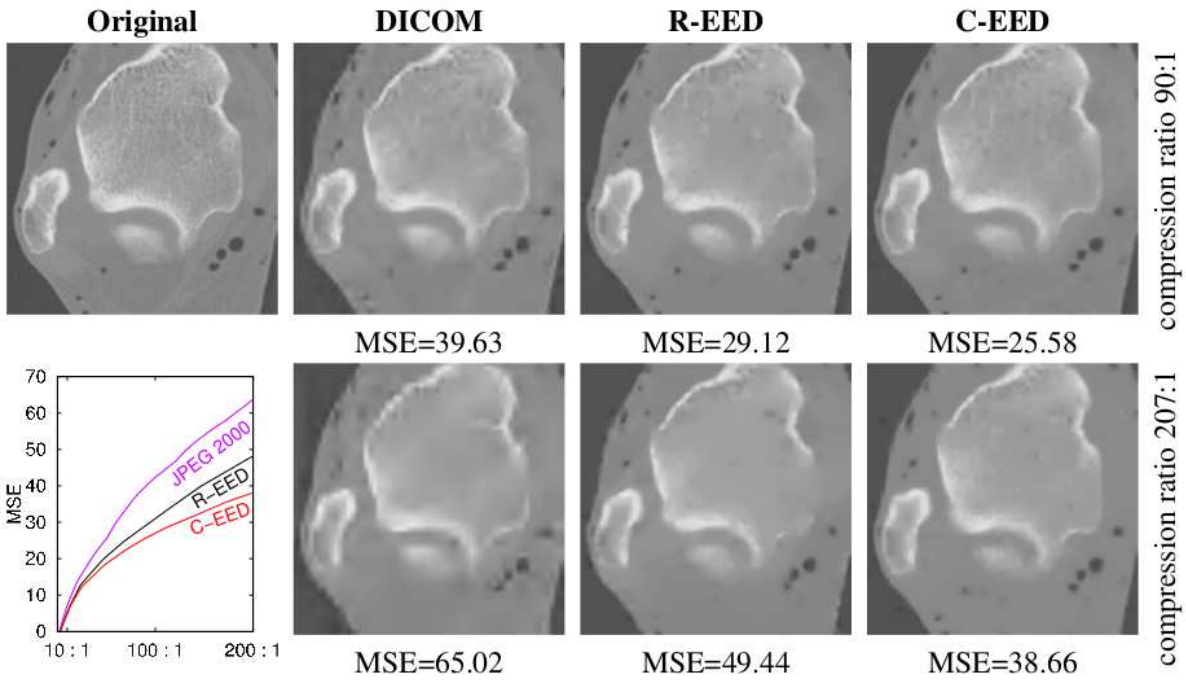
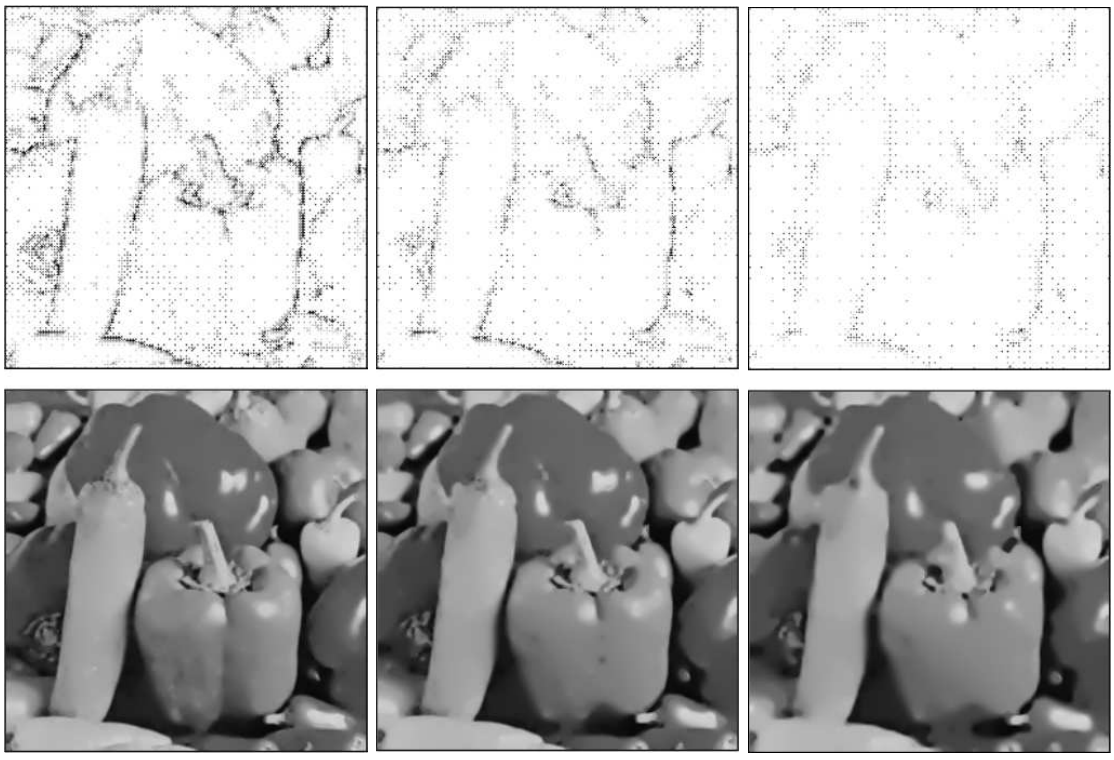


Abb. 10.10: Images obtained with DICOM, R-EED and C-EED with compression rates close to 90:1 (top) and 207:1 (bottom). The test image (size $256 \times 256 \times 64$ consists of the first 64 slices of a trabecular bone. The images above shows a 2D slice of the test image. Smaller MSE values are better [7].



(a) (b) (c)

Abb. 10.11: Different interpolation masks that are used for compression. The masks have more pixels along the shapes of the figures than in areas with low contrast [4].

7.2 Shape and Surface Interpolation

In the previous chapters, it was shown how to inpaint images from scattered data. Now, a method to reconstruct shapes in a two dimensional image or surfaces in a three dimensional image is shown. That means the objective is not to reconstruct a grey-value image, but a binary image. A binary image is an image, where a pixel or voxel can take only two values. In [3], a 2D example of this in context of image compression is presented. So called quadrupoles are selected from the contour of a cat Fig. 10.12. While in areas with contours with high curvature, more quadrupoles are chosen, in areas with relativley streight contours less quadrupoles are chosen. A quadrupole is a square of four adjacent pixels. It encodes the direction of a line. The chosen quadrupoles were saved in the interpolation mask and reconstructed with EED. The result is a grey-value image. In order to obtain a binary image, gray values bigger than a specific threshold were set to one and gray values lower than the specified treshhold were set to zero. The result is shown in Fig. 10.12. By replacing the quadrupoles with a three dimensional version, the algorithm can be extended to surface reconstruction. Since this generalization to the 3D setting is not done in [3, 5, 4], no results can be shown here.

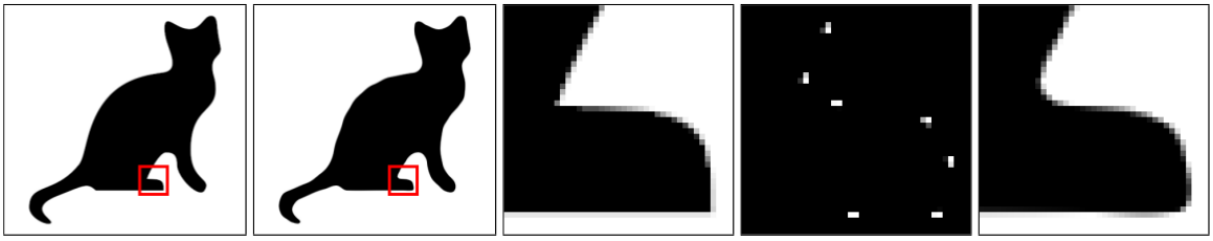


Abb. 10.12: Illustration of the EED shape coding capabilities with 4 percent of all quadrupoles (67 of 1672). The red rectangles marked are curvy areas with more quadupoles. From left to right: First image: Original shape (400 × 380 pixels). Second image: Interpolation result with EED, MSE: 77.79. After treshholding the binary image differs in 464 pixels. This is approximatly 0.3 percent of the pixels. Third image: Zoom into marked area of the original shape. Fourth image: Corresponding selected quadrupoles for zoomed region. Fifth image: Zoom into marked area in the interpolation result [3].

8 Conclusion

It was shown, how different kinds of diffusion - originally designed for image denoising - are applied to image inpainting. These kinds were linear diffusion, nonlinear isotropic diffusion and EED. These methods were constructed in 2D as well as in 3D. Also reasons, why they work, were explained in sections 4.1, 6.1 and 6.2. In section 6, some experiments with dipole images, done in [3, 4], were shown. Here EED provides the best results. Furthermore, in section 6.3 an example image was evaluated quantitatively. Using MSE as well as AAE, again EED provides the best results. Although image inpainting with EED seems to work quite well, it is necessary to see its performance in more examples and test its quality more systematically to give a final answer, how adaptive it is to different kinds of images. In section 7, the relation to surface reconstruction was explained. It was shown, how EED can be applied to binary images to reconstruct shapes. Furthermore, a generalization of these methods to 3D was described. Since this was not the objective in the original papers, no example images where available here. Nevertheless, in section 7.1 it was demonstrated, how EED performs in the context of 3D compression. Since the 3D EED compression algorithm includes image inpainting, its good performance suggests that shape reconstruction with EED can be successfully extended to surface reconstruction. However, it was noticed that there is a fundamental difference between image compression and scattered data interpolation. In contrast to image compression, it is not possible to choose the interpolation mask freely. Nevertheless, in image compression, often most pixels in the interpolation mask are chosen from the contours in the image. Again this suggests that surface reconstruction in this manner could perform well. Since the original sources do not mention surface reconstruction with EED at all, 3D results are limited to the context of image compression.

Literaturverzeichnis

[1] Luis Alvarez, Pierre-Louis Lions, and Jean-Michel Morel. Image selective smoothing and edge detection by nonlinear diffusion. ii. *SIAM J. Numer. Anal.*, 29(3):845–866, June 1992.

[2] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[3] M. Mainberger F. Ebel J. Weickert A. Bruhn C. Schmaltz, P. Peter. Understanding, optimising, and extending data compression with anisotropic diffusion. Technical Report No. 329, Department of Mathematics, Saarland University, SaarbrÃ¼cken, Germany, March 2013.

[4] Irena Galic, Joachim Weickert, Martin Welk, Andres Bruhn, Alexander Belyaev, and Hans-Peter Seidel. Towards pde-based image compression. In Nikos Paragios, Olivier Faugeras, Tony Chan, and Christoph Schnoerr, editors, *Variational, Geometric, and Level Set Methods in Computer Vision*, volume 3752 of *Lecture Notes in Computer Science*, pages 37–48. Springer Berlin Heidelberg, 2005.

[5] Irena Galic, Joachim Weickert, Martin Welk, Andres Bruhn, Alexander Belyaev, and Hans-Peter Seidel. Image compression with anisotropic diffusion, 2008.

[6] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639, Jul 1990.

[7] Pascal Peter. Three-dimensional data compression with anisotropic diffusion. In Joachim Weickert, Matthias Hein, and Bernt Schiele, editors, *Pattern Recognition*, volume 8142 of *Lecture Notes in Computer Science*, pages 231–236. Springer Berlin Heidelberg, 2013.

[8] Pascal Tobias Peter. Kompression dreidimensionaler daten mit anisotropen diffusionsprozessen. Master's thesis, 2011.

[9] Joachim Weickert. *Anisotropic Diffusion In Image Processing.* B.G. Teubner Stuttgart, 1998.

[10] Y.-L. You, Wenyuan Xu, A. Tannenbaum, and Mostafa Kaveh. Behavioral analysis of anisotropic diffusion in image processing. *Image Processing, IEEE Transactions on*, 5(11):1539–1553, 1996.