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 SPITZER

Hauptseminar im Wintersemester 2014/15

Medizinische Bildverarbeitung

Volume 10, Band 1
ISSN 1860-8906
ISBN 978-3-9813213-8-8

Aachener Schriften zur Medizinischen Informatik

Aachener Schriften zur Medizinischen Informatik
ISSN 1860-8906
ISBN 978-3-98-13213-8-8

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 2014/2015 ten 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 three 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, Thomas Seidl, 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 2015

Daniel Haak, Stephan Jonas

Inhalt

Hauptseminar Medizinische Bildverarbeitung

Vorträge der Blockveranstaltung im Wintersemester 14/15

theoretischer Teil/Prof. Rossmanith, 28.01.2015

08:30-09:15	Application of the Ring Theory in Theory in the Segmentation of Digital Images Application of the Ring Theory in Theory in the Segmentation of Digital Images	Referent: *Ligia Bildea* Betreuer: *F. Reidl*
09:15-09:30	Diskussion	
09:30-10:15	Converge of the Mean Shift Using the Infinity Norm in Image Segmentation	Referent: *Christopher Tenter*
10:15-10:30	Diskussion	Betreuer: *F. Sánchez V.*
10:30-11:15	Head Tracking and Flagellum Tracing for Sperm Motility Analysis	Referent: *Daniel Hariri* Betreuer: *S. Sikdar*
11:15-11:30	Diskussion	
11:30-12:15	Detection and Segmentation of Cell Nuclei in Virtual Microscopy Images: A Minimum-Model Approach	Referent: *Belavadi Poormina*
12:15-12:30	Diskussion	Betreuer: *D. Haak*
12:30-13:15	Atlas-based 3D registration	Referent: *Thomas Fundament*
13:15-13:30	Diskussion	Betreuer: *A. Serrurier*

praktischer Teil/Prof. Seidl/Prof. Ney, 29.01.2015

14:00-14:45	Efficient EMD-based Similarity Search in Medical Image Databases	Referent: *Andreas Straub* Betreuer: *C. Beecks*
14:45-15:00	Diskussion	
15:00-15:45	Signature Matching Distance for Content-based Image Retrieval	Referent: *Christoph Rackwitz*
15:45-16:00	Diskussion	Betreuer: *C. Beecks*
16:00-16:45	Recurrent neural networks for medical image processing	Referent: *Robert Lau* Betreuer: *P. Dötsch*
16:45-17:00	Diskussion	
17:00-17:45	Image fusion with neural networks	Referent: *Jan Kehren*
17:45-18:00	Diskussion	Betreuer: *Patrick Dötsch*
18:00-18:45	Fine-grained image classification with convolutional neural networks	Referent: *Ike Kunze* Betreuer: *H. Hanselmann*
18:45-19:00	Diskussion	

theoretischer Teil/Prof. Rossmanith, 30.01.2015

08:30-09:15	Oriented Tensor Reconstruction Tracing Neural Pathways from Diffusion Tensor MRI	Referent: *Lukas Boersma* Betreuer: *S. Jonas*
09:15-09:30	Diskussion	
09:30-10:15	The Statistical Analysis of fMRI Data	Referent: *Wang Haiqing*
10:15-10:30	Diskussion	Betreuer: *E. Kutafina*
10:30-11:15	Flow measurements in sewers based on image analysis: automatic flow velocity algorithm	Referent: *Jayapal Manasi* Betreuer: *E. Sirazitdinova*
11:15-11:30	Diskussion	
11:30-12:15	Deformed lattice detection in real-world images using meanshift belief propagation	Referent: *Matthias Möller* Betreuer: *T. Deserno*
12:15-12:30	Diskussion	

Head Tracking and Flagellum Tracing for Sperm Motility Analysis Wintersemester 2014/15

Artin Daniel Hariri

Zusammenfassung

Sperm motility analysis plays an essential role in human fertility and animal breeding. The manual observation of sperm quality is time-consuming and subject to intra- and inter-observer variability. In this paper a framework is introduced to detect and track spermatozoon and its flagellar beat pattern. The framework consists of three modules, which build on top of each other. The head detection module detects the spermatozoon heads every 100 frames and passes their position to the head tracking module to keep track of the position and determine the movement direction. This information is then passed to the flagellum tracing module.

Keywords: Sperm Motility Analysis, CASA, Image Processing

1 Introduction

The most important attribute of semen quality is sperm motility. Sperm quality assessment plays an essential role in human fertility and animal breeding. Since manual analysis is time-consuming and subject to intra- and inter-observer variability, Computer-assisted sperm analysis (CASA) systems were introduced[1].
In this seminar paper the algorithms that are used to realize such CASA systems, as suggested in [1], are presented.

2 Spermatozoon Model

Modeling the sperm cells is essential in order to extract and analyse its movement. A model for the head and the flagellum of the spermatozoon is introduced as seen in Figure 1[1].

2.1 Head Model

The head is modeled by an ellipse. Hence it has 5 parameters, denoted as (x,y,a,b,ϕ), where (x,y) are coordinates of the center point, a and b are the lengths and θ is the angle between the major axis and the horizontal direction.

2.2 Flagellum Model

The flagellum is modeled by a sequence of N_s circles. Each circle is closely connected to its adjacent neighbors and has a radius r and an angle θ. The angle θ is defined by the angle between the connecting line of the center of the circle and the center of its predecessor circle and the direction of sperm movement.

3 Head Detection

3.1 Introduction

The goal of the head detection module is to detect an unknown number of sperm heads in a 2D image. To achieve this goal a Multiple Birth and Cut algorithm based on marked point processes is used [1] . I will first introduce marked point processes and then explain the methods of the Multiple Birth and Death algorithm and the Cut-Graph algorithm. Finally I will outline the functioning of the Multiple Birth and Cut algorithm which is a combination of the latter two[2].

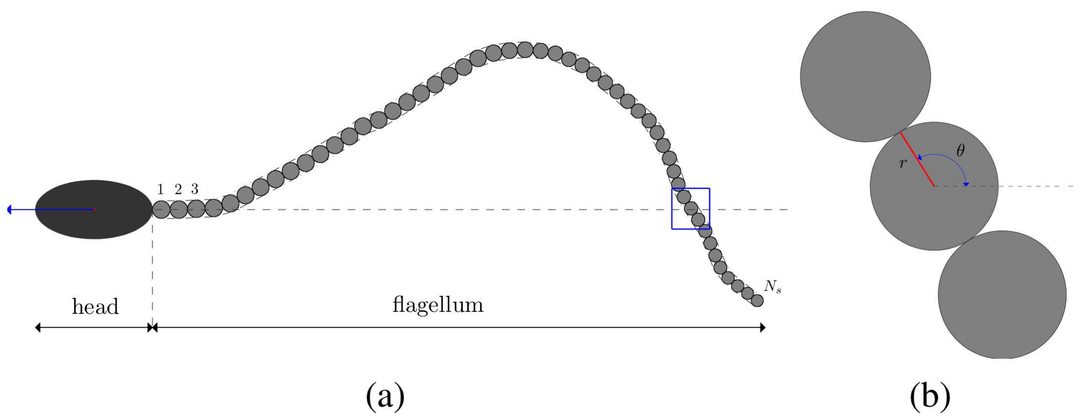


Abb. 1.1: Representation of a spermatozoon. (a) The spermatozoon consists of a head represented by an ellipse and a flagellum represented by N_s circles. (b) Close-up of the blue bounding box in (a). Each circle is parameterized by a radius r and an angle θ.

3.2 Multiple Birth and Cut

3.2.1 Marked Point Processes Marked point processes allow to model scenes made of objects. In this case the objects are ellipses, which represent the sperm heads. The geometrical parameters of any of these objects are called marks. Each mark m_i is associated to a point x_i. Let

$$M = [a_{min}, a_{max}] \times [b_{min}, b_{max}] \times [0, \pi[\tag{1.1}$$

be the mark space, which a and b are the major and the minor axis respectively, for which we select a minimum and a maximum value. An object is defined by $\omega_i = (x_i, m_i) \in K \times M$. A configuration

$$\Omega_n = \{\{\omega_1, ..., \omega_n\} \subset K \times M\} \tag{1.2}$$

is a set containing n objects, where n is a random variable, the number of objects to be detected.
The configuration space is then defined as

$$\Omega = \bigcup_{n=0}^{\infty} \Omega_n. \tag{1.3}$$

3.2.2 Markov Point Process A Markov point process is a marked point process which allows to model interaction between objects. The density of the process is defined by an energy form of the sum of potential over interacting objects.

$$f(x) = \frac{1}{Z} exp[-U(x)], \tag{1.4}$$

where Z is a normalizing constant and

$$U(x) = \underbrace{U_d(x)}_{data\ term} + \gamma_p \underbrace{U_p(x)}_{prior\ term} \tag{1.5}$$

, where γ_p is the weight assigned to the prior term. The data term $U_d(x)$ considers the relevance of the objects in the configuration and the prior term takes interactions between objects into account. Minimization of the energy function corresponds to the right configuration detection.

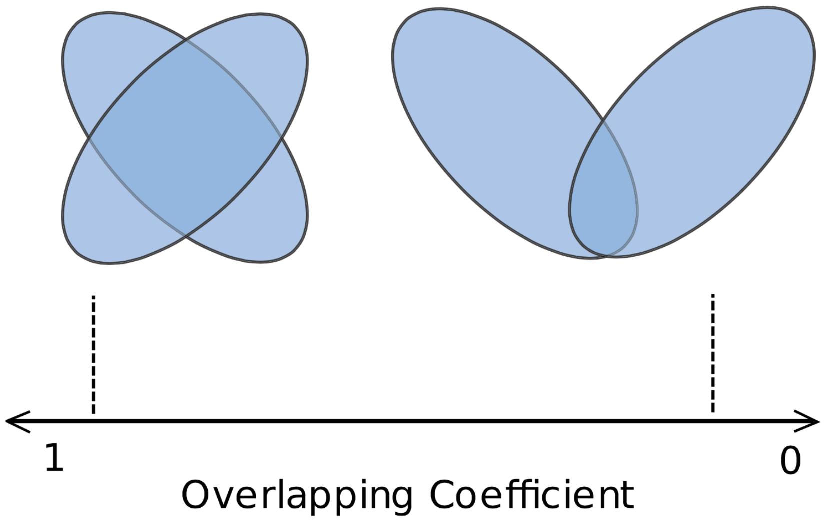


Abb. 1.2: Overlapping Coefficient

3.2.3 Prior Term The use of prior information helps us to regularize the configuration to match the real objects. In reality our objects should almost not overlap, we penalize this situation and introduce an overlapping coefficient between two objects, as illustrated in Figure 2:

$$C(\omega_i, \omega_j) = \frac{A(\omega_i \cap \omega_j)}{min(A(\omega_i), A(\omega_j))} \tag{1.6}$$

, where $A(\omega_i)$ is the area of object ω_i.

The prior energy of a local configuration $\{\omega_i, \omega_j\}$ is given by:

$$u_p(\omega) = \begin{cases} 0, & if\ C(\omega_i, \omega_j) < 0.1 \\ \infty, & if\ C(\omega_i, \omega_j) \geq 0.1 \end{cases} \tag{1.7}$$

, which means that we forbid a configuration with overlapping coefficient greater than 10%. The total prior energy is then given by:

$$U_p(\omega) = \sum_{\omega_i \sim \omega_j} u_p(\omega_i, \omega_j), \tag{1.8}$$

, where $\sim$ is symmetric relation $\sim_r \in K \times M$.

3.2.4 Data Term Given the independence of the data term of each object, data term energy of a configuration ω is given by:

$$U_d(\omega) = \sum_{\omega_i \in \omega} u_d(\omega_i), \tag{1.9}$$

where $u_d(\omega_i)$ is the output of a local filter, evaluating from a data point of view, the relevance of object ω_i. Since every object contains information about its location and its shape, the data term can be interpreted as an adaptive local filter by favoring a specific shape (ellipse) and object depending locally on the data. In the sperm motility scenario a sperm head can be modeled as a dark ellipse surrounded by a brighter background.

3.2.5 Multiple Birth and Death Algorithm (MBD) The idea behind the MBD is that at each iteration of the algorithm, we propose the addition of a new configuration ω' an we treat the new configuration $\omega_{(n)} \cup \omega'$ by removing non-fitting objects. The probability of the death (removal) of a object corresponds to the relevance of the object in the configuration. The birth process is a realization of a *Poisson process.*

3.2.6 Cut-Graph Algorithm The Cut-Graph Algorithm constructs a graph from the energy function to be minimized. Finding the minimum cut of this graph also minimizes the energy. The minimum cut is efficiently calculated using the max flow algorithm.

Let

$$G = (V, E, C) \tag{1.10}$$

be an directed graph, which consists of finite sets of vertices, a set $E \subset V^2$ of edges and a cost function

$$C : E \to \mathbf{R}^+ \cup \{0\}. \tag{1.11}$$

The graph has two special vertices: A source S and a sink T. An S-T cut is a partition (S,T) of the vertices (S $\cup$ T $= V$ and S $\cap$ T $= \emptyset$, such that $S \in$ S and $T \in$ T. The cost of a S-T cut is the sum of the costs of the edges that start in S and end in T:

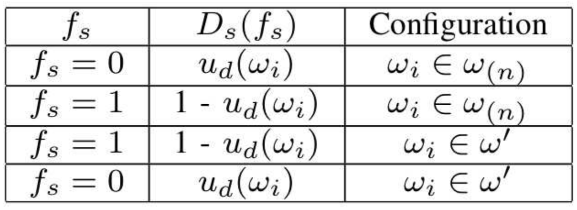
$$C(S, T) = \sum_{u \in S, v \in T : (u,v) \in E} C(u, v). \tag{1.12}$$

Finding a minimal cut in the graph is equivalent to finding the maximum flow from source to sink. In this case the *Ford and Fulkerson theorem* is used. It finds the problem solution in polynomial time with small constants. For a S-T cut, and a labeling function f which maps from V to $\{0, 1\}$ for a binary partitioning, $f(v) = 0$ means that $v \in$ S and $f(v) = 1$ means that $v \in$ T.

3.2.7 Multiple Birth and Cut Algorithm (MBC) Initialization In the first step the unique variable R, the number of objects to be added in each birth iteration, is initialized. In step two a candidate configuration ω' is generated which is set to $\omega_{(0)}$. ω' is a sample of non-overlapping ellipses. $\omega_{(0)}$ is represented in figure 3.(a) in green, $\omega_{(0)} = \{a, b, c\}$.

Birth In each iteration we propose a new configuration ω', e.g. $\omega' = \{d, e, f, g\}$ non-overlapping"(may overlap to a certain threshold) ellipses, which are shown in figure 3.(a) in blue.

TABLE I: Data Term

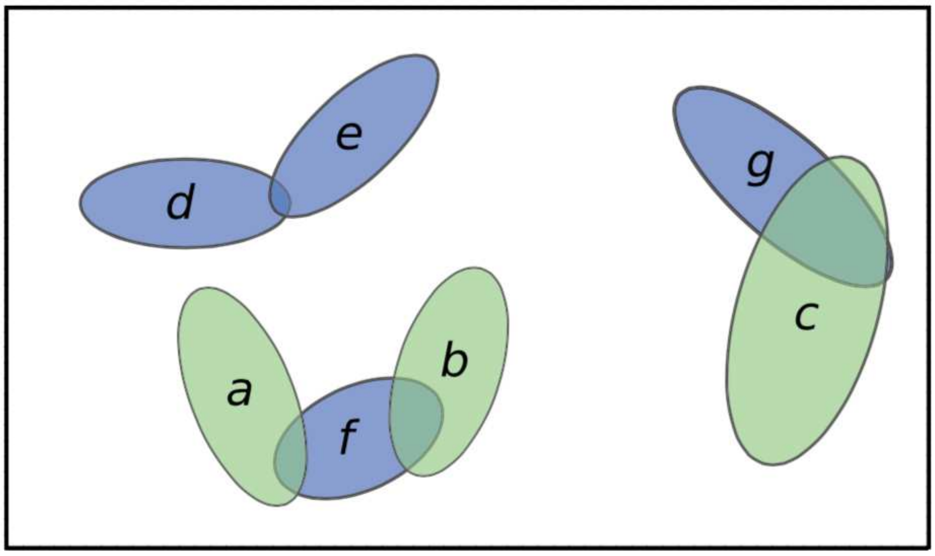


f_s	$D_s(f_s)$	Configuration
$f_s = 0$	$u_d(\omega_i)$	$\omega_i \in \omega_{(n)}$
$f_s = 1$	1 - $u_d(\omega_i)$	$\omega_i \in \omega_{(n)}$
$f_s = 1$	1 - $u_d(\omega_i)$	$\omega_i \in \omega'$
$f_s = 0$	$u_d(\omega_i)$	$\omega_i \in \omega'$

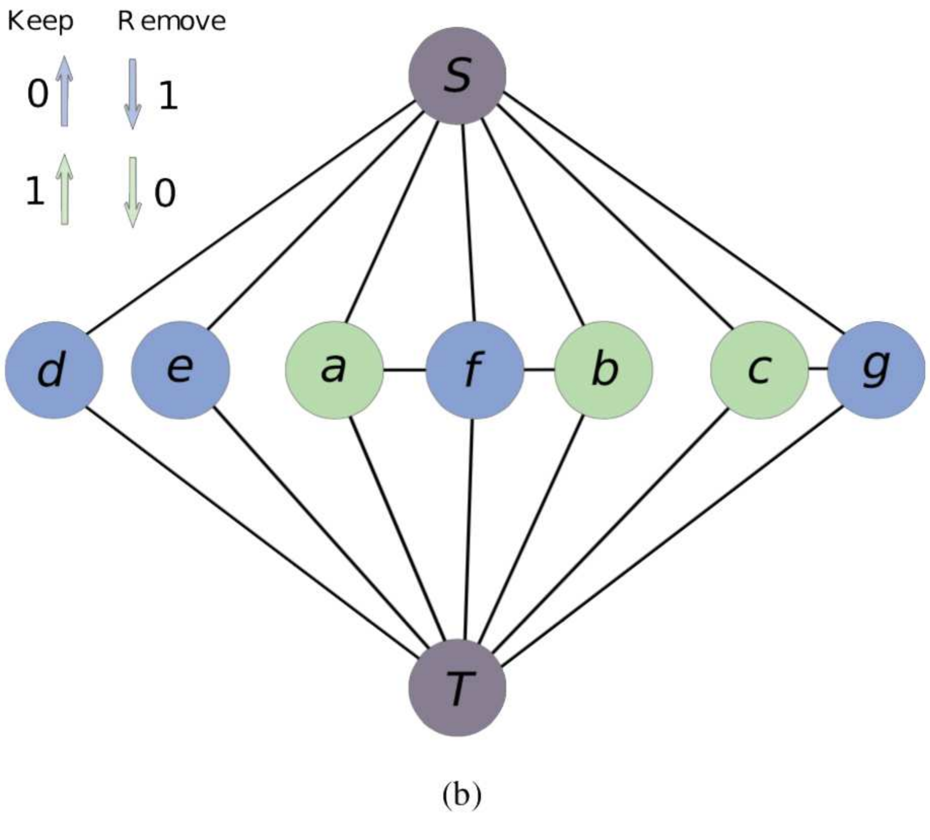
Cut, Graph construction: We construct the graph for the proposed configuration $\omega_{(n)} \cup \omega'$ as shown in figure 3.(b). Each node represents an object ω_i. Between each object and the source and each object and the sink edge weights are assigned as shown in **table I**.
For $\omega_i \in \omega_{(n)}$ the weight to the source is the data term $u_d(\omega_i)$ and $1 - u_d(\omega_i)$ to the sink, while it is the inverse for $\omega_i \in \omega'$, it is $1 - u_d(\omega_i)$ to the source and $u_d(\omega_i)$ to the sink. For the edges between objects, we assign the prior term $u_p(\omega_i, \omega_j)$.

Optimizing: After the construction the graph-cut algorithm is applied, to assign labels $\{0, 1\}$. Label '1' for $\omega_i \in \omega_{(n)}$ means 'keep' this object, label '0' means 'kill' it while '1' for $\omega_i \in \omega'$ means 'kill' this object and label '0' means to 'keep' this object.

These steps are repeated until the algorithm converges[2].



(a)



(b)

Abb. 1.3: a) proposed configuration $\omega_{(n)} \cup \omega' = \{a, b, c\} \cup \{d, e, f, g\}$
b) constructed graph for the in a) proposed configuration

4 Head Tracking

4.1 Introduction

To keep track of the previously detected sperm heads we use a block matching algorithm iterating two stages in a multi-scale manner.The algorithm takes a reference image I and a floating image J as input. The output is the transformation T and the image $J' = J \circ T^{-1}$, which is aligned with I. The composition of computed transformations allows us to track the sperm head through the sequence[1]. The following is based on the work of [3].

4.2 Block Matching Algorithm

4.2.1 Computation of Correspondences by a Block Matching Strategy To detect correspondences of objects we move a block A of the floating image in a neighborhood Ω, and compare it to blocks B that have coincident positions in the reference image. The block size $N_x \times N_y$ and the size of the search neighborhood $\Omega_x \times \Omega_y$ are constant at a given scale level.
The best corresponding block B with respect to given similarity measure, allows to define a match (a_i, b_i) between the centers of the blocks A and B. To improve the computation time the search set Ω is subsampled by considering tentative matches every Σ_x (rest. Σ_y) pixels along x (resp. y) direction. $\Sigma = (\Sigma_x, \Sigma_y)$ determines the resolution of the displacement field. The centers of the test blocks A are taken from a sub-lattice introducing another set of sub-sampling factors $\Delta = (\Delta_x, \Delta_y)$, the density of the displacement field.
To avoid outliers we reject matches corresponding to similarity measures that are below a given threshold.

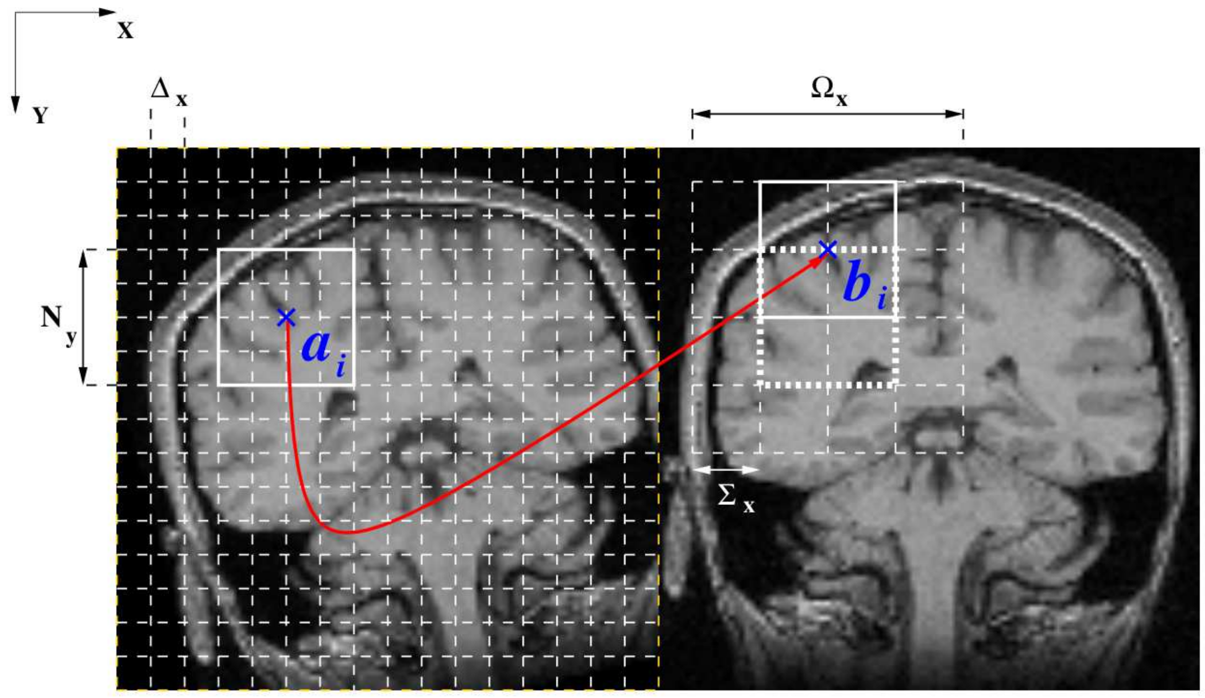


Abb. 1.4: Illustration of the bock matching strategy. On the left the floating image and on the right the reference image. N is the block size; Ω is the size of the search set; Σ is the resolution of the displacement field; Δ is the density of the displacement field.

4.2.2 Multi-scale Scheme Since the complexity of the block matching procedure is proportional to $(N \times \Omega)/(\Delta \times \Sigma)$, we optimize the performance in respect to the computational cost by making use of a multi-scale scheme. We start out with a coarse scale, large values for N, Ω, Δ and Σ. When refining the scale, all parameters are halved. Thus the complexity is independent of the scale level.

4.2.3 Similarity measure For multimodal registrations the correlation ratio (CR) or the mutual information (MI) is used as similarity measure. The cost of the block matching procedure strongly depends on the required time to evaluate the similarity measure. To compute the MI between two blocks we first have to evaluate the joint histogram and then perform $n_a \times n_b$ logarithm computations, n_a and n_b being the respective number of intensity classes in the blocks A and B. As consequence the cost of the evaluation of MI is one order of magnitude greater than when using CR.

4.2.4 Least Trimmed Squares Minimization To find the rigid transformation T that best fits the displacement field computed through block matching we use the least trimmed squares (LTS) regression. In consists in solving the following minimization problem:

$$\min_T \sum_{i=1}^{h} \|r_{i:n}\|^2, \tag{1.13}$$

where $\|r_{i:n}\|^2$ are the squared ordered Euclidean norms of the residuals,

$$r_i = (a_i - T/b_i), \tag{1.14}$$

n is the total number of displacement vectors, and h is set to $h = \lfloor n/2 \rfloor$ to achieve a 50% breakdown point.

5 Flagellum Tracing

5.1 Introduction

According to our model in section 2.2 the flagellum consists of a sequence of N_s circles,

$$C = (c_1, ..., c_{N_s}), \tag{1.15}$$

where c_i represents the ith circle with parameters (r_i, θ_i). Given the image Data D, the goal of the flagellum tracing is to find the most probable sequence of circles

$$\hat{C} = \arg\max_C p(C|D). \tag{1.16}$$

This is achieved by the use of a Markov Chain Monte Carlo algorithm.
By Bayes' rule, the Equation can be rewritten as:

$$p(C|D) \propto \ p(D|C)p(C), \tag{1.17}$$

where $p(D|C)$ is the observation likelihood, and $p(C)$ is the prior.
First the circle prior and the observation likelihood are introduced and then the Markov Chain Monte Carlo algorithm is explained, based on the work of [1].

5.2 Circle Prior

Assuming that the prior, $p(C)$, follows a Markov process, which means that the circle c_i depends only on its predecessor c_{i-1}, it can be factorized as

$$p(C) = p(c_1) \prod_{i=2}^{N_s} p(c_i|c_{i-1}) \ = \ p(r_1, \theta_1) \prod_{i=2}^{N_s} p(r_i, \theta_i|r_{i-1}, \theta_{i-1}) \tag{1.18}$$

We assume that the radius r_i and orientation θ_i are independent of each other. The product of equation (17) can then be split into:

$$p(C) = p(r_1)p(\theta_1) \prod_{i=2}^{N_s} p(r_i|r_{i-1}) \prod_{i=2}^{N_s} p(\theta_i|\theta_{i-1}). \tag{1.19}$$

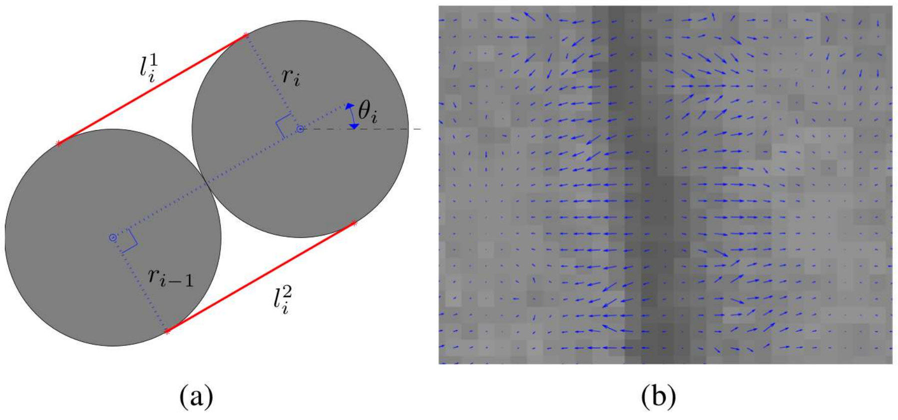


Abb. 1.5: a) l_i^1 and l_i^2 are the line segments that connect the tangent points of c_i and c_{i-1}. b) The vector fields diverge from the centerline of the tail. Therefore the centerline has positive divergence.

Under Markov process assumption, the radius and the orientation of the circle c_i are more likely to be similar to those of the previous circle. A Gaussian centered around r_{i-1} and θ_{i-1} is used to model $p(r_i|r_{i-1})$ and $p(\theta_i|\theta_{i-1})$ which can be mathematically represented as:

$$\begin{cases} p(r_i|r_{i-1}) \propto \exp(-\lambda_r(r_i - r_{i-1})^2) \\ p(\theta_i|\theta_{i-1}) \propto \exp(-\lambda_\theta(\theta_i - \theta_{i-1})^2) \end{cases}, \tag{1.20}$$

where λ_r and λ_θ are constant parameters. r_1 and θ_1 are determined from the detected head and thus $p(r_1) = 1$ and $p(\theta_1) = 1$.

5.3 The Observation Likelihood

The observation model incorporates two constraints: gradient and convergence constraints. Both consider the interaction between adjacent circles.
$p(C)$ can thus be factorized as:

$$p(D|C) = \prod_{i=2}^{N_s} p_g(D|c_{i-1}, c_i) \prod_{i=2}^{N_s} p_d(D|c_{i-1}, c_i), \tag{1.21}$$

where $p_g(\cdot)$ and $p_d(\cdot)$ are the gradient and divergence likelihoods, respectively.

Gradient Constraint The aim of the gradient constraint is to locate the object boundary by observing the line segments between two circles c_{i-1} and c_i as shown in figure 5(a). The line segments are the lines that connect the tangent points of two adjacent circles. The likelihood is then given by:

$$p_g(D|c_{i-1}, c_i) \propto \exp(\lambda_g G(l_i^1, l_i^2)), \tag{1.22}$$

where l_i^1 and l_i^2 are the line segments of c_i and c_{i-1} and $G(\cdot)$ is a contrast invariant energy function.

Divergence Constraint The divergence constraint helps us to better align the centers of the circles with the centerline of the tail. The divergence indicates how much of the vector field is converging or diverging from a given point. If the divergence is positive, the region is called source, if it is negative it is called sink. In Figure 5 b) the vector fields are diverging from the centerline, thus the centerline has positive divergence. The divergence of a vector field v is called is calculated as div

$$v = \frac{\delta v_x}{\delta x} + \frac{\delta v_y}{\delta y}. \tag{1.23}$$

The likelihood is defined as the mean divergence of the line connecting the centers of two adjacent circles:

$$p_d(D|c_{i-1}, c_i) \propto \exp(\lambda_d D_{l_i}), \tag{1.24}$$

where D_{l_i} is the mean divergence of the line l_i connecting the centers of c_i and c_{i-1} and λ_d is a constant parameter.

5.4 Markov Chain Monte Carlo Inference

We want to calculate the maximum a posteriori of equation (17). A Markov Chain Monte Carlo method is used to generate the solution space by constructing a Markov Chain. In this case the Metropolis-Hastings algorithm is applied.
For each circle c_i to be estimated, we construct a chain of N_c samples by starting with an arbitrary initial sample c_i^1 and then iterating for $s = 1...N_c - 1$:
(a) Propose a new circle candidate c_i',
(b) calculate the acceptance ratio

$$\alpha = p(c_i'|D)/p(c_i^s|D), \tag{1.25}$$

(c) accept c_i' with probability α and reject otherwise.

The sample with the highest probability in the chain is selected as the estimated state. The radii and the orientations are estimated in the image in which the head is detected. Since the length of the flagellum does not change in our model only the orientations of the circles are calculated in the subsequent images.

6 Experimental Results

6.1 Data

The framework was evaluated by using two confocal microscopy image sequences of ram semen samples. One sample had a high concentration (50 mM) of beta-mercaptoethanol (bME) and the other one was the control sample. Both sequences were recorded with a frame rate of 200 fps. The frame size was 1280×1084 $x \times y$ pixels, with a resolution of $0.3 \times 0.4 \mu m^2$/pixel[1].

6.2 Pre-processing

Due to the inhomogeneous background and noise, a background subtraction step was performed on a temporal sliding window basis[1].

6.3 Head Detection Results

The Head Detection Module was implemented in C++ and executed on a laptop with CPU 2.4 GHz and 8 GB memory. Its performance was evaluated by comparing the detection results against manually constructed ground truth. The result of the head detection considered the numbers of ground truth, detected objects, correct detections, missed detections and false alarms, as well as precision, for 3 different parameter settings. With tuned parameter settings the highest precision was 88.14% with 78.79% correct detections, 21,21% false alarms and 11.86% missed detections. The computation time for a single frame was about 3 minutes. Figure 6 shows a part of the result with the best parameter setting[1].

6.4 Head Tracking Results

To obtain the head trajectories the head positions were tracked and the angles of rotation of the heads were estimated. The result for each detected head is a sequence of images in which the head is still. Figure 7 (a) shows a part of the minimum intensity projection (MinIP) of 10 consecutive images from such a sequence. Figure 7 (b) shows the head trajectory of a single sperm. The sperm moves forward in a wiggly motion (red) with respect to the average path (blue). The graph in figure 7 (c) shows the angle of head rotation in the subsequent frames with respect to the first frame. The sperm movement involves a periodical behavior [1].

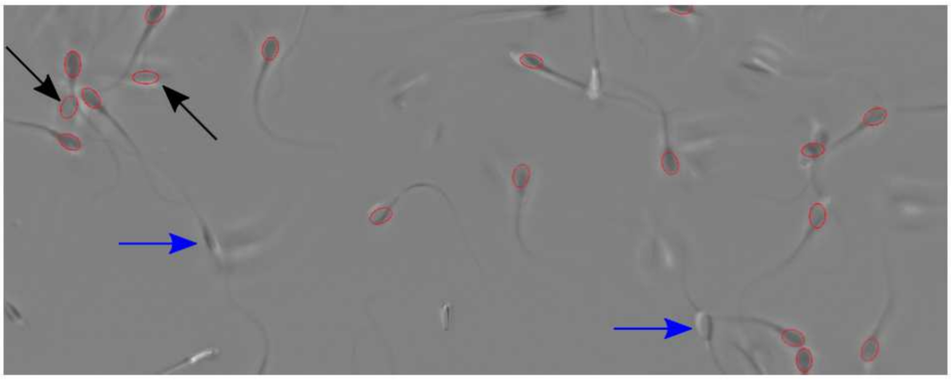
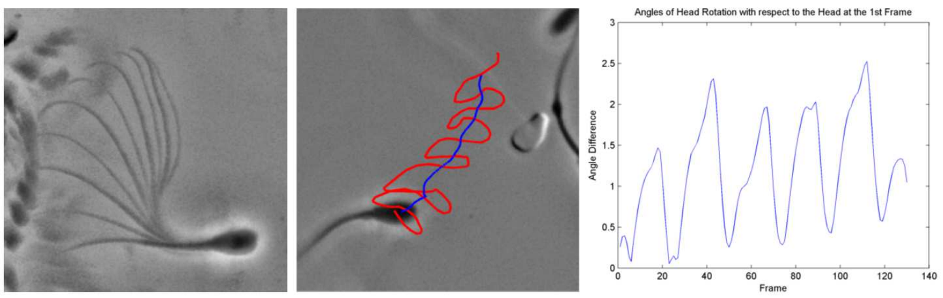


Abb. 1.6: Part of head detection result. The contours of the detected heads are depicted in red. The blue arrows indicate the missed detections and the black arrows (diagonal arrows) indicate the false alarms.



(a) MinIP of 10 imgs (b) Head trajectory (c) Angles of head rotation

Abb. 1.7: a) Minimum Intensity Projection (MinIP) of head-registered images of a sperm b) its head trajectory (red) and average path (blue) c) angles of head rotation with respect to the head on the first time.

6.5 Flagellum Tracing Results

Figure 8 shows the tracing results of the control group and the group under bME effects. More tracing results are shown in figure 9. To evaluate the performance of the flagellum tracing the computer-generated traces were compared against the manual annotations. The missed detection rate (MDR) is the percentage of pixels in the annotation that are more than d pixels away from the computer-generated result. The false detection rate (FDR) is the percentage of pixels in the generated result that are more than d pixels away from the annotation. 4 selected sperms were each traced for 10 images and the value for d was set to 3. The result was 6.12% for the MDR and 1.78% for the FDR.

7 Discussion

The covered abstract described the approach to detect and track sperm heads and trace their flagellum beat patterns but left open some important questions. The head detection for example takes about 3 minutes computational time on a single frame. It was not stated why the performance of the head detection was rather bad. On top of that there was no information about the run time of the head tracking and flagellum tracing module, which would answer the question if the framework is suitable for real-time analysis. Since the results only were experimental results the sample size of 2 microscopy image sequences is too small to make any significant assertions about the performance of the presented framework.

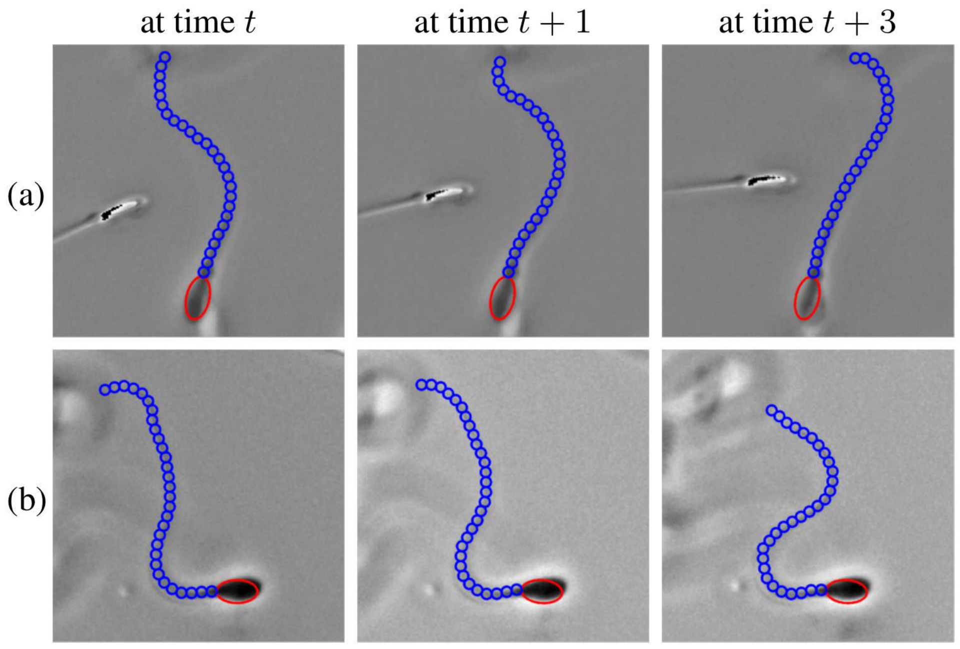


Abb. 1.8: Flagellum Tracing Results. (a) Results of a sperm in the control group. (b) Results of sperm under bME effects. The obtained sequences of circles, along with ellipses, are overlaid on the images at different times. Visually, the tracing module successfully traces the tails.

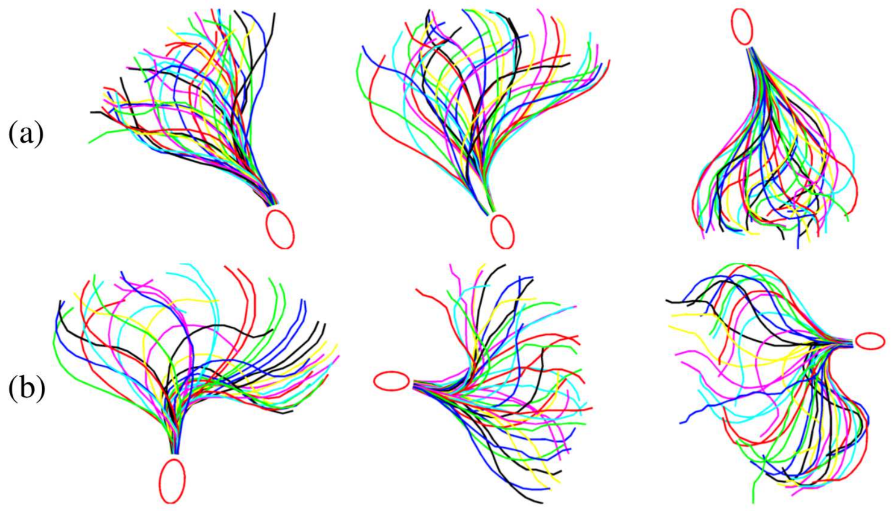


Abb. 1.9: Flagellar beat patterns of sperms under different conditions. Sperms are (a) at normal condition (control) and (b) at a high concentration of bME. Sperms under the high concentration of bME show larger beat amplitudes. Forty traces for each sperm are shown.

Literature

[1] H. Yang, X. Descombes, S. Prigent, G. Malandain, X. Druart, F. Plouraboué: Head tracking and flagellum tracing for sperm motility analysis. International Symposium on Biomedical Imaging, 2014

[2] Josaine Zerubia A. Gamal-Eldin, Xavier Descombes: Multiple birth and cut algorithm for point process optimization. In Sixth International Conference on Signal-Image Technology and Internet Based Systems, 2010.

[3] S. Ourselin, A. Roche, S. Prima, N. Ayache: Medical Image Computing and Computer-Assisted Intervention, volume 1935, chapter Block Matching: A gen- eral framework to improve robustness of rigid registration of medical images, pages 557-566. Springer, 2000.

Efficient EMD-based Similarity Search in Medical Image Databases

Andreas Straub

Zusammenfassung

Technological advances have made imaging ubiquitous in many fiels, including medicine and biology. However, in order to make the high volume of existing data usable for routine diagnostic use, efficient and highly selective query processing techniques are needed. The Earth Mover's distance has enjoyed great popularity in the field of image processing due to its high performance as a similarity metric for visual data. The big drawback of EMD, its high computational cost, has made its application in similarity search computationally prohibitive. This is due to the very large number of distance function evaluations needed to search big databases. Wichterich et al. have proposed a dimension reduction technique for EMD that yields significant speed-up. As a result, search on large data sets can be performed in sufficiently short time frames to facilitate everyday use.

1 Introduction

The Earth Mover's Distance (EMD) has found a wide range of application in the computer vision domain. It can be used as a similarity measure for different kinds of feature descriptors, but is particularly popular popular on histograms. Many of the other distance functions that are used in this domain are built upon statistical interpretations of the data, like histograms as probability distributions (e.g. Bharracharyya distance). In contrast to this, EMD defines distance between two feature descriptors by the minimal amount of work needed to transform one into the other.

One important application for distance functions is similarity search. When searching large databases for images that are similar to a given query image, the quality of the result set depends heavily on the distance function. It implicitly defines which kinds of raw differences in the source images correspond to semantic differences, and which ones are due to intra-class variability. This is of special importance for medical applications, as measurement inaccuracies and quantization effects are prevalent in this domain.

1.1 Motivation

The major downside of EMD is its high computational complexity. For similarity search in large databases, efficiency of distance function computation is very important, as these functions are evaluated very often. Thus, optimization of the distance function computation yields great benefits, and could very well mean the difference between a merely academically interesting method and a practically viable solution.

The EMD's high computational complexity stems from the fact that, at its core, it is a complex linear optimization problem. Although algorithms for faster optimization exist, a more promising solution lies in filter and refine approaches. The idea behind filter and refine is to first apply much more easily computed filters in order to discard "obviously dissimilar" items early. This strongly reduces the number of EMD computations that have to be performed by pruning the candidate set. This work presents one such approach.

In the following, we will first lay the foundation for the presented approach by gving background information regarding multistep query processing frameworks, the formulation of EMD, and how EMD fits into such a framework in section 2. In section 3, we present the approach to speed-up taken by Wichterich et al. [1]. Finally, we present their evaluation in section 4 and conclude in section 5.

2 Background

In order to understand how to optimized EMD computation for similarity search, we first need to understand the environment in which EMD is applied, as well as how the distance function itself operates.

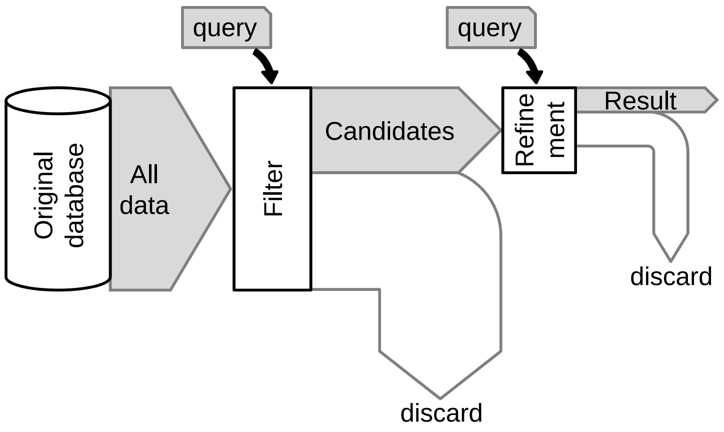


Abb. 2.1: Multistep query processing. Image from [1]

To that end, we first introduce multistep query processing frameworks and filter properties necessitated to apply them. Next, we motivate cross-bin distance functions in general and explain the mechanics of EMD in more detail.

2.1 Multistep query processing

As mentioned in section 1.1, filter and refine approaches present a promising solution to efficiency problems in search applications. By applying a cascade of cheaper filters to remove highly dissimilar candidates in advance, the number of expensive computations of the target distance function is strongly reduced, as visualized in figure 2.1. However, in order for this method to provide actual speed-up while preserving the result set, the filters have to fulfill certain properties.

Efficient The net speed-up attained by applying filter-and-refine can be expressed as the difference between time saved on target distance function computation and time expended on filter function computation. As such, efficiency of filter computation is vital for speed-up.

Lower-bounding In order to preserve the original result set, filter functions can not have any false-negatives, i.e. any item in the original result set must be returned as a candidate by all filter functions. Another way to put this is that the filtering distance function may only ever underestimate distances. If distances could ever be overestimated, some actual result might be discarded by a filter, which would lead to a difference in the final result set from the original result set.

Selective While underestimation of the distance will not corrupt the final result set, it does cause a drop in efficiency. The lower the false-positive rate of the filter, the less target distance function evaluations have to be computed on non-result candidates. Thus, the higher the filter selectivity, the tighter the lower bound on target distance it computes.

2.2 EMD

In image comparison tasks, operations are often not carried out on the raw source images themselves, but rather on feature representations thereof. In particular, histograms are a popular choice for image representation. To compute an images histogram, the pixel value space is quantized into bins $i = 1, \cdots, d$. The histogram then is the feature vector $x = (x_1, \cdots, x_d)$, where x_i is the count of pixels in the image that fall into bin i. This vector essentially encodes a discrete probability distribution of the pixel values in the image. Note that we are always operating on normalized histograms, i.e. the bins do not contain absolute counts, but are relative, such that $\sum_{i=1}^{d} x_i = 1$ holds.

There exist many different candidates for distance functions on histograms, each with their own advantages and drawbacks. The set of histogram distance functions can broadly be separated into two categories: bin-by-bin and cross-bin. Bin-by-bin distance measures only compare the values of individual bins without considering their neighborhoods. The simplest example for this are L_p norms ($L_p(x, y) = \sqrt[p]{\sum_{i=1}^{d} |x_i - y_i|^p}$). The big advantage is that these are usually easy to compute. However, they can be very sensitive to certain patterns of distortion. For example, due to imprecisions in the image acquisition like lighting changes, the color distribution of two images of the same scene might be shifted relative to one another, while preserving the same general shape. A bin-by-bin distance measure would interpret this as a large number of discrepancies, because there are differences in every bin. In contrast, a third image which shares some of the histogram values of the query images with significant changes in others could be classified as more similar.

To solve such misclassifications, cross-bin approaches must be applied. These distance measures can take into account a bin's neighbors – or, even more generally, all other bins. This means that not only simple differences in counts, but rather entire distributions, are compared.

One such cross-bin distance measure is the Earth Mover's Distance. Intuitively, it computes the amount of "work" needed to transform one histogram into another. This "work" is formalized as a so-called "flow", which represents a set of operations moving certain amounts from one bin to another. Larger flows, i.e. those which move bigger amounts, are considered more expensive. Furthermore, flows are weighted by the distance they span. This way, moving a certain amount to an adjacent bin is regarded as much cheaper than moving that amount to a far-away (and thus semantically less-related) bin. There are many different ways of redistributing the amounts from a given histogram such that the result matches a second given histogram. The EMD is defined as the cost of the minimum flow solving the problems, or, more formally:

Definition 2.1 (Earth Mover's Distance) *For two d-dimensional vectors* $x = (x_1, \cdots, x_d)$ *and* $y = (y_1, \cdots, y_d)$, $\forall 1 \leq i \leq d : x_i, y_i \geq 0$ *of normalized total mass* $\sum_{i=1}^{d} x_i = \sum_{i=1}^{d} y_i = 1$ *and a cost matrix* $C = [c_{ij}] \in \mathbb{R}^{d \times d}$, *the EMD is defined as a minimization over all possible flows* $F = [f_{ij}]$ *under positivity constraints* $CPos$, *source constraints* $CSource$, *and target constraints* $CTarget$:

$$EMD_C(x, y) = \min_F \{ \sum_{i=1}^{d} \sum_{j=1}^{d} c_{ij} f_{ij} | Constraints \} \tag{2.1}$$

with $Constraints = CPos \wedge CSource \wedge CTarget$:

$$CPos : \forall 1 \leq i, j \leq d : f_{ij} \geq 0 \tag{2.2}$$

$$CSource : \forall 1 \leq i \leq d : \sum_{j=1}^{d} f_{ij} = x_i \tag{2.3}$$

$$CTarget : \forall 1 \leq j \leq d : \sum_{i=i}^{d} f_{ij} = y_i \tag{2.4}$$

Together, these three constraints enforce that only viable flows are considered as candidates, meaning that only non-negative flows are allowed ($CPos$), the flow neither creates nor destroys any amount ($CSource$), and that the result of applying the flow to the source histogram is the target histogram ($CTarget$). The cost matrix C encodes the pairwise ground distances of the bins, which are used to weight the flows to obtain the EMD.

This formulation of the EMD is a linear program. A popular way of solving it is the simplex algorithm. Although its worst time complexity is exponential in input dimensionality, in practice cubic runtimes are achieved [2]. In our application, input dimensionality is directly related to histogram granularity. Coarser histograms have less bins and therefore lower-dimensional feature vector representations, which decreases runtime. However, as shown by Rubner and Tomasi [2], coarse histograms do not preserve enough information to benefit from cross-bin dissimilarity assessment.

As such, auxilliary approaches must be applied in order to make EMD calculation runtimes feasible. In section 3, we will look at filtering techniques in a multistep query processing framework.

3 Speed-up approach

The idea for speed-up in this approach is to apply "weaker"but faster distance measures first in order to discard candidates that are guaranteed to be too dissimilar to be included in the final result set, before performing similarity search using the target distance function. As we have seen in subsection 2.1, such filters need to fulfill certain properties. While efficiency and selectivity of the filter only impact runtime performance, the lower-bounding property is necessary for result integrity. As such, we will now look at how to design a lower-bounding filter for EMD, and then we will evaluate its efficiency and selectivity.

In contrast to many bin-by-bin distance measures, it is not sufficient to simply drop input dimensions entirely for EMD. In bin-by-bin distance measures, the individual bins usually contribute non-negative error terms to the final distance, which are then summed up. Therefore, leaving out dimensions can only ever reduce the final result by a non-negative term, preserving lower-boundedness. However, in EMD, removing a dimension could remove a cheap option of equalizing source and target histograms via flow, necessitating moving the amount to some other, more expensive bin. This would increase the overall distance computed, and thus breaks lower-boundedness. In the following, we examine how to reconcile these problems by modifying the cost matrix C. Then we will look at how to reduce the input vector dimensions in detail. Lastly, we present two algorithms that approximate this reductions in an efficient manner.

3.1 Dimensionality reduction

First, we introduce a general formalism to describe dimensionality reductions mathematically. Then, we define a subclass of general dimensionality reductions, and derive the optimal cost-matrix reduction for a given input vector reduction.

Definition 3.1 (General linear dimensionality reduction) *A general linear dimensionality reduction from dimensionality d to d' is characterized by a reduction matrix $R = [r_{ij}] \in \mathbb{R}^{d \times d'}$. The reduction of a d-dimensional vector $x = (x_1, \cdots, x_d)$ to a d'-dimensional vector $x' = (x'_1, \cdots, x'_{d'})$ is defined as:*

$$x' = x * R \tag{2.5}$$

For $d' < d$, the resulting vector x' will be of lower dimensionality than the original vector x. However, this very general formulation makes no statement on the form of the reduction matrix R. A very useful restriction is to only consider the subset of reductions that combine one or more dimensions to form a single reduced dimension.

Definition 3.2 (Combining dimensionality reduction) *The set $\Re_{d,d'} \subset \mathbb{R}^{d \times d'}$ of linear dimensionality reduction matrices that reduce the data dimensionality from d to d' by combining original dimensions to form reduced dimensions is defined by:*

$$R \in \Re_{d,d'} \Leftrightarrow \forall 1 \leq d \forall 1 \leq j \leq d' : r_{ij} \in \{0, 1\} \tag{2.6}$$

$$\wedge \forall 1 \leq i \leq d : \sum_{j}^{d'} r_{ij} = 1 \tag{2.7}$$

$$\wedge \forall 1 \leq j \leq d' : \sum_{i}^{d} r_{ij} \geq 1 \tag{2.8}$$

The first restriction 2.6 ensures that dimensions can only ever be assigned in one atomic chunk. As a consequence of this and 2.7, it follows that there can only be exactly one reduced dimension to which one original dimension is assigned. By including 2.8, it is also ensured that no new, empty dimension can be created. Together, these restrictions define a class of reductions that is much easier to reason about than general linear dimensionality reductions.

Definition 3.3 (Reduced Earth Mover's Distance) *For two d-dimensional vectors x, y and cost matrix C according to definition 2.1 and a reduction matrix $R \in \Re_{d,d'}$, the lower bounding reduced EMD is defined as:*

$$EMD_C^R(x, y) = EMD_{C'}(x * R, y * R) \tag{2.9}$$

where $C' \in (R)^{d' \times d'}$ is a lower bounding reduced cost matrix.

With these definitions in place, we now turn our attention toward deriving optimal cost matrix reductions for a given input vector dimensionality reduction. Recall that the cost matrix C used in EMD defines the distance between bins, and as such is dependant on the input dimensionality. Therefore, it immediately follows that a dimensionality reduction on the input vectors has to go hand in hand with a reduction of the cost matrix. Furthermore, the cost matrix reduction is parameterized on the input vector dimensionality reductions in order to be able to preserve lower-boundedness.

As we are taking great care to preserve lower-boundedness, and thus result set integrity, optimality in this context is to be understood in terms of selectivity of the resulting filter. This stems from the fact that a more selective filter will lead to smaller candidate sets, which means fewer refinement computations have to be performed, which increases overall efficiency of the filtering process. Clearly, selectivity is higher for tighter, i.e. greater, lower bounds obtained in the reduced dimensionality space.

Definition 3.4 (Optimal Reduced Cost Matrix) *For two d-dimensional vectors x, y and a cost matrix C according to definition 2.1 and for a reduced EMD_C^R according to definition 3.3, the optimal reduced cost matrix $C' = [c'_{i'j'}$ is defined by:*

$$c'_{i'j'} = min\{c_{ij} | r_{ii'} = 1 \wedge r_{jj'} = 1\} \tag{2.10}$$

The cost matrix with entries for the costs of reduced dimenstions equal to the minima over the costs of the original dimensions that were merged to obtain that reduced cost matrix – as defined in 3.4 – provides a lower bound on the original EMD:

Theorem 3.1 (Lower bound) *Given a reduction matrix $R \in \Re_{d,d'}$ and a cost matrix $C \in \mathbb{R}^{d \times d}$, the reduced cost matrix C' according to 3.4 provides a lower bound:*

$$\forall x, y \in \mathbb{R}^d : EMD_{C'}^R(x, y) \leq EMD_c(x, y) \tag{2.11}$$

The proof can be found in [3] and is a generalization of the proof in [4].

In fact, this is actually already the greatest lower bound. Intuitively, this follows from a simple worst-case assumption. Given a input vector dimensionality reduction matrix, for any two reduced dimensions a, b, under all dimensions merged to create either reduced dimension, find those two original dimensions A, B with the lowest ground distance d according to the original cost matrix C. Construct vectors X, Y with all entries set to 0 except those corresponding to A, B respecitvely. These will be reduced to vectors x, y with all entries set to 0, except those corresponding to a, b respectively. It immediately follows that, by construction, the EMD of X and Y has to be d. Therefore, in order for lower-boundedness to hold, the reduced EMD of x and y can not be higher than d either, which means the ground distance of a and b must not be higher than d.

We observe that this construction can be applied to any pair of reduced dimensions to obtain an upper limit on costs for those reduced dimensions that is equal to the minimum distance between any two of the original dimensions making up either reduced dimension.

More formally, this can be shown using the monotonicity of the EMD in the cost matrix. Higher values in the cost matrix increase the final EMD:

Theorem 3.2 (Monotonicity of the EMD) *Given two cost matrices $C1, C2 \in \mathbb{R}^{d \times d}$, it holds:*

$$C1 \leq C2 \Leftrightarrow \forall x, y : EMD_{C1}(x, y) \leq EMD_{C2}(x, y)$$

where

$$C1 \leq C2 \Leftrightarrow (C1 = C2) \vee (\forall i, j \in \underline{d} : c1_{ij} \leq c2_{ij} \wedge \exists i, j \in \underline{d} : c1_{ij} < c2_{ij})$$

See [3]

Using this theorem and our previous observations on the worst-case assumption, we obtain optimality:

Theorem 3.3 (Optimality) *Given a cost matrix $C \in \mathbb{R}^{d \times d}$ and a reduction matrix $R \in \Re_{d,d'}$, there is no greater lower bound than the one provided by C' accodring to 3.4:*

$$\neg \exists C'' \in \mathbb{R}^{d' \times d'} \forall x, y \in \mathbb{R}^d : \ Tighter \wedge LB \wedge (C'' \neq C')$$

where

$$LB : EMD_{C''}^R(x, y) \leq EMD_C(x, y)$$
$$Tighter : EMD_{C'}^R(x, y) \leq EMD_{C''}^R(x, y)$$

For proof, see [3]

Note that this derivation depends heavily on the choice of restriction on reduction matrices. The authors of [1] state that they also examined alternative dimensionality reduction rechniques such as PCA, but attained much worse runtime performance because of the different cost reductions that had to be applied in order to preserve the lower-bounding property.

Using the method just presented, we can now obtain the optimal cost matrix corresponding to any given input vector reduction. As such, the quality of any reduction as far as efficiency goes is entirely specified through the reduction matrix. However, how we obtain these input vector dimensions is still unsettled. This is the problem we will tackle next.

3.2 Flow-based reduction

In this section, we will examine how to choose the input vector dimensionality reduction matrix in an optimal way. To this end, we must first define an optimality criterion. Next, we present how to incorporate domain knowledge into the reduction process in order to improve the reduction quality. We will discuss problems regarding computability of the optimal reduction, and introduce an approximation method to solve this issue.

As we have previously discussed, a filter's performance relies heavily on its selectivity. Our goal is to reduce the set of candidates as much as possible, so as to minimize the number of refinement steps necessary to obtain the final result. In order to evaluate performance, we can thus measure this number of refinement steps, and pick the reduction that minimizes this number:

Definition 3.5 (Optimal EMD reduction) *Given a workload $w = \{(x_1, \varepsilon_1), \cdots, (x_i, \varepsilon_i)\}$, where x_i is a query vector and ε_i the corresponding range threshold, the optimal reduction $R \in \Re_{d,d'}$ for w is:*

$$R = argmin_{R' \in \Re_{d,d'}} \sum_{(x,\varepsilon) \in w} |\{y \in DB | EMD_C^{R'}(x,y) \leq \varepsilon\}|$$

Using this defintion of reduction quality, the öptimality scoreïs always relative to the workload w. Thus we need to choose a workload that is representative of our target application in order for the optimality to extend to the target use case. But this also means that the reduction optimization is tailored specifically to this application, while disregarding other workloads.

However, this definition is not immediately usable for derivation of reduction matrices due to the large search space. As all pairwise distances between workload items and database items need to be computed, $|w| * |DB|$ individual linear optimiation problems have to be solved, meaning that a total number of $d'^{(d-d')} * |w| * |DB|$ reduced EMDs need to be computed for exhaustive enumeration of the search space. Clearly, this is infeasible. As such, the authors introduced heuristics to obtain efficient reductions for specific target applications.

The idea behind the presented approximation method is to incorporate knowledge of the underlying data by examining flows computed for unreduced EMDs. Considering the purpose of EMD reduction is to speed up unreduced EMD evaluation, it may seem counter-intuitive to apply the computationally more expensive unreduced EMD as preprocessing for the reduction. Nevertheless, the speedup attained by guiding the reduction using unreduced EMD flows far outweighs the cost incurred by it. This is because the expensive computation of unreduced EMDs only has to be carried out once, while the time savings incurred by the reduction accumulate over each query.

As we have seen in section 3.1, combining reduction matrices and associated optimal reduced cost matrices provide a lower-bounding filter on the unreduced EMD. In order to obtain a selective filter, it is thus desirable to find the reduction matrix that maximizes EMD scores. Because the cost terms $c'_{i'j'}$ in the reduced EMD sum $c'_{i'j'} * f'_{i'j'}$ are derived from the reduction iself, the maximal EMD score is attained if the reduction is chosen such that the reduced flows $f'_{i'j'}$ are maximized with respect to $c'_{i'j'}$.

To speed up this maximization process, the flows in the reduced EMDs are approximated by averaged flows from the unreduced EMD rather than computed explicitly. To this end, the average flow matrix $F^S = [f^S_{ij}]$ with $f^S_{ij} = \frac{1}{|S|^2} \sum_{x,y \in S} \hat{f}_{ij}(x,y)$ over a sample S of the database is computed for unreduced EMDs. Given a reduction matrix R, the reduced EMD flow $f'_{i'j'}$ can then be approximated by summing up those f^S_{ij} where i is reduced to i' and j is reduced to j' by R.

$$aggrFlow(F^S, R, i', j') = \sum_{\{i | r_{ii'}=1\}} \sum_{\{j | r_{jj'}=1\}} f^S_{ij}$$

By computing the optimal cost matrix C' from R and using it to weight the approximated reduced EMD flows, an approximate tightness score for the reduction R is obtained:

$$\sum_{i'=1}^{d'} \sum_{j'=1}^{d'} aggrFlow(F^S, R, i', j') * c'_{i'j'} \tag{2.12}$$

Again, this term can not be globally optimized over all possible reductions R, as exhaustively enumerating them is infeasible. Thus, the authors propose two variants of an algorithm that iteratively reassigns single original dimensions to reduced dimensions in order to continually improve the reduction matrix.

The starting point for these improvement algorithms is an initial reduction matrix. Because we are only looking at combining reductions, the reduction is basically equivalent to a clustering of the input dimensions. One possible initialization is *k-medoid* clustering, as this allows use of arbitraty distance metrics. In this case, the cost matrix of the EMD is used as distance metric between the input dimensions. Running the clustering algorithm then yields a partition of the input dimension, with the dimensions in a single cluster being similar to one another and different from those in other clusters according to the EMD cost function. This and other initializations are described in more detail in [3].

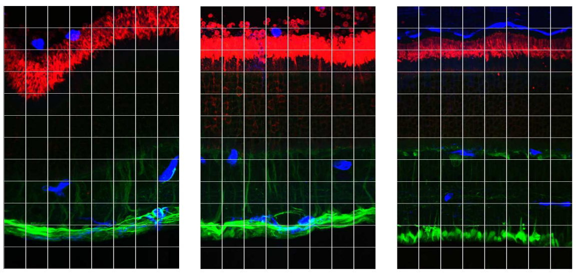
Algorithm 1 $optimizeFB - MOD(R, C, F, d, d')$

```
1:  $origDim \leftarrow 0; lastOrigDimChanged \leftarrow 0;$
2:  $currentTightness \leftarrow calcTight(R, C, F, 0, R.getAssignment(0), d');$
3:  repeat
4:    $threshold \leftarrow currentTightness * min_improve;$
5:    for $redDim \leftarrow 1$ to $d'$ do
6:      $swapTightness \leftarrow calcTight(R, C, F, origDim, redDim, d');$
7:      if swapTightness - currentTightness > threshold then
8:        $currentTightness \leftarrow swapTightness;$
9:        $R.reassign(origDim, redDim);$
10:       $lastOrigDimChanged \leftarrow origDim;$
11:       break;
12:     end if
13:   end for
14:   $origDim \leftarrow (origDim + 1) mod d;$
15: until origDim = lastOrigDimChanged
16: return R;
```

Algorithm 2 $calcTight(R, C, F, origDum, newRedDim, d')$

```
1:  $result \leftarrow 0;$
2:  $R' \leftarrow R;$
3:  $R'.reassign(origDim, newRedDim);$
4:  $C' \leftarrow C.reduce(R');$
5:  for $i' \leftarrow 1$ to $d'$ do
6:    for $j' \leftarrow 1$ to $d'$ do
7:      $result \leftarrow result + aggrFlow(F, R, i', j') * C'[i'][j'];$
8:    end for
9:  end for
10: return result;
```

This initial matrix R is then improved upon by iterative reassignment of original to reduced dimensions. The algorithm loops through all original dimensions. For each original dimension d, it is checked for each reduced dimension d' whether assigning d to d' would improve the reduction according to equation 2.12. If so, the dimension d is immediately assigned to reduced dimension d'. Then, the next original dimension is evaluated in the same manner, restarting with the first original dimension once the end is reached. This process repeats until each original dimension was looked at in succession once without making any assignment. The authors refer to this variant of the algorithm als FB-Mod (flow-based reduction – modulo). Detailed pseudocode is given in algorithm 1 and algorithm 2, both taken from [1].


(a) RETINA

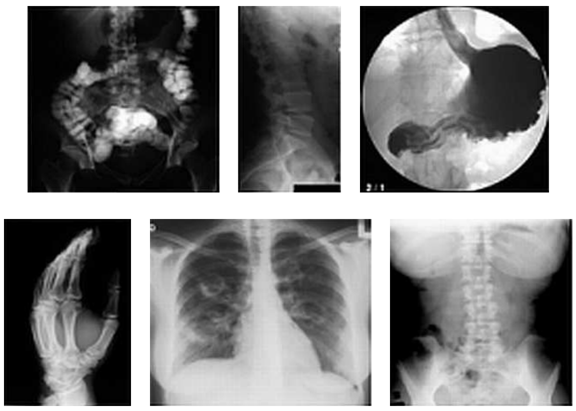

(b) IRMA

Abb. 2.2: RETINA (left) and IRMA (right) dataset examples. Images from [4] and [6]

The second variant follows the same basic scheme, but it does not commit to the first improving reassignment it encouters. It loops through all possible reassignments once, and in every iteration chooses that reassignment which yields the biggest improvement in tightness. The authors refer to this variant of the algorithm as FB-All.

3.3 Query processing algorithm

As previously explained, the proposed dimensionality reduction technique can easily be combined with other filtering techniques for EMD, because the reduced-dimension distance function is again an EMD problem. This property allows us to reuse previous results in the field of EMD effiency research, making full use of the multistep query processing framework.

For their evaluation, the authors applied the LB_{IM} technique introduced in [5], performed on dimensionality reduced features, as a first filtering step. As thesecond step, they applied general EMD, again on dimensionality reduced features. Lastly, they perform refinement using EMD on unreduced features.

Furthermore, these steps are not run in sequence. Rather, individual objects are passed through the entire filter cascade in order from best to worst according to the individual filters' distance functions. This allows the query processing to break early if the final result set has already been found. This can be determined due to the lower-bounding property of the individual filters. If the largest distance computed in refinement of any of the objects in the candidate result set is smaller than the next object's distance computed by the filter, the candidate set is the final result set and query processing can be ended. This is because the objects are passed on in order from best to worst, and the filters only ever underestimate target distance. Note that for this property to work, the individual filters have to be lower-bounding relative to the next filter step, not just the final distance function. For more detail, see [3].

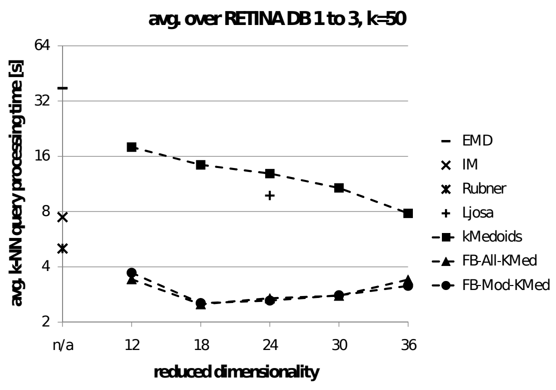
4 Evaluation

For evaluation of the proposed query processing pipeline, real world medical data sets were used. We now briefly describe these datasets and we give a summary of the results.

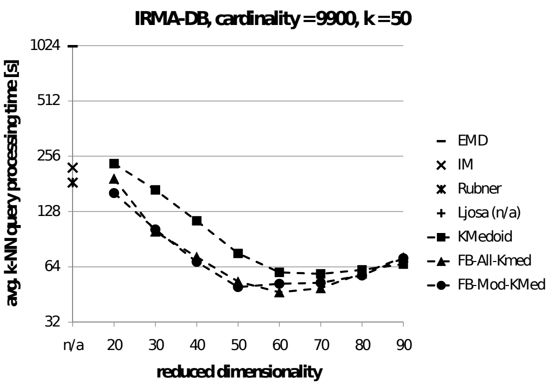
4.1 Data sets

Two different data sets were used for evaluation. Both contain bio-medical images, and examples are shown in figure 2.2.

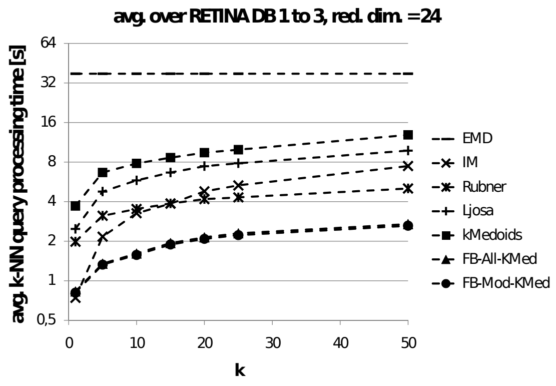
The first one is the RETINA set from the UCSB Bioimage Database. It contains images of retina scans from cats, that have been labeled with various antibodies. For evaluation of EMD lower bounds techniques, Ljosa et al. [4] computed histograms representing tile-based spatial distribution of color layout descriptor measures on this data set. These histograms were reused for this evaluation. Specifically, the authors truncated the 96-dimensional histograms, preserving only the first three dimensions, denoted RETINA1-ALL through RETINA3-ALL. For the dimensionality reduction matrix computations, sample



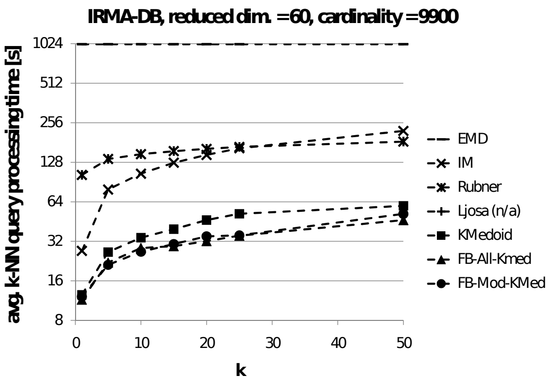
(a) RETINA



(b) IRMA



(c) RETINA



(d) IRMA

Abb. 2.3: Timing results for both datasets, IRMA and RETINA, graphed against competing approaches from literature. Shown is the impact of different target dimensionalities (top), and different neighborhood sizes (bottom). Images from [1]

sets of size 383 were used, which equals about 10% of the total data set. Out of the 3932 images in the original dataset, the histograms of 100 were used as query sets, and the rest were used as database sets.

Secondly, the authors used the IRMA dataset. This dataset contains radiographs from the Image Retrieval in Medical Applications project [6]. As with the RETINA set, the authors again made use of previous work on feature extraction for this dataset, performed by Deselaers et al. [7]. In particular, the evaluation was performed on the image-wise cluster frequencies. They are represented by 199-dimensional histograms. The cost matrix for the EMD ground distance is computed from Euclidean distances between the 199 cluster centers. A sample set of size 1000 was used to compute the reductions. The database contained 9900 images, and the query set contained the remaining 100 images.

4.2 Result summary

A general trend that is visible in the evaluation of the dimensionality reduction technique is the influence of the number of reduced dimensions. A distinctive “U-shape” is evident in the graphs for the proposed approach in figures 2.3(a) and 2.3(b), meaning there is a sweet spot of number of target dimensions which optimizes query processing time.

This classic tradeoff is the result of two interacting properties influencing processing time.Reducing to too few dimensions gives up discriminability in the features, which yields lower selectivity. Therefore, even though the reduced computations themselves are faster for low dimensions, too few candidates can be filtered out at this stage. As a result, a high number of refinement computations have to be performed, which negates the time saved during the filtering stage. On the other hand, while byreducing the number of dimensions by only a small amount, a high selectivity can be obtained for the filtering stage, the reduced EMD computations now take such a large amount of time themselves, that the savings in time spent on refinement is lost.

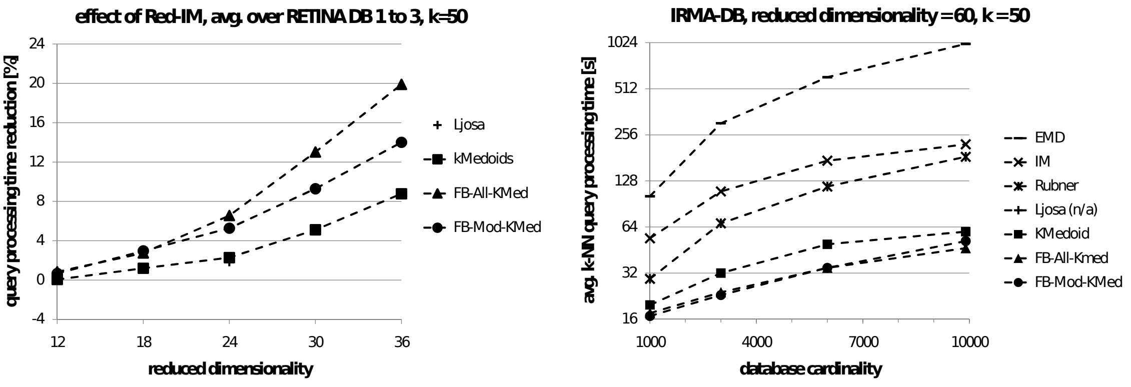


Abb. 2.4: Impact of existing filtering technique Red-IM 2.4(a) and database size 2.4(b) on query processing time. Images from [1]

Notably, the proposed dimensionality reduction technique still outperformed the competition at all reduction levels for both data sets. As can be seen in figures 2.3(c) and 2.3(d), speed-up factors varied with different k values for the k-nearest-neighbor queries. However, the overall ordering of the evaluated approaches stayed consistent, with the proposed approach winning out or tieing the competition.

Additionally, the authors have shown that their proposed method integrates well with existing filtering techniques for the EMD. Figure 2.4(a) visualizes the effect of combining the proposed dimensionality reduction with the Independent Minimization filtering technique from [5].

Lastly, it can be observed in figure 2.4(b) that query processing times scale similarly for different approaches across increasing database sizes. The authors conclude from this, that the sample size used to compute reduction matrices was sufficient in order to identify important flows.

5 Conclusion

The authors showed that leveraging domain knowledge in the form of target database analysis can be used to greatly speed up similarity search. In particular, a specific dimensionality reduction technique, that makes use of characteristics extracted from the target database in order to guide the reduction process, was introduced. This proposed method has the peculiar property of performing a number of computations of the target distance metric that is supposed to be approximated by the dimension-reduced distance function. While this seems counter-intuitive, their evaluation has shown that significantly reduced query times more than make up for the upfront cost incurred by this approach. As a result, the highly effective Earth Mover's Distance was sped up sufficiently to make its application in real-world medical scenarios feasible.

Literaturverzeichnis

[1] Wichterich M, Kranen P, Assent I, Seidl T. Efficient EMD-based Similarity Search in Medical Image Databases. In: Plant C., Böhm C. (eds.): Database Technology for Life Sciences and Medicine, World Scientific Publishing. Singapore: World Scientific; 2010. p. 175–201.

[2] Rubner Y, Tomasi C. Perceptual metrics for image database navigation. vol. 1. Springer; 2000.

[3] Wichterich M, Assent I, Kranen P, Seidl T. Efficient EMD-based similarity search in multimedia databases via flexible dimensionality reduction. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data. ACM; 2008. p. 199–212.

[4] Ljosa V, Bhattacharya A, Singh AK. Indexing spatially sensitive distance measures using multi-resolution lower bounds. In: Advances in Database Technology-EDBT 2006. Springer; 2006. p. 865–883.

[5] Assent I, Seidl T. Efficient multi-step query processing for EMD-based similarity. In: Content-Based Retrieval; 2006. .

[6] Lehmann TM, Gold M, Thies C, Fischer B, Spitzer K, Keysers D, et al. Content-based image retrieval in medical applications. Methods of Information in Medicine. 2004;43(4):354–361.

[7] Deselaers T, Keysers D, Ney H. Discriminative training for object recognition using image patches. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. vol. 2. IEEE; 2005. p. 157–162.

Review of the Signature Matching Distance for Content-based Image Retrieval

Christoph Rackwitz

Zusammenfassung

Beecks, Kirchhoff, and Seidl [4] introduced the Signature Matching Distance (SMD) and showed it to be a simple yet effective approach to content-based image retrieval. I have reviewed their paper and the related work and examined the SMD for properties of a metric.

Keywords: content-based image retrieval, distance function, feature signature, signature matching distance, metric

1 Introduction

Content Based Image Retrieval [1, 2] describes the retrieval of digital images using a query image. The resulting image set should look similar to the query image, which requires algorithms to capture the high-level content of images as judged by humans. For this application, Bag-of-Visual-Words [3] has become a common approach to map images to "visual words" that can be handled similarly to textual words. Improvements in accuracy on the BoVW approach have been made, but these improvements typically are computationally more demanding.

Signatures are the alternative approach. The signature approach describes an image in terms of an image-specific visual vocabulary that adapts to specific image properties, rather than an universal vocabulary learned in advance. The resulting signatures then need to be compared to determine similarity between images. Popular distance-based similarity measures [5, 6] for this task are the *earth mover's distance* [7] or the *signature quadratic form distance* [8].

The *Signature Matching Distance* [4] now incorporates a matching-based similarity definition into the signature-based model. The SMD defines image dissimilarity solely based on a matching of image signatures, without needing a common visual vocabulary. The SMD outperforms thus outperforms other signature-based approaches in efficiency and accuracy. The SMD can even compete with most BoVW approaches by using only a simple color and texture-based descriptor [9]

2 Local Feature Descriptors

One method to describe the contents of an image is called *Local Feature Descriptors.* This is in contrast to *global* descriptions of images, such as color histograms.

The idea is to find corners, called *feature points*, in the image that can be found very likely in another image of the same scene. The immediate area around each feature point is described using a *feature descriptor* in a way that is invariant to various image perturbations such as rotation and scaling, or varying illumination.

2.1 Common Definitions

2.1.1 Descriptor Space This is a vector space that contains the feature descriptor vectors. Signatures may also be in this space, or a subspace of it. Mathematically, it is generally $\mathbb{R}^d$, but computer implementations use machine floating point numbers, fixed point numbers, or even integers.

2.1.2 Ground Distance $\delta : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ describes the dissimilarity between two descriptors or signatures.

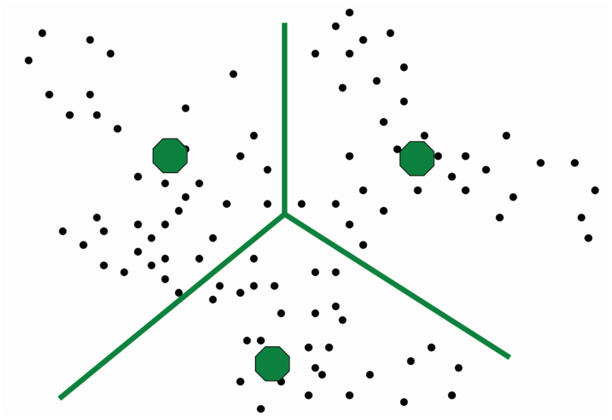


Abb. 3.1: Clustering a set of descriptors into voronoi cells and their centroids

2.1.3 Set of representatives $R_X \subset \mathbb{R}^d$ This denotes a small set of $k = 10..100$ points in the descriptor space that results from clustering (Abb. 3.1) the set of feature descriptors and taking the centroid of each cluster. The elements of this set are called representatives, or more technically, centroids.

2.1.4 Weights $w_X : R_X \to \mathbb{R}^{\geq 0}$ describe the size of each cluster and are attributed to the representative of a cluster. This value is commonly the set size of a cluster.

2.1.5 Signature $X = \{(x, w_X(x)) \mid x \in R_X\}$ denotes the set of all centroids and their weights. Its size for evaluations in [4] was on the order of $k = 10..100$.

2.2 Position, Color, Texture (PCT)

This is a conceptually simple feature descriptor and fast to compute [5]. It contains the pixel position, a measure of its color (CIELAB color space), and a description of local texture (coarseness and contrast, Tamura features [9]).

2.3 History of Oriented Gradients (HOG)

A histogram of the gradient orientations in a small area around a point of interest is computed. This may be a feature point, or it may be a regular grid across the image (usable for template matching to find shapes).

2.4 Scale-Invariant Feature Transform (SIFT)

This descriptor [12] incorporates scale invariance (extrema in a Laplacian pyramid) and rotation invariance (orientation of dominant gradients) to yield very robust descriptions. It is computationally expensive, but optimized, GPU-accelerated implementations exist.

3 Image Matching Approaches

As hinted at in the introduction, there are two popular approaches to comparing images. I will briefly describe and contrast them now.

3.1 Bag of Visual Words

The Bag-of-Visual-Words approach makes use of a previously learned visual vocabulary (Abb. 3.3) that is modelled to fit the problem domain. The descriptors of an image are reduced to this set of visual words (Abb. 3.4).

Common visual vocabularies must be learned from large databases of images and may contain tens of thousands or even millions of visual words. These vocabularies may be structured into Vocabulary Trees [13] for fast lookup.

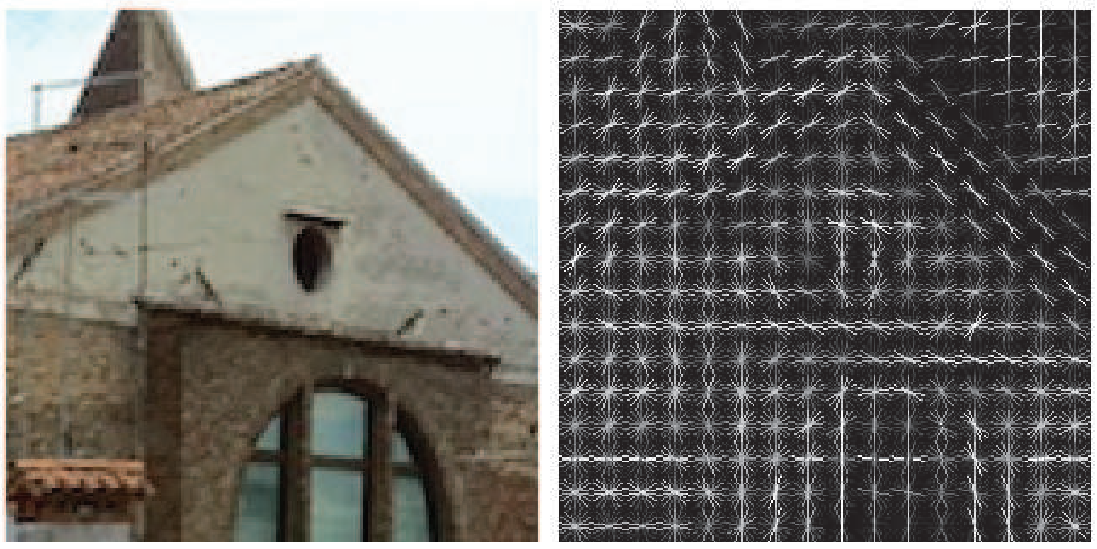

(a) House (b) Gradients

Abb. 3.2: Example of the HOG descriptor (images [17])

Abb. 3.3: A Common Visual Vocabulary, a model of the expected images and their features

3.2 Signatures

The Signature-based approach quantifies the descriptors of an image only in terms of the image content itself. The advantage of this approach is that it does not require any previously learned model or vocabulary and instead matches sets of image-specific words directly to each other. The problem consists of finding a good matching (Abb. 3.5), which wasn't the case in the BoVW approach.

4 Signature Distances

To deal with the previously mentioned problem of matching centroids computed from different images, there are several several good choices for distance measures. They all make use of a ground distance δ between descriptors to come to a number that quantifies the distance betwen both signatures.

The *Earth Mover's Distance* [7] is a *transformation-based* measure and expresses the work required to reshape one distribution (or histogram, or signature) into another, given a few constraints. It transforms the problem into a minimum cost flow problem that can be solved efficiently using appropriate graph algorithms.

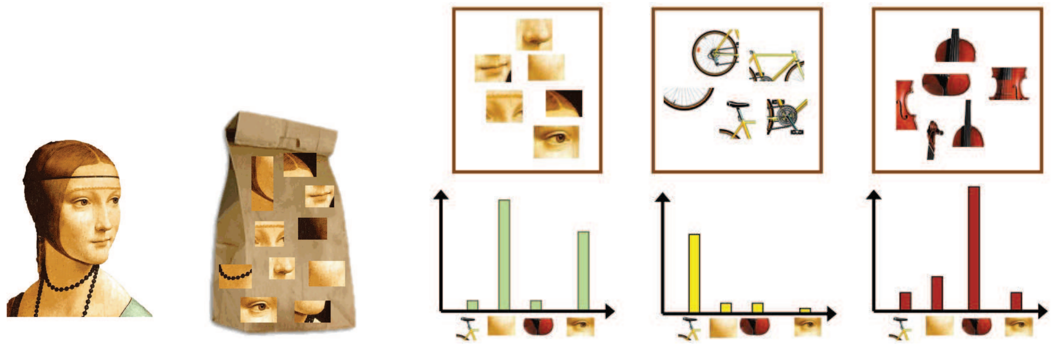
The *Perceptually Modified Hausdorff Distance* [15] is a *matching-based* approach and uses a weighting function proportional to $\frac{distance}{weight}$ similar to the matching strategy *Distance Weight Ratio* presented for the SMD.

The *Signature Quadratic Form Distance* [5] is a *correlation-based* approach.

5 Signature Matching Strategies

5.1 Nearest Neighbor

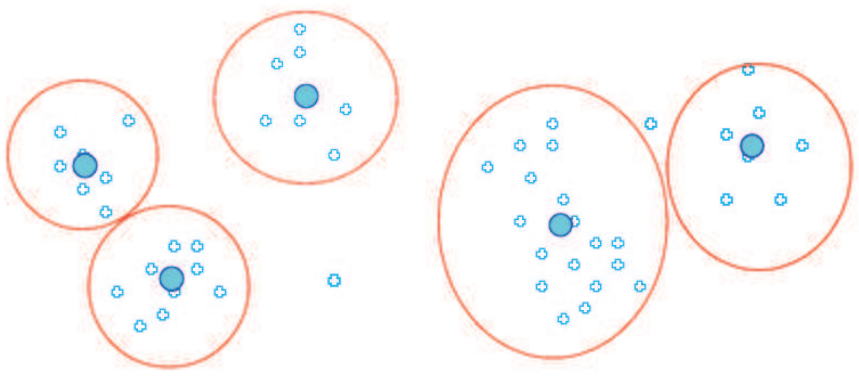
This matching algorithm matches every representative $x \in X$ to exactly one $y \in Y$, namely the nearest one. Thus, $n = |X|$ matches are guaranteed.



(a) an image and its set of descriptors (symbolized)

(b) images quantized to a common vocabulary

Abb. 3.4: The concept of Bag-of-Visual-Words



(a) one image

(b) another image

Abb. 3.5: Image-specific quantization (images [14]) yields centroids that are not trivially matched

5.2 Distance Ratio

This approach only matches x to y if there is no y' within a certain relative distance because y' could have been a candidate and the small variations in distance might just be noise that should not have an impact on the matching. We get $0..n$ matches.

5.3 Inverse Distance Ratio

The opposite approach to Distance Ratio matching. If y' is close enough, it too is matched. We get $n..2n$ matches.

5.4 Distance Weight Ratio

Here, potential matches are evaluated by their distance and inversely by their weight. A close but light representative will not be matched, but instead a slightly more distant but much heavier one. The idea is that close but light representatives might just be noise and should be ignored in favor of heavy centroids.

6 Cost Functions

A simple cost function to evaluate the quality of a matching would be $c(m) := |m|$, or simply counting the number of edges.

A better cost function would be the following. Here the sum of ground distances of the edges of a matching is weighted by the respective centroids of each edge. Close matches are good, hence cheap. Matches between heavy centroids are important, hence expensive unless also close together.

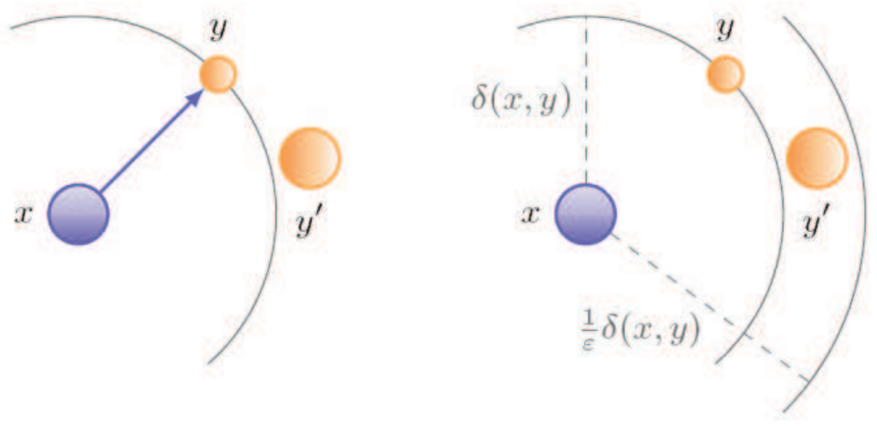


Abb. 3.6: Nearest Neighbor (left) and Distance Ratio (right) matching (images [4])

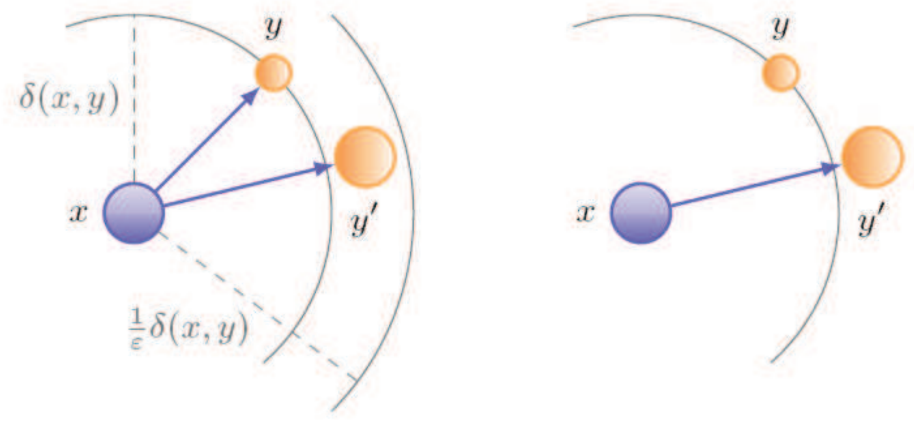


Abb. 3.7: Inverse Distance Ratio (left) and Distance Weight Ratio (right) matching (images [4])

$$c_\delta(m_{X\to Y}) := \sum_{(x,y)\in m_{X\to Y}} w_X(x)\cdot w_Y(y)\cdot \delta(x,y) \tag{3.1}$$

Another cost function uses instead of the plain ground distance the distance/weight ratio:

$$c_{\delta/w^*}(m_{X\to Y}) := \sum_{(x,y)\in m_{X\to Y}} w_X(x)\cdot w_Y(y)\cdot \delta/w^*(x,y) \tag{3.2}$$

7 The Signature Matching Distance

The Signature Matching Distance evaluates a given matching. If a matching contains many bidirectional matches, this is taken to mean that representatives matched each other well. This condition should not be penalized by counting the cost of such edges. A parameter λ discounts such bidirectional matches, such that their cost can be scaled from full to zero. Unidirectional matches suggest a bad fit and are counted in any case.

7.1 Definition

The SMD is a function of two signatures X, Y, a matching strategy m, a cost function c, and a parameter λ that expresses the exclusion of cost of bidirectional matches.

$$SMD(X,Y,m,c,\lambda) := c(m_{X\to Y}) + c(m_{X\leftarrow Y}) - 2\lambda\cdot c(m_{X\leftrightarrow Y}) \tag{3.1}$$

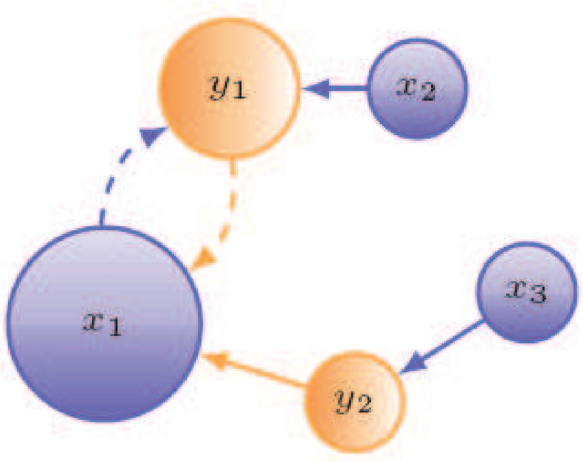


Abb. 3.8: A bidirectional match consists of two unidirectional matches. Hence the factor of 2.

Thus, for $\lambda = 0$, all matches are counted, whereas for $\lambda = 1$, only unidirectional matches count and bidirectional matches (which signify a good matching) cost nothing.

8 Discussion of Performance

8.1 Compared to Signature-based approaches

An evaluation of Mean Average Precision [4] over EMD, PMHD, SQFD, and SMD using the descriptors PCT, SIFT, CSIFT (colored SIFT) against two image databases showed that all distance measures performed best with PCT or CSIFT, and that SMD competed well with PMHD and even outperformed the Earth Mover's Distance.

8.2 Compared to BoVW-based approaches

Again MAP was evaluated on the same databases. Competitors were BoVW (from [16]) and some refined approaches of the original BoVW. BoVW was outperformed by 30 percentage points. SMD showed competitive numbers compared to all other approaches.

8.3 Tuning

Variation of λ and the ϵ parameter found in some matching strategies revealed that very little tweaking is needed. SMD performance increased up to a signature size of $k = 30$ representatives, then leveled off. Varying λ and ϵ improved MAP only by single percentage points.

8.4 Computational Efficiency

Timed computational performance tests showed that SMD is about as fast as PMHD and orders of magnitude faster than SQFD and EMD. One million distance computations took between $1.4s(k = 10)$ and $82.8s(k = 100)$ on a 3.4 GHz single-core machine. This execution speed allows for interactive applications and enters the realm of real-time.

9 Properties of a Metric

If the SMD had the properties of a metric, all the methods of Metric Indexing [10] could be applied to efficient database query and retrieval using the SMD.

Beecks, Kirchhoff, and Seidl suspected [4] that the SMD is not a metric. I have investigated their claim and given below are the results to confirm.

9.1 Assumptions

9.1.1 $\delta(x, y) \geq 0$ The ground distance is assumed to be a metric.

9.1.2 $w_X(x) \geq 0$ The weight of a centroid is non-negative because it was defined to be the size of the cluster it represents, i.e. the number of descriptors in this set.

9.1.3 $\lambda \in [0, 1]$ λ models the exclusion of bidirectional matches and must be in that range by definition.

9.1.4 $c(m) \geq 0$ The cost of a matching assesses the quality of that matching. All presented cost functions fulfill this.

9.1.5 $c(m_{X \leftrightarrow Y}) = c(m_{X \to Y}) = c(m_{Y \to X})$ The cost of an edge is the same, irrespective of its direction.

9.1.6 Matchings can be flipped The presented algorithms for generating matchings generate the same directed edges (with opposite direction) for input (Y, X) as they do for input (X, Y). This is clear because the presented algorithms may generate edges for each $x \in X$ and for each $y \in Y$, but independently of the result of the other stage.

9.2 Non-Negativity

$$SMD(x, y) \geq 0 \tag{3.1}$$
$$c(m_{X \to Y}) + c(m_{Y \to X}) - 2\lambda \cdot c(m_{X \leftrightarrow Y}) \geq 0 \tag{3.2}$$

W.l.o.g. ignoring unidirectional matches because those would only contribute positively to the left-hand size. Bidirectional matches are left. W.l.o.g. consider a matching $m' \subseteq m, |m'| = 1$ that contains a single bidirectional edge of cost $a = c(m'_{X \to Y}) = c(m'_{X \leftarrow Y}) = c(m'_{X \leftrightarrow Y})$. The distance reduces to:

$$a + a - 2\lambda \cdot a \geq 0 \tag{3.3}$$
$$2a(1 - \lambda) \geq 0 \tag{3.4}$$
$$1 - \lambda \geq 0 \tag{3.5}$$

The property of non-negativity holds for the SMD.

9.3 Identity of Indiscernibles

9.3.1 $x = y \implies d(x, y) = 0$ Signatures being equal ($x = y$) implies that both have the same number of centroids, in the same positions ($\delta = 0$), with the same weights w associated to them. A matching would match every descriptor in x to the same descriptor in y bidirectionally, each with zero cost, i.e. $c(m_{X \to Y}) = c(m_{X \leftarrow Y}) = c(m_{X \leftrightarrow Y}) = 0$. There are no unidirectional matches. W.l.o.g. consider $X' \subseteq X, |X'| = 1, m' \subseteq m, |m'| = 1$.

$$SMD(X', X') = c(m'_{X \to Y}) + c(m'_{X \leftarrow Y}) - 2\lambda \cdot c(m'_{X \leftrightarrow Y}) \tag{3.1}$$
$$= 0 + 0 - 2\lambda \cdot 0 \tag{3.2}$$
$$= 0 \tag{3.3}$$

The SMD of identical signatures is zero.

9.3.2 $d(x, y) = 0 \implies x = y$ This property does not hold for $\lambda = 1$. Consider a counterexample, where $SMD(X, Y) = 0$ but $X \neq Y$.

The centroids are not equal, but paired close enough to make the matching completely bidirectional. For $\lambda = 1$, bidirectional matches cost nothing. Thus, $SMD(X, Y) = 0$.

To fulfill this condition, choose $\lambda \lneq 1$ or take care to have unidirectional matches.

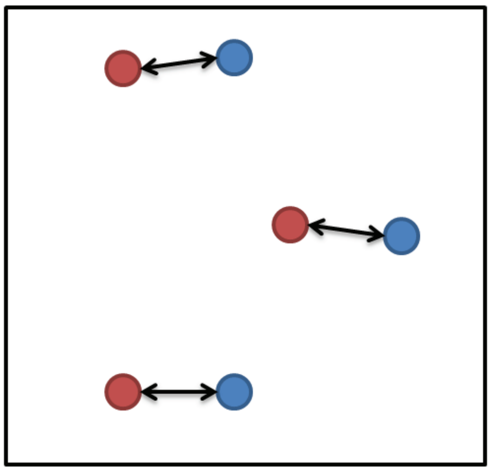


Abb. 3.9: X in red, Y in blue.

9.4 Symmetry

Assuming a matching algorithm that produces the same set of directed edges, but of opposite direction, when given signatures the other way around.

$$SMD(X,Y) = SMD(Y,X) \tag{3.1}$$
$$c(m_{X\to Y}) + c(m_{X\leftarrow Y}) - 2\lambda \cdot c(m_{X\leftrightarrow Y}) = c(m_{Y\to X}) + c(m_{Y\leftarrow X}) - 2\lambda \cdot c(m_{Y\leftrightarrow X}) \tag{3.2}$$
$$c(m_{X\leftrightarrow Y}) = c(m_{Y\leftrightarrow X}) \tag{3.3}$$
$$\tag{3.4}$$

The SMD satisfies symmetry.

9.5 Triangle Inequality

$$SMD(X,Y) = SMD(X,Z) + SMD(Z,Y) \tag{3.1}$$

This property can not be satisfied by the SMD. The following counterexamples are constructed such that $m_{X\leftrightarrow Z}$ and $m_{Z\leftrightarrow Y}$ yields bidirectional matches only ($SMD = 0$), but $m_{X\leftrightarrow Y}$ has unidirectional matches ($SMD > 0$). Some arrows are curved for illustration purposes only.

The counterexample for Distance Weight Ratio Matching chooses the weights and distances of the centroids adversarially such that both X centroids match the upper Y centroid; one match must remain unidirectional, thus costly.

A counterexample can also be constructed for Nearest Neighbor Matching and its derivatives Distance Ratio Matching and Inverse Distance Ratio Matching. The centroids are skewed such that for matching against Z in the middle, “corresponding” pairs are closer together than to other candidates of the signature, while matching for X and Y the skew starts to matter and provokes unidirectional matches as in the first example.

Such counterexamples work for $\lambda = 1$, but also for $\lambda < 1$ to some extent, until the adversarial gain from the above constructions is compensated. For $\lambda = 0$, Abb. 3.11 would indeed satisfy the triangle inequality; all edges have full cost, but the diagonal one (lower x to upper y) is shorter than going horizontally twice.

9.6 Conclusion

The SMD, for some λ, satisfies neither the Triangle Inequality nor the Identity of Discernibles.

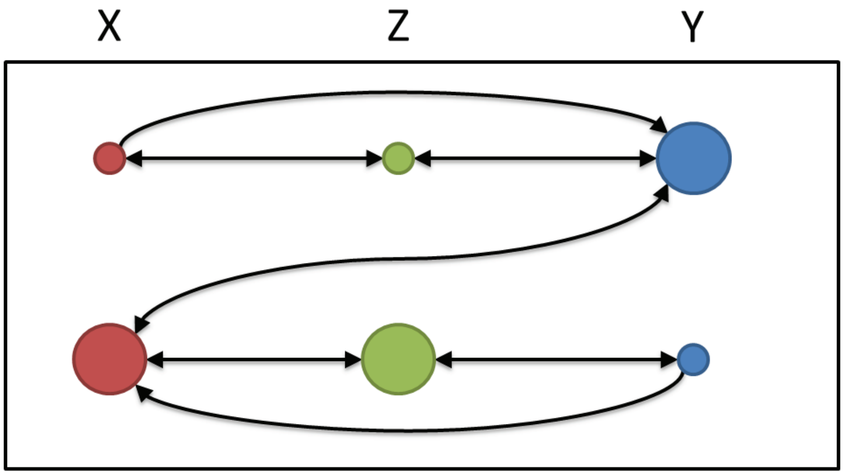


Abb. 3.10: Counterexample for Distance Weight Ratio Matching

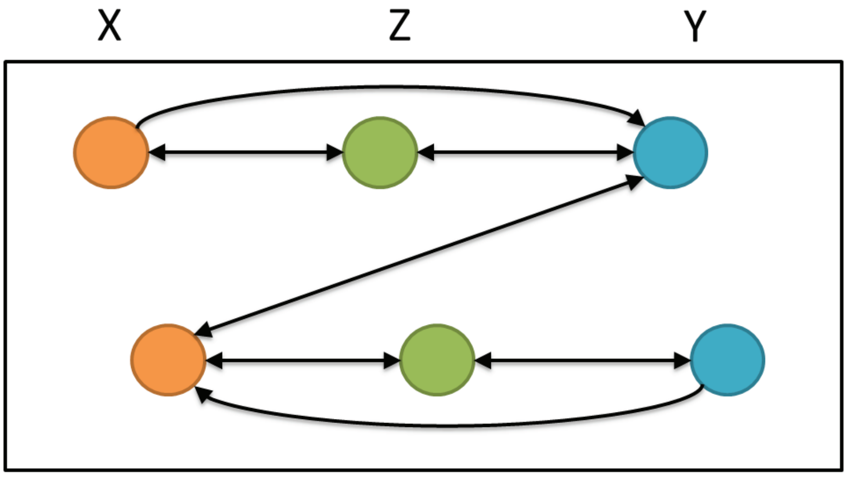


Abb. 3.11: Counterexample for Nearest Neighbor Matching

10 Algorithmic Considerations

A signature consists of $k = 10..100$ centroids which are the set of representatives (computed from the set of descriptors).

Computational effort to match a signature against one other image/signature:

$$\mathcal{O}(k \cdot \ln k) \tag{3.1}$$

This is the effort expected for k instances (for each representative in the first image) of nearest neighbor lookup using Binary Space Partitioning, which is a binary search algorithm. For our given k, this is close to constant and might practically even be turned into a few clock cycles of effort using vector instructions.

Due to the lack of common visual vocabulary, advanced database schemes such as the Vocabulary Tree [13] can not be leveraged for the SMD. A linear scan over n images results in this complexity:

$$\mathcal{O}(n \cdot k \ln k) \tag{3.2}$$

This might still be improved upon though using data structures specially suited to databases.

11 Conclusions

The SMD is model-free (requires no pretrained Common Visual Vocabulary) and matched or outperformed the state-of-the-art competition using a very simple feature descriptor in both precision and execution speed. It is simple yet effective. Unfortunately, it is not a metric and hence can not benefit from fast retrieval algorithms that make use of the property.

Literatur

[1] Datta R, Joshi D, Li J, Wang JZ. Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv., 40(2), 2008.

[2] Smeulders AWM, Worring M, Santini S, Gupta A, Jain R. Content-Based Image Retrieval at the End of the Early Years. IEEE TPAMI, 22(12):1349?1380, 2000.

[3] Sivic J, Zisserman A. Video Google: A Text Retrieval Approach to Object Matching in Videos. In ICCV, pages 1470?1477, 2003.

[4] Beecks C, Kirchhoff S, Seidl T. Signature Matching Distance for Content-based Image Retrieval. Proc. ICMR 2013, 2013.

[5] Beecks C, Uysal MS, Seidl T. A comparative study of similarity measures for content-based multimedia retrieval. In ICME 2010, p. 1552-1557.

[6] Rubner Y, Puzicha J, Tomasi C, Buhmann JM. Empirical Evaluation of Dissimilarity Measures for Color and Texture. CVIU, 84(1):25 ? 43, 2001.

[7] Rubner Y, Tomasi C, Guibas LJ. The Earth Mover?s Distance as a Metric for Image Retrieval. IJCV, 40(2):99?121, 2000.

[8] Beecks C, Uysal MS, Seidl T. Signature Quadratic Form Distance. In CIVR, pages 438?445, 2010.

[9] Tamura H. Texture features corresponding to visual perception. IEEE Trans. on Systems, Man, and Cybernetics, 8(6):460-473, 1978.

[10] Zezula P, Amato G, Dohnal V, Batko M. Similarity Search - The Metric Space Approach. Series: Advances in Database Systems, Springer, 2006.

[11] Leibe B. Computer Vision (Winter 14/15). Lecture slides, 2014 [cited 2015 Feb 28]. Available from: http://www.vision.rwth-aachen.de/

[12] Lowe DG. Distinctive Image Features from Scale-Invariant Keypoints. IJCV, 60(2):91?110, 2004.

[13] Nistér D, Stewénius H. Scalable recognition with a vocabulary tree. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 2161-2168, June 2006.

[14] Cluster Analysis and Diversity Analysis [Internet] 2015 [cited 2015 Feb 28]. Available from: http://icep.wikispaces.com/Cluster+Analysis+and+Diversity+Analysis

[15] Park BG, Lee KM, Lee SU. A New Similarity Measure for Random Signatures: Perceptually Modified Hausdorff Distance. In ACIVS, pages 990?1001, 2006.

[16] Jégou H, Douze M, Schmid C. Improving Bag-of-Features for Large Scale Image Search. IJCV, 87:316?336, 2010.

[17] VLFeat. Tutorials: HOG features [Internet] 2013 [cited 2015 Feb 28]. Available from: http://www.vlfeat.org/overview/hog.html

Mean Shift in Infinity Norm

Christopher Tenter

Zusammenfassung

This paper explains the theory behind the mean shift algorithm with focus on image segmentation. Basics of image processing and required background knowledge is also discussed. The mean shift is an iterative process which converges at some point. Usually, distances are measured in the euclidean norm when the mean shift operations are executed. However, the infinity norm can be used just as well and can affect the convergence rate.

Keywords: Mean Shift, Image Segmentation, Convergence, Infinity Norm

1 Introduction

Image segmentation is the process of grouping image regions based on their object affinity. Usually, image segmentation is part of an image processing pipeline that involves steps such as noise reduction, smoothing or sharpening, color correction. The segmented image can be further processed via artificial intelligence techniques, which detect objects in an image or track the position of an object in a sequence of animated images. It has medical applications, such as detecting tumors and defective cells, surgery simulation and any type of tissue analysis. There are various techniques available to perform image segmentation. The mean shift algorithm is one of them [1].

In fact, the mean shift algorithm has many other uses next to image segmentation. It is not only limited to images, but can be used for any type of real-valued data of any shape and dimension. The intuition of the mean shift is to find high density areas in a data set. The algorithm is an iterative process, meaning that the same operation is repeated several times. Such algorithms may not terminate with a good solution after a finite number of steps, if they do not converge to some optimal solution. This paper describes the ideas behind the mean shift algorithm and also takes a closer look at convergence. Furthermore, application of the infinity norm with the mean shift is explored [2]. An example of a segmented image is shown in Figure 4.1. The left image shows a building with grainy walls and the left image is the result after applying mean shift segmentation. It is a lot easier and more robust to perform object detection on the segmented image, as noise and fine details have been removed while object edges were left unchanged. Pixels belonging to the same region have the same color after segmentation.

2 Preliminaries

A basic understanding of signal processing is necessary in order to grasp the idea of mean shift. This paper mainly focuses on image segmentation, so mainly image processing concepts as discussed in [3] are

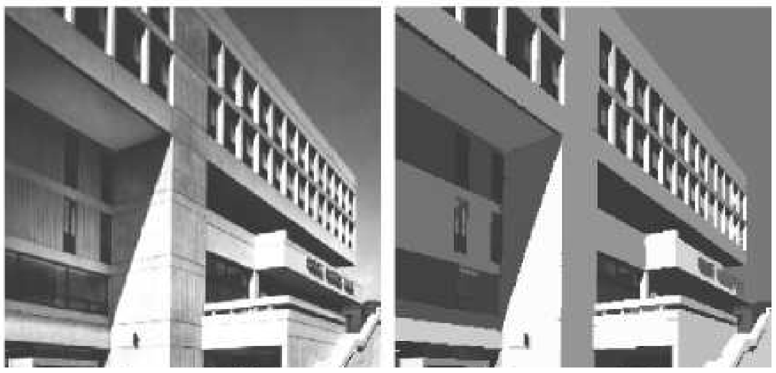


Abb. 4.1: Input and segmented image. Source: [1]

required for understanding the mean shift. An image is defined as a 2d matrix, which stores color values. A pixel is an entry within that matrix. Say I is an image represented by a matrix of type $\Re^{m \times n}$, then the pixel at row i and column j is a single real value that stores the luminance as color. Also, the image has width of m and height of n pixels.

$$I(i,j) = p_{ij} \in \Re$$

This can be extended to a color image by storing three real values instead of a single value for each pixel. That is, the color is stored as a triplet of real values and each component of that triplet is associated with one of three color channels: red, green and blue. The image is then a matrix of type $\Re^{3^{(m \times n)}}$.

$$I(i,j) = p_{ij} \in \Re^3$$

However, the mean shift algorithm operates on a set of samples instead on a matrix. Given an image $I \in \Re^{m \times n}$, the set of samples $X = \{x_1, x_2, ...\}$ has size $m \cdot n$ and is given as follows.

$$x_k = (i, j, p_{ij})^T \in \Re^3$$

where k is the index of the k'th pixel of the image, i and j are row and column of the k'th pixel and $p_{i,j}$ is the luminance of the k'th pixel. For colored images this becomes:

$$x_k = (i, j, p_{ij_r}, p_{ij_g}, p_{ij_b})^T \in \Re^5$$

where the sub-indices r, g and b indicate the red-, green- and blue color channels of the pixel. Furthermore, since pixel index (i,j) and pixel color are merged together in the sample set, this can be improved by scaling the position with a constant factor. Basically, the mean shift algorithm looks for density maxima in the sample set X. Position and color both influence the density of the sample set, so scaling one of them by a coefficient $s \in \Re$ gives control about the strength of influence in the common space.

$$x_k = (s \cdot i, s \cdot j, p_{ij_r}, p_{ij_g}, p_{ij_b})^T \in \Re^5$$

Images can be filtered, which means that a filter-operation is applied on the image [3]. This operator takes the input image, performs certain computations based on pixel colors and maybe also positions and finally returns the result of the computations as a new image. This new image is the filtered version of the input image. Blurring, sharpening and noise reduction are common image filtering operations. The mean shift filter is another filter in the sense, that it has an image as input and returns a segmented image as output. In the segmented image, pixels belonging to the same content or visual object ideally all have the same color. Additionally, pixels that do not belong to the same content ideally have different colors after image segmentation.

Pixels that are grouped together are said to form a cluster. To be more precise, a cluster is a collection of samples that are related or close together. That is, ares with high sample density form a cluster. In physics, density is a measure of mass in a normalized unit volume. This is similar in signal processing, where the number of sample points in a unit volume is a measure of mass. Areas with many sample points have a higher density, which is what the mean shift algorithm tries to find.

Unfortunately, this idea of density is not a very precise one. In fact, given a sample set as input, one can only guess the density of each region in the sample set. The intuition is that regions with lots of samples in one spot have high density, while regions with sparse samples have low density. This intuition is not very precise, but the mean shift algorithm has to actually measure a quantifiable value that describes density. Since the density is unknown, it has to be reconstructed based on assumptions and educated guesses. This is where a kernel function comes into play. A kernel is a function that is suited to build probability distribution functions (PDFs). A PDF is usually an accumulation of kernel functions. Thus, density can be modeled as a PDF based on kernels and evaluated at a random point in space [1].

3 Related Algorithms

There exists various techniques and methods to perform image segmentation on digital images. The goal of such an operation is always to cluster pixels belonging to the same region of a visual object together. Applications in medical image processing usually involve object detection such as diseases and tumors. Two methods other than mean shift are commonly used for segmenting images: k-means clustering and normalized cuts [3].

3.1 K-means Clustering

The basic idea of the k-means clustering clustering is rather intuitive. It is an optimization problem of the following form:

Given a set of n input samples $X = \{x_1, x_2, ..., x_n\}$, distribute these samples into k buckets such that the sum of internal variances within each bucket is minimal. Say there are k buckets and S_i is the set of samples that are in the i'th bucket. Then the variance within one of the buckets is given as

$$\sum_{x \in S_i} ||x - \mu_i||^2$$

where μ_i is the mean of the samples in S_i:

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

So the k-means clustering solution is the partition of sample points into the buckets S_i, such that the following term is minimal:

$$\sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \rightarrow min$$

In this case, variance is somewhat related to the sample density. If samples are wide apart, the variance is high. On the other hand, if samples are close to each other, variance is low. After partitioning, the samples are clustered into k buckets and each bucket represents a certain region of an object if the input was an image. This way, each pixel can be classified into one of k classes. Solving the k-means clustering problem is not an easy task unfortunately. There are heuristic algorithms such as Lloyd's algorithm, that try to find a good approximation to the optimal solution. Clearly, one problem with the k-means clustering approach is that the number of classes k has to be guessed by the user. The user might have to apply the k-means clustering filter possibly several times for various choices of k before getting a satisfying result. Also, implementing a k-means clustering solver is difficult.

3.2 Normalized Cuts

Similar to k-means clustering, the normalized cuts approach is also an optimization problem. It does not operate on a discrete sample set, but converts the input into a weighted graph instead. For an image for example, pixels are the nodes of the graph and nodes are connected via edges to each other. There is a weight assigned to each edge based on heuristics, which describe the similarity between two pixels. For example, pixels that have similar color and are close to each other, have high similarity. The nodes of the graph are then partitioned by computing normalized cuts, which is a modified version of the minimal cut problem.

Each cut partitions one subset of nodes into two subsets. Basically, this method recursively subdivides the partitions in the graph by cutting. Once one cut has been applied, it stays forever so partition borders do not move around. There must be a stopping criterion such as a threshold for the value of a normalized cut within a subset of nodes. In the end, the nodes of the graph are partitioned into several group and can be reinterpreted as pixels. The partition of the corresponding node is the classification of that pixel after segmentation.

This approach requires a graph cut solver algorithm, a heuristic weight function and a stopping criterion. In practice, solvers do not find the optimal solution for the normalized cut problem, but again approximate it for performance reasons.

4 Mean Shift Details

In contrast to the previous two methods, the mean shift algorithm does not solve a discrete partitioning problem. The result is certainly a partition, but the partition sets are implicitly computed by finding local maxima in the probability density function of the input samples [1]. Finding maxima of a real-valued function is also an optimization problem and the mean shift can be compared to the Newton's method for maxima searching. Newton's method tries to find maxima by taking the derivative at some initial point and stepping along it in domain space. This is repeated until convergence. The idea behind mean shift is similar and explained in more detail in the following sections.

4.1 Definition

The mean shift clustering is an iterative approach, where each iteration improves the previous state. The goal is to find local density maxima, so each iteration yields a point that is closer to the density maxima than the previous state. The mean shift algorithm works as follows: Given a sample set $X = \{x_1, ..., x_n\} \subset \Re^d$, a window size $h \in \Re$ and an initial window position $y_0 \in \Re^d$, try find the next window position y_1 by performing one mean shift operation.

$$y_1 = y_0 + u(y_0)$$

where $u(y) \in \Re^d$ is called the mean shift vector at position y. The main idea is to search for local density maxima. The search starts at some initial position y_0 and is limited to all local samples within a d-dimensional sphere of radius h around that position. This sphere is called window, since one mean shift operation only takes samples within that sphere into account. Figure 4.2 visualizes this state for the 2-dimensional case. The points in the sample set X are painted in red, the window is painted in blue and the mean shift vector is the yellow arrow. Intuitively, the mean shift vector tries to shift the window towards a region with higher density. In this example, samples are wide apart in the left area inside the window, while they are closer together in the right half. This also means that the density is lower in the left half, but higher in the right half. The mean shift vector should therefore point into the general direction towards the high density region to the right of the window.

A very simple approach to computing such a mean shift direction is to calculate the mean sample position within the current window. The mean point is attracted to high density areas. If there is a cluster of many samples that are close together in one area, then the mean point is drawn towards that cluster. Let $S(y)$ be the set of samples inside the window around point y with radius h:

$$S_y = \{x \in X | \, ||x - y|| \leq h\}$$

Then, the mean point is the average of all samples inside the window:

$$\frac{1}{|S(y)|} \sum_{x \in S(y)} x$$

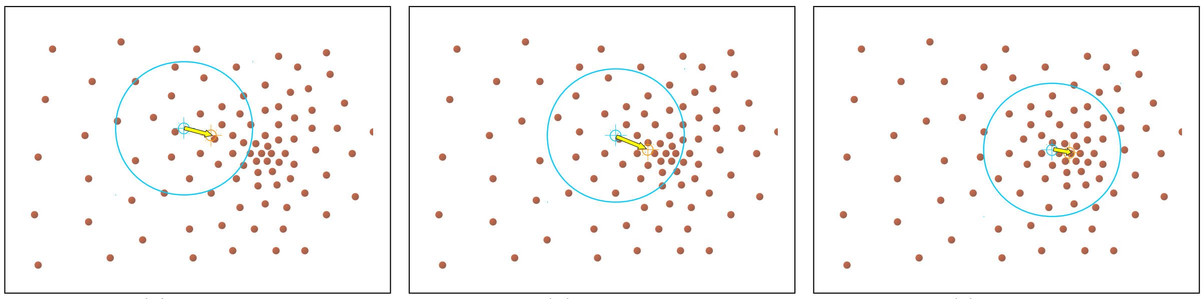

(a) Step 1 (b) Step 2 (c) Step 3

Abb. 4.2: Mean shift iterations. Source: [5]

and the mean shift vector is the difference between the window position and the mean point:

$$u(y) = \left(\frac{1}{|S(y)|} \sum_{x \in S(y)} x \right) - y$$

This can be easily computed and the only challenge in implementing this is to find the samples within a window around a random point in space. There are various methods to do such a neighborhood search of samples in an efficient manner. This is a basic problem of collision detection and common solutions are space-partitioning trees or spatial hashing [4].

After computing the mean shift vector $u(y)$, the window is shifted by that vector to a new position. Then, the process is repeated for the samples inside the window at the new location. So after k iterations, the new window position is given as:

$$y_{k+1} = y_k + u(y_k)$$

Ideally, this process is repeated until the mean shift vector reaches 0. Often, it will not be exactly 0 due to numerical issues or sample layout. A stopping criterion such as absolute or relative thresholds $\epsilon_{abs}, \epsilon_{rel}$ or a maximal number of total iterations k_{max} can be introduced to avoid infinite recursion.

$$\begin{gathered} ||u(y_k)|| \geq \epsilon_{abs} \\ ||u(y_{k+1})|| \geq \epsilon_{rel} \cdot ||u(y_k)|| \\ k \leq k_{max} \end{gathered}$$

If any of these constraints are violated, the search is stopped. In such a case, the window does not move a lot anymore since the mean shift vector has become small or the maximal number of iterations has been exceeded. The final location of the search is then said to be a local density maxima.

This is the most basic attempt of the mean shift algorithm. To recap, the main idea is that the mean can be used to find high density areas. Regions with sparse samples contribute far less to the mean point than regions with many samples that are tightly packed. This actually is a valid model for reconstructing the unknown density function in an implicit manner. Still, this can be improved by making use of kernel functions.

4.2 Kernels

Kernels were already mentioned in Section 2. They are basis functions, which can be used to construct a probability density function (PDF) for the input samples. Basically, it is assumed that the data points X were sampled from such a PDF. In that case, one can use the PDF to compute more accurate mean shift vectors. Unfortunately, the PDF is unknown and has to be guessed. There are several kernel functions to choose from. In any case, the shape of the kernel function should resemble the assumed density function around local maxima. Figure 4.3 shows the relation between PDF and data points. One can see two clusters in the data points in this example. One of these clusters has a higher density and this is also reflected in the PDF. There are two local maxima and one maxima is higher than the other. A more important fact is that the area around these maxima-peaks resemble the underlying kernel function of the PDF.

Common kernel functions are the Epanechnikov K_E, Gaussian K_G and uniform kernel K_U:

$$K_E(u) = \begin{cases} c(1 - ||u||^2), & \text{if } ||u|| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

$$K_G(u) = \begin{cases} c \cdot e^{-\frac{||u||^2}{2}}, & \text{if } ||u|| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

$$K_U(u) = \begin{cases} c, & \text{if } ||u|| \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

where $c \in \Re$ is the kernel normalization factor. This factor is different for each kernel and each dimension. In any case, it can be calculated by integrating the kernel term, since a normalized kernel has to satisfy the following constraint:

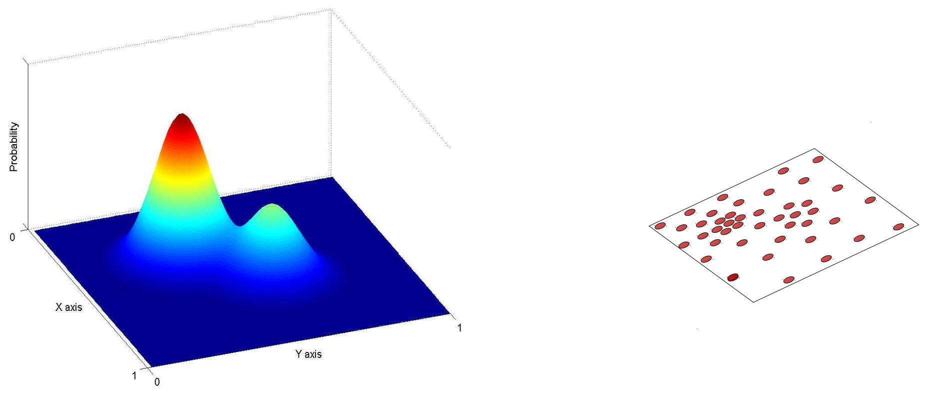
$$\int_{u \in \Re^d} K(u)du = 1$$

Kernels also only consider points in a unit radius. Points outside the unit radius are cut off. The reason behind this is the neighborhood search for samples within a window. Basically, the kernel should be applied to samples within a window only. Samples outside are not included to improve performance. After choosing a kernel function, the density of the center point y in a window of radius h can be estimated as follows:

$$f(y) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{y - x_i}{h}\right)$$

The kernel function K is centered around the origin and has a radius of 1, so the argument of the kernel has to be shifted and scaled accordingly. Intuitively, this estimate is maximal for high density regions. The kernel fits the region around maxima in the assumed PDF best. Also, since all kernels make use of the distance between the window position y and sample positions x_i, it gives a measure of density. Furthermore, this estimate can be interpreted as a weighted average computation. For instance, the uniform kernel assigns each sample the same constant weight. The result of using a uniform kernel is the same as the mean shift algorithm without kernels, as it was introduced earlier. Each sample had the same influence on the mean position, so all samples were equally weighted. Choosing a different function such as the Epanechnikov or Gaussian kernel introduces a non-uniform weighting scheme. Samples that are close to the window center y have a higher influence than samples that are farther away. This is a common concept in kernel based image filters. For instance, the Gaussian blur filter is basically a convolution of the image with the Gaussian kernel. Basically, the Gaussian filter computes a weighted average of the pixel colors in a circle around each pixel. Pixels close to the center are weighted higher.

So the mean shift filter can be improved with a kernel function in the same vein as many other image filters. Given the kernel function, it is possible to estimate the density around any point in space as $f(y)$. Since the goal is to find local density maxima, the mean shift algorithm tries to find the closest maxima in the density estimate f that is reachable from point y. The algorithm has already been shown



(a) PDF (b) Sampled data points

Abb. 4.3: Probability density function and sampling. Source [5]

for the uniform kernel, where the search direction is given by the mean point of samples in the search window around y. This is no longer the case for a general kernel function and a different approach has to be used instead. First of all, further restrictions have to be made to the kernel function. It has to be differentiable and non-negative. The non-negativity guarantees that the weights do not cause any trouble in the weighted average summation. Filter kernels are used for computing a weighted averages, so negative weights do not make any sense. The differentiable property on the other hand proves useful for the maxima search. From the theory of function analysis it is known that the gradient of a function is a vector pointing in the direction of steepest ascent in domain space. Here, $\nabla f(y)$ is the gradient of the density estimate evaluated at point $y \in \Re^d$ and points to the direction of steepest ascent in $\Re^d$. Stepping in that direction by some small amount δ and evaluating it again leads to an area with higher density:

$$f(y) \leq f(y + \delta \cdot \nabla f(y))$$

If $f(y)$ is already locally maximal, the gradient at that point becomes 0 and the equation still holds. The idea of the mean shift algorithm is exactly the same as stepping along gradients to find local density maxima. In fact, the mean shift vector at point y is the gradient of the assumed density function f and does not even have to be scaled by a factor δ.

$$u(y) = \nabla f(y)$$

The mean shift iteration after k steps then becomes:

$$\begin{aligned} y_{k+1} &= y_k + \nabla f(y_k) \\ y_{k+1} &= y_k + \frac{1}{nh^d} \sum_{i=1}^{n} \nabla K\left(\frac{y_k - x_i}{h}\right) \end{aligned}$$

Starting at window position y_k, the gradient of f at y_k is evaluated and used to shift the window to a new position y_{k+1}. So the used kernel function must be differentiable at all potential points $y \in \Re^d$. For analytically defined kernels such as the Gaussian, Epanechnikov or uniform kernels the gradient function can also be found analytically. Thus, the mean shift algorithm does not have to actually evaluate density estimates, but only gradients of the density estimates. If the kernel does not have an analytical definition, the gradient can still be evaluated by taking finite differences with some performance hit. All in all, the mean shift iteration is rather straight forward to implement. The only challenge is to find data points of the input samples X that are within the sphere of radius h around any random point in space y. This is an optimization step, because the kernel evaluates to 0 for samples x_i outside radius h. As already mentioned, there are collision detection algorithms which are suited to solve this problem efficiently.

So far the algorithm is independent of the used norm. In fact, the norm is only taken into account in the kernel function. First, the kernel usually makes use of the norm to compute a weight. Second, the kernel filters data points that are out of range. Data points are out of range if they are too far away from the kernel center, that is if the distance measured in some kind of norm exceeds radius h. It is intuitive to use the euclidean distance for all operations, however it is just as valid to use the infinity norm for example. The difference between these two norms for a vector $x \in \Re^d$ is as follows:

$$\begin{aligned} ||x||_2 &= \sqrt{x^T \cdot x} \\ ||x||_\infty &= \max_{i=1 \ldots d} |x_i| \end{aligned}$$

So the euclidean norm is equivalent to the euclidean distance of x to the origin, while the infinity norm is the maximal absolute component of x. The kernels in euclidean norm are cut off for samples outside of a unit-sphere in dimensions d. This actually has the geometry of a hyper-sphere in general dimensions. However, a unit-sphere in infinity norm does not have the geometry of a sphere anymore, but that of a unit-cube. This has to be taken into account in the collision detection implementation for finding neighboring data points around a random point in space. The neighborhood search has to find samples in a cube with sides $2h$ instead of a sphere with radius h. This can still be efficiently done however.

5 Convergence

The iterative mean shift algorithm for searching local density maxima does converge as shown in [2]. This section explains the proof in more detail and adds remarks for steps, that might not be as obvious. So the hypothesis to show is that applying the mean shift iterations repeatedly yields a sequence of window positions y_k, which converge to some local maxima in the density function f. This is shown for the euclidean norm first.

The proof is very much like any convergence proof for sequences in analysis theory. In this case, the sequence of density estimates $f(y_k)$ shall converge given the iteration step formula

$$y_{k+1} = y_k + \nabla f(y_k).$$

So, for a random starting point $y_0 \in \Re^d$, the sequence to examine is given as

$$F = \{f(y_k) | k \in \mathbb{N}\} \cup \{f(y_0)\}.$$

A sequence converges if it is bounded and strictly monotonic. Since the goal is to find local maxima, sequence F should have an upper bound and be strictly monotonic increasing. Sequence F is bounded, because there are only a finite number of input data points in X and each point $x \in \Re^d$ has finite components: $||x|| < \infty$. By definition of the PDF f, the sequence F has an upper bound:

$$f(y_k) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{y_k - x_i}{h}\right)$$

The sum and all atomic terms in the definition of f are finite. Clearly, the number of samples $n = |X|$, window radius h and dimension d are all greater than 0 and finite. To guarantee convergence, it is also important that the kernel maps to finite scalars. Theoretically, this was not required previously, but now it is to guarantee boundedness. Standard kernels naturally satisfy this condition.

So boundedness of F is given and monotonicity remains to be shown. The sequence F is monotonically increasing if the following relation is valid for all $k \in \mathbb{N}$:

$$f(y_k) \leq f(y_{k+1})$$

This means that the estimated density after $k+1$ iterations is at least as high as after k iterations. Actually, the sequence should be strictly monotonically increasing, but this detail is discussed later on. The remainder of the convergence proof depends on the used kernel. In fact, convergence has to be shown for each kernel individually, but in this section the Epanechnikov kernel is applied. So inserting the definition of the kernel into f and considering only those samples $S(y_k)$ within the sphere of radius h around y_k yields:

$$f(y_k) = \frac{c}{nh^d} \sum_{x_i \in S(y_k)} \left(1 - \frac{||y_k - x_i||^2}{h^2}\right).$$

In this formula, c is the kernel normalization factor and samples outside of the sphere $S(y_k)$ do not contribute to the sum, as the kernel evaluates to 0 for these points. Without loss of generality, y_k can be assumed to be located at the origin. One can just shift the PDF such that y_k is the origin and shifting the domain space of f has no effect on either boundedness or monotonicity. With $y_k = 0$, the equation simplifies to:

$$f(y_k) = \frac{c}{nh^d} \sum_{x_i \in S(y_k)} \left(1 - \frac{||x_i||^2}{h^2}\right).$$

y_{k+1} is the new location after applying a single mean shift operation to y_k and the density estimate $f(y_{k+1})$ can not be fully expressed in terms of samples from the previous window $S(y_k)$. But it is known that the two windows $S(y_k)$ and $S(y_{k+1})$ overlap. This observation proves useful in order to relate $f(y_k)$ and $f(y_{k+1})$ with each other. The definition of $f(y_{k+1})$ is given as:

$$f(y_{k+1}) = \frac{c}{nh^d} \sum_{x_i \in S(y_{k+1})} \left(1 - \frac{||y_{k+1} - x_i||^2}{h^2}\right).$$

The intersection of samples $S(y_k) \cap S(y_{k+1})$ is a subset of $S(y_{k+1})$. Thus, if only those samples in the intersection are considered, the sum becomes less or equal to the actual density estimate at $f(y_{k+1})$:

$$f(y_{k+1}) \geq \frac{c}{nh^d} \sum_{x_i \in S(y_k) \cap S(y_{k+1})} \left(1 - \frac{||y_{k+1} - x_i||^2}{h^2}\right).$$

Furthermore, the samples in the intersection of these two spheres are also a subset of $S(y_k)$, so it is possible to combine the two terms for $f(y_k)$ and $f(y_{k+1})$ by taking the difference. The difference $f(y_{k+1}) - f(y_k)$ should be positive for a monotonically increasing sequence.

$$f(y_{k+1}) - f(y_k) \geq \frac{c}{nh^d} \left(\sum_{x_i \in S(y_k) \cap S(y_{k+1})} \left(1 - \frac{||y_{k+1} - x_i||^2}{h^2}\right) - \sum_{x_i \in S(y_k)} \left(1 - \frac{||x_i||^2}{h^2}\right) \right)$$

The inner 1 in both sums evaluate to the number of samples in $S(y_k)$ and $S(y_k) \cap S(y_{k+1})$ respectively, so this can be simplified to:

$$\sum_{x_i \in S(y_k) \cap S(y_{k+1})} 1 - \sum_{x_i \in S(y_k)} 1 = |S(y_k) \cap S(y_{k+1})| - |S(y_k)| = -|S(y_k) \setminus S(y_{k+1})|.$$

$S(y_k)$ is the set of samples in the sphere around y_k, while $S(y_k) \cap S(y_{k+1})$ is a subset of samples in $S(y_k)$, which are also in the next sphere at position y_{k+1}. Thus, $S(y_k) \setminus S(y_{k+1})$ is the same set as $S(y_k) \setminus (S(y_k) \cap S(y_{k+1}))$. By taking care of the signs and factorizing with $\frac{1}{h^2}$ the relation becomes

$$f(y_{k+1}) - f(y_k) \geq \frac{c}{nh^{d+2}} \left(\sum_{x_i \in S(y_k)} ||x_i||^2 - \sum_{x_i \in S(y_k) \cap S(y_{k+1})} ||y_{k+1} - x_i||^2 - |S(y_k) \setminus S(y_{k+1})| h^2 \right).$$

The next step is to eliminate dependency of $S(y_{k+1})$ altogether in the sum of samples in the intersection of both windows. A simple observation regarding the distance of some samples in $S(y_k)$ to y_{k+1} helps with this. Samples in intersection $S(y_k) \cap S(y_{k+1})$ are inside both windows, but samples in $S(y_k) \setminus S(y_{k+1})$ are outside of $S(y_{k+1})$. This is shown in Figure 4.4 and mathematically described as

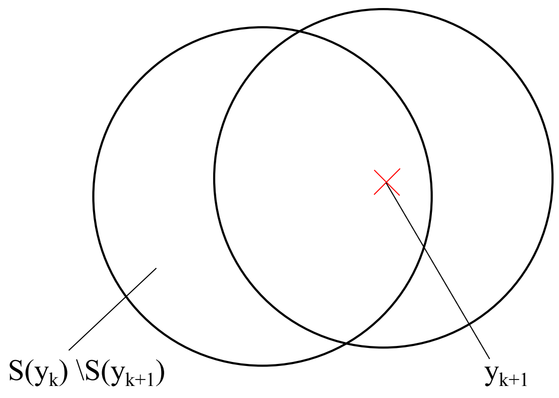


Abb. 4.4: Sample sets for two consecutive window positions at y_k and y_{k+1}.

$$||y_{k+1} - x_i||^2 \geq h^2, \forall x_i \in S(y_k) \setminus S(y_{k+1}),$$

because $S(y_{k+1})$ is the set of samples within a sphere of radius h around y_{k+1} and $S(y_k) \setminus S(y_{k+1})$ only contains samples outside of $S(y_{k+1})$. Thus, the distance of samples outside has to be at least h^2. With this observation, the last term can be expressed as a sum:

$$|S(y_k) \setminus S(y_{k+1})|h^2 = \sum_{x_i \in S(y_k) \setminus S(y_{k+1})} h^2 \leq \sum_{x_i \in S(y_k) \setminus S(y_{k+1})} ||y_{k+1} - x_i||^2.$$

Ideally, both sums are combined into one sum. This is now possible with simple set theory:

$$\begin{aligned}\sum_{x_i \in S(y_k)} ||y_{k+1} - x_i||^2 &= \sum_{x_i \in S(y_k) \cap S(y_{k+1})} ||y_{k+1} - x_i||^2 + \sum_{x_i \in S(y_k) \setminus S(y_{k+1})} ||y_{k+1} - x_i||^2 \\ &\geq \sum_{x_i \in S(y_k) \cap S(y_{k+1})} ||y_{k+1} - x_i||^2 + |S(y_k) \setminus S(y_{k+1})|h^2\end{aligned}$$

The last term is the same term as it appears on the right hand side of the relation for $f(y_{k+1}) - f(y_k)$. So this can be replaced as follows

$$\begin{aligned}f(y_{k+1}) - f(y_k) &\geq \frac{c}{nh^{d+2}} \left(\sum_{x_i \in S(y_k)} ||x_i||^2 - \sum_{x_i \in S(y_k)} ||y_{k+1} - x_i||^2 \right) \\ &= \frac{c}{nh^{d+2}} \sum_{x_i \in S(y_k)} \left(||x_i||^2 - ||y_{k+1} - x_i||^2 \right) \\ &= \frac{c}{nh^{d+2}} \sum_{x_i \in S(y_k)} \left(||x_i||^2 - (||y_{k+1}||^2 + ||x_i||^2 - 2y_{k+1}^T x_i) \right) \\ &= \frac{c}{nh^{d+2}} \sum_{x_i \in S(y_k)} \left(2y_{k+1}^T x_i - ||y_{k+1}||^2 \right) \\ &= \frac{c}{nh^{d+2}} \left(-||y_{k+1}||^2 |S(y_k)| + \sum_{x_i \in S(y_k)} 2y_{k+1}^T x_i \right) \\ &= \frac{c}{nh^{d+2}} \left(-||y_{k+1}||^2 |S(y_k)| + 2y_{k+1}^T \sum_{x_i \in S(y_k)} x_i \right)\end{aligned}$$

The last challenge is to evaluate the remaining sum. This sum over all samples in $S(y_k)$ computes the mean point of the samples in $S(y_k)$ scaled by $|S(y_k)|$:

$$\sum_{x_i \in S(y_k)} x_i = |S(y_k)|\mu_k,$$

where μ_k is the mean of samples in window $S(y_k)$. The authors in [2] now argue that this mean point is the next window position. Basically, the window at y_k is shifted to its mean point $y_{k+1} = \mu_k$. This would be correct if the employed kernel was uniform, but the proof made use of the Epanechnikov kernel instead. To be mathematically correct, it is actually necessary to compute the gradient of the density function at y_k as the mean shift vector instead of just taking the mean. The proof given in [2] may not be perfectly correct in that regard, but assuming the window is actually shifted to the mean point μ_k, the relation becomes

$$
\begin{aligned}
f(y_{k+1}) - f(y_k) &\geq \frac{c}{nh^{d+2}} \left(-||y_{k+1}||^2|S(y_k)| + 2y_{k+1}^T y_{k+1}|S(y_k)|\right) \\
&= \frac{c}{nh^{d+2}}|S(y_k)| \left(-||y_{k+1}||^2 + 2y_{k+1}^T y_{k+1}\right) \\
&= \frac{c}{nh^{d+2}}|S(y_k)| \left(-||y_{k+1}||^2 + 2||y_{k+1}||^2\right) \\
&= \frac{c}{nh^{d+2}}|S(y_k)| \cdot ||y_{k+1}||^2
\end{aligned}
$$

Finally, the last term contains only positive terms. To recap, c is the kernel profile, n is the number of total samples in X, h is the window radius and $|S(y_k)|$ is the number of samples inside the window around y_k. All of these scalars greater than 0. The last term $||y_{k+1}||^2$ is the magnitude of the mean shift vector, since y_k was assumed to be at the origin. This is greater than 0, if the mean shift vector is not 0. In the case where the mean shift vector becomes 0 the algorithm is also convergent, because the window does not move anymore and a local maxima has already been found in y_k. Since all terms are strictly positive, the sequence F is strictly monotonically increasing. Finally, F converges because it is also bounded by an upper limit. This means that the mean shift algorithm for searching local density maxima converges.

The proof has been executed in the euclidean norm so far, but it can be easily applied to the infinity norm. From the definition of the euclidean distance it follows that the infinity norm is less or equal to the euclidean norm for all vectors $x \in \Re^d$. The infinity norm is the absolute of one of the components of the vector. Say it is the i'th component $|x_i|$, then the squared euclidean norm is at least $|x_i|^2$. Furthermore, if the vector contains at least two non-zero components, then the infinity norm is strictly less than the euclidean norm.

$$
\begin{gathered}
x_1^2 + x_2^2 + ..x_d^2 = ||x||_2^2 \\
||x||_\infty \leq ||x||_2, \forall x \in \Re^d
\end{gathered}
$$

The difference of two consecutive density estimates in F measured in the infinity norm becomes

$$
\begin{aligned}
f(y_{k+1}) - f(y_k) &\geq \frac{c}{nh^{d+2}}|S(y_k)| \cdot ||y_{k+1}||_2^2 \\
&\geq \frac{c}{nh^{d+2}}|S(y_k)| \cdot ||y_{k+1}||_\infty^2
\end{aligned}
$$

Thus, the sequence F is also strictly monotonically increasing in the infinity norm. In conclusion, changing from the euclidean norm to infinity norm guarantees convergence as well. The next sections deal with applications of the mean shift for image segmentation.

6 Mean Shift Image Filtering

6.1 The Mean Shift Filter

The mean shift algorithm performs a search for local density maxima starting at some point y_0. This is useful for image segmentation, where the goal is to group pixels with high similarity and spatial proximity. The main idea now is to perform the maxima search for each pixel and assign the color of the local maxima to that pixel. That is, the color of each pixel in the filtered image is the color of its nearest density maxima. This is the reason, why image segmentation based on mean shift works. Similar pixels are attracted to the same density maxima and have the same color after filtering.

At first, the image has to be converted to a sample set $X = \{x_1, ..., x_n\}$, where n is the total number of pixels in the image. Say the k'th pixel is located at row i and column j of the image and has a certain color p_k, then the k'th sample is initialized to $x_k = (s \cdot i, s \cdot j, p_k)^T$. Here, s is a scaling factor to transform pixel indices i,j and color p_k into a common space, as already explained in Section 2. Depending on the number of color channels, x_k has either three or five components for gray-scale and colored images respectively.

The mean shift based density search is performed for each sample $x_k \in X$ individually until it converges. Then, the result of the search for sample x_k is a point $\hat{y_k} = (\hat{u}, \hat{v}, \hat{p_k})$ that locally maximizes the density estimate function. The color of the point at the density maxima is stored in the filtered output image:

$$I'(i, j) = \hat{p_k},$$

where I' is the filtered image, and i, j are row and column of the k'th pixel.

This algorithm can be improved by executing it repeatedly. In [2] the authors propose an algorithm that automatically detects, whether repeated filtering can improve the result.

6.2 Entropy-based Mean Shift Filtering

Repeatedly filtering the image with the mean shift filter can improve the result [2]. The reasoning is that applying the mean shift filter only once may not correctly group pixels if there are lots of small local maxima in the density function. Repeating the whole filter process might alleviate the problem of having lots of small clusters, that should actually be grouped together. If local density maxima are close together and not too different from each other, they might get grouped together in the next mean shift filtering. It is desirable to find an automatic algorithm, that detects whether filtering an image again is justified. That is, the algorithm should decide to stop filtering after k steps if the current image I_k and the filtered version I_{k+1} are not too different from each other. The authors in [2] suggest to use entropy as a measure of similarity between two images. Entropy is a measure of chaos in a data set. It is computed as follows for an image I:

$$E(I) = -\sum_{u \in \text{Colors}} p_I(u) \log_2(p_I(u)),$$

where $p_I(u) \in [0, 1]$ is the probability that a uniformly randomly chosen pixel has color u and Colors is the set of all colors in the image. The mapping p_I is also called the normalized image histogram. For color u, it can be computed by counting the number of pixels colored in u and dividing it by the total number of pixels in the image. The sum in the equation for entropy E is negated, since the logarithm on probability values in $[0, 1]$ is negative.

It is now possible to measure entropy before and after applying the mean shift filter on an image. If the difference of entropies falls below a threshold, the mean shift filter has little effect on the image and another repetition is unnecessary. The entropy-based mean shift image filter works as follows.

Given an input image I, a window size h and an entropy threshold ϵ_{ent}, compute a sequence of images $I_1, I_2, ...$ by recursively applying the mean shift filter on I_k, such that the entropy-based stopping criterion is satisfied:

$$|E(I_k) - E(I_{k+1})| > \epsilon_{\text{ent}}.$$

This sequence is initialized with the input image $I_1 = I$ and I_{k+1} is the mean shift filtered version of I_k. The sequence eventually stops after n iterations, so the final image I_n is returned as output.

7 Summary

The mean shift algorithm can be used for image segmentation. It does so by searching for local density maxima in an estimated probability distribution function of the input samples. A kernel can help to shape the PDF such that it resembles the distribution of samples of the given image. The algorithm is straight forward to implement and also easy to use. The user only has to choose a window size and a kernel function as the main parameters. It is possible to specify more parameters such as custom stopping criteria to prematurely stop the maxima search in favor of speed. The sequence of mean shift operations converges to some local density maxima if the operation is carried out in the euclidean norm and also in the infinity norm. Experiments presented in [2] show that using the infinity norm can improve or worsen performance, which seems to depend on the image content and window size. The infinity norm tends to give faster performance for larger window sizes, but this is also not guaranteed for every image. The downside of the mean shift filter is that a good window size is not an obvious choice for all images. It might be even beneficial to define a variable window size depending on the window position. Overall, the mean shift gives good results for image segmentation problems and is a valuable tool in an image-based object recognition pipeline.

References

[1] Comaniciu D, Meer P, Member S: Mean shift: A robust approach toward feature space analysis. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2002; 24(5): 603–619.

[2] Dominguez D, Rodriguez R: Convergence of the Mean Shift Using the Infinity Norm in Image Segmentation. Journal of Pattern Recognition Research, 2011; 6(1): 32–42.

[3] Forsyth DA, Ponce J: Computer Vision: A Modern Approach, 1. ed. Prentice Hall Professional Technical Reference, 2002.

[4] Ericson C: Real-Time Collision Detection, 1. ed. Morgan Kaufmann, 2004.

[5] Ukrainitz Y, Sarel B: Mean Shift: Theory and Applications. Lecture presented at; 2004; Weizmann Institute

Fine-Grained Image Classification with Convolutional Neural Networks

Ike Sebastian Kunze

Zusammenfassung

The subject of this paper is fine-grained image classification with convolutional neural networks (CNNs), i.e. distinguishing between very similar classes of images. In this context, the general concept of CNNs will be explained first, before a network by Krizhevsky et al. [1], the AlexNet, is presented, as a lot of other approaches build on the architecture of this network. After that, two part-based approaches, the Part-Based R-CNN by Zhang et al. [2] and Ensemble of Localized Learned Features (ELLF) by Krause et al. [3], will be discussed and compared to the AlexNet, as they use other additional techniques to further enhance the performance of the CNN. Finally, it will be analyzed to what extent convolutional neural networks are currently used in medical image processing.

Keywords: CNN, R-CNN, ELLF, Medical Image Processing

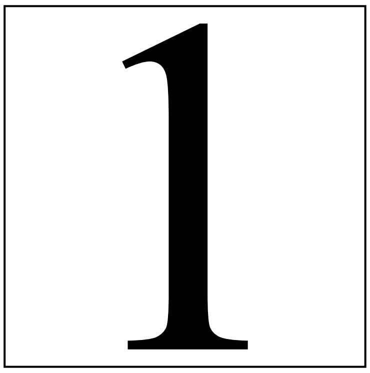
1 Introduction

Fine-grained image classification is a difficult task, although this might not be obvious. In general, human beings can easily differentiate between certain categories (coarse-grained classification), as well as between corresponding subcategories (fine-grained classification), as long as one is familiar with the respective topic. For example, it should not be a problem, even for laypersons, to first identify an arbitrary finger as a finger (coarse-grained) and then to distinguish between the thumb and the index finger (fine-grained), as the index finger is longer and leaner than the thumb. However, differentiation in applied medical context is not as easy as this, e.g. when it comes to the interpretation of computed tomography (CT) or magnetic resonance imaging (MRI) images. One example for this is the detection of breast cancer in MRI images as shown in Figure 5.1. As can be seen there, healthy and cancerous regions within breasts can have subtle differences, thus making it hard to distinguish between them. Modern imaging techniques already facilitate this process by preprocessing the images in a beneficial way, e.g. increasing contrast within the image or hiding unimportant body layers. Nonetheless, mistakes may happen, as some radiologists have better expertise or more experience than others, thus making fewer mistakes, and as humans tend to make mistakes now and then in general. However, it would be desirable to reliably detect breast cancer, i.e. without making any mistakes, as such mistakes can be dangerous for human life.

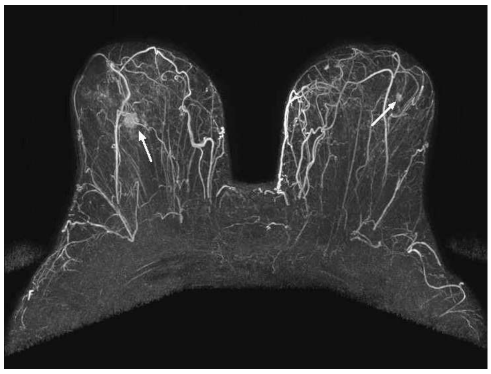
This is where Computer Aided Detection (CADe) methods, i.e. computer-based image processing techniques, can help to decrease the number of errors. However, computers have not been able to achieve results anywhere near human performance for a long time.

Yet, latest progress has made it possible that computers now can classify images quite successfully, i.e. with relatively low rates of error [1]. This recent success is partly explained by technical progress, which accelerates the computations of the processing techniques and improves the quality of imaging techniques, thus yielding images of higher quality that are more suitable to further processing by computers. Apart from these improvements of the technical foundations, image processing techniques themselves have been extended, too, so that the newly created technical possibilities can be utilized efficiently.

One line of theoretical ideas has built upon convolutional neural networks (CNNs), which were first proposed by LeCun et al. [6] in 1989 and shall be the topic of this paper. First, the basic concepts of convolutional neural networks will be reviewed, because a well-founded knowledge of them is inevitable to understand further approaches. After that, the AlexNet, a CNN by Krizhevsky et al. [1], will be presented, as the architecture of this network is used as template for a lot of recent state-of-the-art approaches. Building up on that, two part-based approaches will be shown, as they augment the basic strategy with additional techniques to further improve the performance of the network. Finally, the applicability of convolutional neural networks to a medical context and the extent of current usage will be analyzed.



(a) The MRI image on the left shows a normal breast, whereas the image on the right shows a breast with potential cancerous regions. Source: [4]



(b) The MRI image shows two breasts, each having one potential cancerous region (arrows). Source: [5]

Abb. 5.1: MRI images of breast cancer

2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a special type of feed-forward artificial neural networks, which are used to process and classify images. Every feed-forward network consists of processing units called neurons which are organized in one input layer, one output layer and an arbitrary number of layers in between (as shown in Figure 5.2).

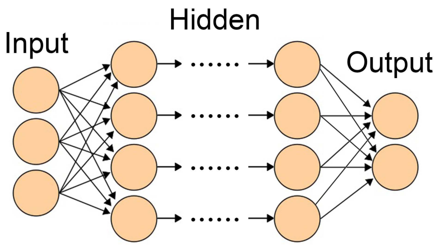


Abb. 5.2: This feed-forward neural network consists of one input, one output and several hidden layers.

Building up on these processing units, the networks compute a function on a presented input image to determine a corresponding classification. The function is adjusted during the training of the network in such a way that the computed classification becomes more and more equal to the correct classification, i.e. after some time, the training images should be classified correctly. After the training is completed, the function is fixed and, in the ideal case, suitable to correctly classify images that are not from the training dataset aswell.

As mentioned before, convolutional neural networks are one special type of feed-forward networks, which is why their concept will be described in more detail in the following. First, the processing units are characterized in Section 2.1, before different types of layers and their purposes are presented in Section 2.2. Last, the training of the network will be further elaborated in Section 2.3, i.e. it will be shown how networks can be adjusted properly.

2.1 Neurons

Neurons are the network's processing units typically having both incoming and outgoing connections to other neurons. They work on an input obtained from the incoming connections and complemented by an additional value, called bias. After applying a so-called activation function on the input, which is illustrated in Figure 5.3, the so generated output is passed along all outgoing connections.

In practice, non-linear functions are used as activation functions, which permits modeling of quite complex scenarios. Typical examples for such non-linearities are the saturating functions $f(x) = \frac{1}{1+e^{-x}}$

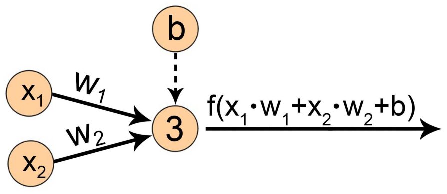


Abb. 5.3: Neuron 3 receives the outputs x_1 and x_2 of neurons 1 and 2 as input, weighted by w_1 and w_2, respectively. It then applies its function f on the weighted data, which is also complemented by the bias value and puts out the result along its outgoing connection.

(sigmoid function) and $f(x) = tanh(x)$ or the non-saturating function $f(x) = max(0, x)$, a very popular variant. Following Nair et al. [7], neurons with such an activation function are called Rectified Linear Units (ReLUs).

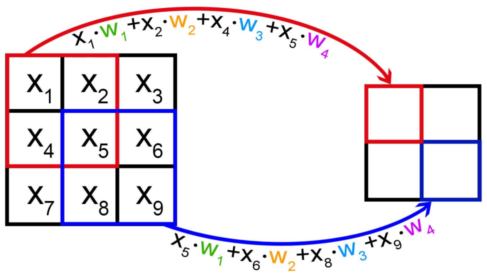
The interconnection between neurons of different layers is dependent on the layer they are in, as there are different types of layers with different properties. For example, in some layers the connections between neurons are weighted, like illustrated in Figure 5.3, whereas other layers do not make use of weights. Thus, the different types of layers shall be further explained in the now following Section 2.2.

2.2 Layers

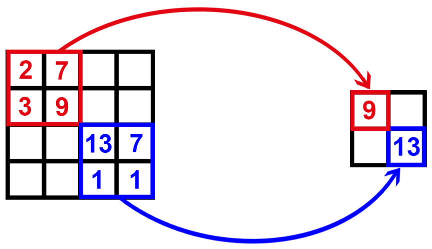
Layers are collections of neurons. There are different types of layers, which fulfill different tasks within a network. As already mentioned, the type of layer has influence on the neurons belonging to it, e.g. on the connections between the neurons.

One general constraint, which is valid for all neurons within a convolutional neural network, is that neurons of a layer $l - 1$ can only have outgoing connections to a layer l, which means that no layer can be skipped and that no connections to the same or a previous layer are allowed. This is reasoned by the fact that CNNs are feed-forward networks, i.e. data can only flow in one direction. Other constraints are highly dependent on the different types of layer, which are described hereafter.

The first type of layers are the convolutional layers. They are the most important ones in convolutional neural networks and they basically generate local descriptions of their input by using several different filters. For this, the neurons in such layers apply their standard activation function to overlapping areas of the input, which is shown in Figure 5.4 a). The influence of each part of the area, i.e. of each contained neuron, is hereby determined by a weight, which simply scales the output accordingly. Furthermore, the weights of connections between the same relative position within the overlapping areas and the neurons of the convolutional layer dealing with the respective area are shared, which is also depicted in Figure 5.4 a). For example, the output x_1 in the red area is scaled with weight w_1, which is also done for the output x_5 in the blue area. Both outputs share their relative position within their respective area, which is why their


$x_1 \cdot w_1 + x_2 \cdot w_2 + x_4 \cdot w_3 + x_5 \cdot w_4$
x_1 x_2 x_3
x_4 x_5 x_6
x_7 x_8 x_9
$x_5 \cdot w_1 + x_6 \cdot w_2 + x_8 \cdot w_3 + x_9 \cdot w_4$

(a) A convolutional layer of size 2×2 with 1 filter and filtersize of 2×2 is here applied to a 3×3 input to obtain a result of size $2 \times 2 \times 1$.



(b) A max-pooling layer with non-overlapping 2×2 pooling is applied to a 4×4 input. For example, the output of the top-left neuron would be 9.

Abb. 5.4: Examples for a) convolutional and b) max-pooling layers

influence on the processing neuron of the convolutional layer is also weighted with the same shared weight. One set of such shared weights can be interpreted as a filter applied to the whole input, generating local descriptions for each area while it is slided over the input. All local descriptions combined are then called a feature map. In most cases, convolutional layers do not only have one such filter, but several different ones, each of them having a different set of shared weights yielding different feature maps. Following the interpretation from before, one could say that convolutional layers apply multiple independent filters to generate multiple different local descriptions for the input. Typically, convolutional layers represent the first layers of a network, as they only generate local information. Before this local information is put together, which is done by fully-connected layers (as they can combine local information from all over their input), the information is often condensed to reduce its dimension.

This is done by another type of layer, the pooling layers, which simply summarize areas of the preceding layer in one neuron of their own. For this, the pooling neurons apply their pooling function, which is chosen from a variety of possiblities, to areas of the preceding layer. The connections, which are used for this, do not have any weights, as the pooling layers are not adjusted during the training of the network. One way of pooling is the so-called max-pooling, where the maximum value in an area is chosen as the representative for the whole area. This is shown in Figure 5.4 b) for non-overlapping pooling regions of size 2×2. In such non-overlapping regions, one neuron can only be part of one area, which is then summarized by one neuron of the pooling layer. However, pooling regions can also be overlapping, but this limits the resulting reduction of dimension. As stated before, a lot of convolutional layers are directly followed by a pooling layer to reduce the size of the generated feature maps, before the still local information is combined by fully-connected layers.

These fully-connected layers are perhaps the most basic layer type, as they are used in a variety of different neural networks. Neurons in such layers are connected to every neuron of the previous layer by weighted connections. In contrast to convolutional layers, there is no weight sharing in fully-connected layers. Still, such layer use the same activation function and they apply it in a similar way, as their input is also composed of the sum of all weighted incoming connections and the complementing bias term. As the neurons of these layers take the input from all the neurons of the preceding layer, fully-connected layers are well suitable to combine the local information generated in convolutional layers. Thus, it is not surprising that they are typically used as the last but one layers of convolutional neural networks. They are only followed by the final classification layer which uses their results to compute the final output vector of the network.

This vector has as many entries as there are different classes in the classification task, each of them containing the probability that the input belongs to the respective class. The vector is represented by neurons of the final layer, which again have weighted connections to the previous layer. The probabilities can be computed by different kinds of probability functions, one variant being the softmax function, a generalization of the logistic function. Layers with this probability function are called softmax layers and represent the most common classification layer.

One exemplary network is shown in Figure 5.5, where all the before mentioned aspects, except for the final softmax layer, are clearly visible, above all the convolutional layers followed by pooling (sub-sampling) layers.

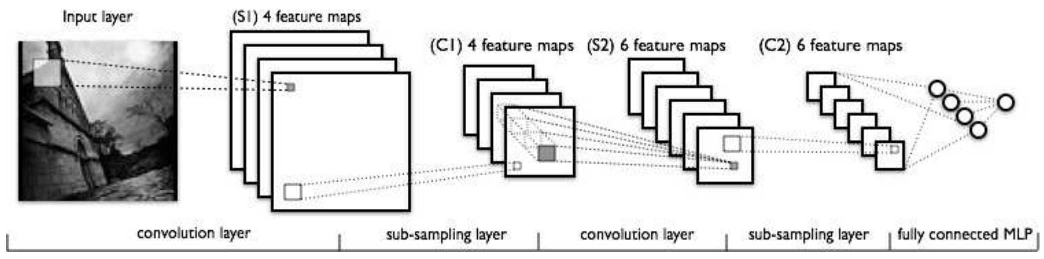


Abb. 5.5: This convolutional neural network consists of one input layer and two convolutional layers, which are followed by a pooling (sub-sampling) layer. Finally, there is a fully-connected layer and the final output layer. Source: [8].

Altogether, the network computes a chained function consisting of the functions of each layer to classify the input. As the weights of the neurons in the convolutional, fully-connected and classification layer can be adjusted, as well as the bias terms, there are a lot of parameters that can be tweaked to influence the overall output and thus to improve the obtained results, i.e. to achieve a correct classification. This is done during the training of the network, which will be looked at more closely in the now following Section 2.3.

2.3 Training Techniques

Convolutional neural networks are trained on image datasets by applying the network to the images and adjusting it to yield better results. The intention behind this is that unknown images can also be classified correctly, once the training of the network on the training dataset is completed. As stated before, the adjustment of the network is realized by tweaking the weights of connections between neurons and the values of bias terms and it is for example computed by a technique called stochastic gradient descent.

Stochastic gradient descent basically finds a new value for each weight so that the overall function moves into the direction of a minimal deviation between the network's output and the desired output, i.e. the correct classification, measured by an error function (e.g. the squared error loss function). The only requirement for the error function is derivability, as will become clear in the following.

The adjustment of the weights starts with those of the last layer. First, all partial derivatives of the error function for all weights of the last layer are computed. The so obtained gradient then indicates the direction of the largest decent of the error function, which means that it determines how the weights have to be adjusted to get closer to the minimal difference between computed and desired output. The weights are then adjusted accordingly by adding the gradient to the former values of the weights, which then moves the function in the direction of the minimal deviation. However, it might happen that the minimum is skipped by accident, if a too large step is used. For this, an additional parameter, the learning rate, is used which scales the gradient and thus determines how far the movement in the direction of the minimum shall be. Like this, the unwanted effect of skipping a minimum is treated and with suitable choices of the learning rate avoided.

After the weights of the last layer have been adjusted, the second to last layer is regarded next. This procedure is called backpropagation algorithm, which simply backpropagates the error from the final layer towards the input layer until all weights have been adjusted, i.e. the input layer is reached.

In contrast to standard gradient descent, which needs a complete run on the input dataset, stochastic gradient descent can be performed after an arbitrary number of images, which speeds up training for huge datasets. In the context of improved training techniques, pre-training and fine-tuning are two other important terms, above all when the intended training dataset is quite small. Pre-training means that the network is first trained on an additional dataset, which is (much) bigger than the (small) dataset which is supposed to be classified. Like this, the weights within the network can be trained with much more training examples. After the training on this additional dataset is completed, the final classification layer is exchanged with a new classifcation layer to fit the number of classes of the new dataset. The network is then fine-tuned on the desired dataset, which means that the network is trained as usual with the slight difference, that all weights but those of the classification layer are fixed, so that only those of the classification layer are adjusted to enable a correct classification.

Summarizing the characteristics of convolutional neural networks, one can say that they combine local descriptions generated all over the input to compute probabilities for the membership of the input in one or another class, thus classifying the input. The obtained classification is adjusted during the training of the network to improve the results, so that in the end all training images and a lot of other images belonging to the classes of the dataset can be classified correctly, atleast in the ideal case. Although this adjustment is the main strength of CNNs, it is also their main weakness, as it can happen that weights are adjusted in such a way that only the training images can be recognized correctly whereas other, potentially similar images might be classified into completely wrong classes. This is called overfitting and it is mostly the case when the used dataset is too small for the size of the used network, i.e. if there are too many parameters that need to be adjusted for too few images. Thus, it is difficult to find a suitable network architecture and suitable accompanying methods, as wrong choices, e.g. a bad or too big network architecture, can prevent good results. Still, there are a lot of successful architectures available, which will be presented in the following sections.

3 Image classification with convolutional neural networks

Recently, research has seized on convolutional neural networks using them to exceed previous results in fine-grained classification by far so that convolutional neural networks can now be considered as state of the art in a lot of areas of application, like face detection or speech recognition. A paper by Alex

Krizhevsky et al. [1], who have developed an efficient two-GPU implementation for large convolutional neural networks, has played an outstanding role in this process. The authors use their network, often called 'AlexNet', to beat previous state of the art by classifying 150,000 images of the ImageNet dataset into 1000 categories for the ImageNet Large-Scale Visual Recognition Challenge (LSVRC) [9]. Thus, the 'AlexNet' can now be regarded as state of the art, which is why it serves as principle for a lot of current approaches to image classification. Both, its architecture and how it is trained, shall be described in the following.

3.1 Network Architecture

Krizhevsky et al.'s network, the AlexNet, basically consists of 5 convolutional layers, some followed by max-pooling layers with overlapping pooling regions (layers 1,2,5), 2 fully-connected layers and a final 1000-way softmax layer. The whole network is spread across two GPUs, which is achieved by splitting up every layer into two parts of equal size and putting them on different GPUs (architecture as depicted in Figure 5.6). Like this, a higher efficiency for larger datasets can be achieved, as multiple computations can be performed simultaneously, although less efficient GPU-to-GPU communication becomes necessary due to this architecture, aswell. However, this negative aspect is limited by another trick, as the inefficient GPU-to-GPU communication is only needed in some of the layers (convolutional layer 3, the fully-connected layers and the softmax layer), whereas in the other layers the neurons only take input from neurons that reside on the same GPU (convolutional layers 2,4,5; see Figure 5.6.

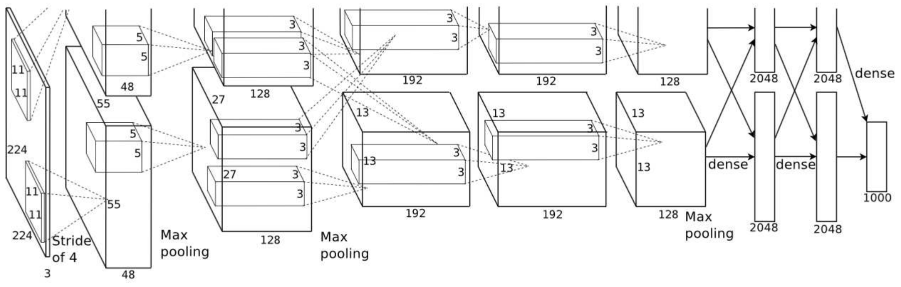


Abb. 5.6: Schematic illustration of the structure of the CNN. Source: [1]

The network's neurons are modeled as ReLUs, because the authors believe networks with such neurons train several times faster than networks with saturating neurons. This choice has contributed to the fact that ReLUs can be seen as the current standard activation function. Furthermore, the activity of neurons of the second and third convolutional layers is adjusted by Local Response Normalization (LRN), a technique that scales the activity of a neuron in relation to the activities of a certain number of other filters applied on the same spatial position. In particular, n filters inhibit the activity evoked by a filter i, which is similar to a behavior found with real neurons called 'lateral inhibition', where neurons inhibit each other to intensify contrast between them. Still, it is doubtful if this technique is really necessary, as there is no other approach which also uses it.

3.2 Training

The AlexNet is trained with standard training techniques for convolutional neural networks, i.e. with stochastic gradient descent and backpropagation. The whole training process is designed such that overfitting is reduced as much as possible. This becomes necessary as the AlexNet consists of 650,000 neurons and 60 million parameters, which need to be adjusted during the training.

One thing that is done against overfitting is an artifical augmentation of the training dataset, i.e. the network can be trained on a greater amount of images. The augmentation is performed by first altering the color of the original image using a random variable, which yields different results each time the image is used as the input. After that, the so altered image is resized to a fixed resolution and 1024 random patches are extracted. The network is finally trained on those 1024 patches and their horizontal reflections, totalling in 2048 overall training images generated from one single image in one iteration of the training process (mostly, one image is used as the input more than once, each time yielding different 2048 patches). At test time however, only 5 patches are extracted, namely the 4 corner patches and the center patch, and used as network input along with their reflections. The network's result on these 10 patches is then averaged to obtain the classification of the original image.

Another technique to avoid overfitting is called Dropout [10], which literally drops a hidden neuron out of the architecture of the network for one training image by a 50% chance, i.e. it does neither take part in the forward feeding (as its output is set to 0), nor in the backpropagation for this image. Thus, the network's architecture looks different for every training input presented which prevents neurons from relying on their neighbors too much and forces them to learn more robust features. During the training of the AlexNet, dropout is used in the first two fully-connected layers, which makes it neccessary that the out of these neurons is multiplied by 0.5 at test time to keep the results in a manageable dimension.

3.3 Results

Krizhevsky et al. performed several tests on the AlexNet, achieving record breaking results on the ILSVRC 2009 and 2010 test datasets and winning the 2012 competition. Top-1 error rates (the percentage of images, which are not classified correctly) and top-5 error rates (the percentage of images, for which the top-5 computed results of the network do not contain the correct classification) of the AlexNet and of the next best approaches are shown in Table 1 and will be presented in the following.

In the 2009 edition, top-1 and top-5 error rates of 67.4% and 40.9% are achieved, compared to 78.1% and 60.9% reported by the next best approach. Still, there is no established test dataset for this competition, which is why the much improved error rates might also be explained by the differences between the test datasets (Krizhevsky et al. simply use a random half of the dataset for training and the other half for testing)

ILSVRC-2009

Model	Top-1	Top-5
Previous best	78.1%	60.9%
AlexNet	67.4%	40.9%

ILSVRC-2010

Model	Top-1	Top-5
Previous best	45.7%	25.7%
AlexNet	37.5%	17.0%

ILSVRC-2012

Model	Top-1	Top-5
Next best	—	26.2%
AlexNet	36.7%	15.3%

Tab. 1: Comparison between the results achieved by AlexNet and previous or next best results in the ILSVRC-2009, 2010 & 2012 competitions.

In contrast to that there is a fixed dataset for the 2010 edition in which top-1 and top-5 error rates of 37.5% and 17.0% beat previous results (45.7% and 25.7%) by significant margins. Here, the AlexNet can clearly be seen as an improvement. This opinion can be substantiated by the results of the 2012 competition, where Krizhevsky et al. have actually participated and won with the AlexNet, beating the runner-up with a top-5 error rate of 15.3% compared to 26.2%.

Thus, it can be claimed that the AlexNet has set a new standard in the field of fine-grained image classification, atleast when only talking about the final classification accuracy. This is why a lot of other approaches have build on the architecture of Krizhevsky et al. One group of ideas has focused on part-based approaches, i.e. applying the network to selected parts of images. The ideas behind this will be looked at more closely in Section 4.

4 Part-based approaches to image classification

Part-based approaches try to further enhance the performance of a single convolutional neural network by using it in combination with additional region proposal algorithms, which propose certain regions of the input image with the intention of restricting the data, which is processed by the network. Ideally, this reduces the input data to the most useful one, i.e. those regions which are most suitable for discrimination. Doing so, unnecessary information, like background noise, is dropped as this is not needed for the correct classification of the image. Furthermore, part-based approaches aim at comparing the appearance of certain parts of the displayed object, rather than the appearance of certain regions within the image.

Both basic ideas can be realized in a lot of different ways, as will be shown with the help of two supplementary papers.

4.1 Part-Based R-CNNs

Ning Zhang et al. [2] propose a generalization of a method by Girshick et al., called Region-CNN [11], to classify 6000 bird images of the Caltech UCSD bird dataset [12] into 200 different breeds. In Region-CNN, images are first put through a region proposal algorithm, where certain regions of the image are extracted. These regions are then fed to a CNN to generate appropriate descriptors, which are finally used to classify the image with a linear support vector machine (SVM). A (linear) SVM divides two sets of points (representing descriptors belonging to two different classes) by a (linear) function while maximizing the distance between the points located near the border on each side, as depicted in Figure 5.7. Like the network, the SVM function is trained with the descriptors of training images and fixed at test time, where a point belongs to one of the classes, if it lies on the respective side of the border. For the actual classification, the authors one one-versus-all SVM for each occurring class, i.e. SVMs which decide whether a descriptor belongs to a certain class or not.

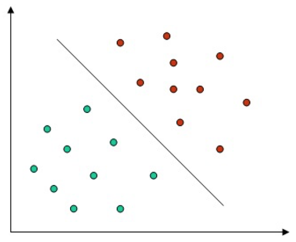


Abb. 5.7: The SVM divides the set of points in two subsets, maximizing the distance between the points located near the border. Source: [13].

4.1.1 General Structure The basic idea of Zhang et al.'s Part-Based R-CNN approach is the same like in the original Region-CNN method. Parts of the images, which are suitable to distinguish between different objects, are first proposed by a method called Selective Search [14], which uses a form of over-segmentation to propose regions, i.e. the images are first divided into a huge number of very small parts and then put back together step by step to obtain a desired amount of regions. The so proposed regions are then processed by a convolutional neural network, which uses the network architecture of the original AlexNet (see Figure 5.6), but a different implementation (Caffe [15]). After that, the computed descriptors are fed to a collection of one-versus-all SVMs. However, Zhang et al. do not only have SVMs for each class, but also for a certain number of different parts of the objects, e.g. the bird's head or body, which enable them to further perform part detection. They use this additional information to generate a representation for the object, which is based on the relation between the different parts and their respective appearance. This representation is finally used to classify the input with another SVM.

In the following, the training of the overall system will be explained first, before details of the classification process will be pointed out.

4.1.2 Training In comparison to the training of the AlexNet, the training of the Part-Based R-CNN method seems quite expensive, as there are a lot of different components that need training. However, each component can be easily trained as will be shown in the following.

The convolutional neural network is pre-trained on the ImageNet dataset and fine-tuned on the desired bird dataset afterwards, as the bird dataset is quite small compared to the ImageNet dataset (6,000 images compared to 1.2 million images). For the training of the network, Zhang et al. use the same standard techniques like Krizhevsky et al. At test time, the network architecture only differs in the final 200-way softmax layer which replaces the original 1000-way version to fit to the 200 bird breeds contained in the dataset. As the pre-trained AlexNet is publicly available, i.e. the weights of each connection, the only time in this part is spend on the fine-tuning of the network.

In contrast to that, the SVMs need a complete training. The authors do not only use one SVM for each breed of birds, but also one for each part of breed, e.g. one for the head. As the bounding boxes for the birds and a fixed number of their parts is annotated in the training data, the authors use feature descriptors of proposed regions with more than 0.7 overlap with the bounding box of a certain object or part as positive examples for the respective SVM, and regions with less than 0.3 overlap as negative examples.

The final classification SVM is trained on a pose normalized object representation, which uses the geometric relation between the detected objects and parts to filter out false detections. As this object representation is quite complex, it will be explained in the following Section 4.1.3.

4.1.3 Pose normalized object representation The pose normalized object representation is basically achieved by translating intuitive properties of correct detections into formulas.

For this, the locations of an object p_0 and its n parts $\{p_1, p_2, \cdots, p_n\}$, i.e. their bounding boxes, are denoted by $X = \{x_0, x_1, \cdots, x_n\}$, with corresponding SVM scores $\{d_0, d_1, \cdots, d_n\}$, $d_i(x)$ being the score for a feature descriptor generated at location x and put into the SVM for part i.

The first property is that the highest scored detections ought to be the right ones. This is achieved by using Equation (5.1) without $\Delta(X)$, as this simply defines an optimization problem to find that combination of proposed regions, which maximizes the product of all object and part detector scores. The solution for this is simply the combination of the highest scored regions for each object and part detector.

$$X^* = \underset{x}{\operatorname{argmax}} \left(\Delta(X) \prod_{i=0}^{n} d_i(x_i) \right) \tag{5.1}$$

However, the detectors are not perfect, which makes false detections possible. Thus, the authors use Equation (5.1) with $\Delta(X)$, as this function is designed to establish a relation between the object and part locations to filter out false detections. They experiment with several different definitions for $\Delta(X)$, each of them representing one intuitive constraint to the correct detections.

The first one, which the authors call box constraints, simply filters out those combinations, where parts are detected outside of the bounding box of the overall object, i.e. when not all parts p_1 to p_n are located inside the bounding box x_0 of object p_0. For this, they define $\Delta(X)$ as

$$\Delta_{box}(X) = \prod_{i=1}^{n} c_{x_0}(x_i). \tag{5.2}$$

Hereby, $c_{x_0}(x_i)$ is defined such that part locations x_i, which fall outside of the object's location x_0 by more than a fixed tolerance are considered to be not valid ($c_{x_0}(x_i) = 0$), whereas part locations within this tolerance limit are considered as valid ($c_{x_i}(x_i) = 1$). Thus, this definition of the scoring function $\Delta(X)$ fulfills the primary task and eliminates all detections, where certain parts are detected outside of the overall object. Like this, the simple combination of the highest scored regions is not always possible anymore.

Still, the geometric relation between the parts is not regarded by the box constraints, which allows unwanted constellations, e.g. detections of a bird's head close to the bird's feet. To avoid this, the box constraints are extended to filter out incorrect detections by enforcing constraints over the relative position between parts and their object. They define their geometric constraints as

$$\Delta_{geometric}(X) = \Delta_{box}(X) \left(\prod_{i=1}^{n} \delta_i(x_i) \right)^{\alpha}, \tag{5.3}$$

where δ_i is a function that fulfills the task of scoring the relative position of part p_i using information learned from the training data and α is an adjustable parameter. For δ, the authors propose two different definitions.

$\delta_i^{MG}(x_i)$ fits a Gaussian mixture model with 4 components to locations of part p_i learned from the training data. In other words, the location x_i of a detection for part p_i is scored by the probability distribution for the location of part p_i in the images of the training data. Thus, this definition tries to create a general model describing all possible shapes of an object category. However, this might be inaccurate, if the objects appear in different poses, which is not captured sufficiently. Just think of a data set in which a lot of birds stand upright and only few have their heads near the ground (which is also a normal pose for birds). Then the model would state that it is quite unlikely that the head of a bird is located below the body, thus scoring object and part candidates in this pose with a lesser score than candidates similar to an upright pose. Like this, detections might be discarded, although they were the correct ones, just in a pose, which is underrepresented in the dataset.

$\delta_i^{NP}(x_i)$ is an adaption of $\delta_i^{MG}(x_i)$ and tries to eliminate the described mistake. Here, the Gaussian model is not fit to the whole training data, but to 20 images, which are nearest neighbors to the top-scoring window of the object detector. This means that the location x_i of the detection for part p_i is scored by the position of part p_i in 20 images, which show the highest resemblance to the whole object detection. In contrast to the model created by $\delta_i^{MG}(x_i)$, the $\delta_i^{NP}(x_i)$ model does not describe all possible shapes of an object, but only one very specific shape, thus minding the scenario described above, as well.

The authors reportedly use both of the definitions of δ, achieving results with only slight differences. The latter one is illustrated in Figure 5.8

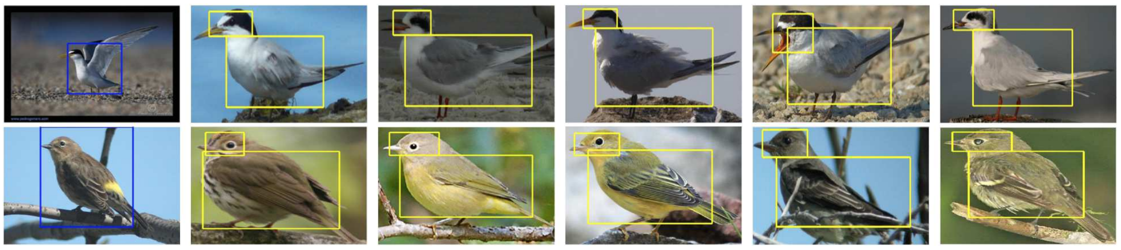


Abb. 5.8: The birds in the first column are the respective test images with bounding boxes determined by the approach of Zhang et al., the 5 other birds in each row are the top-5 nearest neighbors used in δ^{NP}. Source: [2]

As soon as the optimization problem is solved, the feature descriptors of all regions, which are used in the optimal constellation, are concatenated and used as the input for a final SVM. Thus, the final classification is performed using the pose-normalized representation created before, which makes a more stable classification possible compared to a representation that does not take different poses into account. Apart from the mere classification of an object, the authors can also use their approach to locate parts, as this is an elementary component of their system. Results for both, classification and part localization, will be shown in Section 4.1.4.

4.1.4 Results Zhang et al. use their system to classify images of the Caltech-UCSD bird dataset into the 200 different bird species contained in the dataset. They conduct several experiments in a setting where the bounding boxes of the objects and parts are unknown at test time. The classification and part localization results achieved by the different versions of the scoring function will be reported in Table 2.

Classification with bounding box unknown

Model	Accuracy
Previous best	44.94%
Δ_{null}	64.57%
Δ_{box}	65.22%
$\Delta_{geometric}$ (δ^{MG})	65.98%
$\Delta_{geometric}$ (δ^{NP})	65.96%
Δ_{box} (finetuned)	72.73%
$\Delta_{geometric}$ (δ^{MG}) (finetuned)	72.95%
$\Delta_{geometric}$ (δ^{NP}) (finetuned)	73.89%

Part localization accuracy

Model	Head	Body
Previous best	37.44%	47.08%
Δ_{null}	60.50%	64.43%
Δ_{box}	60.56%	65.31%
$\Delta_{geometric}$ (δ^{MG})	61.94%	70.16%
$\Delta_{geometric}$ (δ^{NP})	61.42%	70.68%

Tab. 2: Tables showing accuracies of the Part-Based R-CNN. Table on the left shows the overall accuracy, whereas the right table shows the accuracy of part localization, measured as percentage of correctly localized parts.

The approach of Zhang et al. beats previous state of the art by significant margins, which shows that convolutional neural networks can also be used successfully in a part-based context. Furthermore, the results are suitable to judge the effect of the different scoring functions on the overall outcome. As can be seen in the left table, the usage of a scoring function slightly improves the performance of a system without a scoring function ($\Delta_{null} = 1$, see Equation (5.1)). Moreover, the extension of Δ_{box} to $\Delta_{geometric}$ further improves the performance, although the difference might be less distinct than expected. However, the difference becomes clear, when the part localization accuracy is regarded separately. Here, the usage of $\Delta_{geometric}$ boosts the performance of a system with Δ_{box} by a more significant margin, which shows that the idea behind $\Delta_{geometric}$ actually works. The lower impact on the overall classification can possibly be explained by the fact that descriptors generated by a CNN are robust against small translations, which limits the impact of false localizations.

All in all, it can be said that the constraints enforced by the authors help to improve the results of the convolutional neural network. Moreover, they show that it is wise to rather compare the appearance of parts than the appearance of same regions within images to perform a successful classification. In the following Section 4.2, another approach is presented to show another possible strategy for realizing the basic idea of part-based approaches.

4.2 Learning Features and Parts for Fine-Grained Recognition

Krause et al. [3] use feature descriptors generated by a convolutional neural network in connection with part detectors based on a technique called HOG [16] to form an object representation: Ensemble of Localized Learned Features (ELLF). ELLF stores detected parts of an object together with their appearance, i.e. their respective feature descriptors, and is then used to classify images of a self-collected cars dataset [17] with the help of a support vector machine.

The approach introduces one novelty, as all part feature descriptors are generated from the feature descriptor of the whole image, i.e. the image is divided into parts after the application of the network. This is contrary to other part-based approaches, like Part-Based R-CNN, where first patches of the image (containing desired parts) are extracted and then used as the input for the network to obtain the respective descriptors, i.e. the division into parts happens before the application of the network.

This is only possible because of the unique architecture of Krause et al.'s convolutional neural network, which is presented in Section 4.2.3 and enables a generation of feature descriptors for specific regions of an image. Before that, the general principle of the Ensemble of Localized Learned Features representation will be described in Section 4.2.1. Thereafter, it will be described how discriminative parts of the objects are first discovered and then detected by part detectors based on HOG [16] in Section 4.2.2. Finally, the efficiency of the overall system will be pointed out with the help of results published by Krause et al.

4.2.1 Ensemble of Localized Learned Features The Ensemble of Localized Learned Features (ELLF) stores each detected part of an object together with its respective feature descriptor, e.g. if the front bumper of a car is detected, a feature descriptor for the front bumper region of the image is computed and stored as the front bumper's appearance, i.e. the object parts are directly connected to their appearances. Thus, an object is represented by the concatenation of the descriptors of each part, i.e. if an object category consists of n parts and a_i denotes the descriptor of part i, the resulting ELLF representation is $(a_1, a_2, \cdots, a_n)$.

The generation of ELLF is as follows. Assume that an object category consists of n parts and that training has already happened, i.e. parts have been discovered, the corresponding detectors are trained, etc. Given an input image, parts of the depicted object are first detected by applying the trained part detectors to the image. For each detected part, the convolutional neural network will generate a corresponding feature descriptor of the region defined by the part's bounding box. If a part is not detected, the corresponding entry in the ELLF representation (its feature descriptor) is set to 0. As soon as all entries are set, the ELLF representation will be fed to a linear support vector machine, which will then perform the final classification of the image.

In this procedure, part discovery and detection and the convolutional neural network fulfill important tasks within the overall system. Thus, the process of part discovery and part detection is presented in Section 4.2.2 and details of the unique architecture of the convolutional neural network are pointed out in Section 4.2.3.

4.2.2 Part Discovery and Detector Learning Part discovery is a key component of the approach at hand. It works unsupervised and is not customized to the classification of cars, so that it is applicable to a variety of areas and not limited to the car dataset used in this paper.

The part discovery and detector learning algorithm can be divided into 3 steps. First, sets of aligned images are searched for, i.e. images which show objects in the same position (see Figure 5.9), so that spatial positions of the images can be compared to select certain parts. After that, those parts, which differ most across the whole set of images, are selected to be the defining parts for the object class, i.e. only those parts are used, which are most suitable for differentiation. Finally, patches of the original images are used to train one part detector for each selected part, which concludes the part discovery algorithm. Details of each step will be described in the following.

As mentioned before, the algorithm starts with the search for sets of aligned images. For this, one of the images contained in the dataset is randomly selected as root image and compared to the rest of the data set in terms of HOG features. A certain number n of images that show the highest resemblance is then picked, centered to make comparison easier and stored together with the root image to form

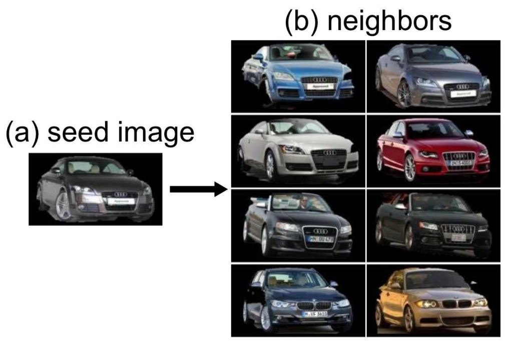


Abb. 5.9: 8 aligned images, which show nearly the same pose like the seed image. Source: [3]

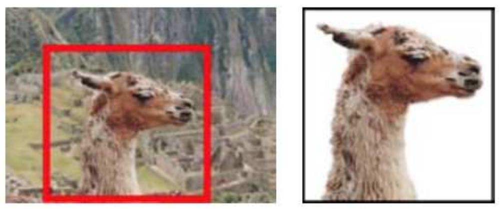


Abb. 5.10: The foreground of the region specified by the user (red frame) is extracted from the image

one set of aligned images (see Figure 5.9). As the background might have unwanted influence on the comparison, the foreground of all images is beforehand extracted using a form of image segmentation called GrabCut [18], which extracts an image's region determined by a bounding box (see Figure 5.10). As the bounding boxes for all objects are annotated in the training data, no additional manual input is needed for this step of the algorithm, which is an important aspect for this approach.

From the so found sets of aligned images, certain parts are selected to become the defining parts for the object class. This is done by randomly sampling some thousand patches of various sizes as part candidates, whereby the same regions are selected on each image. The variance of the candidates' HOG features is then compared across all images, to find parts which differ most, as these are most suitable for discrimination, e.g. cars can easier be distinguished by their front bumper than by their tires. The 10 candidates with the highest variance are finally selected as object parts. Whenever a candidate is selected, all parts that overlap more than a fixed value with the selected part are removed from the list of candidates, so that the selection of redundant parts is avoided. Note that this step is also performed in an unsupervised way, as patches are randomly sampled and as the 10 parts are automatically selected.

Finally, one detector is trained for each selected object part to enable automatic part detection. For this, patches of each aligned image at the part's location are used as positive example, whereas patches from other locations, which must not overlap with the part's location, are used as negative examples. However, it cannot be guaranteed that the images are indeed well-aligned, so that the part's location might not be exactly the same in every image, which has to be considered as well.

Krause et al. propose the following learning objective

$$\min_w \left\{ \sum_j \max\{0, 1 - \max_{z_j^+} w^T h(I_j, z_j^+)\} + \sum_j \sum_{z_j^-} \max\{0, 1 + w^T h(I_j, z_j^-)\} \right\}, \tag{5.4}$$

where $h(I_j, z)$ represents the HOG features extracted from image I_j at location z. Furthermore, z_j^+ is a latent variable representing the true location of the respective part on image I_j, whereas z_j^- are randomly chosen locations on image I_j that do not overlap with the true location of the part and are thus used as negative examples.

The objective is then to find a vector w which yields maximal values when multiplied with the HOG feature vector extracted at positive locations, i.e. maximizing $w^T h(I_j, z_j^+)$, and minimal values when multiplied with vectors extracted at negative locations, i.e. minimizing $w^T h(I_j, z_j^-)$. For this, the latent variable z_j^+ is initialized with the original location z^+, which is only the true location, if the images are well-aligned. However, w is first optimized for this location and after that the best match for all possible locations z_j^+ is chosen. The objective is optimized by alternately optimizing w or z_j^+ while fixing the other variable or by including a penalty for locations z_j^+ which are too far away from the starting location z^+.

At test time, the vector w^T is multiplied by the HOG feature vector of a certain region of an image to obtain the detector score. If no part of the image surpasses a certain threshold, the part is considered to be not visible in the image, which will automatically set the associated feature descriptor in the ELLF representation to 0. The generation of those feature descriptors shall now be explained in the following Section 4.2.3.

4.2.3 Network Architecture The convolutional neural network used by Krause et al. is the second key component of their approach, as it enables the generation of feature descriptors for specific regions within an image, i.e. feature descriptors for different parts of an image can be generated without the need of applying the network to respective image patches like in Part-Based R-CNN for example.

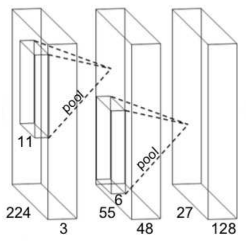


Abb. 5.11: At test time, the convolutional neural network consists of one input layer and 2 convolutional layers. Feature descriptors for specific regions are generated by performing max-pooling on the second convolutional layer. Source: [3]

This is only possible due to the unique architecture of the network at test time, as the fully connected layers are removed after training and feature descriptors are thus only generated by the remaining 2 convolutional layers (architecture at test time as shown in Figure 5.11). The feature descriptor for a specific region is then generated by performing max-pooling on the respective region of the second convolutional layer, which is not possible in a standard convolutional neural network, as spatial information is lost in fully connected layers. Convolutional layers, however, retain spatial information so that a direct relation between a value in the feature descriptor and a region within the image can be established. Like this, Krause et al. can generate feature descriptors for each part by simply pooling the overall feature descriptor at those positions, where a part is detected.

Although the network at hand is based on the AlexNet, there are some more differences to the original architecture, as the network uses 2 convolutional layers instead of 5, which is an adjustment made due to the much smaller scale dataset (17,000 images compared to 1.2 million images), and linear units in the fully connected layers (at least during training). Still, the number of fully connected layers (2), the usage of dropout and of ReLUs in the convolutional layers and the various forms of data augmentation techniques are just like in the approach of Krizhevsky et al.

What is confusing is the fact that the Krause et al. do not use pre-training for their network although their training data is quite small, which is a difference to the Part-Based R-CNN. Although they say that this aspect will not be important in the future anymore, as datasets will become much bigger, this explanation is not convincing, as pre-training could have further improved their results.

Nonetheless, their results are quite impressive, even without pretraining, as will be evaluated in the following Section 4.2.4.

4.2.4 Results Krause et al. apply their system to classify their own car dataset, which makes a comparison to other techniques harder, as no other approach has worked on the same data before. Thus, they also apply other techniques to the same dataset to compare the results and to make founded statements regarding the efficiency of their own technique.

As can be seen in Table 3, ELLF beats the results of the previous best approach by 4.4%. Moreover, it can be seen that the sole application of a convolutional neural network, e.g. the AlexNet, is also better than the previous best, which solidifies the impression obtained in the before discussed approaches that convolutional neural networks can be regarded as state-of-the-art.

The results for CNN-SPM (small and large) further show the importance of the ELLF representation. In those two adaptions of the approach at hand, the ELLF representation is adjusted in such a way that the same entries of the representations of two images refer to the same location within the image. The locations used in the representation are obtained by performing spatial pyramid matching (SPM) on the image, i.e. the image is divided into patches of same size which are then processed by a CNN to generate descriptors for each patch. In the case of CNN-SPM small, first the whole image is used, then the whole image divided into 4 patches and finally into 16; for the CNN-SPM large the image is additionally divided into 64 patches. The descriptors for each of those patches is then concatenated and used as the representation. The adjusted variants are outperformed by the original ELLF representation by 6.0% and 4.6%, respectively, and even the previous best approach is better than them.

Classifying Cars

Model	Accuracy
Previous best	69.5%
CNN-SPM (small)	67.9%
CNN-SPM (large)	69.3%
CNN	70.5%
ELLF	73.9%

Tab. 3: This table shows the performance of several models on the car classification task.

Thus, it is again shown, that the comparison of the actual appearance of parts and not same spatial positions within images is more useful for a successful classification, as was already seen in the Part-Based R-CNN approach. The results of the two previous sections will be summarized in the now following Section 4.3.

4.3 Evaluation

In this section, the most important facts that can be learned from the three different approaches will be presented in order to develop some kind of guideline concerning the application of convolutional neural networks. For this, basic components of all networks will be evaluated and pointed out in connection with other general aspects.

The main problem of all deep convolutional neural networks, i.e. networks with multiple convolutional and fully connected layers, is overfitting, which can be avoided by training the networks with huge datasets. However, such datasets are not necessarily available, which is why a lot of dataset augmenting techniques, like color perturbation or the training on image patches, are used to augment the available training data.

Another possibility is to pre-train the network on a large, additional dataset and then later fine-tune the network on the desired, smaller dataset, which is for example done by Zhang et al. It is also possible to simply use a smaller network, like Krause et al. showed in their approach. However, it one can be assumed that smaller networks without pre-training, which are not used in a part-based system, perform worse than larger networks with pre-training, as the fewer number of layers restricts the level of abstraction computed by the network.

In general, it can be said that the network's size, especially the number of convolutional layers, has to be chosen fitting to the size of the available dataset, which should always be increased as much as possible by multiple data augmentation techniques, because the larger the resulting dataset is the more convolutional layers can be used, thus deriving more and more complex descriptors of the input.

The network's neurons are preferably modeled as rectified linear units. Doing so, response normalization is not needed to prevent neurons from saturating, i.e. making training possible, as Krizhevsky et al. explain in their paper.

Several fully connected layers should be used during the network's training. After that, they can theoretically be removed from the network to obtain feature descriptors that preserve spatial information. This means that every part of the resulting feature vector can directly be associated with a specific region of the input image, which is not possible after the application of fully connected layers, as these mix up the spatial information. Thus, convolutional neural networks can be used to generate two different types of feature descriptors, namely one that can directly be related to the input image and the other being an abstract description of the image.

Another indispensable component of convolutional neural networks are (max-)pooling layers. They are normally used to summarize information of fixed areas in between layers, but as Krause et al. have shown, they can also be converted to pool specific regions of the network's output vector. In connection with cutting off the fully connected layers, this method enables the network to generate feature descriptors for different parts of one input image. This way, Krause et al. perform unprecedented work in the area of part-based approaches, as other approaches, like Zhang et al., need to first extract the desired region from the image and then use the extraction as the network's input to generate the desired feature descriptors. However, both approaches show that it is clever to compare the appearance of parts rather than to compare certain positions within images, as both build representations connecting object parts with their feature descriptors.

Summarizing it can be said that it is not surprising that convolutional neural networks are the new state of the art in object classification, as all three approaches show the potential of these networks on basic research classification tasks. In contrast to that, the application of convolutional neural networks to medicine, a field of particular interest, shall be discovered in the following.

5 Application to Medicine

The requirements for image classification in medicine are a great deal higher than for usage in fields that do not concern life, because false detections, both false-positive and false-negative, can cause life-threatening consequences. Thus, medical image processing systems should have high precision, i.e. a high detection rate and a low number of false positive detections.

Even slight improvements in either of both aspects make such systems more applicable. Knowing that convolutional neural networks can be successfully combined with other techniques, it becomes plausible that they could be used for these improvements in connection with already existing methods. And indeed, a lot of combined methods can be found. First, early development is shortly described, before current progress, especially based on Krizhevsky et al.'s basic research, is stated.

5.1 First Attempts

First attempts in this domain were undertaken in the mid 90's, after LeCun et al. had proposed convolutional neural networks in 1989 [6].

Lo et al. [19] applied convolutional neural networks to the detection of lung nodules on chest radiographs, which can be indicators for lung cancer. As assumed above, they used the network in combination with other methods, namely to verify pre-scan results, i.e. to filter out false-positive detections (FP) and to confirm true-positive detections (TP). The results achieved by the network were later validated. The authors report that their network reduces the number of FPs by 79% and preserves 80% of TPs.

This shows that convolutional neural networks can effectively improve other techniques to come near to performances of radiologists, especially considering studies by Stitik et al. [20] from 1985, who stated that one radiologist was only able to detect 68% of lung nodules back in that time, which shows that tasks like this were even hard for human beings not too long ago.

Despite the obvious potential of convolutional neural networks in Computer Aided Detection (CADe), they were still rarely used, as a paper review by Shi et al. [21] showed in 2011. They point out that convolutional neural networks were only used in 2 of the 23 reviewed papers concerning medical object detection and recognition. Although their choice of papers might not be representative for the real application of such networks, it clearly shows that there were not many common papers proposing CNNs for medical image processing, as these would have been surely included in their work. Following the publication of Krizevshky et al. [1], this has changed however, as there have been a variety of newly developed techniques in medical image processing involving convolutional neural networks. Two of them will be exemplarily described in the next subsection.

5.2 Recent Progress

Holger Roth et al. [22] build upon state-of-the-art Computer Aided Detection methods for computed tomography (CT) images by Liu et al. [23] (mediastinal CT volumes) and Cherry et al. [24] (abdominal) to detect enlarged lymph nodes. The CADe methods can detect almost 100% of the occurrences of lymph nodes, although this high sensitivity, i.e. high number of true positive detections and low number of false negative detections, is at the expense of a relatively high number of false positive detections, namely about 30 per image. Thus, a challenge is to filter out those false detections, while simultaneously preserving correct detections.

The network used for this works with the cuda-convnet implementation by Krizhevsky et al., ReLUs and DropConnect, an advancement of Dropout which drops out single connections instead of whole neurons, with the architecture being illustrated in Figure 5.12.

The preprocessing CADe systems by Liu et al. and Cherry et al. yield marked regions in the 3-dimensional CT volumes, indicating where enlarged lymph nodes have been detected. Each marked region is then divided into one set of three 2-dimensional images, namely axial, one coronal and one sagittal image, which are all centered at the center of the region (see Figure 5.13). Doing so, the authors can use a CNN designed for RGB images by simply using each slice of the 3-D volume as the input for one of the

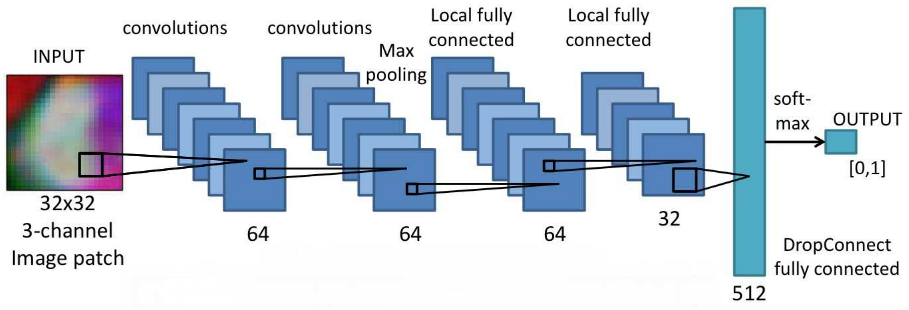


Abb. 5.12: The CNN consists of 2 convolutional layers, followed by a max-pooling layer, 2 fully-connected layers without DropConnect, 1 fully-connected layer with DropConnect and a final 2-way softmax. Source: [22]

3 input channels, i.e. the axial image for the first channel, the coronal image for the second channel and the sagittal image for the third channel.

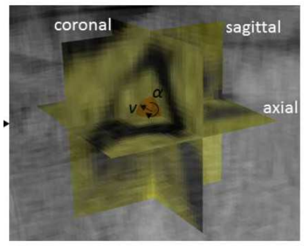
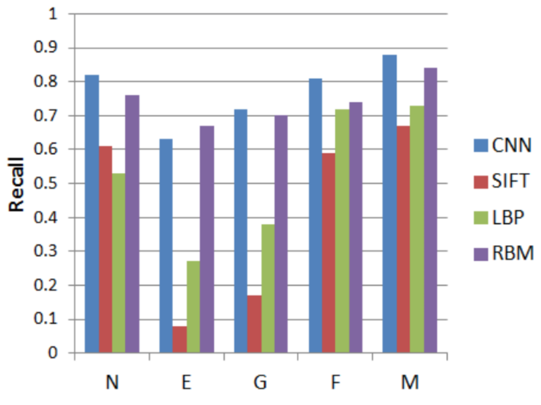


Abb. 5.13: Decomposition of the 3-dimensional CT volume into 3 2-dimensional images. Source: [22]

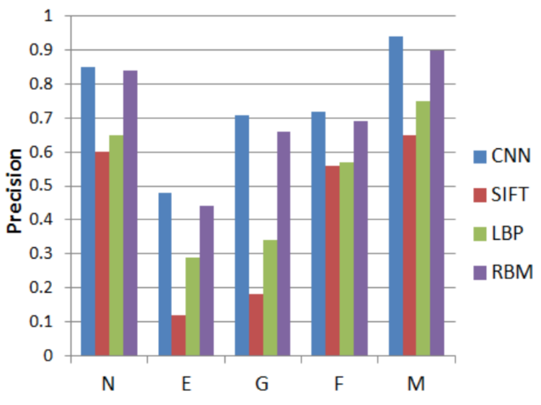
Finally, two different networks are trained, one for each category of CT volumes, i.e. one for mediastinal images and one for abdominal images. The application of convolutional neural networks improves the sensitivity of the mediastinal lymph node detection from 55% of the stand-alone CADe method (Liu et al.) to 70% and from 30% (Cherry et al.) to 83% in the abdominal regions, all achieved with a fixed rate of 3 false positive detections per image, e.g. 83% of the true positives were detected, before 3 false positives occured. This proves that the performance of existing CADe systems can be drastically improved by the application of convolutional neural networks. Furthermore, it shows one easy way to use 2-dimensional networks for 3-dimensional volumes is shown, which is important as the typical data type in medical imaging is 3-dimensional, e.g. CT or MRI volumes.

Another application to medicine is shown by Li et al. [25], who use a CNN to classify HRCT lung volumes of the interstitial lung disease (ILD) dataset [26] into 5 different classes, one of them representing healthy lungs, whereas the other classes are different types of diseases. The authors apply a quite small convolutional neural network, which is not pre-trained, to 2-D slices of the original HRCT volumes and they do not use any kind of other techniques to further improve the performance. Still, they achieve a recall, i.e. sensitivity, which is better than previous best for all but one class of disease, and a precision, i.e. the fraction of true positives on all positives (false and true positives), which is better for all classes (see Figure 5.14).

Thus, it can be concluded that convolutional neural networks also show promising results in a medical context, which implies that they could be used regularly in the near future. Moreover, there are easy ways to use the common 3-dimensional medical imaging data with standard convolutional neural networks, although this causes a loss of information, as slices of the volumes only cover a small percentage of the available data. This is why it could be beneficial to further research efficient, true 3-dimensional convolutional neural networks, which is a difficult task as they would have a huge number of parameters and the available datasets in medicine are very small, even when used with standard CNNs. Nonetheless, the research should be continued in any case to avoid a stagnation like before.



(a)



(b)

Abb. 5.14: Results for the approach by LI et al. on the ILD classification. Source: [25]

6 Conclusion

In this paper, current progress in fine-grained image classification has been shown. Although convolutional neural networks have already been proposed in 1989 by LeCun et al. [6], their application to medical problems has stagnated from the mid 90's. However, early attempts back then, like Lo et al. [19], have already shown that convolutional networks could be beneficial for medical image processing, above all as verifiers for other image processing tools. Despite that, CNNs have not been extensively used in medical context, as the results of a paper review by Shi et al. [21] from 2011 imply. This could be explained by the fact that there had not been substantial progression in the basic techniques of convolutional neural networks, which helped to overcome deficits which were crucial for the application in medicine. What can be mentioned here is e.g. the need for a huge number of training examples, which is especially difficult to obtain for rare diseases, and for a high accuracy, because both false-positive and false-negative detections can have serious consequences for a patient's health.

The paper by Krizhevsky et al. [1] in 2012 has certainly helped to bring convolutional neural networks back to the center of attention. In their work, they showed how to efficiently train deep convolutional neural networks, thus making them more useful in general. Although their approach can be regarded as new state of the art, it uses the network as a stand-alone system and works on a huge dataset, which is different to the real 'work environment' of such networks in medical context. Thus, the papers by Zhang et al. [2] and Krause et al. [3] provide important insights regarding the applicability of this new kind of networks in connection with other techniques, as they successfully use them in part-based surroundings together with techniques like HOG feature descriptors and support vector machines.

Approaches by Roth et al. [22] and Li et al. [25] transform the newly learned techniques in a medical context. They both use standard 2-dimensional convolutional neural networks to work on the common 3-dimensional volumes in medicine, taking slices of the original volumes as the input for their networks. Roth et al. use their network to verify the detections of state-of-the-art Computer Aided Detection methods with a 2-way classification network, i.e. either verifying or rejecting the CADe detections. Li et al. use a 5-way classification network to classify lung images into 5 different classes. Both achieve promising results.

Summarizing, it can be said that the importance of convolutional neural networks in medical context will certainly rise in the (near) future. On the one hand, this can be explained by the enormous positive effects that CNNs already provide today, as shown above. On the other hand, it is nearly safe to say that the performance of convolutional neural networks will further improve with progression in hardware power and the availability of larger datasets of medical images. Still, the basic principles of convolutional neural networks should also be further improved, e.g. to make true 3-dimensional convolutional neural networks feasible, as they are the only ones which really make use of all the information present in 3-dimensional volumes.

Literaturverzeichnis

[1] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems; 2012. p. 1097–1105.

[2] Zhang N, Donahue J, Girshick R, Darrell T. Part-based R-CNNs for fine-grained category detection. In: European Conference on Computer Vision; 2014. p. 834–849.

[3] Krause J, Gebur T, Deng J, Li LJ, Fei-Fei L. Learning Features and Parts for Fine-Grained Recognition. In: International Conference on Pattern Recognition; 2014. p. 26–33.

[4] Breast MRI results. Mayo Foundation for Medical Education and Research; 2014. [Online; Accessed January 9, 2015]. Available from: http://www.riversideonline.com/source/images/image_popup/mcdc7_breast_mri_results.jpg

[5] O'Leary M. MRI Breast Cancer Screening in High-Risk Women Boosts Detection Rates. Medical Digest Publications; 2011. [Online; Accessed January 9,2015]. Available from: http://medicaldigestpublications.blogspot.de/2011/09/mri-breast-cancer-screening-in-high.html

[6] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, et al. Backpropagation applied to handwritten zip code recognition. Neural computation. 1989;1(4):541–551.

[7] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: International Conference on Machine Learning; 2010. p. 807–814.

[8] My LeNet. Theano Development Team; 2015. [Online; Accessed January 15,2015]. Available from: http://deeplearning.net/tutorial/_images/mylenet.png

[9] Berg A, Deng J, Fei-Fei L. Large scale visual recognition challenge. Technical report; 2010. [Online; Accessed January 9, 2015]. Available from: http://www.image-net.org/challenges

[10] Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580; 2012.

[11] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524; 2013

[12] Welinder P, Branson S, Mita T, Wah C, Schroff F, Belongie S, et al. Caltech-UCSD Birds 200. Technical report; 2010. CNS-TR-2010-001.

[13] Wikipedia. Support Vector Machine. In: Wikipedia, Die freie Enzyklopaedie; 2015. [Online; Accessed January 9, 2015]. Available from: http://de.wikipedia.org/w/index.php?title=Support_Vector_Machine&oldid=136138406

[14] Uijlings JR, van de Sande KE, Gevers T, Smeulders AW. Selective search for object recognition. International journal of computer vision. 2013;104(2):154–171

[15] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, et al. Caffe: Convolutional architecture for fast feature embedding. In: ACM International Conference on Multimedia; 2014. p. 675–678.

[16] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: Conference on Computer Vision and Pattern Recognition; 2005. p. 886-893.

[17] Krause J, Stark M, Deng J, Fei-Fei L. 3D Object Representations for Fine-Grained Categorization. In: IEEE International Conference on Computer Vision Workshops; 2013. p. 554–561.

[18] Rother C, Kolmogorov V, Blake A. Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM Transactions on Graphics; 2004. p. 309–314.

[19] Lo SCB, Chan HP, Lin JS, Li H, Freedman MT, Mun SK. Artificial convolution neural network for medical image pattern recognition. Neural Networks. 1995;8(7):1201–1214.

[20] Stitik FP, Tockman MS, Khouri NF. Chest radiology. In: Screening for cancer; 1985. p. 163–191.

[21] Shi Z, He L. Current status and future potential of neural networks used for medical image processing. Journal of multimedia. 2011;6(3):244–251.

[22] Roth HR, Lu L, Seff A, Cherry KM, Hoffman J, Wang S, et al. A new 2.5 D representation for lymph node detection using random sets of deep convolutional neural network observations. In: Medical Image Computing and Computer-Assisted Intervention; 2014. p. 520–527.

[23] Liu J, Zhao J, Hoffman J, Yao J, Zhang W, Turkbey EB, et al. Mediastinal lymph node detection on thoracic CT scans using spatial prior from multi-atlas label fusion. In: SPIE Medical Imaging; 2014. 90350M.

[24] Cherry KM, Wang S, Turkbey EB, Summers RM. Abdominal lymphadenopathy detection using random forest. In: SPIE Medical Imaging; 2014. 90351G.

[25] Qing L, Weidong C, Xiaogang W, Yun Z, Dagan F, Mei C. Medical Image Classification with Convolutional Neural Network. In: International Conference on Control, Automation, Robotics and Vision; 2014. p. 844–848.

[26] Depeursinge A, Vargas A, Platon A, Geissbuhler A, Poletti PA, Muller H. Building a reference multimedia database for institial lung diseases. In: Computerized Medical Imaging and Graphics; 2012. p. 227–238.

Image fusion on neural networks

Jan Kehren

1 Abstract

In this paper we will discuss image fusion on neural networks in relation to medical imaging. First of all we will give you a short introduction about what image fusion is and what neural networks are. After this we will see how you can combine those with medical imaging and we will see that the inventions of the last decades prepared new ways for researchers and other people. This paper will focus on the classic feed-forward neural networks but will touch on fuzzy approaches and wavelet transformation. In the end you will see a short prospect on the topics of current research.

2 Introduction

2.1 What is image fusion?

Image fusion is a technique to combine two images into a single image that has the maximum information content without producing details that are non-existent in the given images.[8]
There are many application areas for image fusion i.e. on a weather map at first you get the map oft the earth's surface and then you fuse this map with clouds, air pressure and so on. An other example is the medical technology where you merge different medical images to get more information about the disease picture of the patient.
Before we can start with the image fusion we need to register the images to each other.
The following graphic shows the single steps of image fusion:

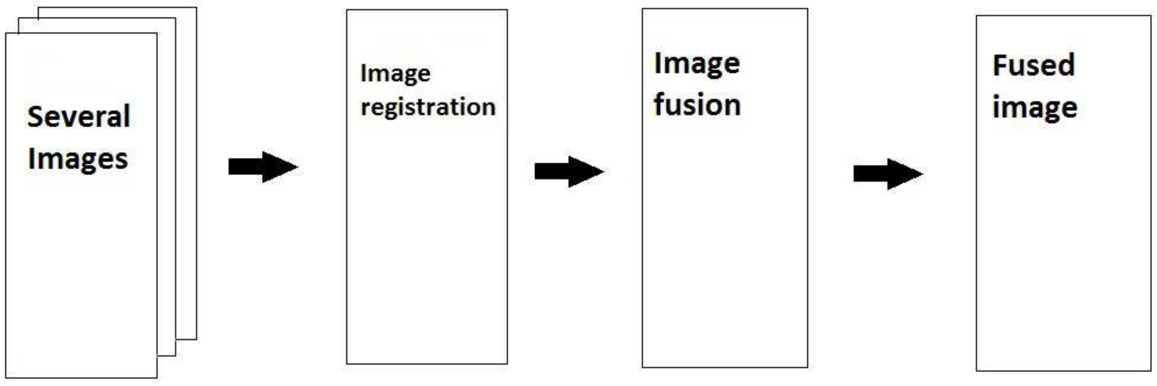


Abb. 6.1: Single steps of image fusion.

2.1.1 Image registration Image registration is when we register two or more images to each other.
We register them to each other thereby we transforming the different sets of data into one coordinate system.
At first we choose an reference image which is the coordinate system into which we transform the other sets of data. We call the other images object images.
There are several methods for image registration i.e. intensity vs. feature-based , Spatial vs. frequency-domain methods, etc.
We now have a closer look at one at one easy registration algorithm. It is called rigid transformation.[4]

Definition: rigid transformation
The rigid transformation is defined by $X_B = R_{AB}X_A + t_{AB}$ where X_B and X_A are two points and the point X_A is mapped on point X_B. With rotation R_{AB} and shift t_{AB}[4]

As we can see by the definition the rigid transformation does not change the structure of an object so its clear we can only use the rigid transformation if there is a solid structure. The rigid transformation also retained distances between every point. This is an ability that exists because there are only displacement, rotation, reflection or a combination from these allowed.
The following image shows an registration of an MRI (first two images in the first line) and an CT (last two images in the first line) image and the registered result in the second line [4]:

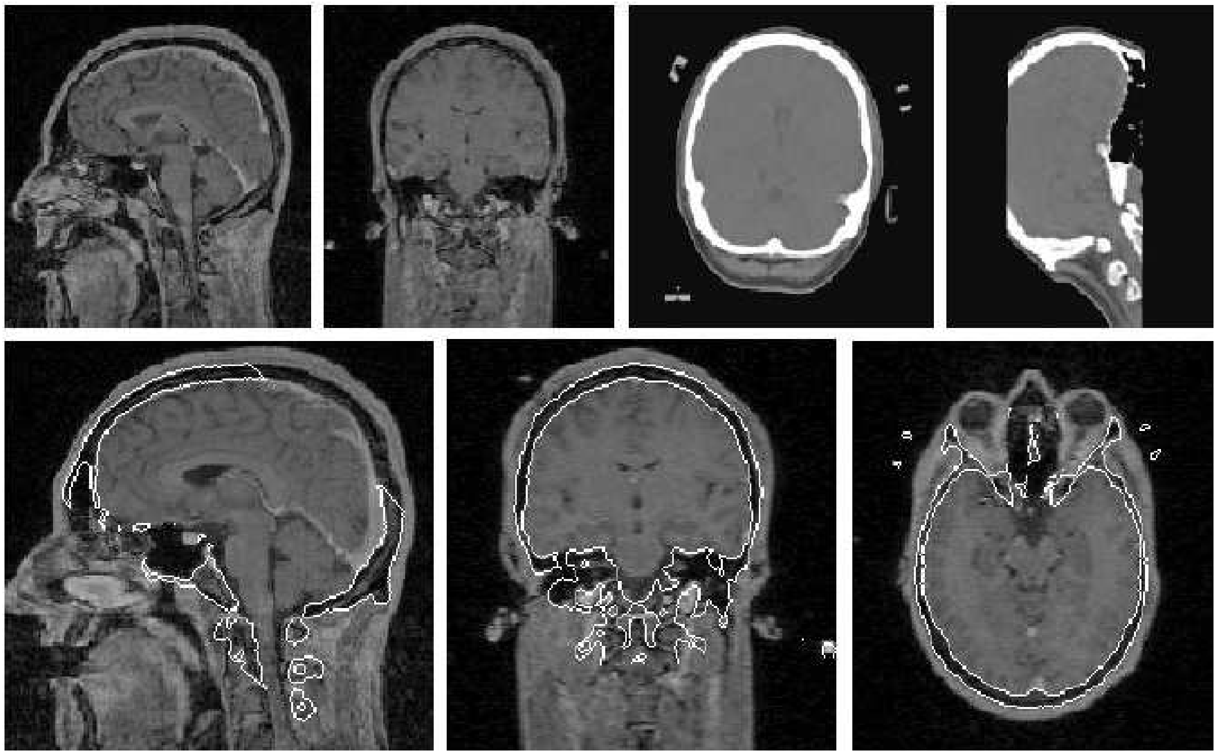


Abb. 6.2: Registration of an MRI

2.1.2 Image fusion

The different application areas ensure that there are several methods of image fusion we will present in this paper: multisensor, multitemporal, multifrequeny, multipolarization and multiresolution image fusion.
The different pictures are often represented in different colors in the final image to discover the different images in the end.[2]
The current state of technique allows us to get a high quality fused image with spatial and spectral information by different algorithms. In the following we will discuss some algorithms for image fusion and we will compare them with some performance parameter like peak signal to noise ratio (PSNR), Normalized correlation (NC), and Men square error (MSE) and their advantages and disadvantages. The methods of image fusion can be split up into two groups:

1. Spatial domain fusion method
2. Transform domain fusion

In the spatial domain fusion method we directly work with each pixel. The pixel values are manipulated to achieve desired result. A big disadvantage of spatial domain fusion is the spatial distortion which

appear in the fused image. Therefore we have some problems with image blurring which is really bad in medical imaging because there is a need of high accuracy when dealing with a human life. One big advantage of spatial domain fusion is that we a high high spatial resolution.
In the following I will present two easy algorithms for spatial domain fusion.At first we will have a look at the simple average algorithm[8]:

Definition: Simple average
The simple average algorithm is defined by $K(i,j) = \dfrac{X(i,j)+Y(i,j)}{2}$ *where* $X(i,j)$ *and* $Y(i,j)$ *are two input images with coordinators* i *and* j *and* $K(i,j)$ *is the Output image with the corresponding pixel* i,j *[8]*

The simple average algorithm is quite easy. Its a fact that the regions of images that are in focus tend to be of higher pixel intensity so the algorithm brings all regions in focus by calculating the average intensity of 2 pixels from 2 given images and assign this value to the corresponding pixel in the output image.
Now we will have a look at the simple maximum algorithm:

Definition: Simple maximum
The simple maximum algorithm is defined by $K(i,j) = \max X(i,j) + Y(i,j)$ *where* $X(i,j)$ *and* $Y(i,j)$ *are two input images with coordinators* i *and* j *and* $K(i,j)$ *is the Output image with the corresponding pixel* i,j *[8]*

The simple maximum algorithm uses the fact that the regions of images that are in focus tend to be of higher pixel intensity too. It just assign the greatest value to the corresponding pixel. This ensures that we have an highly focused output image.

In the transform domain fusion we first transfer the image so we do not directly deal with each pixel. In frequency domain we have to transform the image into the frequency domain that mean we compute the Fourier Transform of the image then we start with other computations like discrete wavelets transform (DWT). In the end we use the Inverse Fourier transform to get the fused image. One big advantage is that we can easily handle spatial distortion with the frequency domain methods. So we do not have spatial distortion but we have some spectral noise. Overall in the end we have an high quality spectral content.
Now we will have a closer look to the DTW [8]:

Definition: Discrete wavelets transform
Wavelets can be described by using two functions. The scaling function f(t), also known as father wavelet and the wavelet function t or mother wavelet. t undergoes translation and scaling operations to give self similar wavelet families as given by Equation:

$$\psi_{a,b}(t) = \tfrac{1}{\sqrt{a}}\psi(\tfrac{t-b}{a}), (a, b \in R), a > 0$$

The DWT different the wavelets into low-high, high-low, high-high spatial frequency bands at different scales and the low-low band at the coarsest scale. In the low-low band we see the average image information. The other bads contain things like directional information or spatial orientation. Higher absolute values of wavelet coefficients in the high bands correspond to salient features such as edges or lines.
The following picture from *http://www.intechopen.com/source/html/45735/media/image1.jpeg* shows the wavelet based image fusion:[8]

The following table from [8] is a summery of different fusion techniques with their advantages and disadvantages.

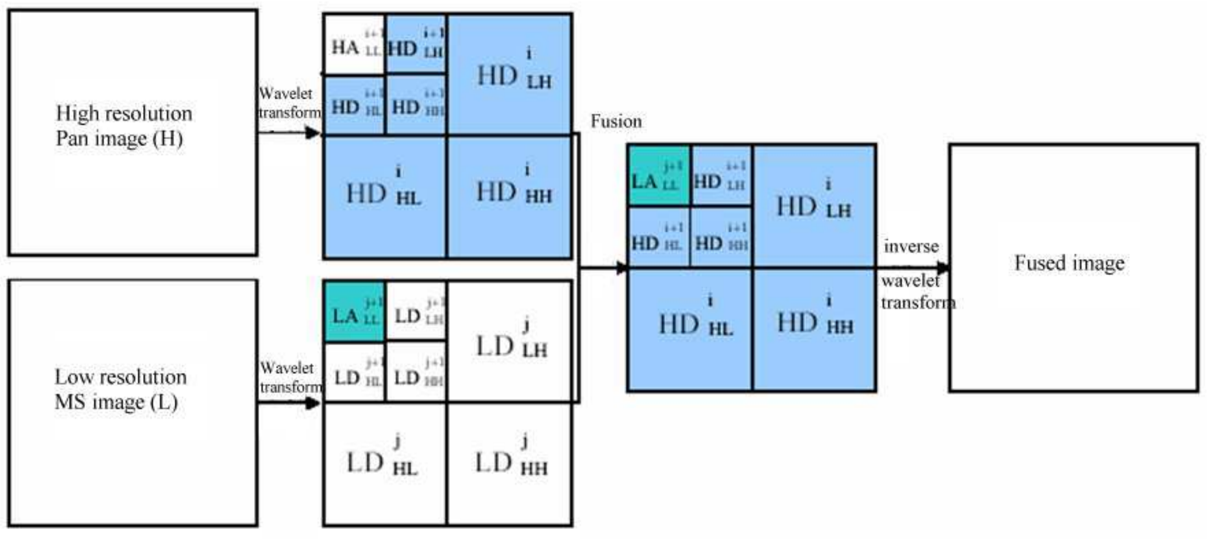

High resolution Pan image (H)
Wavelet transform
Low resolution MS image (L)
Wavelet transform
Fusion
inverse wavelet transform
Fused image

Abb. 6.3: Wavelet based image fusion.

Number	Fusion Technique/Algorithm	Domain	Measuring Parameters	Advantages	Disadvantages
1	Simple Average	Spatial	PSNR = 25.48 EN = 7.22	This is the simplest method of image fusion.	The main disadvantage of Pixel level method is that this method does not give guarantee to have a clear objects from the set of images.
2	Simple Maximum	Spatial	PSNR = 26.86 EN = 7.20	Resulting in highly focused image output obtained from the input image as compared to average method.	Pixel level method are affected by blurring effect which directly affect on the contrast of the image.
3	Principal component analysis	Spatial	PSNR = 76.44 NC = 0.998	PCA is a tools which transforms number of correlated variable into number of uncorrelated variables, this property can be used in image fusion.	But spatial domain fusion my produce spectral degradation.
4	DWT	Transform	RMSE = 2.06 EN = 7.42	The DWT fusion method may outperform the slandered fusion method in terms of minimizing the spectral distortion. It also provide better signal to noise ratio than pixel based approach.	In this method final fused image have a less spatial resolution
5	Combine DWT, PCA	Transform	PSNR = 67.08 EN = 7.24	Multi level fusion where the image undergoes fusion twice using efficient fusion technique provide improved result .output image contained both high spatial resolution with high quality spectral content.	This method is complex in fusion algorithm. Required good fusion technique for better result.
6	Combination of Pixel and Energy Fusion rule	Transform	PSNR = 27.75	Preserves boundary information and structural details without Introducing any other inconsistencies to the image.	Complexity of method increases.

Overall we can see that PSNR of the average method is less then in frequency domain method like SWT, that means fused image are not precisely to registered image. This is the reason why transform domain fusion is more suitable then spatial domain method. However we ca not say that we don't need the spatial domain methods because the fused image contains high spatial information. So we can say the algorithm we choose still depend on the problem we want to solve.
At least we can say spatial domain has high spatial resolution but problems with image blurring and transform domain has a high quality spectral content. If we bring both domains together like combining DWT and PCA we get both good features in the fused image. [8]

2.2 What are neural networks?

A neural network consists of several computational units, we call these units neurons. Between two neurons exists a directional, wight connection, so you can imagine a neural network as an graph. The different nodes of that graph are the computational units, there are different nodes for input and output.
We need a way to give the neural network an input and to get an output from the network to compute something. We have an input vector and an output vector for every network. The dimension of these vectors are given trough the neural network, the dimension of the input vector is the number of the neurons in the input layer and the dimension of the output vector is analogous to that the numbers of neurons in the output layer.
Every neuron in the neural network has an threshold value, this threshold value indicate the level of activation. The level of activation is defined by an activation function that refers to the old level of activation and the threshold value.
The connections are used to send data between the different neurons due to that there are only directional connections and it is only possible to send data in one direction with one connection. The wight of the connection reinforce or inhibit the data transmission. We define the connection between two neurons i and j as Wij.[3] [5]

Definition: Neural network
A neural network is a sorted triple (N, V, w) with two setsN, V and a function w, where N is the set of neurons and V a set $\{(i, j)|i, j \in N\}$ whose elements are called connections between neuroni and neuron j. The function $w : V \rightarrow R$ defines n. The weights, where $w((i, j))$, the weight of the connection between neuron i and neuron j, is shortened to w_{ij} . Depending on the point of view it is either undefined or 0 for connections that do not exist in the network.[5]

Neural networks are derived by the biological neural network, for this reason the neural network has to learn. This happens through learning algorithms, therefor exist many different learning algorithms for the many different application areas.
As we now know the basics of what a neural network is and how it works we will have a look at 3 different topologies how to construct a neural network and how the network learn. So we get the ability to learn complex transformations.[5]

2.2.1 How neural networks learn We have different kinds of learning algorithms for neural networks. These algorithms train the network to give the right output for a given input.
The common way to train a neural network is to change the weight of the connections so we only will discus the change of connection wights.
For every network we have a training set. A training set is a set of training patterns, which we use to train our neural network.
Also there are 3 different types of learning: Unsupervised learning, reinforcement learning and supervised learning. In this paper we will mostly have a look at the supervised learning.
In supervised learning we have a training set and to this training set a given output. So we give the training set to the network and after the network did his computation we compare the original output with the from the training set. Through this output we create our error vector which train our network by changing the wights of the connections.
We can do this method online and offline. That means we can change the wights after every training pattern or we can run the complete training set on the network and change the wights after the complete training set.
In the following part we will talk about the back propagation of error algorithm.[5]

Definition: Back propagation
Back propagation can be defined by the change of the wights by $\Delta\omega_{ij} = \eta\frac{\partial E}{\partial \omega_{ij}} = n\delta_j x_i$ where $\Delta\omega_{ij}$ the change of the wight , n the learning rate, δ_j the error signal of the neuron j, E the error function and x_i the output of the neuron i is.[5]

Back propagation is an supervised learning where we start with forward propagation of a training pattern through the neural network. Then we compare the given output and the real output through this comparison we get the error vector. If we have our error vector we start with back propagation of the error from the output layer to the input layer and while we doing this we change the wights of the edges.[5]

2.2.2 Feed-forward networks Feed forward networks work with a layer model which contains 3 kinds of layers. This kind of neural network will be the main focus in this paper. The first layer is the input-layer which manages the input which comes to the network. After the input-layer there are several computational-layer which do all computation in the network, these layers are not visible from the outside. The last layer is the output layer which manages the output from the neural network.
A point that fell account is that in feed forward networks the connections are always directional to a layer that is nearer to the output-layer then the layer before so it the directions always are from n to n+1. If every neuron in layer n has a connection to every neuron in layer n+1 then the network is called completely linked.

Definition: Feed-forward network
The neuron layers of a feed forward network are clearly separated: One input layer, one output layer and one or more processing layers which are invisible from the outside (also called hidden layers). Connections are only permitted to neurons of the following layer.[5]

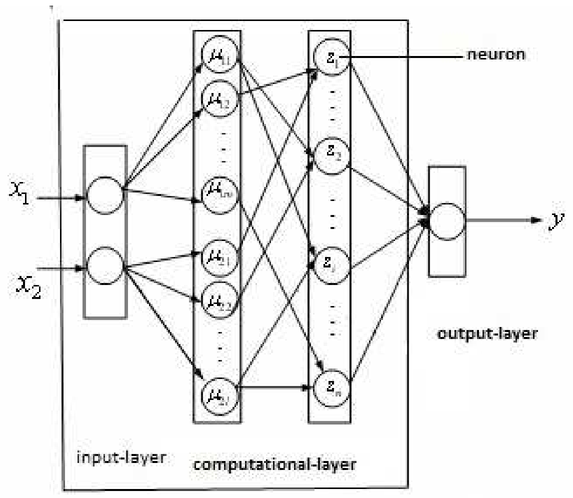


Abb. 6.4:

In some feed forward networks it is possible to jump over a layer. The data starts in layer n and arrives in layer n+2, we call that jumps ShortCut-Connections.[5]

2.2.3 Recurrent networks Recurrent networks are an extension of the feed-forward networks. An important point is that in recurrent networks the neurons are not explicit input or output neurons. Recurrent networks are split up into three groups: direct recurrence, indirect recurrence and lateral recurrences.
In **direct recurrence** networks we add the possibility to make connections from a neuron to itself so it by influencing itself directly therefore direct recurrence.
In **indirect recurrence** networks are also an extension of the feed forward networks with the possibility to have connections from layer n to layer n-1 so the neuron can have influence on itself by influence the neuron where the input comes from.
The last ones are the **lateral recurrences** networks. They are also an extension of the feed foreword networks with the possibility to have connections inside the layer.[5]

2.2.4 Completely linked networks Completely linked networks have the special feature that the layers model is canceled because every neuron can have a connection to every other neuron except itself. So the direct recurrent is not allowed in this kind of network.[5]

2.2.5 Hopfield model Hopefield models are inspired by the behavior of particles in magnetism [5]. It is a completely linked network where every particle tries to minimize his requirements energy. The idea is that we use the rotation angle of every particle to represent our date and we use that the connections have symmetric wights. Also we only use 2 rotation angle so we have a binary level of activation.
Therefore Hopefield networks have the property that they always converge to 0. That is the point why we always know when the computation is done. It is done if the network does not change anymore.
Then it is clear how we give an input and get an output from this network. We have a binary string and let each neuron represent an number of this string, this is how we put an input to this Network. Then we let the network converge and when the network do not work any longer we can read our output as the same way we give our input. The output is the last configuration of the network, so our output is an binary string too.

3 Usage of image fusion and neural networks in medical imaging

One big application area for image fusion is Medical imaging. We need image fusion in the tomography to compute and manipulate the images of i.e. mrt or ct. Neural networks become a part of this to reduce the computational time.
To reduce the computational time of tomographic images in medical imaging we use hopefield models. It is shown that the time where we start using the neural network is really impotent. There are 2 variations applying the neural network before or after the wave field collection. Applying the neural network in the beginning of the wave field collection we need around 10.000 iterations and after the wave field collection we need around 200 iterations. So we see there is an impressive improvement if we use the neural network after the wave field collection.[6]
But why do we need image fusion and neural networks in medical imaging?
Therefore are several reasons at first we have necessity of high accuracy when dealing with a human life. If we want this high accuracy we can get it if we keep the false negative cases at a very low rate.
Another point of view is that hospitals need to control their costs and good software can assist human and help them work more efficient.
A following point is that each of the different diagnostic tools like X-ray, MRI, CT, PET scan etc. have different advantages, disadvantages and application areas. For example MRI is good to see tissue structures and a PET scan maps biochemical and physiological functions.[1] Now we are able to combined these two images into one image with image fusion to see where is a malfunction like in this example (http://www.medicalexpo.de/) where you see in the first row an MRI scan in the second row a PET scan and in the third row the fused image:

3.0.6 Group Method of Data Handling (GMDH) The Group Method of Data Handling (GMDH) is a collection of algorithms that has a wide field of usage. It is used for data mining, forecasting, optimization and knowledge discovery. In medical imagining GMDH is used in a variation the inductive GMDH to find the best neural network for an algorithm and to increase his correctness. In [6] a list of useful scenarios of GMDH is given :

- Optimal complexity is known
- NN parameters are calculated automatically in hidden layers
- Most accurate model should be selected
- Simple programming is needed
- Apriori assumptions of modeling should be minimized

Now we will present some applications areas for GMDH in medical imaging.
At first we have the a GMDH based algorithm for analyzing images in 3D to recognize the heart. The algorithm uses a feedback methodology to detected the properties of the images. The algorithm self selecting the neural network. The advantages are that this algorithm shows a good performance in extracting the features from the image. A disadvantage of this algorithm are the complex computations which are distress but overall we can say that the algorithm based on GMDH efficiently recognize the heart in images.

An other usage of GMDH is an algorithm that recognize the different organs in an CT image of the

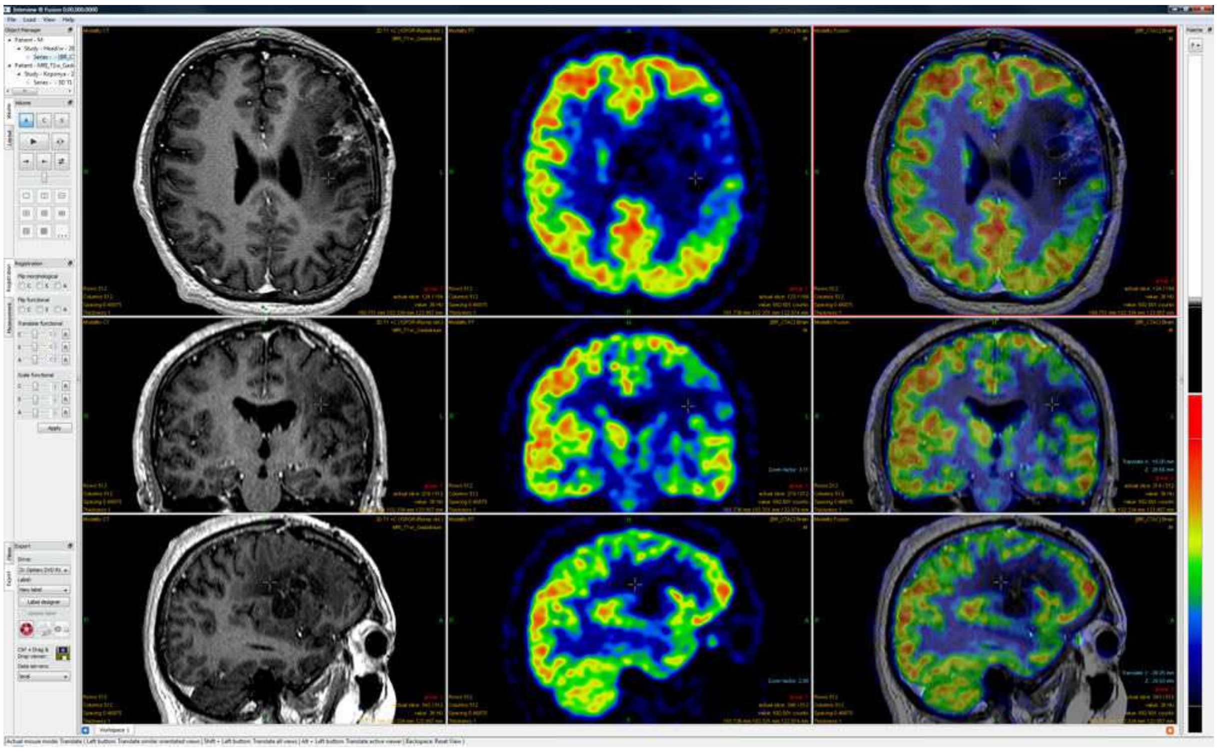


Abb. 6.5: Image fusion

belly space. The algorithm extract the single organs for analysis or further observation. this algorithm also self select the neural network by setting a lot of variables to find the most fitting network. A big advantage is the we only need one algorithm to detect several organs. The disadvantage is that the algorithm is only compatible with CT images that makes him inflexible. Overall the algorithm has a good detection of lungs stomach and spleen.

Because the detection of lungs from chest X-ray image fails using the normal neural networks, GMDH brings the solution. The algorithm is extended to detect cancer. It works in 2 steps with multi-detector row images what an disadvantage is because it only work on those.

3.0.7 Probabilistic Neural Networks (feed forward network) Probabilistic neural networks or feed forward network have great influence in medical imaging. There are several application areas for them like the following table from [6] shows :

Number	Application	Advantages	Limitations	Results
1	Probabilistic neural networks based tumor detection	Fast process and involves less complexity.	None	A good performer and can be helpful in detection of tumors
2	PNN vs MLP	Multi-layer perceptron is high performer. performance then MLP.	Probabilistic has less	In some cases MLP is better performer.
3	Image evaluation based in PNN	Better decision making is archived with machine learning	Compound of dual systems and less reliable.	A good tool to be considered as a helping program but cannot fully rely on this system.
4	Contour detection in ultrasound	Effective for 3D data	Computationally complex and expensive.	The proposed method accelerates the reconstructions by roughly a factor of three on average for typical 3D multi slice geometries.
5	Probabilistic Shape representation	Easy mapping of simplex to real world units.	No reverse mapping	An average algorithm to be used in helping medical imaging.
6	Image registration using random coefficient	Minimized energy function	None	The experiment showed satisfactory results.
7	Probabilistic image validation	improved accuracy	No detailed performance testing for performance.	Accuracy more focused whereas no stats
8	Kidney contour detector	Simple user interface	None	A good system with light weight application.
9	Deformable density matcher	None	Relatively complex computations	No promising results are shown
10	Probabilistic validation tool	Improved reliability	None	Significant improvement in reliability of the tool for extraction of 3D shapes is proposed. A good implementation work
11	Temporal analysis of brain MRI	Brain disease can be detected earlier	Complex system	Shown good results but performance is missing
12	CT image reconstruction	Reconstructs the series of 2D image and display the result as 3D	A variety of problems exist in the method that needs improvement.	The results obtained by the method are satisfactory.

We will now have a closer look in the probabilistic neural networks based tumor detection from the table above.
In this paper the authors uses the principal component analysis for the feature extraction. Its one of the most successful techniques that have been used in image recognition and compression because PCA reduce the large dimensionality of the data. In the paper a set of MR images is used as training set to classify the brain tumors. The task is now to find the most similar feature vector to the feature vector of a given test image .
In the training phase of the network the feature vectors of all test images extracted with the assistance of PCA. Therefore the image were transferred into single pixel vectors. Now PCA is applied to reduce the dimensionality of these vectors. So each feature vector is computed and stored.
We go over to the testing phase where the feature vector of the test image is computed with PCA. Then we compute the similarity between the feature vector of the test image and of all the training images and if we assign it to the most similar. The similarity is computed with the Euclidean distance. The following image from [7] shows the schematic diagram of a MR image recognizer:

The probabilistic neural network has a lot of advantages for example it has a very high training speed, the training is instantaneous and it is robust to noise examples.
We have instantaneous learning because we do not have any wight changes. We train the network by adding new connections and the wights get assigned. So the network is also robust to wight changes. These abilities makes the network very fast.
The network classifies an input vector into the class that has the maximum probability to be correct. The structure of the network is described by the following picture from [7]:

In [7] were 2 sets of data. The first set is the training set which consists of 20 images and the second one is the testing set which consists of 15 images. The training set is to train the network and the testing set is used to verify the accuracy and the effectiveness of the network. The spread value of the radial basis

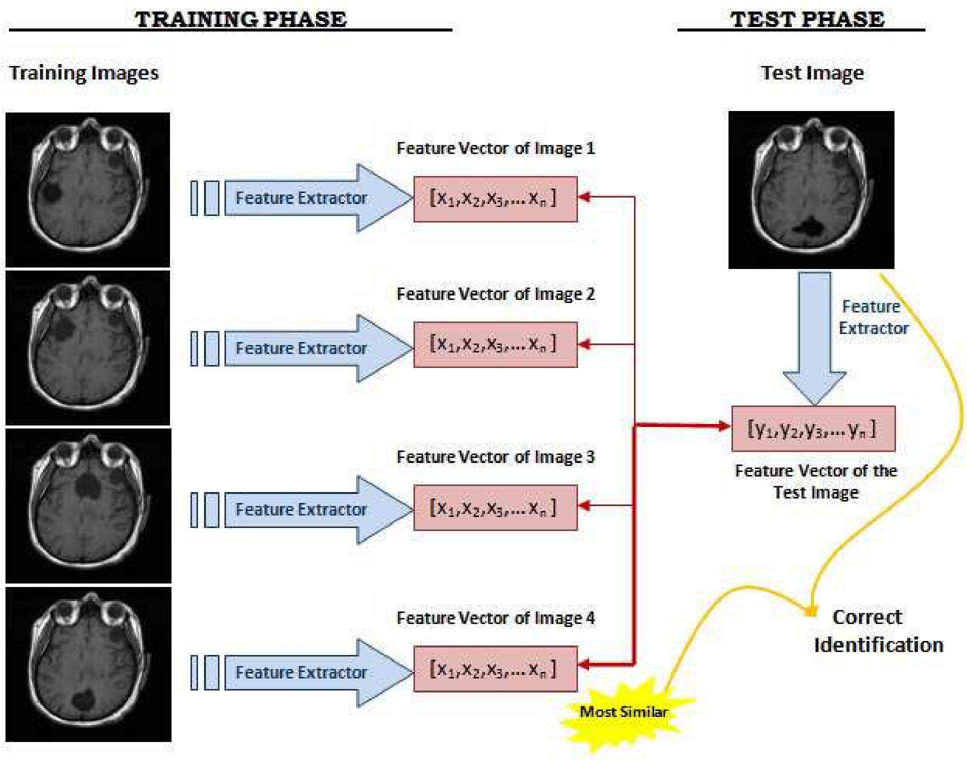


Abb. 6.6: Schematic diagram of a MR image recognizer

function was used as a smoothing factor and the with the different spread value they achieve different accuracy from 73% to 100%.

4 Conclusion

Overall we can say every modality of imaging has its own practical advantages and limitations so we need multi-modal approaches to have an high accuracy when dealing with the human life. So the combining of image fusion and neural networks is interesting for medical imaging because several scopes to apply both e.g. feature processing, feature extraction or making decision.
At least we can say that the current high technologies medical imaging would not exist without neural networks and image fusion. In addition there is to say that application of PNN is not fully utilized yet so it is an interesting and forward-looking field of research.

Literaturverzeichnis

[1] B. V. Dasarathy A.P. James. Medical image fusion: A survey of the state of the art. 2014.

[2] F.C.Maorabito S.B.Serpico G.Simone, A.Farina and L.Bruzzone. Image fusion techniques for remote sensing . Technical report, University of Trento, 2002.

[3] Michael I. Jordan and Christopher M. Bishop. Neural Networks. 1996.

[4] Berthold Krevert. Registrierung.

[5] D. Kriesel. Ein kleiner ́Uberblick ́uber Neuronale Netze. 2005.

[6] Muhammad Sharif Mussarat Yasmin and Sajjad Mohsin. Neural networks in medical imaging applications: A survey. *World Applied Sciences Journal*, 22:85–96, 2013.

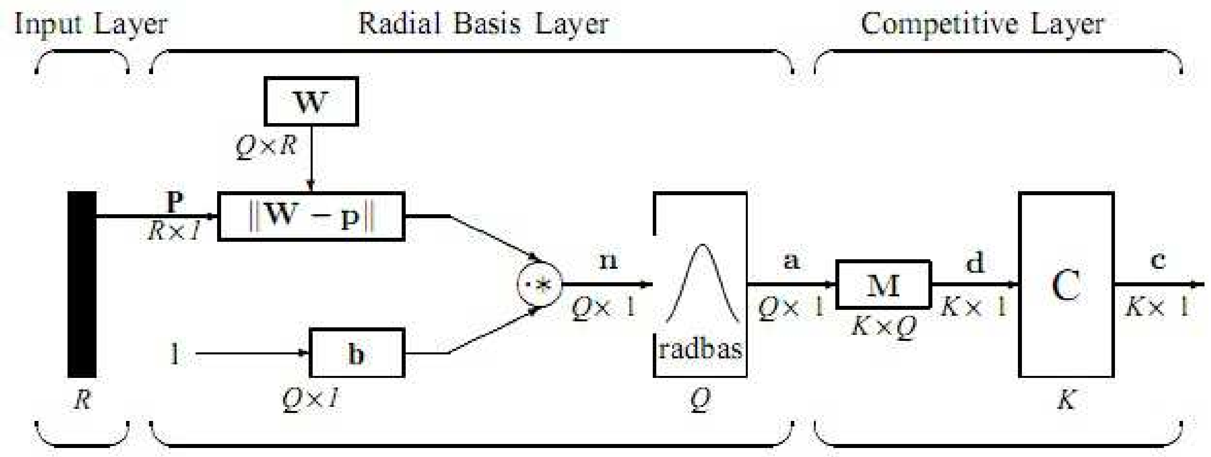


Abb. 6.7: Structure of the network

[7] M.F. Othman and M.A.M. Basri. Probabilistic neural network for brain tumor classification. In *Intelligent Systems, Modelling and Simulation (ISMS), 2011 Second International Conference on*, pages 136–138, Jan 2011.

[8] Deepak Kumar Sahu and M.P. Parsaii. Different image fusion techniques a critical review. In *International Journal of Modern Engineering Research (IJMER)*, volume 2, pages 4298–4301, Sep 2012.

Review - Application of the Ring Theory in the Segmentation of Digital Image

Ligia Bildea

1 Introduction

Image segmentation is representing the technique of dividing an image into multiple segments such that the image becomes more easy to process and analyze. Due to its importance in image processing which is used in different areas such as imaging, machine learning, biology etc. this procedure took the interest of many researchers. Today we are facing with a lot of algorithms that improve just parts of the image segmentation, specializing themselves in some particular task. None of the techniques succeeded to return the best performance in all the cases as a result of the enormous number of features in a picture. This paper is based on the [2] and its aim is to explain and to review the idea presented by the authors. The reviewed paper is introducing a new stopping criterion for the Mean Shift algorithm with the purpose of improving the segmentation.

Mean Shift algorithm was invented in 1975, but it was not widely used until it was applied the computer vision and obtained very good results. Therefore, many researchers focus on involving it as much as possible in the image segmentation task.

The authors are concentrating on doing a similar thing, hence they propose a new similarity index among images based on Ring theory and entropy function. This index is the new stopping criterion specified before.

Each section of this paper will contain a brief explanation of the information found in the handed-in paper followed by a personal opinion related to it. In the first section, the purpose of the paper is discussed as well as the motivation. Once the questions about what the authors are going to do and why are answered the theoretical aspects are going to be presented. In the next step, section 4 will throw a light on the similarity index with entropy function. Having all the theoretical concepts needed explained, we continue with the experiments and the results. At the end the conclusion is pointing out the most important ideas drawn from the handed-in paper.

2 The purpose of the paper

The goal of the paper can be easily depicted already from the abstract. The aim of the authors was to reach a better performance for segmentation process in digital images than the performance achieved by [7, 8, 9, 6] and a more stable algorithm.

The solution proposed, was to use a new stopping criterion, based on $\mathbb{Z}_n$ ring and entropy function, for the Iterative Mean Shift algorithm. The authors claim that the usage of Ring Theory for the Iterative Mean Shift algorithm, together with the new stopping criterion has a good performance and a better stability. Between the ideas which motivate the solution of the paper, we found on the first place the inflexibility of the existing methods, due to the complex variation in shape of the objects within an image. Techniques such as thresholding, histograms or other traditional methods are consider rigid methods by the authors, because of their complexity in automation process of the classical approximation. As a consequence the segmentation process, in this case, is based on Mean Shift algorithm, considering the good results provided by it in the image processing domain.

The reason presented for choosing the entropy function is represented by its importance in image data. Entropy function is a fundamental concept in the information theory. It is claimed that, if the pictured are defined by means of Ring Theory, the properties of entropy can be increased.

Since the authors want to consider the spatial information they use the Ring Theory. In this case all the images are represented as matrices with elements belonging to the cyclic ring $\mathbb{Z}_n$. In addition, Ring Theory has also been successfully used in cryptography and other computer vision tasks.

Regarding the goal of the paper, due to the lack of further explanations I will categorized the goal as vague. This affirmation is based on [5], where it is presented that a better segmentation relies on the purpose of the application, on the features of the input image and on the knowledge of the user, none of which being specified here. Being dependent on such a multitude of factors, it is as well the reason why there exists no perfect segmentation that can be applied in all the cases.

In what regards the motivation for choosing to represent the images as matrices with values belonging to $\mathbb{Z}_n$ ring, the authors mention that by the help of this method, the information it is not lost, the negative values are not set to 0, but they are wrapped up.

3 Theoretical aspects

The paper presents in section 2 the theoretical aspects that concern the algorithm used and the new similarity index found. They are meant to help the reader to understand better the basic concepts, underlying the solution proposed by the authors.

In the beginning, Mean Shift basic aspects are highlighted. The history of it starts in 1975 when Fukunaga and Hostetler [1] invented this algorithm. It didn't caught too much of the attention at that time, therefore it was not widely used until it was applied in Computer Vision tasks. The performances achieved were exceptional and they had as a consequence the propulsion of the interest for this algorithm in the top. As a reader can depict from the paper itself the Mean Shift is a non-parametric iterative algorithm which maps the feature space to empirical probability density function and uses a generalized kernel approach.The logic behind it is the following: the algorithm takes an images and converts it into feature space. In this case the feature space is 1 dimensional, because the authors consider just the intensity values. Further, the search windows are distributed over the feature space and the mean value is computed for each of them. Next step is to shift the center of the windows to the mean value and algorithm is repeated until it converges. After each iteration, the windows are considered to be moved into a denser region of the feature space. Whenever more windows reach the same location they are merged.

The authors use a mathematical approach in order to explain the Mean Shift concept, starting with the formula of the estimation of the probability density function f(x) and continuing with Epanechnikov Kernel formula which is applied to compute f(x). Step by step, using trivial calculation, they reach to the Mean Shift Vector formula. The computation presented in here is utilizing the general case of d - dimensional space, but it is mention that only the case d equals 1, corresponding to gray level images, will be taken into account because of simplicity reasons. This part of the paper is representing the exact same work described in the previous papers [7, 9, 6] published by some of the same authors, but in this case without having any reference to them. In other words, even if the authors mentioned a revision of the Mean Shift theory as an improvement of the paper with regard to [11], in fact we find just a copy of a section that was contained in previous papers published by them.

The next subtopic portrayed is related to the theoretical aspects of entropy. In brief the entropy function, proposed by Shannon [10], is a measure of unpredictability, which obtains the lowest value 0 within a totally uniform region. This case is utopian because in practice the images present all the time noise. Thanks to the fact that it can be seen as a measure of disorder in a system, it was successfully used as stopping criterion for the Mean Shift algorithm.

4 Similarity index with entropy function

The section 3 puts the accent on the similarity index, which is defined by the use of the entropy function. In this moment, the authors bring concrete arguments about the motivation underlying the paper. The question they want to answer is why the entropy function is not a sufficient stopping criterion for the Mean Shift algorithm. In order to explain it, they make use of an example which shows that the entropy function will return the same values in the case of two pictures having the same pixel distribution, even if they are different from the point of view of the images. Based on this fact, the authors claim that the similarity index between two images calculated as the difference among the entropy of the two images will led to wrong results when the difference among them is computed. The solution they present is a combination of the entropy function and Ring Theory such that the spatial information between the images is also taken into account .

Next topic introduced is the Ring Theory with respect to images. As the reader can find in the definition given in the paper, the $\mathbb{Z}_n$ Ring is a partition of $\mathbb{Z}$ in which the elements are related by the congruence module n. Hence, the authors are defining the images as matrices with the values of the pixels belonging to the cyclic ring $\mathbb{Z}_n$. After a very basic theory introduction which starts with the definition of a $\mathbb{Z}_n$ ring and continues with the theorem of the addition and multiplication with respect to a ring structure, the trivial proof of the theorem and the properties of the ring are introduced. These are the preceding concepts based on which the strong equivalence in images is defined. Furthermore, the equivalence class concept is introduced and this part of the paper is concerned with proving that two images are in the same class of equivalence if and only if the two images are strong equivalent. This result also helps in the next definition of Natural Entropy Distance which will play the role of the new stopping criterion for Mean Shift. In the following I will enunciate its definition, as found in the hand-in paper.

Definition 1 *Let C_A and C_B be two elements in the quotient space $\frac{G(\mathbb{Z})}{N}$, $A_1 \in C_A$ and $B_1 \in C_B$ are images. The Natural Entropy Distance between A_1 and B_1 is defined by*

$$\widehat{v}(A_1, B_1) = E(A_1 + (-B_1)) \tag{7.1}$$

In this segment of the paper, the reader can observe the transition from very basic explanation of some of the concepts, to a very abstract one for some of the main concepts. For example, in the case of Ring Theory there are given trivial demonstrations for the theorems used, but not such a clear and easy explanation when it comes to the formula of the new stopping criterion.

5 Experiments and results

The experiment results, together with the comparison and the discussion are detailed in section 4 of the handed-in paper.

The method chosen to demonstrate the benefits of the new stopping criterion is the comparison with an old stopping criterion which relies on the Entropy function. The distinction between the two criteria is that in the case of the new stopping criterion, first a difference between the images is performed and after that the entropy function is applied. For the old criterion the entropy values of the two images are determined and the difference between them is computed.

The test images which are taken into account are based on their frequency. First one is called ”Birdänd it belongs to low frequency category. For high frequency the picture called ”Baboonïs tested and representative for a combination of low and high frequency is the image Montage”. If a picture is belonging to the low frequency category it means that the pixel values in that image are changing slowly over space, at the opposite side for high frequency images the value of the pixels changes fast over space.

Among the comparisons conducted, the reader can find the profiles of the segmented image ”Birdöbtained using each criterion at a time. About the graphics attached, it is pointed out that they correspond to equal intensity level. The outcome highlighted by the authors is the achievement of a better segmentation using the new stopping criterion. Their claim emerges from the fact that in the same region of segmentation, the profile, when the new criterion was used, has less variation of pixels intensities. The second comparison is related to the performances of the Mean Shift algorithm using the new, respectively old criterion. The authors focus on the less homogenous areas found in the pictures resulted by applying the old stopping criterion, which are emphasized by mean of arrows. As a consequence to the better homogeneity found in the images obtain when the new stopping criterion was used, the conclusion that the new stopping criterion is better then the old one was drawn.

In the next sub-sections of the paper a personal opinion will be given regarding the experiments and the results presented.

5.1 General Overview

The first struggle that the reader faces is that the purpose of using the new stopping criteria is very unclear. If in section 4 and in the section 5 the authors mention that the new stopping criterion for MSHi algorithm is more stable than the old criterion, in section 4, in the description of the measures performed, the authors claim that the new criterion gives a greater instability to the MSHi.

The general problem that persists in this paper is that the results are roughly described by the authors. The diagrams that are presented have no axis titles and the reader has to search in the text what the axis represent. The lack of details lowers the trust in the results presented.

In the literature belonging to the same topic, the method most used, which follows the diagrams, is displaying the performances obtained in a table. It should contain the best values achieved and also the

Abb. 7.1: The results diagram presented in the reviewed paper.

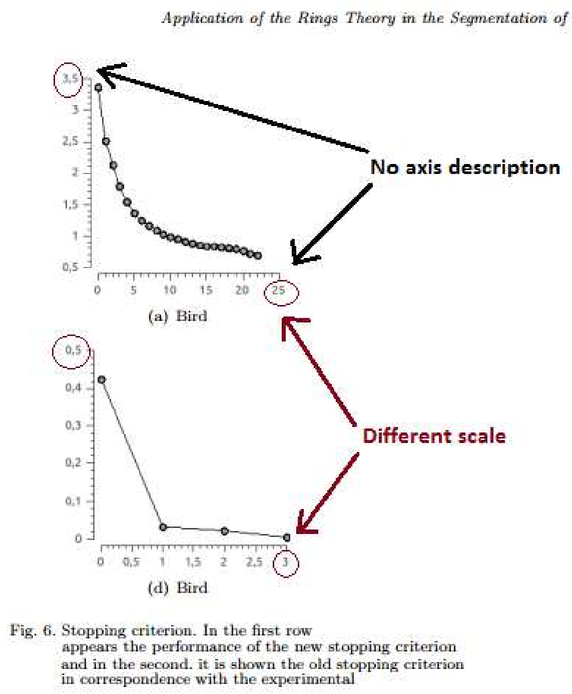

Application of the Rings Theory in the Segmentation of Digital Images 15

Fig. 6. Stopping criterion. In the first row appears the performance of the new stopping criterion and in the second. it is shown the old stopping criterion in correspondence with the experimental

worst one in comparison, in this case, with the other stopping criterion. We can find example of these representation in [12, 4] and many others. In the same time, in papers that make a parallel between two or more methods, the authors always specify the type of machine that was used to perform the tests. Here nothing is specified concerning this subject.

The experiment results that sustain the new approach for image segmentation, proposed by the paper aim to prove that using the new stopping criterion, the segmentation process gives more homogeneous results. For this purpose, there are provided the original versions of the images and the versions of segmented image using the old criterion and the new one. The differences are pointed out using arrows and a small text description.

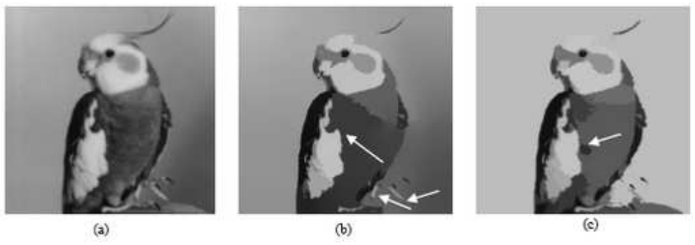
Furthermore, in the final measures for comparing the two stopping criteria using the experimental images, different scales for the diagrams are used. There exist no detailed explanation why and how this sustains the performance of the new stopping criterion based on Natural Entropy Distance. The only thing which is mentioned is that on the diagram corresponding to the new criterion applied to MSHi, the behavior is more smooth and therefore the algorithm is more stable. In Figure 7.1 , a part of the results was taken from the paper to underline the problems found.

5.2 Algorithm Overview

The Iterative Mean Shift algorithm is a technique mostly known in literature for its good performance in fulfilling the clustering tasks[3]. In the case of image segmentation it also provides reliable solutions. It seeks the modes in a feature space composed of spatial and color information. The results obtained with this method are efficient, the only drawback that can appear in case of some inputs is the calculation speed. Obtaining such promising results, led in the last years to many optimizations techniques able to improve the calculation time. Some of them are oriented on using specialized test datasets and others, with the risk of affecting the accuracy of the algorithm output, are using different heuristics. In this case, it is used the Iterative Mean Shift algorithm using Natural Entropy Distance as stopping criterion.

Moreover, when an Iterative Mean Shift algorithm is used, the quality of the segmentation relies also on the value of input parameters hs(spatial resolution) and hr(range resolution)[5]. If the spatial resolution hs increases, then only the features with a large spatial support are present in the segmented image. If the range resolution is large, then the features that have a high contrast will be represented in the segmented image. Regarding this scientific work the values for hs and hr are 12, respectively 15. There is no specification why these values are chosen, even though in [5] it is claimed that the bigger

Abb. 7.2: Results taken from [6] a)Original Image b) Image segmented using MSHi with entropy as stopping criterion hs=12, hr=15, threshold=0,001 (9 iterations) c)Image segmented using MSHi with other stopping criterion- this is not relevant for the future discussions.



the window sizes are the uniformity increases. It is good to mention that this case doesn‘t apply to the number of iterations, because for them the value oscillates.

5.3 Results Comparison

In [6] the Iterative Mean Shift algorithm is using the entropy as stopping criterion. The window sizes are hs = 12 and hr = 15 and the stopping threshold is set to 0,001. Utilizing this configuration it is shown that even if the images are more complex, and the initial entropy value is bigger, the convergence was always reached in 8 to 10 iteration. In the case of the Bird, the number of iterations needed were 9 and the segmented image is presented in Figure 7.2 .

In this paper the new stopping criterion is compared with the criterion presented before which is referenced as old criterion. For this purpose, the settings for the old criterion are hs=12, hr=15 and the threshold is equal to 0,01. The result in case of the Bird can be seen in Figure 7.3. In order for the convergence to be reached there were 3 iterations performed using the old criterion.

What I want to emphasize here is that, the same algorithm, with the same window sizes, but different threshold, therefore different number of iterations, leads to different results. After 3 iterations in the case of Figure 7.3 c) the support on which the bird is staying is still visible, in the Figure 7.2 b), after 9 iterations the support is matching the wall and it is not visible anymore. The conclusion that can be taken is that depending on the threshold value, for the old criterion, the image becomes more homogeneous. The question which is without an answer in this paper is why that value is taken for the stopping threshold for the old criterion.

6 Conclusion

The new similarity index proposed, used as stopping criterion for Iterative Mean Shift algorithm was reviewed in this paper.

Even from the beginning, the purpose and the motivation underlying the handed-in paper were established and discussed. Further, the theoretical aspects needed were introduce to determine the mathematical arguments that sustain the better performance obtained by Mean Shift algorithm using the new

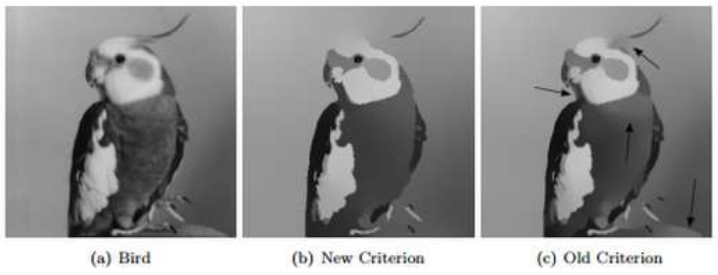


Abb. 7.3: Results taken from the reviewed paper a)Original Image b)Image segmented using MSHi with NED as stopping criterion hs=12, hr=15, threshold=0,9 c)Image segmented using MSHi with entropy as stopping criterion hs=12, hr=15, threshold=0,001 (3 iterations).

stopping criterion. The experiments concluded and the results obtained were presented, as well as a personal opinion about them.

The main points that are still unclear are if this method can be applied on different window sizes obtaining the same performance. If the number of iterations is so big, while using the new criterion, isn't that a drawback from the point of view of the time needed to process the segmented image?

The paper has very vague explanations of the results, beside that, it is full with grammatical or spelling mistakes which lead to a very difficult understanding of the idea. In the same time, these factors also lower the trust in the performances achieved.

Furthermore, the work contains many information found on the previous papers published by the same authors and even though big chunks of text are taken completely from there, there is no specification.

All in all, even if a new similarity index was found we can not be sure if the results are reliable or not.

Literaturverzeichnis

[1] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, 1975.

[2] Yasel Garcés, Esley Torres, Osvaldo Pereira, Claudia Pérez, and Roberto Rodríguez. Application of the ring theory in the segmentation of digital images. *arXiv preprint arXiv:1402.4069*, 2014.

[3] Jens N. Kaftan, AndrÃ© A. Bell, and Til Aach. Mean Shift Segmentation Evaluation of Optimization Techniques. In *Proceedings of the Third International Conference on Computer Vision Theory and Applications, VISAPP 2008*, pages 365–74, Funchal, Madeira - Portugal, January 22–25 2008. INSTICC - Institute for Systems and Technologies of Information, Control and Communication.

[4] Sang Uk Lee, Seok Yoon Chung, and Rae Hong Park. A comparative performance study of several global thresholding techniques for segmentation. *Computer Vision, Graphics, and Image Processing*, 52(2):171 – 90, 1990.

[5] Roberto Rodríguez Morales, Didier Domínguez, Esley Torres, and Juan H Sossa. Image segmentation through an iterative algorithm of the mean shift. 2012.

[6] Roberto RodrÃ̃guez, AnaG. Suarez, and JuanH. Sossa. A segmentation algorithm based on an iterative computation of the mean shift filtering. *Journal of Intelligent and Robotic Systems*, 63(3-4):447–63, 2011.

[7] Roberto Rodríguez. Binarization of medical images based on the recursive application of mean shift filtering: another algorithm. *Advances and applications in bioinformatics and chemistry: AABC*, 1:1, 2008.

[8] Roberto Rodríguez, Esleys Torre, and Juan H Sossa. Image segmentation via an iterative algorithm of the mean shift filtering for different values of the stopping threshold. *International Journal of Imaging and Robotics*, 7(1):27–43, 2012.

[9] Roberto Rodríguez, Esley Torres, and Juan H Sossa. Image segmentation based on an iterative computation of the mean shift filtering for different values of window sizes. *International Journal of Imaging and Robotics*, 6(A11):1–19, 2011.

[10] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–56, 1948.

[11] Yasel Garcés Suárez, Esley Torres, Osvaldo Pereira, Claudia Pérez, and Roberto Rogríguez. Stopping criterion for the mean shift iterative algorithm. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 383–90. Springer, 2013.

[12] Y.J. Zhang. A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8):1335 – 46, 1996.

Tracing Neural Fibers from DT-MRI Data

Lukas Boersma

Zusammenfassung

This report gives an overview of neural fiber tractography from DT-MRI scans. A simple approach for reconstruction of neural fibers from such 3D scans is presented, and an overview of the current state of the art in this field of research is given.

Keywords: diffusion tensor imaging (DTI), neural fiber tracing, fiber trajectory, fiber tracking, tractographx

1 Introduction

For a long time, the human brain was a largely unexplored area in medical research. Today, the recent launches of the *Human Brain Project* by the European comission and the *Brain Activity Map Project* by the US executive office show how big the public interest in more knowledge about the human brain still is. Advances in technology like 3D scanning techniques have made it much easier to research the structure and biological mechanisms of the brain. Apart from research, 3D scanning of the brain structure also is relevant for clinical purposes, because certain neurological disorders like Alzheimers alter structures in the brain, and the presented method allows to detect such changes in brain structure.

1.1 General Overview

The human brain is not only a highly complex organ, but also a highly structured one. Inside the brain, a number of separated regions exist. Inside of these regions, neural cells (classified as *gray matter*) perform the computational and processing tasks that make up the capabilities of the brain. These structures of the brain are interweaved with a network of long fiber strands of neural cells classified as *white matter*, connecting the different parts of the brain to each other and to the rest of the body.

This article surveys an algorithm first presented in 2002 [zhukov2002] that helps to understand the structure of the human brain by reconstructing neural fibers from 3D DT-MRI scans for visualization and other purposes (cf. figure 8.1).

It should be noted that while the algorithm presented here originates from the year 2002, neural fiber tracing is still an active field of research. In section 1.3 we give an overview of the current state of art and some more recent related works.

1.2 Problem Statement

With the availability of DT-MRI scanning devices, researchers and doctors are now able to retrieve 3D data from tissues with information about the diffusion tensor properties (cf. section 3.1) inside these tissues. This information distinguishes tissues that are hard to separate in traditional MRI data. However, because of the high complexity of the diffusion information, it is hard for a human observer to interpret this data.

The goal of the presented approach is to take DT-MRI scans of brains as input and output reconstructed pathways of neural fibers. It can thus be seen as a visualization technique of data that is otherwise hard to interpret. However, the 3D pathways could also be the basis for further processing, like the creation of connectivity maps in the brain that indicate how strong different regions in the brain are connected.

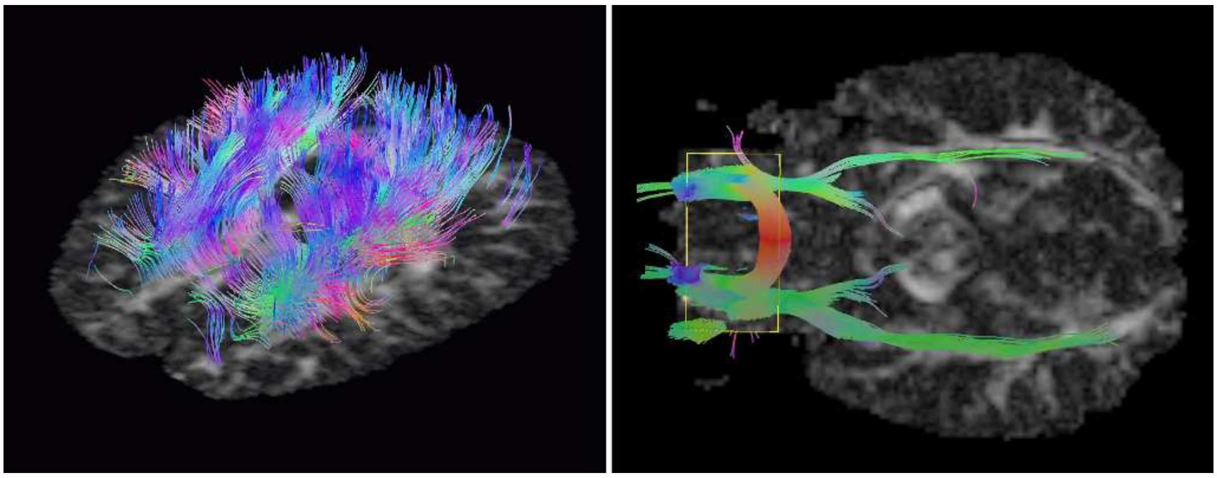


Abb. 8.1: The presented algorithm extracts neural fibers (visualized here as colored lines) from 3D scans of the human brain (visualized here as grayscale 2D slices in the background). [zhukov2002].

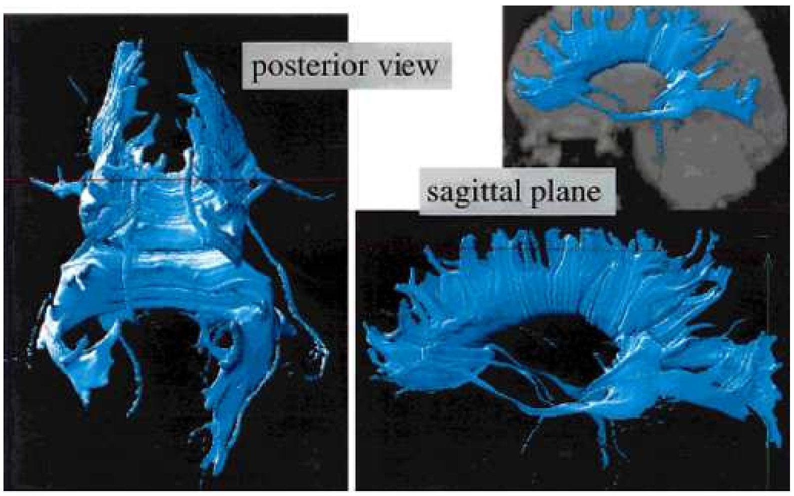


Abb. 8.2: Apart from reconstructing single fibers, isolating 3D volumes corresponding to fiber strands are another possibility to visualize fiber strands. [basser2000].

1.3 State of the Art

The algorithm presented in section 3 traces fibers in the input data to create 3D line segments that can be visualized to give a better understanding of the fiber structures. This algorithm, together with the one presented in [mori1999], is one of the first to follow this approach on reconstructing neural fibers.

Several newer tracing approaches exist that try to improve on the algorithm presented here. For example, in [lazar2005], biological constraints are incorporated into the algorithm for better results.

Another improvement of the presented method was introduced by Andrew Zalesky [zalesky2008], who proposes an algorithm that traces fibers in a non-greedy way to achieve results that are better on a global scale. Instead of retracing fibers, he uses the same measures used by the local decisions in previous works to assign weights between the voxel grid cells, which is then interpreted as a graph and used for a shortest path optimization to retrace all the fibers.

Apart from fiber tracing algorithms, other works [brun2003] focus on coloring and visual rendering of the 3D data to improve visibility and perception of the underlying fiber structure for human viewers, or on extracting the 3D volumes that are likely to belong to neural fibers [basser2000, wenger2004] (cf. figure 8.2).

Some more recent works present approaches to preprocessing data coming from scanning devices with lower resolutions that are common in clinical environments [fillard2007, chen2005]

2 Mathematical Background

The input data used for the algorithm presented in section 3 will rely on a *tensor field* as input data. Tensors can be thought of as a generalized form of matrices, where higher dimensions than two are allowed.

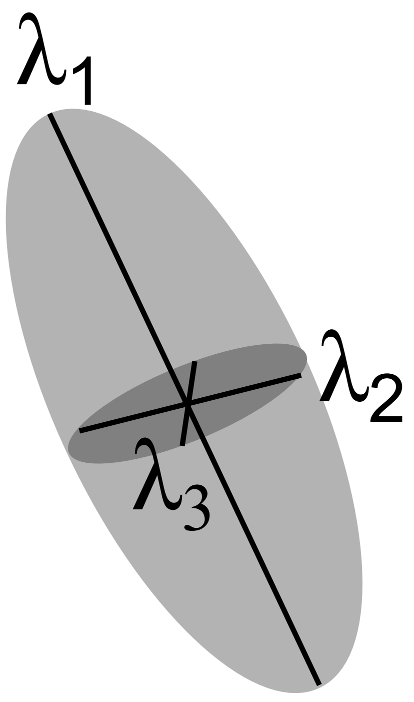


Abb. 8.3: A 3D tensor visualized as an ellipsoid with the tensor's eigenvalues as axes.

In this paper, we will use symmetrical tensors of size 3x3, which is equivalent to a 3x3 matrix:

$$T \in \mathbf{R}^{3\times 3} \tag{8.1}$$

For our purposes, we define products of tensors and vectors as the usual matrix product, leading us to a straightforward definition of eigenvectors for our tensors, being the vectors e_i for which eigenvalues λ_i exist so that:

$$Te_i = \lambda_i e_i \tag{8.2}$$

Because we only use symmetrical tensors, there are always three eigenvectors and the eigenvectors form an orthogonal vector basis. For consistency, we assume that their indices are sorted by the eigenvalues as $\lambda_1 \geq \lambda_2 \geq \lambda_3$, so e_1 always has a maximal eigenvalue and e_3 always a minimal one. For the rest of this article, we will visualize tensors as ellipsoids with the tensor's eigenvectors as their axes (cf. figure 8.3). This makes sense because the tensors used in the rest of this article measure the diffusion of molecules, which is visualized nicely with this method (cf. section 3.1).

For our algorithm, we will be interested especially in tensors which have one large eigenvalue and two smaller ones. These tensors would appear long and thin (*cylindrical*) in our ellipsoid visualization. We introduce a measure c_l to classify these tensors:

$$c_l = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2 + \lambda_3} \tag{8.3}$$

So, for long and thin ellipsoids c_l would approach to equal one:

$$\lambda_1 \gg \lambda_2 \simeq \lambda_3 \tag{8.4}$$

When c_l approaches zero, the corresponding tensors would appear as spherical ellipsoids, with all three eigenvalues almost at the same value.

3 Methods

In the following, we give an overview of the process for generating fiber pathways out of DT-MRI scans.

3.1 Input Data

For retrieving our input data, we will use a modified version of the well known *magnetic resonance imaging* (MRI). MRI scanning works by applying a high-frequency magnetic field around the body, which then induces a *magnetic resonance* of molecules (mostly the water molecules) in the body for each pulse in the magnetic field, which can be measured. The strength of the resonance varies for different types of tissues, and can thus be used to distinguish between certain types of tissues in the body.

3.1.1 Diffusion Unfortunately, white and gray matter is hard to distinguish in MRI scans. Because of this, we will exploit that the different structure of both tissues results in different patterns of *diffusion*.

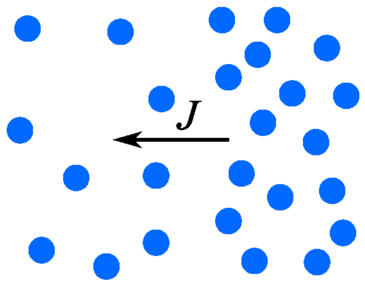


Abb. 8.4: Without obstacles, molecules in a fluid will show movement in the opposite direction of the concentration gradient J.

Diffusion describes the fact that in a fluid, molecules will tend to move in the opposite direction of the concentration gradient, thus resolving differences in concentration (cf. figure 8.4). This is stated in Fick's first law:

$$J = -D\nabla p \tag{8.5}$$

Where J is the flux, D is the *diffusion tensor*, and ∇p is the concentration gradient. Different values for D can not only account for different scales of the flux, but also account for restrictions of diffusion in certain directions (cf. figure 8.6).

The diffusion tensor D indicates the amount of movement of water molecules in different directions. We can interpret the eigenvalues of the tensors as the strength of diffusion in the respective eigenvector direction. In other words, if the water diffuses mostly in one specific direction (i.e. in an *anisotropic* way), we would get one large eigenvalue and two small ones. Likewise, if the diffusion is the same in all directions, we would expect the three eigenvalues to have roughly the same value.

3.1.2 DT-MRI DT-MRI extends traditional MRI scanning by using magnetic fields with a gradient in field strength. When a water molecule moved along or against the gradient between two measurements, the magnetic resonance will be weaker or stronger in the second measurement, respectively. After repeating measurements with gradients in several directions, we can then construct the diffusion tensor using a simple least squares estimation.

Because the neural fibers we are looking for consist of very long and thin cylindrical cells (cf. figure 8.6), we would expect to cause neural fibers to appear as highly anisotropic data in our DT-MRI scans. For the algorithm presented in the following, we assume that all points with high anisotropy correspond to either neural fibers or noise.

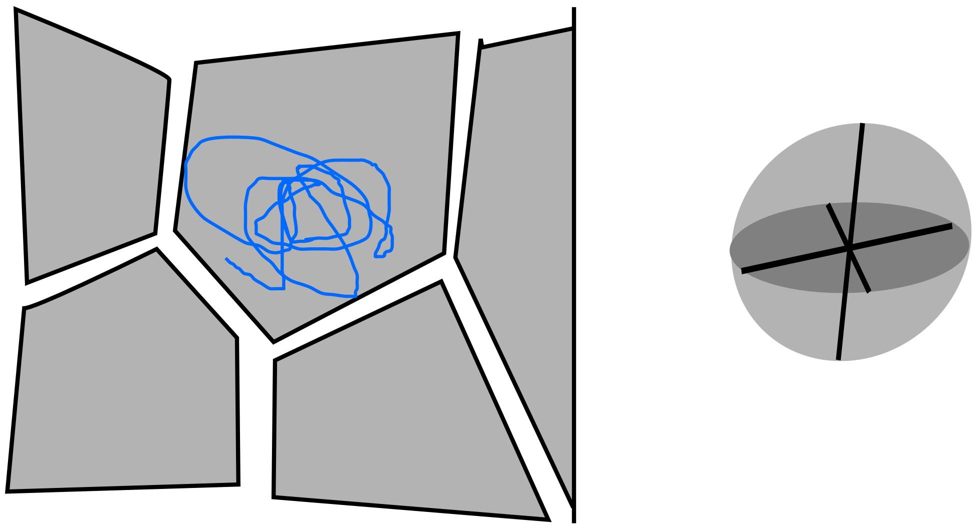


Abb. 8.5: In tissues where cells are not cylindrically shaped in one main direction, the average diffusion tensor for a certain region will be spherical, because on a larger scale, water diffuses mostly isotropically.

3.2 Tensor Sampling

The DT-MRI scans output 3D tensor fields, so the data is a grid of Tensor matrices, giving a 3 times 3 matrix for each grid cell with coordinates (i, j, k):

$$T_{i,j,k} \in \mathbf{R}^{3\times 3} \tag{8.6}$$

For for the following filtering and tracing steps, we will need to sample from arbitrary positions in $\mathbf{R}^3$ instead of integer grid cells. We achieve this by simply linearly interpolating the tensors component-wise, we get a tensor-valued function for every position $x = (x_1, x_2, x_3)$ in our sampling domain:

$$f_T : \mathbf{R}^3 \to \mathbf{R}^{3\times 3} \tag{8.7}$$

Where $f_T(x)$ is simply a linear combination of all eight neighboring $T_{i,j,k}$ of x, barycentrically weighted by their distance to x.

3.3 Noise Filtering

To improve robustness against noise, we will have to filter the input data first. Instead of filtering the whole input data by smoothing or other approaches, the authors of the algorithm propose a local *moving least squares* approach, where instead of sampling from the input data, a polynomial is locally fitted to the input data and then evaluated instead of sampling from the actual input data.

This moving least squares approach is a local optimization approach that, for a given sampling position $x \in \mathbf{R}$, fits a polynomial f_F to approximate the local neighborhood of x. This is done by minimizing the energy E:

$$E(x, f_F) = \int_{\mathbf{R}^3} G_x(t, f_T(x)) \cdot [f_F(t) - f_T(t)]^2 dt \tag{8.8}$$

Where G is a weighting function that weights points closer to x more than points far off, and thus defines the shape of the region for which f_F is fitted. Loosely speaking, this energy expresses the integral of the squared differences between f_F and f_T, weighted by G.

When sampling from a point x, the following minimization problem has to be solved to obtain f_F:

$$f_F = \text{argmin}_{f_F}(E(x, f_F)) \tag{8.9}$$

The authors recommend to use a polynomial of some degree up to three for f_F. The minimization problem is then discretized and solved by setting the first derivative to zero.

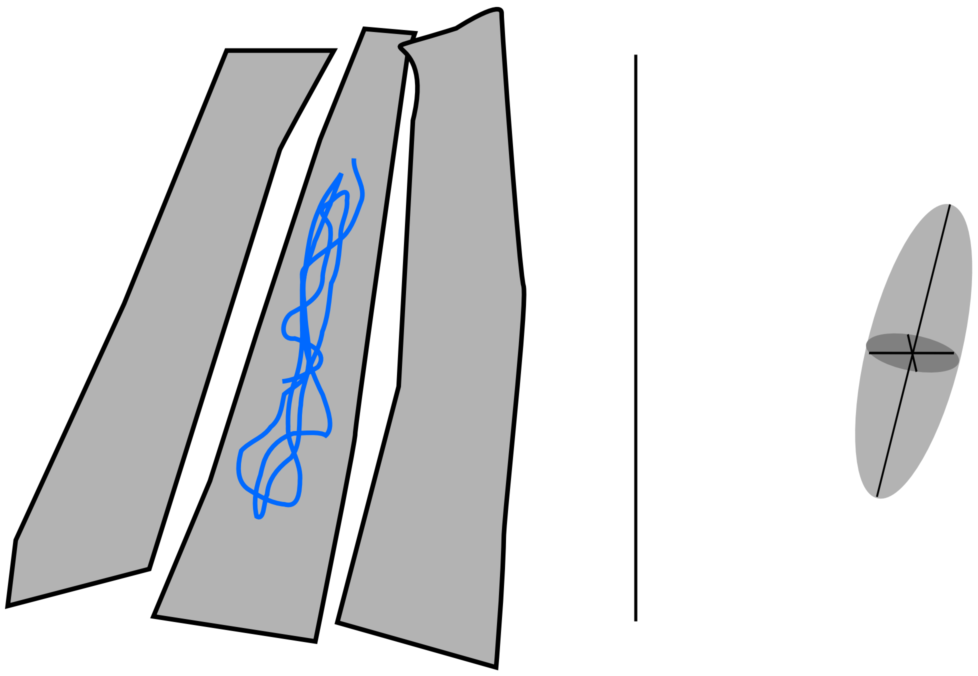


Abb. 8.6: Because of the shape of neural fiber cells, diffusion is mostly restricted to one main direction, which results in tensors that can be distinguished from tensors corresponding to gray matter.

3.4 Fiber tracing

The algorithm works in an interactive way. The user has to determine a threshold for c_l for the fiber initialization and a plane on which all points that exceed this threshold should be initialized as fibers. The actual tracing procedure then is straightforward: Starting from the initial points, we follow the longest eigenvectors until c_l drops below a certain value where we can no longer assume a neural fiber in the data. The tracing is done in steps of the grid's size, and turns above 90 degrees result in the end of the fiber. The algorithm's authors also suggest to remove fibers below a certain length for more robustness against noise. Figure 8.7 shows a sketch of the tracing process.

4 Results

The images shown in this article all come from input data with a grid of 121x88x60 voxels, thus with a scanning resolution of about 0.7cm. This resolution appears to be rather low compared to the high inner complexity of the brain, but figure 8.8 shows that the algorithm can reconstruct at least most of the more significant fiber strands in the brain.

In practice, the resolution of the 3D data is usually not high enough to expect all the actual neural cells to be represented by single traced fibers in the output data.

So, although we talked about fiber *reconstruction* as being the purpose of the algorithm, we can not expect to get exact data about actual single neural cells, but we can expect to get a good visualization of the fiber *strands* in the brain. Figure 8.8 and 8.9 show schematics of known fiber structures and compare them to the tracing results of brain scans.

Figure 8.10 shows a comparison between the algorithm's results with and without applying the MLS filter described ins section 3.3.

DT-MRI scans can also be useful outside of the brain. Figure 8.11 shows the results for a spinal cord, which also contains neural fibers like in the brain.

5 Discussion

Unfortunately, we were unable to find any clinical studies or even simple surveys that try to measure the accuracy and robustness of the presented algorithm by other means than showing various images as results. Thus, we can not provide any quantitative measures of the algorithm's quality.

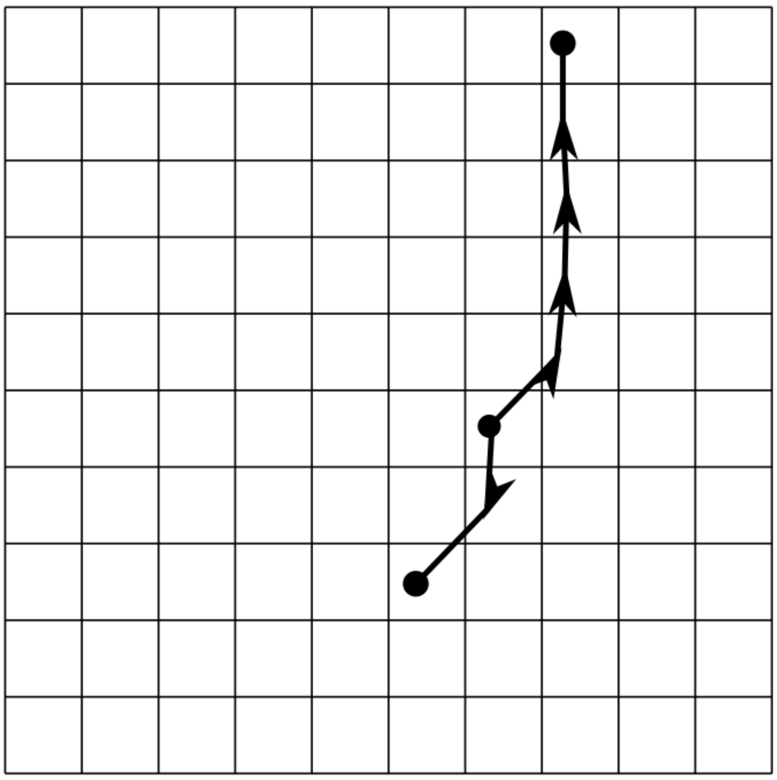


Abb. 8.7: The fibers are reconstructed by tracing along the directions of the eigenvectors of the tensor field.

Furthermore, we were unable to find any studies that try to verify that the results actually correspond to the real structures inside the brain. The fact that altered versions of this method are widely used in practice could be seen as an indication that the results are good enough for practice, but there seem to be no efforts to actually verify that.

Obviously, the quality of the reconstructed fibers heavily depend on the resolution and quality of the input data. The resolution in which 3D scans can be obtained depends on the accuracy of the scanning devices that are used. The algorithm can output traced pathways in the same resolution of the input data, so resolution issues are more a question of the input device that is used. To a certain degree, this is also the case for noise issues, because the signal-to-noise ratio in the input data can usually be improved by using the magnetic fields with higher Tesla values.

Because of the low resolution, we can not expect to get exact fiber reconstructions regardless of the method used. However, the results achieved with this method can still be of use for determining which parts of the brain are connected to each other and how strong these connections are. Other purposes could be to use the reconstructed fiber pathways for further processing, for example to map the “signal distance” between different parts of the brain.

In figure 8.10, we can see that the filtering smoothes the traced fibers and reduces the amount of fibers that seem to be incorrectly traced. We have, however, no way to measure the actual increase of accuracy, so we can only say that the filtering gives a visually more pleasing result. Also, the authors do not explain why they deem polynomial fitting to be more adequate that traditional filtering approaches, and how one could be sure that no new peaks and artifacts can occur when fitting polynomials of higher degrees to arbitrary input data.

It should be noted that because the algorithm assumes that any tensor with a c_l measure above the threshold belongs to a neural fiber strand, it gives poor results for scans of other tissues than the human brain, where other types of body cells have cylindrical diffusion properties, too. Apparently, the polynomial fitting approach is unable to cope with the high amount of noise present in bone structures like

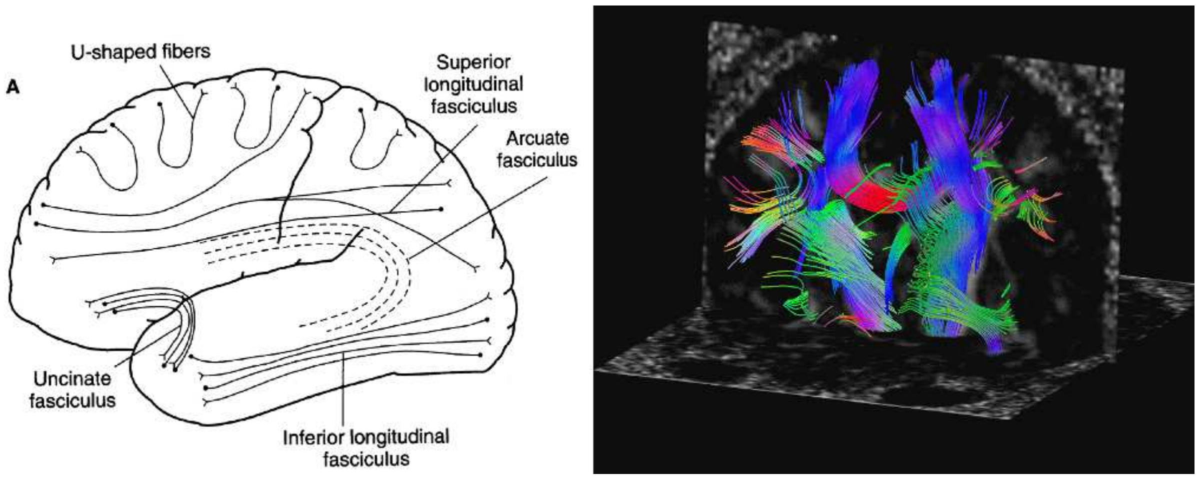

A
U-shaped fibers
Superior longitudinal fasciculus
Arcuate fasciculus
Uncinate fasciculus
Inferior longitudinal fasciculus

Abb. 8.8: A schema of neural fibers in the brain, compared to results of the algorithm. [zhukov2002].

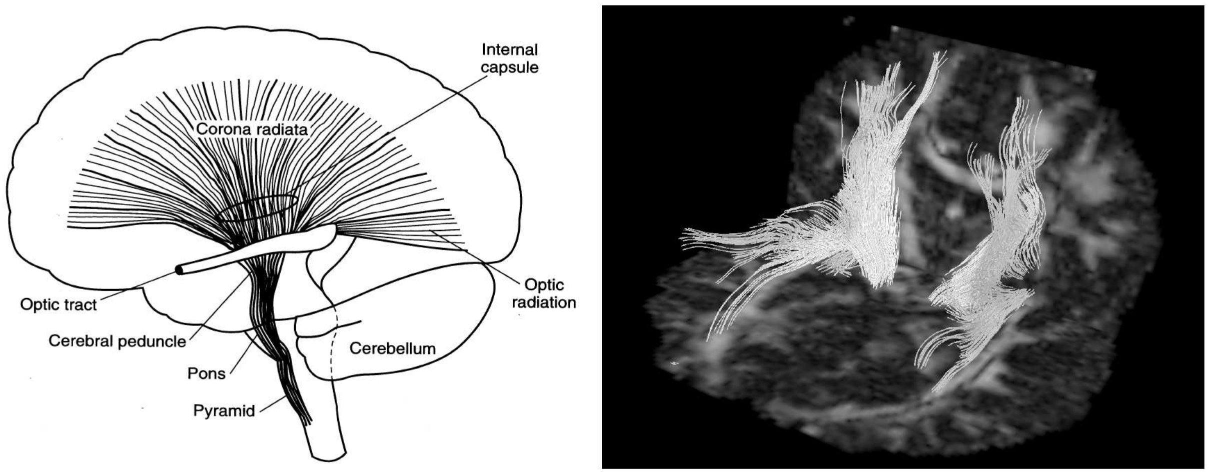


Abb. 8.9: Another schema of neural fibers in the brain, compared to results of the algorithm. [zhukov2002].

the spinal chord (figure 8.11). As shown in the picture, newer methods with different filtering approaches produce better results in these scenarios.

The tracing of the fibers is done in a straightforward way and produces good results in most cases when scanning inside of the brain. But at least in some synthetically generated tensor fields, the greedy approach of always following the direction of the tensor field and the moving least squares approach for filtering can lead to bad results. Figure 8.12 shows such a scenario. Non-greedy approaches like presented in [zalesky2008] with a global optimization can deliver better results in situations like these in exchange for lower performance.

6 Conclusion

In section 1.3 we only gave a very short overview of the variety of research that is done in the area of fiber extraction from T-MRI scans. It is safe to say that as of today, many algorithms exist that produce better results than the one presented here.

So, in conclusion, we can say that this algorithm works well enough for many applications, but has been superseded by newer methods. Today, its relevance is mostly based on the fact that this is one of the most basic approaches, on which the more sophisticated methods improve.

The current amount of research gives us reason to believe that improved methods and approaches suited for wider applications will continue to be developed.

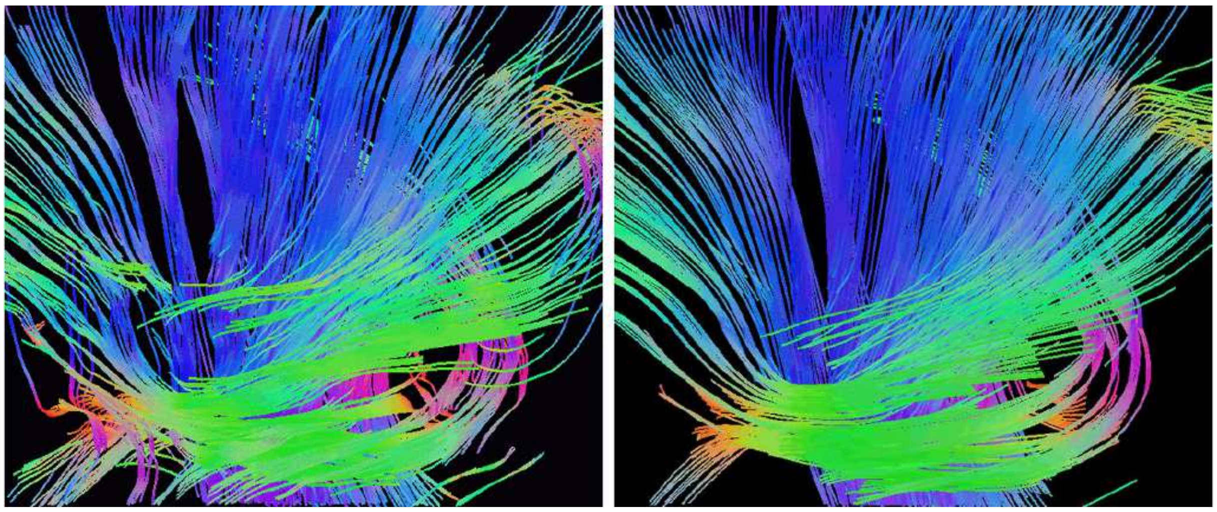


Abb. 8.10: Results of the algorithm described in section 3 without any filtering (left) and with the MLS filter applied (right). [zhukov2002].

Literaturverzeichnis

[zhukov2002] ZHUKOV, Leonid; BARR, Alan H. Oriented tensor reconstruction: Tracing neural pathways from diffusion tensor MRI. In: Visualization, 2002. VIS 2002. IEEE. IEEE, 2002. S. 387-394.

[basser2000] BASSER, Peter J., et al. In vivo fiber tractography using DT-MRI data. Magnetic resonance in medicine, 2000, 44. Jg., Nr. 4, S. 625-632.

[brun2003] BRUN, Anders, et al. Coloring of DT-MRI fiber traces using Laplacian eigenmaps. In: Computer Aided Systems Theory-EUROCAST 2003. Springer Berlin Heidelberg, 2003. S. 518-529.

[fillard2007] FILLARD, Pierre, et al. Clinical DT-MRI estimation, smoothing, and fiber tracking with log-Euclidean metrics. Medical Imaging, IEEE Transactions on, 2007, 26. Jg., Nr. 11, S. 1472-1482.

[zalesky2008] ZALESKY, Andrew. DT-MRI fiber tracking: a shortest paths approach. Medical Imaging, IEEE Transactions on, 2008, 27. Jg., Nr. 10, S. 1458-1471.

[alexander2005] ALEXANDER, Daniel C. Multiple Fiber Reconstruction Algorithms for Diffusion MRI. Annals of the New York Academy of Sciences, 2005, 1064. Jg., Nr. 1, S. 113-133.

[chen2005] CHEN, Bin; HSU, Edward W. Noise removal in magnetic resonance diffusion tensor imaging. Magnetic Resonance in Medicine, 2005, 54. Jg., Nr. 2, S. 393-401.

[lazar2005] LAZAR, Mariana, et al. White matter tractography using diffusion tensor deflection. Human brain mapping, 2003, 18. Jg., Nr. 4, S. 306-321.

[wenger2004] WENGER, Andreas, et al. Interactive volume rendering of thin thread structures within multivalued scientific data sets. Visualization and Computer Graphics, IEEE Transactions on, 2004, 10. Jg., Nr. 6, S. 664-672.

[mori1999] MORI, Susumu, et al. Three dimensional tracking of axonal projections in the brain by magnetic resonance imaging. Annals of neurology, 1999, 45. Jg., Nr. 2, S. 265-269.

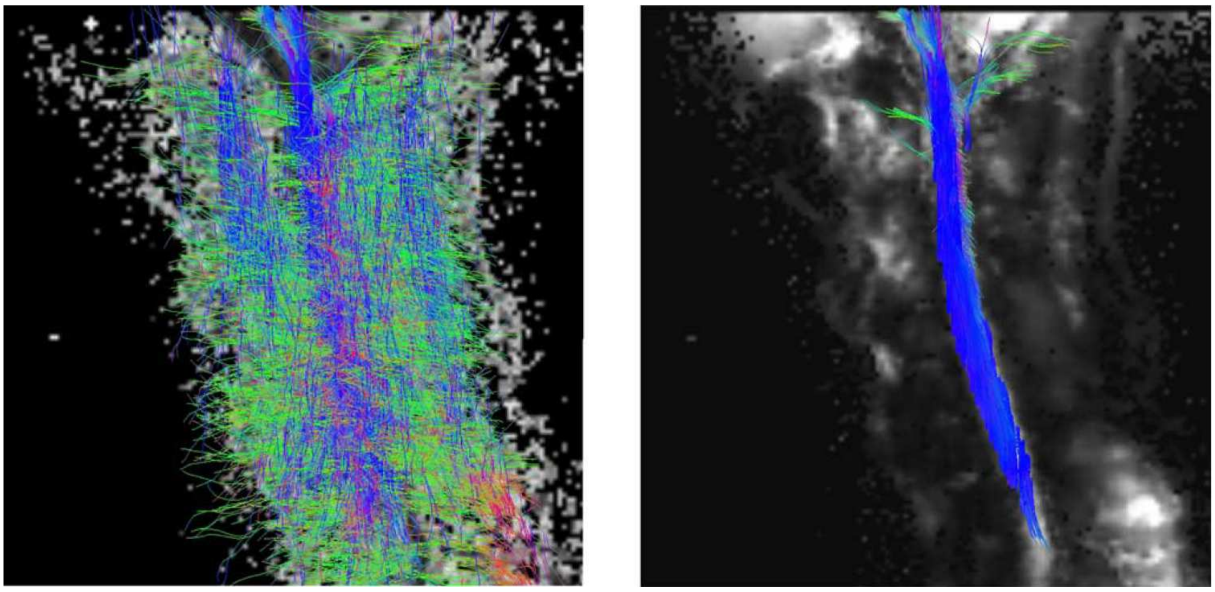


Abb. 8.11: Tracing results on a spinal cord with the algorithm presented here (left) and an improved approach from 2007 (right). The fibers traced in the left image do mostly not exist in real spinal chords. [fillard2007].

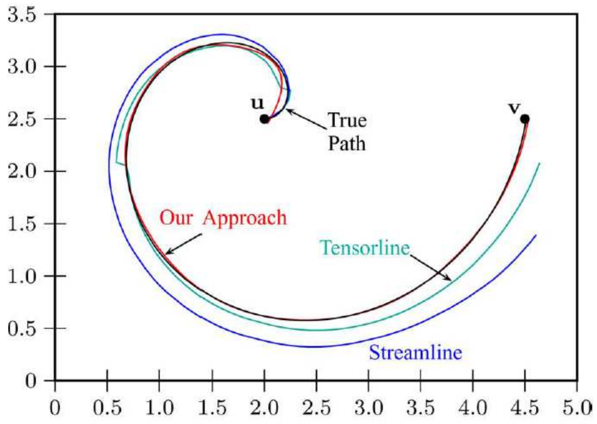


Abb. 8.12: This image shows a situation where the tensor tracing approach (*Streamline*) results in incorrect fiber reconstruction in a synthetic tensor field because of the polynomial fitting being adapted too slowly to the changing tensor direction. Here, the shortest path approach (*Our Approach*) leads to a better result. [zalesky2008]

Seminar: Deformed Lattice Detection in Real-World Images Using Mean-Shift Belief Propagation

Matthias Moeller

Zusammenfassung

Common problems in computer vision are detecting objects in images with a searched property. One property could be, that the given object is textured with a lattice and it is necessary, that the shape and the position of the object can be determined. For this purpose, an algorithm was provided and here discussed, which finds such an object and its position. Also, some basic computer vision techniques, which are used by the algorithm are presented.

Keywords: Lattice Detection, Computer Vision, Mean-shit Belief Propagation

1 Introduction

In computer vision, one is interested in identifying objects in a image. To do this, the shape and the position of an object must be extracted. This seminar is about identifying lattices in a given image, getting the position and the shape of a lattice. To do this, some terminology are introduced first.
What is a lattice? A perfect lattice is a 2D texture, which consists of one reference texture or tile. By repeating this reference tile in all dimensions, constructs a lattice. On a perfect lattice, all tiles are equal to the reference tile (see figure 9.1).
Extracting such a lattice is pretty easy, but what is a deformable lattice in real world images? In real-world

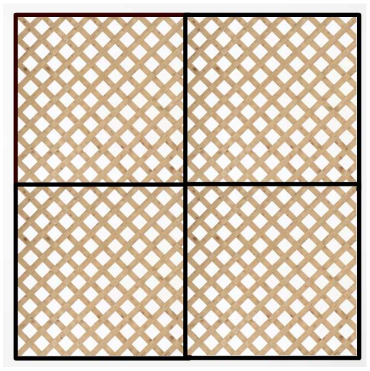
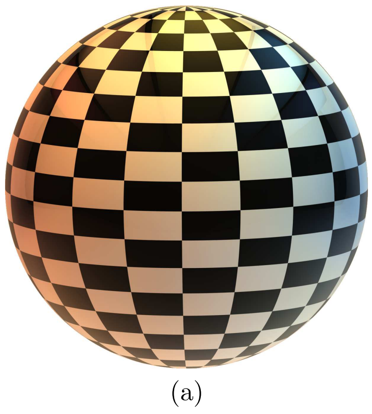


Abb. 9.1: Perfect lattice. Each tile surrounded by black borders is a copy of the reference tile (not in image) with different position. Lattice is the whole texture, not only the pattern of the reference tile. Source: http://www.lowes.com/

images, not all lattices are perfect, so the tiles are slightly different and it is not possible to determine a reference tile. Extending the definition of a reference tile to one tile, which minimize the error of all tiles. Also, not all latices are repeated with the same transformation (see figure 9.2) The position from one tile to the next can be described as vectors, taking only the left and top tile resulting in two vectors. It is enough to pick only the vector to the tile above/left of a tile, because you can iterate through it like a single linked list. These vectors are like local coordinate system, because they describe the transformation which must be done for his special tile to get the next tile.



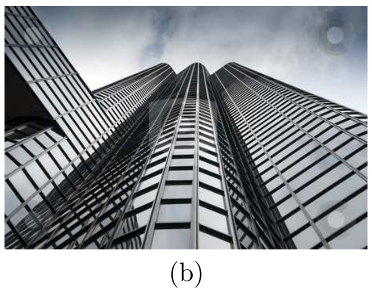


Abb. 9.2: Types of deformable lattices. (a) shows the 2D lattice projected on a non-planar surface. The 2D lattice gets a deformation. (b) shows a deformed lattice on a planar surface, but the surface is not perpendicular to the camera, resulting a deformation through camera perspective. Source: (a): deiby.deviantart.com/ (b): www.drawing1.cf

2 Basic Computer Vision Techniques

Before the lattice detection algorithm is presented, some basic computer vision techniques are explained, which are used by the lattice detection. In detail, those are the Mean-Shift and Mean-Shift Clustering algorithm and a short introduction in Markov Models, and the Beliefe Propagation algorithm

2.1 Mean-Shift

2.1.1 Mean-Shift Alogrithm The Mean-Shift Algorithm describes a way to find the maxima of a density function for a given dataset[5]. It is used for discontinuity preserving filtering and segmentation of colour and black/white images[3].

The algorithm is iterative and guarantees to converge at some point, given two parameters, a window h and a kernel function K. There exists several different kernel functions, all with their advantages and disadvantages.
Given those parameters and a dataset of n points $\mathbb{R}^d$, the multivariate kernel density estimate is defined as follows:

$$f(x) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{9.1}$$

To find a maximum of the function f, a update vector called mean-shit $m(x) - x$ is introduced[5].

$$m_h(x) = \frac{\sum_{i=0}^{n} K(x_i - x)x_i}{\sum_{i=0}^{n} K(x_i - x)} \tag{9.2}$$

The mean-shift vector $m(x)$ points toward the maximum increase of in density (See Figure9.3). Due, for finding the maximum density, compute on iteration t on the given point x_i^t, where $x_i^0 = x_i$ is, the update vector $m(x_i^t)$ which points toward the maximum increase of density and adds it to the vector x_i^t, resulting in $x_i^{t+1} = x_i^t + m(x_i^t)$. This is done, until the maximum is reached ($\nabla f(x) = 0$ or $m(x_i^t) = 0$). This method is a hill climbing method because the density increases all iterations until it reaches the highest point.

Note that the algorithm performs just on feature space. This means, if the input is the colour of all pixels of an image, only the colour values are updated and the position remains, but the window moves in colour space.

2.1.2 Mean-Shift Clustering The Mean-Shift Clustering algorithm is based on the Mean-Shift Algorithm but extends it for a small amount. Instead of finding the density maximum of a point in a dataset,

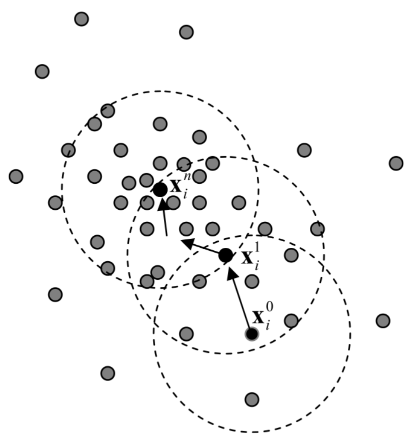


Abb. 9.3: Mean Shift Algorithm finding the maximum density of point x. Starting at point x_i^0 and updated with via mean-shift vector. Circled line describes the window h. Shaded points are points in the dataset.[4]

the clustering algorithm classifies all points. The classification is based on the maximum density function. Since the Mean-Shift algorithm always converges, the Mean-Shift Clustering algorithm computes for each point in the dataset the maximum density.
If two or more points in the dataset have the same maximum density ($x_i^t = x_j^u$), then they are in the same cluster. Obviously, no other parameters are necessary than the same parameters required for the Mean-Shift Algorithm, especially the number of clusters can be unknown.

2.2 Markov

Markov Chains, Trees and Random Fields are stochastic models expressing the probability of a event depending on other events which might be come true. The simplest form of a relations of probabilities which are connected for easier computing are Markov Chains. More sophisticated relations and the Hidden Markov Model are shown in the section Markov Random Fields and in the last section, the Believe Propagation Algorithm is introduced which allows to compute a model configuration with maximum probability.

2.2.1 Markov Chains Markov Chains are models containing random variables $X = (X_1, X_2, ...)$ which are connected via probability conditions $P = (X_i|X_{i-1}, X_{i-2}, ..., X_1)$. So, the simplest form of the Markov Chain can be constructed, if the random variable X_i just depends on the previous variable $P = (X_i < X_{i-1})$. This would be a first-order assumption.[1] An example to explain Markov Chains presented in [1] is the weather example:
The random variables $X = (X_1, X_2, ...)$ describing the weather of day i, $\{rain, sun\}$ the question is: What is the probability, that on day i the sun shines $X_i = sun$?
For the first order assumption, the stochastic model assumes, that the weather depends on the weather of the day before, so when day X_i is a rainy day, the probability that on day X_{i+1} for another rainy day differs than if day X_i was a sunny day. In the example it is assumed, that after a rainy day it is more likely to get a sunny day, which can lead to the following matrix (figure: 9.4 describing the probability $P(X_i|X_{i-1})$. Given that model, it can be simply computed with which probability the next three days are sunny days. To compute that, one can connect the days directionally, like a directional weighted graph, where each Vertex represents a random variable and is connected to its next random variable via an edge (See Figure 9.5). The example above just handle first oder Markov chains because of its simplity. For a non first order Markov chain, one can extend the model where the weather of day X_i not only depends on day X_{i-1} but also on e.g. day X_{i-2}.

2.2.2 Markov Random Fields Markov Random Fields are special Markov Chains where the random variables $X = (X_1, X_2, ...)$ can be imagined not as a big chain, but as a raster. Due, the random variable

		Yesterday (X_{i-1})	
		Rain	Sun
Today (X_i)	Rain	0.4	0.8
	Sun	0.6	0.2

Abb. 9.4: Example of first order Markov Chains with Weather probability model.[1]

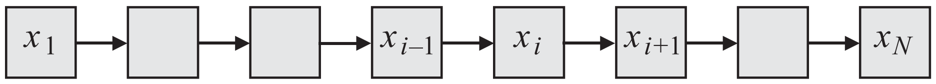


Abb. 9.5: Markov Chain represented as a directed graph.[1]

X_i does not only depend on the previous variables, but on the variable above, behind, upper and lower. For doing this, a so called `observation` z_i is introduced for each random variable X_i. For each X_i it is now possible to compute a likelihood depending on z. The function describing the likelihood $\Phi_i = (X_i, z_i)$ is often described as `evidence` function[11]. The observation can be for example a pixelcolour in a image and the X_i can stand for a position on a image. Evidence function is describes the likelihood of X_i with a position and the observation which is done on this position. If the position changes, also the observation changes.

Drawing the Markov Random Field structure would lead to a figure described in Figure 9.6. As described above, the random variables $X = (X_1, X_2, ...)$ are also connected via a probability function. This function is called `compatibility` function $\Psi_{ij}(X_i, Y_i)$. This function is the same function as the function described in the weather example in the Markov Chains section.
The overall joint probability of a Markov Field X with given observations $z_i \in Z$ can be computed as follows:

$$P(X, Z) = \frac{1}{n} \prod_{(ij)} \Psi(X_i, X_J) \prod_i \Phi(X_i, z_i) \tag{9.3}$$

Where (ij) denotes all neighbours j of i.

2.2.3 Believe Propagation Algorithm The Believe Propagation Algorithm computes the configuration of a Markov Model (e.g. Chains) including observations with the highest probability in a efficient way. The brute force way would lead to an algorithm computing the probability of each configuration and take the maximum. Let's assume, the Markov Model has k nodes and the observations can have n many states. For one single node X, the brute force model has the complexity of $O(n^k)$ (recompute all k-nodes probabilities for each state n) and for all nodes together $O(kn^k)$.

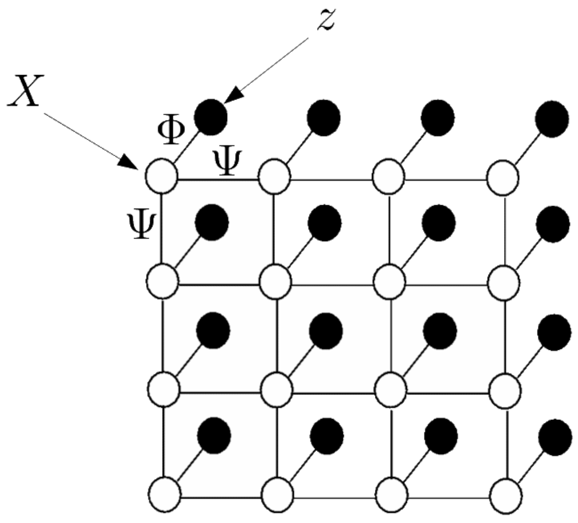


Abb. 9.6: Markov Random Field Structure. Black dots are the observations, which are connected to the white random variable dots via evidence function Φ. Random variables connected with the compatibility function Ψ.[11]

One faster solution is proposed with the Believe Propagation Algorithm (BP). [1] BP uses so called messages $m_{i \to j}$ from node i to node j. The message (in this case for MAP (Maximum A Posteriori) estimation) is defined as follows:

$$m_{i \to j}(X_j) = \max_{X_i} P(X_j, X_i) \prod_{k \in N\ j} m_{k \to i}(X_k) \tag{9.4}$$

Where $N\ j$ denotes all neighbours of i except of j.
Each node has a believe value $b(X_i)$ which can be computed with the Believe Propagation Algorithm and in a directed loop free Markov Model, the believe value will give the exact probability.[11]

$$b(X_i) = k\Phi(X_i, z_i) \prod_{j \in N} m_{j \to i}(X_i) \tag{9.5}$$

The believe is the product of the current evidence with all incoming messages. To get the MAP estimate, so the most likely solution for a given node X_i, can be received from the believe value, taking the maximum.

$$X_i = arg_x \max b_i(X_i) \tag{9.6}$$

3 Lattice Detection

3.1 Reference Tile Extraction

A lattice in a image contains one specific tile or texture-tile, which is repeated. In a perfect world, there exists only one texture-tile which is repeated without any changes, but the world isn't perfect all the time. So, a lattice contains several pieces of texture-tiles, which differs all slightly. One possibility of an lattice with different texture-tiles can be seen on a photo with objects, which aren't planar or not perpendicular to the camera. This would be a deformed object. In this case, the tiles can be matched, but they are transformed not only by a translation, but also by rotation, shearing and rotation in 3D. In a 2D photo, this information is lost and must be reconstructed.

One possible solution recognizing a lattice on a deformed object was presented by Wen-Chieh Lin[10]. The first step and one big part of the algorithm concentrates on extracting a possible reference texture-tile which fits the deformed texture-tiles is as equal to the rest of the tiles as possible. The algorithm can be used for general lattice without any pre-assumptions.

Given an image I, get some feature points via feature point detector. It doesn't matter which feature point detector is chosen, but a feature detector which recognize repeated features gives better results. Additionally, the image can be separated into blocks, hoping the regularities are unique in each block and therefore can be properly recognized by the feature detection.
After getting the feature points $f_0, ..., f_{n-1}$ on the image, they have to be classified. For classifying, look at the neighbours of the points (Wen-Chieh Lin[10] recommend $11x11$, should depend on the number on feature points and the picture size) and bring it this e.g. $11x11$ matrix in one row, so a 121-size vector. This n many vectors of colour values can now be classified by the Mean-Shift Clustering Algorithm, giving m many clusters $C = \{C_i | 0 \leq i < m, C_i \text{ unique cluster}\}$ where each feature points $f_0, ..., f_{n-1}$ belongs to one cluster ($f_i \in C_j$).

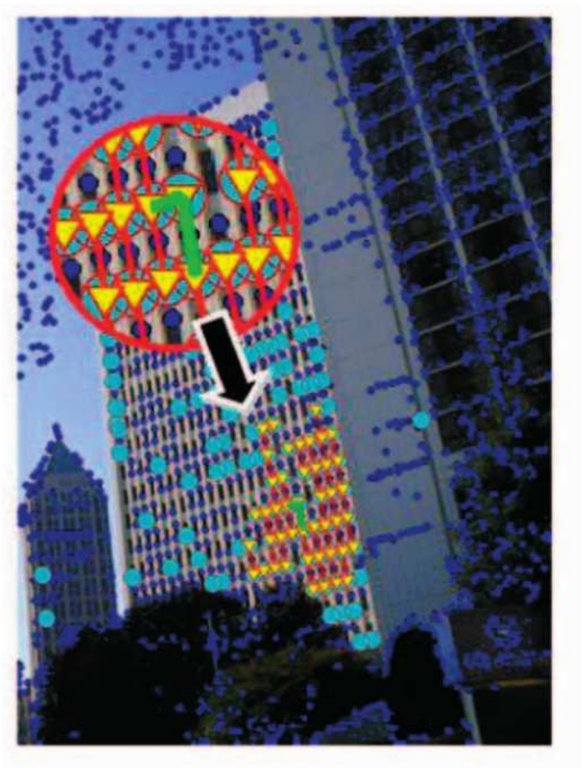


Abb. 9.7: Image after lattice extraction. Dark blue dots are extracted feature points. Bright blue dots the feature points of the current cluster. Green arrows the proposed (t_1, t_2) vectors and the red lines represent the vectors, which are also a orthonormal basis after applying the transformation.

A reference tile can be defined using 3 points. It makes sense choosing the 3 points for our reference lattice out of the feature points and since the lattice contains so many similar tiles, it is a good idea getting the feature points from the same cluster, since their neighbours are similar. But which one should be used, if in one cluster C_i contains more than 3 points and which cluster C_i should be selected?

A perfect lattice contains repeated tiles which are all equal but on different position. The relative neighbour position of one of these tiles is described by t_1, t_2 vectors. Taking the position of the tile and adding t_1 and t_2 to the position is the position of one neighbour. On a perfect lattice, these vectors are perpendicular and are equal for each tile. The most common transformation of a tile in a lattice is the shearing or rotation. No scaling or translation because the image has to be filled in the resulting holes interrupting the lattice of no translation/scaling was done. This gives the information, that the length of the vector are equal, true on the perfect lattice but also by a deformed lattice.

Using the observation explained above, one can construct an algorithm providing 3 points which have the most feature points in their cluster fulfilling the property of the same length and the same orientation/angle between them. This can be done using a brute force method. For each cluster C_i take

the vectors and connect randomly 3 feature points in the cluster $f_0, f_1, f_2 \in C_i$. This give the t_1, t_2 vectors. Do the same for each other feature points in the cluster given vectors t_{1_i}, t_{2_i}. With some linear algebra, you can convert the vector t_1, t_2 to an orthogonal coordinate system pointing to $(0,1), (1,0)$. Apply the same transformation to each t_{1_i}, t_{2_i}. If the tiles have the same property like in our observation, saying the vector from one tile to the next have all the same orientation/angle and the same length, the vectors t_{1_i}, t_{2_i} should also be transformed into a orthonormal coordinate system. Counting those vectors t_{1_i}, t_{2_i} which are, after applied transformation, nearly a orthonormal coordinate system (ϵ-ball around).
The algorithm described above can now be done for each cluster and for each vector combination. The proposed reference lattice is extracted from the 3 vectors with the highest count. Knowing the 3 vectors, we can extract the t_1, t_2 vectors describing the local coordinate system and the position of the reference lattice.

3.2 Lattice Extension

On the first step, a reference texture-tile was extracted inside the image, with a given position, dimension and local transformation. This local transformation is represented by the vectors (t_1, t_2). Since a lattice is a continuous pattern, its a good assumption searching the next tile near the reference which is translated. Normalized Cross Correlation (NCC)[2] is used, when the „position of a pattern is unknown“[2] given the pattern, an estimate position and the image. The NCC algorithm returns a value, which can be used to estimate the likelihood of similarity of the given pattern and the given shifted image patch. It is also a robust algorithm, if the pattern are not exactly matching, like signals or images[6] and it can be faster computed than Fast-Fourier Transformation (FFT)[2].

Finding the best position with the highest likelihood given by the NCC, a Markov Random Field can be given, where the random variables are the positions and the observations are given by the NCC value. Using an algorithm like the BP Algorithm leads to fast way finding the positions with the highest likelihood. Thus, the evidence function $\Phi(X_{(i,j)}, z_{(i,j)})$ of the Markov Field based on NCC an additionally for increasing the power, the edge magnitude[10] yielding to the following equation for the observation $z_{(i,j)}$ (observation on position (i,j)):

$$z_{i,j} = NCC(T_0, I_{(i,j)}) \times NCC(e_m(T_0), e_m(I_{(i,j)})) \tag{9.7}$$

Where T_0 is the reference tile and $I_{(i,j)}$ the image patch around position (i,j) and e_m is the edge magnitude. α is a freely constant which can be set to 5.[10] This value expresses just the amount of probability change, since the probability function is not linear but exponential. Note that (i,j) are coordinates, which are not in image space but describing the coordinates of the Markov Field Nodes. The evidence function Φ can be expressed as follows:

$$\Phi(X_{(i,j)}, z_{(i,j)}) = exp(-\alpha(1 - z_{i,j})) \tag{9.8}$$

The Markov Field also needs a compatibility function $\Psi(X_{(i,j)}, X_{(i\pm1,j\pm1)})$, which describes the connection between to neighbouring nodes. Since each tile is spanned by a (t_1, t_2) vector, defining the local transformation inside the tile and one can assume, that a lattice is not deformed too much, it is obvious that the compatibility function should minimize the differences between the (t_1, t_2) vectors per tile. Note, the vectors (t_1, t_2) of a node X_i which is not yet validated by the algorithm and might has not the right position since the algorithm is searching for it, are just proposed vectors and not yet vectors of this tile and must be computed whenever the position of the node is changed. The equality of two pairs of vectors can be expressed as follows[10]:

$$E(t_1^i, t_2^i, t_1^j, t_2^j) = \max\left(\frac{||t_1^i - t_1^j||_2}{||t_1^i||_2}, \frac{||t_2^i - t_2^j||_2}{||t_2^i||_2}\right) \tag{9.9}$$

The compatibility function $\Psi(X_{(i,j)}, X_{(i\pm1,j\pm1)})$ can be defined pairwise as follows:

$$\Psi(X_{(i,j)}, X_{(i,j\pm1)}) = exp(-\beta \times h(X_{(i,j)}, X_{(i,j\pm1)}), \tag{9.10}$$

$$h(X_{(i,j)}, X_{(i,j\pm1)}) = E(\overrightarrow{x^{it}_{[i,j]}x^{it}_{[i,\pm j]}}, \overrightarrow{x^{it}_{[i,j]}x^{0}_{[i+1,j]}}, \overrightarrow{x^{0}_{[i,j]}x^{0}_{[i,j\pm1]}}, \overrightarrow{x^{0}_{[i,j]}x^{0}_{[i+1,j]}}) \tag{9.11}$$

where x^0 are the local coordinate system vector of the reference lattice and $x^{(it)}$ the local coordinate system vectors of the node on iteration it. This function shows, that the divergence of the local coordinate system should be minimized and the probability of too strong deformed local coordinate system is low. Thus, the evidence function checks, if the colour and edge magnitude are similar and computes a probability function, the compatibility functions checks, if the deformation is not too high and compute for the deformation a probability function. Combined together it means, that higher probabilities are similar to the reference tile in shape and in colour/edge magnitude.

With the Markov Field it is possible now, computing the most probable positions of the new tiles. One efficient way to do this, is the Belief Propagation Algorithm introduced in section 2.2.3. A modified form of the Belief Propagation Algorithm is the Mean Shift Belief Propagation Algorithm (see chapter 3.2.1 which is based on the iterative updating, hill climbing principle of the Mean-Shift algorithm reducing the complexity of the high range of the observations.

After getting the highest probabilities of each node, not every node is an element of the lattice. Let's say 0.1% is the highest probability of a node, the probability that this node is contained in the lattice is pretty low. If the node isn't removed before the next step, the tiles can be deformed, that tiles can get a low probability. Therefore, a verification step must be done before the lattice warping step is performed. This verification step can be a threshold or more dynamic solutions considering the probabilities of the neighbours. The dynamic one can be preferred, if the reference tile has a unknown and maybe bad probability.

3.2.1 Mean-Shift Believe Propagation Algorithm The Mean-Shift Believe Propagation Algorithm gives a fast approximation of the Believe Propagation Algorithm introduced in section 2.2.3 by reducing the complexity.[9]
Given X_i with a predicted state. Instead of searching the best solution, trying out all states for the observation z_i, only a range near to the observations is used. This reduces the complexity, specially with Markov Model using loops. The random variable X_i will be up updated iterative, like in the Mean-Shift Algorithm depending on the messages.

One message for communicating the believe is also updated each iteration. Note, that n stands for the n-th iteration, therefore for the iteration 0, no message from the neighbours are needed. The following defines the message with the MAP rule

$$m_{i\to j}^{(n+1)}(X_i) = \max_{X_i} P(X_j, X_i) \prod_{k\in N\setminus j} m_{k\to i}^{n}(X_k) \tag{9.12}$$

Getting all messages from the neighbours, the believe value can be computed.

$$b(x_i) = k\Phi(X_i, z_i) \prod_{k\in N} m_{j\to i}(X_i) \tag{9.13}$$

On the next step, the nodes can be updates according to the Mean-Shift rule influenced by the believe value.

$$X^{(n+1)} = \frac{\sum_{i=0}^{n} K(X_i - X^n) b(X_i) X_i}{\sum_{i=0}^{n} K(X_i - X^n) b(X_i)} \tag{9.14}$$

Where X^0 is the initial predicted position. Now, the next iteration can begin with computing the new messages. This is done until the algorithm converges.

3.3 Lattice Warping

The Lattice Extension proposed new positions of tiles which extends the lattice. Since the algorithm should perform on deformable objects, a warping step is performed reducing the error and increasing the possible probability of tiles in further iterations of the algorithm.
One example of warping is shown in Figure 9.9. In the shown example, images, which show a scene from different perspectives, are so warped, that the position of feature points (red dots) are equal to the position of the feature points in a reference image.
For the Markov Field nodes there do not exists any reference positions, but the local transformation. On the first step (chapter 3.1), the local transformation (t_1, t_2) are extracted from the reference tile. Warping

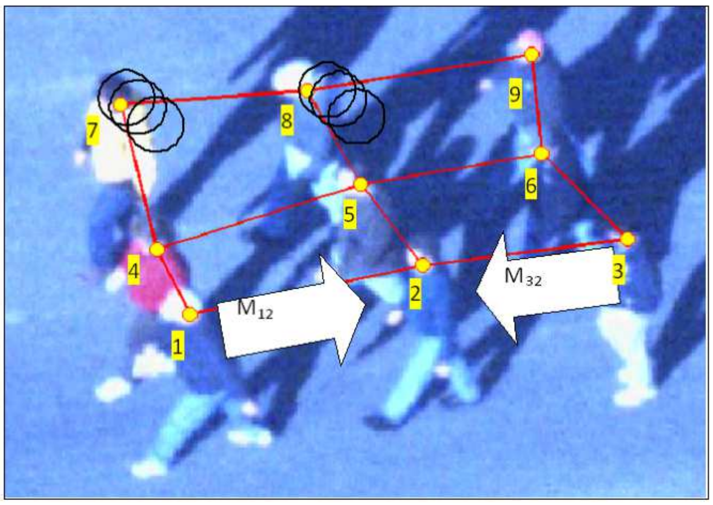


Abb. 9.8: Mean-shift example on Markov Fields. Here, the window are the black circles. After computing the local belief under the window, the belief propagation is performed. Based on that the window is updated and another belief propagation is performed. This actions loops until the algorithm converges.[9]

the image, so that all (t_1^i, t_2^i) of each node X_i in the Markov Field pointing into the same direction as (t_1, t_2), brings the lattice defined through the Markov Field into a wallpaper shape. Each tile inside the lattice share the same transformation on the warped image. Additionally, all objects on the image are warped improving the probability of further Markov nodes coming after the deformed tile.
The tiles can be seen as a graph with the reference tile as root. The new neighbours of a tile are the children of the node in the graph. Going from the reference tile to another tile, one have to pass all the neighbours and for each neighbour, all the local transformation must be applied getting the global transformation of the current tile. By equalizing all the local transformations, this steps aren't necessary and leads also to a simpler compatibility function in the Markov Model, which only depends on the reference coordinate system and not on the neighbours.

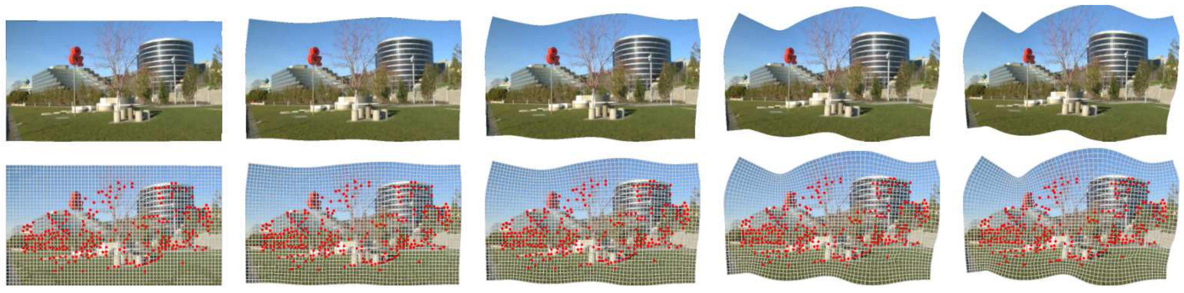


Abb. 9.9: Warping Example. Several slightly different images are wrapped (top), so the feature points (red dots) are on the same position on each image. The original images show the scene with different camera distances to the scene.[8]

4 Evaluation

4.1 Critism

The presented algorithm in the main paper[10] is based on the previous published paper of the same authors[9] and the paper by Wen-Chieh Lin et. al.[7]. One can say, one of the major improvements of the presented algorithm by Wen-Chieh is the use of the Mean-Shift Belief Propagation algorithm and it is also a very good improvement. The Mean-Shift Believe Propagation algorithm runtime is very good, if the window is small and the predicted states are in the area of the maximum density (in this case probability).

In this case, the number of iterations are small and the mean-shift converges fast. The mean-shift part of the algorithm converges slow, if the predicted state is far away of the maximum and therefore, the window needs to be updated often and each update includes a belief propagation. However, as the tile transformation doesn't include any translation and the vectors to the next tile should have nearly the same length, the predicted state of the next tile's position is somewhere nearly the predicted state and so, the number of iterations are low (See figure 9.10).

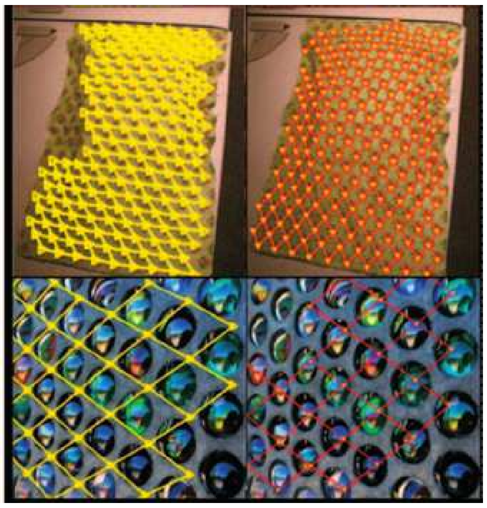
Average Run Time (min)	Data set 1 (67 images)	Data set 2 (73 images)	Data set 3 (121 images)	D (261 images)
Hays et al [12]	43.13 ± 33.1	35.45 ± 32.56	36.69 ± 17.53	38.00 ± 26.65
Ours	4.48 ± 2.47	4.91 ± 2.98	5.89 ± 4.30	5.09 ± 3.57

Abb. 9.10: Runtime of several Belief Propagation algorithms compared to the base paper. Base paper[7] first column. No machine/implementation information provided.[10]

Another great advantage of the algorithm is the independence of the first part, the reference tile extraction, and the iterative part, lattice extension, verification and warping. The presented reference tile extractor is a general solution, having nearly no assumptions except of searching for a lattice. Depending on the application, it can be improved by e.g. human help, where the lattice area is roughly selected but it will remove the full automatized functionality. If some more details about the lattice is given, they can also be programmed as constraints and therefore increasing the reference lattice heuristic (See probabilities for general approach on figure 9.11). The same constraints can also help for the Markov model functions. The presented equations for the evidence function works well in general case, but if the algorithm extracts only transparent lattices such as shown on figure 9.12 (b), the colour component can be erased. here again, it depends on the application and the assumptions which can be made.

Lattice Proposal Success Rate	Data set 1 ($D1$) (67 images)	Data set 2 ($D2$) (73 images)	Data set 3 ($D3$) (121 images)	All (261 images)
ECCV2008 [31]	70.1%	64.4%	66.1%	66.7%
Proposed	82.1%	80.1%	89.3%	84.7%

Abb. 9.11: Recognition rate of the reference tile on several image datasets containing lattices compared to the base paper. Base paper[7] first column.[10]


(a)

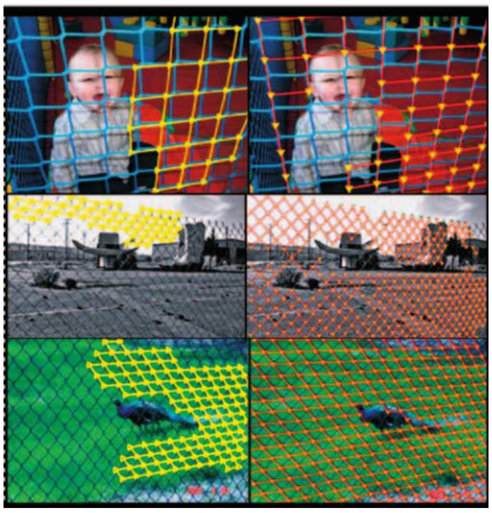
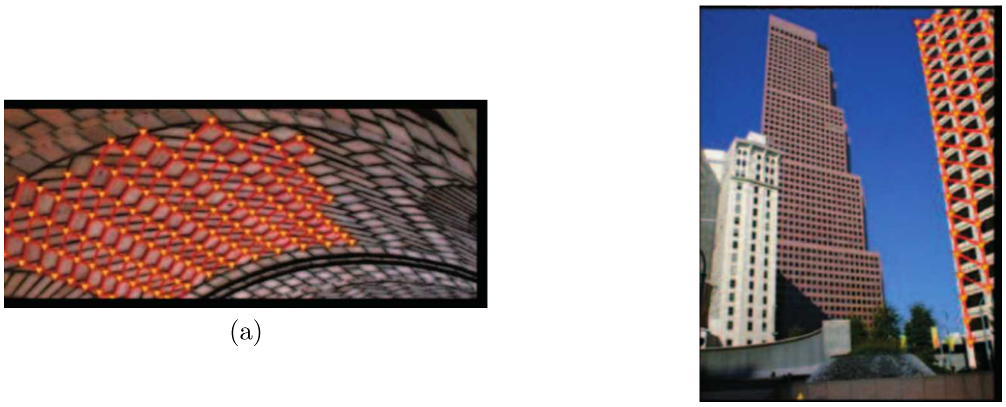

(b)

Abb. 9.12: Results of the lattice extraction. Red is the result of the presented algorithm compared to the derived algorithm[7].[10]

Take into account, that the lattice detection can also fail. First error can be the reference lattice extraction, which yields to a wrong lattice and not correctly recognized shape, the second can be a too tolerant and too strict Markov Model. These errors are hardly to detect, especially the algorithm doesn't detect them and gives a wrong result. Thus, a review by human can only detect false detected lattices. Figure 9.13 shows some examples, where the algorithm fails.



(a) (b)

Abb. 9.13: Failed results of the lattice detection. (a) shows a lattice which is deformed too much and the verification terminates the algorithm too early. (b) is ambiguous, because a lattice could also be detected on the centred building.[10]

4.2 Application

As the presented algorithm is full automatized, say it doesn't need any human input or additional parameters, except of the picture, one application could be one dealing with large image databases. For medicine purposes this has to be a medical image database, where the algorithm detects a lattice. This lattice can be for example a colour scheme template or a lattice for measurements (similar to ruler, but better detectable). Given the measure device or the colour scheme palette, one can reconstruct information about the image like the distance of a recognized object or the colour code. This extracted information can be used as input for another algorithm, which needs the local distance dimension for a given image, e.g. tagging each picture by the wound size. Having such an algorithm, one can easily create a image database which automatically tags all the input image for wound size and maybe wound type. Given such a database, the doctor can make a photo and searching for similar wounds. If the database doesn't contain not only images, but also references to therapy and diagnostic information per image, the doctor can receive experience about that type of wound from other doctors.

5 Conclusion

As a conclusion, one can say, getting a deformed lattice from a real-world image is not trivial and may take some computation time. First, the reference tile must be extracted and a position of it. In a iterative way, a Markov random field is build, where each node position is a node. Via Mean-Shift Belief Propagation algorithm it is possible getting the most probable solution for the Markov Field giving the position of each node. After the verification, if the node could be a tile, all local coordinate system get uniformed to the reference lattice coordinate system. Then the Markov Field is build up again and this is done until no new tile could be found.

Literaturverzeichnis

[1] Pushmeet Kohli Andrew Blake and Carsten Rother. *Markov Random Fields for Vision and Image Processing.* MIT Press, 2011.

[2] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. In *Proceedings of SPIE: Optical Pattern Recognition XII*, volume 4387, pages 95–102, March 2001.

[3] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, May 2002.

[4] Konstantinos G. Derpanis. Mean shift clustering. August 2005.

[5] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Inf. Theor.*, 21(1):32–40, September 2006.

[6] J. P. Lewis. Fast normalized cross-correlation, 1995.

[7] Wen-Chieh Lin. *A Lattice-based MRF model for Dynamic Near-regular Texture Tracking and Manipulation.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 2005.

[8] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph.*, 28(3):44:1–44:9, July 2009.

[9] Minwoo Park, Yanxi Liu, and Robert Collins. Efficient mean shift belief propagation for vision tracking. In *Proceedings of CVPR 2008*, June 2008.

[10] Minwoo Park, Student Member, Kyle Brocklehurst, Student Member, Robert T. Collins, Senior Member, Yanxi Liu, and Senior Member. Deformed lattice detection in real-world images using mean-shift belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009.

[11] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Exploring artificial intelligence in the new millennium. chapter Understanding Belief Propagation and Its Generalizations, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

Detection and Segmentation of Cell Nuclei in Virtual Microscopy Images Report for Medical Image Processing Seminar Winter Semester 2014/15

Poornima Belavadi

Zusammenfassung

The exponential growth in the technology and the advancement in the image segmentation techniques have made the job of a pathologist a lot easier today. Invention of virtual microscopy technique accentuated the prospects of image segmentation in the field of medical science. However, choosing the appropriate segmentation technique suitable for a particular type of cell tissue has always remained a challenge, given the diversity in the morphological structures of the cells. To choose an appropriate segmentation technique, the already applied techniques should be first evaluated and compared. This paper tries to evaluate some segmentation techniques which were applied to segment cancer cell images and compare their results, with the sole motivation that a solid comparison would lead to choose an appropriate segmentation technique. In the process we discuss the potential problems for evaluation.

Keywords: Segmentation, cancer cells, edge based segmentation, virtual microscopy images

1 Introduction

The concept of image segmentation has been there for over 50 years now [1]. The first light microscope was invented in 1675. Though the two concepts existed since a very long time, not much work was done to try to club both these concepts and come up with an efficient way to segment images. The idea of segmenting the medical images was handicapped because of the lack of technology to capture the structures which could be seen under a microscope. As the technology grew, the idea was implemented. Hence, image segmentation found its application in the field of medical science in only recent times. Until then, the whole process of image segmentation was done manually by a pathologist, which was not only tedious but also time consuming.

Though image segmentation techniques were available since a very long time, it did not catch much attention in the field of medical science mainly because the images captured were of low quality and the segmentation techniques applied on those images did not give good results. Though the structures could be quite clearly observed under a microscope, there was still a need for sophisticated methods or devices to capture a microscope image [2]. This was later solved with the invention of sophisticated cameras and microscopes. One of the major factors influencing the interest of medical science in the field of image segmentation was the invention of virtual microscopy. Virtual microscopy techniques offered a way of capturing a microscopic image to its finest details and also store them, so that they could be retrieved in an efficient manner. Since image segmentation required high resolution images to give best results, virtual microscopy images solved that problem [2].

To summarize, the whole process can be seen as a chain reaction. Medical science needed a faster technique to segment huge number of cell images and segment accurately, this could be solved by applying Image segmentation techniques on cell images. To apply image segmentation techniques, it required high resolution images. With the invention of high resolution cameras and virtual microscopy the images could be retrieved. This has now led to image segmentation becoming an integral part of medical science.

Motivation: Most papers have adopted edge based segmentation techniques to segment cell tissues or cell nuclei images, reason being, the morphological diversity of the cells. Segmentation of cancer cell images comes up with an additional challenge of not being able to know the shape or the structure of the cell in prior. On such situations edge based segmentation techniques work best, provided the images have high contrast. We can compare the method adopted by the edge based segmentation to the way a blind man would identify an object: by touching and feeling the edges of the object.

The literature survey resulted in a large number of papers on this area, every paper proposing a new segmentation technique to segment cancer cells. Though a lot of work is done in finding an appropriate and robust segmentation technique, the amount of work done to analyse the results of all the techniques and compare the methodologies applied is surprisingly very little. However, we believe that evaluating and comparing the results of the previously presented methods is as important as finding a new method.

In this paper we discuss three such segmentation techniques adapted by the three papers which we have reviewed and try to evaluate their results, with the motive of choosing the best among them. The first two techniques is based on segmenting cell images to find the cancer cells and the final one is to observe the effect of drugs on cancer cells. By this we also try to give an overall view about how image segmentation techniques are applied in cancer research: from segmenting and separating each cancer tissue or cells to observing the effects of drug on the cells, along with keeping the motive of comparing the results alive.

2 Methods

In this section we describe in detail the three segmentation methods whose results we try to compare in the end.

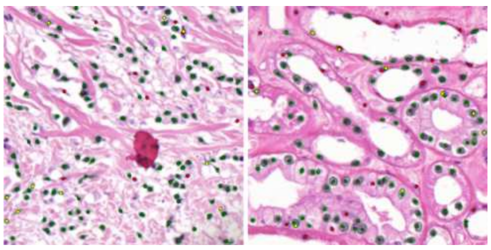
2.1 Background Information

Preparing the specimens for study: The tissue or cell samples are removed from any part of the body for the purpose of studying them. The collection process is usually based on the type of the suspected cancer. Some of the methods like Pap test, biopsy, Fine Needle Aspiration(FNA) are used to collect the samples [3]. After the collection, several laboratory techniques are applied on the samples in order to make them viewable under a microscope. Techniques like cytochemistry and histochemistry are applied, where special stains are applied on the samples to view the structures of cells and tissues more clearly and the selection of these stains is done on the basis of the chemical properties of these cell samples. Thus, a glass slide of the specimen is made by following the above described technique and these slides are stored for future use [3].

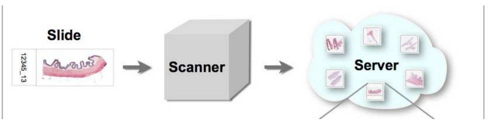
Traditionally pathologists viewed these slides under a microscope and manually labelled the cells to segment them which was tedious and time consuming. This has been now replaced by faster image segmentation techniques, which take high resolution digital images as input. Virtual microscopy is one such method for acquiring the high resolution images of the specimens. In virtual microscopy, the slides described previously are scanned with the help of some high resolution scanners resulting in a high resolution image of the specimen in the slide(refer to Fig 1). Such images are stored in huge databases, with this accessing the images has become a lot easier and the images are available to a broader audience at the same time [4]. Finally, image segmentation techniques are applied on such high resolution images. Fig 1(a). shows such virtual microscopy image of cells and (b) shows how the process is carried out.

2.2 Image Segmentation

Image segmentation is a process of dividing an image into segments with the purpose of understanding the image elements or identifying the region of interest. The process of image segmentation can be compared to the way humans recognise objects in a scene, we concentrate only on the important objects in the



(a)



(b)

Abb. 10.1: (a) Virtual microscopy image of tissues [2]. (b) Steps of virtual microscopy [4].

scene leaving out the other objects in the scene which does not concern us. Technically we could call it as our region of interest. Medical science uses image segmentation to extract the region of interest. Apart from medical science, image segmentation finds its application in various other fields. Some of the applications of image segmentation would be computer vision, machine learning, content based image retrieval, recognition tasks, object detection and many more.

Image segmentation methods can be mainly classified into five major groups,each having its own advantages and disadvantages [4]:

- Segmentation based on thresholding: Segmentation carried out based on the specified threshold.
- Edge based segmentation: Segmentation process, where the edges of the objects are identified first and then the region within the edges are found.
- Region based segmentation: Opposite to the previous method, first the region is found and then the edges are found.
- Theory based: Mathematical functions are applied to segment images, more specific to the kind of the image being segmented.
- Model based segmentation: The model information (shape, color etc.,) are known in prior to the segmentation, it helps to segment the objects precisely and in a faster way.

Each of the above mentioned techniques have further many methods under them, for example, global and local thresholding under the threshold based image segmentation, Region growing and merging under Region based and so on. All of these segmentation techniques have their own advantages and disadvantages, because of which each method have their own set of target images with which they work really well and fail on other images.

2.3 Segmentation by Contour Detection

The first method which we describe is the segmentation of cell nuclei by contour detection. The whole segmentation process is divided into six major steps [2].

Detection of closed contours: In the first step, the blind man approach is adapted and all the contours in the image are marked. To identify the contours two important things should be identified first: start pixels, object pixels.

A digital image can be defined as a combination of pixels on a square tessellation as show in Fig 2(a). Considering a binary image, a pixel is said to be an edge pixel if it shares either any of its edges(pink) or vertices with another pixel of different type(white)(say in a binary image, 1 should share its edge or vertex with a 0). After detecting the edge pixels the boundary is detected by starting from an arbitrary edge pixel and moving across the image connecting the edge pixels which share a common edge [6].

For a grey scale image, this is done as follows: each image row is considered as a one dimensional image function and the distance between each pixel and its local maxima or minima is calculated Refer Fig 2. The pixel is chosen as a start pixel if it has a maximum distance between its correlated local maxima or minima. All pixel values whose intensity lies between the intensity of the start pixel and the corresponding local maxima or minima are considered as object pixels. This is done by scanning the whole image row wise and storing the local minima, maxima along with the corresponding start pixels which form the maximum local gradients. Finally, the contours are detected by starting at a start pixel and continuing towards the corresponding edge pixels in its 8- connected neighbourhood along the object contours clockwise. A contour is considered valid only if it reaches back to the start pixel. Fig 2(b) shows the one dimensional image function, with the corresponding local maxima and minima and the maximum local gradient.

Contour evaluation: The first step in the segmentation process resulted in:

- Multiple and often overlapping contours not representing the object at all.
- Multiple contours representing same object.

Implies that the previous step resulted in over segmented images. To separate the overlapping images, it is important to choose the prominent objects in the image. The authors have tried to differentiate the segments representing the actual objects from other segments by applying the basic idea that, in any image the most prominently silhouetted structures define the object. Likewise, an object is considered important if it has a higher mean gradient along the object contour. To choose the contour that best represents the object among the multiple contours representing same objects gradient fit of the image is calculated using Sobel operator(S). If the similarity between the contour pixels and the maximum local gradient is high, then those contours are regarded as fitting best. Hence, gradient fit is defined as the relative number of contour pixels that are similar to the maximum local gradient in its 3 × 3 neighbourhood.

$$MeanGradient_i = \frac{\Sigma_j |S(P_{ij})|}{C_i} \tag{10.1}$$

$$GradientFit_i = \frac{\Sigma_j (P_{ij}^{max})}{C_i} \tag{10.2}$$

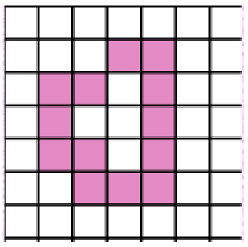
Generation of non-overlapping segmentation: To generate a non-overlapping segmentation of the images a 2D-map of the image of the size almost as that of the image is created. Each contour is labelled with a unique identifier and labelling of each contour is done in a sorted manner using the contour value, which is defined in equation 3. Labelling starts with the pixel which has the highest contour value and is not repeated for the pixels which are already labelled.

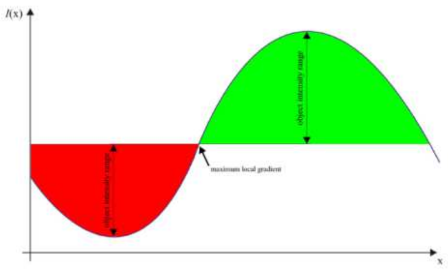
$$ContourValue_i = MeanGradient_i * GradientFit_i \tag{10.3}$$

Contour Optimization: Though the previous step successfully generates non-overlapping segments, the results still had a thin filament surrounding each segment which does not belong to the actual object Fig 3(a). To remove these filaments the contour optimization is done by calculating the compactness of each object pixel. Manhattan metrics are used to compute the distance transformation based on the chamfering method. A distance value d is defined to check the compactness of the object pixels. A pixel P_i with a distance value d_i and $d_i < d$ is considered as a compact pixel if it is connected to another pixel $P_j$with a distance d_j over $d - d_j$edges and $d_j = d$. An efficient algorithm is used to remove the non-compact pixels by parsing the image row wise.

Concave object separation: The previous steps result in non-overlapping segments without any filaments around, but as a cluster i.e., the cells would be still attached to each other. Concave object separation process handles these clusters by separating the objects to their concave borders by removing the object pixels along the cutting line between two concavities. The criteria to choose the cutting line is that the length of the cutting line should be minimal compared to the depth of the corresponding concavities which are located opposite to each other. This process is done by detecting the concave contour segments of each object by computing the convex hull. In Fig 2(a), if a pixel C exists between two other pixels A and B in the convex hull , with a distance from AB > 0. All the contour pixels between AB are considered as the concave points and they form the concave contour segment. All concave points of all concave contour segments are combined of an object are combined to find the best fitting pair for separation.Fig 3 gives a clear picture of the above mentioned method.

Classification into cell nuclei and other objects: To determine if the segmented objects actually represent the cell nuclei the nuclear specific color information acquired from the histochemical staining is used. The main purpose of staining is to extract the structure information of the elements which we observe under a microscope. Every stain will be characterised by a specific color, and this color is used as an aid


(a)



(b)

Abb. 10.2: (a) Square tessellation of pixels[6]. (b) One dimensional image function[2].

to extract the structure information. So staining becomes a very important step in the process of making the slides. Fig 4(a) shows an image of stained cells. The stain used for the samples is a nuclear specific stain named hematoxylin, characterized by a deep blue or lilac color along with a cytoplasm specific stain named eosin. The staining intensity of each pixel is computed and an optimal threshold is derived from this distribution. All the objects with the mean color intensity below this threshold are removed.

2.4 Segmentation by Graph cuts and Laplacian of Gaussian filter

The second segmentation approach explained here intends to improve the automatic segmentation process by the combination of two ideas: graph cuts based image binarization and Laplacian of Gaussian(LoG)filter [7].

Graph cuts: In graph cuts algorithm an image is divided in to foreground and background, or object and background. This is done by applying minimum cut concept in graph cuts on the image. We consider the image as a graph, the pixels forming the nodes of the graph. Two nodes - source node and a sink node are added to the graph, where source represents the object and sink represent background. Each pixel in the graph is then joined to the source and the sink nodes and the weight of the corresponding edges are calculated. The weights of the edges determine if the pixels belong to foreground or background. The pixels are also joined to their neighbouring pixels and their weights are also calculated. Finally, a minimum cut is applied on the graph which separates the foreground pixels from the background. A cut is said to be minimum if it separates the source and sink along the minimum path. Minimum cut also corresponds to maximum flow [8]. A clear picture of the graph cuts is shown in Fig 4(b).

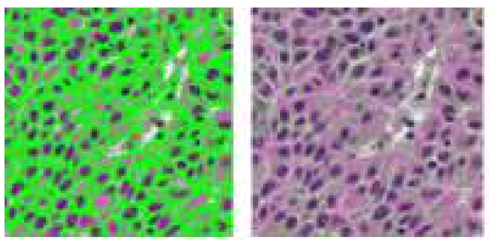
Image binarization by Graph cuts: The first step is to separate the foreground pixels in the nuclear channel from the background and this is done by Image binarization. To achieve this, the authors have chosen an advanced approach using Graph cuts and level set algorithm. Careful initialization or training must be done to adopt the above mentioned approach. Initial binarization is done by first computing the histogram of the given image samples and it was figured that the resulting histogram was a bimodal histogram and was modelled well by a mixture of two Poisson distributions. The authors have supported their modelling choice by the analysis of image processing results and prior literature and finally the Poisson-distribution-based modelling algorithm was used.The normalised image histogram for the mixture of Poisson distribution is given in equation(4), where P_0 and P_1 are the probabilities of the nodes belonging to background or foreground and $p(i|j), j = 0, 1$ are Poisson distribution with mean μ_j [7].

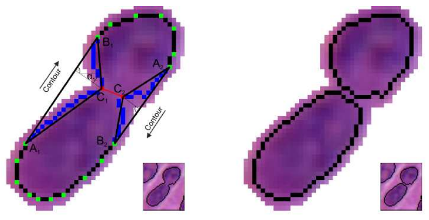
$$h(i) = P_0 \times p(i|0) + P_1 \times p(i|1) \tag{10.4}$$

Poisson mixture parameters are calculated for a threshold t, which is chosen to minimize the error criterion. Spatial continuity constraints are applied on the results of thresholding. The globally optimal labelling of pixels is done using the graph cuts segmentation algorithm.

Automatic seed detection and initial segmentation: The results of graph cuts binarization are a cluster of nuclei and they have to be separated. To achieve this identification of initial markers: one marker per cell is required. For this purpose multi-scale LoG filter is applied [7]. The LoG filter is given by:

$$LoG(x, y, \sigma) = \frac{\delta^2 G(x, y, \sigma)}{dx^2} + \frac{\delta^2 G(x, y, \sigma)}{dy^2} \tag{10.5}$$


(a)



(b)

Abb. 10.3: (a) Thin filaments around the cells, result of non overlapping segmentation, contour optimization results.(b)Concave object separation:Cluster composed of two cell nuclei. Contour pixels (black), vertices of convex hull(green), pixels of concave contour segment (blue), start pixels of the optimal cutting line (red [2]).

$G(x, y, \sigma)$ is Gaussian with 0 mean and σ scale. The LoG filter is a 2nd derivative filter and it is capable of detecting blobs in an image. The peaks in the distribution of the image, on which the filter is applied, represent the blobs in that image. The biggest advantage of this filter is that the location of the peaks is much robust because the chromatin texture scale value is much smaller compared to the nuclear blobs. The filtering results form a topographic surface that for basis for cell segmentation.

2.5 Segmentation by marker controlled watershed

The last segmentation technique discusses the application of image segmentation to study the effect of drugs on cancer cells [9]. To achieve this, the authors have adopted a method called time lapse microscopy. This method stitches the images taken of the cells under observation throughout the effect of drugs, into a video. Finally, an object tracking mechanism called as the kalman filter is applied on this video to track the objects in motion. The segmentation technique adopted here is the marker controlled watershed, as using the normal watershed algorithm would result in over segmentation of the cell images.

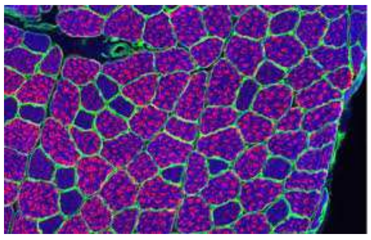
Background information: A watershed segmentation technique assumes an image as a topological surface. The minima and maxima forming the peaks and the valleys of the surface. To segment the image, it is assumed that the surface would be flooded with water starting from the minima of the surface. The flooding will continue until it reaches in a topological surface as shown in the Fig 6. As a result of this, in the end we would get a segmented image [10]. Fig 5(a) shows the topological surface and that being flooded by water, thereby showing the result of the segmentation

Marker controlled watershed: The authors define marker as a connected component of an image, which represents the existence of an object. The usual watershed algorithm considers the image as a topographic surface and starts flooding the image from its minima, which usually results in over segmented images. To avoid this, marker controlled watershed algorithm starts flooding from the pre-defined marker. The marker here can be regarded as the first segment in the image and therefore it is important to extract the object markers correctly [9].

Marker extraction: The proposed method is based on the concept of condition erosion, where the erosion is carried out only when the object is larger than the predefined threshold. The authors here have designed two masks according to the shape of the cells, one for fine erosion and other for coarse erosion purpose. The fine erosion structures comprise of 2 3*3 masks and coarse erosion consists of 4 types of $7 * 7$ masks as shown in the Fig 6 [9].

2.6 Evaluation

The main goal of a cancer diagnosis system is to automatically decide the existence of cancer in a cell or not. To achieve this most of the cancer diagnosis system follow three main steps: pre processing, feature extraction and diagnosis. In the preprocessing step the background noise is removed and segmentation techniques are applied to find the region of interest.This is the most crucial step among all the three steps, as any wrongly segmented region would lead in false diagnosis. From the segmented parts the required features are extracted for further study. This focuses on extracting the features of an individual cell.Morphological, fractal, textural and intensity based features are extracted from the segmented parts.


(a)

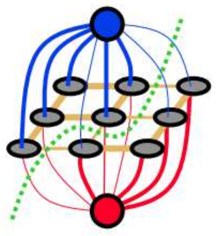

(b)

Abb. 10.4: (a) Examples of staining, the colour in the picture is because of the stain that's used. (b)Graph cuts, the blue and the red nodes represent source and the sink and the thick lines show the strong connection and thin lines weak the green dashed line shows the min cut [8].

Finally the main goal of diagnosis is to find out the benignity and malignancy of the cells or to find out the level of malignancy in each cell.

The idea of an automated cancer diagnosis system is to extract the information from the histopathological images and examine the extracted information by statistical analysis or applying some machine learning algorithms, and the second part comprise of the process of evaluating the system. We have discussed some examples of the first part (extract information from the histopathological images) in the previous sections. Before discussing the results it is important to know that evaluation is done on the whole system and not just the segmentation part. Some papers like the third method discussed above also give the segmentation results, but most other systems evaluate themselves on how precisely would it help in the cancer diagnosis. So the ultimate criteria for evaluating a system is to find out how precisely the system can distinguish between the cancer and the normal cells [11]. Fig 6 shows the picture of cancer cell and a normal cell, surprisingly the first image is a normal cell and the second a cancer cell.

Therefore for any given sample a system may lead to one of the four possible catogories:

- True Positive (TP): where the system would declare that the sample has cancer and the sample actually has cancer.
- True Negative (TN): where the system would declare that the sample does not have cancer and the sample actually does not have cancer.
- False Positive (FP): where the system would declare that the sample has cancer and the sample does not
- False Negative (FN): where the system would declare that the sample does not have cancer when the sample actually has.

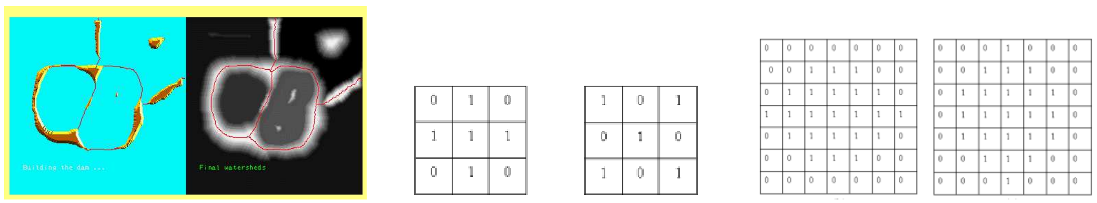
Out of all the above cases the most dangerous one is the false negative case [11]. To evaluate a system we have to count out of the total samples given to the system how many fall in each of the above mentioned categories. Then find out the probability of a system to give true positives and the probability to give true negatives. This is done by calculating sensitivity and specificity, there are also other formulas which help in determining the performance of the system.

$$Sensitivity = \frac{numberofTP}{numberofTP + numberofFN} \tag{10.6}$$

$$Specificity = \frac{numberofTN}{numberofTN + numberofFP} \tag{10.7}$$

$$Precision = \frac{numberofTP}{numberofTP + numberofFP} \tag{10.8}$$

$$Conglomerate = \frac{numberofdetectedcells - FP}{numberofTP} \tag{10.9}$$



0	1	0
1	1	1
0	1	0

1	0	1
0	1	0
1	0	1

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

0	0	0	1	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	1	0	0	0

(a) (b) (c)

Abb. 10.5: (a) Watershed segmentation technique [10](b) Erosion fine structures (c) Erosion coarse structures [9].

3 Results

Fig 7 shows the table of comparison of the results of the three segmentation methods explained in the previous sections.

The first method evaluates the system with the gold standard data set of 7931 cells which were morphologically diverse. The analysis yielded 0.908 precision, recall of 0.859 and conglomerate of 0.953. A comparison was made with the Al-Kofahis method first in the fully automatic mode, and second using an optimised parameter which was given by the author's of the first paper. It was found that Al-Kofahis method required less processing time but yielded 0.197 lower precision, however with the optimised approach the precision remained less. The conglomerate score of Al-Kofahis method was slightly method. The third method proposes the results in a different way, the method correctly segments 98.8

4 Discussion

After seeing the comparison of results one could argue, if this is the right way to compare the results or is it even possible to compare the results of different systems. There are some important aspects that we should consider here. First, different evaluation system estimate their system performance differently. From fig 8 it is clear that there is a difference in the way that the first two systems have evaluated the results from the rest of the systems. One of the main criterion ßpecificitymentioned in the results sections is not being considered by any of the systems. Whereas, it is well known that to measure a system's performance, we check the sensitivity and the specificity of that system. Second factor is that, all these systems work on different kinds of cancer, which means that the dataset that they consider also would be different. In such a case when the system is developed to diagnose completely different case of cancer, comparing the results would not make any sense. Finally, the basic dataset preparation is still done manually by pathologists, so this can vary to a very large extent from the way one person does it to another. Hence the basis on which each system gets its results could be only subjective.

The analysis on the first paper which describes the segmentation technique based on contour evaluation would be the following: The paper claims no segmentation bias", this would be agreeable if the bias they are referring to is about not having any prior knowledge about the morphological data of the cells. They also tell that the data samples they have chosen is of wide morphological variety. Here, we should really pay attention to how they have evaluated the system, because if the evaluation is done in the right way and if the data set that they have chosen is of wide morphological variety, then we could apply this technique on all kinds of cancer data to check if we could get similar kind of results. This could also be a possible future work for this paper.

Finally, the literature survey results in lot of papers working on the same problem and every work is coming up with a new method to solve the problem. It is very important now to sit and analyse all these work and do a complete survey about what is obsolete now and what can be improved and give a proper overview about all the techniques applied. Interestingly most of the papers apply edge detection techniques, no body explains why they choose this technique, this paper tries to give an explanation by comparing the blind man's approach, which to an extent sound like a feasible explanation for the problem. However there are possibilities for a different explanation for this. Hence, why most papers have applied this particular technique and are there any other techniques which could be better than this could also be a possible research question.

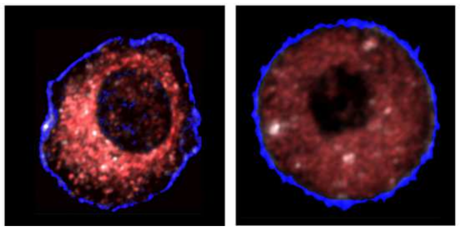
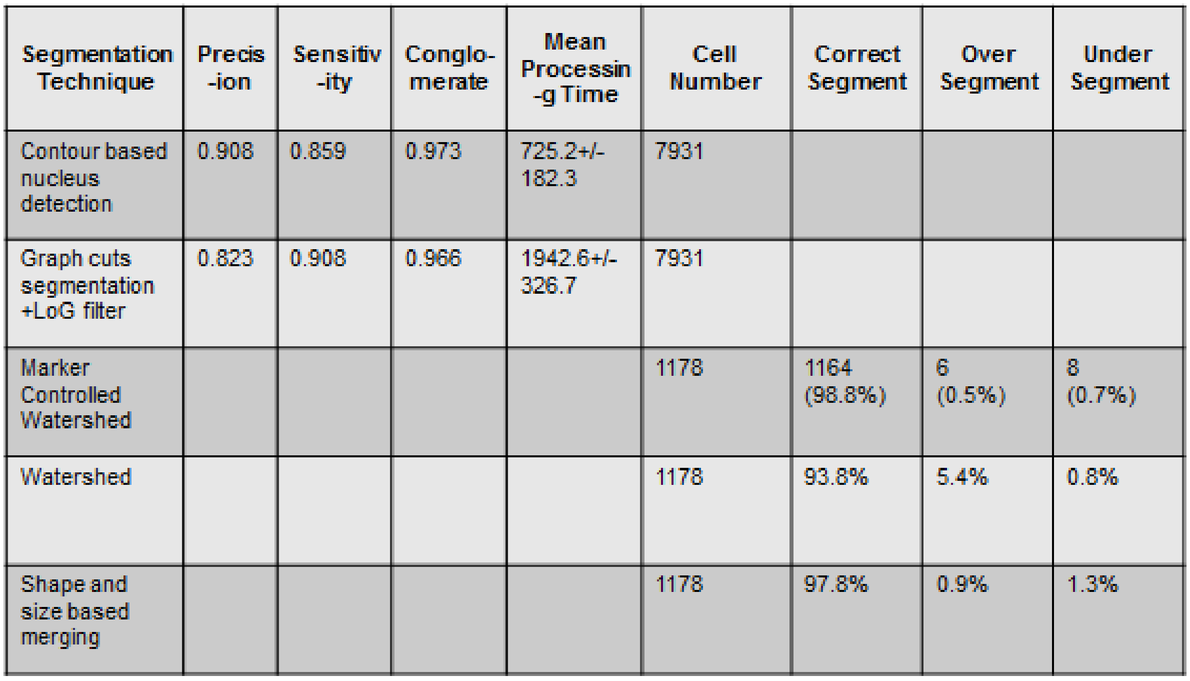

(a)

Abb. 10.6: (a) normal cell and cancer cell [11].



Segmentation Technique	Precis -ion	Sensitiv -ity	Conglo-merate	Mean Processin -g Time	Cell Number	Correct Segment	Over Segment	Under Segment
Contour based nucleus detection	0.908	0.859	0.973	725.2+/- 182.3	7931			
Graph cuts segmentation +LoG filter	0.823	0.908	0.966	1942.6+/- 326.7	7931			
Marker Controlled Watershed					1178	1164 (98.8%)	6 (0.5%)	8 (0.7%)
Watershed					1178	93.8%	5.4%	0.8%
Shape and size based merging					1178	97.8%	0.9%	1.3%

Abb. 10.7: Comparison of the results given by each system

5 Conclusion and Future Work

Analysis of previous work led to another important finding that, though there are so many papers published every year on the same problem and though every paper has come up with a new solution to the problem, the traditional cell segmentation techniques still remain the same, nobody has come up with completely new segmentation technique so far [1]. They have tried to modify the existing techniques by combining it with other techniques. But, the amount of work being done on this problem is continuously increasing every year. I try to reason this out, research is being done to solve the problem, but all of them are adopting the same approach to solve it and hence coming up with new method in the same approach. It is also important to try to find an alternative approach to solve the problem. This could be one future work for this problem. Coming up with one universal method to segment all kinds of cells, though is an ambitious but a valid future work for the problem.

Literature

[1] Meijering E. Cell Segmentation: 50 years down the road. IEEE Signal Processing Mag.2012;

[2] Wienert S, Heim D, Saeger K, Stenzinger A, Beil M, Hufnagl P et al. Detection and Segmentation of Cell Nuclei in Virtual Microscopy Images : A Minimum Model Approach. Sci.Rep., 2012;

[3] Canadian Cancer Soceity. Cell and tissue studies [Internet]. Available from: http://www.cancer.ca/en/cancer-information/diagnosis-and-treatment/tests-and-procedures/cell-and-tissue-studies/?region=on?how

[4] Unknown. Virtual Microscopy[Internet]. 2010. Available from: http://www.virtual-microscopy.net/

[5] Kandwal R, Kumar A, Bhargawa S. Review: Existing Image Segmentation Techniques.IJARCSSE.2014;4

[6] Ghuneim A G. Contour Tracing [Internet]. Available from:http://www.imageprocessingplace.com/downloadsV3/rootdownloads/tutorials/contourtracingAbeerGeorgeGhuneim/connectivity.html

[7] Al-Kofahi Y, Lassoued W, Lee W, Roysam B. Improved Automatic Detection and Segmentation of Cell Nuclei in Histopathology Images.IEEE.2010;

[8] Leibe B. Computer vision lecture slides RWTH Aachen[Internet].Available from:http://www.vision.rwth-aachen.de/teaching/lecturecomputervision/winter-14-15/lectures/cv14-part07-segmentation2.pdf.2014;

[9] Yang X, Li H, Zhou X. Nuclei Segmentation Using Marker-Controlled Watershed, Tracking Using Mean-Shift, and Kalman Filter in Time-Lapse Microscopy.IEEE.2006;53

[10] Beucher S. Image Segmentation and Mathematical Morphology[Internet]. Availabe from: http://cmm.ensmp.fr beucher/wtshed.html.2010;

[11] Demir C, Yener B. Automated cancer diagnosis based on histopathological images: a systematic survey.2009;