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

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

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

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

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

Hauptseminar im Wintersemester 2012/13

Medizinische Bildverarbeitung

Volume 8, Band 1
ISSN 1860-8906
ISBN 978-3-9813213-5-7

Aachener Schriften zur Medizinischen Informatik

Aachener Schriften zur Medizinischen Informatik
ISSN 1860-8906
ISBN 978-3-9813213-5-7
Herausgeber: Institut für Medizinische Informatik der RWTH Aachen
Pauwelsstr. 30
D-52074 Aachen

Geschäftsführender Direktor: Universitätsprofessor Dr. Dr. Klaus Kabino

Preface

As every year, students of the Medical Image Processinglecture had the possibility to apply and deepen their newfound knowledge of medical imaging in this seminar towards the analysis of research publications.

Again, the seminar language was English, making tribute to the increasing number of international students at the RWTH Aachen University. The seminar was held jointly by the Chairs VI and VII of the Department of Computer Science, Theoretical Informatics, and the Department of Medical Informatics, which is associated to the Medical School of RWTH Aachen University. By joining these diverse research areas, a broad base for interdisciplinary discussions and knowledge exchange was formed.

Students were handed research papers closely related to the seminar and lecture topics and had to investigate the validity and contribution of these works guided by their supervisors. The publications were composed of practical and theoretical works, applicable to medical image processing. By using academic methods like literature research, critical thinking and experimental evaluation of algorithms, the students were able to analyze the given topics in detail. After presenting their first take on the research topics in front of a group consisting of PhD students and professors, students were engaged in discussions to provide further insight into the field. The students' contributions and final analysis can be found in this proceeding.

Many thanks are owed to the participating professors Thomas Deserno, Leif Kobbelt, Hermann Ney and Peter Rossmanith, and the involved supervisors, without whom this interdisciplinary 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 2013

Stephan Jonas, Daniel Haak and Ibraheem Al-Dhamari

Inhalt

Hauptseminar Medizinische Bildverarbeitung

Vorträge der Blockveranstaltung im Wintersemester 12/13

theoretischer Teil/Prof. Rossmanith, 21.01.2013

13:00-13:45	Affine Image Matching Is Uniform TC0-Complete	Referent: *Michael Buse* Betreuer: *Felix Reidl*
13:45-14:00	Diskussion	
14:00-14:45	Reliable Path for Virtual Endoscopy: Ensuring Complete Examination of Human Organs	Referent: *Erik Noorman* Betreuer: *Fernando Sanchez Vilaamil*
14:45-15:00	Diskussion	
15:00-15:45	Brain tumor diagnosis systems based on artifical neural networks and segmentation using MRI	Referent: *Manfred Smieschek*
15:45-16:00	Diskussion	Betreuer: *Daniel Haak*
16:00-16:45	Graph-based Pigment Network Detection in Skin Images	Referent: *Ekaterina Sirazidinova*
16:45-17:00	Diskussion	Betreuer: *Stephan Jonas*

praktischer Teil/Prof. Ney, 31.01.2013

10:00-10:45	Perceptually Based Depth-Ordering Enhancement for Direct Volume Rendering	Referent: *Tobias Funke* Betreuer: *Dominik Sibbing*
10:45-11:00	Diskussion	
11:00-11:45	Features and Flow-Correspondace in Video	Referent: *Igor Dudschenko*
11:45-12:00	Diskussion	Betreuer: *Jens Forster*
12:00-12:45	Object segmentation using Active Shape Model and Good features	Referent: *Isaak Lim* Betreuer: *Yannick Gweth*
12:45-13:00	Diskussion	
14:00-14:45	Graph-Based Segmentation	Referent: *Phan Ahn Nyguyen*
14:45-15:00	Diskussion	Betreuer: *Christian Oberdörfer*
15:00-15:45	Image Matching for Object Categorization	Referent: *Hendrik Pesch*
16:45-16:00	Diskussion	Betreuer: *Harald Hanselmann*

Graph-based Pigment Network Detection in Skin Images

Ekaterina Sirazitdinova

Contents

Abstract

Since melanoma can only be treated in the first stages, it is very important to detect melanoma as early as possible. Such method as dermoscopy make it possible to scan for melanoma on a regular basis without biopsy. In particular, this technique allows examination of skin structures and patterns, and especially the identification of the most important feature of melanocytic lesions, known as pigment network. In this work I review a novel graph-based pigment network detection method that can find and visualize round structures belonging to the pigment network in dermoscopic images of skin lesions. The authors claim a very high efficiency of the method. In order to analyze the algorithm in detail I implemented the method following the steps described in the article. Arising problems led me to the conclusion, that the proposed algorithm in the paper is not suitable for clinical use as described by the authors.

Keywords: Dermoscopy, Skin lesion, Pigment network detection, Texture analysis, Graph, Cyclic subgraph

1 Introduction

Although malignant melanoma is less common than non-melanoma skin cancer, it is known to be one of the most lethal types of the skin cancers, causing more than 75 percent of all skin cancer deaths [1]. World Health Organisation estimates 132,000 new cases of melanoma per year internationally. In general, melanoma-related mortality rates continue to climb in line with increasing worldwide melanoma incidence [2]. Concerning melanoma mortality, Europe's rate of 1.5/100 000 is the third highest in the World, after Australia/New Zealand (3.5/100,000) and North America (1.7/100,000). Among the four geographical regions of Europe, the CEE countries have the largest share (35.5%) of the more than 20,000 melanoma deaths estimated to occur annually throughout the continent [3].

Melanomas can occur in different body organs (e.g., eyes or nails) but this work focuses on skin melanoma. Patients with this kind of melanoma usually present with skin lesions that have changed in size, color, contour, or configuration. Those changes serve as a first alarm for melanoma detection, and the acronym "'ABCDE"' is used to remember those physical characteristics suggestive of malignancy, which are, in other words, **A**symmetry, irregular **B**orders, **C**olor variations (especially red, white, and blue tones in a brown or black lesion), **D**iameter greater than 6 mm, and **E**volving over time [4]. Sometimes only first 4 characteristics are taken into consideration, and in that case the rule is called "'ABCD"' rule. It is very important to detect the "'ABCDE"' characteristics on early stage of melanoma.

The stage of a melanoma is an indicator of how deeply it has grown into the skin, and whether it has spread. Melanoma can be treated on early and medium stages by means of surgery: the skin cancer and some surrounding areas are removed [5]. The most difficult case (advanced stage) is metastasis, when melanoma spreads to other organs. When it happens, it usually cannot be cured and becomes fatal [6]. This explains the high necessity of melanoma recognition on early stages.

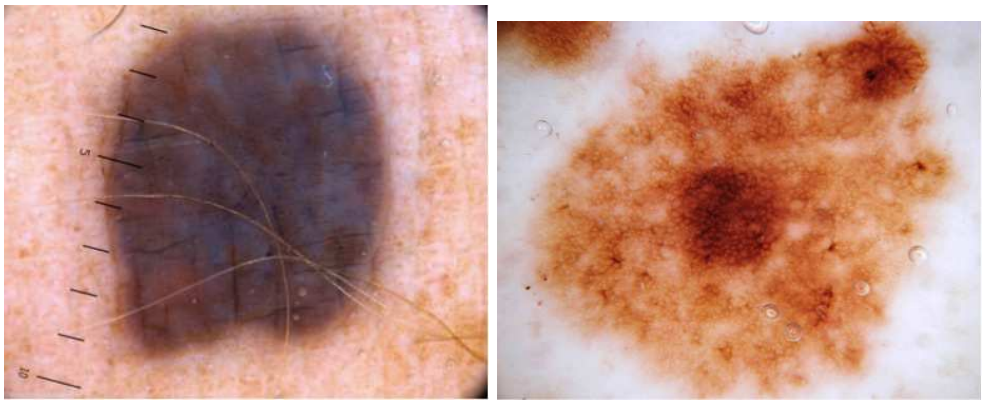
There are several ways to detect melanoma, for example, biopsy and imaging (radiology) tests. In the case of biopsy a sample of skin is taken and analyzed under a microscope. Experts look at certain features such as the tumor thickness and the portion of cells that are actively dividing. There are different ways to do a skin biopsy and all methods are likely to leave at least a small scar. Imaging (radiology) tests like chest x-ray, magnetic resonance imaging, positron emission tomography or computed tomography scans are done to create pictures of the inside of the body. They are used to look for the spread of melanoma and are not appropriate for people with very early melanoma, which is not likely to have spread. Blood tests are not used to find melanoma, but some tests may be done before or during treatment, especially for more advanced melanomas. Experts often test for blood levels of lactate dehydrogenase (LDH) before treatment. If the melanoma has spread to distant parts of the body, a higher than normal level of LDH is a sign that the cancer may be harder to treat [7]. Obviously, all those methods require work of experts and specific equipment.

In recent years a new approach for melanoma detection is being developed. It's based on detection of a pigment network in dermoscopic images of skin lesions. Dermoscopy (also known as dermatoscopy or epiluminescence microscopy) is a non-invasive method that allows the in vivo evaluation of colors and microstructures of the epidermis, the dermo-epidermal junction, and the papillary dermis not visible to the naked eye. This technique refers to the examination of the skin using skin surface microscopy. Dermoscopy requires a high quality magnifying lens and a powerful lighting system. This allows examination of skin structures and patterns. Several different lightweight, battery-powered hand-held devices are available at

the moment. Convenient attachments allow video or still photography. Computer software can be used to archive dermoscopy images and allow expert diagnosis. Computer aided diagnosis (CAD) programs may aid in diagnosis by comparing the new image with stored cases with typical features of benign and malignant pigmented skin lesions [8].

The most important feature of melanocytic lesions is the pigment network, which consists of pigmented network lines and hypopigmented holes. The network of hypopigmented holes corresponds to the suprapapillary plate, which is relatively thin and contains less melanin. The network lines correspond to the "'rete ridges"', which are thicker and have a greater quantity of melanin.

In normal melanocytic nevi, the pigment network is slightly pigmented. Light-brown network lines are thin and fade gradually at the periphery, holes are regular and narrow, the distribution is symmetric and sometimes accentuated in the center of the lesion. In cutaneous malignant melanoma, the pigment network usually ends abruptly at the periphery and has irregular holes, thickened and darkened network lines, and tree-like branching at the periphery. Moreover, the pigment network features change between different sectors of the pigmented skin lesion edge (border): some areas of malignant lesions manifest as a broad and prominent pigment network, while others have a discrete irregular pigment network; the pigment network may also be totally absent in some areas. Clinically atypical nevi (i.e., the nevi defined as dysplastic nevi at pathology evaluation) are often identified because they show areas of irregular and discrete pigment network, distributed asymmetrically (Figure 1.1) [9].



(a) (b)

Figure 1.1: (a) An image of a lesion without pigment network. (b) A lesion containing a pigment network.

There are exist different approaches for detection of pigment network in dermoscopic images. In this work I am going to observe and analyze the method proposed by M. Sadeghi, M. Razmara, M. Ester, T. K. Lee and M. S. Atkins [10]. In their approach they chain the following sequence of steps in order to find and visualize round structures belonging to the pigment network:

1. Manual image segmentation by an expert;
2. Highlighting the texture features;
3. Converting the image into a luminant;
4. Detecting sharp change of intensity;
5. Converting the resulting binary image to a graph;
6. Detecting all cyclic sub-graphs;
7. Removing noise or undesired cycles discriminating globules from meshes of the network;
8. Connecting the detected cyclic structures into a new graph;
9. Classifying the image as "'Absent"' or "'Present"' according to the density ratio of the new graph of the pigment network.

Being "'Present"' means that a pigment network is detected in the skin lesion and there is a high probability that the patient has melanoma.

The authors claim the accuracy of the method as 92.6% in classifying images to two classes of "'network is absent"' and "'network is present"'. Comparing to another approaches, it is a very high result.

2 Existing approaches observation

For this work, I have observed and compared several methods for melanoma detection by means of image processing, proposed by different authors. Some of the approaches are also based on pigment network detection, while others operate with different methods, like, for example, "'ABCD"' rule or decision trees. Almost all of the observed methods operate with epiluminescence microscopic (ELM) images, when one of the observed approaches work with images, produced with a consumer level camera. In 50% of approaches the lesion is segmented from the healthy skin manually, while others have derived special algorithms, giving good results in the segmentation. I also was surprised, realizing that in a pre-processing phase, only one of the observed systems consider removal of artifacts such as hair or reflected light. A brief description of each observed approach follows.

One of the oldest approaches, I have found, is the method proposed by S. Fischer et al. in 1996 [11]. In this method a color based segmentation applied to the Karhunen-Loeve transform of the RGB color vectors to separate brightness information or enhancement of structures with a poor contrast, then a scale-space filter is applied to every histogram to calculate fingerprints that allow for determination of color classes. A coarse classification assigns pixels in intervals to a class defined by a maximum histogram, unclassified pixels are then assigned to the closest class using the fuzzy c-means technique. After that they apply local histogram equalization to enhance the pigmented network homogeneously. And finally, to smooth out impulsive noise without destroying the structure of the pigmented network, morphological grey scale closing followed by an opening. Their experiments demonstrate that using a color based segmentation does not allow for the extraction of pigmented networks because of the weak contrast within the network. However, the authors claim that it is a robust method for separating lesions from the surrounding skin and for extracting homogeneous and differently colored regions within the lesion.

In 2004 T. Tanaka et al. published the paper "'A Study on the Image Diagnosis of Melanoma"' [12], dealing with features of melanoma and nevus for computer diagnosis. For melanoma detection they do not detect a network in images, but consider one hundred five values of features based on "'ABCD"' rule. However, this work is interesting from the point of contour extracting of lesioned region, where they get good results in segmentation of images.

G. Di Leo et al. in 2008 [13] succeeded in their work to detect whether pigmented network is Atypical, Typical or Absent applying decision-tree classification techniques to the results of specific image processing algorithms. As in many other approaches, their method is divided in the sequence of steps: I) image segmentation; II) pigment network detection; III) pigment network classification. For separation of the healthy skin from lesion they do sequentially color to monochrome image conversion obtaining three different monochrome images from the source image corresponding to the red, green and blue color components, respectively; image binarization using an adaptive threshold and, finally, border identification with an adopted simple blob-finding algorithm. Structures of pigmented network are detected mainly by searching for the occurrence of texture and successively by evaluating its possible chromatic and spatial distribution. 13 features have been extracted from each network. The solutions are represented by Decision Tree Classifiers. Given a large training set, in fact, decision tree classifiers could produce rules that perform well on the training data but do not generalize well to unseen data. Sensibility and specificity are both greater than 85%.

One year later Jose Fernandez Alcon et al. [14] have designed and implemented an automatic imaging system with decision support for inspection of pigmented skin lesions and melanoma diagnosis. They do not detect pigmented network either, but follow the "'ABCD"' rule. However the work is remarkable because it supports images of skin lesions acquired using a conventional (consumer level) digital camera. Authors in their method start with background correction by simply removing the low frequent spatial component of the image to correct for the uneven illumination. To separate the pigmented lesion from the background automatically they derived an algorithm inspired by Otsu's thresholding algorithm, which relies on maximizing the between-class variance. In order to decide whether the lesion is benign or malignant they follow the "'ABCD"' rule of dermatoscopy: scores and weights of "'ABCD"' features applying algorithms for the features quantification. The system classified images with an accuracy of 86%, a sensitivity of 94%, and a specificity of 68%.

Some works were published after the observed one. Catarina Barata et al. first published their work in 2011 [15] and updated their approach in 2012 [16]. They base their approach on a bank of directional filters. The latest system is able to detect, whether a skin lesion is with or without pigment network with sensitivity of 91.1% and and specificity of 82.1%. In a pre-processing phase the system detects artifacts such as hair or reflected light and removes them from the image using directional filters to reduce the influence of artifacts and avoid false alarms. Next, regions with pigment network are detected using two of its distinctive properties: intensity (i.e., the transitions between the dark lines and the lighter "'holes"') and geometry or spatial organization (since it is assumed that the lines of pigment network form connected structures). The intensity property is used to perform an enhancement of the network by applying a bank of directional filters, while the spatial organization is used to perform the actual detection and generate a binary "'net-mask"'. The final block aims to assign a binary label to each image: with or without pigment network. To accomplish this objective, features which characterize the topology of the detected regions in a given image are extracted and used to train a classifier, using a boosting algorithm. The authors claim that their system is different from the others, because it is able to: 1) report quantitative validation, 2) present qualitative results for the location of network, 3) extract the actual network (the mesh), and 4) compare the detected regions with a ground truth.

Finally, in August, 2012, there was an article of Leszek A. Nowak et al. published [17]. For detection process they have developed an adaptive filter, inspired by Swarm Intelligence (SI) optimization algorithms. The introduced filtering method is applied in a non-linear manner, to processed dermatoscopic image of a skin lesion. The non-linear approach derives from SI algorithms, and allows selective image filtering. In the beginning of filtration process, the filters (agents) are randomly applied to sections of the image, where each of them adapts its output based on the neighborhood surrounding it. Agents share their information with other agents that are located in immediate vicinity. This is a new approach to the problem of dermatoscopic structure detection, and the authors state it to be highly flexible, as it can be applied to images without the need of previous pre-processing stage. However, the method inherited high computation complexity of the optimization problems and this makes it very difficult to develop and fine tune, as processing one image can take up to 5 minutes depending on the agents count, iterations and image size.

3 Implementation of the observed method

To see how the system works practically I decided to implement the method following the steps, described in the article. For my experiments I have chosen Matlab (MathWorks, version R2012b) for the Image Processing Toolbox, providing the users with high range of functions for image analysis.

In the given approach the following phases can be distinguished: image pre-processing, network detection and, finally, classification. Pre-processing phase includes manual segmentation of the lesion, color enhancement, conversion to the luminance image, and detection of sharp changes of intensity. At the network detection phase the holes belonging to the pigment network pattern are connected together in a graph. And, after all, the resulting graph is analyzed and the presence of pigmented network in the image is accepted or rejected.

As the authors of the observed approach stated that they used random images from the Atlas of Dermoscopy [18], for my experiments I also have taken 5 images known to be melanoma positive from the same source (Figure 1.2). I have selected only pure images without noise and artifacts like hair or light reflections for accuracy of my experiment.

As it was said in the paper, the authors used images of size 768x512px, so I scaled the images to this size, using Adobe Photoshop.

I also used Adobe Photoshop for manual segmentation, filling the regions, belonging to the free of lesion skin with black color (Figure 1.3). As I am not an expert in dermatology, that was very challenging for me to classify, whether the structure shall be accepted or rejected, which explains the need of automatical detection.

For the contrast enhancement I used the Matlab function $h = fspecial(type,\ parameters)$ (Figure 1.4). As it was described in the paper: "'A 3-by-3 contrast enhancement filter created from the negative of the Laplacian filter with parameter α is used. α controls the shape of the Laplacian and must be in the range 0.0 to 1.0. Our default value for α is 0.2"'. That corresponds to $h = fspecial('unsharp',\ 0.2)$.

To convert the image to grayscale I used the Matlab function $G = rgb2gray(RGB)$, which converts RGB values to grayscale values by forming a weighted sum of the R, G, and B components: $0.2989 * R + 0.5870 * G + 0.1140 * B$. The same parameters were mentioned in the observed article (Figure 1.5).

Figure 1.2: One of the images chosen for my experiments: original photo of a melanoma lesion.

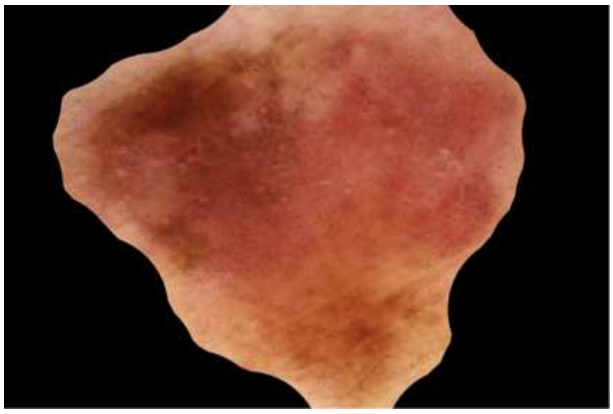


Figure 1.3: Segmented image of a melanoma lesion.

For the detection of sharp changes of intensity and transformation into a binary image the authors refer to the Laplacian of Gaussian (LoG) filter, which can detect the "'light-dark-light"' changes of the intensity well. The detection criterion is the presence of a zero crossing in the second derivative with the corresponding large peak in the first derivative. LoG looks for zero crossings and their transposes. All zeros are kept and edges lie on the zero points. If there is no zero, it arbitrarily chooses the edge to be the negative point. Therefore, when all zero responses of the filtered image are selected, the output image includes all closed contours of the zero crossing locations. The Matlab function *BW = EDGE(I,'log',THRESH,SIGMA)* was used for this purpose. The authors of the observed approach set up the threshold to zero. As the value for the sigma was not mentioned, I used the default value, which is equal to 2 (Figure 1.6).

Having a binary image of the connected components (the edges of the images), the authors convert them to a Graph (Gi) with 8-connected neighbors. Each pixel in the connected component is a node of Gi and each node has a unique label according to its coordinate.

In order to do that, the components of the binary image were labeled using the function *L = bwlabel(BW, n)*, returning a matrix L, of the same size as BW, containing labels for the connected objects in BW. The variable n can have a value of either 4 or 8, where 4 specifies 4-connected objects and 8 specifies 8-connected objects. If the argument is omitted, it defaults to 8. After that a function running through all labels and returning a list of edges for each connected component was written.

Having the list of edges as an input, it is possible to utilize the Iterative Loop Counting Algorithm [19], used by the authors of the approach, who state that morphologic techniques, used in the previous approaches are error-prone in detecting the round shape structures and therefore ILCA is a better tool to use. However, in my case it turned to be very slow for detection of structures in the whole image, and for that reason I reduced the whole number of labeles (connected components) to the certain range to continue experiment. I modified the ILCA that way, so it became possible to get the values of connected edges directly to the function, avoiding storing of intermediate values in a text file, however, it did not speed up the process very much. Found cyclic components are labeled with green (Figure 1.7).

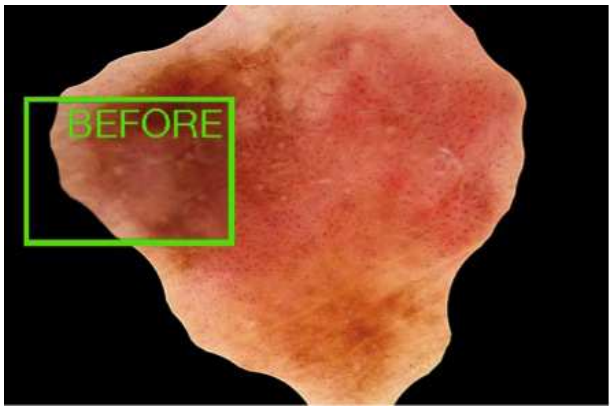


Figure 1.4: Image of a melanoma lesion with contrast enhancement implemented.

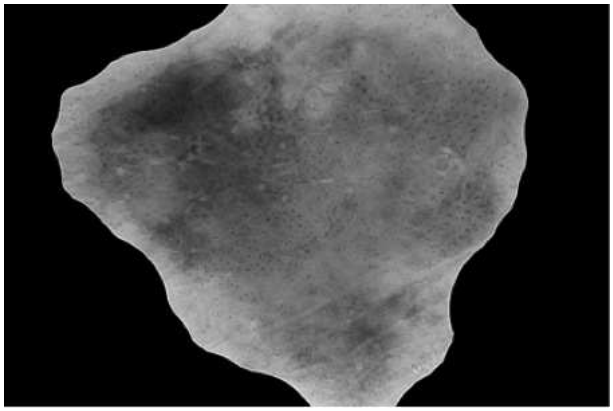


Figure 1.5: Image of a melanoma lesion converted to grayscale.

After receiving the list of edges contained in a loop, cyclic components are filtered according to the fact that in globules color of the inside area of the structure is darker than the border pixels or the outside area to avoid influence of noise such as hair or oil. To compare levels of intensity I calculate mean values of colors in grayscale for the loop area and the loop area extended by 2 pixels. If the loop area has larger intensity than extended area, the component is rejected. Rejected components marked with red color (Figure 1.8).

After identifying the holes, we are prepared to detect the presence of a pigment network pattern, creating a new graph. The cyclic structures become nodes of the new graph. Considering spatial arrangement of holes of the pigment network, a threshold for their Euclidean distance is set. Nodes within a maximum distance threshold (MDT) are connected together. The value of the MDT is computed based on the average diameter of all meshes in the image. The MDT should be proportional to the size of cyclic structures and it is defined as alpha (set to 3) times the average diameter of meshes. In order to calculate the average diameter of the meshes, it is used the parameter 'EquivDiameter' of the Matlab function "'regionprops"': *diameter = regionprops(BW,'EquivDiameter')*. It provides a scalar that specifies the diameter of a circle with the same area (*Area*) as the region with the Equation 1.1. The searched diameter is obtained by averaging the resulting vector.

$$diameter = \sqrt{\frac{4 * Area}{\pi}} \tag{1.1}$$

The nodes of the graph are the centroids of the meshes. To find the coordinates of centers the function *center = regionprops(BW,'Centroid')* is used. Having the list of center coordinates makes possible to go through it and to calculate distances between the centers of meshes. Those distances are estimated as Euclidean Distances. For each pair of center coordinates the Euclidean formula is computed with Equation 1.2. The centers separated from each other by a distance lower than 3 times MDT are connected. The labels for connectivity are stored in an adjacency matrix.

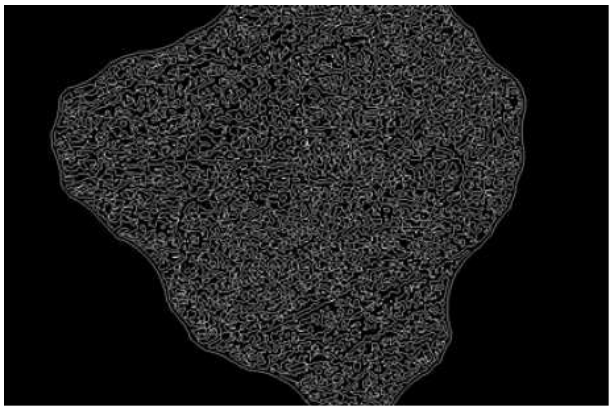


Figure 1.6: Image of a melanoma lesion transformed into a binary image.

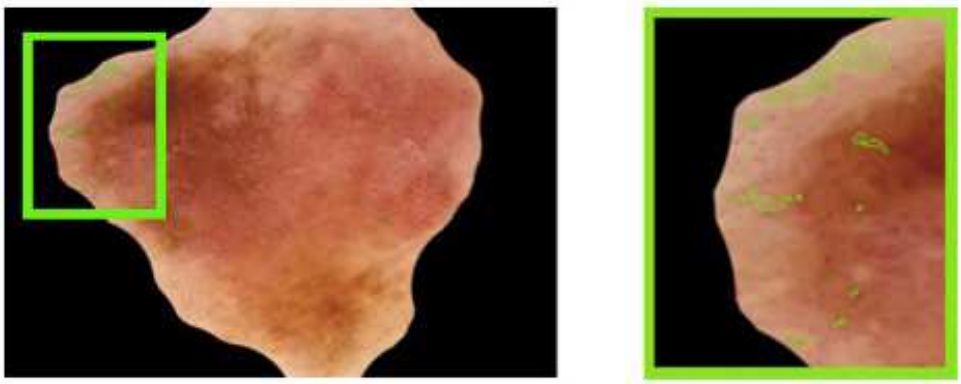


Figure 1.7: Cyclic structures found.

$$d(P1, P2) = \sqrt{(x2 - x1)^2 + (y2 - y1)^2} \tag{1.2}$$

Depending on the density of the graph, we can assume the presence of a pigment network. Density is calculated with the equation 1.3, where E is the number of edges in the graph, V is the number of nodes of the graph and *lesionSize* is the size of the area of the image within the lesion boundary being investigated.

$$density = \frac{|E|}{|V| * log(lesionSize)} \tag{1.3}$$

Images containing a density ratio higher than a threshold (set to 1.5) are classified as "'Present"' and in the opposite case - as "'Absent"'.

4 Results of my experiment

With Iterative Loop Counting Algorithm and hardware I used (Processor 1.7 GHz Intel Core i5, Memory 4Gb 1333 MHz DDR3) I failed to get any results for the whole image, because of the computation time. I tried to speed up the process, disregarding long loops in ILCA, setting up a treshold for number of edges to 100, what led me, obviously, to false results. The pigmented network in the processed images (Figure 1.9) was classified as Absent in cases known to be melanoma positive.

5 Discussion

Having my experiments done, I found some issues in the observed article.

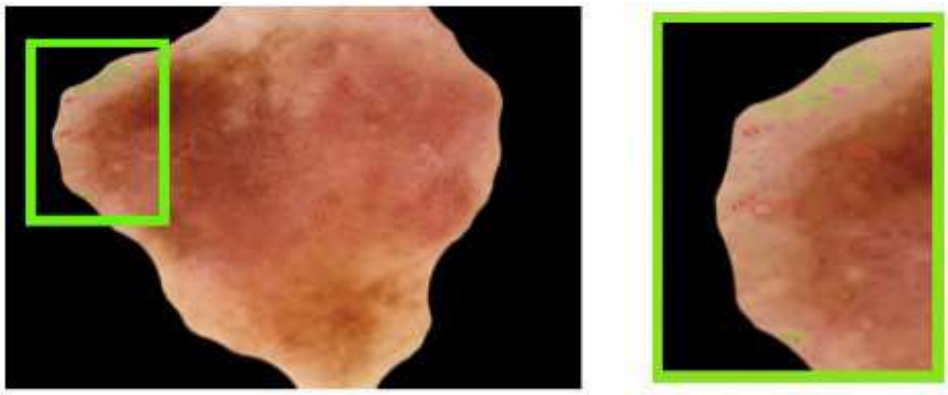


Figure 1.8: Filtering of cyclic structures.

One important issue is lesion segmentation. It turned out to be complicated to do it manually without any prior knowledge of dermatology and using just a graphic editor like Adobe Photoshop which gives a very rough output. Moreover, existing approaches already offer techniques for segmentation, which give rather good results. So, it can be the way of improvement for the observed approach.

The Iterative Loop Counting algorithm turned out to be very slow, making the whole process inefficient. It takes hours to analyze a single image. Having a big number of edges in undirected graph leads to exponential worst-case running time. It was proven that the problem of finding in a graph a cycle cover of smallest total length is NP-hard [20]. For that reason, I wonder, how the authors managed to handle this. The authors refer to morphological techniques as error-prone in detecting the round shape structures. However, implementation of morphologic techniques can speed up the process extremely, because the algorithms used for are not so complex. Thus, to find the cyclic structures, it is possible to create a binary mask layer, by filling in the structures. Matlab function *BW2 = imfill(BW,locations,conn)* can be used. Next, we can filter this masque, according to the prior knowledge of what a hole is. Knowing that a hole is small and at the same time big enough not to be a dot, we can define a threshold for its size. Than we need to discriminate the dots, globules and other dark artifacts like hair or ruler mark lines. The technique used in the observed approach, comparing levels of darkness of the structure and its rim, is appropriate for that reason. However, we also need to identify oil bubbles and white cysts, which the observed approach fail to do. Those structures, oppositely, have a very high levels of intensity and, therefore, can be filtered according to some value of threshold, which needs to be defined experimentally.

Furthermore, the authors do not define a suitable scaling for the investigated images. It is unknown, whether resolution and magnification is the same for all images of the atlas. Obviously, the influence of scaling is sufficient for the thresholds and all images shall be normalized in the pre-processing phase.

I have also noticed that detection of structures in light images and dark images gives different results, which explains the need of unification of the contrast level and illumination correction.

It also became obvious that the method works only for certain types of skin lesions. And in most of those cases the network pattern can be seen without implementing any algorithms.

6 Conclusion

Despite on the high efficiency claimed by the authors and all the issues mentioned above into account, the observed approach seems to be not suited for clinical use and requires further investigations and improvements.

References

[1] Shawn A: Melanoma Screening Saves Lifes. http://www.skincancer.org/skin-cancer-information/melanoma/melanoma-prevention-guidelines/melanoma-screening-saves-lives (accessed 22 Feb 2013).

[2] Giblin AV, Thomas JM: Incidence, mortality and survival in cutaneous melanoma. Journal of Plastic, Reconstructive & Aesthetic Surgery. 2007; 60(1): 32-40.

[3] Forsea AM, Marmol V, Vries E, Bailey EE, Geller AC: Melanoma incidence and mortality in Europe: new estimates, persistent disparities. British Journal of Dermatology. 2012; 167(5): 1124-1130.

[4] Heistein JB: Melanoma. http://emedicine.medscape.com/article/1295718-overview (accessed 21 Feb 2013).

[5] Cancer Research UK: Treating melanoma skin cancer - a quick guide. http://www.cancerresearchuk.org/ (accessed 21 Feb 2013).

[6] A.D.A.M., Inc.: Melanoma. http://www.nlm.nih.gov/medlineplus/ency/article/000850.htm (accessed 21 Feb 2013).

[7] American Cancer Society: Melanoma skin cancer overview. http://www.cancer.org/ (accessed 21 Feb 2013).

[8] Oakley A: Dermoscopy. http://www.dermnetnz.org/procedures/dermoscopy.html (accessed 21 Feb 2013).

[9] Stanganelli I: Dermoscopy. http://www.mmmp.org/MMMP/import.mmmp?page=dermoscopy.mmmp (accessed 21 Feb 2013).

[10] Sadeghi M, Razmara M, Ester M, Lee TK, Atkins MS: Graph-based pigment network detection in skin images. SPIE Digital Library. 2010 Apr 26. Proc. SPIE 7623 762312-1.

[11] Fischer S, Schmid P, Guillod J: Analysis of skin lesions with pigmented networks. IEEE. 1996; 323-6.

[12] Tanaka T, Yamada R, Tanaka M, Shimizu K, Tanaka M, Oka H: A study on the image diagnosis of melanoma. IEEE. 2004 Sep 1-5; 1597-600.

[13] Leo GD, Liguori C, Paolillo A, Sommella P: An improved procedure for the automatic detection of dermoscopic structures in digital ELM images of skin lesions. IEEE. 2008 Jul 14-16.

[14] Fernandez Alcon J, Ciuhu C, Kate W, Heinrich A, Uzunbajakava N, Krekels G, Siem D, Haan G: Automatic imaging system with decision support for inspection of pigmented skin lesions and melanoma diagnosis. IEEE Journal of Selected Topics in Signal Processing, vol. 3, no. 1. 2009 Feb; 14-25.

[15] Barata C, Marques JS, Rozeira J: Detecting the pigment network in dermoscopy images: a directional approach. IEEE. 2011 Aug 30 - Sep 3; 5120-3.

[16] Barata C, Marques JS, Rozeira J: A system for the detection of pigment network in dermoscopy images using directional filters. IEEE. 2012 Oct; 2744-54.

[17] Nowak LA, Ogorzalek MJ, Pawlowski MP: Pigmented network structure detection using semi-smart adaptive filters. IEEE. 2012 Aug 18-20; 310-4.

[18] Dermoscopy Atlas. The International Atlas of Dermoscopy and Dermatoscopy: http://www.dermoscopyatlas.com/ (accessed 25 Dec 2012).

[19] Kirk J: Count loops in a network. http://www.mathworks.com/matlabcentral/fileexchange/10722-count-loops-in-a-graph (accessed 25 Dec 2012).

[20] Thomassen C: On the complexity of finding a minimum cycle cover of a graph. SIAM J. COMPUT. 1997; 26(3): 675-676.

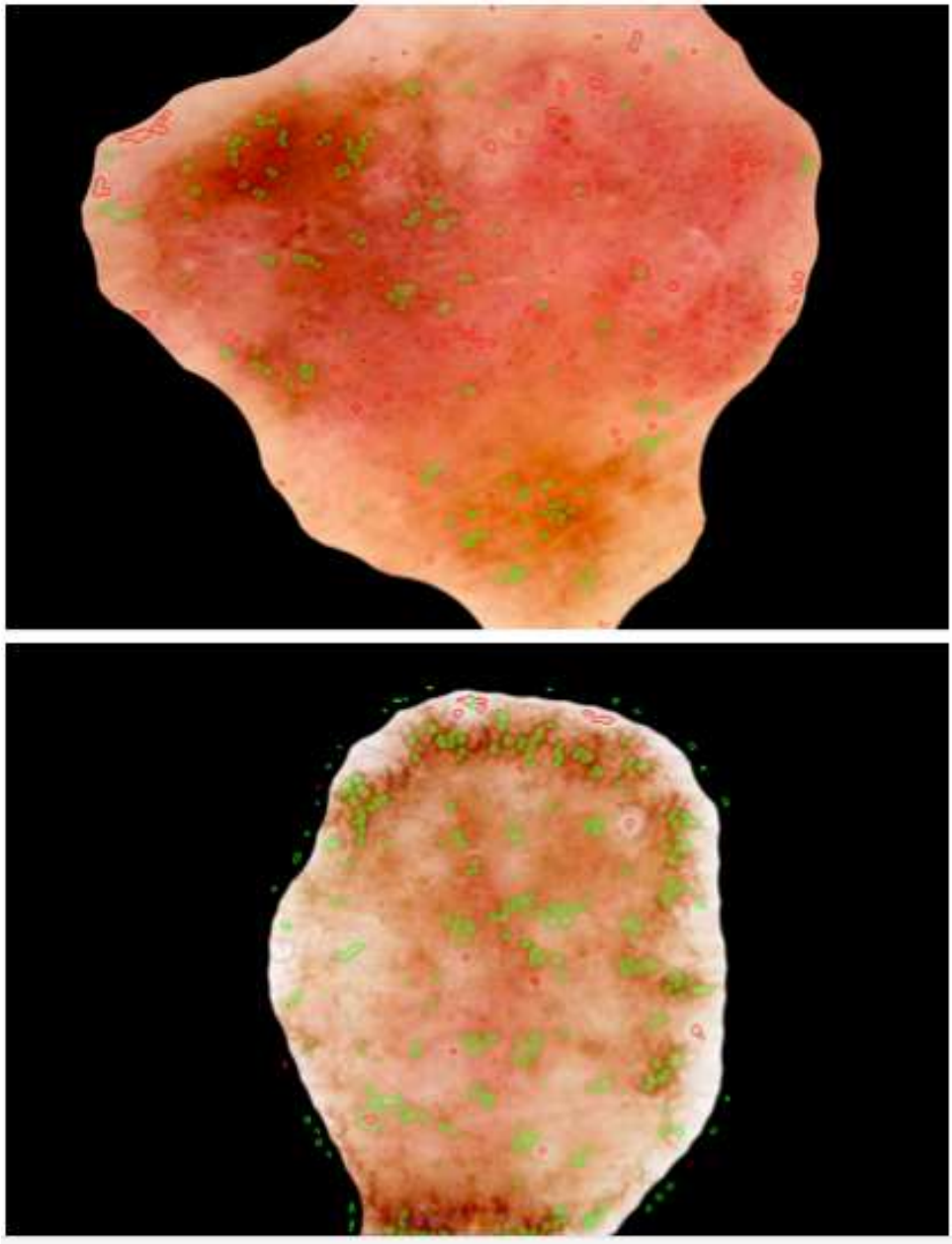


Figure 1.9: Output images of my experiment.

Reliable Path for Virtual Endoscopy: Ensuring Complete Examination of Human Organs

Erik Noorman

Inhalt

Zusammenfassung

Diese Seminararbeit behandelt eine von Taosong He, Lichan Hong, Dongqing Chen und Zhengrong Liang im Jahr 2001 vorgestellte Methode zur effizienten Berechnung eines Fly-Through-Pfades *durch ein virtuelles Hohlorgan, welcher zur Navigation bei der virtuellen Endoskopie in der medizinischen Diagnostik eingesetzt werden kann. Eine gute Navigation ist wichtig für die Zuverlässigkeit und Effektivität dieser Untersuchung. Die Suche nach einem geeigneten Pfad, um die virtuelle Kamera durch das Modell des Organs zu navigieren stellt ein schwieriges Problem dar. Die vorgestellte Methode* Reliable Path *dient dazu einen Pfad zu generieren, welcher sicherstellt, dass die gesamte Innenwand des Hohlorgans im Laufe der Untersuchung sichtbar ist.*

Keywords: Virtuelle Endoskopie, Virtuelle Navigation, Reliable Path, Reliable Navigation

1 Einleitung

In der präventiven und diagnostischen Medizin ist es wichtig eventuelle Läsionen der inneren Organe aufzuspüren und zu lokalisieren, um diese anschließend gezielt zu behandeln. Mit Hilfe der Endoskopie kann der Mediziner nach Abnormitäten der inneren Hohlorgane suchen. Eine medizinische Sonde, das Endoskop, wird dem Patienten je nach vermuteter Art der Läsion, wie z.B. Lungenkarzinome oder Darmgeschwülste, in die entsprechenden Organe eingeführt. Das Endoskop überträgt Bilddaten, welche der Mediziner zur Navigation im Organ und zur Diagnose nutzen kann. Diese Untersuchung ist mit nicht zu vernachlässigenden Risiken und Belastungen für den Patienten verbunden und sollte daher nur in Fällen mit medizinischer Relevanz angewendet werden.

Für einige Anwendungsbereiche der Endoskopie bietet die virtuelle Endoskopie eine mögliche Alternative, da sie wesentlich weniger Risiken birgt. Hierzu wird mit Hilfe eines bildgebenden Verfahrens, wie der Computertomographie (CT) oder der Magnetresonanztomographie (MRT), Schnittbilder des entsprechenden Organs gewonnen, welche anschließend mit Hilfe von automatisierten Methoden so aufbereitet werden, dass der Mediziner das Innere des virtuellen Organs am Monitor betrachten kann. Je nach implementiertem Navigationsschema wird dem Mediziner im nächsten Schritt ein Video zur Betrachtung bereitgestellt oder ihm wird die Möglichkeit gegeben, die Kamera im 3D-Modell frei zu navigieren. Neben der Lokalisierung von Läsionen, ist ein häufiges Ziel der virtuellen Endoskopie, das Operationsgebiet im Vorfeld einer realen Endoskopie oder einer Operation darzustellen, um mögliche Hindernisse und Komplikationen einschätzen zu können. Außer der Anwendung in Magen-Darm-Trakt und den Luftwegen, bietet die virtuelle Endoskopie auch die Möglichkeit Bereiche des Körpers zu untersuchen, welche mit der realen Endoskopie, aufgrund ihrer flüssigen Füllung, nur schwer oder gar nicht zugänglich sind. Dazu gehört das Aufspüren von Aneurysmen in Blutgefäßen sowie die Untersuchung der Gallenwege. Um eine möglichst gute Lokalisierung und Effizienz der Untersuchung zu erhalten, ist es wichtig die Bilddaten so aufzuarbeiten, dass der Mediziner keine Probleme hat sie zu interpretieren. Damit der Mediziner sich nicht zwischen realer und virtueller Endoskopie umstellen muss, sollte eine virtuelle Endoskopie ähnlich verlaufen wie eine reale Endoskopie. Der visuelle Eindruck der auf dem Bildschirm dargestellten Bilddaten sollte möglichst realistisch sein und die Navigation im Organ möglichst ähnlich ablaufen. Dazu werden eine Reihe von automatisierten Methoden verwendet auf die hier nicht eingegangen wird. In der vorgestellten Arbeit von He et. al geht es hauptsächlich um den Pfad, welcher zur Navigation während der virtuellen Endoskopie genutzt wird, um die virtuelle Kamera durch das Organ zu steuern.

Die vorgestellte Arbeit von Taosong He, Lichan Hong, Dongqing Chen und Zhengrong Liang aus dem Jahr 2001 handelt von dem Konzept des *Reliable Path*, welcher sicherstellen soll, dass das menschliche Organ im Verlaufe einer virtuellen Endoskopie vollständig sichtbar ist [1]. Die Autoren geben einen effizienten Algorithmus an, mit dem ein *Fly-Through-Pfad* durch ein virtuelles Hohlorgan berechnet wird, welcher der Navigation bei der virtuellen Endoskopie dient.

Diese Arbeit ist in vier weitere Abschnitte gegliedert. Im zweiten Abschnitt werden die medizinischen und technischen Grundlagen erläutert, welche dem Verständnis der folgenden Abschnitte dienen. Im dritten Abschnitt wird das Problem umrissen, welches von der vorgestellten Arbeit gelöst wird. Des weiteren enthält dieser Abschnitt vorangegangen Lösungsansätze. Der vierte Abschnitt stellt das neue Konzept des *Reliable Path* vor, einen effizienten Algorithmus für dessen Berechnung und einen Vergleich dieser Methode mit den vorangegangenen Lösungsansätzen. Der letzte Abschnitt enthält eine kritische Reflexion der vorgestellten Methode und einen Fazit dieser Arbeit.

2 Grundlagen

In diesem Abschnitt werden die zum Verständnis der folgenden Kapitel notwendigen medizinischen sowie technischen Grundlagen erläutert. Der erste Teilabschnitt handelt von den relevanten medizinischen Grundlagen und befasst sich vorrangig mit der Endoskopie. Die technischen Grundlagen werden im zweiten Teilabschnitt näher betrachtet. Neben einer Einführung in die virtuelle Endoskopie wird erklärt, wie medizinische Bilddaten aufgenommen, auf dem Computer repräsentiert und weiterverarbeitet werden.

2.1 Medizinische Grundlagen

Viele Krankheiten lassen sich anhand von strukturellen Veränderungen wie Geschwülsten, Polypen und Tumoren diagnostizieren. Um diese in der diagnostischen Medizin gezielt zu suchen bietet sich die Endoskopie an. Die Endoskopie stellt eine minimal invasive Untersuchungsmethode dar, mit der man Bilder verschiedener innerer Hohlorgane aufnehmen kann. Dazu wird dem Patienten eine medizinische Sonde, das Endoskop, in den Körper eingeführt. An der Spitze des Endoskops befindet sich eine kleine Kamera welche Bilddaten nach außen überträgt. Diese Bilddaten können dann vom Mediziner am Monitor begutachten werden. Diese Untersuchungsmethode hat neben ihren Vorteilen eine Reihe von Nachteilen: zum einen ist sie kostenintensiv, üblicherweise ist sie für den Patienten sehr unangenehm, außerdem ist sie mit vielen Risiken für den Patienten verbunden, wie innere Blutungen und bakterielle Infektionen. Ein großer, eventuell diagnostisch relevanter Untersuchungsbereich kann aufgrund von Beschwerden des Patienten und der eingeschränkten Flexibilität des Endoskop nicht erreicht werden. Die virtuelle Endoskopie, auf die im folgenden Teilabschnitt näher eingegangen wird, bietet für viele Anwendungen der realen Endoskopie eine sehr gute Alternative, da sie die aufgeführten Nachteile nicht aufweist. Für die virtuelle Endoskopie werden digitale Bilddaten der zu untersuchenden Organe benötigt.

Ein wichtiger Zweig der medizinischen Diagnostik befasst sich mit der Akquise und Verarbeitung von medizinischen Bilddaten. Es gibt eine Reihe von medizinischen bildgebenden Verfahren; wichtig für die vorgestellten Anwendungen sind vor allem die Computertomographie (CT) und die Magnetresonanztomographie (MRT). Ziel beider Verfahren ist es, digitale Schnittbilder vom menschlichen Körper zu erzeugen, welche Aufschluss über krankhafte Veränderungen von Organen geben können. Der Patient legt sich dafür auf eine Liege, welche durch eine Röhre gefahren wird. Diese Röhre enthält die bildgebende Technik; ein Detektor dreht sich im Kreis um den Patienten und erzeugt viele einzelne Schnittbilder oder ein spiralförmiges Schnittbild. Diese können dann zu einem dreidimensionalen Bilddatensatz verrechnet werden. Dies bietet dem Mediziner die Möglichkeit, die erzeugten Bilddaten für eine Reihe von Untersuchungen zu nutzen, zum Beispiel der individuellen Therapieplanung und Therapieevaluation, der Diagnostik und der bildgestützten Chirurgie [2].

2.2 Technische Grundlagen

Bei der virtuellen Endoskopie werden zunächst, mit Hilfe von medizinischen bildgebenden Verfahren, digitale Bilddaten des zu untersuchenden Organs gewonnen. Aus diesen wird anschließend, ein 3D-Bilddatensatz berechnet. Ziel der nächsten Schritte ist es, eine reale Endoskopie, mit Hilfe der aufgenommenen Bilder, möglichst realistisch nachzuempfinden. In anderen Worten: Dem Mediziner soll der Blick in das menschliche Hohlorgan ermöglicht werden, ohne dem Patienten ein Endoskop einzuführen.

Die Computergraphik ermöglicht es das aufgenommen Organ auf dem Monitor darzustellen. Allerdings müssen dazu die Volumenelemente des 3D-Bilddatensatzes, die Voxel, in drei verschiedene Gruppen aufgeteilt werden. Das Voxel enthält einen Grauwert, welcher abhängig von dem an der entsprechenden Stelle im Körper vorhandenem Gewebe ist. Es hilft dadurch verschiedene Gewebe und Strukturen voneinander zu unterscheiden. Entweder das Voxel entspricht dem inneren, äußeren oder der Wand des Hohlorgans. Jedem Voxel wird, automatisiert, einer von drei Labels zugewiesen: intern, extern oder Organwand. Die Organwand wird im folgenden Oberfläche genannt. Eine virtuelle Kamera kann nun durch das Hohlorgan navigiert werden und der Mediziner kann die Oberfläche der inneren Organwand begutachten. Allerdings ist es sehr schwer sich in einem hochverzweigtem Organ, wie z.B. der Lunge, zurecht zu finden. Es ist deswegen erforderlich den Mediziner bei der Navigation durch das Organ zu unterstützen. Es bieten sich verschiedene Navigationsschemata an. Die erste Möglichkeit ist dem Mediziner eine freie Navigation der Kamera im 3D-Modell des Organs zu ermöglichen. Jedoch kann es mitunter sehr schwierig sein sich in hochverzweigten Hohlorganen zurecht zu finden. Damit läuft der Mediziner Gefahr, die Orientierung zu verlieren und diagnostisch relevante Bereiche auszulassen. Als zweite Möglichkeit bietet es sich daher an,

die Kamera auf einem zuvor berechneten *Fly-Through-Pfad* durch das Hohlorgan zu navigieren. Dem Mediziner wird dadurch die komplizierte Aufgabe der Navigation abgenommen. Die so gewonnenen Bilder können dann als Video aufgezeichnet und anschließend dem Mediziner am Monitor gezeigt werden. Diese Methode schränkt den Mediziner jedoch stark ein, da er keine Möglichkeit mehr hat die Blickrichtung der Kamera zu beeinflussen. Die dritte Möglichkeit kombiniert die Vorteile der zuvor genannten Methoden. Die Kamera durchfliegt das Hohlorgan anhand eines zuvor berechneten *Fly-Through-Pfades*. Die Bilder werden jedoch nicht als Video aufgezeichnet, sondern direkt am Monitor zur Begutachtung dem Mediziner angezeigt. Dieser erhält nun die Möglichkeit den Fokus der Kamera zu manipulieren, indem er die Blickrichtung, den Zoom und die Geschwindigkeit der Kamera, seinen Wünschen entsprechend, einstellt. Der Mediziner muss sich dadurch nicht mehr um die Navigation kümmern und kann sich auf die Untersuchung konzentrieren. Jedoch ist diese Methode der Navigation bei weitem die rechenaufwendigste, daher sind effiziente Algorithmen zwingend erforderlich.

3 Problem

Bei der realen Endoskopie steuert der Mediziner das Endoskop und damit die Kamera. Er legt damit fest, welche Bereiche während der Untersuchung sichtbar sind und welche nicht. Eine entscheidende Frage bei der virtuellen Endoskopie ist, wie die Bilddaten aufgearbeitet werden sollen, um sie dem Mediziner zur Begutachtung zu präsentieren. Es existieren eine Reihe von Navigationsschemata, welche in Abschnitt 2.2 beschrieben sind. Die Grundlage für die virtuelle Endoskopie bietet ein dreidimensionaler Bilddatensatz des zu untersuchenden Organs, der bereits in interne, externe und Oberflächenvoxel unterteilt ist. Das Problem besteht darin, einen *Fly-Through-Pfad* durch das Hohlorgan zu generieren, welcher sich für die Kameranavigation eignet. Dazu muss der Pfad folgende Kriterien erfüllen:

1. Der Pfad sollte niemals durch die Wand des Hohlorgans verlaufen.
2. Der Pfad sollte in der "Mitte" des Hohlorgans verlaufen.
3. Der Pfad sollte möglichst kurz sein.

Aus dem ersten Kriterium lässt sich folgern, dass der Pfad innerhalb des Organs verlaufen sollte. Daher lässt es sich leicht erfüllen, indem man nur interne Voxel für die Pfadgenerierung zulässt. Eine restriktivere Anforderung ist, dass der Pfad Abstand von der Oberfläche halten sollte, damit die Kamera einen guten Überblick über die Organwand erhält. Diese Anforderung soll das zweite Kriterium sicherstellen. Das dritte Kriterium erhöht die Effizienz der Untersuchung. Es gibt allerdings noch weitere Kriterien die ein, für die virtuelle Endoskopie geeigneter, *Fly-Through-Pfad* erfüllen sollte:

4. Der Pfad sollte kontinuierlich sein und keine Sprünge und/oder scharfe Kurven enthalten.
5. Jeder Punkt der Oberfläche sollte von einem Punkt auf dem Pfad zu sehen sein.

Das vierte Kriterium sollte erfüllt sein, damit der Mediziner die Orientierung nicht verliert. Damit keine für die Diagnose relevanten Bereiche ausgelassen werden, sollte die vollständige Innenwand des Organs im Verlaufe der Untersuchung sichtbar sein. Dies wird von Kriterium 5 sichergestellt. Der in Abschnitt 4 vorgestellte Verfahren *Reliable Path* von He et al. erfüllt insbesondere das fünfte Kriterium im Gegensatz zu alternativen Verfahren, wie dem im Abschnitt 3.1 vorgestellten *Skeleton*.

3.1 Bisheriger Lösungsansatz: Skeleton

Bevor im nächsten Abschnitt der *Reliable Path* eingeführt wird, wird in diesem Abschnitt kurz auf *Skeleton* eingegangen. *Skeleton* stellt ein alternatives Verfahren dar um einen *Fly-Through-Pfad* durch ein virtuelles Organ zu berechnen. Die Vor- und Nachteile der beiden Verfahren werden dann in Abschnitt 4.3 erörtert.

Ein häufig verwendeter Lösungsansatz ist *Skeleton*, auch *Medial Axis* genannt, da dieser viele der in Abschnitt 3 geforderten Eigenschaften erfüllt. Wenn man zu einem Punkt p im Modell die minimal Distanz zur Oberfläche des Modells sucht, ist diese Strecke meist eindeutig definiert durch den Punkt selbst und den Punkt auf der Oberfläche, welcher den minimalen Abstand zu p hat. Eine Definition des *Skeleton* ist die Menge alle Punkte innerhalb des Modells, zu denen es jeweils mehr als eine Punkt auf der Oberfläche gibt, welche minimalen Abstand zu dem Punkt haben [3]. Eine alternative Definition des *Skeleton* ist, die Menge der Zentren von maximalen Kugeln, das heißt, von Kugeln in einer Region P

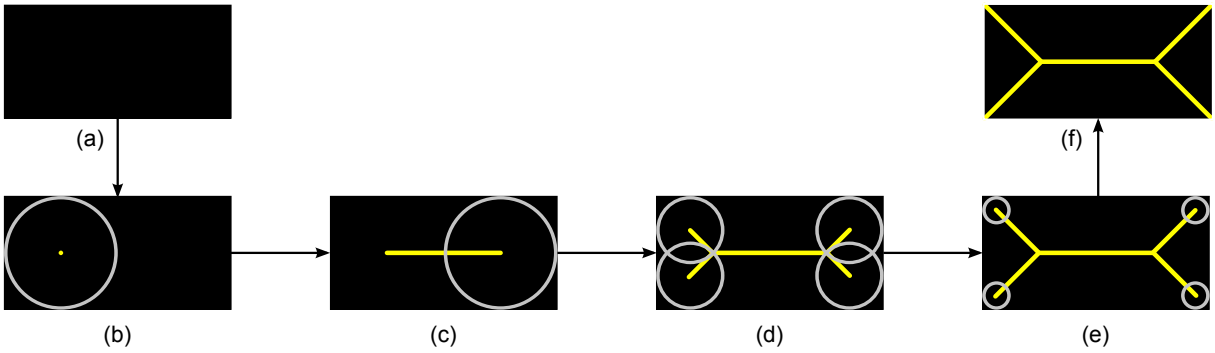


Abb. 2.1: Erstellung des *Skeleton* eines Modells

welche selbst nicht vollständig enthalten sind in einer anderen Kugel in P [4]. Das *Skeleton* ist also eine Menge von zusammenhängenden Punkten innerhalb des Modells und ergibt somit einen Pfad durch das Modell, welcher nahe der "Mitte" des Modells liegt.

In Abbildung 2.1 soll die Erstellung des *Skeleton* eines Modells visualisiert werden. Der Prozess startet bei Bild (a). Das schwarze Rechteck stellt das Modell dar, von dem das *Skeleton* berechnet werden soll. Der graue Kreis in Bild (b) stellt die maximale Kugel dar. In dessen Mitte befindet sich, in gelb, das Zentrum der Kugel. Schiebt man nun diese maximale Kugel von links nach rechts und speichert sich alle Zentren, gelangt man zu Bild (c). Da keine weiteren maximalen Kugeln dieser Größe mehr in das Modell passen, nutzt man nun immer kleine Kugeln, wie in Bild (d) und (e). Der Prozess endet mit Bild (f). Dieses enthält, in gelb, das *Skeleton* des Modells.

Aus der Sicht der Navigation hat *Skeleton* per Definition die Eigenschaft, dass dessen Punkte in der Nähe der Mitte liegen und erfüllt somit Kriterium 2. Die Menge der Zentren entspricht im dreidimensionalen Bilddatensatz einer Teilmenge der internen Voxel, wodurch Kriterium 1 erfüllt ist. Da alle internen Voxel in mindestens einer maximalen Kugel enthalten sind, entspricht die Vereinigung aller Kugeln den internen Voxeln. Zugleich entspricht die Hülle der maximalen Kugeln der Grenze zu der Oberfläche, da jeweils mindestens eine maximale Kugel existiert, welche die Oberfläche berührt. In andern Worten, die vollständige innere Oberfläche des virtuellen Organs ist sichtbar und damit erfüllt *Skeleton* Kriterium 5. Das *Skeleton* beschriebt somit einen Pfad innerhalb des Modells welcher zur Navigation genutzt werden kann. *Skeleton* erzeugt per Definition viele zusätzliche Abzweigungen, welche für die Navigation nicht zwangsläufig notwendig sind. Daher sind Kriterium 3 und 4, ohne zusätzliche Nachbearbeitung des von *Skeleton* erzeugten *Fly-Through-Pfades*, nicht zufriedenstellend erfüllt.

4 Reliable Path

Anfangs lag der Fokus der Forschung der virtuellen Endoskopie darin, die reale Endoskopie zu simulieren. Daher haben alle bisherigen Navigationstechniken ein ernstzunehmendes Problem: Ein Mediziner kann sich nicht sicher sein, alle vorhandenen Abnormitäten gesehen zu haben, nachdem eine virtuelle Endoskopie durchgeführt wurde. Das liegt daran, dass die von bisherigen Verfahren berechneten *Fly-Through-Pfade* nicht garantieren können, dass jede interne Oberflächenregion von mindestens einem Punkt auf dem Pfad sichtbar ist. Der Kern des Ansatzes von He et al. ist es, vorab eine *Fly-Through-Pfad* zu berechnen, welcher die Beseitigung von blinden Bereichen während der Navigation sicherstellt. Solch ein Pfad wird im folgenden zuverlässiger Pfad, beziehungsweise *Reliable Path* genannt. Konzeptionell ist ein zuverlässiger Pfad in einem virtuellen Organ eine Menge von zusammenhängenden Punkten innerhalb des Modells, von denen die vollständige Oberfläche sichtbar ist. Diese Definition des zuverlässiger Pfades lässt eine große Menge von Pfaden zu, welche sich nicht zur Navigation eignen. Zum Beispiel denjenigen Pfad, welcher alle internen Punkte beinhaltet; dieser Pfad wäre viel zu lang und könnte keine effiziente Untersuchung garantieren. Auf Grund dessen ist es sinnvoll den kürzesten dieser Pfade für die Navigation zu wählen. Der optimale zuverlässige Pfad oder *Optimal Reliable Path* ist definiert als derjenige zuverlässige Pfad mit minimaler Länge.

4.1 Algorithmus

Das Problem des optimalen zuverlässigen Pfades ist laut He et al. NP-vollständig. Die Autoren geben einen Beweis der NP-vollständigkeit an, welche jedoch nicht vollständig und somit falsch ist; dies wird in Abschnitt 4.2 näher betrachtet. Falls das Problem wirklich NP-vollständig ist, ist davon auszugehen, dass kein effizienter Algorithmus zu dessen Berechnung existiert. In diesem Abschnitt wird der von He et al. entwickelte heuristische Algorithmus zur Berechnung des optimalen zuverlässigen Pfades vorgestellt. Der hier vorgestellte Algorithmus berechnet somit eine Pfad der nicht von minimaler Länge ist; die minimale Länge wird lediglich angenähert.

Damit relevante Bereiche mit einer ausreichenden Detailgenauigkeit dargestellt werden, dürfen diese nicht so klein auf dem Monitor erscheinen, dass der Mediziner sie nicht erkennen kann. Neben den bereits in Abschnitt 2.2 genannten Kriterien eines *Fly-Through-Pfades* ist es in der Praxis daher außerdem wichtig, dass die zu visualisierenden Strukturen nicht zu weit von der Kamera entfernt sind. Dazu muss zu jedem Punkt auf der Organwandoberfläche ein Punkt auf dem Pfad existieren, welcher einen maximalen Abstand nicht überschreitet. Dieser Abstand muss geeignet gewählt werden.

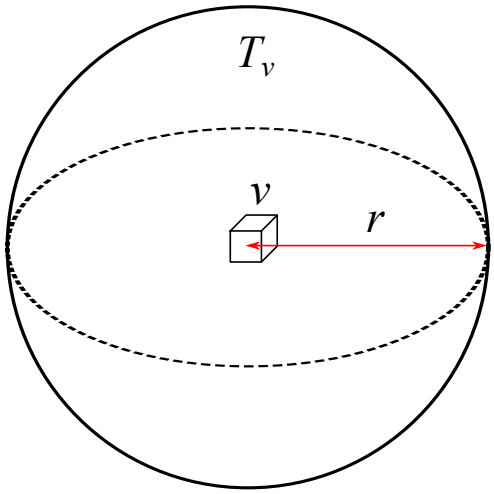


Abb. 2.2: *Visible Set*

Die Menge von Oberflächenvoxel, welche von einem internen Voxel sichtbar sind wird *Visible Set* genannt. Zur Erfüllung des zuvor genannten Kriteriums wird diese Menge nun eingeschränkt. Dies wird in Abbildung 2.2 dargestellt. Zuerst wird ein Radius r festgelegt. Dieser definiert die maximale Sichtweite. Von einem internen Voxel v sind alle Oberflächenvoxel sichtbar, dessen Abstand zu v maximal r beträgt. Alle Oberflächenvoxel dessen Abstand zu v größer ist als r sind somit nicht sichtbar von v. Somit wird das *Visible Set* auf ein kugelförmiges Volumen mit Radius r eingeschränkt. Außerdem wird die Berechnung, der von einem internen Voxel aus sichtbaren Oberfläche, vereinfacht, da nur noch eine Teilmenge der Oberflächenvoxel betrachtet werden muss.

Der Algorithmus besteht aus zwei Phasen. Die erste Phase (Schritte 3–6) ermittelt eine Prioritätsliste von zentralen Voxeln, welche man bevorzugt in den Pfad aufnimmt, da angenommen wird, dass diese bereits die Mehrheit der Oberflächenvoxel abdecken. Die zweite Phase (Schritte 7–9) betrachtet die noch nicht abgedeckten Oberflächenvoxel und garantiert, dass der resultierende Pfad zuverlässig und effizient ist. Die einzelnen Schritte lauten wie folgt:

- Im ersten Schritt wird das dreidimensionale Volumen V aus den zweidimensionalen Schnittbildern berechnet.
- Im zweiten Schritt werden die Voxel des Volumens V in externe E, interne I und Oberfläche B unterteilt.
- Im dritten Schritt wird für jedes Voxel aus der Menge I die kürzeste Distanz zur Oberfläche mit Hilfe der euklidischen Distanztransformation berechnet.
- Im vierten Schritt wird eine Menge L initialisiert, welche alle Voxel aus Schritt drei enthält, die den lokalen Maxima entsprechen. Anschließend wird die Menge nach absteigender Distanz sortiert. Die

Berechnungszeit des Sortierens ist annehmbar, da die lokalen Maxima nur eine kleine Teilmenge aller internen Voxel sind.

- Im fünften Schritt werden Voxel v der Menge L der Reihe nach dem zu erzeugenden Pfad hinzugefügt, da diese schon einen Großteil der Oberflächenvoxel abdecken. Dafür muss das *Visible Set* für jedes v berechnet werden. Falls ein v keine neuen Oberflächenvoxel sichtbar macht, fügt man es dem Pfad nicht hinzu. Die Menge U wird mit der Menge aller Oberflächenvoxel initialisiert. Für jedes zum Pfad hinzugefügte v entfernt man alle von v sichtbaren Oberflächenvoxel aus der Menge U.
- Im sechsten Schritt werden die im Pfad vorhandenen Voxel, mit Hilfe des Dijkstra Algorithmus zur Berechnung kürzester Pfade, miteinander verbunden.
- Im siebten schritt werden die Voxel v' welche auf den kürzesten Wegen liegen dem Pfad hinzugefügt. Für jedes zum Pfad hinzugefügte v' entfernt man alle von v' sichtbaren Oberflächenvoxel aus der Menge U.
- Im achten Schritt betrachtet man nun die Menge der noch nicht gesichteten Voxel U, da diese nach den vorherigen Schritten wesentlich kleiner als die initiale Menge I ist. Dadurch wird die Anzahl der durchlaufenen Voxel drastisch minimiert. Für jedes noch nicht gesichtete Grenzvoxel v' in U wird das *Visible Set* berechnet. Diese enthält alle internen Voxel v, von denen v' sichtbar ist. Anschließend wird dem Pfad das v mit dem größten Abstand zu v' hinzugefügt, da dieser nahe der Mitte liegt und von ihm in der Regel die meisten Oberflächenvoxel sichtbar sind.
- Im neunten Schritt werden die im achten Schritt hinzugefügten Voxel, mit Hilfe des Dijkstra Algorithmus zur Berechnung kürzester Pfade, verbunden.

Zur Übersicht wird der Algorithmus in Abbildung 2.3 als Pseudocode zusammengefasst:

1. Reconstruct volume V from 2D slices;
2. Classify V into: external E, internal I, surface B;
3. Perform Euclidean distance transform on I;
4. Initialize L with all the local maximum voxels;
   Sort L in the order of decreasing distance values;
5. $P = \emptyset$; $U = B$;
   Repeat until $L == \emptyset$
   $v = L.head$; $L = L - v$; get *Visible Set* T_v for v;
   if $(T_v \cap U \neq \emptyset)$ $P = P \cup \{v\}$; $U = U - T_v$;
6. Connect P into $C \subset I$ using shortest path algorithm;
7. $P' = C - P$; $U = U - \bigcup_{v \in P'} T_v$; $P = \emptyset$;
8. Repeat until $U == \emptyset$
   select $v \in U$; get $K \subset I$ that is visible from v;
   select $v' \in K$ with the highest distance value;
   $P = P \cup \{v'\}$; $U = U - T'_v$;
9. Connect $C \cup P$ into $C \subset I$ using shortest path algorithm;

Abb. 2.3: Pseudocode des *Reliable Path* Algorithmus.

4.2 Komplexität

Das Problem des optimalen zuverlässigen Pfades ist laut He et al. nicht effizient berechenbar. Diese Aussage basiert auf deren Beweis, dass das *Optimal Reliable Path* Problem NP-vollständig ist. Um die NP-schwere des Problems zu beweisen geben die Autoren eine polynomielle Reduktion vom *Minimal Set Cover* Problem zum *Optimal Reliable Path* Problem an. Die Reduktion erzeugt aus einer *Minimal Set Cover* Instanz: die Menge der Oberflächenvoxel, die Menge der internen Voxel und pro Oberflächenvoxel dessen *Visible Set*. Jedoch fehlen die geometrischen Angaben zu den genannten Voxel, also die Angaben wie diese Voxel im Bilddatensatz der *Optimal Reliable Path* Instanz anzuordnen sind. Durch das Fehlen

dieser relevanten Informationen ist die Reduktion als falsch anzusehen. Daraus folgt, dass der Beweis der NP-Vollständigkeit des *Optimal Reliable Path* Problem nicht vollständig ist und daher ebenfalls falsch ist.

4.3 Vergleich mit Skeleton

Skeleton wurde nicht für die Navigation entwickelt und eignet sich für diese auch nur bedingt. In Abbildung 2.4 ist ein Vergleich eines erzeugten *Fly-Through-Pfades* durch *Skeleton* und *Reliable Path* zu sehen. Der vom *Skeleton* erzeugte *Fly-Through-Pfad* enthält viele kleine Abzweigungen. Während der Navigation müsste die Kamera viele kleine Drehungen ausführen, die es dem Mediziner schwer machen sich zu orientieren und die Daten zu interpretieren. Des weiteren ist der Pfad viel länger als es zur Navigation nötig wäre. Auf der rechten Seite der Abbildung 2.4 ist ein möglicher *Reliable Path* zu sehen, welcher zu einer besseren Navigation führen würde, da die Kamera nur gerade durch das Modell fliegt und sich nicht, wie beim *Skeleton* Pfad, ständig drehen muss. Weiter ist die Berechnung eines 3D-*Skeleton* ein schweres Problem, vor allem für große Datensätze. Es existieren eine Reihe von effizienten Algorithmen zur Berechnung eines approximativen *Skeletons* von diskreten 3D-Datensätzen (z.B., [5], [6]), wie solche die in der virtuellen Endoskopie genutzt werden. Jedoch bewahren diese nicht mehr die Eigenschaft der Zuverlässigkeit. Zusammenfassend ist festzuhalten, dass *Reliable Path* viele Vorteile gegenüber *Skeleton* bietet.

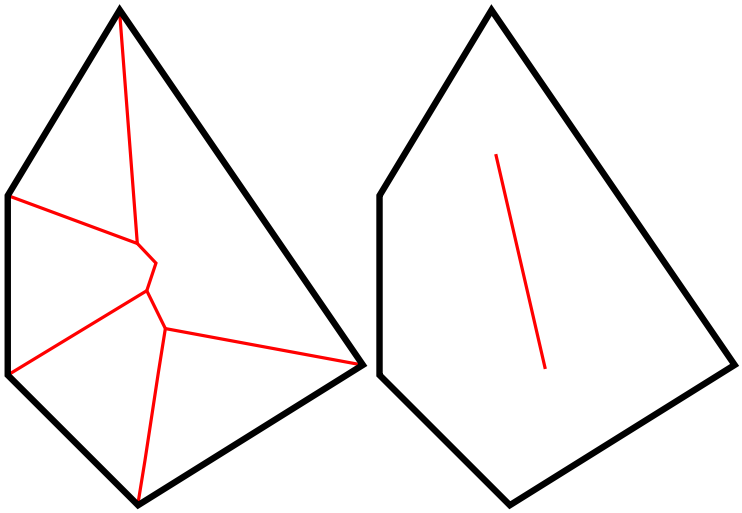


Abb. 2.4: (a) 2D-*Skeleton* für ein konvexes Polygon; (b) ein *Reliable Path*, der zu einer besseren Navigation führt.

5 Diskussion

Das vorgestellte Konzept des *Reliable Path* und dessen heuristischer Algorithmus, welche in den vorangegangenen Abschnitten ausführlich dargestellt wurden, bieten einen sinnvollen Lösungsansatz des Problems der Navigation im virtuellen Organ. Der größte Vorteil des vorgestellten Konzepts ist die Garantie, dass während eine virtuellen Endoskopie die Organwand vollständig sichtbar ist. Dies ist ein Vorteil, welcher in der praktischen Anwendung einen realen Mehrwert gegenüber anderen Navigationstechniken darstellt, da der Mediziner gewiss sein kann, dass alle vorhandenen Abnormitäten der inneren Organwand während der virtuellen Endoskopie sichtbar sind. Allerdings bedeutet dies auch, dass es nicht möglich ist irrelevante Bereiche zu überspringen [7]. Außerdem ist nicht ersichtlich, ob alternative Verfahren, an praktischen Datensätzen angewendet, Bereiche diagnostischer Relevanz nicht anzeigen. Ob die vorgestellte Methode einen Mehrwert im Gegensatz zu bereits vorhandenen Methoden darstellt, muss noch anhand von Vergleichen mit praktischen Datensätzen bewiesen werden. Leider enthält die Arbeit von He et al. keine Laufzeitanalyse des heuristischen Algorithmus; dies macht es schwer den Algorithmus mit alternativen Algorithmen zu vergleichen, welche ebenfalls das Problem der Navigation im virtuellen Hohlorgan lösen. Das Hauptproblem des Algorithmus ist es den Kugelradius zu bestimmen, welcher festlegt, wie viele Voxel

in die Berechnung des *Visible Sets* einbezogen werden. Hier wird derselbe Radius für alle Punkte verwendet. Es wäre allerdings sinnvoll einen adaptiven Radius zu bestimmen, welcher abhängig von lokalen Eigenschaften des Pfadpunktes ist [8]. Obwohl der vorgestellte heuristische Algorithmus zur Berechnung des *Reliable Path* nur anhand von drei Datensätzen getestet wurde, ist bereits ein hohes Potenzial der Methode zu erkennen. Grund dafür ist, dass die berechneten Pfade die in Abschnitt 3 beschriebenen Kriterien erfüllen. Wie bereits in Abschnitt 4.2 beschrieben, wird zwar ein Beweis angegeben, warum der Optimal Reliable Path nicht effizient berechnet werden kann, jedoch fehlen dem Beweis wichtige Angaben, welche für die Korrektheit zwingend nötig wären.

Literatur

[1] He T, Hong L, Chen D, Liang Z: Reliable path for virtual endoscopy: ensuring complete examination of human organs IEEE Transactions on Visualization and Computer Graphics, 2001; 7(4): 333–342

[2] Landini L, Positano V, Santarelli M: 3D Medical Image Processing. Image Processing in Radiology. 67–85, Springer Berlin Heidelberg, 2008.

[3] Blum H: A Transformation for Extracting New Descriptors of Shape. In Models for the Perception of Speech and Visual Form. 362–380, MIT Press, 1967.

[4] Duda RO, Hart PE, Stork DG: Pattern Classification. Wiley, 2. Auflage 2001.

[5] Zhou Y, Toga AM: Efficient Skeletonization of volumetric objects. IEEE Transactions on Visualization and Computer Graphics, 1999; 5(3): 196–209

[6] Capson DW, Fung AC: Connected Skeletons from 3D distance transforms. In IEEE Southwest Symposium on Image Analysis and Interpretation, 174–179, 1998.

[7] Bartz D, Wu Y: Advanced virtual medicine: Techniques and applications for virtual endoscopy. In ACM SIGGRAPH Course 52, 2002.

[8] Li G, Tian J, Zhao M, He H: Three-dimensional interactive virtual endoscopy. Journal of X-Ray Science and Technology, 2004; 12(3): 129–141

Region Matching For Object Categorization

Hendrik Pesch

Contents

Abstract

Enforcing spatial consistency is useful to improve the performance of object categorization methods. One way to enforce spatial consistency is by matching similar image regions with geometric consistency constraints. This seminar report presents two current methods that use region matching, the MRF-kernel [1] and region-to-image matching [2]. We will discuss the methods and compare their performance. We will also give examples of possibles applications in the field of medical image processing.

Keywords: Image matching, object categorization, medical image processing, Markov random field kernel, region-to-image matching

1 Introduction

In this seminar report we will consider the problem of object categorization. More precisely, we will be concerned with the application of region matching methods to perform object categorization. Image matching approaches can be adapted to enforce spatial consistency constraints on the images. The spatial relation of object parts or regions gives additional information that should help to discriminate between object categories. However, the spatial relation is ignored in the typical bag-of-visual-words framework that is often used in state-of-the-art object categorization methods. Thus, it is of interest to develop methods and algorithms that use the spatial relations to increase recognition performance. We will consider two such methods.

The first method presented [1] matches similar regions in two given images under some geometric constraints. The geometric constraints are encoded in a Markov Random Field (MRF). The energy values of the optimized MRF are used to construct a kernel. The kernel is then suitable for use in a Support Vector Machine (SVM), which performs the final categorization.

The second method was proposed in [2]. It is not concerned with a matching between regions of two images, but with a matching from the regions of one image to the whole second image. The geometric constraints are thus encoded within the regions and the objective function that is used to establish a final matching. This objective function resembles a string matching problem and can be solved using dynamic programming.

1.1 Relation to Medical Image Processing

This seminar report will discuss the proposed methods in the context of general object categorization. But this does of course not mean that they are not applicable in medical image processing settings. On the contrary, it should be expected that methods that are good at general object categorization should perform just as well in a more narrow defined domain.

Object categorization is of interest for medical image processing for several reasons. It reduces the amount of manual work, for example by performing automatic labelling of images in a medical database. This in turn can be useful for image retrieval based on keyword search. The image label can also be used as additional powerful feature for example-based image retrieval. Both image retrieval scenarios are of interest in medical settings. The existence of the annual ImageCLEF evaluation campaign underlines the importance of automatic annotation for the medical field.

Image matching methods have actually been used for automatic annotation of medical images. An example would be [3], where the Image Distortion Model and pseudo 2-dimensional Hidden Markov Models were successfully used for this task. The same group also participated in the aforementioned ImageCLEF competitions [4].

But image matching methods cannot only be used for automatic annotation. The provided matchings can be used to align two given images. The process of computing an alignment of two images is called image registration. Medical image registration is of interest for several reasons. Often it is the case that medical practitioners have to combine multiple information sources to get a better overview over the problem at hand. These information sources could for example be a MRI image and a PET image of the same patient. An example of the registration of such images is given in Figure 3.1. Not only different types of images could be aligned, but also the same type of image as it changes over time. An overview of medical image registration and methods that are used is given in [5].

The rest of the seminar report is structured as follows. First we will review the state-of-the-art in object categorization. Then the aforementioned MRF-kernel will be described. This section also contains

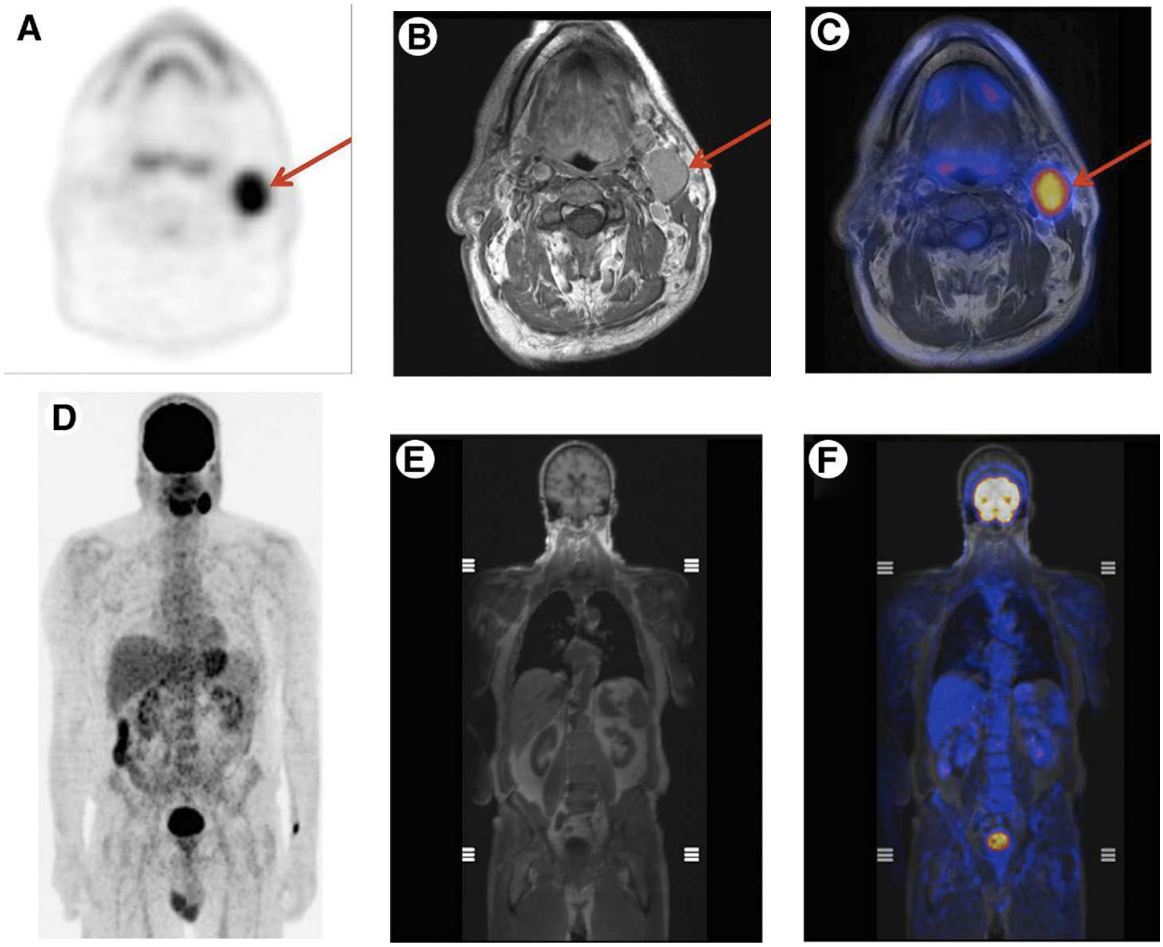


Figure 3.1: Examples of medical image registration. Left column: PET image; Middle column: MRI images; Right: Fusion of PET and MRI images after registration. Images taken from [6].

details about the optimization procedure that is used to optimize the MRF. After that the region-to-image matching approach is described in more detail. In the section on experiments we will see the actual performance of the two presented methods, including a direct comparison. Finally, a conclusion and discussion of the methods and their corresponding papers is given.

2 State of the Art

In this section we will review three state-of-the-art methods for object categorization. The methods are the Naive Bayes Nearest Neighbour (NBNN) method by [7], the subcategory learning method by [8] and Group-Sensitive Multiple Kernel Learning (GSMKL) by [9]. These methods were chosen for two reasons. First, they were evaluated on the same databases as the image matching methods that are covered in detail in this seminar report. Furthermore, they achieved competitive results on these databases. Second, they are not based on image matching. This enables us to do a direct comparison between methods that employ image matching and those that do not.

The NBNN approach is surprisingly simple. For each test image a bunch of image descriptors are extracted. The classifier then computes a score for each class by computing the sum of distances between the descriptors of the test image and their nearest neighbour in the current class. The class with the lowest score is the predicted class. This can be formalized in the following decision rule:

$$r(x) = \underset{c}{\operatorname{argmax}} \sum_{i=1}^{n} ||d_i(x) - NN_c(d_i(x))||^2$$

Here x denotes the input image, with corresponding descriptors $d_1(x), \ldots, d_n(x)$. $NN_c(d_i(x))$ is the nearest neighbour of descriptor $d_i(x)$ in class c. This decision rule describes an image-to-class distance

rather than an image-to-image distance. The authors show that the NBNN method approximates the optimal naive Bayes classifier. Furthermore, no quantization of the image descriptors is required, which according to the authors degrades the performance of nearest neighbour based classification methods. At the time of publication the authors were able to outperform several state-of-the-art methods on the Caltech databases.

The method by [8] is based on subcategory learning. Subcategories can be thought of as object parts and can also be shared between object categories. The approach is based on detecting subcategories in a test image and predicting a category based on the likelihood of the subcategory given the class, a prior on the subcategory and a relevance for the subcategory for the given class. However, the subcategories in an input image are not obtained through image matching, but by computing a segmentation tree representation of the image. The authors describe how all the learned parameters can be converted to a feature vector with which a linear classifier can be learned. This approach was able to outperform prior work on the Caltech databases.

Finally, the GSMKL approach tries to accomodate for intra-class diversity and inter-class correlation. This is done by defining an intermediate representation between images and classes. This intermediate representation is called a group. Images belong to the same group if they belong to the same object category and are strongly correlated. The classifier that is used in this work is a kernel-based SVM. The kernel itself is a weighted sum of several kernel functions. The weights are learned during training. In the GSMKL approach the training incorporates the group membership information of each image to optimize the kernel weights. This method achieved good results on the Caltech 101 database and outperformed even the subcategory learning approach by [8]. However, no results on the more difficult Caltech 256 database are reported.

We will see detailed results for all these methods in the results chapter and compare them to the image matching approaches that are covered in detail in the next two chapters.

3 An MRF-based Kernel For Object Categorization

In this section we will introduce the object categorization approach proposed in [1]. The method makes use of a Markov Random Field (MRF). MRFs are often utilized in computer vision for structured prediction tasks, where we are interested in predicting a structure that is more general than just simple class labels. Examples include for example binary masks for image segmentation, where the output space would correspond to $\{0,1\}^{M\times N}$ for an $M \times N$ image. In the region-to-region matching task the structured output is a displacement field. MRFs are graphs, in which the nodes correspond to the elements for which we would like to predict a label. The edges in the MRF encode the dependence of a nodes label on the labels of other nodes. The compatibility of the labels assigned to a node and its neighbours is encoded in a term called the binary potential. The compatibility of a label and the observation at this node is encoded by the so-called unary potential. Each configuration of an MRF has an associated energy value, which is given by the sum of all unary and binary potentials given the current states. Figure 3.2 shows the structure of an MRF. The shown structure is typical for computer vision applications, though different structures are also possible.

The region-to-region matching problem can also be formulated as an MRF. The presented method represents images as graphs, where each node represents a region in the image. The edges encode the neighbourhood-relation of these regions. The task is now to match the regions in two given images. The authors of [1] give the constraint that regions that are matched should have a similar representation. At the same time the adjacent regions should also match well and regions should not cross each other, meaning that the regions should not be allowed to swap positions. These constraints should enforce spatial consistency.

Thus, in our MRF the unary potential measures how well two regions match, while the binary potential penalizes configurations that are spatially unlikely. The resulting energy of the optimized MRF can be used to construct a kernel. This kernel is then used to perform object categorization with a Support Vector Machine.

Before we describe the details of the MRF-kernel we first discuss how the image regions are represented in [1]. The features are based on the methods from [11]. They are created by concatenating four SIFT descriptors from overlapping regions. The regions do not have to correspond to the regions that are defined by our MRF, rather 32×32 pixel windows are used. The resulting vectors are then used to learn a sparse dictionary, such that each vector can be represented as a linear combination of the dictionary entries. The coefficients of these local sparse features are combined over a larger image region using

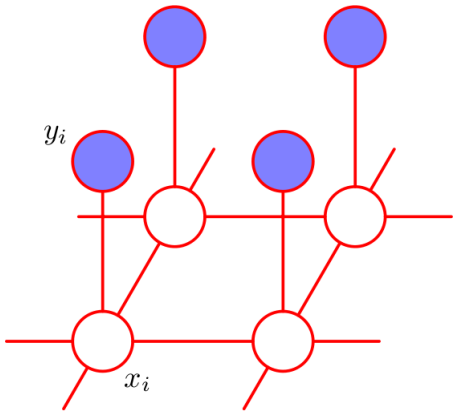


Figure 3.2: Example of an MRF. y_i is the observation at node (region) i. x_i is the displacement that we would like to predict. Image taken from [10].

max pooling, meaning that for each dimension the maximum is taken. The resulting vector is the final representation of an image region. The authors of [1] claim that these features are in generally better for object categorization than basic SIFT descriptors.

3.1 The MRF-Kernel

The optimization problem to be solved using an MRF is formulated in terms of a displacement d_n of a node n in the image graph $G = (V, E)$. This displacement describes the offset that has to be added to the position p_n of node n such that it matches with the position $p_{n'}$ of a node n' in the second graph G' ($p_{n'} = p_n + d_n$). The energy function of the MRF is then defined as:

$$E_{\rightarrow}(d) = \sum_{n \in V} U_n(d_n) + \sum_{(m,n) \in E} B_{m,n}(d_m, d_n)$$

U_n denotes the unary potentials and $B_{m,n}$ denotes the binary potentials. A maximum displacement in each direction of K is specified, leading to K^2 possible displacements for each node. Thus, the problem at hand can be seen as a multi-class labelling problem, where the labels correspond to the displacements.

The unary potential $U_n(d_n)$ is defined to be the correlation between the feature vectors F_n and F'_n of the current node n and the corresponding node n' in graph G', which is specified through the given displacement d_n.

The binary potential is the sum of two terms:

$$B_{m,n}(d_m, d_n) = u_{m,n}(d_m, d_n) + v_{m,n}(d_m, d_n) \tag{3.1}$$

$u_{m,n}(d_m, d_n)$ penalizes differences between displacements of adjacent nodes n and m. This difference between two displacements is measured by the L_1 norm and is weighted by a factor λ. This results in the following equation:

$$u_{m,n}(d_m, d_n) = -\lambda ||d_m - d_n||_1$$

The second term $v_{m,n}(d_m, d_n)$ in the binary potential is used to penalize crossing of image regions, since it is typically expected that displacements of image regions that result through shape variability vary smoothly for a given image. The penalty function that penalizes crossings is given by:

$$v_{m,n}(d_m, d_n) = \begin{cases} -\mu[dx_n - dx_m]_+ & \text{if } x_n = x_m + 1 \\ & \text{and } y_n = y_m \\ -\mu[dy_n - dy_m]_+ & \text{if } x_n = x_m \\ & \text{and } y_n = y_m + 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

dx_n and dy_n denote the displacement of n in x-direction and y-direction respectively. $[z]_+$ is a short form notation for $\max(0, z)$ and μ is a positive constant that acts as a weight for the penalty. As we can see

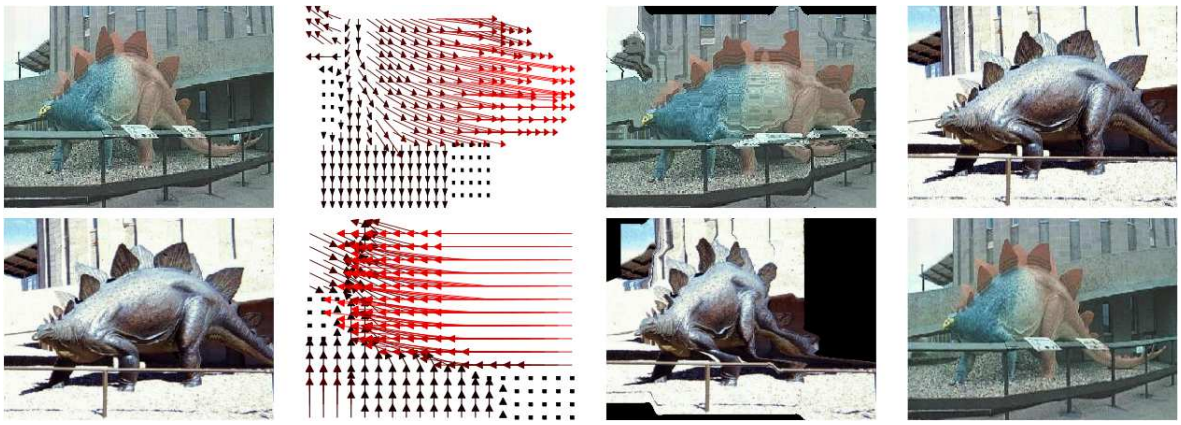


Figure 3.3: An example of the image matching performed by the MRF-kernel. The first column shows the image that should be matched to the image in the last column. The second column shows the displacements that the MRF-kernel estimates. The third column shows how the first image would look like when the displacement is applied. The matching is applied in both directions, since it is assymetric. Image taken from [1].

in Equation 3.2, $v_{n,m}$ is only non-zero, when two regions would cross either in the horizontal or in the vertical direction.

Now that every term in the energy function is defined, one could already perform categorization using the energy as similarity measure. Instead of doing this, the authors construct a kernel using the energy values, which can be used in any kernel-based classifier to do object categorization. The authors of [1] decide to use a SVM that was trained in a one-vs-all fashion, meaning that that for each class a SVM was trained that scores this class against all others. It should be noted that the energy function $E_{\rightarrow}(d)$ was defined for matching graph G to graph G'. However, the optimal matching from G' to G might be different. Thus, the energy function is computed in both directions, from G to G' and from G' to G and averaged to get a symmetric kernel. The reverse direction is denoted by $E_{\leftarrow}(d)$. The kernel is then defined as $\frac{1}{2}(\max_{d_1} E_{\rightarrow}(d_1) + \max_{d_2} E_{\leftarrow}(d_2))$. This kernel is not positive definite, which according to Mercer's theorem is actually the requirement for a well-defined kernel function. However, this function still gives reasonable results when one sets the negative eigenvalues of the kernel matrix to zero.

An example of a matching computed by the MRF-kernel is shown in Figure 3.3. However, before the matching can be applied the energy function of the MRF must be optimized. The next section will describe how this problem is tackled.

3.2 Solving the Optimization Problem

To compute the kernel that was constructed in the previous section we have to find the maximum energy of the MRF twice, for matching G to G' and in the reverse direction. But optimization of a multi-label MRF is NP hard in general [12]. Therefore, to work with large image databases we have to either constrain ourselves to very small graphs, look for approximate solutions or use properties of our specific MRF that might allow for an exact solution in polynomial time. In [1] Ishikawas method [13] is adopted. This algorithm is able to solve MRFs exactly if the binary potentials are a convex function of the differences between labels. This also means that there has to be a linear order in the set of possible labels. The authors of [1] adopt Ishikawas method to their problem and thus create a new algorithm that approximates the optimal energy for their MRF.

3.2.1 Ishikawas Method As already mentioned the method by Ishikawa [13] is able to exactly solve MRFs that are subject to some constraints in polynomial time. The first constraint is that there has to exist a linear order on the set of possible labels. This enables us to take the difference of two labels and ensures that this difference has a meaningful interpretation. The second constraint is that the binary potentials, also called the priors, are a convex function of the difference of two labels. The energy of an MRF with a given configuration X (that is, a specific mapping from nodes of the graph G to the set of

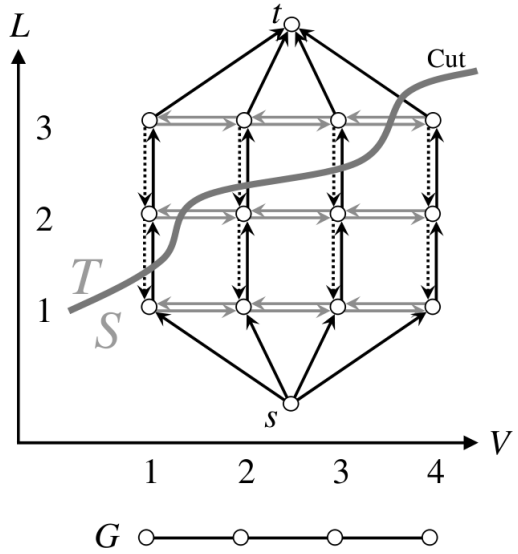


Figure 3.4: The flow network constructed in Ishikawas method. For every possible combination of nodes in G and a label a node is created. Data edges are represented by the black arrows in upwards direction. Constraint edges are depicted as the dotted arrows in downward direction. The horizontal gray arrows within a layer of nodes are the penalty edges which realize the prior. Image taken from [13].

labels) can be formulated as follows:

$$E(X) = \sum_{(u,v)\in E} \alpha_{uv} g(\iota(X_u) - \iota(X_v)) + \sum_{v\in V} h(v, X_v)$$

As usual V denotes the set of nodes in our graph G and E are the edges between these notes.The factor α_{uv} fullfils the conditions that $\alpha_{uv} \geq 0$ and $\alpha_{uv} = \alpha_{vu}$. ι is a function that maps a label to its index, e.g. $\iota(l_i) = i$. This is where the linear order on the set of labels is important.

It has already been mentioned that the function $g(x)$ has to be convex in order to make Ishikawas method work. If $g(x)$ would be a real function this would mean that for all $x, y \in \mathbb{R}$ and any $0 \leq a \leq 1$ the following inequality holds:

$$g(ax + (1-a)y) \leq ag(x) + (1-a)g(y)$$

However, in our case $g(x)$ is defined on the discrete set of labels and not on the real numbers. This leads Ishikawa to the following definition for convexity of a function on a discrete set A:

Definition. *A real-valued function $g(x)$ on a set A of real numbers is convex if*

$$g(ax + (1-a)y) \leq ag(x) + (1-a)g(y)$$

holds for any $x, y \in A$ and $0 \leq a \leq 1$ such that $ax + (1-a)y \in A$.

This definition of convexity implies that the second difference of $g(x)$ is nonnegative:

$$g(x+1) - 2g(x) + g(x-1) \geq 0$$

We will see that this fact will become important later.

Using these definitions, we can now construct a flow network from the nodes in our MRF and the set of labels. A cut in this flow network should then correspond to a configuration of the MRF and the value of the cut is the energy value for this configuration. Thus, we can find the minimum energy configuration by finding the minimum cut. The max-flow min-cut theorem states that we can find the minimum cut by computing the maximum flow. Several polynomial time algorithms are known to solve the max-flow problem, for example the Ford-Fulkerson algorithm.

Figure 3.4 shows the layout of the flow network. It has a node for every combination of a label and a node in our original graph. Additionally, a source node s and a sink node t is added. Ishikawa distinguishes between three types of edges, so-called data edges, constraint edges and penalty edges.

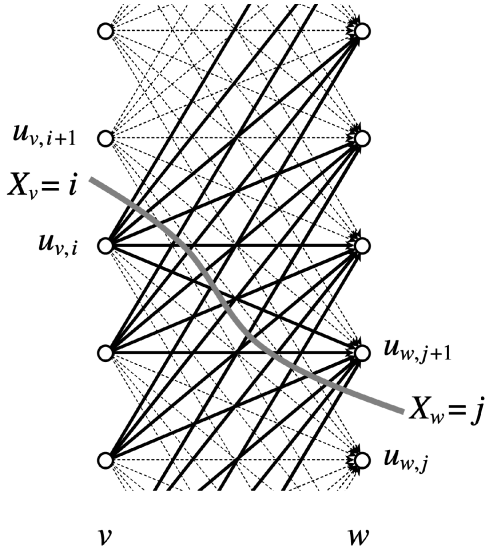


Figure 3.5: Depiction of general penalty edges. All edges shown are penalty edges going from the column over v to the column over w. The edges shown as dashed arrows are not in the cut, the solid edges are. Image taken from [13].

The data edges implement the term $h(v, X_v)$ of our MRF. They exist between nodes that represent the same node in our original graph G but differ in the associated label. To be more precise, the edge goes from the node $u_{v,l}$ to $u_{v,l+1}$, where v is the corresponding node in G and l is the associated label. Additionally, the source node s connects to the earliest layer of nodes, that is we have an edge $(s, u_{v,1}$ in the network for all $v \in G$. The capacity of these edges is infinite, thus preventing them to become part of the cut. The capacity of the edges between the higher layers realize the term $h(v, X_v)$:

$$c(u_{v,i}, u_{v,i+1}) = h(v, i), i = 1, \ldots, k-1, c(u_{v,k}, t) = h(v, k)$$

The second type of edges are the constraint edges. They exist to make sure that the minimum cut defines a unique mapping from nodes to labels. They prevent that a column of edges is cut more than once, as seen in Figure 3.4. This is achieved by having the edges going from a higher level to the next lower level and having an infinite capacity:

$$c(u_{v,i+1}, u_{v,i}) = \infty, i = 1, \ldots, k-1$$

The last kind of edges are the penalty edges, which realize the prior. Ishikawa shows that the capacity of these edges must be set to the second difference of the prior $g(x)$, divided by two:

$$c(u_{v,i}, u_{w,j}) = \frac{g(i-j+1) - 2g(i-j) + g(i-j-1)}{2}$$

The second difference of $g(x)$ will always be non-negative if $g(x)$ is a convex function, which explains this restriction to the prior. Note that in Figure 3.4 there are only penalty edges between nodes, which are within a layer and which are also in neighbouring columns. But of course a penalty edge can exist between any two nodes in different columns. An example of more general penalty edges is given in Figure 3.5.

Finally, we have to define how a cut in the flow network corresponds to a configuration of the MRF. For all nodes $v \in G$ we assign the label i to v if the edge $(u_{x,i}, u_{v,i+1})$ is in the cut. This can be seen in Figure 3.5. Here the node v would get the label $X_v = i$ and the node w would get the label $X_w = j$.

3.2.2 Curve Expansion Ishikawas method is not directly applicable to the MRF that was constructed in [1]. This is due to the fact that the labels in this case correspond to displacements in the two-dimensional plane and we cannot put a linear order on $\mathbb{N}^2$ while keeping our binary potential convex. Thus, the authors of [1] propose a method called curve expansion, which builds on Ishikawas method.

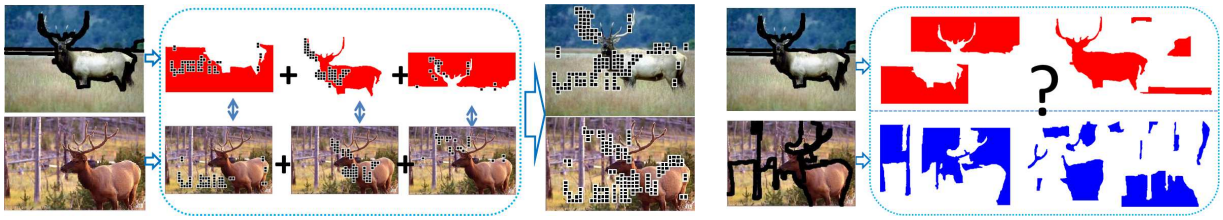


Figure 3.6: Comparison between region-to-image matching (left) and region-to-region matching (right). The region-to-image matching method is able to find correspondences between the two input images (the black squares). Possible correspondence points are extracted from segmented regions in the first image and matched to points anywhere in the second image. In contrast, the region-to-region matching method is unable to find correspondences between the regions that were extracted from *both* images, since they are very dissimilar. Image taken from [2].

The binary potentials defined in Equation 3.1 can be decomposed as:

$$B(d) = \sum_{(m,n)\in E} g_x(dx_m - dx_n) + g_y(dy_m - dy_n)$$

g_x and g_y are convex function as defined by Ishikawa and they now only depend on the displacement in the x-dimension or the y-dimension respectively. By keeping $d_y = (dy_1, \ldots, dy_N)$ fixed one can now find the optimal displacement $d_x = (dx_1, \ldots, dx_N)$ using Ishikawas method. Alternatively, by fixing d_x one can find the optimal displacement dy in y-direction. Alternating between those two steps gives an algorithm which is able to find a good MRF configuration:

$$dy^{t+1} \leftarrow \operatorname*{argmax}_{dy} \sum_{n\in V} U_n(dx_n^t, dy_n) + \sum_{(m,n)\in E} g_y(dy_m - dy_n)$$

$$dx^{t+1} \leftarrow \operatorname*{argmax}_{dx} \sum_{n\in V} U_n(dx_n, dy_n^t) + \sum_{(m,n)\in E} g_x(dx_m - dx_n)$$

The method is initialized by computing:

$$\max_{dx} \sum_{n\in V} \max_{dy_n} U_n(dx_n, dy_n) + \sum_{(m,n)\in E} g_x(dx_m - dx_n)$$

This can clearly be solved optimally by applying Ishikawas method. While Ishikawas method was guaranteed to find the global optimum of the energy function, the proposed method of curve expansion only finds local optima. However, the authors of [1] claim that the energy of the resulting configuration is lower than the energy of at least $2 \cdot (\sqrt{N_l})^{N_n}$ other configurations. In this case N_l corresponds to the number of labels and N_n denotes the number of nodes. In contrast, the alpha expansion method only guarantees a lower bound of $N_l 2^{N_n}$ [14]. Thus, for a large number of labels curve expansion should be better.

Before we analyse the performance of the MRF-Kernel for object categorization, another method is introduced which is also based on matching image regions to compute similarities. In the section on experiments we will then compare the MRF-Kernel to this method.

4 Region-to-Image Matching

In contrast to [1], the authors of [2] do not use region-to-region matching. Instead, given a pair of images, one of them is divided into regions which are then matched to descriptors in the other image. The descriptors in the second image do not have to correspond to continuous regions. Therefore, the authors call their approach a region-to-image matching method. This approach has the advantage that a geometrically consistent match can be found anywhere in the second image. This should lead to a greater robustness against variations in shape and pose that may appear within an object category.

The difference between region-to-image matching and region-to-region matching is further illustrated in Figure 3.6. In this example the region-to-image matching is obviously better since it is not inhibited by dissimilar segmentations. We now turn to a more detailed explanation of the approach that was proposed in [2].

4.1 General description

The method starts by dividing one of the two input images into regions. But instead of a regular grid as in [1], the regions are extracted through a bottom-up segmentation. A dense grid is put over these regions. SIFT descriptors are sampled from the grid points. Each region is represented by a string of these SIFT descriptors. The individual descriptors in the strings are from now on called points. To get a better representation of the 2D layout of a region actually two strings are created. The first string is the column-wise string, which is constructed by starting at the top-left point of a region and then adding all points in this column. The points of the column to the right are then added from the bottom to the top and so on. The second string is the row-wise string, which is constructed in a similar fashion, but using the rows instead of the columns. By representing regions as one-dimensional strings rather than two-dimensional grids we can make use of efficient matching algorithms.

We now want to match the regions to the unsegmented image. For this, candidate points for each point in one of the strings are found. The candidate points are selected by computing the distance between the SIFT descriptors in the unsegmented image and the current points. This happens across multiple scales, to enable scale invariant matching. Given the candidates for each point in a string, one can compute the optimal correspondence according to some cost function. The cost function has to take appearance and also spatial consistency into account. The minimization itself is done via dynamic programming.

4.2 The Dynamic Programming Solution

To define the cost function that will be minimized some notation has to be introduces. The string for the i-th region is defined by the set of points $P_i = \{p_1, \ldots, p_{l_i}\}$. p_k are the image coordinates for the k-th point. Point p_k and its successor p_{k+1} are neighbours in the string and are thus also spatially related. The set of candidate matches in the unsegmented image for a point p_k is denoted by C_k.

Given a set of candidate points $[C_1, \ldots, C_{l_i}]$ for each point in a string, we want to find the matching $M^* = \{m_1, \ldots, m_{l_i}\}$ with $m_k \in C_k$, such that the cost function $\mathcal{C}(P, M)$ is minimized. $\mathcal{C}(P, M)$ is contains four parts, which will now be described in detail:

$$\begin{aligned}\mathcal{C}(P, M) = \\ \sum_{k=1}^{l_i-1} w_g G(p_k, p_{k+1}, m_k, m_{k+1}) + \sum_{k=1}^{l_i} w_a A(p_k, m_k) \\ + \sum_{k=1}^{l_i-1} w_o O(p_k, p_{k+1}, m_k, m_{k+1}) + \sum_{k=1}^{l_i} w_d D(p_k, m_k)\end{aligned} \tag{3.3}$$

As we can see, two of the four terms depend on neighbouring points p_k and p_{k+1} and their corresponding matches m_k and m_{k+1}. The other two terms, $A(\cdot)$ and $D(\cdot)$, only depend on the current point p_k and its match m_k. This resembles the optimization problem for MRFs, where also a disctinction between unary and binary terms was made. The main difference is that the MRF optimization problem was two-dimensional rather than one-dimensional. Additionally, each of the four terms has a corresponding weight (w_g, w_a, w_o and w_d). These weights scale the importance of the individual terms.

The first term, $G(\cdot)$, is called the geometric distortion term and penalizes neighbouring match pairs that have a dissimilar distortion:

$$G(p_k, p_{k+1}, m_k, m_{k+1}) = ||(p_k - p_{k+1}) - (m_k - m_{k+1})||_2$$

The other pairwise term $O(\cdot)$ is called the ordering constraint term and is also used to enforce spatial consistency. This is achieved by penalizing different orders between neighbouring points and their corresponding matches. So the value of $O(\cdot)$ is 1, if the ordering of p_k and p_{k+1} is different from the ordering of m_k and m_{k+1}. However, if the order is consistent the value of $O(\cdot)$ is simply 0.

The first unary term, $A(\cdot)$, measures the similarity between a point and the potential match and is called the appearance similarity:

$$A(p_k, m_k) = \frac{1}{1 + \exp\left(-\tau_a \left(\frac{1}{\mu_a}||f_{p_k} - f_{m_k}||_2 - 1\right)\right)}$$

f_{p_k} and f_{m_k} are the SIFT descriptors at point p_k and its match m_k. It should be noted that the distance $||f_{p_k} - f_{m_k}||_2$ is computed across all scales. Then the minimum is selected to compute the appearance

similarity. μ_a and τ_a are positive constants that are used to adjust the shape of the sigmoid function in such a way that the resulting costs are in a similar range as the other terms in $\mathcal{C}(P, M)$.

The last term is the displacement constraint $D(p_k, m_k)$. It encodes the assumption that objects appear in similar scene layouts and that this should mean that the location at which a possible match for a point is found should not be too far away from the original point. This can be expressed by the following formula:

$$D(p_k, m_k) = \begin{cases} ||p_k - m_k||_2, & \text{if } ||p_k - m_k||_2 > t \\ 0, & \text{otherwise} \end{cases}$$

Here t is the threshold of the maximal displacement that we want to allow without penalty.

What is not yet considered is the fact that not every point in the strings might have a corresponding match in the unsegmented image. To deal with this, a null match is added to each candidate set C_k. If a null match is selected in one of the terms of Equation 3.3 then its value is set to a constant χ. The value of χ should be selected such that it is larger than the typical costs of a reliable match.

Equation 3.3 is optimized over all regions P_i of the segmented image and both their corresponding strings. Each optimization gives a set of correspondences between the image. The union over all these sets is then used to compute a final score between the two given images.

For the images I_1 and I_2, the scoring function $S(I_1, I_2)$ should give high values when the two images are similar and low values when they are dissimilar. It should measure not only the appearance similarity between two matched points, but also the geometric deformation of all points in the union of correspondences. $S(I_1, I_2)$ is defined as follows:

$$S(I_1, I_2) = \frac{1}{\sqrt{N_1 N_2}} \sum_{i=1}^{n} \omega_i s_a(p_i, m_i) s_g(p_i, m_i)$$

N_1 and N_2 are the number of points in the images I_1 and I_2. ω_i is a weight term that measures the impact of the i-th matching pair on the final score. It will be defined in more detail below. $s_a(\cdot)$ and $s_g(\cdot)$ measure the appearance similarity and geometric similarity. In particular $s_a(p_i, m_i)$ is simply defined as $s_a(p_i, m_i) = 1 - A(p_i, m_i)$, where $A(p_i, m_i)$ is the appearance similarity already defined above. On the other hand, $s_g(p_i, m_i)$ measures the spatial consistency between the current matching pair (p_i, m_i) and all other matching pairs:

$$s_g(p_i, m_i) = \frac{1}{n-1} \sum_{k} \frac{1}{1 + \exp\left(\tau_g \left(\frac{G(p_i, p_k, m_i, m_k)}{\alpha_{ik}}\right) - 1\right)}$$

$G(\cdot)$ is the geometric distortion term defined above. k runs over all matching pairs except the current one, that is $k \in \{1, \ldots, n\} \backslash i$. τ_a is a constant that is used to adjust the shape of the sigmoid function. α_{ik} is a weight that gives more importance to matching pairs that are in the same region. If p_i and p_k are in the same region then α_{ik} is set to α, else the value is set to 2α, where α is a positive constant. By using α_{ik} like this we enforce a more similar layout within regions, while matching pairs in different regions are allowed to have more different geometric distortions.

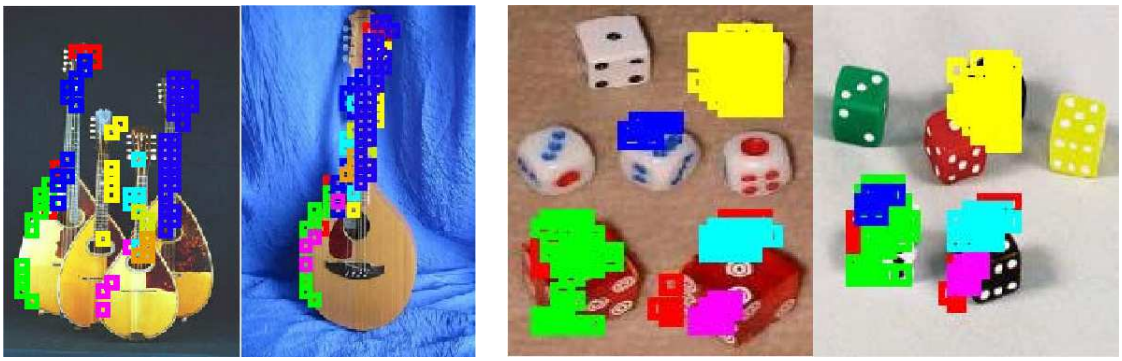
The last term we have to define is the weight ω_i that measures the importance of a matching pair. The definition of ω_i follows the assumption that points which have a low number of initial matches are more discriminative and should thus get a larger weight. The number of matches for point p_i is given by $|C_i|$, leading to the following equation:

$$\omega_i = \frac{1}{1 + \exp\left(\tau_\omega \left(\frac{|C_i|/N_2}{\mu_\omega} - 1\right)\right)}$$

The number of matches $|C_i|$ is normalized by the total number of points N_2 in the unsegmented image. τ_ω and μ_ω are again constants that adjust the shape of the sigmoid function.

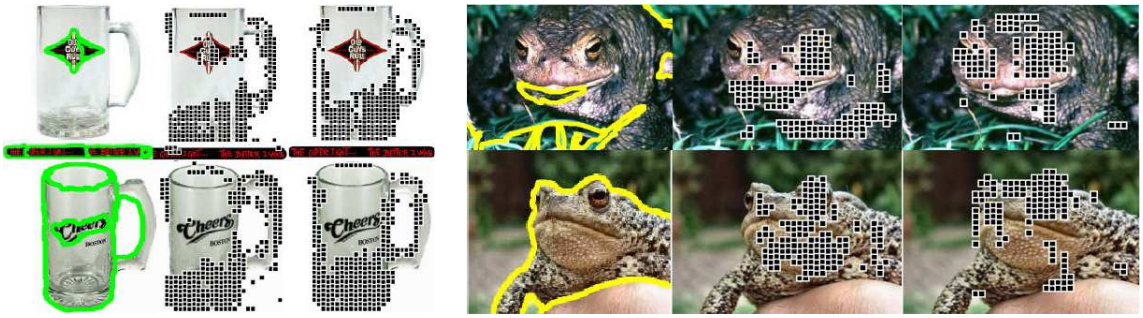
Figure 3.7 gives an example of the advantages of the method proposed in [2]. The method finds correspondence between images that are consistent region-wise but do not constrain the geometric relations between regions too much. The authors also claim that their method is robust to the choice of which input image is segmented and which is left unsegmented. An example for this is shown in Figure 3.8.

The score $S(I_1, I_2)$ can now be used for object categorization in a nearest-neighbour framework. We will now turn to the experiments that the authors of both [1] and [2] performed to evaluate the effectiveness of their methods. Since they used the same databases for evaluation we can directly compare their results.



(a) (b)

Figure 3.7: Example of matches between images of the same category. The left column shows the segmented image, the right column shows the unsegmented image. Correspondences are indicated by points in the same color. **(a)** The method can match regions taken from multiple instance of the same object to a single instance of the same object. **(b)** Multiple instances of the same category can also be matched. Note that the layout of points in a region is consistent, but that the layout over the regions is less constrained. Image taken from [2].



(a) (b)

Figure 3.8: The method is robust to the choice of which image is segmented. The first column shows the segmentation for both images. The second image shows the matching when the segmentation of the first image is used. The third column shows the matching when the segmentation of the second image is used. Matches are indicated as black dots. Image taken from [2].

5 Experiments and Comparison

We will first consider the experiments performed with the MRF-kernel method from [1], then the region-to-image matching method before we finally compare both.

The experiments to evaluate the classification performance are conducted on two different database, the Caltech 101 database and the Caltech 256 database. The Caltech 101 dataset consists of more than 9000 images, depicting objects from 101 different categories. Since the amount of training images per category can vary wildly, one typically takes a fixed amount of random images per category and trains on those. This procedure is repeated several times and then the average of the results is computed. The class imbalance problem can also be seen in Table 1. The statistics are taken from [15]. The Caltech 256 database is the successor to the Caltech 101 database. With around 30000 images it is larger and the number of object categories was raised to 256. In both datasets the image size is not uniform, however in [2] it is stated that 320×240 is a typical image size.

5.1 Experiments with the MRF-kernel Method

5.1.1 Runtime The experiment is conducted on 100 images of the Caltech 101 database and compares the performance of alpha expansion, Tree-Reweighted Message Passing (TRW-S) [16] and two variants of curve expansion. The first variant is the one described in the chapter on the MRF-kernel. The second

	Caltech 101	Caltech 256
Number of images	9146	30607
Number of classes	101	256
Images per class (min)	31	80
Images per class (max)	800	827
Images per class (avg)	90	109

Table 1: Statistics for the Caltech 101 and the Caltech 256 databases

Method	15 Examples	30 Examples
MRF-Kernel	**75.3**	80.3
NBNN [7]	65.0	-
NBNN [7]	72.8	-
Subcategory learning [8]	72.0	83.0
GSMKL [9]	73.3	**84.3**

Table 2: Average recognition rate on the Caltech 101 database. Methods in the upper half use only a single type of features, methods in the lower half use multiple types of features. Top performances are shown in bold.

variant is a generalization which also allows diagonal moves and is therefore computationally more expensive. This generalization is called multi-step curve expansion, while the faster variant is referred to as 2-step curve expansion.

The authors report that 2-step and multi-step curve expansion are much faster than alpha expansion and TRW-S for graphs with up to 1000 nodes. For graphs with more nodes (around 4000 and more) alpha expansion is faster. Furthermore, the authors also analysed the energy values that are found by the different optimization algorithms. Multi-step curve expansion and TRW-S find energies that are around 2% and 5% lower respectively, than those found by 2-step curve expansion and alpha expansion, which have similar minimization performance. The lower energies found by TRW-S and multi-step curve expansion are not expected to improve the categorization performance by much. Thus, the authors decide to use 2-step curve expansion for further experiments, because it is able to match two images with 500 nodes in 0.04 seconds or less.

5.1.2 Classification Performance The authors of [1] selected a fixed parameter setting for all their experiments. The constants λ and μ from the MRF energy function are set to $\lambda = 0.1$ and $\mu = 0.5$. These values were chosen through visual inspection of a matching. The maximal displacement is set to $K = 11$. The grid size which determines the image regions is chosen as 18×24.

On the Caltech 101 database two experiments are conducted, with a varying size of the training set (15 or 30 images per category). The final result is achieved by taking the average performance over 20 random splits. The results are listed in Table 2, together with a comparison with state-of-the-art methods. We can see that the MRF-kernel method is the best method on the smaller training set size of 15 images per category. However, with more training data the methods presented in [8] and [9] perform better by 3% to 4% absolute. It should be noted that those methods use multiple feature representations. Figure 3.9 shows examples from the Caltech 101 database and the warping field that is estimated by the MRF-Kernel. We can see that the warping helps to compensate for different shapes and poses of objects of the same category.

Method	30 Examples
MRF-Kernel	38.1
NBNN [7]	37.0
NBNN [7]	42.0
Subcategory learning [8]	**49.5**

Table 3: Average recognition rate on the Caltech 256 database. Methods in the upper half use only a single type of features, methods in the lower half use multiple types of features. Top performances are shown in bold.

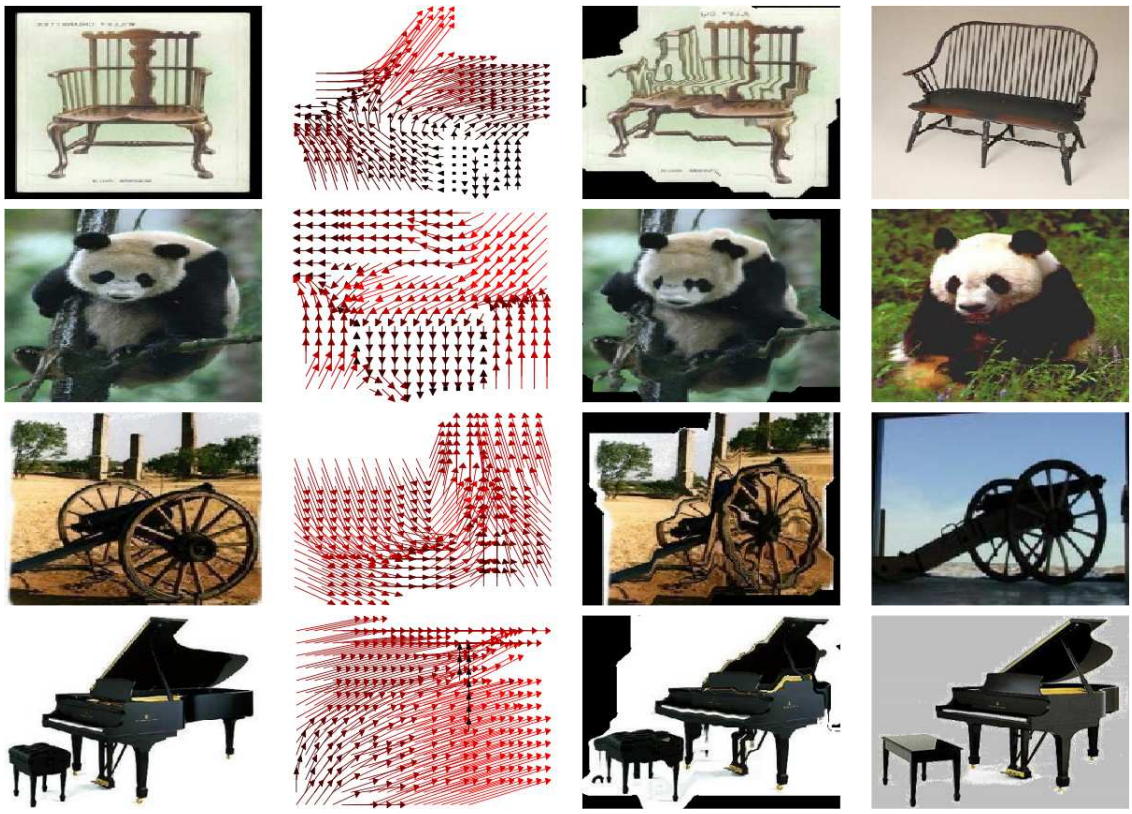


Figure 3.9: Examples of image matching using the MRF-kernel on the Caltech 101 database. The first and the last column show the images that are to be matched. The second column shows the displacements between image regions that are estimated by the MRF-kernel. The third column shows how the first image would look like when this warping is applied. Image taken from [1].

The experiments on the Caltech 256 dataset are conducted with 30 training images per object category. The results are shown in Table 3. The method by Todovoric et. al. [8] is again the best performer. It is interesting to see that the MRF-kernel beats the Naive Bayes Nearest Neighbour approach [7] when only a single type of features is used. But when multiple feature representations are used, the NBNN method performs better. This leads to the conclusion that multiple features are indeed very helpful and it is possible that the MRF-kernel would also perform better with additional features.

5.2 Experiments with the Region-to-Image Matching Method

We now turn to the experiments used to evaluate the region-to-image matching method. The experiments are again conducted on the Caltech 101 and Caltech 256 databases. For both datasets SIFT descriptors are densely sampled (every 8 pixels) across four different scales. This leads to about 1200 points per scale for the typical images in the databases. The method that is used for segmentation for one of the images that are to be matched is described in [17]. Initial candidate matchings are found by approximate nearest neighbour search. All SIFT descriptors with a distance that is within $1.25\mu_a$ are considered to be candidates. The weight parameters in the objective function $\mathcal{C}(P, M)$ (see Equation 3.3) are set by visual inspection of resulting matchings of images within the same object category. The final weights are $w_d = 4.0$, $w_o = 1.5$, $w_a = 1.25$ and $w_g = 1.0$.

The nearest-neighbour approach that is used for recognition avoids matching each image to the query image by a pruning step. The pruning is based on distances between SIFT descriptors and reduces the pool of considered classes. For Caltech 101 only the 25 classes with lowest distance are used. For Caltech 256 the number of considered classes is raised to 30. The algorithm chooses the class that reaches the highest sum of matching scores for the best-scoring k images per class. k is chosen to be 2.

5.2.1 Runtime The authors report that with the above described dense sampling it takes about 5 to 15 seconds to match two images. However, this assumes that the images in the database are already

Method	15 Examples
Region-to-image matching	61.3
NBNN [7]	65.0
NBNN [7]	72.8
Subcategory learning [8]	72.0
GSMKL [9]	**73.3**

Table 4: Average recognition rate on the Caltech 101 database. Methods in the upper half use only a single type of features, methods in the lower half use multiple types of features. Top performances are shown in bold.

Method	30 Examples
Region-to-image matching	36.3
NBNN [7]	37.0
NBNN [7]	42.0
Subcategory learning [8]	**49.5**

Table 5: Average recognition rate on the Caltech 256 database. Methods in the upper half use only a single type of features, methods in the lower half use multiple types of features. Top performances are shown in bold.

segmented and that the query image is used as the unsegmented one. The segmentation algorithm that is used [17] requires about 3 to 5 minutes per image, making it a computationally very expensive step. The authors argue that a parallelized version of the algorithm could bring the cost of the segmentation down to about 2 seconds.

5.2.2 Classification Performance The results for the classification experiments on the Caltech 101 dataset are shown in Table 4. [2] cites a wide variety of methods which are surpassed by the new region-to-image matching approach. However, even in the category of methods that are not based on multiple feature types the new method is not the top performer, but is beaten by the NBNN approach. As expected, the performance gap only gets wider when multiple feature representations are considered. The authors argue that the NBNN method only computes an image-to-class distance and does not consider image-to-image distances. The region-to-image matching however does, and this can be an advantage. The found correspondences can for example be used for detection purposes.

The results for the Caltech 256 database, listed in Table 5, are similar to those on the Caltech 101 database. The main differences are that the recognition rates are much lower, since the Caltech 256 dataset is in general more difficult. Also the NBNN approach is no longer better than the method from [8]. The region-to-image matching is still outperformed by both approaches. Figure 3.10 shows some example images from the Caltech 256 dataset and the nearest neighbours found by the region-to-image matching method. The method can find clearly find nearest neighbours that do match well to the query image, even if there is some variability in pose and shape. The shown errors in the last two rows are also quite intuitive. In the last-but-one row we can see that the query image is a building with a tower and that its nearest neighbour is also a tower, namely the Eiffel tower. In the last row we can see that the query image, depicting a giraffe, has several animals as its nearest neighbours.

5.3 Comparison

Finally, we will compare the two main methods of this seminar report, the MRF-kernel and the region-to-image matching. The results for the methods were already listed and discussed above, but are now shown together in Table 6 and Table 7. For the Caltech 101 dataset the MRF-kernel clearly beats the region-to-image matching method by 14% accuracy absolute. It is also better than the other state-of-the-art methods, when only a training set size of 15 images per category is considered.

On the Caltech 256 database the performance gap between the MRF-kernel and the region-to-image matching gets much smaller, the difference is now only 1.8% absolute. The MRF-kernel is still the best method of those that use only one kind of feature, but the multi-feature methods perform better.

The MRF-kernel has in general better recognition results than the region-to-image matching method. In terms of runtime the authors of [1] claim that they can match two images in 0.04 seconds using their

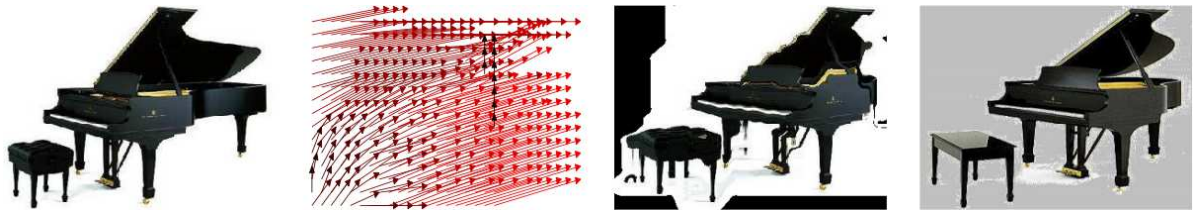


Figure 3.10: Matching results on the Caltech 256 database using region-to-image matching. The first column is the query image, the columns two to seven are the nearest neighbours, ordered by distance. The first seven rows show correct matches for the nearest neighbour. The last two rows are wrong matches, but the errors are very intuitive. Image taken from [2].

Method	15 Examples
MRF-kernel	**75.3**
Region-to-image matching	61.3
NBNN [7]	65.0
NBNN [7]	72.8
Subcategory learning [8]	72.0
GSMKL [9]	73.3

Table 6: Comparison of recognition rates of the MRF-kernel and region-to-image matching on the Caltech 101 database. Other state-of-the-art methods are also listed. Methods in the upper half use only a single type of features, methods in the lower half use multiple types of features. Top performances are shown in bold.

variant of Ishikawas method. The region-to-image matching needs 5 to 15 seconds to match two images, under the assumption that a segmentation is already given. Since the segmentation would only raise the computational time that is needed, it seems that the MRF-kernel is also the more efficient of the two algorithms.

6 Conclusion

In this seminar report we have presented two methods for object categorization that try to incorporate spatial constraints on the input images to improve recognition performance. The first method, the MRF-kernel, is based on region-to-region matching. The regions are matched by optimizing an MRF. The energy values of this optimized MRF are then used to construct a kernel which can be used for SVM classification. The second method allows to match from regions to a whole image. The problem is formulated as a string matching problem, which can be solved using dynamic programming. Both presented methods improve on the state-of-the-art or are at least close. In a direct comparison we find that the MRF-kernel performs better in both accuracy and runtime.

However, in both papers there are points of criticism. The first point of criticism applies to both papers. The authors seem to recognize that multiple types of features perform in generally better than a single feature type. Since both methods perform worse than state-of-the-art methods that use multiple feature representations it would have been reasonable to adapt the algorithms to use multiple features

Method	30 Examples
MRF-kernel	38.1
Region-to-image matching	36.3
NBNN [7]	37.0
NBNN [7]	42.0
Subcategory learning [8]	**49.5**

Table 7: Comparison of recognition rates of the MRF-kernel and region-to-image matching on the Caltech 256 database. Other state-of-the-art methods are also listed. Methods in the upper half use only a single type of features, methods in the lower half use multiple types of features. Top performances are shown in bold.

as well. In both papers the features were sampled from a dense grid, so sampling multiple features from the gridpoints would have been a simple procedure. Especially for the much more difficult Caltech 256 database the usage of more than one feature type seems to help.

For the MRF-kernel it would have been nice if the authors gave a reason why their method does not scale as well as other methods, when the size of the training data is increased. This can be seen in the results for the Caltech 101 dataset, where the gain in performance is about 5% absolute when the size of the training data is doubled. The other two competing state-of-the-art methods gain more than 10% absolute, which is a significant difference.

The authors of [1] also propose both the 2-step curve expansion and multi-step curve expansion. They analyse both variants for their speed when optimizing an MRF. But they only give recognition results for the faster 2-step curve expansion. However, the computing time for multi-step curve expansion should not be prohibitive for recognition experiments, since only graphs with a size of 500 nodes are considered. Graphs of this size clearly fall into the range where multi-step curve expansion is still tractable, as shown in the runtime comparison in [1].

A criticism that applies to the region-to-image matching paper is that some parameters were chosen through visual inspection. It could be the case that performance would be much improved if the parameters were tuned on a validation set. The parameters are also fixed for experiments on different databases, even though different databases might require different parameter setting. This also applies to the MRF-kernel paper, though there are less parameters in this approach.

Another possibel shortcoming of both methods is that the authors assume that images within a category should have a similar layout. In the MRF-kernel paper this assumption is represented in the choice of the parameter K, the maximum displacement. In the region-to-image matching approach the authors penalize dissimilar layouts in their objective function through the term $D(\cdot)$. It is not clear if the assumption always holds and thus experiments should be conducted to verify this hypothesis.

Despite these issues both papers offer interesting approaches to the problem of object categorization. The design choices for the algorithms are overall very reasonable and the experimental results support this. Thus, the two approaches build a good basis for future work in the area of object categorization with spatial constraints.

We have also seen that both image matching and object categorization have applications in the field of medical image processing. Image matching on its own can be used for registration of medical images. Object categorization methods are applicable for automatic annotation, for example in a image retrieval context. Thus, one can say that the presented methods are also relevant for medical image processing, though adaptions might be necessary.

References

[1] Duchenne O, Joulin A, Ponce J. A Graph-matching Kernel for Object Categorization. In: Proceedings of the International Conference in Computer Vision (ICCV); 2011. p. 1056.

[2] Kim J, Grauman K. Asymmetric region-to-image matching for comparing images with generic object categories. In: CVPR. IEEE; 2010. p. 2344–2351.

[3] Keysers D, Gollan C, Ney H. Local Context in Non-linear Deformation Models for Handwritten Character Recognition. In: International Conference on Pattern Recognition. vol. 2. Cambridge, UK; 2004. p. 511–514.

[4] Deselaers T, Weyand T, Keysers D, Macherey W, Ney H. FIRE in ImageCLEF 2005: Combining Content-based Image Retrieval with Textual Information Retrieval. In: Accessing Multilingual Information Repositories, 6th Workshop of the Cross-Language Evaluation Forum, CLEF 2005. vol. 4022 of Lecture Notes in Computer Science. Vienna, Austria; 2006. p. 652–661.

[5] Maintz J, Viergever M. A survey of medical image registration. Medical Image Analysis. 1998;2(1):1–36.

[6] Loeffelbein DJ, Souvatzoglou M, Wankerl V, Martinez-Moeller A, Dinges J, Schwaiger M, et al. PET-MRI Fusion in Head-and-Neck Oncology: Current Status and Implications for Hybrid PET/MRI. Journal of Oral and Maxillofacial Surgery. 2012;70(2):473 – 483.

[7] Boiman O, Shechtman E, Irani M. In defense of Nearest-Neighbor based image classification. In: CVPR; 2008. .

[8] Todorovic S, Ahuja N. Learning subcategory relevances for category recognition. In: CVPR. IEEE Computer Society; 2008. .

[9] Yang J, Li Y, Tian Y, Duan L, Gao W. Group-sensitive multiple kernel learning for object categorization. In: ICCV; 2009. p. 436–443.

[10] Bishop CM. Pattern Recognition and Machine Learning (Information Science and Statistics). Secaucus, NJ, USA: Springer-Verlag New York, Inc.; 2006.

[11] Boureau YL, Bach F, LeCun Y, Ponce J. Learning mid-level features for recognition. In: CVPR; 2010. p. 2559–2566.

[12] Shimony SE. Finding MAPs for belief networks is NP-hard. Artificial Intelligence. 1994 Aug;68(2):399–410.

[13] Ishikawa H. Exact optimization for markov random fields with convex priors. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2003;25:1333–1336.

[14] Boykov Y, Veksler O, Zabih R. Fast Approximate Energy Minimization via Graph Cuts. IEEE Transactions on Pattern Analysis and Machince Intelligence. 2001 Nov;23(11):1222–1239.

[15] Griffin G, Holub A, Perona P. Caltech-256 Object Category Dataset. California Institute of Technology; 2007. CNS-TR-2007-001.

[16] Kolmogorov V. Convergent Tree-Reweighted Message Passing for Energy Minimization. IEEE Transactions on Pattern Analysis and Machince Intelligence. 2006 Oct;28(10):1568–1583.

[17] Arbelaez P, Maire M, Fowlkes CC, Malik J. From contours to regions: An empirical evaluation. In: CVPR; 2009. p. 2294–2301.

Features and Flow-Correspondence in Video

Seminar Medical Image Processing

Igor Dudschenko

Abstract

Every hospital produces and processes a huge collection of data. A big part of this data is produced by CT scans, CT angiography scans or, depending on the hospital, video data of robot tracking in operations and human movement for rehabilitation purposes. Radiologists have a very tedious and time consuming task, to analyse this huge amounts of data. Methods like SIFT (Scale Invariant Feature Transform) or HOG (Histogram of Oriented Gradients) can reduce the time to analyse the collection of data. In order to analyse video or image data, a detection of key points in the data has to be done. The next step is to describe the found key points. This description of key points is called the Descriptor. The Descriptor is needed to match data with a set of database informations in order to recognize specific objects or optical flow in image and video data. The results are good indication points for radiologists. These methods reduce time consumption by accentuating key points, objects or movement in video or image data.

Keywords: SIFT, Computer Vision, Image Alignment, Object Recognition

1 Introduction

Feature and Flow-Correspondence in Video or Images is a challenging problem in computer vision. This work describes the state of the art approaches to distinguish feature (description of a key point) and flow-correspondences. It is a difficult problem to find and analyse reliable image features. The usual approach to detect key points, which are part of the image features, is to search for global or local extrema in images. The standard approach of detecting extrema is to compute the derivations and checking for extrema properties, in this work I will discuss another approach that is called the DOG (Differences of Gaussian) method, which computes the extrema by differentiation of Gaussians, this approach reduces the time consumption greatly. Therefore I will discuss the DOG method and the properties of a Gaussian briefly. The result contains a lot of edges of objects, which include ambiguous edge responses. The most common way to delete ambiguous edge responses is to add thresholds, I am going to introduce how to determine the computation of these thresholds in order to keep the unambiguous edge responses in the results. Other problems like scale changes, rotation, illumination and 3D camera viewpoint changes are still challenging tasks in computer vision. In my work I will introduce a method which is invariant to image scaling, rotation and partially invariant to illumination and 3D camera viewpoint. The process of detecting image features will be discussed in the Detector section.
The next stage of computation is to describe a key point, which is introduced in the Descriptor section. The Descriptor computes a set of informations about the features, which allows matching with databases. The common approach of describing a key point is to compute gradients and orientation of the surroundings of each key point. The key point descriptors have to be highly distinctive, in order to be able to find its correct matches in a large database of features.
The last stage of computation is to match the computed features with a database of features using SVM (Support Vector Machines). This work won't go into the details of SVMs. The usual approach is to provide positive and negative information about features of an object in order to distinguish if an object is found in the data or not.
Many different applications are using these approaches in order to provide information about image data. One example of an application in medical image processing of the introduced method could be detection of coronary arteries in CTA, which is currently researched by Sergei Fotin (Cornell University). It is a very time consuming work for radiologists to skip through each cardiac image of an CTA in order to analyse coronary arteries. The methods that I am going to introduce in this work are capable of detecting

coronary arteries and analyse them. After analysing the image data, accentuating of coronary arteries themselves and threatening properties of coronary arteries can be accentuated to provide a good overview of the whole image data close to real-time.

2 Detector

The first stage of computing Feature and Flow-Correspondences is to detect key points, which is provided by the Detector. The Detector has to provide as many key points as possible in order to distinguish objects correctly in images or videos. However it is also important to reduce ambiguous or irrelevant responses. The Detector also has to provide invariance of scale and rotation changes, which allows robust description of key points. Several different approaches were introduced for this stage of computation. In this section we will discuss about the different Detectors and their functionalities.

2.1 SIFT Detector

SIFT was developed by David G. Lowe in 1994. SIFT is famous for its robustness and is a state of the art method to detect feature correspondence in images and videos. SIFT uses a scale invariant Detector and a DOG (Differences of Gaussian) method to detect and localize key points. The first step of computation is to compute a Scale-Space Extrema Detection and localize key points in the different scaling of the input image.

2.1.1 Scale-Space Extrema Detection The first step of SIFT is the Scale-space extrema detection, Lowe used a cascade filtering approach with efficient algorithms to identify candidate locations that are used in the following steps. In order to search for stable features, Lowe used the Extrema of the Gaussian function, because of its fast computation time and its useful property of an infinite amount of derivations. The scale space of an image is defined as the function:

$$\mathcal{L}(x, y, \sigma) = \mathcal{G}(x, y, \sigma) * \mathcal{I}(x, y) \tag{4.1}$$

Obviously $\mathcal{L}(x, y, \sigma)$ is the smoothed image of the original image. It is a part of the octaves in figure 1. Its visualized as yellow squares in the figure. $\mathcal{G}(x, y, \sigma)$ is the Gaussian function and the input image is $\mathcal{I}(x, y)$, where x and y indicate the position of the actually smoothed pixel, also * is the convolution operation in x and y, and

$$\mathcal{G}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{4.2}$$

In this Gaussian parameter changes of σ will affect the smoothness of the Gaussian. The next step of scale-space extrema detection is to calculate a difference-of-Gaussian function:

$$\mathcal{D}(x, y, \sigma) = \mathcal{L}(x, y, k\sigma) - \mathcal{L}(x, y, \sigma) \tag{4.3}$$

$\mathcal{L}$ has to be computed in any case for scale space feature description, that is the reason why Lowe has chosen the difference-of-Gaussian, because with this function, only a simple image subtraction is needed to compute $\mathcal{D}$, which is very efficient in terms of time complexity. The input image is smoothed several times by the Gaussian function which gives us the amount of images in the octaves. In each octave the input image has been scaled differently, Lowe uses this method to allow scale invariance by iterating through scales. The next step is to repeatedly convolve the octaves with Gaussians to produce a set of scale space images. Adjacent Gaussian images are subtracted, in this case adjacent means the differences of the Gaussian smoothing. The Gaussian image is down-sampled by a factor of 2 in every step, and the process repeated. To calculate the extrema the usual approach would be now to calculate the derivations first and analyse key points for extrema properties. Figure 1 shows the explained process briefly. In this approach we are using images which have been smoothed by a Gaussian, this means to calculate the extrema we only need to calculate the differences of the differently smoothed images. With the heat diffusion equation (4,5) Lowe showed that this approach gives a good approximation.

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \tag{4.4}$$

$$\implies G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G \tag{4.5}$$

With the down sampling in each octave the location of possible key points is of course changed, the next section describes how to locate key points through different scales. To detect the local maxima and minima of the difference-of-Gaussian images, Lowe compared the 26 nearest neighbours in 3x3 regions in their current and adjacent scale of each pixel. Figure 2 describes this process.

After detecting key point candidates the next step is to perform a detailed fit to the nearby data for location, scale and ratio of principal curvatures, the next section will describe this process.

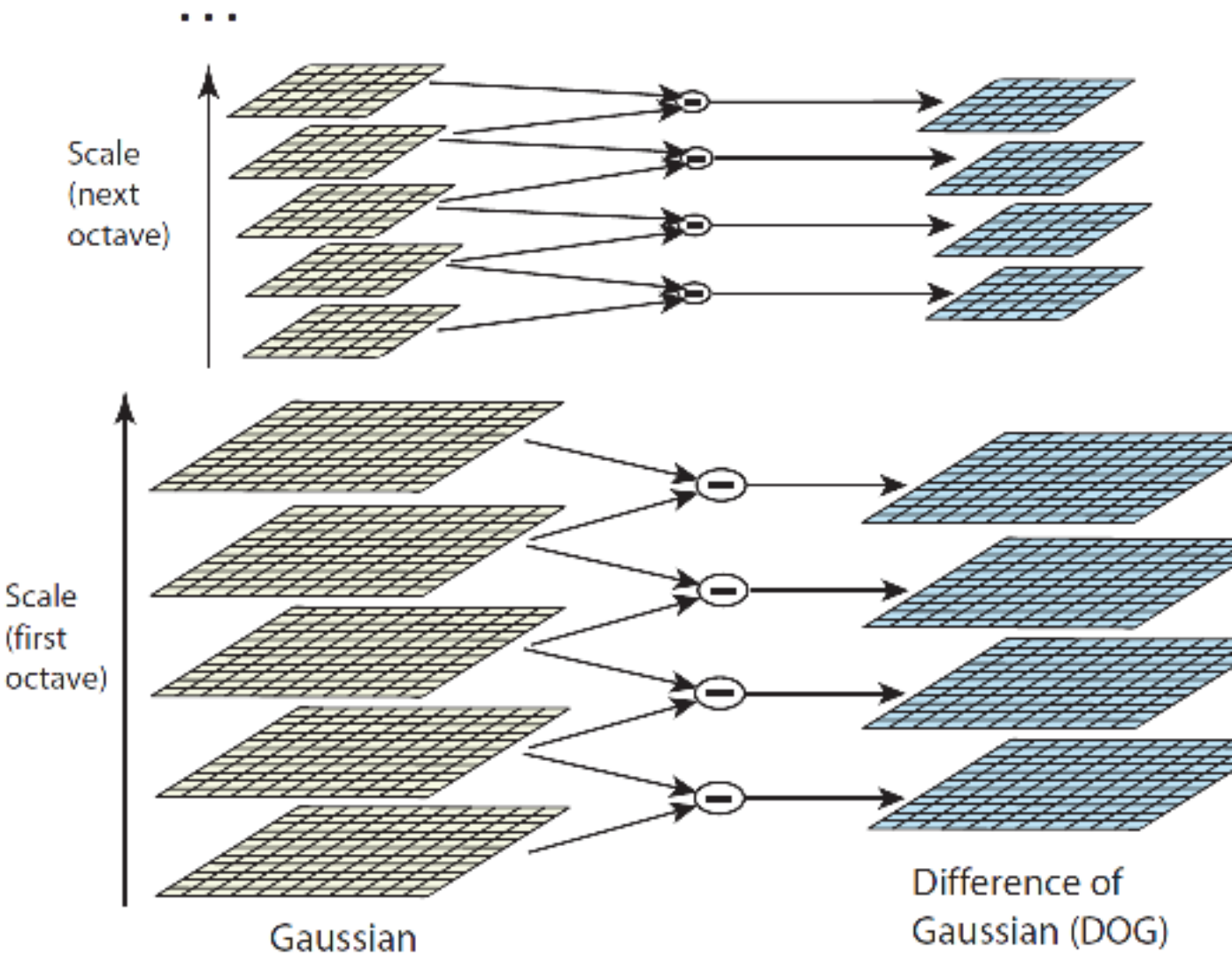


Figure 4.1: The first two steps of the scale-space extrema detection. Starting with down-sampled and Gaussian smoothed input image. The octaves consist in this example of four smoothed images and the original image. The differences indicating in the figure result in the extrema for this image. The next octaves are down sampled by the factor 2, in order to be scale invariant.

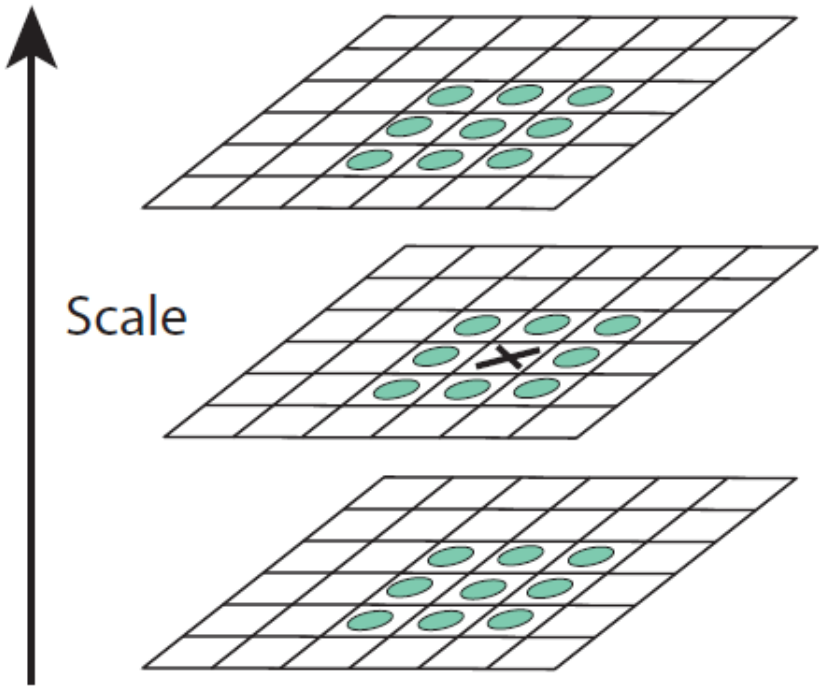


Figure 4.2: The X marks the pixel which is actually checked for extrema behaviour. The adjacent scales are visualised as circles. If the value of X is the highest or lowest, in comparison of the circles, than X will be the local extrema.

2.1.2 Key point localization and thresholds For this step I will take the Brown and Lowe Method, instead of the initial implementation of Lowe, which had matching and stability problems [1]. Brown and Lowe used a 3D quadratic function to the local sample points. This way they determined the interpolated location of the extrema. They used the Taylor expansion of the introduced scale-space function $\mathcal{D}(x, y, \sigma)$. They also needed to shift the scale-space function so that the origin is at the sample point:

$$\mathcal{D}(o) = \mathcal{D} + \frac{\partial \mathcal{D}^{\mathcal{T}}}{\partial o} o + \frac{1}{2} o^{\mathcal{T}} \frac{\partial^2 \mathcal{D}}{\partial o^2} o \tag{4.6}$$

where $\mathcal{D}$ and its derivatives are evaluated at a current sample point with the offset-vector $o = (x, y, \sigma)^{\mathcal{T}}$To locate the extrema, the usual approach by determine the derivative of this function with setting x to zero is used, which leads to:

$$\widehat{x} = -\frac{\partial^2 \mathcal{D}^{-1}}{\partial o^2} \frac{\partial \mathcal{D}}{\partial o} \tag{4.7}$$

Still unstable extrema with low contrast could be chosen, if the offset $\widehat{x}$ is larger than 0.5 in any dimension. This value indicates that the extrema lies closer to a different sample point, therefore the sample point is changed and the interpolation performed instead about that point. Now the original offset $\widehat{x}$ has to be added to the location, in order to get the interpolated estimate for the location of the extrema, which leads to:

$$\mathcal{D}(\widehat{x}) = \mathcal{D} + \frac{1}{2} \frac{\partial D^T}{\partial o} \widehat{x} \implies |\widehat{D}(\widehat{x})| < threshold \tag{4.8}$$

Based on experienced data from low-resolution scenery images, Lowe used the threshold 0.03. This value depends on the content of the data and should always be tested and changed for data with different contents than sceneries. However the difference-of-Gaussian still contains ambiguous edge responses. Lowe used in his paper a threshold on ratio of principal curvatures to detect ambiguous edge responses. The principal curvatures can be computed from a 2x2 Hessian matrix. Equation 9 is showing the Hessian

matrix which is using the derivations of $\mathcal{D}(o)$, as shown in equation 6.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix} \tag{4.9}$$

Instead of computing the eigenvalues of the Hessian matrix, Lowe used the approach from Harris and Stephens (1988) [1]. In this approach the computation of eigenvalues it not necessarily needed, since Lowe is only interested in the ratio of the eigenvalues. Therefore let α be the eigenvalue with the highest magnitude and β be the eigenvalue with the lowest magnitude. To compute the sum of eigenvalues, Lowe used the trace of the Hessian matrix and their product was computed from the determinant, as shown in equation 10 and 11.

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \tag{4.10}$$

$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta \tag{4.11}$$

For the next equation let r be the ratio between α and β, so that $\alpha = r\beta$.

$$\frac{Tr(H)^2}{Det(H)} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r+1)^2}{r} \tag{4.12}$$

Equation 12 is at minimum when two eigenvalues are equal and it increases with r. Therefore, a threshold was introduced by Lowe, to check the ratio of principal curvatures. This is a very efficient way to compute [1]. Again the used threshold is based on experienced data. Lowe used a threshold of $r = 10$, which eliminates key points that have a ratio between the principal of curvatures greater than 10. Therefore equation 13 shows the threshold computation, also the transition from figure 3 (c) to (d) shows the effects of this operation.

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{4.13}$$

Figure 3 is showing the process, as an example the key points of a house were computed.

2.2 Harris Detector

Schmid et al. (2000) demonstrated an excellent and reliable performance of the Harris detector [5]. However this detector is not invariant to scale changes. Therefore Mikolajczyk, Schmid et al. (2002) started to work on a Harris detector with automatic scale selection to obtain a scale invariant detector [5]. This detector is based on a second moment matrix, which is also called the auto-correlation matrix, in order to adapt scale changes to make it independent of the image resolution:

$$\mu(x, \sigma_I, \sigma_D) = \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{bmatrix} = \sigma_{\mathcal{D}}^2 g(\sigma_I) * \begin{bmatrix} \mathcal{L}_x^2(x, \sigma_{\mathcal{D}}) & \mathcal{L}_x\mathcal{L}_y(x, \sigma_{\mathcal{D}}) \\ \mathcal{L}_x\mathcal{L}_y(x, \sigma_{\mathcal{D}}) & \mathcal{L}_y^2(x, \sigma_{\mathcal{D}}) \end{bmatrix} \tag{4.14}$$

where σ_I is the integration scale, $\sigma_{\mathcal{D}}$ is the differentiation scale and $\mathcal{L}_a$ is the derivative computed in the a direction, using Gaussian smoothing. Equation 14 is describing the gradient distribution in a local neighborhood of a point. The derivatives are computed with Gaussian kernels of the size determined by the local scale σ_D and than averaged in the neighborhood of a point by smoothing with a Gaussian with an integration scale σ_I. The eigenvalues of the matrix shown in equation 14 lead to principal signal changes in a neighborhood of a point, this property can be used for key point detection. The detected key points are stable in arbitary lightning conditions and are representative of an image, because the curvatures of the signal changes are significant in orthogonal directions i.e. corners and junctions etc. The Harris measure combines the trace and the determinant of the second moment matrix, where local maxima of the cornerness-function determines the location of interest points [5] :

$$cornerness = det(\mu()x, \sigma_I, \sigma_{\mathcal{D}})) - \alpha * trace^2(\mu(x, \sigma_I, \sigma_{\mathcal{D}})) \tag{4.15}$$

2.3 Harris-Laplace Detector

In order to keep the efficiency and reliability of the Harris-Detector, but still being able to detect scale-invariant keypoints, Mikolajczyk and Schmid combined the Harris-Detector with the Automatic Scale Selection that has been extensively studied by Lindeberg(1998). Lindenberg used low-resolution images of sceneries. Figure 5 illustrates an example of the Automatic Scale Selection. The idea behind it is to select

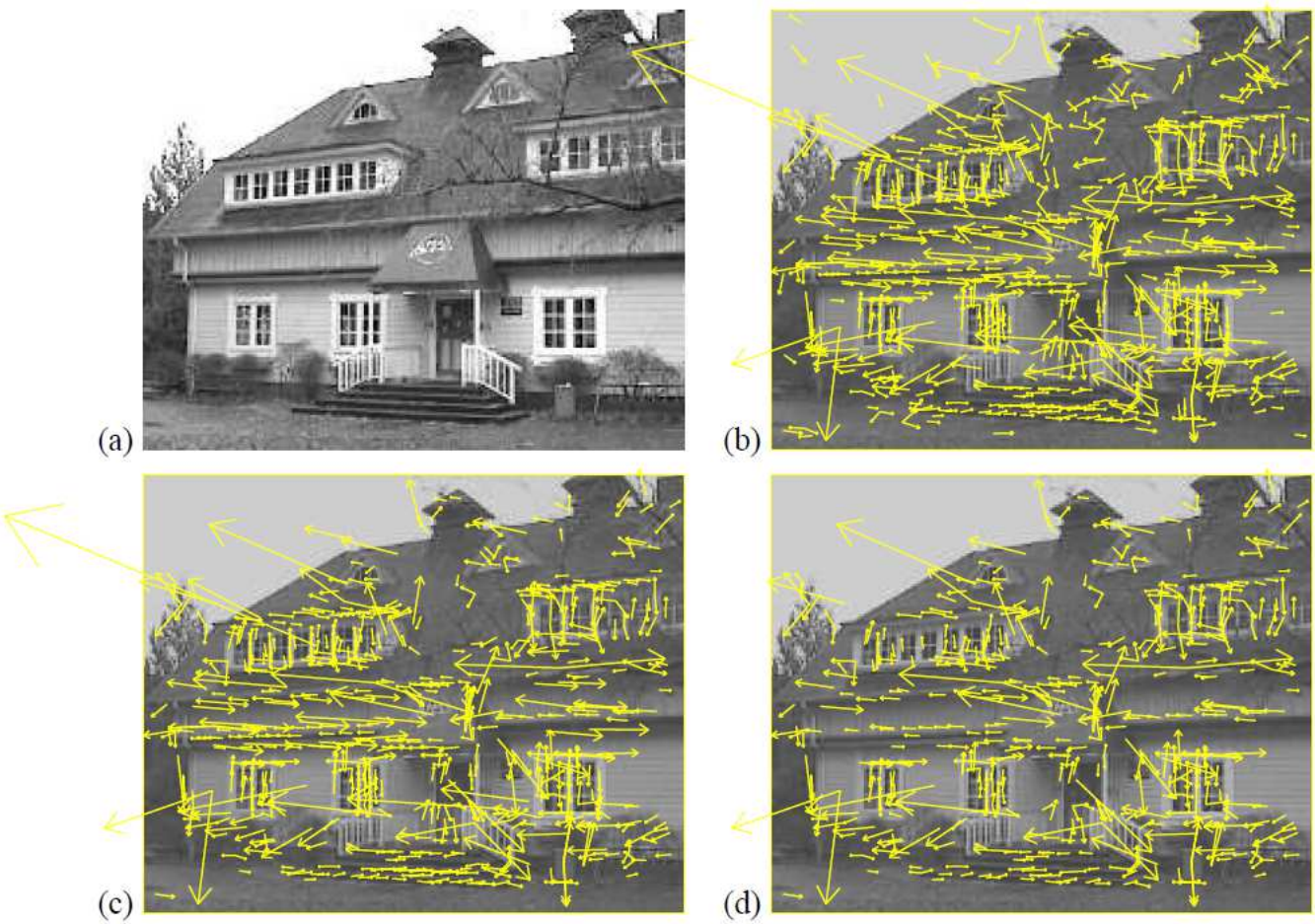


Figure 4.3: (a) Shows the 233x189 pixel original image. (b) 832 keypoints locations, indicating with yellow arrows scale, orientation and location. For further explanation of how scale and orientation are computed, read section Orientation Assignment. (c) Remaining 729 keypoints after applying a minimum contrast. With reference to equation 8. (d) The final 536 key points, after eliminating ambiguous edge responses.

the characteristic scale of a local structure, for which a given function attains an extrema over scales [5]. Mikolajczyk and Schmid experimented with that knowledge and found out that the Laplacian-of-Gaussians finds the highest percentage of correct characteristics scales. Equation 16 shows the Laplacian-of-Gaussian, refering to the key point x in the scale σ_n.

$$\mid LoG(x, \sigma_n) \mid = \sigma_n^2 \mid \mathcal{L}_{xx}(x, \sigma_n) + \mathcal{L}_{yy}(x, \sigma_n) \mid \tag{4.16}$$

The algorithm of the Harris-Laplace Detector can be divided into two parts: 1) a multi-scale point detection and 2) an iterative selection of the scale and the location. Interest points get extracted at each level of the representation. Each time local extrema have to be detected in the 8-neighborhood of a possible key point. These extrema over scale of the LoG are than used to select the scale of interest points. For a possible key point x, in the scale space k, with scale σ_I, the algorithm works as followed [5]:
1. Find the local extrema over scale of the LoG for the point, otherwise the actual point is not a good possible key point and can be rejected.
2. Detect the spatial location $x^{(k+1)}$ of a maximum of the Harris measure nearest to $x^{(k)}$ for the selected σ_I^{k+1}
3.Go to Step 1 if $\sigma_I^{k+1} \neq \sigma_I^k$ or $x^{k+1} \neq x^k$

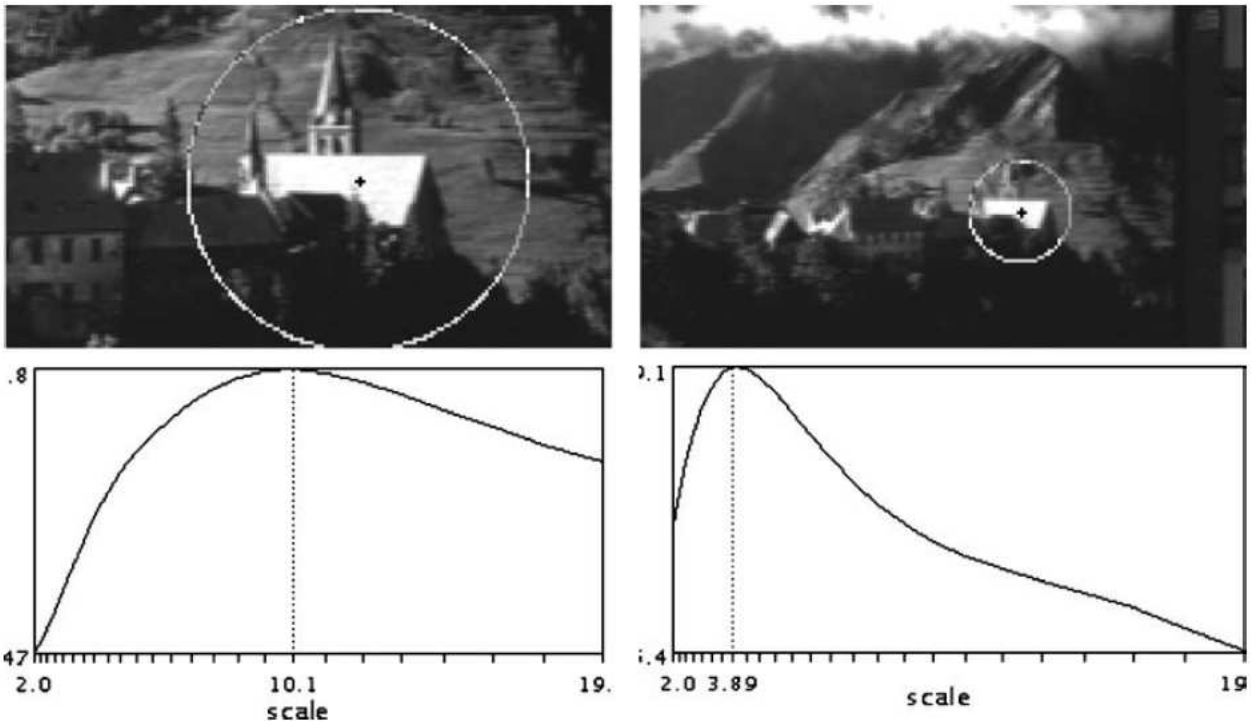


Figure 4.4: Automatic Scale Location for two different pictures with the same object at a different scale. The top row is illustrating the used images and the bottom row is illustrating the normalized LoG (cf. Eq.(16)) The characteristics scales are 10.1 for the left image and 3.89 for the right image.

3 Descriptors

After detecting key points in images or video will be no other information except the location of the key points, which in the most cases is not enough to distinguish objects in images or video. Therefore some kind of description is needed for every key point detected by the Detector. This description will be used to check a trained SVM database for matches with objects or optical flow. The Descriptors use high dimensional vectors, e.g. SIFT-Descriptor 128-dimensional vector, to describe a key point. The most popular Descriptors use the same approach to describe key points, they compute image gradients (orientation and magnitude of the key point), break the image region into spatial bins and histogram the gradient magnitudes according to their orientations and location. Therefore I will introduce the computation of orientation assignment and than introduce the most popular Descriptors.

3.1 Orientation assignment

After some experimentations Lowe used the following approach for orientation assignment. In order to be scale-invariant Lowe used the Gaussian smoothed image, $\mathcal{L}$ (cf. Eq. (1)) with the closest scale. For each differently smoothed image the gradient magnitude, $m(x, y)$ and the orientation $\theta(x, y)$ is precomputed using the Equations 17 and 18. $L(x, y, \sigma)$ represents $L(x, y)$ in the following Equations, due to readability.

$$m(x,y) = \sqrt{(\mathcal{L}(x+1,y) - \mathcal{L}(x-1,y))^2 + (\mathcal{L}(x,y+1) - \mathcal{L}(x,y-1))^2} \tag{4.17}$$

$$\theta(x,y) = tan^{-1}\frac{\mathcal{L}(x,y+1) - \mathcal{L}(x,y-1)}{\mathcal{L}(x+1,y) - \mathcal{L}(x-1,y)} \tag{4.18}$$

An orientation histogram is formed, using the gradient orientations, $\theta(x, y)$, of sample points within a 3x3 region around the keypoint (Figure 2 illustrates the region in detail). The computed orientation histogram contains 36 bins, which are covering a 360 degree range of orientations. Each sample point is weighted by its gradient magnitude $m(x, y)$ and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint [1]. Weighting the gradient magnitudes by a Gaussian-weighted circular window is not necessarily needed, but it leads to better matches, since information about closer neighbors are more reliable.

3.2 SIFT Descriptor

A descriptor is computed for every key point. The descriptor will be used to compare the features of images. The SIFT Descriptor is a local Descriptor. The Descriptor computes a vector containing 16 regions with 8 histograms per region, in total it is a vector containing 128 Elements. Figure 4 shows an example of a descriptor Lowe used. To compute the Descriptor we need the previous described steps to compute gradient magnitude and orientation at each detected point of the image. The next step is to compute a Gaussian window, which is weighting the gradient magnitude and orientation. These samples are accumulated into orientation histograms, Lowe used eight different angles for his histogram, summarizing the 4x4 subregions by summing up the weighted gradient magnitudes near that direction within the region[1].

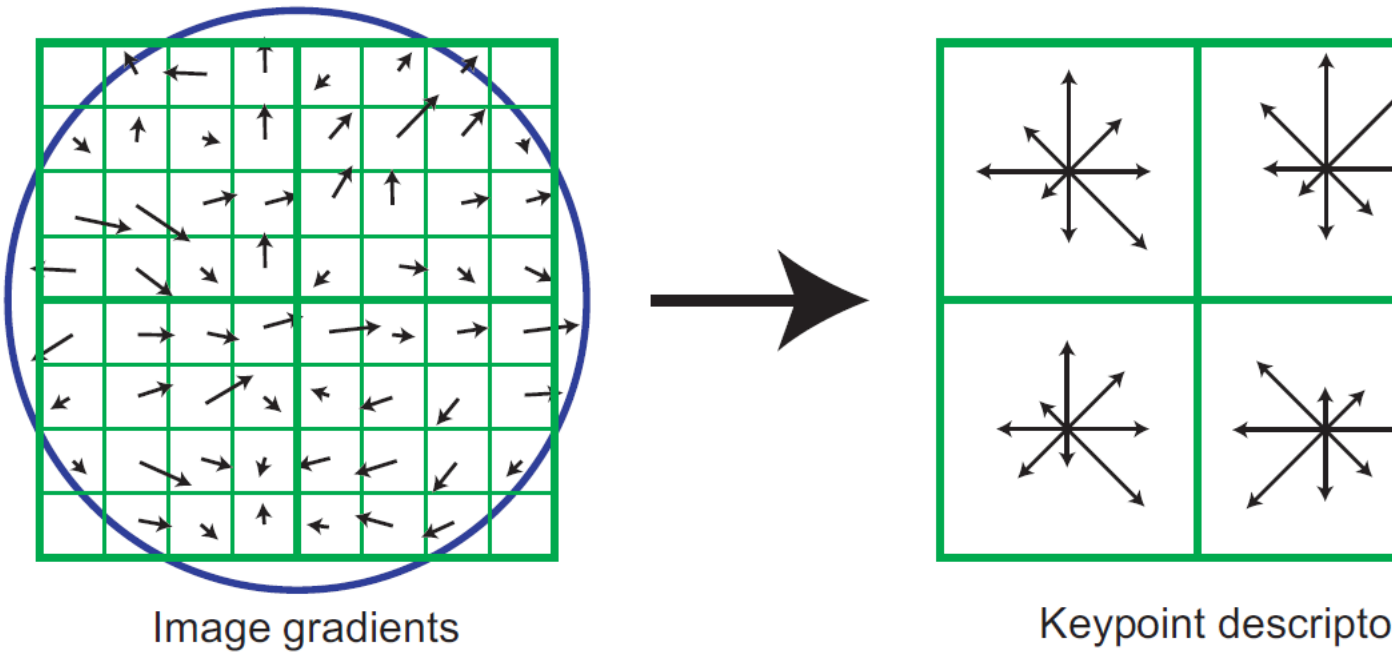


Figure 4.5: The left image is showing a 16x16 sample array, containing the gradient magnitudes and orientations at each image sample point in that region. The blue circle is showing the Gaussian window. The right image shows the summarized orientation histograms.

3.3 HOG Descriptor

The HOG Descriptor is a global Descriptor. Still HOG works similar to SIFT. It is using the same approach of calculating orientation and gradient magnitude. However HOG is not using a Detector, which is why HOG is a global Descriptor. Therefore HOG is not scale invariant, because the iteration through scales is also done by the descriptor, which makes SIFT invariant to scale. The similarities become better visible as soon as you look on the computed histograms, which would be exactly the same for SIFT, if the bins of the angles and the location of the key points are the same. Figure 6 is illustrating the result of a HOG Descriptor, using an example image with a 64x128 resolution. The first step is to create 2x2 cells subregions, which contain 8x8 pixels. HOG is always using the nearest neighbors of each point, which means that it has an overlay in every computed subregion. The next step of computation is now to sum up the subregions and, as SIFT does, quantize the gradient magnitudes and orientations into different angle bins. Which leads us to a vector containing 7x15(=105) subregions containing 2x2(=4) cells and 9 different angle bins. Summarized the HOG Descriptor is a vector containing 105x4x9 = 3780 elements for a low resolution image.

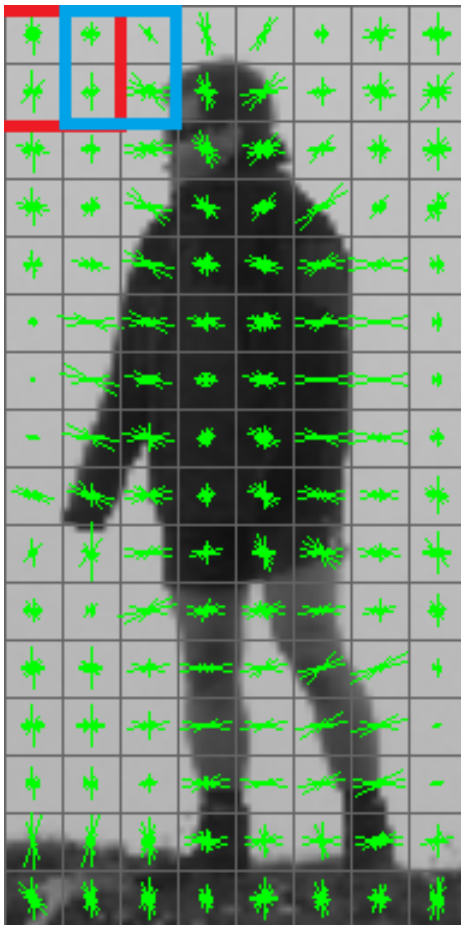


Figure 4.6: The image has a resolution of 64x128 pixels. The red and blue squares are indicating the subregions and their overlays. The green arrows are illustrating the gradient magnitude and orientation.

3.4 3D SIFT by Scovanner

For the 3D SIFT Descriptor a new Orientation Assignment has to be introduced, in order to recognize the third dimension. In this approach the third dimension will be time. As described in the previous orientation assignment section, two functions have to be computed, the gradient magnitude and the orientations in 3D we need two angles now instead of one, the second angle will be dependent by time and location change of the key point. Let $\mathcal{L}_x$ be the scale space function represented by $L_x = L(x+1, y, t) - L(x-1, y, t)$, $\mathcal{L}_y$

represented by $\mathcal{L}_y = L(x, y+1, t) - L(x, y-1, t)$ and $\mathcal{L}_t$ represented by $\mathcal{L}_t = L(x, y, t+1) - L(x, y, t-1)$.

$$m_{3D}(x, y, t) = \sqrt{\mathcal{L}_x^2 + \mathcal{L}_y^2 + \mathcal{L}_t^2} \tag{4.19}$$

$$\theta(x, y, t) = tan^{-1}(\frac{\mathcal{L}_y}{\mathcal{L}_x}) \tag{4.20}$$

$$\phi(x, y, t) = tan^{-1}(\frac{\mathcal{L}_t}{\sqrt{\mathcal{L}_x^2 + \mathcal{L}_y^2}}) \tag{4.21}$$

Figure 7 is illustrating 3D SIFT in comparison to 2D SIFT.
In the approach of Scovanner et al. the weighted histograms around a interestpoint of a 3D neighborhood

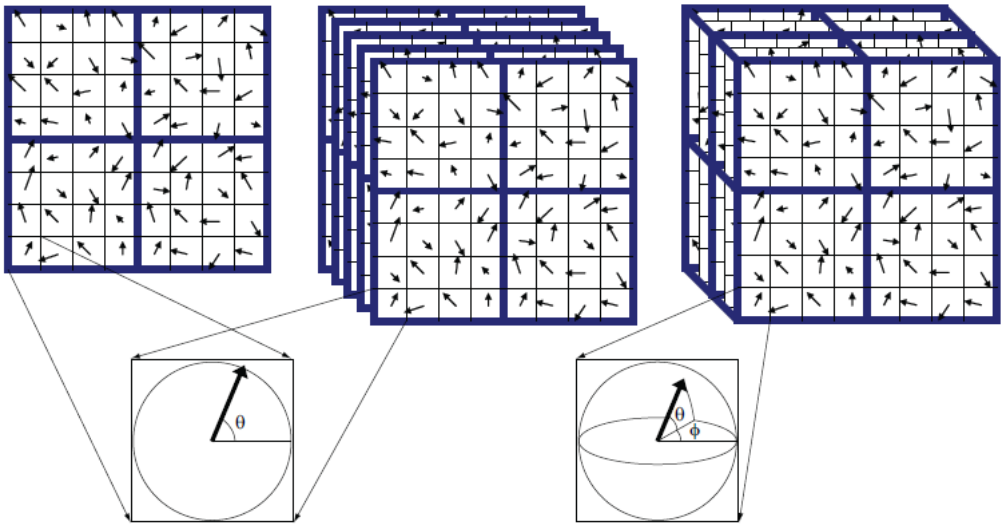


Figure 4.7: Left: The ordinary 2D SIFT image with its descriptors. Middle: 2D SIFT apllied to a movie. Right: 3D SIFT with its sub-histograms and the added angle.

for a given interest point, were computed by dividing θ and ϕ into equally sized bins and creating a 2D histogram out of it. By dividing θ and ϕ this approach is using meridians and parallels in its 2D histograms. This method is simpler and faster in finding peaks of a 2D orientation histogram. However the bins still have to be normalized by their solid angle (ω). An example for required normalization is a 2D map of the earth, it has to be either stretched in areas near the poles, or create discontinuities. Incorrect weighting of histograms would lead to the cause if one were to skip this step of computation. Equation 22 is showing the calculation of the solid angle.

$$\omega = \int_{\phi}^{\phi+\Delta\phi} \int_{\theta}^{\theta+\Delta\theta} sin\theta d\theta\, d\phi = \Delta\phi \int_{\theta}^{\theta+\Delta\theta} sin\theta\, d\theta = \Delta\phi[-cos\theta]_{\theta}^{\theta+\Delta\theta} = \Delta\phi(cos\theta - cos(\theta+\Delta\theta)) \tag{4.22}$$

The value that is added to the histogram (cf. Eq.(23)), where (x, y, t) represents the location of the interest point, and (x', y', t') represents the location of the new interest point which has to be added to the histogram. Depending on the peaks of the histogram, the dominant peak is stored as it can be used to rotate the neighborhood around the keypoint, which leads us to the final histogram function, where i_θ represents the 2D histograms with θ as the second angle and i_ϕ with ϕ as the second angle:

$$hist(i_\theta, i_\phi) = \frac{1}{\omega} m_{3D}(x', y', t') e^{\frac{-((x-x')^2+(y-y')^2+(t-t')^2)}{2\sigma^2}}$$

Now only the Descriptor Representation has to be added. First the sub-histograms have to be computed. Scovanner et al. rotated the 3D neighbourhood so that the dominant orientation points in the direction of $\theta = \phi = 0$. A simple rotation-matrix can be used to do this rotation, as shown in the following matrix.

$$\begin{bmatrix} \cos\theta\cos\phi & -\sin\theta & -\cos\theta\sin\phi \\ \sin\theta\cos\phi & \cos\theta & -\sin\theta\sin\phi \\ \sin\phi & 0 & \cos\phi \end{bmatrix} \tag{4.23}$$

The final descriptor will be a vectorization of the sub-histograms. Figure 8 is illustrating a visualization of the abbreviated Descriptor used to view the feature vector [7]. Each of the 8x4 sub-plots represent an orientation bin. Each gray sub-plot represents a reshaped sub-histogram (4x2) value of the 2x2x2 sub-histogram.

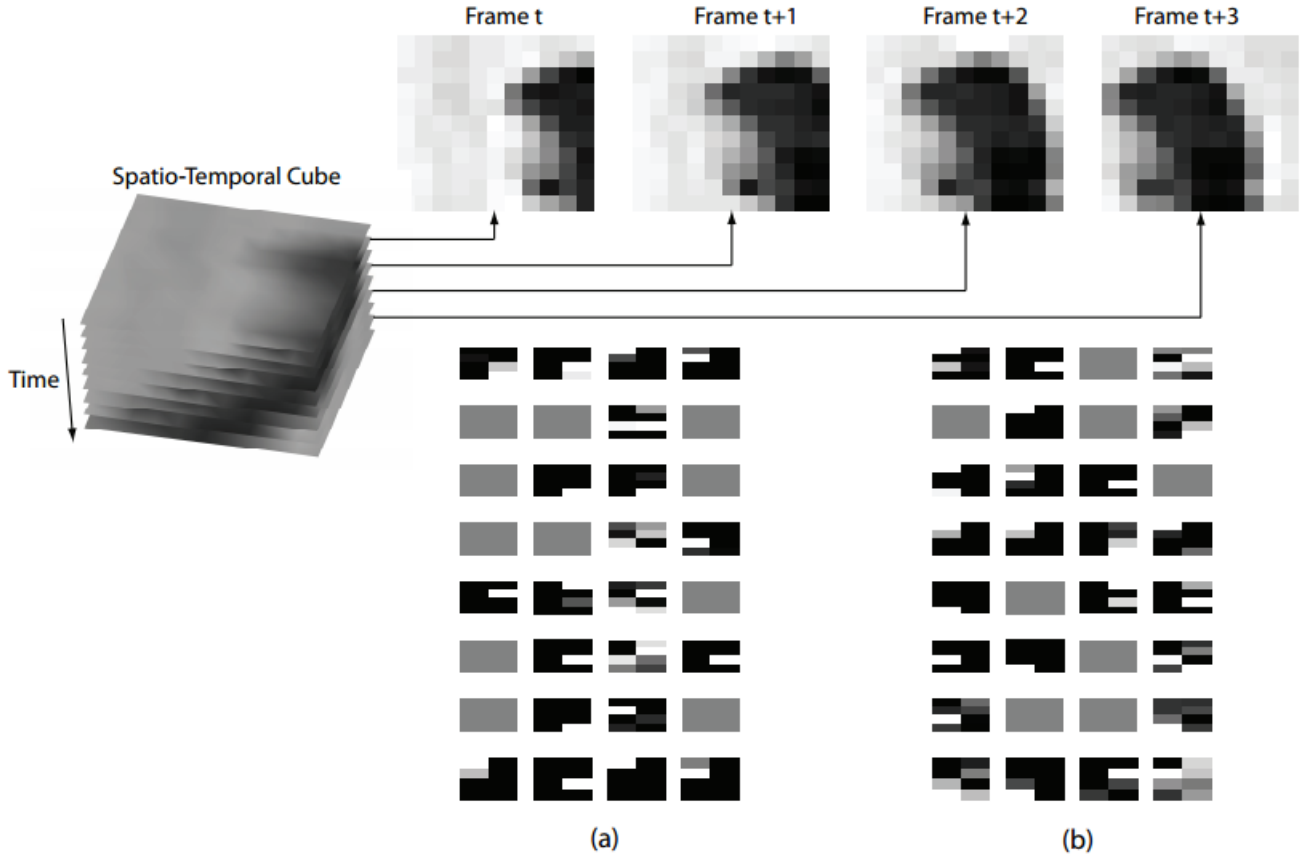


Figure 4.8: (a) Shows the Descriptor before global reorientation by the overall maximum orientation direction. (b)Shows the Descriptor after global reorientation.

3.5 3D HOG using a spatio-temporal descriptor

Based on the work of Scovanner et al., Klaeser et al. developed a spatio-temporal descriptor based on 3D Gradients. In the first step Klaeser et al. reorganize the orientation quantization. Whileas in a 2D image polygons have to be analysed, in 3D you have to analyze polyhedrons. Klaeser et al. chose the dodecahedron (12-sided) and the icosahedron (20-sided) for 3D gradient quantization. To quantize a 3D gradient vector, it gets first projected on the axes running through the origin of the coordinate system and the center positions of all faces [6]. Let $v(x, y, t)$ be a video sequence and its partial derivatives along x,y,t be $v_{\partial x}$, $v_{\partial y}$, $v_{\partial t}$. The integral video for $v_{\partial x}$ (and $v_{\partial y}$, $v_{\partial t}$ respectively) can be described as followed:

$$iv_{\partial x} = \sum_{x' \leq x, y' \leq y, t' \leq t} v_{\partial x}(x', y', t') \tag{4.24}$$

Further on I will describe any cuboid with $b = (x, y, t, w, h, l)^T$, where $(x, y, t)^T$ relates to its position, (w) relates to its width, (h) relates to its height, and (l) relates to its length. Let $\overline{g}_b$ be the mean gradient, which means $\overline{g}_b = (\overline{g}_{b\partial x}, \overline{g}_{b\partial y}, \overline{g}_{b\partial t})^T$ with $\overline{g}_{b\partial x}$ (and $\overline{g}_{b\partial y}, \overline{g}_{b\partial t}$ respectively):

$$\overline{g}_{b\partial x} = [iv_{\partial x}(x + w, y + h, t + l) - iv_{\partial x}(x, y + h, t + l) - iv_{\partial x}(x + w, y, t + l) + iv_{\partial x}(x, y, t + l)] - \tag{4.25}$$

$$[iv_{\partial x}(x + w, y + h, t) - iv_{\partial x}(x, y + h, t) - iv_{\partial x}(x + w, y, t) + iv_{\partial x}(x, y, t)]$$

Klaeser et al. realised the quantization of the 3D graduebt vector with matrix multiplication, where P is the matrix of the center positions $p_1, ..., p_n$ of all n faces:

$$P = (p_1, ..., p_n) \text{ with } p_i = (x_i, y_i, t_i)^T \tag{4.26}$$

The projection $\widehat{q}_b$ of $\overline{g}_b$ is computed as followed:

$$\widehat{q}_b = (\widehat{q}_{b1}, ..., \widehat{q}_{bn})^T = \frac{P\overline{g}_b}{\|\overline{g}_b\|_2} \tag{4.27}$$

To optimize the projections Klaeser et al. also added a threshold histogram $\widehat{q}'_b$:

$$q_b = \frac{\|\overline{g}_b\|_2 \widehat{q}'_b}{\|\widehat{q}'_b\|_2} \tag{4.28}$$

With the gathered informations we now can compute the histograms with the quantized mean gradients q_{b_i} of all sub-blocks b_i and a fixed number of supporting mean gradient vectors S^3:

$$h_c = \sum_{i=1}^{S^3} q_{b_i} \tag{4.29}$$

Out of those previous thoughts we are now able to compute our descriptor-vector. Let w_s be the width, h_s the height and l_s the length for a local support region around a position s:

$$w_s = h_s = \sigma_0 \sigma_s$$

$$l_s = \tau_0 \tau_s. \tag{4.30}$$

Where σ_0 and τ_0 describe the relative size of the support region around s. The support region is now divided into $MxMxN$ cells. Finally all histograms are computed and concatenated to the descriptor-vector: local support region around a position s:

$$d_s = (h_1, ..., h_{M^2N})^T \tag{4.31}$$

Figure 9 illustrates the described functions.

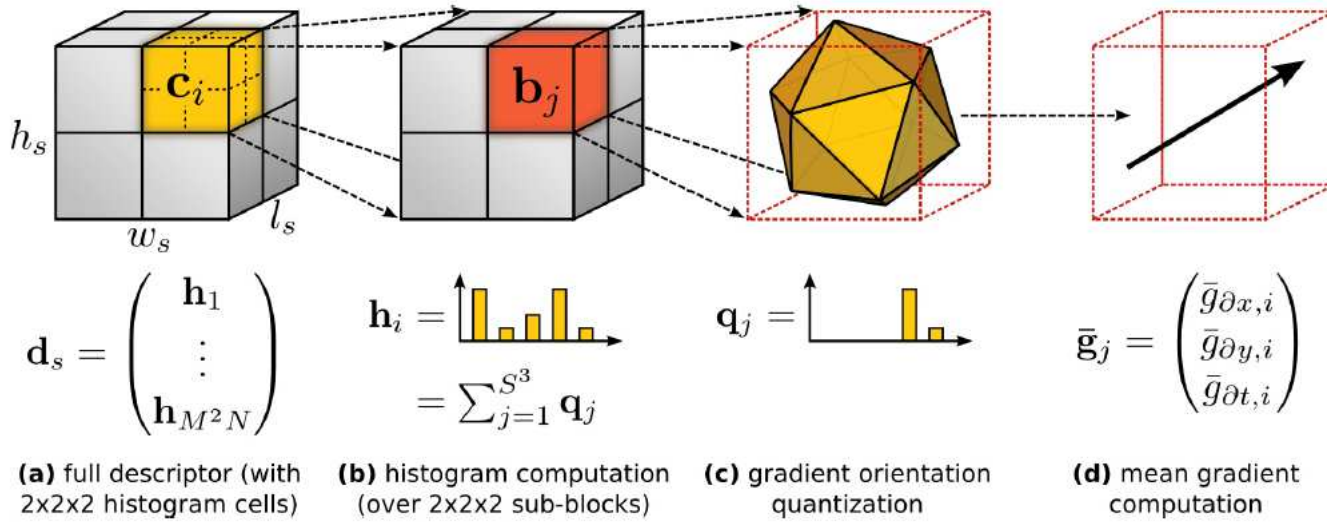


Figure 4.9: (a) Grid of gradient orientation istograms computed with the support region around a point of interest. (b) Histogram computed over a grid of mean gradients. (c) Gradient quantization is quantized using regular polyhedrons(Reminder: the 2D SIFT Descriptor used a Gaussian window for quantization) (d) Mean gradients used to compute integral videos.

3.6 Dense SIFT

The Dense SIFT approach of Liu et al. is trying to match objectives along flow vectors, and smoothing flow fields. Flow vectors describe the euclidean distance between changing locations of key points and flow fields are simply a set of the flow vectors for an object. Liu et al. defined a energy function, which is going to compute the flow vectors, to match objectives: local support region around a position s:

$$E(w) = \sum_{p} min(\|s_1(p) - s_2(p + w(p))\|_1, t) +$$

$$\sum_{p} \eta(|u(p)| + |v(p)|) + \sum_{(p,q)\in\varepsilon} min(\alpha|u(p) - u(q)|, d) \tag{4.32}$$

$$+ min(\alpha|v(p) - v(q)|, d)$$

Where $p = (x, y)$ is the grid coordinate of images, $w(p) = (u(p), v(p))$ is the flow vector at the grid coordinate and s_1 and s_2 the two images with the moving object have to be matched with t and d as thresholds obtained by experienced data. Equation 32 can be divided into three parts. A data term part, which constrains the SIFT descriptors along with the flow vector. A small displacement term, which is using as small as possible flow vectors. And a smoothness term, which constraints similar flow vectors of adjacent pixels. Figure 10 illustrates the decoupled horizontal and vertical components of a Dual-Layer Belief Propagation. However this dual-layer belief propagation scales poorly with respect to image

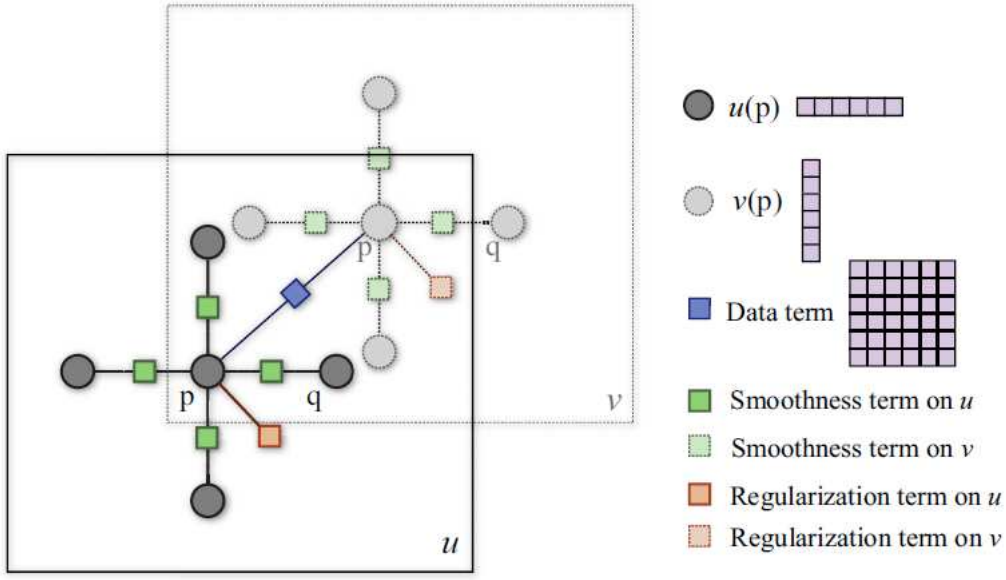


Figure 4.10: Matching Objectives with decoupled horizontal and vertical components.

dimension. Analysing a 80 x 80 pixels image would take 50 seconds, whereas a 256 x 256 image requires more than two hours. It is still possible to optimize that with matching schemes like Liu et al. did with his coarse-to-fine matching scheme. Still Dense SIFT loses partially scale-invariance, which is what SIFT is used for. In the next Descriptor section, I will analyse the Scale-Less-SIFT (or: SLS) Descriptor and give an example of comparing SLS with Dense-SIFT.

3.7 Scale-Less-SIFT

The Scale-Less-SIFT (or: SLS) Descriptor is trying to represent a scale for each pixel, rather than the more common approach of using subspaces. Therefore Hassner et al. employed a subspace-to-point mapping, to produce the SLS descriptor for each subspace. Let $\mathcal{H}_p$ be the subspace produced at p, representing a 128 x d matrix with orthonormal columns. Hassner et al. produce the SLS representation by mapping subspace to a Point p by rearranging the elements of the projection matrix $A = \widehat{H}\widehat{H}^T$ using the following operator [4]:

$$P \triangleq SLS(\widehat{H}_p) = (\frac{a_{11}}{\sqrt{2}}, a_{12}, ..., a_{1d}, \frac{a_{22}}{\sqrt{2}}, a_{23}, ..., \frac{a_{dd}}{\sqrt{2}})^T \tag{4.33}$$

In order to capture the behaviour of SIFT descriptors throughout scale space the following normalization has to be performed, for a constant μ that is used as a threshold based on experienced data from video sets, at a point P:

$$\|P - P'\|^2 = \mu \ \mathrm{dist}^2(\mathcal{H}_p, \mathcal{H}_{p'}) \tag{4.34}$$

Figure 11 is showing a direct comparison of the solutions of Dense-SIFT(or: DSIFT) and SLS.

Figure 4.11: The first two images show the original images. The third image is showing the DSIFT solution and the fourth image is showing the SLS solution. Keep in mind that DSIFT is trying to compute a vector flow here, while SLS is trying to find a scale space and resolution for the best match of the two original images.

4 Applications

SIFT can be used in a variety of ways to solve complex tasks in Image Processing, which can be applied to Medical Image Processing as well.

4.1 SIFT Flow

Dense-SIFT (or: DSIFT) is an example that is using vector flows and flow fields, which are applied by SIFT Flow. SIFT Flow can be used for several tasks, e.g. Motion Prediction and Face Recognition.

4.1.1 Motion Prediction is a common task in image alignment. Liu et al. tested their Dense-SIFT method for motion prediction. They used a database consisting of common events and objects, like cars driving or children playing. Each individual frame was stored in the database as a vector of word-quantized SIFT features. Also the temporal motion field is estimated using every two consecutive frames of each video. Figure 12 illustrates the motion prediction from a single image.

4.1.2 Face Recognition In this application example of Dense-SIFT Liu et al. used a ORL database, containing ten samples of a person with different poses, expressions and lighting changes for face recognition. First an image has to be chosen for the query. The next part is an aligning with the rest of the images to the query, and than a warping of the images with respect to dense correspondence, using Dense-SIFT. Figure 13 illustrates the solutions.

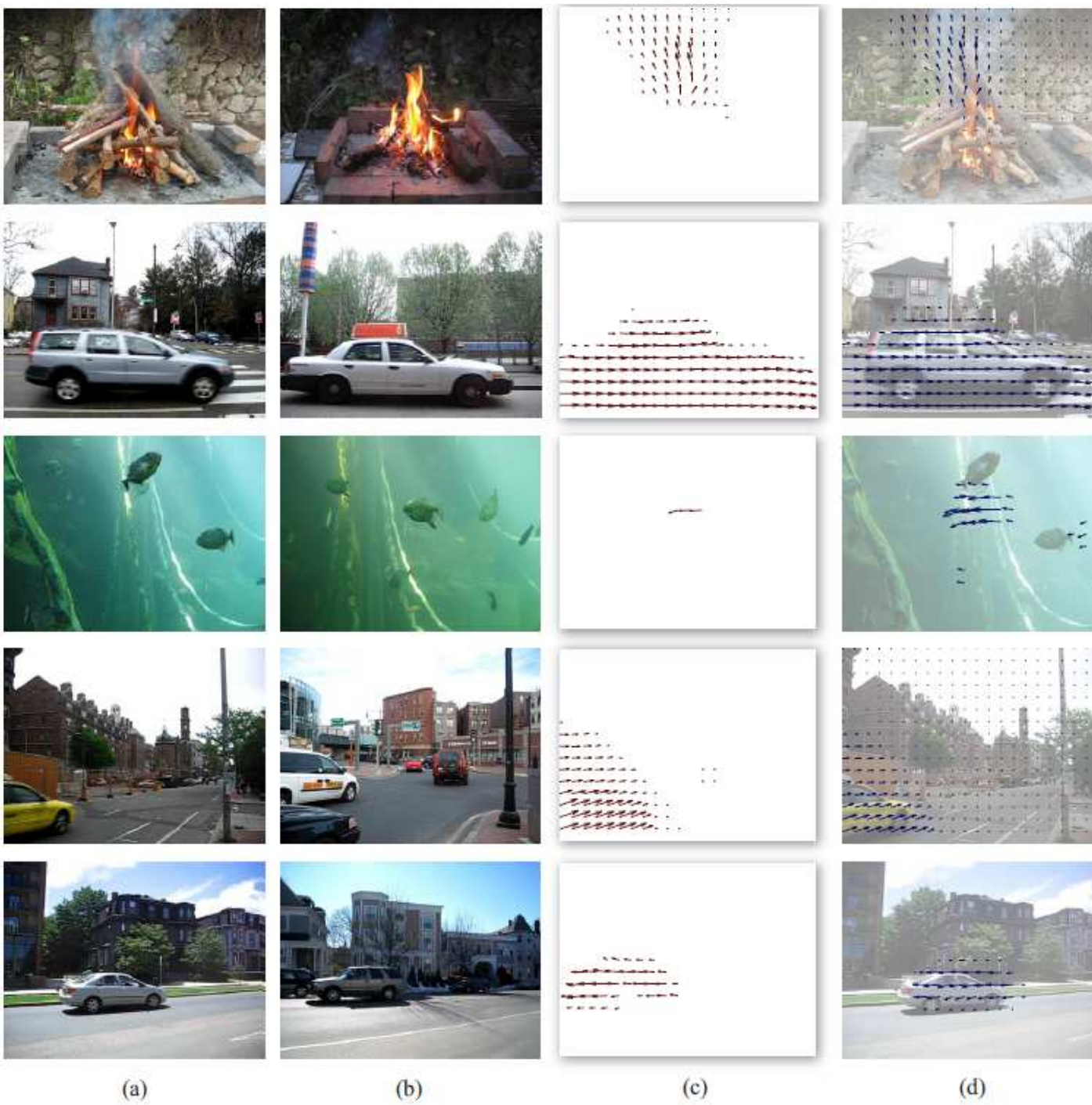


Figure 4.12: (a) is showing the analysed image. (b) is the best matched image of the database. The temporal motion field of (b) is illustrated in (c). (d) illustrates the warped motion of (c) on (a). (e) is the ground truth of (a)

Figure 4.13: (a): The ten samples of the ORL database. (b): Solution with the first image used as query image. (c) Solution with the fifth image used as query image.

4.2 Computation of Panoramas

Lowe and Brown used SIFT to employ the Auto-Stitch Application for smart-phones. The smart-phone has to be rotated while taking pictures of a scene. Each photo should have an overlay to the closest pictures around it, in order to be able to recognize the location of the actual picture. While taking the photos the Application provides direct Feedback, recognizing the already photographed parts of the picture and indicating them with a blue square. After Auto-Stitch used the SIFT method to stitch the overlays together and smoothing the edges, the user can crop the panorama. Figure 14 is showing the result of a panorama, where Auto-Stitch has been used.

4.3 Robotic Surgical Tools

Reiter et al. experimented with tool tracking for Robotic Surgical Tools. They used a $daVinci^{®}$ robot for their evaluation. Their experiment analysed that SIFT is capable of tool tracking. Figure 15 shows the robot-arm and the tracked keypoints.

5 Discussion

The most common approaches to detect and describe feature and flow correspondence in video or image data are SIFT and HOG. Since SIFT was introduced (Lowe, 1999) many different approaches have been made to optimize SIFT. The most efficient approach is the Dense-SIFT method, which is using flow vectors and smoothed flow fields with schemes to optimize the computation time, still Dense-SIFT lacks in robustness. Other approaches like the Scale-Less-SIFT approach are very robust, but can't be efficiently used since they lack in real-time capability, because of the long computation time. Also it is highly questionable to compare their Descriptor with other Descriptors that are fulfilling ohter purposes than detecting and describing a matching scale and resloution of an image. Dense-SIFT is using coarse-to-fine pyramids, in order to obtain a better time complexity. For real-time solutions Dense-SIFT is the most common method used. These approaches only take account in 2D SIFT, 3D SIFT and 3D HOG on the other hand is mostly used for action recognition in videos. Still 3D SIFT and 3D HOG approaches lack in time complexity. HOG is still the most common used. However SIFT is resulting in more reliable description of images. There are different reasons why HOG could be more common than SIFT. First of all SIFT is patanted, which leads to the more cost efficient approach of HOG. SIFT and HOG have a high amount of responses on edges, which makes HOG a viable option for applications like pedestrian recognition or uncluttered sceneries.

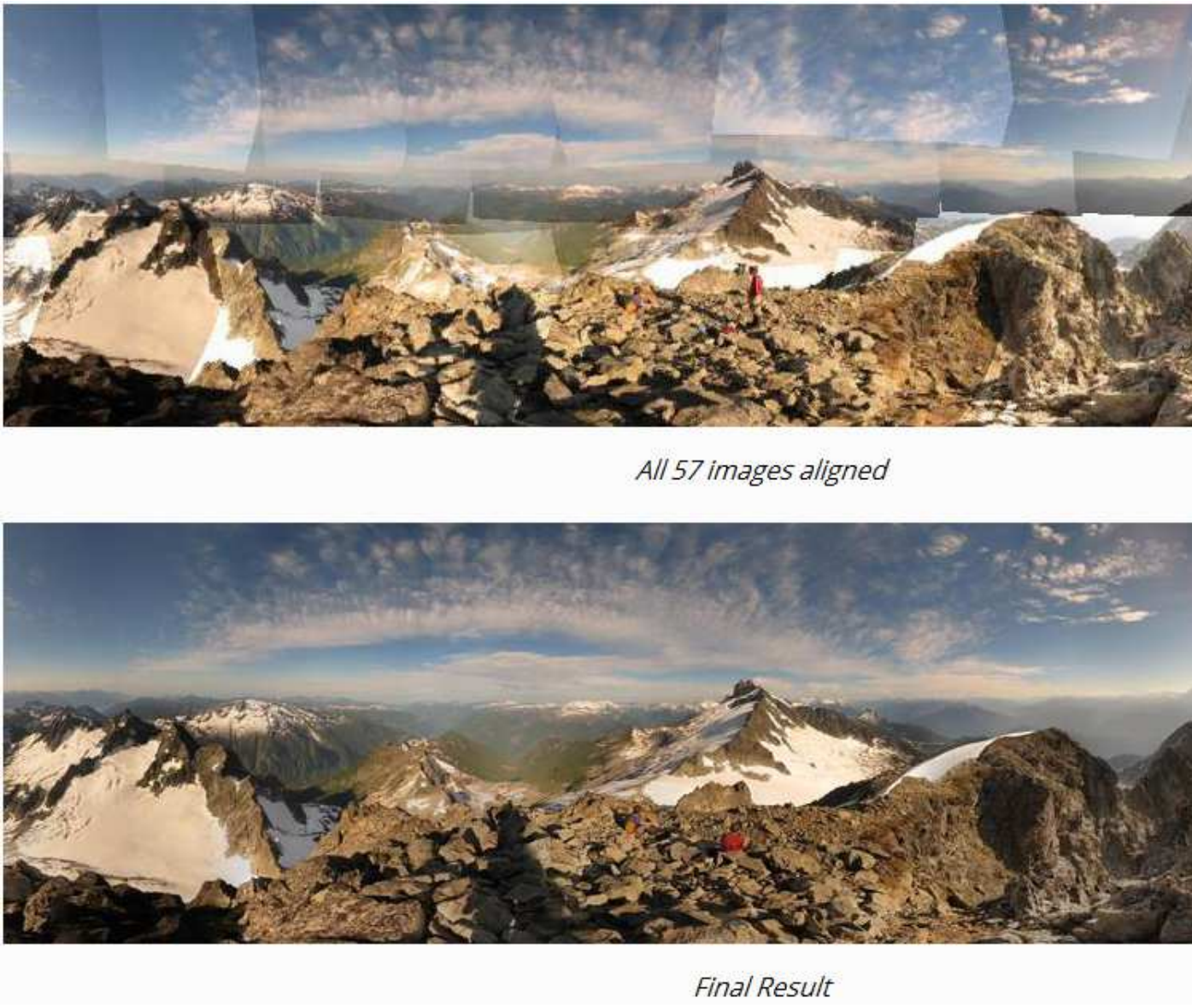


Figure 4.14: For more details visit: http://www.cs.bath.ac.uk/brown/autostitch/autostitch.html

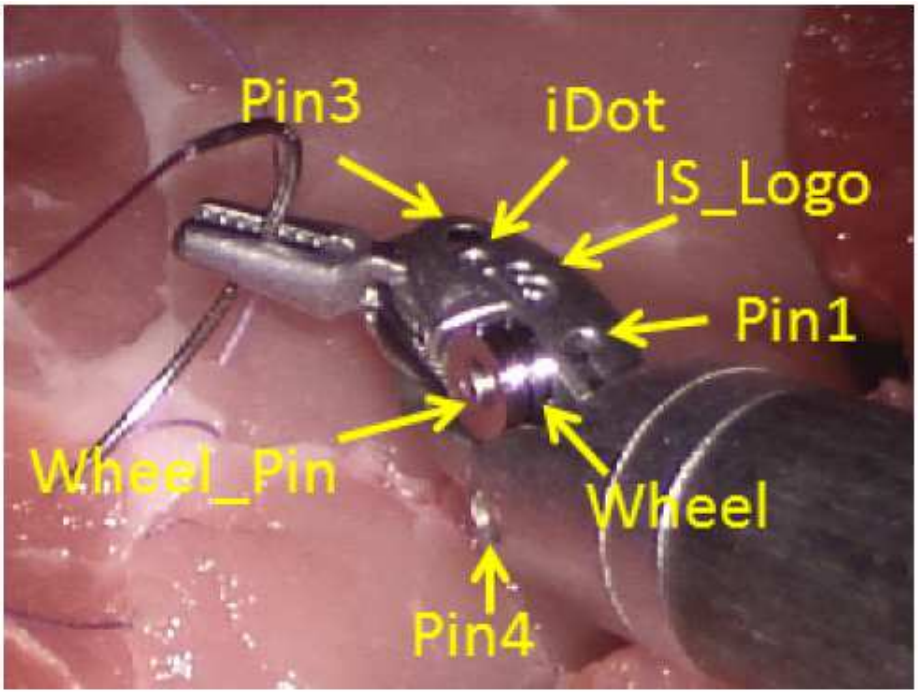


Figure 4.15: The robot-arm and the tracked keypoints on it.

References

[1] Wolfhard Lawrenz and Nils Obermöller. *CAN Controller Area Networt: Grundlagen, Design, Anwendungen, Testtechnik*. VDE Verlag GmbH, Bismarckstr. 33, 10625 Berlin, 5., aufl. edition, 2011.

[1] Lowe: Distinctive Image Features from Scale-Invariant Keypoints
[2] Lowe: Object Recognition from Local Scale-Invariant Features
[3] Liu: SIFT Flow: Dense Correspondence across Scenes and its Applications
[4] Hassner: On SIFTs and their Scales
[5] Mikolajczyk: Scale and Affine Invariant Point Detectors
[6] Klaeser: A Spatio-Temporal Descriptor Based on 3D-Gradients
[7] Scovanner: A 3-Dimensional SIFT Descriptor and its Application to Action Recognition
[8] Reiter: Learning Features on Robotic Surgical Tools

Object Segmentation using ASM and Good Features

Isaak Lim

Contents

Abstract

In this report we examine Object Segmentation using Active Shape Models. For this we will present the original approach and further improvements. The advances presented are the introduction of the Gray-level Appearance Model and a method to handle the problem of converging to local minima when training and segmenting with Active Shape Models. We also present an application of Active Shape Models in the field of Medical Image Processing and discuss the merits of the different presented approaches.

1 Motivation

As computers become more powerful over time, the possible applications of software to enhance and support daily life also grows. This is also true for the fields of computer vision and machine learning. With the increasing amount of data generated by hardware or software it is becoming harder to extract useful information from it. Segmentation in the domain of computer vision strives to handle the extraction of useful information from images. Its aim is to identify and extract given objects and classes of objects from images.
While computer vision has some well known applications in the automotive industry, it can also be applied to solve problems in the medical field. A major point of interest is to support medical professionals in their evaluation of medical images and the subsequent decision making process.
Since the correct assessment of medical-related images (e.g. CT images) is often vital, physicians frequently ask for second opinions in cases where the analysis is not easy. Thus, it would greatly support their work if software were to identify interesting regions or objects in question. It would not only help physicians make informed decisions but also give warnings if possibly vital information was missed or neglected.
In recent years Active Shape Models have emerged as robust solution for supervised segmentation of images. Apart from being used to identify and segment objects in a production line setup, they can also be used in the context of medical image processing.
In this report we will first explain what Active Shape Models are and how they are constructed in Section 2. We start by giving a general overview in Section 2.1. Next, we will then examine some improvements and developments that were proposed in recent years in 2.2. We continue with an example of an application of Active Shape Models in medical image processing in Section 3. A discussion of the methods and results presented in Section 2.2 follows in Section 4. Finally we conclude this report in Section 5.

2 Active Shape Model Segmentation

As van Ginneken et al. [6] point out, segmentation can generally be approached from two sides. When taking the bottom-up approach the image structure is analyzed and described as a collection of low-level elements and their spatial relationship to each other. The top-down approaches try to build a high level model that describes the class of objects that should be identified in images and segmented correctly. As van Ginneken et al. note, the problem with bottom-up approaches is that image structure alone does not necessarily provide a meaningful segmentation.
Active Shape Models are a top-down approach as the name implies. The model interprets points on the contour of the corresponding object as a point distribution. The goal is to capture the variations of the shape of the object class during training. The model is then used during fitting by displacing the points on the boundary of the model until they match the shape of the object in a given image.

2.1 General Method

We will now give an overview of the general method used to generate and apply Active Shape Models as presented by Cootes et al. [2]. The general approach is to first train the model with a set of training images as explained in Section 2.1.1. The second step is to use this model during the fitting process for new images that we would like to segment, which will be presented in Section 2.1.2.

2.1.1 Training The Active Shape Model Segmentation with ASMs is supervised, which means that we first need to a set of training images to construct the models in question. This is done by manually placing points on boundaries or features of the objects in the set of training images that we would like to recognize.

Once the points have been placed, a number of representative points are chosen. Cootes et al. [2] name three types of representative points or landmark points that can be chosen. The first type are application dependent points, which means that they depend on the type of objects we would like to recognize. An example for this would be selecting the points that mark sharp corners. The second type are application independent points, which as the name suggest do not depend on the type of objects that have to be picked out, like curvature extrema along the boundaries of the shape defined by the placed points. The third and last type of landmark points are points that can be interpolated from the first two set of points. Taking the midway points on edges between the landmark points of the first two types, would be an example.
After acquiring the set of landmark points we have to align the shapes from all training images in order to build a meaningful model. Therefore, we have to find a transformation, namely a rotation, scaling and translation, that minimizes the weighted sum of squares of the distances between corresponding landmark points of the shapes. Once all shapes are aligned to a given shape we calculate the mean shape from the aligned shapes.
Next we normalize the mean shape, which means that we choose some default orientation, scale and origin. Finally we realign all shapes to the mean. We repeat this process of calculating the mean, normalization and realignment until convergence. The normalization is essential to ensure convergence, since otherwise we would not have any meaningful comparison measure of alignment. Van Ginneken et al. note that in some cases alignment is not desired and therefore omitted [6]. This can be if the shape of the objects only varies slightly within a given range.
Now that we have aligned the shapes we want to derive a model that describes these shapes. This is done by statistically analyzing the set of given shapes. We can describe each shape by the coordinates of its landmark points. Assuming that the shapes consist of n landmark points each, we can describe each shape as a vector of n 2-dim points.
Furthermore, we can specify each shape to be a point in $2n$-dim space. Since the landmark points are partially correlated the points representing the shapes form a cluster in $2n$-dim space. Cootes et al. make the assumption that the points roughly approximate an high-dimensional ellipsoidal object. Following the assumption it is now possible to describe this cluster in terms of a center and major axes. The center is simply the mean $\overline{x}$. In order to capture the possible variations of the shape we calculate the deviation from the mean

$$dx_i = x_i - \overline{x} \tag{5.1}$$

where x_i is the shape vector. The deviations allow us to build a covariance matrix

$$\Sigma = \frac{1}{k} \sum_{i=1}^{k} dx_i dx_i^T \tag{5.2}$$

where k is the number of shapes extracted from training images.
Now we can perform PCA on the covariance matrix Σ in order to calculate the major axes defining the ellipsoidal object. Naturally, the eigenvector corresponding to the largest eigenvalue gives us the longest axis, along which the greatest variation can be found as visible in Figure 5.1. The eigenvalues of Σ are the variance of the shapes or the squared standard deviation from the mean. We can approximate the ellipsoid by choosing the l eigenvectors that cover the most variation.
It is now possible to model possible variations in the shape via linear combination of the eigenvectors of Σ. In order to keep the variations reasonable, Cootes et al. propose to clamp the scaling b_i of each eigenvector e_i to 3σ in the linear combination. This can also be described in terms of the eigenvalues of Σ as follows:

$$-3\sqrt{\lambda_i} \leq b_i \leq 3\sqrt{\lambda_i} \tag{5.3}$$

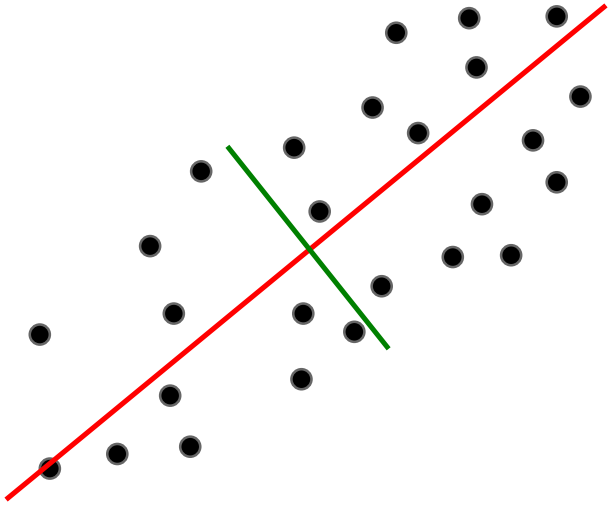


Figure 5.1: Illustration of the Principal Component Analysis in 2D. The red line represents the first Principal Component (eigenvector to the largest eigenvalue) and the green line represents the second Principal Component (eigenvector to the second largest eigenvalue)

2.1.2 Fitting The Active Shape Model Assuming the shape we are searching for is visible in the image, we now need to deform our ASM in such a way that it matches the shape. The boundary of the object we want to segment is given by the displaced landmark points of the ASM.
While there are several methods and approaches to training and fitting the ASM we need an initial estimate of the position of the landmark points in the image. Once the points are placed, we then iteratively transform the landmark points. This transformation is repeated until the position of the landmark points does not change significantly anymore. By examining the image structure around the landmark points in question we can determine if we should continue displacing the points.
Ideally this would result in landmark points that are placed on the boundary of the object in the image we are segmenting. However, this hugely depends on the quality of the model we have trained. Possible approaches will be discussed in more detail in Section 2.2.

2.2 Advanced Approaches

In this section we will mostly focus on the works of van Ginneken et al. [6] and Brunet et al. [1], which expand on the general method.
A first idea for fitting a model, might be to simply move each landmark point along one of the major axes computed with the PCA for the ASM or even along the normals of the boundary. However, the problem here is that the previous model is based on the assumption that there is a linear relationship between landmark points. This is why van Ginneken et al. propose an approach, where a gray-level appearance model is constructed in addition to the ASM and non-linear kNN-classifiers are used for training ASMs and then segmenting new images [6].

2.2.1 The Gray-level Appearance model In order to illustrate the proposed concept we will give an overview of what a gray-level appearance model is. In essence, a gray-level appearance model describes the image structure around a given landmark point. By calculating this model from the training images, we can then displace the landmark points of the mean model during the fitting process for a new image in such a way that the image structure around the transformed points is most similar to those of the model.
If we imagine the landmark points to be connected by edges as shown in Figure 5.2, the gray-level appearance model describes the slice through the landmark point along the normal defined by the edges to it's neighboring points in this case.

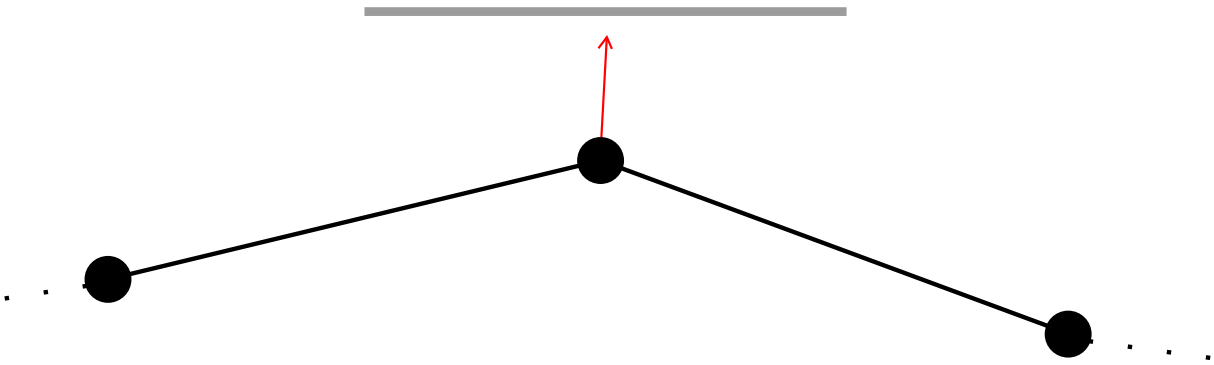


Figure 5.2: Illustration of landmark points (black dots) connected by edges. The red arrow represents the normal at one landmark point.

Along the normal we can sample an image k times in each direction, resulting in $2k+1$ samples overall for each landmark point. Since we want to describe the structure around landmark points in terms of the possible boundaries of the shape for our model, we need gradient information. From now on we will refer to the captured image structure around each landmark point as the profile.
Cootes and Taylor proposed to use normalized first derivates of the structure or the profile around the landmark point in question [3]. The mean $\overline{g}$ of these normalized first derivatives of profiles and the covariance matrix Σ_g describing the relationship between the normalized first derivatives profiles, composes the gray-level appearance model. g_i denotes an i-th profile for the i-th landmark out of the set of all landmarks. This allows us to compute the Mahalanobis distance of a profile of a displaced landmark g_i to the model during the fitting process as follows:

$$(g_i - \overline{g})^T \Sigma_g^{-1} (g_i - \overline{g}) \tag{5.4}$$

We can now compute the shape by minimizing the Mahalanobis distance.
However, this approach by Cootes and Taylor assumes that the landmark points and indeed their profiles are distributed normally. This is where van Ginneken et al. propose a different procedure to build the gray-level appearance model, since they argue that non-normal distributions of profiles occur frequently in practice [6].
Van Ginneken et al. point out that the best possible location to move the landmark point in question to during fitting, is the point where the profile of that location is “inside” the object we want to find on one side and “outside” on the other side. Since we would like to segment images of varying resolutions, van Ginneken et al. propose to use multi-scale Gaussian derivatives when constructing the gray-level appearance model. Therefore, we do not sample the original image we want to perform segmentation on, but instead first filter the image by convolution with derivates of bivariate Gaussians. In practice the first and second derivates are used.
It is now possible to describe the image structure or profile around each landmark points with a histogram of the intensity values of the filtered image. The first moment of the histogram gives us the mean value, while the second moment specifies the variance from the mean. It has to be noted that finding good parameters for the standard deviation σ of the Gaussians and the size of the window that the histogram covers for every conceivable image is not easy.

2.2.2 Training We will now present the proposed approach to training by van Ginneken et al. [6]. Since multi-resolution segmentation has to be supported, the task is to find the best set of features for each profile for all given resolutions as mentioned above. This means the set of features that enable the correct identification of the corresponding landmark in a given image. The features found in a profile for each landmark point can be described as a feature vector. We want to train a classifier such that on examining such a feature vector describing a sample of the profile around a landmark point we can classify it as “inside” or “outside” an object. Van Ginneken et al. arbitrarily define the landmark points themselves to be ”inside” the object.
In order to train the classifier, we have to validate its predictions. Van Ginneken et al. propose to use half of the image training set for training and the other half for validation. As a classifier 5-Nearest-Neighbor classifier is used. The idea is that when a new sample has to be labeled, its 5 nearest neighbors are

examined, which are already classified samples. A majority vote is then used to determine the class of the new label. So if 3 out of 5 nearest neighbors belong to class 'A', the new sample will also be assigned to class 'A'. An example of this is shown in Figure 5.3.

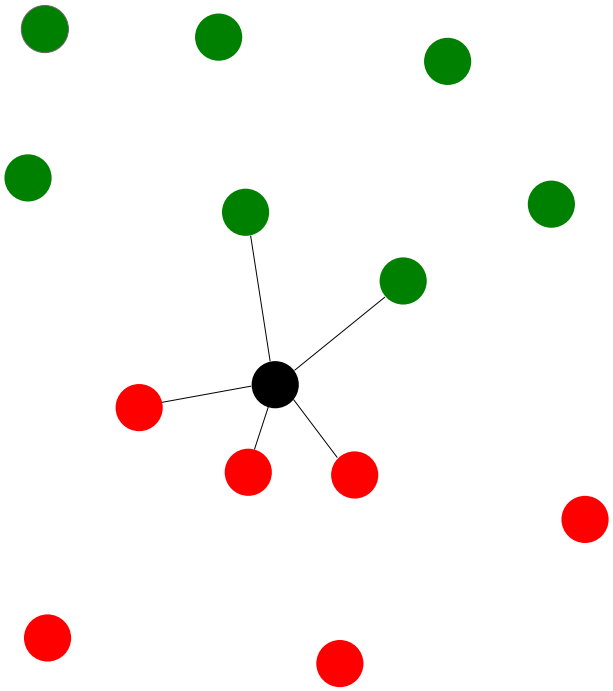


Figure 5.3: Illustration of the 5-NN classifier. The red and green dots represent two classes. The new sample is represented by a black dot. Its five nearest neighbors are connected to it by black lines. Since the majority of the 5 nearest neighbors belong to the red class, the new sample would also be labeled as red.

Van Ginneken et al. use feature vectors of size 60 and therefore introduce weighting to voting. The weight for each vote is $exp(-d^2)$, where d is the euclidean distance. We will further discuss the merits and disadvantages of this approach in Section 4.
In order to reduce the amount of features used to describe the model for performance reasons, while minimizing the loss of information van Ginneken et al. use sequential feature forward selection to select a given number of features out of the complete feature set for each landmark. Sequential feature forward selection is an algorithm that sequentially adds features that maximize some criterion function. While van Ginneken et al. do not explicitly state what criterion function they use, an example could be the Bhattacharyya distance function.
This function can be used to measure the similarity between two distributions. In the case of feature selection we select features that are most dissimilar to the other features in order to get a good coverage of the complete set of features for each grid of samples around each landmark point. Afterwards, sequential feature backward selection is applied. This means that features are removed from our subset if they improve the prediction accuracy. Somol et al. further expand on this method [4].

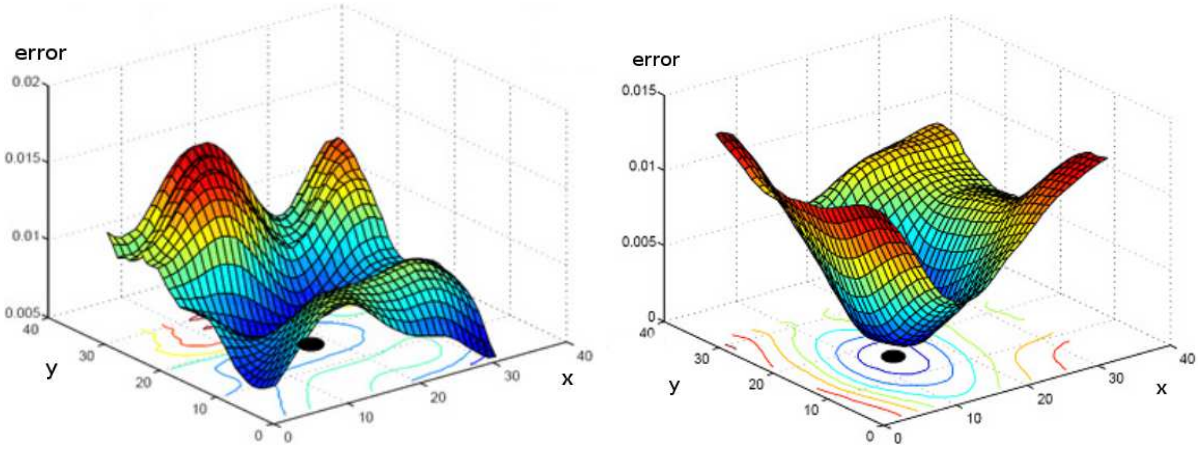
2.2.3 Fitting Now that we have trained the classifier, we can segment new images containing the object we have trained our models for. Van Ginneken et al. propose to initialize with the mean shape and start at the lowest level of resolution. The points on the profiles are used as input for the kNN-classifier. Unlike the approach presented in Section 2.2.1, we do not try to minimize the Mahalanobis distance in Equation 5.4, but the objective function

$$\sum_{i=-k}^{-1} p_i + \sum_{i=0}^{+k} (1 - p_i) \tag{5.5}$$

for all $2k+1$ points on each profile g. p_i stands for the probability of the i-th point being inside or outside the object we want to segment computed by the classifier. We then move the landmark points to the

points that minimizes this function. The index i denotes the index of each point on the profile running from $-k$ to k. The landmark point is positioned at the index $i = 0$. The probability 0 means that the point is outside the object and the probability 1 means that the point is inside the object. Therefore, minimizing the objective function for all profiles means displacing the landmarks in such a way that one side of the profiles with the indices of the points running from $-k$ to -1 are "outside" and the other side with indices of the points running from 0 to $+k$ are inside the object.
The two steps of evaluation and the movement of landmark points is repeated a set amount of times. Afterwards, these to steps are repeated for all higher resolutions. This results in the segmentation of a given image, where the shape of the object we are looking for is defined as the edges between the moved landmark points.

2.2.4 Handling Local Minima In this Section we will present the problem of converging to local minima in the context of ASMs and the proposed solution by Brunet et al. [1]. In order to train the model the image structure around the landmark point has to be considered as explained in Section 2.2.1. However, instead of using a slice throught the landmark here we use a 2D region or patch around the landmark as the profile for the gray-level appearance model.



(a) Error surface of a bad profile [1]

(b) Error surface of a good profile [1]

Figure 5.4: Examples of the error surface of good and bad regions around landmarks [1]

If we use PCA during training and pick l eigenvectors as the basis to describe a shape, we essentially reduce the dimensionality of the point cloud distribution. In the same way we can use a PCA of the feature vectors of the profiles in order to make an efficient evaluation and comparison of the image structure in the image we want to segment and our trained model possible. For every profile of each landmark point we can then calculate the reconstruction error for each point in the region as a measure of quality. The reconstruction error is the distance between the original value at that point and the linear combination of the l eigenvectors or any subspace that approximates the value at that point.
As visible in Figure 5.4(b), good features are regions around landmark points that have one local minimum instead of several as shown in Figure 5.4(a). If we have several local minima in the region we might move our landmark point to a minima which is not optimal during the fitting process. Therefore, a subspace needs to be found that minimizes the amount of local minima of the error surface around the landmark. We also want to dispose of the landmarks whose regions are not well suited for training our model.
Brunet et al. propose an approach to solve this problem by defining a criterion which makes it possible to select good regions around landmarks and introducing a supervised PCA method, which allows learning a good error surface. Let us first discuss the criterion by which landmarks and their regions are evaluated. Brunet et al. define the optimal placement of the landmark during fitting as minimizing:

$$\min_{a_1,a_2} \frac{||d_i^{x+a_1,y+a_2} - \mu - Bc_i||_2^2}{||d_i^{x+a_1,y+a_2}||_2^2} \tag{5.6}$$

a_1 and a_2 are translation parameters with regard to the positions x and y. $d^{x+a_1,y+a_2}$ are the feature values of the profile around the position $(x + a_1, y + a_2)$. B is the basis for an appearance model such as

the gray-level-appearance model given by PCA for example, while c contains the coefficients of the linear combination of the basis vectors. Finally, μ is the mean of the appearance model. The minimization is done by searching for the best parameters for a_1, a_2 and c. The results of this search result in the error surface as seen above.
In order to describe the properties of the error surface we count the number of local minima, consider the distance between the global minimum of the error surface and the expected position of the landmark and finally the distance between the global minimum of the error surface and the closest minimum. These properties are influenced by varying the size of the profile and the amount of basis vectors we use for our appearance model.
Brunet et al. compute local PCA with varying size of the profile and amount of basis vectors for each landmark independently. They found that the more basis vectors are taken into account the more local minima exist. Furthermore, increasing the size of the profile also increases the number of local minima. However, as expected the density of local minima decreases with increasing size of the profile. After evaluating these factors for each landmark Brunet et al. found a good trade-off to be profile sizes of 30 x 30 pixels and a basis, which has 80% of the original dimensionality of the appearance model.
Using these parameters on all the landmarks and their profiles gives us a criterion, which allows us to compare profiles and rank them. Brunet et al. do this by describing each profile as follows:

$$\mu_{profile} = \mu_{1_{profile}} + \mu_{2_{profile}} \tag{5.7}$$

$$\sigma_{profile} = \frac{\mu_{1_{profile}}\sigma_{1_{profile}} + \mu_{2_{profile}}\sigma_{2_{profile}}}{\mu_{1_{profile}} + \mu_{2_{profile}}} \tag{5.8}$$

Here $\mu_{1_{profile}}$ and $\sigma_{1_{profile}}$ are the mean and variance of the distance between the global minimum of the profiles to the expected position of the landmark. $\mu_{2_{profile}}$ and $\sigma_{2_{profile}}$ are the mean and variance of the number of local minima of the profiles. The profiles are then ranked according to how well they minimize $\mu_{profile}$. This allows us to select a given number of the best landmarks and their profiles for training our model.
As mentioned above we would also like to find a subspace of features for our ASM that minimizes the amount of local minima. The best case would be if the error surface were to resemble an inverted Gaussian as visible in Figure 5.5.

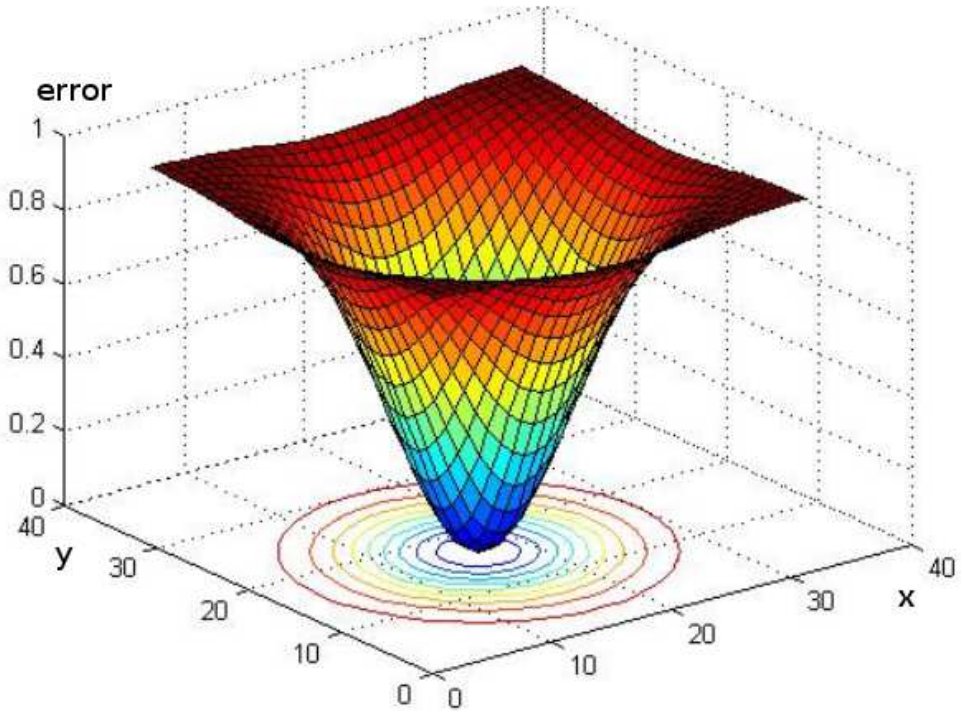


Figure 5.5: Inverted Gaussian response function [1]

Therefore, Brunet et al. evaluate the error surface of each profile by comparing it to an inverted Gaussian

$$E = \sum_{i=1}^{n} \sum_{(x,y)} (\alpha_i r_{x,y} + o_i - \frac{||d_i^{(x,y)} - \mu - Bc_i||_2^2}{||d_i^{(x,y)}||_2^2}) \tag{5.9}$$

Here $r_{x,y}$ is the response of the inverted Gaussian at the position (x, y). α_i is the scaling factor of the inverted Gaussian and o_i is the translation factor. Minimizing this equation means finding a subspace for the appearance model where the minimum is at the position of the landmark and the response in the profile around it is similar to that of an inverted Gaussian. However, minimizing the equation by finding a suitable subspace B is not easy. Brunet et al. state that there is no closed form solution for the minimization. Therefore, they use the gradient descent method in order to minimize to above equation. The idea behind gradient descent is that a multivariate function $F(x)$ decreases fastest if we assume that we start at $F(x_0)$ and move x_0 towards the negative gradient of $F(x_0)$ as shown in Figure 5.6.

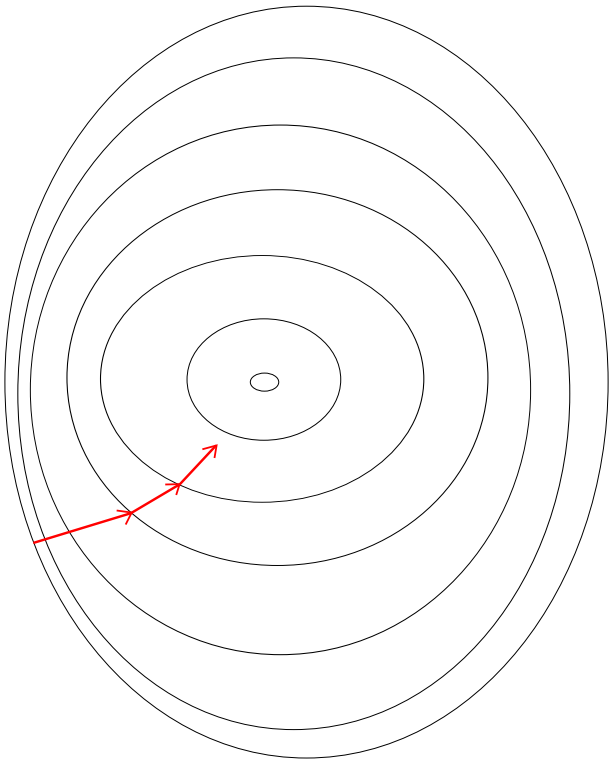


Figure 5.6: Illustration of gradient descent on contour lines of a multivariate function

Obviously $F(x)$ has to be defined and differentiable around x_0. In terms of the Equation 5.9 we can specify the gradient as follows:

$$G = \frac{\delta E}{\delta B} \tag{5.10}$$

The $n + 1$ iteration of the gradient descend method for the subspace B would thus be:

$$B^{(n+1)} = B^{(n)} - \eta G \tag{5.11}$$

For the translation and scaling factors o_i and α_i a closed form solution exists. The method proposed by Brunet et al. alternates between apply gradient descent to B and solving for o_i and α_i. After each iteration the new subspace basis B is orthonormalized, which means that each basis vector is normalized and made to be orthogonal to all other basis vectors.
Finding an optimal η in Equation 5.10 as stepsize can be done by apply line search, which minimizes an objective function. In this case

$$E(B - \eta G) \tag{5.12}$$

is used. Line search applies a similar idea to gradient descend, where the objective function is minimized by searching for a minimum along the descend direction.

Brunet et al. note that the minimization of Equation 5.9 is prone to converging to local minima. For this reason they start gradient descend from several initial points and then pick the solution with the smallest error. Furthermore, B is initialized with a PCA. By selecting landmarks with good profiles surrounding them and finding a subspace with good features for training the ASM, Brunet et al. show that they are able to achieve better results during fitting than the standard PCA approach. We will further discuss the results in Section 4.

3 Applications In Medical Image Processing

After having introduced ASMs above, we now want to present an application in the field of medical image processing. As an example we have picked *Automated 3-D Segmentation Of Lungs With Lung Cancer In CT Data* by Sun et al [5]. Since our main focus in this report is on ASMs we will not go into great detail concerning the method presented in this paper.

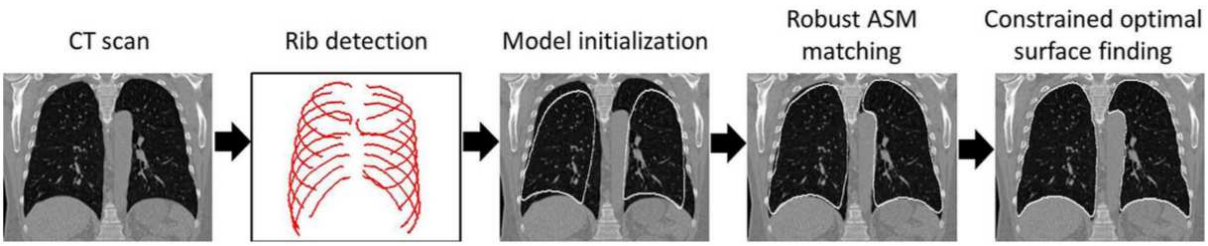


Figure 5.7: Pipeline of the approach [5]

The goal is to have a robust segmentation lungs with diseased tissue. The segmentation is achieved by using a ASM approach visualized in Figure 5.7. As a training set healthy lung CT scans are used. Since we have volumetric data we can compute 3D segmentation instead of a 2D one. The landmark points of each slice through the volume make up the points on the surface of each lung in the training set. It is then possible to extract a triangle mesh from the surface by using marching cubes. Next a point distribution model is build from the mesh points by using a similar method to the one described in Section 2.1.
In order to have a good initial position of the ASM during fitting, Sun et al. propose to use the rib cage as a point of reference. The rib cage itself is found by examining the density values of the volumetric data. The standard ASM fitting approach would be to search along the surface normal at the landmark positions and evaluate some cost function which considers the density values along the normal. This process would be iterated until convergence as explained in Section 2.1.
However, Sun et al. point out that the density of diseased lung tissue is much higher than that of normal lung tissue. This causes the fitting to converge to a solution where to updated landmark points do not lie on the surface but at the transition of the diseased and the normal lung tissue. For this reason they propose to train a model which recognizes normal lung shape patterns. The model can then be used during fitting to identify outliers and reject solutions via a voting mechanism.
Finally, a smooth surface is found by using a maximum flow algorithm where the graph edges reflect the properties of the volumetric data.

3.1 Results

Examples of the results achieved with this method are shown in Figure 5.8.

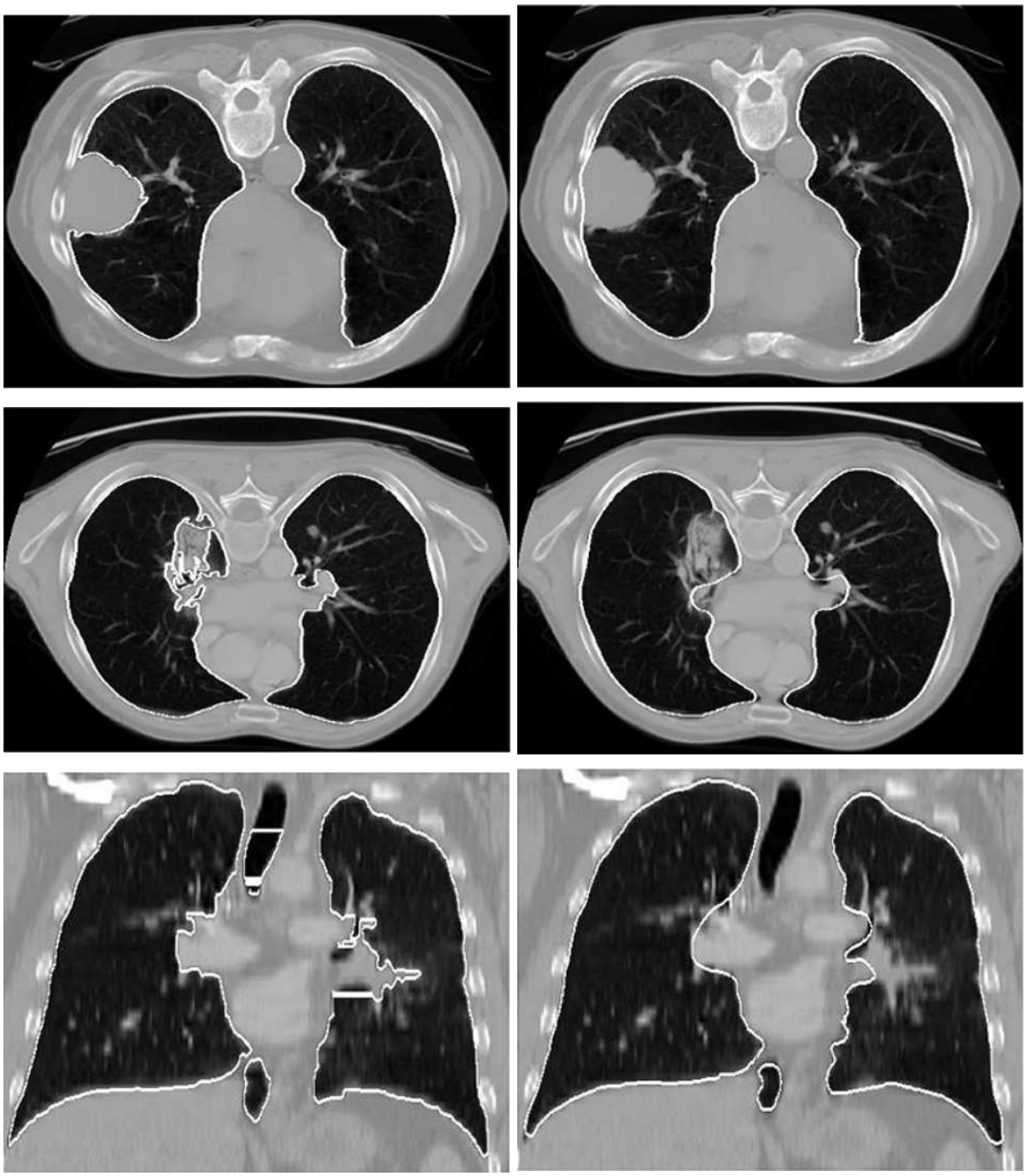


Figure 5.8: The images show different slices through CT data. The segmentation in the left column shows the manually done defined reference, while the right column show examples of the segmentation results for the same set of images [5]

Sun et al. note that they managed to produce robust segmentation of both normal and diseased lungs, while the standard ASM method, which produces a least squares optimization during fitting has problems with non-normal lung structures, since it is sensitive to outliers.

(a) Standard method [5] (b) Proposed method [5]

Figure 5.9: Differences in the results show that in this case the standard method causes the segmentation to leak into the gas filled colon, while the proposed robust ASM method still performs well [5]

4 Discussion

Since we mostly focused on the proposed approaches to ASMs by van Ginneken et al. [6] and Brunet et al. [1] in Section 2.2, we will now discuss possible shortcomings and results in this Section.

4.1 Active Shape Model Segmentation With Optimal Features [6]

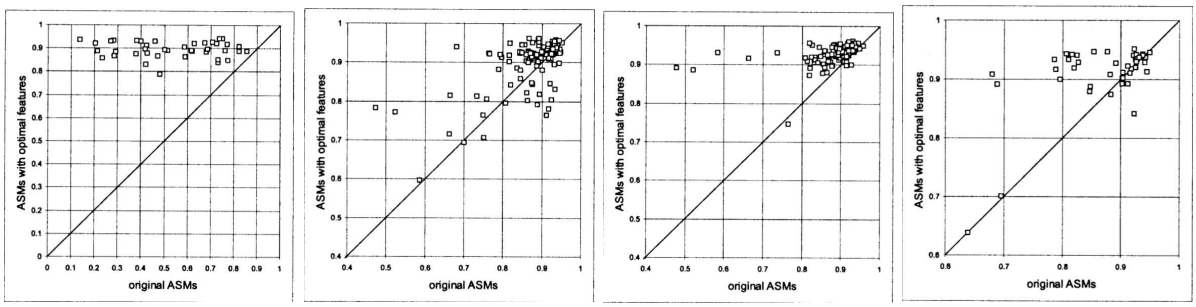
While van Ginneken et al. state that it is possible to use any other classifier for their method, the reasons for the choice of the kNN-classifier are not made clear. Furthermore, selecting an optimal k is not easy. Van Ginneken et al. do not specify why they chose $k = 5$ in their paper. It would have been interesting to see the results of their method with different k and how they compare to the presented results.
We assumed that a weighting with $exp(-d^2)$ was chosen for the votes in the kNN-classifier in order to handle the "*curse of dimensionality*" when comparing 60-dimensional feature vectors. As explained in Section 2.1.1, d is the Euclidean distance between two feature vectors. Van Ginneken et al. do not address why their weighting is constructed in the way it is. A high number of dimensions makes it harder to define neighborhood in the traditional manner since Euclidean distances grow more and more alike. This is due to the fact that the differences in each dimension sum up over the total number of considered dimensions. Using a different distance function or an approach that explicitly take the high number of dimensions into account for the nearest neighbor search might improve the quality of the results.

4.1.1 Results We will now present some of the results by van Ginneken et al. In order to evaluate their method for segmentation they measured the overlap:

$$\Omega = \frac{TP}{TP + FP + FN} \tag{5.13}$$

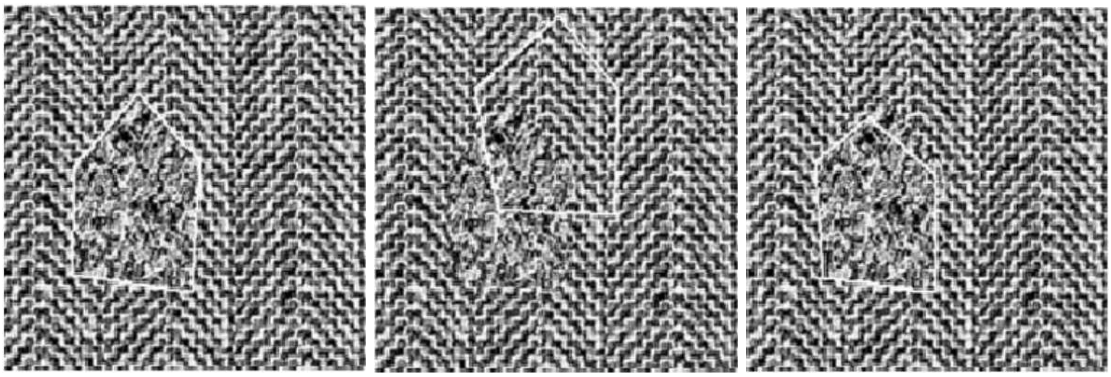
TP stands for *true positives*, FP for *false positives* and FN for *false negatives*. A perfect result would thus be $\Omega = 1$. The experiments where done by selecting half of the respective image set randomly for training and the other half for validation.
As visible in Figure 5.10, van Ginneken et al. mainly compared the original ASM method to their method with "optimal" features. Figures 5.11 and 5.12 show examples of segmentation results on different images. Figure 5.11(a) and 5.12(a) are manually defined ground truths.

We can see that their method generally outperforms the standard ASM method in almost all cases. By choosing "optimal" features the approach proposed by van Ginneken et al. handle different types of texture better.



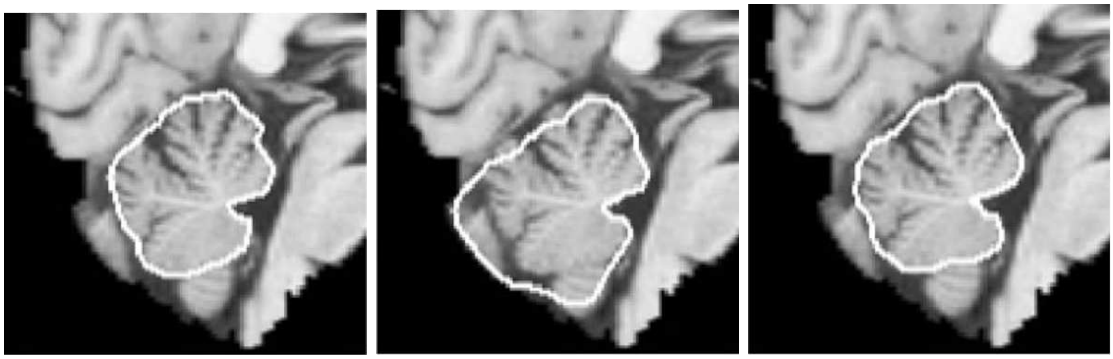
(a) House (b) Left Lung Field (c) Right Lung Field (d) Cerebellum

Figure 5.10: Scatter plots for segmentation experiments. The overlap measure Ω for the original ASM method (x-axis) is plotted against Ω for the ASM method with optimal features (y-axis) [6]



(a) True Shape (b) ASM (c) Optimal ASM

Figure 5.11: Example results for the segmentation of generated textured images with a house shape [6]



(a) True Shape (b) ASM (c) Optimal ASM

Figure 5.12: Example results for the segmentation of MRI images of a cerebellum [6]

4.2 Learning Good Features for Active Shape Models [1]

Brunet et al. clearly state the problem of local minima within the context of good features for ASMs. Their approach to find a good subspace with gradient descent and line search that minimizes the Equation 5.9 however is also prone to local minima as Burnet et al. note. They offer a solution to this problem

by initializing the subspace B with a PCA solution and starting gradient descend from different random points. However this method is not further evaluated. How does the number of different starting points correlate with the quality of the subspace? It would also be interesting to see how other methods like the Newton method for optimization, which converges faster than gradient descend compares in terms of quality and performance. It has to be noted however that the problem of global optimization is very difficult to solve.

4.2.1 Results Brunet et al. did a set of experiments in detecting facial features in order to evaluate their method of selecting good profiles. They compared the mean squared error of an ASM built with 68 landmarks to one built with 34 landmarks. Both ASMs were constructed using their method of selecting the best regions as seen in Section 2.2.4. The ASM with 34 landmarks was built by choosing the best profiles for each part of the face as shown in Figure 4.2.1.
An example of the fitting results with the two ASMs can be seen in Figure 5.14. As visible in Figure 5.15, Brunet et al. managed to half the landmarks and still achieve similar results, which allows an boost in performance without loss of quality.

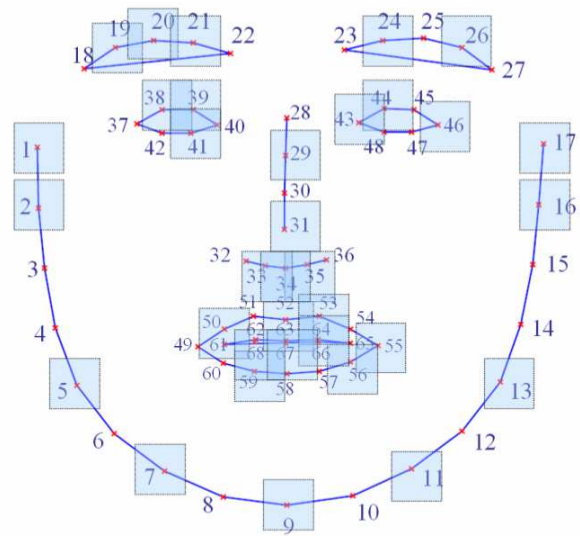
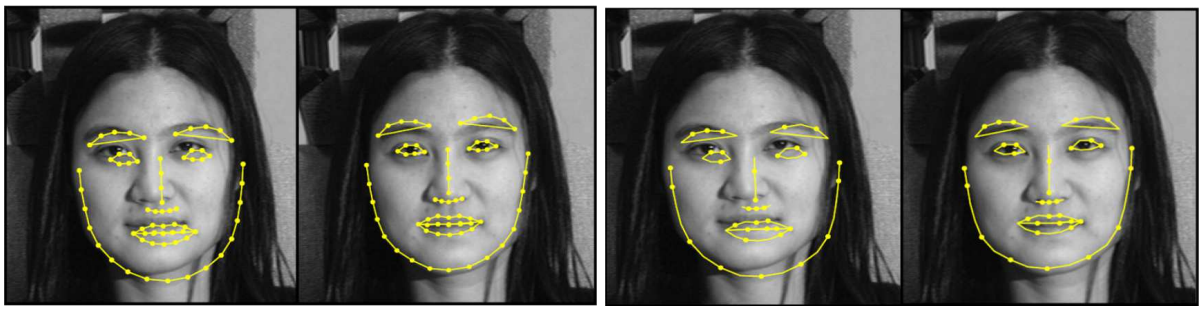


Figure 5.13: Best profiles for each part of the face [1]



(a) 68 landmarks (b) 34 landmarks

Figure 5.14: Example of facial feature detection. On the left in 5.14(b) and 5.14(a) is the initial ASM configuration and on the right the final result is shown [1]

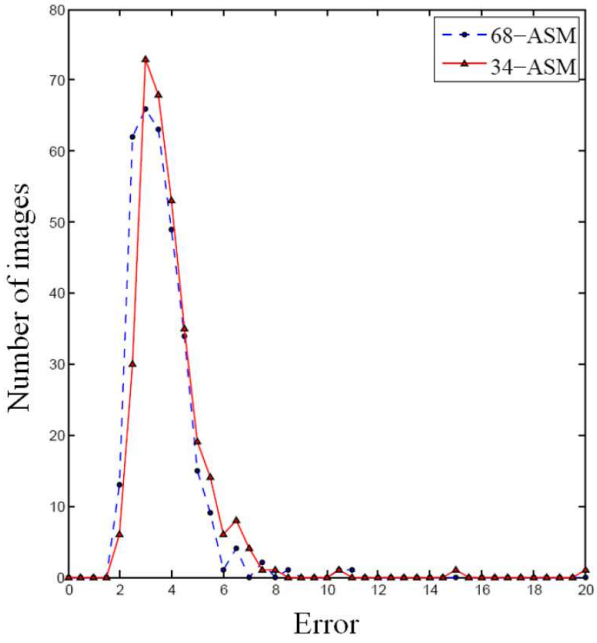


Figure 5.15: Error histogram of the ASM with 68 landmarks and the ASM with 34 landmarks [1]

An illustration of the improvement of learning good subspaces for profiles is shown in Figure 4.2.1. Here the error surface of two profiles from Figure 4.2.1 are shown. On the left the error surface resulting from standard PCA is visible, while on the right the error surface corresponding to the learned subspace optimizing Equation 5.9 is shown.
Brunet et al. also compared the quality of their method in comparison to using a PCA model. 200 images were used to train ASMs with 68 landmarks. 100 image were then used to test the quality of detection. The results are shown in Figure 4.2.1. As can be seen, the method by Brunet et al. outperforms the PCA model in terms of quality.

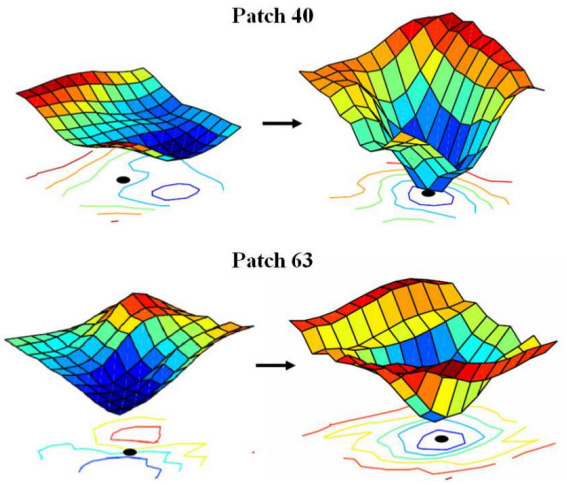


Figure 5.16: Error surface before and after learning good features [1]

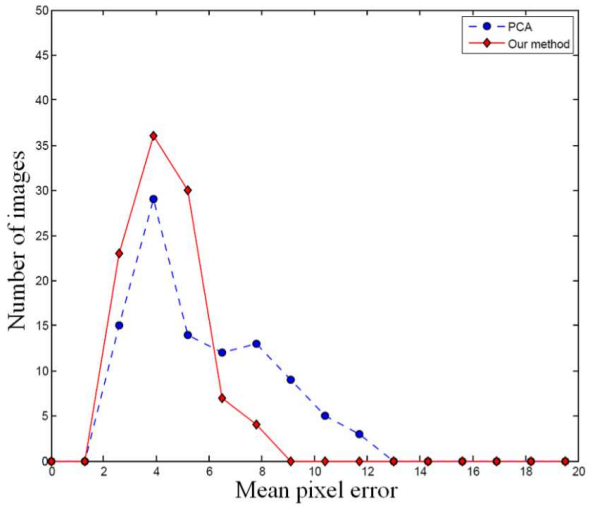


Figure 5.17: Error distribution comparing PCA and the method proposed by Brunet et al. [1]

5 Conclusion

In this report we introduced Active Shape Models and their extensions. We also discussed their application in the context of lung segmentation. While the standard method does not perform well in all cases, the general idea is robust. This is due to the fact, that it relatively easy to work with the existing standard model and extend it as needed.
As shown in Section 3, ASM-based techniques are a valid solution to provide segmentation in the medical field. Furthermore, the current results with regards to learning which features should be tracked presented in Section 4.2 are very promising. With additional advancements in computer vision and machine learning applied to the medical field, patients and physicians are bound to benefit even further from software-aided image analysis in the near future.

References

[1] N. Brunet, F. Perez, and F. De la Torre. Learning good features for active shape models. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 206–211. IEEE, 2009.

[2] T.F. Cootes, C.J. Taylor, D.H. Cooper, J. Graham, et al. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[3] T.F. Cootes, C.J. Taylor, et al. Statistical models of appearance for computer vision. *World Wide Web Publication, February*, 2001.

[4] P. Somol, P. Pudil, J. Novovičová, and P. Paclık. Adaptive floating search methods in feature selection. *Pattern recognition letters*, 20(11):1157–1163, 1999.

[5] S. Sun, C. Bauer, and R. Beichel. Automated 3-d segmentation of lungs with lung cancer in ct data using a novel robust active shape model approach. *Medical Imaging, IEEE Transactions on*, 31(2):449–460, 2012.

[6] B. Van Ginneken, A.F. Frangi, J.J. Staal, B.M. ter Haar Romeny, and M.A. Viergever. Active shape model segmentation with optimal features. *Medical Imaging, IEEE Transactions on*, 21(8):924–933, 2002.

A state of the art review of MRI based automatic brain tumor diagnostic approaches

Smieschek Manfred

Contents

1 Introduction

In the year 2008 there were about 12.7 million new cancer cases and 7.6 million cancer deaths worldwide [1]. Cancer is the second leading cause of death after cardiovascular diseases in more developed countries [2, 3, 4]. At a first diagnosis often only a neoplasm is found which is commonly called a tumour. General speaking a tumour describes an abnormal growth of cells due to abnormal and uncontrolled cell division. A tumour can be benign, pre-malignant or malignant, depending on growth, differentiation to neighbour cells, cell ratio and metastasis. Depending on the location benign tumours can also cause death, for instance every brain tumour is life-threatening because of its invasive character in the limited space of the human skull, but in general malignant tumours are far more lethal. In every case it is important to distinguish between benign and malignant tumours as it massively influences the choice of therapy. Today most commonly a surgical biopsy or resection and histopathologic analysis decides about the kind of tumour, but this invasive procedure carries in itself the danger to damage the affected organ and in the case of a brain tumour is often not even possible. Additionally a biopsy takes some time and is costly as it is performed by experts. Therefore it would be beneficial to categorize tumours and brain tumours in particular with the help of a non-invasive method. A computer-assisted classification of brain tumours would have a lot of advantages: It would expedite the diagnosis, provide objective and reproducible results and at best avoid invasive procedures by medical imaging [5].

There are several different medical imaging methods: positron emission tomography (PET), X-ray computed tomography (CT) and magnetic resonance imaging (MRI).

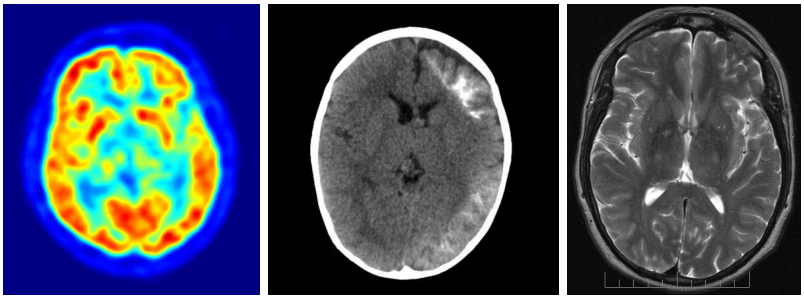


Figure 6.1: From left to right: Example of one PET, CT and MRI of a human brain. [6, 7, 8]

Figure 6.1 illustrates the resulting image of a PET, CT and MRI scan of the human brain. For brain tumour classification mainly MRI is used, because it provides superior soft-tissue contrast compared to CT or PET [9]. The general procedure in computer-assisted classification is to first preprocess and enhance the images. The second step is to determine the regions of interest (ROI), extract the features of these regions, afterwards select the most important features and classify the tumour [9].

The following chapter gives an overview over the different types of brain tumours and a short introduction into medical imaging using MRI. The chapter thereafter describes the state of the art of automatic brain tumour categorization and concludes the different results and classification accuracies. The last chapter gives an outlook on combined approaches and improvements.

2 Basics

2.1 Brain Tumour Classification

As mentioned in the previous chapter there are in general three different kinds of tumours: benign, pre-malignant and malignant. Ordinarily a benign tumour growth slowly, has a low cell ratio and therefore can be distinguished to its neighbouring tissue. Whereas a malignant tumour, commonly called cancer, infiltrates neighbouring tissue, grows rapidly, has a high mutation rate and metastasises into other organs. Malignant tumours are further classified depending on location, type of cell or grade. All classes together decide about the choice of therapy, for example if radiation and chemotherapy are necessary. In the case of a brain tumour additionally it is considered, if the tumour originated in the brain itself, a primary brain tumour, or involved the brain as a metastatic site, a secondary brain tumour. Most of the brain tumours arise from glial cells and are than called glioma.

2.1.1 Classification by cell The World Health Organisation distinguishes between about 98 different tumours of the central nervous system and 34 of the peripheral nervous system [10]. The tumours of the peripheral nervous system include tumours of cranial and paraspinal nerves, lymphomas and haematopoietic neoplasms, germ cell tumours, tumours of the sellar region and metastatic tumours. The two classes of tumours of the central nervous system are tumours of neuroepithelial tissue and tumours of the meninges. The papers presented in the next chapter solely focus on the main categories of these two classes, therefore the corresponding sub categories are not listed in table 1.

- Tumours of neuroepithelial
  - Astrocytic tumours
  - Oligodendroglial tumours
  - Oligoastrocytic tumours
  - Ependymal tumours
  - Choroid plexus tumours
  - Other neuroepithelial tumours
- Tumours of the meninges
  - Tumours of meningothelial cells
  - Mesenchymal tumours
  - Neuronal and mixed neuronal-glial tumours
  - Tumours of the pineal region
  - Embryonal tumours

Table 1: Tumour classes of the central nervous system according to the World Health Organisation [10].

Neuroepithelial cells are the stem cells of the nervous system and differentiate further into other cells of the nervous system, whereas the meninges is the system of membranes which envelops the central nervous system.

2.1.2 Classification by location With regard to location three different classes of brain tumours are distinguished. Supratentorial tumours which are above the tentorium in the cerebrum and infratentorial tumours which are below the tentorium in the cerebellum, see left figure 6.2.

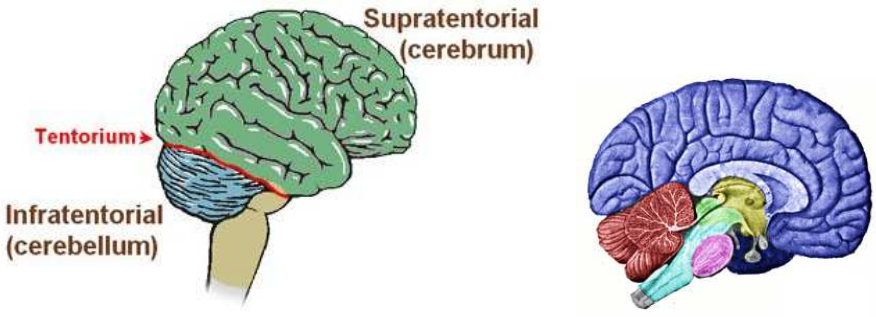


Figure 6.2: Schematics of the human brain [11]

The third class are pontine tumours which are located in the pons of the brain. The pons is marked pink on the right figure 6.2 and controls critical vital functions. Although the location of the tumour plays an important role in terms of a therapy and surgery it is not used in automated brain tumour classification explicitly.

2.1.3 Classification by grade The grade system from I to IV is also used by the World Health Organisation [10]. On the one hand it describes the current state of the tumour, and on the other hand it gives a prognosis of the behaviour of the neoplasm and predicts a response to therapy and outcome. It also reflects how much the tumour cells differ from the cells of the normal tissue they have originated from. Grade I applies to small tumours with the possibility of cure after a surgical resection alone. Whereas grade II tumours are more active, they tend to spread more and recur after a surgical resection. There is also the possibility that they progress to higher grades like grade III, which are evidently malignant and patients require radiation and chemotherapy. The last grade IV is associated with rapid pre- and postoperative disease evolution and a fatal outcome. Most patients succumb to grade IV tumours within a year as they infiltrate surrounding tissue massively, whereas patients with grade II tumours typically survive more than five years.

2.2 Medical Imaging

As already mentioned in the introduction there are several different medical imaging methods: positron emission tomography (PET), X-ray computed tomography (CT) and magnetic resonance imaging (MRI). But for brain tumour classification mainly MRI is used as it provides superior soft-tissue contrast compared to CT or PET [9]. A basic introduction of MRI is given in the next section. The section thereafter revolves around the magnetic resonance spectroscopy (MRS) as this technique is partly used in newer brain tumour classification approaches. Additionally a magnetic resonance tomography can yield information about the cerebral blood flow (CBF) and the cerebral blood volume (CBV). These procedures are described in the last section.

2.2.1 Magnetic Resonance Imaging Describing the underlying physics of MRI would go far beyond the scope of this term paper, therefore only a short transcription is given. The human body consists of about 70% water and each water molecule has two hydrogen nuclei, which continuously spin. This spinning can align parallel or anti-parallel to an outer magnetic field, where the latter occurs six less in a million [12]. A Radio Frequency (RF) impulse with the right frequency can arbitrarily direct the spin of these protons. The RF excitement is followed by exponential relaxation, where the protons return to the thermodynamic equilibrium [12]. During this relaxation, the stored energy is released as a radio frequency signal, which can be measured with receiver coils. In the resulting image different signal strengths and relaxation times can be illustrated by a corresponding luminance as shown in figure 6.3.

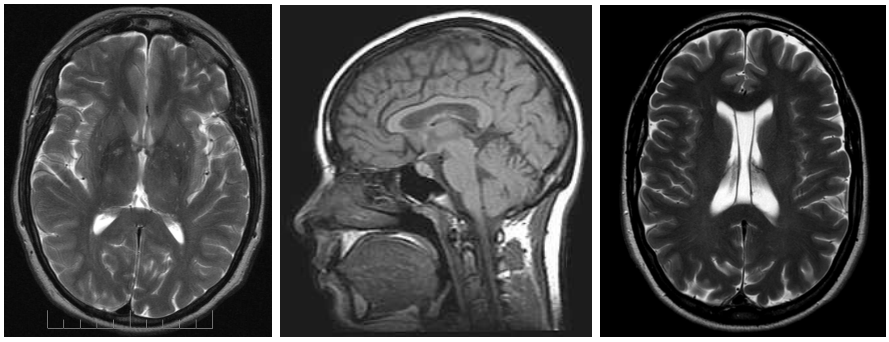


Figure 6.3: Examples of human brain MRI [8, 13, 14]

For a reliable and high-quality relaxation signal, several spin-echo sequences with different RF impulses are applied. An important parameter of these sequences is the echo time, which is half of the delay between a 90° and 180° RF impulse. According to García-Gómez et al. an echo time of 45ms and less is considered short and otherwise long. Usual short-echo time (SET) and long-echo time (LET) are between 20-35ms and about 135ms, respectively [15].

2.2.2 Magnetic Resonance Spectroscopy With the help of a MRI scanner it is not only possible to generate images of the brain as described in the previous chapter, but also possible to analyse the chemical shift of molecules. This procedure is called nuclear magnetic resonance spectroscopy (MRS) and gives a clue about the structure of the molecules [16].

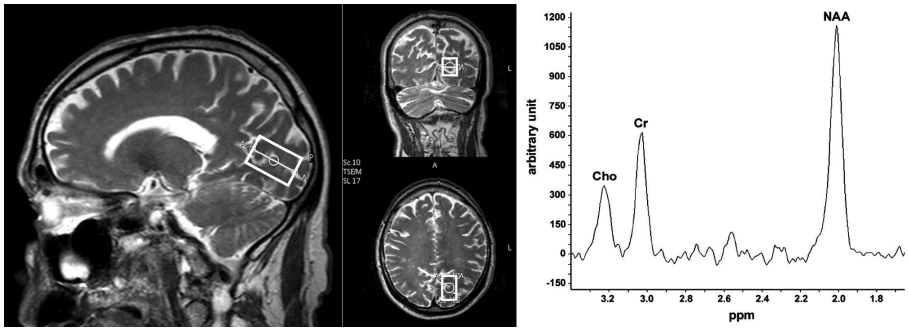


Figure 6.4: Example output: H Chemical Shift [17]

Figure 6.4 shows the MRI on the left and the corresponding MRS of the marked area on the right. Knowing about the existing metabolites in the human brain it is possible to exactly name the metabolites corresponding to the peaks in the graph. The marked peaks are from left to right: Choline (Cho) which is a measure of increased cellular turnover, so that it is increased in the region of a malignant tumour. Creatine (Cr) which is part of the energy metabolism and N-acetyl aspartate (NAA) which is a neuronal marker. Both are decreased by any disease that affects the neuronal integrity [18].

2.2.3 Perfusion There are basically two ways to measure the perfusion of the brain with the help of a MRI scanner. The first one is based on a contrast agent that changes the magnetic susceptibility of blood. The cerebral blood flow (CBF) is determined by rapid MRI of the passage of a contrast agent bolus [19]. The second method is based on arterial spin labelling (ASL), where arterial blood is magnetically tagged and measured [20]. It is also possible to measure the perfusion of a ROI, the so called regional CBF (rCBF).

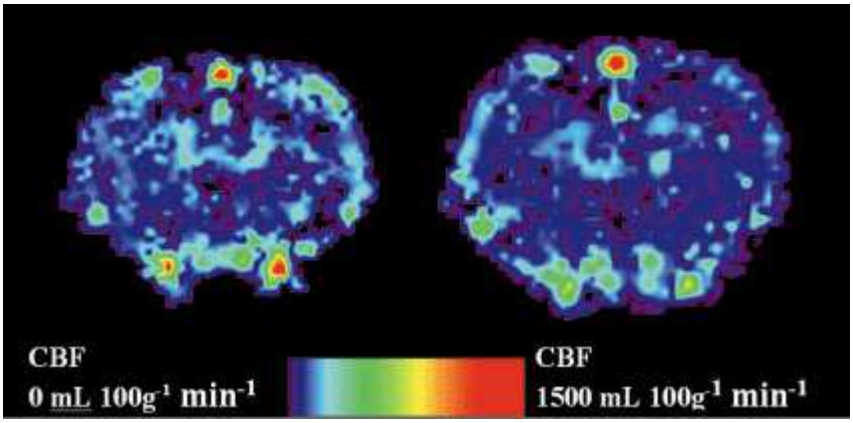


Figure 6.5: CBV and CBF of two healthy rat brains [21]

As the CBF is equal to the cerebral perfusion pressure (CPP) divided by the cerebrovascular resistance (CVR) its change is observed in brain tumours, stroke or ischemia [21]. From the measurements of the CBF also the cerebral blood volume (CBV) and rCBV can be determined. Figure 6.5 illustrates the perfusion of two healthy rat brains.

2.3 Preprocessing and Enhancement

The goal of preprocessing is to remove or minimize noise and artefacts caused by operator performance, equipment and the environment. For MRIs a standard imaging protocol is in place, which is an extension to the traditional fuzzy c-means (FCM) clustering algorithm [22]. But there have been several other approaches like Fourier and wavelet transformations, histogram based ones and component analysis [9].

After removing noise and artefacts several filters can be used to enhance the contrast and improve the visual appearance of the image. The general contrast can be enhanced for example by a median filter and the image boundaries with the help of a Gaussian filter [23, 24]. Having different image sources like short and long-echo time also a co-registration step is required [5].

2.4 Feature Extraction and Classification

MRI itself only provides images in which features are just implicitly present. There are several different approaches to gather all features from the images. Zacharaki et al. considered the shape of tumours like circularity, irregularity, rectangularity, the entropy of radial length distribution of the boundary voxels and the surface-to-volume ratio [5]. Also statistical characteristics of tumours like necrotic volume versus total tumour volume and mean and variance of image intensities were regarded. Finally a Gabor filter was used to average inside selected ROIs. Overall in that way a number of 161 features was gathered [5]. Another feature that is also often considered is relative cerebral blood volume [25]. The INTERPRET project, a brain tumour database, not only provides MRI images for several different brain tumours of about 900 patients, but also delivers features belonging to MRI images [26]. They mainly consist of MRS data, which as previously explained give information about the amount of specific molecules as frequency intensity values, measured in parts per million. That way a vector of 190 points was gathered for SET as well as LET.

To distinguish different sub types of tumours not all feature are equally useful, therefore not always all features are taken into account, but only the most discriminative ones. There are many approaches on how to determine these features: García-Gómez et al. and Luts et al. used ReliefF, an extension to Relief which can deal with noisy and incomplete data sets. For each instance of the data set it takes two nearest neighbours, one from the same and one from a different class. The features are than ranked according to their quality. A high quality feature shall differentiate between instances from different classes and have the same value for instances from the same class [15, 27]. Navarro et al. used the Entropic Filtering Algorithm (EFA), an extension to Mutual Information (MI) to measure the mutual dependence between a set of features. In contrast to the previously mentioned algorithms, EFA does not regard the relevance of each feature separately, but calculates the relevance of a whole set of features. The relevance is in that case a ratio comparing the discriminative power of the subset to the whole set of features. That way a minimal set with a certain threshold relevance can be found [16]. Zacharaki et al. used a two-tailed t-test, which measures the significance of a difference of means between two distributions, and therefore evaluates the discriminative power of each individual feature in separating two classes [5]. They also used constrained linear discriminant analysis (CLDA), which generally works like LDA, but without transforming the original features. Luts et al. used Fisher discriminant criterion, which takes the mean and the within-class scatter into account to compare the correlation between features and classes. The scoring obtained that way is than used to rank the features [27]. Additionally they used the Kruskal-Wallis test, whose null hypothesis is that the medians of two groups are equal. The results are compared with the χ^2 statistic with one degree of freedom less than the Kruskal-Wallis test. Afterwards the features are ranked according to their χ^2 value [27]. There is also the possibility to use a combination of filters and wrappers to determine a suitable subset of features [28].

There are also a few different approaches for binary classification: The most common are support vector machines (SVM), they construct a hyperplane or a set of hyperplanes to divide the feature space into regions, which belong to one class only. Unknown data points are than simply classified by the region they belong to [27, 5, 15, 29]. Another common way to classify are decision trees. The leaves of the tree represent different classes and the branches represent conjunctions of features. C4.5 and random forest are algorithms to automatically generate such trees according to given features [16]. Cruz-Barbosa and Vellido also used SS-GTM and SS-Geo-GTM, which are extensions to generative topographic mapping (GTM), a manifold learning model. The GTM is defined as a mapping from the low-dimensional latent space onto the observed data space. Such mapping is carried through by a set of basis functions generating a constrained mixture density distribution [29].

3 State of the art

3.1 García-Gómez et al.

In 2008 García-Gómez et al. compared the influence of different MRI data on the binary classification accuracy [15]. As data set they used 185 samples from the INTERPRET database [26]. For all cases

there was exactly one short-echo time MRI (SET) and one long-echo time MRI (LET). A combined data set was formed by direct concatenation of all features from both echo times. They grouped the cases to 105 agressive (AG) with 77 Glioblastoma multiforme (GM) and 28 metastatsis (MET), 50 low grade meningioma (MN) and 30 low-grade glial mixture (LG) with 20 astrocytoma grade II (A2), 5 oligodendroglioma (OD) and 5 oligoastrocytoma (OA). They used ReliefF as a feature selection algorithm and for classification a Least Squares Support Vector Machine (LS-SVMs) with Radial Basis Function (RBF) kernel. For evaluation 70% of the data was used as training set and k-Random Sampling Train-Test (kRSTT) was used with 150 repetitions. Table 2 shows, that García-Gómez et al. achieved the highest accuracy in all three classification problems by using the combined data set of short and long echo time.

Data	AG vs. MN	AG vs. LG	LG vs. MN
SET	92.62 [87.79,95.95]	92.07 [86.71,95.74]	96.02 [90.27,98.70]
LET	92.22 [87.30,95.66]	90.49 [84.77,94.58]	94.54 [88.15,97.96]
both	95.27 [91.16,97.79]	92.58 [87.34,96.11]	97.53 [92.59,99.29]

Table 2: kRSTT evaluation of the LS-SVM on data with different echo times according to García-Gómez et al. [15] Classification accuracy with confidence intervals in %.

3.2 Cruz-Barbosa and Vellido

In 2011 Cruz-Barbosa and Vellido introduced a semi-supervised analysis of human brain tumours [29]. They used 22 normal volunteers and a subset of 217 cases from the INTERPRET project [26]. For all 217 cases there were MRI with short-echo time, they include 58 meningiomas (MN), 86 glioblastomas (GM), 38 metastases (MET), 22 astrocytomas grade II (A2), 6 oligoastrocytomas grade II (OA), and 7 oligoden-drogliomas grade II (OD). And for 195 of the 217 cases there are MRI taken with long-echo time (LET), they include 55MM, 78GM, 31MET, 20A2, 6OA, and 5OD. Afterwards a third set consisting of 195 items was built, also by combination of the spectra measured at both echo times. For the control group of 22 volunteers 22 SET and 15 LET MRI were taken with no tumour tissue (NT). The different tumour types (T) were grouped according to the WHO into: Low grade gliomas (LG) with A2, OA and OD, high grade malignant (AG) with ME and GL, and meningiomas (MN). The goal of this study was to compare which echo time is most suitable for classification, similar to García-Gómez et al., and if there is any information gain by combining both echo times. But also another classification problem was added with no tumour tissue and additionally four different binary classifiers were used: SS-Geo-GTM, SS-GTM, LapEM and SS-SVMan. Where SS-Geo-GTM and SS-GTM are extension of GTM, a manifold learning model of the statistical machine learning family, LapEM being the Laplacian Eigenmap and SS-SVMan a semi-supervised SVM for manifold learning. 10% of the data was used as a training set and in over 100 random classification runs the highest average accuracy was achieved with the combined data set and the SS-Geo-GTM algorithm as classifier.

The precise accuracies are shown in the following table 3:

Data	Classifier	LG vs. AG	LG vs. MN	AG vs. MN	T vs. NT
SET	SS-Geo	86.7±12.3	72.2±17.6	76.3±20.9	94.2±4.7
	SS-GTM	82.5±13.5	68.2±16.3	79.8±15.2	71.6±21.8
	LapEM	52.7±3.6	50.1±6.2	52.9±3.9	53.8±3.5
	SS-SVMan	84.7±16.9	60.6±15.8	67.9±17.1	91.0±24.3
LET	SS-Geo	72.5±16.8	68.9±17.5	73.1±16.1	48.7±24.7
	SS-GTM	72.4±13.9	67.0±17.5	72.6±13.5	55.7±28.1
	LapEM	53.8±3.4	53.5±5.2	53.3±4.3	51.4±2.9
	SS-SVMan	62.8±10.2	55.8±14.8	72.9±13.7	40.2±33.6
SET	SS-Geo	86.2±11.5	84.1±14.8	82.5±16.2	98.6±8.3
+	SS-GTM	86.1±15.1	82.2±13.8	81.2±15.6	98.6±3.0
LET	LapEM	52.0±3.5	53.1±4.3	53.3±3.5	51.4±9.1
	SS-SVMan	81.2±18.9	73.8±17.2	80.1±14.3	100.0±0.0

Table 3: Classification accuracy in % for different data and models according to Cruz-Barbosa [29]

3.3 Zacharaki et al.

In 2009 Zacharaki et al. used different feature selection and binary classification methods to classify human brain tumours [5]. The MRI were taken in short and long-echo time and also rCBV maps were generated. The methods were applied on a population of 98 patients with 102 histologically diagnosed brain tumours. The biopsies took place between September 2006 and December 2007 and among the 98 patients which were examined were 52 women and 46 men, all within the age of 17 and 83 years. They grouped the tumours into five groups:

- 24 metastasis (MET)
- 4 meningiomas (MN)
- 22 gliomas grade II (GL2) including astrocytomas, gliomatosis cerebri, oligodendrogliomas, oligoastrocytomas, ependymomas
- 18 gliomas grade III (GL3) including anaplastic astrocytomas and (anaplastic) oligodendrogliomas
- 34 glioblastomas (GBMs) (GL4) including 1 giant cell GBM

Two different feature selection approaches where used: on the one hand a two-tailed t-test (t-test) which evaluates the discriminative power of each individual feature in separating two classes, and on the other hand constrained linear discriminant analysis (CLDA) which tries to maximize the discriminant capability between classes. As binary classifier LDA with Fisher's Discriminant Rule, k-Nearest Neighbour and a nonlinear SVM with Gaussian Radial Basis function kernel were used. As table 4 shows the highest mean accuracy was achieved with SVM as classifier and t-test as feature selection.

Classifier	LDA		k-NN (k=3)		SVM	
Feature Ranking	t-test	CLDA	t-test	CLDA	t-test	CLDA
MET-MN	92.9	85.7	96.4	89.3	96.4	85.7
MET-GL2	95.7	84.8	97.8	97.8	97.8	95.7
MET-GL3	81.0	83.3	90.5	88.1	88.1	88.1
MET-GL4	58.6	77.6	74.1	79.3	84.5	89.7
MN-GL2	100.0	100.0	96.2	96.2	96.2	96.2
MN-GL3	81.8	95.5	95.5	90.9	86.4	90.9
MN-GL4	86.8	100.0	97.4	97.4	97.4	94.7
GL2-GL3	70.0	77.5	67.5	72.5	72.5	75.0
GL2-GL4	76.8	78.6	98.2	98.2	98.2	98.2
GL3-GL4	67.3	69.2	84.6	84.6	94.2	92.3
mean	81.1	85.2	89.8	89.4	91.2	90.6

Table 4: Seperate and mean classification accuracy in % for different feature ranking and classification methods according to Zacharaki et al. [5]

3.4 Luts et al.

In 2007 Luts et al. used LS-SVM with RBF kernel and combined MRI and MRSI data to classify brain tumours [27]. They used different feature selection methods: automatic relevance determination for Bayesian LS-SVMs (ARD), Fisher discriminant criterion (FC), Kruskal-Wallis test (K-W) and the ReliefF algorithm (R-F), to compare them with the well known linear discriminant analysis (LDA). In addition the results without feature ranking were added (BL). As a binary classifier always a LS-SVM with RBF kernel was used. And the data set was 50 times randomly split into two thirds for training and validation and one third for testing. As data set they also used the INTERPRET project database, but they only considered the SET spectra. Additionally they did not group the tumour classes together, but compared the classes directly. The data set includes 10 classes of pathologies:

- 8 normal brain tissue from volunteers and apparently normal tissue from the contralateral half of the brain of patients (NT)
- 8 cerebrospinal fluid (CSF)

- 5 grade II diffuse astrocytomas (A2)
- 2 grade II oligoastrocytomas (OA)
- 2 grade II oligodendrogliomas (OD)
- 2 grade III astrocytomas (A3)
- 1 grade III oligoastrocytomas (OA3)
- 2 grade III oligodendrogliomas (OD3),
- 3 meningiomas (MN)
- 7 grade IV gliomas (G4)

As table 5 shows the highest mean accuracy was achieved with unweighted features (BL). Considering only ranked features on average the automatic relevance determination (ARD) performed the best. Only a subset of all results of possible binary classification problems is given. Apparently all given accuracies are well above 95%.

Feature ranking	ARD	FC	K-W	R-F	BL	LDA
NT vs. A2	99.8	99.2	99.5	99.4	99.3	99.1
CSF vs. A2	98.7	98.2	98.2	98.5	99.1	97.8
A2 vs. OA	98.5	96.2	96.5	95.9	96.1	92.1
A2 vs. OD3	100.0	98.6	99.0	99.3	99.6	99.3
OA vs. MN	99.4	97.8	98.0	99.4	99.9	99.0
OD vs. OD3	99.2	98.7	98.7	98.7	99.5	97.6
A3 vs. OD3	97.4	98.0	97.5	98.9	99.7	97.4
OA3 vs. G4	99.4	99.1	98.5	99.6	99.6	98.1

Table 5: Average accuracy on test sets over 50 runs in % according to Luts et al. [27].

3.5 Navarro et al.

In 2008 Navarro et al. also evaluated the influence of feature reduction on brain tumour classification [16]. They used the long-echo time data set of the INTERPRET project [26]. The 195 cases in this study include: 55 MN, 78 GM, 31 MET, 20 A2, 6 OA and 5 OD. On the one hand the number of feature was reduced with the help of an entropic filtering algorithm (EFA), and on the other hand not reduced (NR) and Relief reduced data were used as a reference. As classifier nearest-neighbour technique with Euclidean metric (NN), the Naive Bayes classifier (NB), a C4.5 decision tree and a Random Forest (RF) were used. 70% of the data was used as a training set and the remaining 30% as a test set to evaluate the classifier. Table 6 shows the average classification accuracy and that the EFA algorithm selected features, which used by all four classifiers yield a higher accuracy compared to the Relief reduced data. On the other hand the not reduced features had the highest accuracy with the Naive Bayes and C4.5 algorithm. Overall the highest accuracy was achieved by the combination of EFA as feature ranking and NN as classifier.

	NN	NB	C4.5	RF
NR	85.0	90.5	83.0	83.0
EFA	94.3	86.8	79.2	85.0
Relief	71.7	60.4	62.2	71.7

Table 6: Average accuracy on the test set in % according to Navarro et al. [16].

3.6 Summary and Conclusion

The previous sections presented five state of the art studies concerning automatic brain tumour classification. Four out of five used MRI and MRS data from the INTERPRET project and consequently their extracted features. Zacharaki et al. used MRI data from own patients and developed a feature extraction process as described in Chapter 2.4. All papers tried to evaluate the influence of different data sets, feature selecting, ranking or weighting methods and different classification algorithms. Cruz-Barbosa et al. and García-Gómez et al. grouped the tumour classes according to the WHO. Although they used different label for the groups in their original papers, their results could be easily merged as the same tumours types were grouped together. Cruz-Barbosa et al. added one additional classification compared to García-Gómez et al., they added 22 normal tissue scans and therefore had one additional binary classification problem between normal tissue and tumours. Table 7 shows the highest accuracy results from both papers.

Paper	Cruz-Barbosa [29]	García-Gómez [15]
Data	217 INTERPRET + 22 patients	185 INTERPRET
Feature selection	-	ReliefF
Classifier	SS-Geo-GTM	LS-SVM (RBF)
LG vs. AG	86.2±11.5	92.58 [87.34,96.11]
LG vs. MN	84.1±14.8	97.53 [92.59,99.29]
AG vs. MN	82.5±16.2	95.27 [91.16,97.79]
T vs. NT	98.6±8.3	-

Table 7: Summary of different classification accuracies. All values are in percent and values in brackets are confidence intervals.

But although their results can be written together in one table, theoretically they are not comparable as both use different validation algorithms. García-Gómez et al. used kRSTT with 70% of data as training set over 150 repetitions, whereas Cruz-Barbosa et al. used 10% of data as training set and averaged the results over one hundred runs. For the other papers there are even more reasons, why they cannot be concerned in a summary. Zacharaki et al. examined own patients and the population they analysed was half as big as the INTERPRET database. Additionally they have chosen a different grouping, where group AG was split into two groups and compared to García-Gómez et al. the LG group contained additional tumour types. They also used a different algorithm for validation called leave-one-out cross-validation. Although Luts et al. and Navarro both used the INTERPRET database, their results cannot be compared for several other reasons. In the case of Luts et al. they achieved peculiar high accuracy percentages, but although they used the INTERPRET database only 40 tumours were regarded. Also they did not group the several tumour types together, but calculated the binary classification directly for their ten different tumour types. So that five classes consisted of only 2 or less samples, which is a far too little data set for reliable results. And in the case of Navarro only the average percentage over all binary classifications was given, so that the results are far to general. Summing up there are severe differences between the state of the art papers, so that their results cannot be compared, particular for the reasons mentioned above.

4 Outlook

The previous chapters showed that human brain tumours can be classified with an average accuracy of about 85-95%. One way to improve these results maybe a combination of different feature ranking algorithms and classification methods, which is generally known as ensemble learning. On the other hand the combination of short echo time and long echo time MRI already showed an increase in accuracy, therefore the next step might be an inclusion of additional information sources like MRS, CSF and rCBV. In the case of MRS, which shows metabolites and therefore reflects a specific biochemical process, the values of NAA, Cr and Cho are strong indicators if a tumour is benign or malign. Astrakas et al. combined the MRS with molecular genomics, which is generally speaking the study of DNA, mRNA sequences and proteins [28]. With these features they achieved an accuracy of above 95%, although their data set was very small and further studies in this direction are needed.

Another trend in the field of automated brain tumour classification is the extension from binary classifiers to multiclass classifiers. From the perspective of computer science this extension is the next

step to generally solve the problem of brain tumour classification, but on the one hand their accuracy is generally lower. And on the other hand from the medical perspective this extension is not absolutely necessary. The automatic brain tumour diagnosis is not supposed to work independently on its own, but is designed as a clinical decision support system (DSS). So that a doctor is always involved and has already an idea about the tumour, but might not be completely certain or might swing between two alternatives. Nowadays he would consult a colleague for a second opinion, but because this is not always possible and the second opinion is also not objective, a DSS could fulfil this role much more effective.

For further studies it is also important to emphasise the need to keep with the WHO classification of tumours and especially the WHO grouping of tumours. The results of different studies are only comparable in any way if this is given. Another weakness currently are different evaluation methods like Friedman test, k-fold cross-validation and k-Random Sampling Train-Test, whose accuracy results are technically not comparable. So there is also a need for a standardized evaluation method. It would also be very desirable that all studies work on the same data set, this is already partly the case, because most use the INTERPRET database. Additionally an expansion of this database would be beneficial for more significant results.

All in all the current binary classifier already give results with a very high accuracy. Nevertheless there is much room for improvement, especially considering that tumours are often grouped together and studies are executed on small populations. Automatic brain tumour classification cannot replace a doctor, and never was supposed to, but it already can help doctors to decide in cases of uncertainty.

References

[1] Ferlay J, Shin HR, Bray F, Forman D, Mathers C, Parkin DM. Estimates of worldwide burden of cancer in 2008: GLOBOCAN 2008. International journal of cancer. 2010;127(12):2893–2917.

[2] Statistisches Bundesamt. Gesundheit - Todesursachen in Deutschland; 2010. [Online; accessed 03-December-2012].

[3] Europäische Kommission. Öffentliche Gesundheit; 2010. [Online; accessed 03-December-2012].

[4] Hoyert DL, Xu J. Deaths: Preliminary Data for 2011. National Vital Statistics Reports. 2011;61(6):1.

[5] Zacharaki EI, Wang S, Chawla S, Soo Yoo D, Wolf R, Melhem ER, et al. Classification of brain tumor type and grade using MRI texture and shape in a machine learning scheme. Magnetic Resonance in Medicine. 2009;62(6):1609–1618.

[6] Wikipedia. Human Brain PET — Wikipedia, The Free Encyclopedia;. [Online; accessed 17-January-2013]. Available from: {http://commons.wikimedia.org/wiki/File:PET-image.jpg}.

[7] Wikipedia. Human Brain CT — Wikipedia, The Free Encyclopedia;. [Online; accessed 17-January-2013]. Available from: {http://commons.wikimedia.org/wiki/File:Sturge-Weber_CT.jpg}.

[8] Wikipedia. Human Brain MRI — Wikipedia, The Free Encyclopedia;. [Online; accessed 17-January-2013]. Available from: {http://commons.wikimedia.org/wiki/File:Cavum_septi_pellucidi_-_Cavum_vergae.jpg}.

[9] Selvanayaki K, Karnan DM. CAD System for Automatic Detection of Brain Tumor through Magnetic Resonance Image-A Review. International Journal of Engineering Science and Technology, 2 (10). 2010;p. 5890–5901.

[10] Louis DN, Ohgaki H, Wiestler OD, Cavenee WK, Burger PC, Jouvet A, et al. The 2007 WHO classification of tumours of the central nervous system. Acta neuropathologica. 2007;114(2):97–109.

[11] Wikipedia. Meninges — Wikipedia, The Free Encyclopedia; 2012. [Online; accessed 03-December-2012]. Available from: {http://en.wikipedia.org/wiki/Meninges}.

[12] Deserno TM. Fundamentals of Medical Image Processing. Springer Handbook of Medical Technology. 2012;p. 1139–1165.

[13] Wikipedia. Human Brain MRI — Wikipedia, The Free Encyclopedia;. [Online; accessed 17-January-2013]. Available from: {http://commons.wikimedia.org/wiki/File:MRI_brain.jpg}.

[14] Wikipedia. Human Brain MRI — Wikipedia, The Free Encyclopedia;. [Online; accessed 17-January-2013]. Available from: {http://commons.wikimedia.org/wiki/File:MRI_T2_Brain_axial_image.jpg}.

[15] García-Gómez JM, Tortajada S, Vidal C, Julià-Sapé M, Luts J, Moreno-Torres À, et al. The effect of combining two echo times in automatic brain tumor classification by MRS. NMR in Biomedicine. 2008;21(10):1112–1125.

[16] Navarro FFG, Munoz LAB. Feature Selection in Proton Magnetic Resonance Spectroscopy for Brain Tumor Classification. Procs of ESANN 2008. 2008;.

[17] Boucard CC, Hoogduin JM, van der Grond J, Cornelissen FW. Occipital Proton Magnetic Resonance Spectroscopy (¹H-MRS) Reveals Normal Metabolite Concentrations in Retinal Visual Field Defects. PLoS ONE. 2007 02;2(2):e222. Available from: http://dx.plos.org/10.1371/journal.pone.0000222.

[18] Ross B, Bluml S. Magnetic resonance spectroscopy of the human brain. The Anatomical Record. 2001;265(2):54–84.

[19] Østergaard L, Smith DF, Vestergaard-Poulsen P, Hansen SB, Gee AD, Gjedde A, et al. Absolute cerebral blood flow and blood volume measured by magnetic resonance imaging bolus tracking: comparison with positron emission tomography values. Journal of Cerebral Blood Flow & Metabolism. 1998;18(4):425–432.

[20] Alsop D, Detre JA, et al. Multisection cerebral blood flow MR imaging with continuous arterial spin labeling. RADIOLOGY-OAK BROOK IL-. 1998;208:410–416.

[21] Adam JF, Elleaume H, Le Duc G, Corde S, Charvet AM, Troprès I, et al. Absolute cerebral blood volume and blood flow measurements based on synchrotron radiation quantitative computed tomography. Journal of Cerebral Blood Flow & Metabolism. 2003;23(4):499–512.

[22] Shen S, Sandham W, Granat M, Sterr A. MRI fuzzy segmentation of brain tissue using neighborhood attraction with neural-network optimization. Information Technology in Biomedicine, IEEE Transactions on. 2005;9(3):459–467.

[23] Niessen W, Vincken K, Weickert J, Romeny BMTH, Viergever M. Multiscale segmentation of three-dimensional MR brain images. International Journal of Computer Vision. 1999;31(2):185–202.

[24] Metaxas DN, Qian Z, Huang X, Huang R, Chen T, Axel L. Hybrid deformable models for medical segmentation and registration. In: Control, Automation, Robotics and Vision, 2006. ICARCV'06. 9th International Conference on. IEEE; 2006. p. 1–6.

[25] Lev MH, Ozsunar Y, Henson JW, Rasheed AA, Barest GD, Harsh GR, et al. Glial Tumor Grading and Outcome Prediction Using Dynamic Spin-Echo MR Susceptibility Mapping Compared with Conventional Contrast-Enhanced MR: Confounding Effect of Elevated rCBV of Oligodendroglimoas. American Journal of Neuroradiology. 2004;25(2):214–221.

[26] International network for pattern recognition of tumours using magnetic resonance;. [Online; accessed 27-December-2012]. Available from: {http://carbon.uab.es/INTERPRET/}.

[27] Luts J, Heerschap A, Suykens JAK, Van Huffel S. A combined MRI and MRSI based multiclass system for brain tumour recognition using LS-SVMs with class probabilities and feature selection. Artificial Intelligence in Medicine. 2007;40(2):87–102.

[28] Astrakas L, Blekas KD, Constantinou C, Andronesi OC, Mindrinos MN, Likas AC, et al. Combining magnetic resonance spectroscopy and molecular genomics offers better accuracy in brain tumor typing and prediction of survival than either methodology alone. International journal of oncology. 2011;38(4):1113.

[29] Cruz-Barbosa R, Vellido A. Semi-supervised analysis of human brain tumours from partially labeled MRS information, using manifold learning models. International Journal of Neural Systems. 2011;21(01):17–29.

[30] Amin SE, Megeed M. Brain tumor diagnosis systems based on artificial neural networks and segmentation using MRI. In: Informatics and Systems (INFOS), 2012 8th International Conference on. IEEE; 2012. p. MM–119.

[31] Kononenko I. Estimating attributes: analysis and extensions of RELIEF. In: Machine Learning: ECML-94. Springer; 1994. p. 171–182.

Seminar Medical Image Processing
Affine Image Matching Is Uniform TC^0-Complete

Michael Buse

Inhalt

Zusammenfassung

Es konnte nachgewiesen werden, dass Affine Image Matching mit polynomieller Laufzeit gelöst werden kann. Nun wurde gezeigt, dass Affine Image Matching in der gleichen Komplexitätsklasse (TC^0) liegt wie die Berechnung einer Summe. Welchen theoretischen Einfluss dies auf die Medizinische Bildverarbeitung hat wird hier erläutert.

Keywords: Affine Image Matching, Circuit Complexity, Bildfusion, uniform TC^0-complete

1 Einleitung

Dem Paper Affine Image Matching Is Uniform TC^0-Complete[1] sind zwei Arbeiten [2, 3] voraus gegangen, welche aufzeigten, dass Affine Image Matching zu den Problemen zählen, die in polynomieller Laufzeit lösbar sind. So konnte nachgewiesen werden, dass für einen quadratischen Bildausschnitt der Größe $n \times n$ *lediglich* eine Menge $\mathcal{D}(A)$ der Kardinalität $O(n^{18})$ untersucht werden muss, um das beste Affine Image Matching zu erzielen. Da in der Medizin jedoch Bilder erzeugt werden, die im Millimeter- oder gar Submillimeterbereich genau sein müssen, sollten die zwei zu vergleichenden Bilder wenigstens eine ähnliche Ausrichtung haben um die Menge $\mathcal{D}(A)$ weiter einzugrenzen. Beschränkt man sich auf die Werkzeuge, die in der Circuit Complexity Klasse TC^0 zur Verfügung stehen, so wird zwar die Menge $\mathcal{D}(A)$ größer, jedoch kann so Affine mage Matching stark parallelisiert werden.

In dieser Ausarbeitung werden Affine Image Matching und Circuit Complexity genauer betrachtet und erörtert. Daran anschließend wird ihre Bedeutung im Einsatz für die Medizinische Bildverarbeitung erläutert. Abschließen wird diese Ausarbeitung mit der Betrachtung der Parallelisierbarkeit von Affine Image Matching.

2 Affine Transformationen

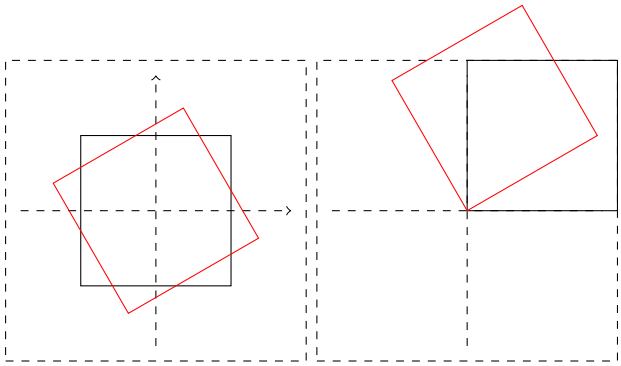
In dieser Ausarbeitung werden – wie in Quelle [1] – nur quadratische Bildausschnitte in *2D* betrachtet. Ein Bildausschnitt A – kurz Bild – der Größe n habe die Maße $(-n, -n)$ bis (n, n). Ein Bild hat also eine Menge $\mathcal{N} = \{(i, j) \mid -n \leq i, j, \leq n\}$ von Pixeln. Jeder Pixel hat eine Farbe $\mathcal{C}(i, j)$, welche sich mit einer natürlichen Zahl $\mathbb{N}$ beschreiben lässt. Sei $\mathcal{C}(i, j) = 0$ für alle $(i, j) \notin A$.

Die affinen Transformation sind die Grundoperationen des Affine Image Matching. beim Affine Image Matching geht darum ein Zielbild A so zu verändern, dass es dem Referenzbild B entspricht. Hierfür stehen die drei *linearen* Transformationen Drehung (Rotation), Skalierung und Scherung zur Verfügung, so wie die *affine* Parallelverschiebung (Translation). Für alle affinen Transformationen gilt, dass sie parallelität- und teilverhältniserhaltend sind. Parallelitätserhaltend bedeutet, dass Linien, die vor der Transformation parallel waren, auch nach einer Transformation noch parallel sind. Teilverhältniserhaltend bedeutet das drei Punkte auf einer Geraden nach einer Transformation noch immer das gleiche Verhältnis zueinander haben wie vorher.

Bei der Rotation ändert sich das Bild nicht, es wird bloß um den Koordinatenursprung gedreht. Abbildungen 7.1(a) und (b) zeigen in rot, wie das schwarze Quadrat nach einer Rotation um 30° um den Koordinatenursprung liegen würde. Die Funktion $f(i, j) = \begin{pmatrix} cos(30°) & -sin(30°) \\ sin(30°) & cos(30°) \end{pmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix}$ wurde auf jeden Eckpunkt des schwarzen Quadrates angewendet.

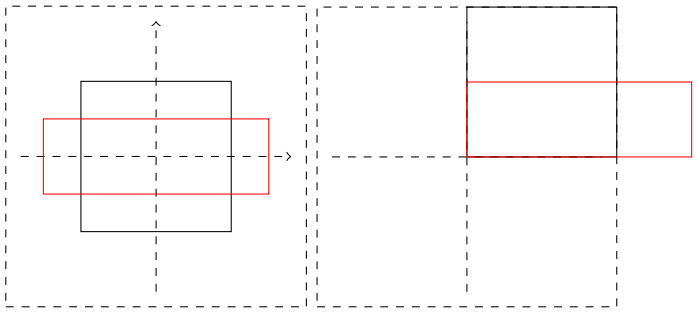
Bei der Skalierung kann gleichzeitig in x-Richtung und in y-Richtung unabhängig voneinander skaliert werden. Abbildungen 7.2(a) und (b) zeigen in rot, wie das schwarze Quadrat aussehen würde, wenn man alle Werte in x-Richtung mit dem Faktor 1.5 skalieren würde (Streckung) und alle Werte in y-Richtung mit dem Faktor 0.5 (Stauchung). Hier wurde die Funktion $f(i, j) = \begin{pmatrix} 1.5 & 0 \\ 0 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix}$ auf jeden Eckpunkt des Quadrates angewendet.

Bei der Scherung wird immer nur in eine Richtung geschert, es sind aber beide Richtungen möglich. Abbildungen 7.3(a) und (b) zeigen in rot, wie das schwarze Quadrat nach einer Scherung von x um 0.5 aussehen würde. Für eine solche Scherung würde die Formel $f(i, j) = \begin{pmatrix} 1 & 0.5 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix}$ auf jeden Eckpunkt angewendet werden. Bei einer Scherung in y-Richtung ist der Eintrag oben rechts 0 und unten links ungleich 0.



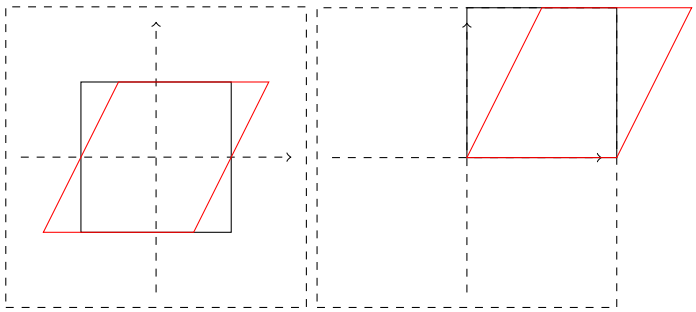
(a) Rotation POG (b) Rotation P1

Abb. 7.1: Rotation um 30°. (a) Schwerpunkt des Quadrates bei (0,0), (b) P1 bei (0,0).



(a) Skalierung POG (b) Skalierung P1

Abb. 7.2: Skalierung um 1.5 in x-Richtung und 0.5 in y-Richtung. (a) Schwerpunkt des Quadrates bei (0,0), (b) P1 bei (0,0).

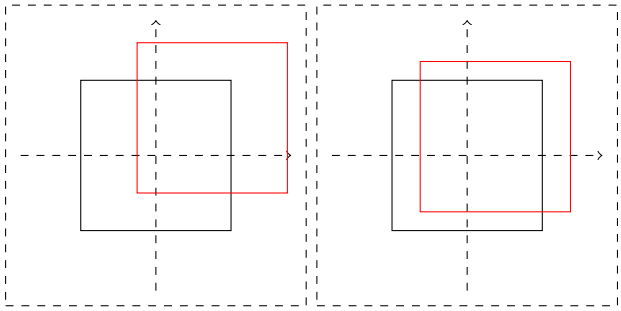


(a) Scherung POG (b) Scherung P1

Abb. 7.3: Scherung um 0.5 in x-Richtung. (a) Schwerpunkt des Quadrates bei (0,0), (b) P1 bei (0,0).

Wie man in diesen 6 Abbildungen sehen kann, ändern sich zwar die Positionen der roten Quadrate in Abhängigkeit zur Position der schwarzen Quadrate, aber die Deformation des Bildes ist unabhängig von der Position.

Bei der Translation wird das Bild nicht relativ verändert (über Faktoren), sondern absolut (über Summanden). Abbildungen 7.4(a) und (b) zeigen in rot jeweils eine Verschiebung um 0.75 (Einheiten) in x-Richtung und 0.5 (Einheiten) in y-Richtung angewandt einmal auf ein Quadrat mit den Eckpunkte (-1,-1) und (1,1) und einmal angewandt auf ein Quadrat mit den Eckpunkten (-2,-2) und (2,2). Die Funktion für die Eckpunkte hier lautet $f(i,j) = \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 0.75 \\ 0.5 \end{pmatrix}$.



(a) Verschiebung bei 2×2 (b) Verschiebung bei 4×4

Abb. 7.4: Translation um (+0.75,+0.5). (a) Quadrat mit Eckpunkten (-1,-1) und (1,1), (b) Quadrat mit Eckpunkten (-2,-2) und (2,2).

Die vier Transformationen können so kombiniert werden, dass sie sich in nur einer Formel f ausdrücken lassen. Eine affine Transformation

$$f(i,j) = \begin{pmatrix} a_1 & a_2 \\ a_4 & a_5 \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} a_3 \\ a_6 \end{pmatrix} \tag{7.1}$$

wird auf jeden Pixel $(i,j) \in \mathcal{N}$ angewendet um das Bild $f(A)$ zu erzeugen. Hierbei wollen wir eine Einschränkung machen: $a_1 \cdot a_5 \neq a_2 \cdot a_4$ oder anders formuliert, die Determinante der Transformationsmatrix soll ungleich Null sein. Dies ermöglicht eine eindeutige Abbildung eines jeden Pixels, und zudem ist f dann invertierbar. Sei $\mathcal{F}$ die Menge an Funktionen, die wie $f(i,j)$ formuliert werden kann und die oben genannte Einschränkung erfüllt.

Jedes f hat Parameter aus dem $\mathbb{R}^6$, und jedes $f(A)$ muss aus A berechnet pixelweise werden. Offensichtlich gewinnt man keine Informationen hinzu bei je zwei Funktionen f_1, f_2 für die gilt $f_1(A) = f_2(A)$. Es genügt eine Diskretisierung über alle Bilder $f_i(A)$ zu untersuchen. Diese Menge wird definiert als $\mathcal{D}(A) = \{f(A) | f \in F\}$. Offensichtlich ist es nicht sinnvoll erst alle $f \in F$ zu berechnen und zu schauen, welche in $\mathcal{D}(A)$ verbleiben. Die Bestimmung eines geeigneten $D(A)$ wird in Teil 5 beschrieben.

3 Affine Image Matching

Das Affine Image Matching beschäftigt sich mit der Suche nach der besten Übereinstimmung zwischen einem Zielbild A und einem Referenzbild B. Exakt formuliert bedeutet dies:

$$\text{Finde ein } f \in \mathcal{F} \text{ (bzw. } \textit{das } f(A) \in \mathcal{D}(A)\text{), so dass } f(A) = B.$$

In den meisten Fällen ist das genaue $f \in \mathcal{F}$ uninteressant und nur das $f(A)$ ist von Interesse. Dennoch wird f benötigt um $f(A)$ zu irgendeinem Zeitpunkt zu berechnen.

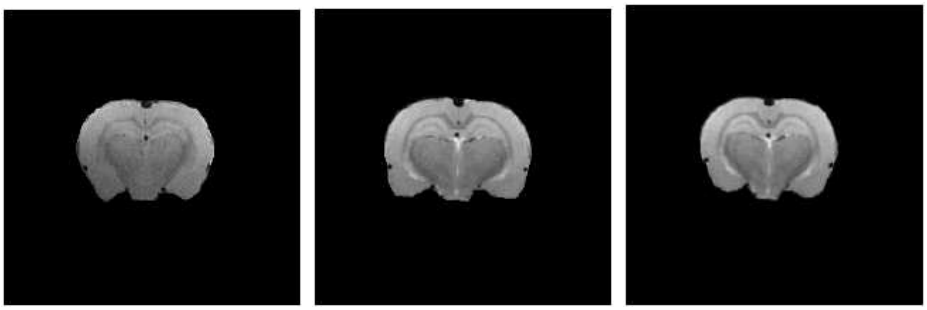
Meistens ist es nicht möglich ein Bild genau in ein anderes zu transformieren, besonders wenn man zwei Aufnahmen von weit auseinander liegenden Zeitpunkten betrachtet. Aus diesem Grund kann man auch eine Optimierungsvariante des Problems betrachten:

$$\text{Finde das } f(A) \in \mathcal{D}(A), \text{ so dass } \Delta(f(A), B) \rightarrow min,$$

wobei Δ eine Funktion ist, die den Unterschied zwischen zwei Bildern berechnen kann. Jeder Pixel aus $\mathcal{N}$ hat einen Farbwert c aus $\mathbb{N}$. Für zwei Bilder A, B seien $c_1 = C_A(i,j)$ und $c_2 = C_B(i,j)$ Farbwerte für den gleichen Pixel (i,j), dann ist Δ folgende Funktion:

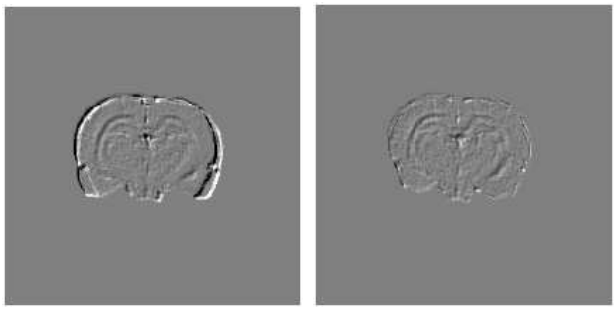
$$\Delta(A,B) = \sum_{i=-n}^{n} \sum_{j=-n}^{n} |C_A(i,j) - C_B(i,j)| \tag{7.2}$$

Die Abbildungen 7.5(a) und (b) zeigen zwei CT-Aufnahmen von Rattengehirnen von zwei unterschiedlichen Ratten. Um die Gehirne besser mit einander vergleichen zu können wurde (b) von der Form her angepasst, so dass es (a) gleicht. Das Resultat ist das transformierter Bild zu sehen in Abbildung 7.5 (c).



(a) Referenzbild (b) Zielbild (c) Transformiertes Bild

Abb. 7.5: CT-Aufnahmen von Rattengehirnen[4].



(a) Referenz- vs. Zielbild (b) Referenzbild vs. transformiertes Bild

Abb. 7.6: Graustufen geben die Unterschiede zwischen dem Referenzbild und dem Vergleichsbild an. (a) Vergleich mit (unverändertem) Zielbild; (b) Vergleich mit (optimal) transformiertem Bild.

In den Abbildungen 7.6 (a) und (b) kann man sehen, wie gut das Referenzbild (Abbildung 7.5 (a)) mit den beiden anderen Bildern (Abbildung 7.5 (b) und (c)) übereinstimmt. Hierfür würde pixelweise die Differenz zwischen den zu vergleichenden Bildern berechnet und mit in Graustufen dargestellt. Auf dem Bild Abbildung 7.6 (a) sind klar große Unterschiede zu erkennen. Abbildung 7.6 (b) hingegen zeigt kaum noch Unterschiede. Da es sich bei den beiden Ausgangsbildern (Abbildung 7.5 (a) und (b)), um die Aufnahmen zweier verschiedener Ratten handelt, war nicht davon auszugehen, dass eine perfekte Übereinstimmung erreicht werden kann.

4 Circuit Complexity

Die Hierarchie innerhalb von P:

$$AC^0 \subset TC^0 \subseteq NC^1 \subseteq \cdots \subseteq L \subseteq NC \subseteq P$$

In der Klasse P (auch $PTIME$) befinden sich die Entscheidungsprobleme, die mittels einer deterministischen Turingmaschine innerhalb von polynomieller Laufzeit lösbar sind.

In der Klasse NC (Nick's Class) befinden sich Entscheidungsprobleme, welche sich effizient parallel lösen lassen. Probleme in NC können auch mittels Circuits aus UND-, ODER- und NICHT-Gattern

gelöst werden[5]. Ein Circuit kann hierbei betrachtet werden als ein azyklischer, gerichteter Graph, dessen Knoten Gatter sind. Ein Knoten kann im Allgemeinen mehrere eingehenden und ausgehende Kanten haben. Die Größe des Circuit (Anzahl der Gatter) ist polynomiell beschränkt. Die Tiefe eines Circuits bezeichnet den längsten, möglichen Pfad.

Für alle $i \in \mathbb{N}$ gilt: NC^i enthält jene Sprachen, die von einer uniformen Circuit-Familie, einer Tiefe von $O(log^i(n))$ und Gattern mit konstanten fan-in (z.B. höchstens 2) erkannt werden.

Uniform bedeutet, dass allen Circuits der Familie etwas gemein sein muss, beispielsweise hier, dass alle Circuits dieser Familie mittels *einer* Turingmaschine beschrieben werden können, deren Ressourcen durch die Eingabelänge n beschränkt wird. Eine Circuit-Familie $\mathcal{C} = \{C_1, C_2, \dots\}$ besteht aus mehreren Circuits C_i, welche für unterschiedlichen Eingabelängen i definiert sind. Dies ist notwendig, da für jede Eingabelänge die gleiche Funktion berechnet werden soll, Circuits aber nicht plötzlich wachsen oder schrumpfen können. Für jedes i existiert immer nur genau ein C_i. Folgende Probleme lassen sich auch mit Hilfe von Circuits lösen: Integer Addition, Multiplikation und Division, sowie Matrix Multiplikation, Determinanten, Inverse und Rang.

Die Klasse AC erweitert die Circuit-Definition der Klasse NC, so dass hier die Gatter einen unbeschränkten fan-in haben. Dies bedeutet aber nicht, dass der fan-in variable auf die Eingabelänge reagiert, sondern lediglich, dass "Bäume" gleicher Gatter zu einem Gatter zusammen gefasst werden können. Es werden weiterhin Circuit-Familien benötigt. Es gilt außerdem: $NC^0 \subset AC^0 \subset NC^1$ und $NC^i \subseteq AC^i \subseteq NC^{i+1}$. Daraus folgt auch $AC = NC$.

Die Klasse TC wiederum erweitert die Klasse AC um Majority-Gatter (mit unbeschränktem fan-in). Ein Majority-Gatter gibt genau dann 1 aus, wenn mehr als die Hälfte der anliegenden Werte 1 sind. Es gilt: $AC^0 \subset TC^0 \subseteq NC^1$ und $NC^i \subseteq AC^i \subseteq TC^i \subseteq NC^{i+1}$. Daraus folgt auch $TC = AC = NC$. Mittels der Majority-Gatter kann das Optimierungsproblem auch als Entscheidungsproblem definiert werden: Gibt es ein $f \in \mathcal{F}$ so dass $\Delta(f(A), B) \leq t$ mit $t \in \mathbb{N}$. Entscheidungsprobleme können üblicherweise schneller gelöst werden als Optimierungsprobleme.

5 Bedeutung von uniform TC^0-complete

Uniform TC^0 bedeutet also,

- das ein Problem mittels UND-, ODER-, NICHT- und MAJORITY-Gattern gelöst werden kann,
- das der Circuit nur eine polynomielle Größe bezüglich der Eingabelänge n hat,
- das die Tiefe der Circuits in $O(log^0(n)) = O(1)$, also konstant ist,
- das der Fan-in der Gatter unbeschränkt ist (das NICHT-Gatter hat immer nur einen Fan-In von 1),
- das jeder Circuit C_i der Familie $\mathcal{C}$ mit der gleichen Turingmaschine $\mathcal{M}_\mathcal{C}$ beschrieben werden kann und nur Ressourcen in Relation zu i bedarf.

Ein Entscheidungsproblem $\Pi : \{0,1\}^* \to \{0,1\}$ ist $TC^0 - complete$, genau dann wenn Π in TC^0 liegt und es für alle anderen $\Pi' \in TC^0$ eine Funktion $r : \{0,1\}^* \to \{0,1\}^* \in AC^0$ gibt, so dass gilt: Für alle $s \in \{0,1\}^*$ für die $s \in \Pi'$ gilt, muss auch $r(s) \in \Pi$ gelten. Dies konnte in [1] gezeigt werden. Es bleibt hier zu erörtern, warum die Entscheidungsvariante von Affine Image Matching in TC^0 liegt.

Wie bereits in Teil 3 beschrieben, muss zur vollständigen Suche der Suchraum durch eine geeignete Diskretisierung $D(A)$ eingeschränkt werden. Die bislang beste gefundene Methode um Affine Image Matching zu lösen ist die vollständige Untersuchung der eingeschränkten Menge $\mathcal{D}(A)$ [1]. Hundt stellt in seinem Paper eine neue Diskretisierung vor, die auf den Entwicklungen der letzten Jahre im Bereich des Combinatorial Pattern Matchings basiert. Der Suchraum für die beste Funktion f oder wenigstens eine Funktion, welche ein Ergebnis kleiner dem Schwellwert (Threshold) t liefert ist in $\mathbb{R}^6$, da f 6 frei wählbare Einträge hat. Dieser Raum kann nun mittels von Hyperebenen in konvexe Faces φ unterteilt werden. Die Formel hierfür ist die Menge $H_n = \{I_{ijk}(a_1, \dots, a_6) : ia_1 + ja_2 + a_3 = k - 0.5 \mid (i,j) \in \mathcal{N}, k \in \{-n, \dots, n+1\}\} \cup \{J_{ijk}(a_1, \dots, a_6) : ia_4 + ja_5 + a_6 = k - 0.5 \mid (i,j) \in \mathcal{N}, k \in \{-n, \dots, n+1\}\}$. Jede Linie in Abbildung 7.7 erfüllt mindestens ein H_n.

Jede von Hyperebenen umschlossene Fläche (Face φ) erfüllt die in Teil 3 beschriebene Eigenschaft, dass für je zwei Funktionen f_1, f_2 für die $f_1(A) = f_2(A)$ gilt, beide im gleichen Face φ_i liegen. Somit ist die Diskretisierung geschafft und man benötigt nur noch einen Algorithmus, der für jedes Face φ_i eine

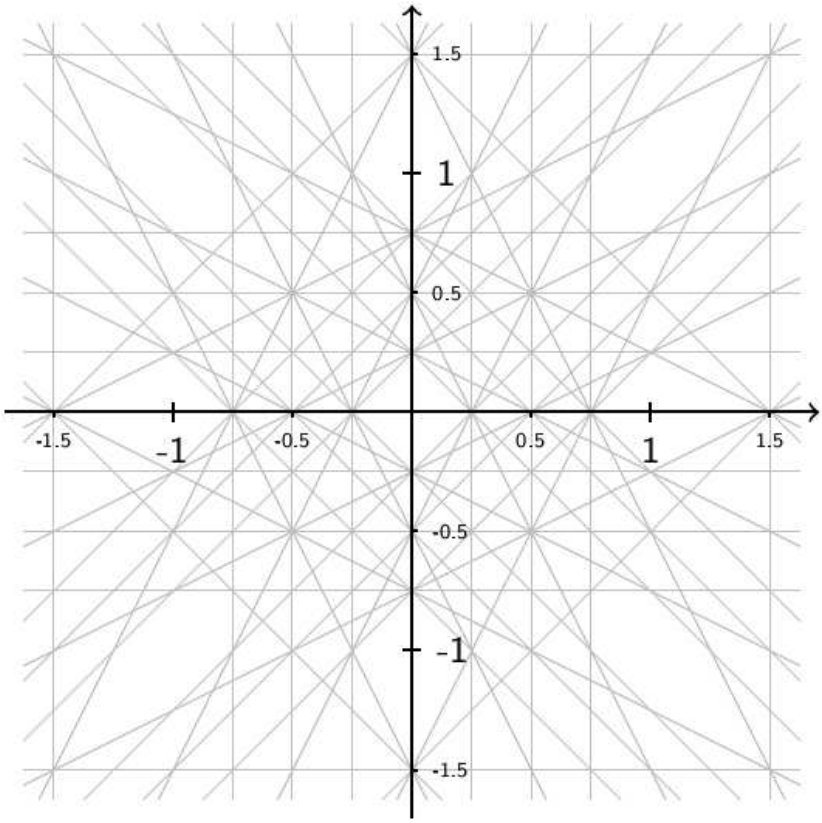


Abb. 7.7: Hyperebenen H_n[6].

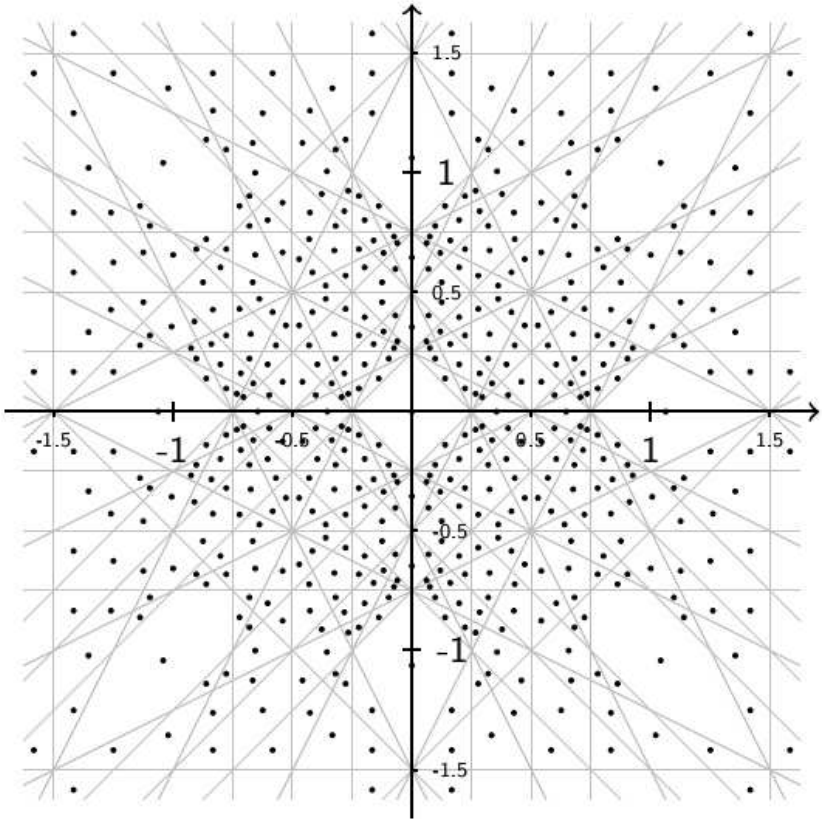


Abb. 7.8: Ein beliebiger Punkt pro Face[6].

Funktion f auswählt. Wie ebenfalls in Teil 3 beschrieben, sind die Funktionen aus $\mathcal{F}$ invertiertbar. Wenn man also in jedem Face φ_i einen beliebigen Punkt wählt (siehe Abbildung 7.8), dann kann man mittels des Inversen ein $f \in \mathcal{F}$ auswählen, und mit diesem $f_i(A)$ berechnen.

Ein erste Ansatz für einen Algorithmus sieht also folgendermaßen aus:

```
1. Erstelle diese Faces                | O(n^18) nach Edelsbrunner
2. Durchlaufe alle Faces               | O(n^18), z.B. Tiefensuche
3.   Berechne f(A)                     | O(1) amortisiert
4.   Berechne Delta(f(A,B))            | O(1) amortisiert
5. Gib f(A) mit kleinstem Delta zurück | O(1)
```

Bereits 1986 fand Edelsbrunner heraus, dass man $O(n^{18})$ viele Faces erhält, wenn man den Raum mit Hyperebenen durchzieht, wie in H_n beschrieben, wobei n die Größe unseres Bildes angibt, also einen Bildausschnitt mit den Maßen (-n,-n) bis (n,n) [7]. Für das optimale Ergebnis des Affine Image Matching muss natürlich jedes Bild betrachtet werden, und es wurden durch Schritt 1 $O(n^{18})$ unterschiedliche Bilder ermittelt. Das Durchlaufen aller Faces (Schritt 2) ist offensichtlich liniar in der Anzahl der Face, also $O(n^{18})$. $f(A)$ in Schritt 3 lässt sich amortisiert in $O(1)$ berechnen, da alle zur Berechnung notwendigen Operationen in TC^0 sind. Diese TC^0-Operatoren sind: Integer Addition (AC^0), Multiplikation (TC^0), Division (TC^0), $\sum$-Operationen (TC^0), Majority-Operator (TC^0), sowie Integer Minimum ($TC^0 - Complete$). Probleme in TC^0 wiederum haben nur eine konstante Tiefe und somit auch nur eine konstante Laufzeit (unter der Annahme, dass alle Gates mit vollständigem Input parallel ihren Output berechnen können). Gleiches gilt für Δ (Schritt 4) und die Rückgabe (Schritt 5), da eben der Summen-Operator und die Berechnung des Minimums in TC^0 liegen. Allerdings ist nicht bekannt, ob Schritt 1, die Erstellung der Faces, in TC^0 berechnet werden kann.

6 Anwendung in der Medizin (Bildfusion)

In der Medizin findet das Affine Image Matching bei der sogenannten der Bildfusion Anwendung. Eine Bildfusion ist die Überlagerung zweier Bilder. Bei Satellitenaufnahmen der Erde werden sich überlappende Ränder affin ausgerichtet um so mehrere Aufnahmen zu einer größeren Karte quasi zusammenzukleben. In der Medizin werden meist Bilder aus zwei verschiedenen bildgebenden Verfahren komplett übereinander gelegt um einen Informationsgewinn zu erzielen.

Es gibt in der Medizin mehrere (meist radiologische) Bildgebungsverfahren. Hier eine Auswahl und ihre typischen Einsatzgebiete:

Röntgen: Beim Röntgen werden die Röntgenstrahlen durch die unterschiedlichen Strukturen im Körper unterschiedlich stark absorbiert. Mit diesem Verfahren lassen sich nicht nur Knochenbrüche analysieren, sondern bei niedrigeren Frequenzen auch Gewebe (z.B. bei der Mammographie). Mittels der Zugabe von Kontrastmitteln können sogar Funktionen von Organsystemen sichtbar gemacht werden. Man spricht von digitalem Röntgen, wenn das Bild mittels eines elektronischen Detektors direkt aufgenommen wurde, oder der Röntgenfilm eingescannt wurde.
In Abbildung 7.9 kann man sehen, dass sich Strukturen (Knochen) überschneiden und aus diesem Grund aus mehreren Richtungen geröntgt werden muss.

Abb. 7.9: Röntgen-Aufnahmen eines menschlichen Ellenbogens[8].

CT: Bei der Computertomographie werden sehr viele (Röntgen-)Strahlen aus unterschiedlichen Winkeln auf das Objekt geschossen und der Absorptionsgrad ermittelt. Aus diesen Strahlen kann mittels eines Computers ein Bild berechnet werden. Mit dem CT können Knochenbrüche, Blutergüsse

(Flüssigkeit) und Weichteilstrukturen sehr gut ermittelt werden. Da dieses Verfahren schneller und auch günstiger ist, wird es meist dem MRT gegenüber vorgezogen. Tumore können mit diesem Verfahren nur sehr schwer erkannt werden.
Abbildung 7.10 zeigt mehrere Schichten eines CT-Scans eines Kopfes. Da die Bilder aus den vielen Messungen berechnet werden, können Strukturen überschneidungsfrei angezeigt werden.

Abb. 7.10: 5 Schichten aus einer CT-Aufnahme eines menschlichen Kopfes[9].

PET: Die Positronen-Emissions-Tomographie gehört zu den funktionellen Bildgebungsverfahren. Durch Gabe eines Radiopharmakons, welches im Patienten zerfällt, kann mittels räumlicher und zeitlicher Messungen die räumliche Verteilung des Radiopharmakons ermittelt werden. Ihr Haupteinsatzgebiet sind Stoffwechselvorgänge, also auch das Aufspüren von Tumoren.
Wie man in Abbildung 7.11 sehen kann, ist im Gegensatz zu Röntgen und CT der Schädelknochen kaum sichtbar, dafür allerdings die Gehirnstrukturen um so besser.

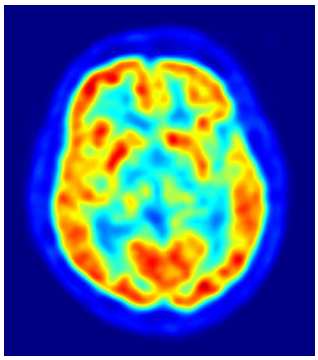


Abb. 7.11: PET-Aufnahme eines menschlichen Gehirns; rot zeigt eine hohe Ansammlung des Radiopharmakons, blau eine niedrige[10].

PET-CT: Seit 2001 werden CT und PET in einem Gerät verbaut um gegenseitig ihre Schwächen auszugleichen (weiteres siehe unten).
Auf der CT-Aufnahme in Abbildung 7.12 kann man im unteren Bereich der grauen Fläche gut den Rückenwirbel erkennen, doch andere Strukturen bleiben verborgen. In den beiden PET-Aufnahmen kann man zwar die Organe besser erkennen, jedoch ist es schwer Knochen zu sehen. Die Bildfusion (rechtes Bild) zeigt die aus dem PET über die Organe gewonnenen Informationen fest verankert in dem strukturgebenden Bild des CTs an.

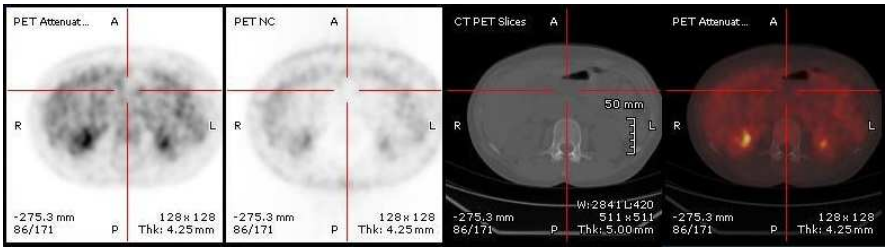


Abb. 7.12: Zwei PET-Aufnahmen (links), CT-Aufnahme (mitte), Bildfusion (rechts)[11].

MRT: Die Magnetresonanztomographie (manchmal auch Kernspintomographie genannt) ist im Gegensatz zu den bisher genannten Verfahren kein radiologisches Bildgebungsverfahren. Im Bereich des Gehirns wird sie oft auch als funktionelles Bildgebungsverfahren eingesetzt. Sie wird hauptsächlich verwendet um Strukturen und Funktionen von Gewebe und Organen darzustellen. Mit ihr lassen sich (krankhafte) Veränderungen an Organen erkennen. Das MRT misst nur Intensitätsdifferenzen, welche durch Zugabe von Kontrastmitteln noch erhöht werden können. Es gibt verschiedene Gewichtungen, welche zu unterschiedlichen diagnostischen Zwecken eingesetzt werden können (z.B. T1-Gewichtung für das Aufspüren von Tumoren, T2-Gewichtung für das Aufspüren von Flüssigkeiten).
In Abbildung 7.13 kann man sehr gut sehen, dass eine MRT-Aufnahme eine wesentlich genauere Strukturierung des Gehirns anzeigen kann als das CT (Abbildung 7.10). Diese Genauigkeit kostet aber nicht nur mehr, sondern bedeutet für den Patienten ein wesentlich längeres Stillliegen.

Abb. 7.13: MRT-Aufnahme eines menschlichen Gehirns[12].

PET/MRT: Die neuste Entwicklung seit 2011. Wie PET-CT, aber mit MRT statt CT.
Das MRT wird gerne eingesetzt um das Gehirn zu scannen. Dies nun zusammen dem PET wird in Abbildung 7.14 gezeigt.

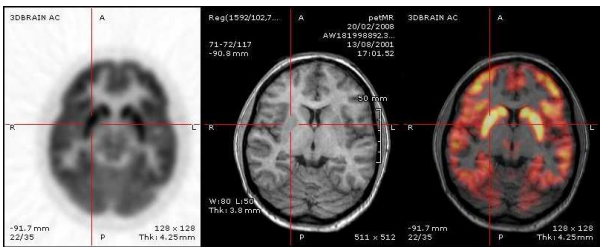


Abb. 7.14: PET-Aufnahme (links), MRT-Aufnahme (mitte), Bildfusion (rechts) eines menschlichen Gehirns[13].

Besonders interessant für diese Ausarbeitung sind Bildfusionen zwischen Aufnahmen aus einem PET und einem CT bzw. MRT. Hierbei spielt es keine Rolle, ob die Bilder in einem PET-CT gemacht wurden, oder auf getrennten Geräten. Da PET- und CT-Aufnahmen, wie man oben sehen kann, sehr unterschiedliche Schwerpunkte haben und entsprechend unterschiedlich aussehen, müssen für das Affine Image Matching in beiden Aufnahmen wenige Punkte exakt markiert werden, welche dann gegeneinander ausgerichtet werden, anstatt dass alle Daten aus den Bildern verwendet werden. Obwohl es nicht zu den Stärken des PETs zählt, sind Knochen meist noch erkennbar. Erst wenn die zwei Bilder aneinander ausgerichtet sind, werden alle Daten beider Bilder fusioniert. Das PET benötigt eine sogenannte Schwächungsmap. Diese kann in einem PET-CT durch einen groben Scan des CTs angefertigt werden. Der Vorteil eines PET-CTs ist es, dass bereits beim Scannen die Bilder aneinander ausgerichtet werden. Das PET-CT liefert dann die eigentliche Ausgabe des CTs, des PETs und anschließend auch die Bildfusion.

Auch für die Medizin interessant ist die Veränderung eines einzelnen Organs über die Zeit. Hierzu kann erneut die Knochenstruktur zum Ausrichten der Bilder verwendet werden (wenn sichtbar), so dass die Veränderung des Organs wie im Zeitraffer betrachtet werden kann.

7 Parallelisierung

Hundt behauptet, dass seine Erkenntnis, das Affine Image Matching in TC^0 liegt, keinen Einfluss auf die Verwendung von Affine Image Matching in der Praxis haben werde[1]. Zuerst einmal stellt sich allerdings die Frage, ob Bildfusion heutzutage noch verwendet wird, wenn es doch sei 2001 ein PET-CT Gerät gibt, welches dies bereits eigenständig erledigt.

Huang hat 2007 für seine Dissertation Mensch gegen Maschine antreten lassen um *3D* CT-Aufnahmen auszurichten[4]. Ein professioneller Mitarbeiter, der eine gewisse Routine hat, benötigte zum Ausrichten der Scans 3 bis 5 Minuten um einen sogenannten Gold-Standard zu erreichten. Ein ungeübter Mitarbeiter, der das gleiche Programm bedienen kann, benötigte für diese Aufgabe 20 bis 32 Minuten. Der von ihm implementierte Algorithmus für Affine Image Matching mit 8 Freiheitsgraden benötigte 135 Minuten (auf einer CPU) und hatte dabei eine Abweichung von 4% vom Gold-Standard.

Ob der Computer so langsam war, weil der Code schlecht war, oder das Problem in *3D* so viel schwieriger ist als in *2D*, und warum eine Abweichung von 4% bleibt, kann an dieser Stelle nicht geklärt werden, auch nicht, ob der Code gut oder schlecht parallelisierbar gewesen wäre um so wenigstens ein bisschen Zeit einzusparen (z.B. auf 1/8 von 135 Minuten, also 17 Minuten Bearbeitungszeit bei einem 8-Kern-Prozessor). In der Theorie[1] wurde jedenfalls bewiesen, dass Affine Image Matching komplett in TC^0 lösbar ist, wenn man auf die Faces (siehe Teil 5) verzichtet und stattdessen ein feines Gitter über den Raum legt.

Legt man ein Gitter $\mathcal{G}_n$ mit der Formel $(a_1, \ldots, a_6)^T = 10^{-7}n^{-7}(t_1 + 0.5, t_2, t_3, t_4, t_5 + 0.5, t_6)^T$ für $t_1, \ldots, t_6 \in \{-10^{12}n^{13}, \ldots, 10^{12}n^{13}\}$ an, so hat dieses Gitter eine Kardinalität von $O(n^{78})$ [1]. Alle Punkte $(a_1, \ldots, a_6)^T$ erfüllen $a_1a_5 \neq a_2a_4$ und zudem liegt jedem Face immer mindestens ein Punkt. Abbildung 7.15 zeigt ein solches Gitter über den Faces. Nun kann für jedes so entstandene f parallel $f(A)$ berechnet und mit B vergleichen werden. Dies kann sogar in einem sehr großen Circuit getan werden, siehe Abbildung 7.16. Abschließend wird aus allen Berechnungen das f gewählt, für welchen Δ am kleinsten war.

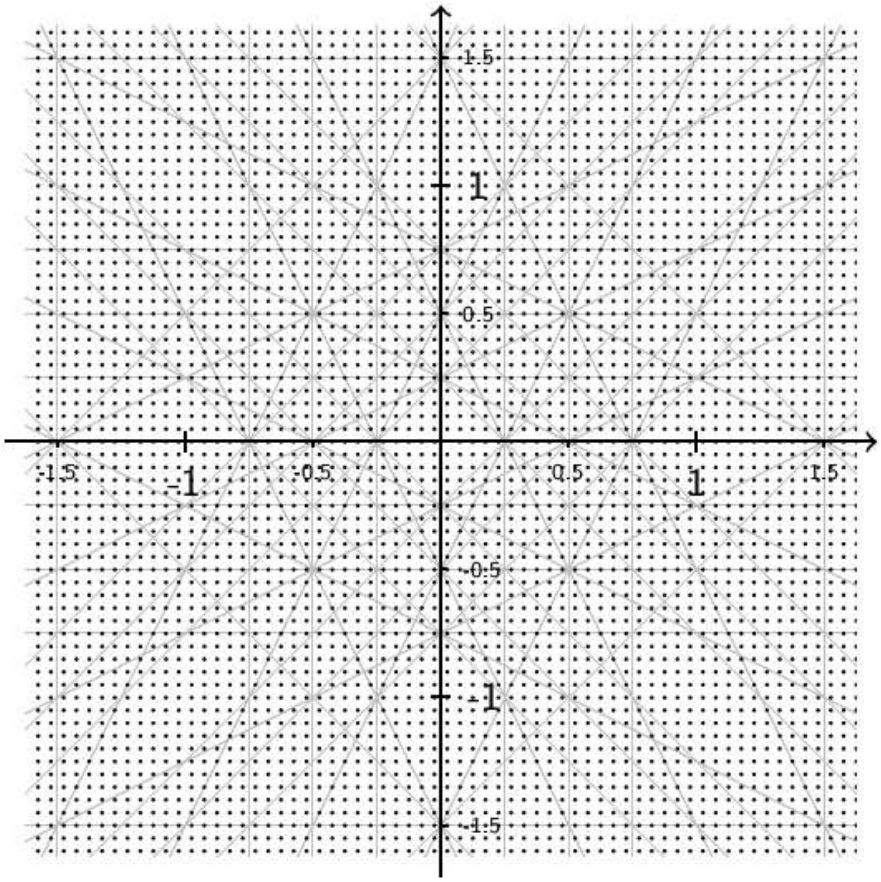


Abb. 7.15: Gitter $\mathcal{G}_n$[6].

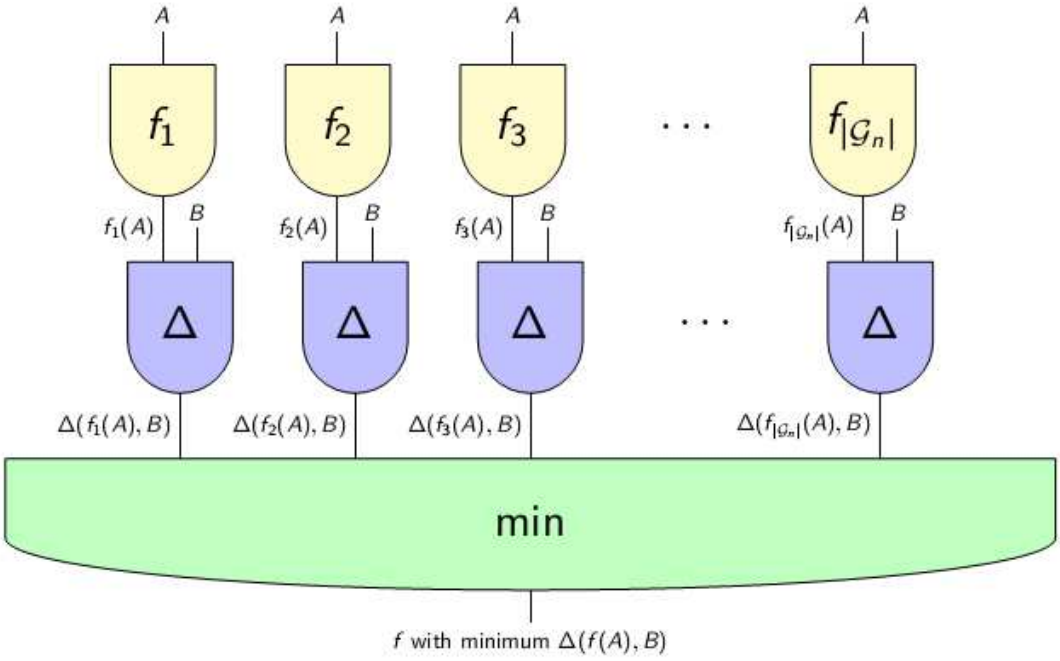


Abb. 7.16: Paralleles Affine Image Matching in TC^0[6].

Es ist also möglich, Affine Image Matching komplett in TC^0 zu lösen und das Ganze sogar zu parallelisieren, wenn auch mit einer zu untersuchenden Menge von $O(n^{78})$ Punkten gegenüber $O(n^{18})$. Da es sich bei TC^0 "nur" um recht einfache Gatter handelt, besteht eventuell die Möglichkeit Affine Image Matching auf Grafikkarten zu lösen, da diese bereits heute bis zu 4096 einfache Prozessoren auf sich vereinen.

Literaturverzeichnis

[1] Hundt C. Affine Image Matching Is Uniform TC^0-Complete. In: Combinatorial Pattern Matching 2010. Springer; 2010. p. 13–25.

[2] Hundt C, Liśkiewicz M. On the Complexity of Affine Image Matching. In: Symposium on Theoretical Aspects of Computer Science 2007. Springer; 2007. p. 284–295.

[3] Hundt C, Liśkiewicz M. Combinatorial Bounds and Algorithmic Aspects of Image Matching under Projective Transformations. Springer; 2008. p. 395–406.

[4] Huang W. Automatic Affine and Elastic Registration Strategies for Multi-Dimensional Medical Images. Worcester Polytechnic Institute; 2007.

[5] Vollmer H. Introduction to Circuit Complexity: A Uniform Approach. Springer; 1999.

[6] Hundt C. Affine Image Matching Is Uniform TC^0-Complete; 2010. Presentation at 21st Annual Symposium on Combinatorial Pattern Matching.

[7] Edelsbrunner H, O'Rourke J, Seidel R. Constructing arrangements of lines and hyperplanes with applications. SIAM Journal on Computing. 1986;15(2):341–363.

[8] Wikimedia Commons. File:Computed tomography of human brain - large.png — Wikimedia Commons; 2008 [Online; accessed 08-December-2012]. http://en.wikipedia.org/wiki/File:Computed_tomography_of_human_brain_-_large.png.

[9] Wikimedia Commons. File:Coude fp.PNG — Wikimedia Commons; 2006 [Online; accessed 08-December-2012]. http://en.wikipedia.org/wiki/File:Coude_fp.PNG.

[10] Wikimedia Commons. File:PET-image.jpg — Wikimedia Commons; 2010 [Online; accessed 08-December-2012]. http://en.wikipedia.org/wiki/File:PET-image.jpg.

[11] Wikimedia Commons. File:Viewer medecine nucleaire keosys.JPG — Wikimedia Commons; 2008. [Online; accessed 08-December-2012]. http://commons.wikimedia.org/wiki/File:Viewer_medecine_nucleaire_keosys.JPG.

[12] Wikimedia Commons. File:Brain Mri nevit.svg — Wikimedia Commons; 2006. [Online; accessed 08-December-2012]. http://en.wikipedia.org/wiki/File:Brain_Mri_nevit.svg.

[13] Wikimedia Commons. File:PET-MR2-Head-Keosys.JPG — Wikimedia Commons; 2012. [Online; accessed 08-December-2012]. http://commons.wikimedia.org/wiki/File:PET-MR2-Head-Keosys.JPG.

Graph-Based Segmentation

Phan-Anh Nguyen

Contents

Abstract

The purpose of image segmentation is to partition an image into meaningful regions with respect to a particular application. Segmentation is commonly used in medical image processing to locate tumors, measure tissue volumes and to study anatomical structure. Segmentation is also a crucial part in scene understanding for autonomous systems. However there are several issues often encountered during segmentation. First, the object we want to detect does not always have the same shape. This can cause troubles for the segmentation methods that depend on pre-defined shapes. Secondly, there are no clear boundaries between objects and background due to image clutters and noises. This seminar paper reviews three novel graph-based approaches to the image segmentation problem. The first approach builds a region graph to detect object as the optimal subgraph weighted by a support vector machine classifier applied to bag-of-features regions. The second approach identifies salient contours within an image by solving an Hermitian eigenvalue problem on a contour grouping graph. The third approach extends the min-cut algorithm to solve the multi-class segmentation problem on a Markov random field. Pros and cons and some comparisons between these graph-cut methods are discussed at the end of the present seminar paper.

Keywords: Classification, Graph-Cut, Segmentation

1 Introduction

Image segmentation is an initial and vital step in the whole image processing pipeline aimed at overall image understanding. Segmentation is often used to identify objects in a scene for object-based measurements such as size and shape. This is particularly useful for computer-aided diagnosis applications which assist physicians in finding abnormalities based on medical images of patients. Segmentation is also used to identify obstacles in depth images taken from a mobile robot to generate the optimal path. There are two main problems that prohibit the segmentation task from delivering good results. First, the object we want to detect does not always have the same shape. This can cause troubles for the segmentation methods that depend on pre-defined shapes. Secondly, there are no clear boundaries between objects and background due to image clutters and noises.

Graph-based segmentation is currently emerging as a promising method and has been applied successfully to many image processing applications ranging from scene understanding and segmentation to medical image analysis. Formulating segmentation problems by means of graph theory has a twofold advantage: Firstly, graph representation is an abstract way to encode generic complex relationships between entities, thereby opening up to a wide variety of interpretations for the segmentation problems provided that they can define some metrics to measure the relationships; Secondly, graph representation often leads to well-known graph problems which have been studied for a while and therefore can be solved efficiently.

In general, a graph-based segmentation method often follows the following pipeline. First, image features, which are signal responses from some filters describing important image properties, are extracted from an input image. These features serve as numerical elements used to construct a graph often based on supervised learning algorithms. Once we have built a graph to represent the image segmentation problem, we can apply state-of-the-art graph-cut algorithms to solve the problem efficiently.

Beside graph-based approaches, a wide variety of segmentation methods have been proposed. Mortensen et al. [1] introduced an interactive segmentation system in which an user picks up some seed points to define a contour that can automatically snap around the object of interest. The system used the Dijkstra's shortest path algorithm to find the lowest cost path in a cost image which is defined based on gradient. Chan and Vese proposed a segmentation method called "Active contours without edges" [2] which used level set formulation to model the evolution of boundaries under internal and external forces. The system did not use gradient for stopping condition thereby being able to identify objects that are very smooth, or even have discontinous boundaries. The system proposed by Fripp et al. [3] uses three-dimensional active shape models to segment cartilages and rebuild a 3D-model of them. The reconstruction error is below 1 mm.

This seminar paper presents several different graph-based approaches to the image segmentation problem and demonstrates some applications of these techniques to real-world problems. Particularly, Section 2 describes several methods to extract features from images notably Speeded Up Robust Features (SURF) [4] is used to describe local image information around a feature point as well as a shape descriptor [5] based on the Globalized Probability of Boundary (gPb) [7] [8]. Section 3 introduces some supervised

learning techniques which are used in Section 4 to construct a graph. Section 5 presents powerful graph-cut algorithms used to select optimal subgraphs as segmentation solutions. The applications of graph-cut algorithms to solve the segmentation problem are illustrated in Section 6 in three different systems namely "Efficient Region Search" [9] to search for an object of arbitrary shape, "Contour Cut" [10] to identify salient contours in images and "Multi-Shape Graph-Cut" [11] to perform lung segmentation based on multiple shape priors. Finally, Section 7 discusses advantages and disadvantages and gives some comparisons between these graph-cut methods.

2 Image Features

Low level image features are essential building blocks for high level image processing tasks such as object detection, image categorization or segmentation. In general, image features capture important properties around local image regions in the form of high dimensional feature vectors that can be used by high level applications. The advantage of the compact representation of features is that it can be readily fed into standard algorithms which often take high dimensional vectors as input. Also features are often made to be invariant to different transformations e.g. rotation, scaling etc. Since features are defined locally, they are invariant to translation. To support real-time systems features need to be efficient to compute. Normally, raw features are extracted at each pixel by applying various kinds of filters. Each filter response corresponds to one dimension in the feature vector space. The shape and size of filters reflect local structure around each pixel. Raw features at each pixel can be collected to form region descriptors which can be used to represent shapes, contours etc. In this section, we present methods to extract raw features and various ways of constructing region descriptors. Specifically, Section 2.1 describes a state-of-the-art feature named Speeded Up Robust Features (SURF) [4] and Section 2.2 explains the Globalized Probability of Boundary (gPb) contour detector [8].

2.1 Speeded Up Robust Features

Speeded Up Robust Features (SURF) [4] was originally developed for the task of finding point correspondences between two images of the same scene. This includes three main steps. First, interest points are selected at distinctive locations in the image. Then a feature vector is extracted from the neighbourhood of every interest point. Finally, the feature vectors are matched between different images. Since the SURF descriptor provides a very good representation of a local region it has also been applied to tasks other than correspondence problems such as image classification or image segmentation.

Regarding the interest point detection problem the choice of a detector varies depending on the needs of the application. An application for image registration requires the same interest points to be detected in two different images of the same scene under different viewing conditions. In this case, a detector would pick up hard-to-miss points which appear in both images such as peaks in signal responses corresponding to blobs or corners. In the case of image segmentation it would be a wise choice to select points on contours to be interest points. In the original SURF paper Bay et al. suggested to use an approximate Hessian detector to search in scale-space domain for points having strong derivatives in two orthogonal directions as interest points (often located at corners and strongly textured areas).

Given an interest point in the input image the SURF descriptor is obtained by extracting distinctive information around its neighborhood in a form of a feature vector. The SURF descriptor is designed to be invariant to image scaling and rotation while it can be computed very fast. Scale invariance is achieved by adopting the scale at which the Hessian detector attains maximal response. In order to be invariant to rotation we first find a reproducible orientation based on information within a circular region around the interest point. We then construct a square region aligned to the selected orientation and extract the SURF descriptor from it.

To find the dominant orientation Haar wavelet responses in x and y directions (d_x, d_y) are calculated within a circular neighbourhood of radius $6s$ around the interest point and weighted with a Gaussian ($\sigma = 2s$), where s is the scale chosen above. Figure 8.1(a) shows the structure of the Haar wavelet filters. The wavelet responses (d_x, d_y) are then represented in a vector space and the sum of all vectors within a sliding orientation window of size $\frac{\pi}{3}$ is calculated, see Figure 8.1(b). The dominant orientation is finally assigned to the sum vector having the maximal length.

To extract the SURF descriptor, we construct a window of size $20s$ centred at the interest point and oriented in the dominant orientation as illustrated in Figure 8.1(c). The window is then subdivided into regular 4×4 grid cells, each being a 5×5 pixel patch. For each cell, the Gaussian weighted ($\sigma = 3.3s$)

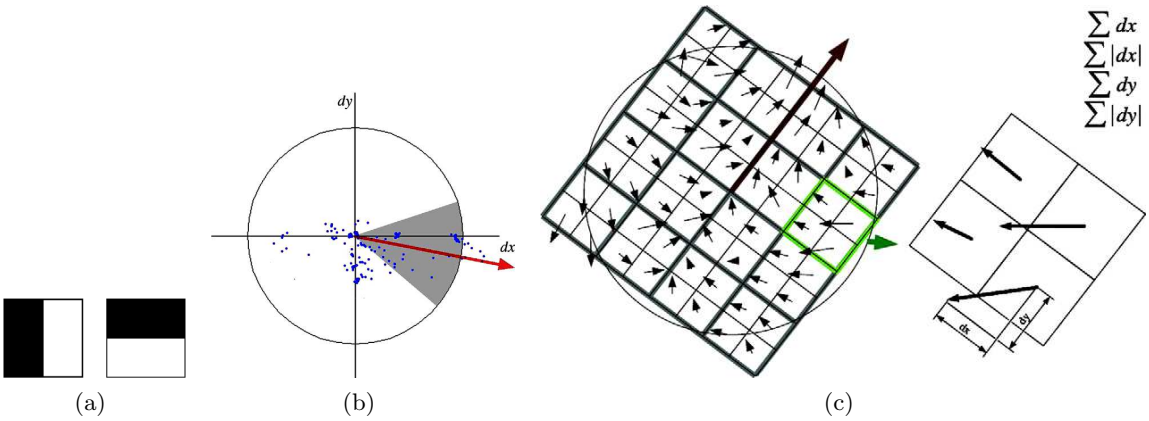


Figure 8.1: (a) Haar wavelet filters for the x (left) and y (right) components. The dark parts are weighted -1 and the light parts +1. (b) Orientation assignment: A sliding orientation window of size $\frac{\pi}{3}$ detects the dominant orientation. (c) The SURF descriptor: An oriented quadratic grid with 4×4 square cells is laid over the interest point (left). For each cell, the wavelet responses are computed from 5×5 samples (for illustrative purposes, only 2×2 sub-divisions are shown here). In each cell, the 4 elements $\sum d_x$, $\sum |d_x|$, $\sum d_y$, $\sum |d_y|$ contribute to the SURF feature vector. The images are taken from [4].

Haar wavelet responses are summed up to form a first set of entries in the feature vector. The sum of the absolute values of the responses, $|d_x|$ and $|d_y|$ are also extracted to capture information about the polarity of the intensity changes. Therefore, each cell has a 4D descriptor vector for its underlying intensity structure $(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Concatenating this for all 4×4 cells results in a descriptor vector of length 64. Finally the descriptor vector is normalized to make it invariant to contrast.

2.2 Globalized Probability of Boundary

The globalized probability of boundary (gPb) contour extractor was developed based on the predecessor, namely probability of boundary (Pb), which uses features extracted from a local image patch to estimate the posterior probability of a boundary passing through the current pixel. gPb improves from Pb by introducing global information based on spectral graph theory. In this section we will first survey the local features used by Pb. We will then explain how gPb incorporates global information into Pb. Finally, we will show one method to extract shape descriptors from regions defined by gPb contours. The method to estimate the posterior probability of a boundary will be explained in more detail in Section 3.2.

2.2.1 Gradient-Based Features for Probability of Boundary The gradient-based paradigm was used by Martin et al. in their *Pb* paper [7] to detect local changes in color, texture and brightness. The algorithm to compute the probability of boundary based on brightness and color features proceeds with the following steps:

1. At each pixel location (x, y), a circle of radius r is drawn and cut half along the diameter having orientation θ.
2. Histograms of color and brightness in the CIELAB color space [1] are built for each half disk. The L* channel is used to compute the brightness histogram and the a* and b* channels are used to compute the color histogram.
3. Let $h(x, y, \theta, r)$ and $g(x, y, \theta, r)$ denote the histograms of the two disk halves. We define the gradient function $G(x, y, \theta, r)$ which is the χ^2 distance between the histograms h and g:

$$\chi^2(g, h) = \frac{1}{2} \sum \frac{(g_i - h_i)^2}{g_i + h_i} \tag{8.1}$$

[1]The CIELAB color space has three dimensions. The L* axis represents lightness ranging from 0 for black to 100 for white. The a* axis is green at one extremity (represented by -a), and red at the other (+a). The b* axis has blue at one end (-b), and yellow (+b) at the other. The color range is between -128 and +127.

4. An edge is detected if there is a large difference $G(x, y, \theta, r)$ between the two half disks along the chosen orientation.

5. A supervised learning method is used to combine the cues into a single detector. This produces soft boundary maps of the form $Pb(x, y, \theta)$.

(a) (b) (c)

Figure 8.2: (a) Filter Bank: The 13-element filter bank used for computing textons. (b) Universal Textons: Example universal textons computed from the 200 training images, sorted by L1 norm for display purposes. (c) Texton Map: An image and its associated texton map. The images are taken from [7].

The same algorithm is used to compute texture gradient but the technique to calculate the texture histogram is different. Specifically, a group of 13 filters covering the most probable shapes is used. It consists of six pairs of elongated, oriented filters and a center-surround filter as shown in Figure 8.2(a). The oriented filters are in even/odd quadrature pairs. The even symmetric filter is a Gaussian second derivative, and the odd-symmetric filter is its Hilbert transform. Finally, a difference of Gaussians is chosen to be the center-surround filter. This filter bank generates a feature vector of 13 dimensions corresponding to 13 filter responses at each pixel.

In order to build up histograms for comparison the Pb algorithm uses the so called textons approach (or bag-of-textures approach) introduced by Malik et al. [6]. The basic idea is to cluster the filter response vectors over a large diverse collection of training images using k-means. Each cluster defines a Voronoi cell in the space of joint filter responses and the cluster centers, textons, represent texture primitives. Figure 8.2(b) illustrates example textons for k = 64 computed over the 200 images in the training set. Once the dictionary of textons has been computed each pixel is assigned to the nearest texton. Figure 8.2(c) shows an image and the associated texton map, where each pixel has been labeled with the nearest texton. Finally, the histograms of activated textons in the two disc halves can be computed and compared using the χ^2 distance operator.

2.2.2 Globalized Probability of Boundary Detector The gPb contour detector [8] is an extension of the Pb contour detector into which global information is integrated via spectral graph theory. Particularly, the gPb algorithm follows the idea of the normalized cut algorithm, which defines an affinity matrix W, whose entries encode the similarity between pixels. The generalized eigenvectors of the following linear system provide global segmentation information:

$$(D - W)v = \lambda Dv \tag{8.2}$$

where D is diagonal matrix with $D_{ii} = \sum_j W_{ij}$. Section 5.2 will explain the normalized cut algorithm in more detail.

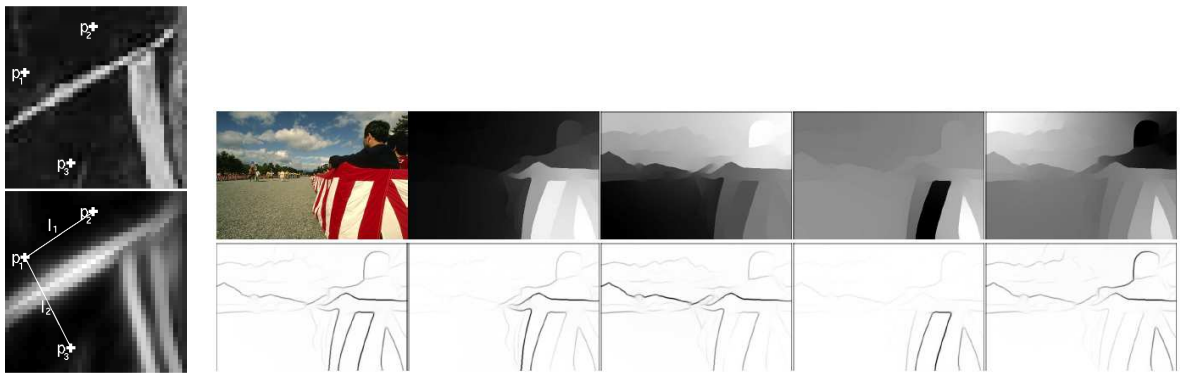
We start by extracting the brightness, color and texture gradients in 3 different scales: $[\sigma/2, \sigma, 2\sigma]$, where σ is the default scale of the Pb detector. This gives 9 different responses G_i. These local cues are then linearly combined into a single multiscale oriented signal:

$$mPb(x, y, \theta) = \sum_{i=1}^{9} \alpha_i G_i(x, y, \theta) \tag{8.3}$$

In order to integrate global information the affinity matrix W is constructed by using the intervening contour cue [29], that is, its entries are the maximal values of mPb along a line connecting two pixels as shown in Figure 8.3(a). The first $k + 1$ generalized eigenvectors v_j of the system 8.2 are chosen and reshaped in the size of the original image. Contours are extracted from each eigenvector v_j by applying Gaussian directional derivatives at multiple orientations θ, resulting in an oriented signal $sPb_{v_j}(x, y, \theta)$. The information from different eigenvectors is then combined to provide the spectral boundary detector:

$$sPb(x, y, \theta) = \sum_{j=1}^{k} \frac{1}{\sqrt{\lambda_j}} sPb_{v_j}(x, y, \theta) \tag{8.4}$$

where $0 = \lambda_0 \leq ... \leq \lambda_k$ are the corresponding eigenvalues. Figure 8.3(b) illustrates an example of the spectral boundary detector sPb.



(a) (b)

Figure 8.3: (a) Example of an intervening contour. Above is an image patch in which the intensity values at pixels p_1, p_2 and p_3 are similar. However, there is an edge in the middle suggesting that p_1 and p_2 belong to the same group while p_3 belongs to a different group. Below is the image computed by using the orientation energy [29]. The orientation energy somewhere on l_2 is strong which correctly proposes that p_1 and p_3 belong to two different partitions. (b) Examples from the spectral boundary detector [8]. Top: Original image and first four generalized eigenvectors. Bottom: Maximum response over orientations θ of $sPb(x, y, \theta)$ and of $sPb_{v_j}(x, y, \theta)$ for each eigenvector v_j.

Finally, the globalized probability of boundary , i.e. gPb, is then defined as:

$$gPb(x, y, \theta) = \sum_{i=1}^{9} \beta_i G_i(x, y, \theta) + \gamma sPb(x, y, \theta) \tag{8.5}$$

where the weights are learned by gradient ascent on the F-measure cost function

$$\text{F - measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{8.6}$$

2.2.3 Shape Descriptor The contours detected in the previous section define over-segmented regions of the image. Gu et al. [5] proposed a way to describe a region by evenly subdividing its bounding box into an $n \times n$ grid. The grid size n = 4 has been reported as effective. Each cell encodes information only inside the region. Different cues are extracted from each cell and each type of cue is encoded by concatenating cell signals into a histogram. The following region cues are included:

- Contour shape, given by the histogram of oriented responses of the contour detector gPb.
- Edge shape, computed by convolution with a [-1 0 1] filter along x and y axes. This captures high frequency information while it has been smoothed by gPb.

- Color, represented by the L*, a and b histograms in the CIELAB color space.
- Texture, described by texton histograms.

There are some advantages of this representation, which were pointed out by Gu et al. Firstly, the scale invariant nature of region descriptors enables us to compare regions regardless of their relative sizes. Secondly, background clutter interferes with region representations only mildly compared to interest point descriptors.

2.3 Discussion

So far, we have seen two ways to extract features from an image, each has its own primary purpose. SURF features were originally developed for the problem of image matching but they are also suitable for compact representation of local isotropic image patches. The advantage of using SURF for object detection is its local nature, thereby being invariant to great viewpoint changes having the capability of detecting deformable objects [9]. gPb, however, was developed to detect contours, hence being suitable for segmentation tasks [12] [5]. It can also be used for part-based object detection [9] [5].

3 Segmentation with Supervised Training

In image processing, it is very important that algorithms can get some knowledge from the tasks they are going to perform. The knowledge often comes from the training process using pre-defined datasets so that the algorithms can learn from our experiences. This section introduces some supervised learning techniques, that are often used in computer vision tasks.

3.1 Support Vector Machine

The support vector machine (SVM) invented by Cortes and Vapnik [13] is a non-probabilistic binary linear classifier, which predicts for each input data point which of the two possible classes the input belongs to. SVM maximizes the margin between positive and negative data points (as shown in Figure 8.4), thereby achieving very good generalization performance.

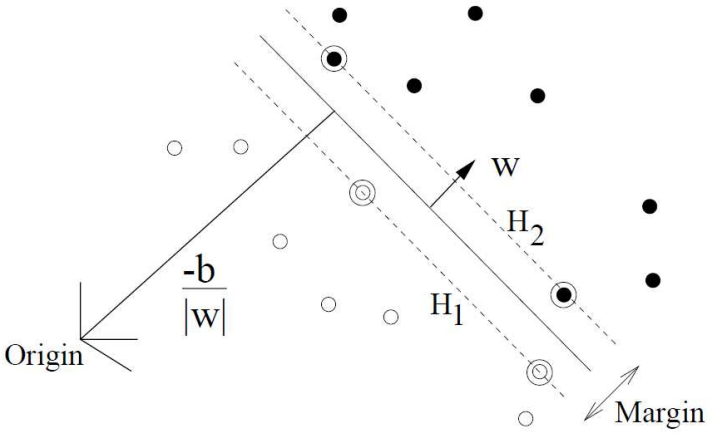


Figure 8.4: Linear separating hyperplanes for the separable case. The support vectors are circled. Image source: [14].

In this paper we only consider the case of linearly separable data. Suppose we have a dataset of S elements $\{(x_i, t_i)\}_{i=1}^{S}$, where $x_i \in R^d$ are training data points with the corresponding target values $t_i \in \{-1, 1\}$. Our goal is to find a hyperplane of the form $w^T x + b = 0$, which separates the data with maximal margin. The margin is formulated by defining d_+ and d_- to be the distances to the nearest positive and negative training examples respectively. We can always scale w and b such that

$d_- = d_+ = \frac{1}{\|w\|}$. Since the data is linearly separable the separating hyperplane must stay within the following region:

$$t_i(w^T x_i + b) \geq 1 \ \forall i \tag{8.7}$$

The equality in Equation 8.7 will hold exactly for the points on the margin. In order to maximize the margin $M = d_- + d_+ = \frac{2}{\|w\|}$ we need to minimize $\|w\|$. This can be formally stated as the following optimization problem: find the hyperplane satisfying

$$\arg\min_{w,b} \frac{1}{2}\|w\|^2 \tag{8.8}$$

subject to the constraint in Equation 8.7. This quadratic programming problem with linear constraints can be solved efficiently using Lagrange multipliers. The solution for w is obtained as a linear combination of training examples:

$$w = \sum_{i=1}^{S} a_i t_i x_i = \sum_{i=1}^{S} \alpha_i x_i, \tag{8.9}$$

where $a_i \geq 0$ is the Lagrange multiplier associate with the training data point x_i and $\alpha_i = a_i t_i$. Note that $a_i > 0$ only for training data points on the margin called support vectors. We get the solution for b by observing that any support vector x_j satisfies the equality condition in Equation 8.7:

$$\begin{aligned} t_j \left(\sum_{i=1}^{S} a_i t_i x_i^t x_j + b \right) &= 1 \\ b = t_j - \sum_{i=1}^{S} a_i t_i x_i^t x_j & \end{aligned} \tag{8.10}$$

In practice, it is more robust to average the solution for b over all support vectors. More detail on extensions to the case of non-separable data as well as non-linear support vector machines can be found in [14].

3.1.1 SVM for Region Ranking In this section we will formulate a SVM classifier in a way that it can reliably score a region of arbitrary shape according to how strongly it belongs to a particular object category. It is desirable that features extracted from local image regions can be combined additively to obtain the classifier response for a larger region. It has been proven [15] that a linear kernel SVM applied to a bag-of-features representation (recall the concept of bag-of-textures from Section 8.2(b)) has this additive property. Similar to the textons approach we obtain a vocabulary of K visual words by clustering a sample of local features (e.g. SURF) from the training images. We represent a region by its set of local features $R = \{(x_i, v_i)\}_{i=1}^{N}$, where x_i is the feature position and v_i the local descriptor. A histogram can be built from this bag-of-features, denoted $h(R)$, by mapping each feature v_i to its closest visual word d_i and recording the frequency of words in a K-dimensional vector.

A linear SVM classifier is learned from the segmented training examples according to the principle described in Section 3.1. The region ranking score is then given as follows:

$$\begin{aligned} f(R) &= b + w \cdot h(R) \\ &= b + (\sum_{i=1}^{S} \alpha_i h(R_i)) \cdot h(R) \\ &= b + \sum_{i=1}^{K} \sum_{j=1}^{S} \alpha_i h_j(R_i) h_j(R) \\ &= b + \sum_{j=1}^{K} w_j h_j(R) = b + \sum_{i=1}^{N} w_{d_i}, \end{aligned} \tag{8.11}$$

where $d_i \in [1, K]$ is the index of the visual word that feature v_i maps to. Note that we associate each j-word in the vocabulary with a weight $w_j = \sum_i \alpha_i h_j(R_i)$ to express $f(R)$ as a sum over per-feature contributions. This means the score of a region is the sum of its N features' word weights. Since we are only interested in maximizing the classifier response $f(R)$ the bias term b is irrelevant and can be ignored.

3.2 Logistic Regression

In the problem of two-class classification, the logistic regression method models the posterior probability of class C_1 by applying a logistic sigmoid function σ on a linear discriminant function of the feature vector ϕ so that

$$\begin{aligned} p(C_1|\phi) &= \sigma(w^T\phi) \\ p(C_2|\phi) &= 1 - p(C_1|\phi) \end{aligned} \tag{8.12}$$

where w is the weight vector of the linear discriminant function. The logistic sigmoid function σ is given as follows:

$$\sigma(a) = \frac{1}{1+e^{-a}} \tag{8.13}$$

Suppose we have a training data set $\{\phi_n, t_n\}_{n=1}^N$, where $t_n \in \{0,1\}$, $t = (t_1, ..., t_n)^T$. With $y_n = p(C_1|\phi_n)$ we can write the likelihood as:

$$p(t|w) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n} \tag{8.14}$$

The error function is then defined as the negative log-likelihood, which gives the cross-entropy error function in the form

$$E(w) = -\ln p(t|w) = -\sum_{n=1}^{N}(t_n \ln y_n + (1-t_n)\ln(1-y_n)) \tag{8.15}$$

Our goal is to learn the weight w such that it minimizes the cross-entropy error function. Due to the nonlinearity of the logistic sigmoid function there is no closed-form solution to this optimization problem. Fortunately, this error function is concave, and hence has a unique minimum. Moreover, this error function can be minimized by using the Newton-Raphson iterative optimization scheme, which uses a local quadratic approximation to the log likelihood function. The Newton-Raphson update, for minimizing a function $E(w)$, is given as follows:

$$w^{new} = w^{old} - H^{-1}\nabla E(w) \tag{8.16}$$

where H is the Hessian matrix, whose entries are the second derivatives of $E(w)$ with respect to the components of w. The gradient and Hessian of this error function are given by

$$\nabla E(w) = \sum_{n=1}^{N}(y_n - t_n)\phi_n \tag{8.17}$$

$$H = \nabla\nabla E(w) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T \tag{8.18}$$

Since the Hessian depends on w through y_n we need to apply Equation 8.16 iteratively, each time using the new weight vector w. Therefore, this algorithm is known as iterative reweighted least squares or IRLS.

3.3 Statistical Shape Model

To incorporate shape information into the process of segmenting an object in an image Leventon et al. [16] introduced a statistical shape model in which a prior on shape variation can be computed based on a set of training instances. Specifically, we need N^d samples in a d-dimensional image of size N to represent a shape. If we consider each voxel as one dimension, then a shape u is a point in a high dimensional shape space $u \in R^{N^d}$.

Given the training set $T = \{u_1, ..., u_n\}$, Our goal is to build a shape model over this distribution of surfaces. Since the shapes live in a high dimensional space with a lot of redundant dimension we can reduce the number of dimensions by using Principal Component Analysis (PCA), which only keeps the most relevant shape variances. Under the PCA transformation, a novel shape u can be written as a linear combination of k principal components:

$$u = \sum_{i=1}^{k} \alpha_k \hat{u}_k, \tag{8.19}$$

where α_k is a weighting coefficient corresponding to the k-th principal component $\hat{u}_k$. This means the shape u has been transformed into a k-dimensional vector through PCA. Finally, the Gaussian distribution of shapes living in the k-dimensional eigen-subspace is estimated resulting in a so-called Statistical Shape Model.

Figure 8.5(a) illustrates some training curves used to estimate the shape models of the corpus callosum. The original segmentations are shown as white curves overlaid on the signed-distance map. Figure 8.5(b) shows the means and three primary modes of variance of the shape distribution of the corpus callosum.

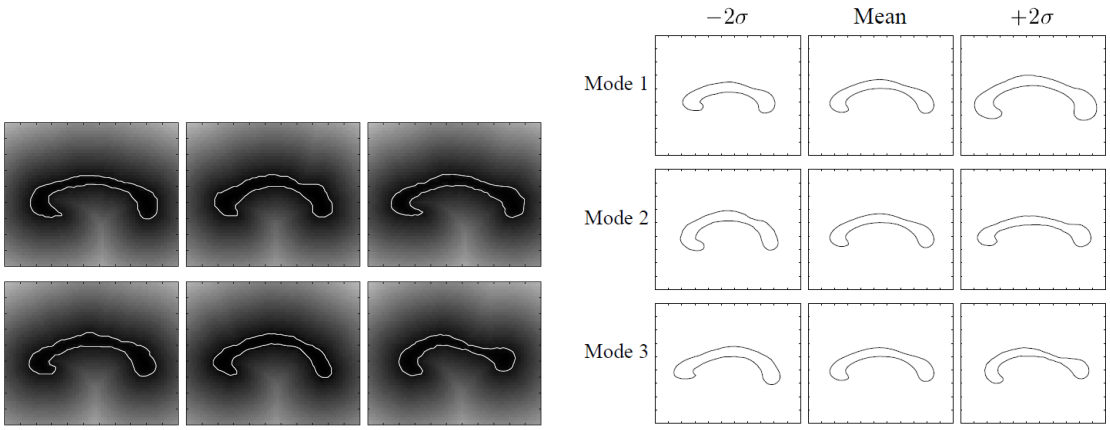


Figure 8.5: (a) some training signed distance maps of Corpus callosum overlaid by the original segmentations. (b) The three primary modes of variance of the corpus callosum training dataset.

4 Graph Construction

This section presents three different approaches to construct graphs that represent specific segmentation problems namely region graph for efficient region search problem, contour graph for identifying salient contours and Markov random field for the multi-shape graph-cut problem.

4.1 Region Graph

In this section we introduce a method to construct a region graph from an input image, which in turn serves as an input to a graph cut algorithm to find a sub-graph corresponding to the object of interest. This method was proposed by Vijayannarasimhan and Grauman in the object detection context [9]. We get the image regions with very high level of detail by applying the oriented watershed transform on the output signal of the gPb contour detector [12]. Particularly, a gradient image, which is generated by the gPb contour detector and can be seen as a topographic surface, is pierced at its minima and progressively immersed in water. The water fills the surrounding regions of the minima and forms lakes. When two lakes meet, the level of the water (the height of the saddle point) determines the saliency of the corresponding watershed arc.

The region graph $G = (V, E)$ is built with the set of vertices V being the oversegmented regions, also called superpixels, and the set of edges E connecting any two superpixels that share a boundary. A vertex is weighted with the region ranking formula given in Equation 8.11. Vijayannarasimhan considered two ways to weight a superpixel vertex depending on the type of feature used to construct the histogram $h^v(R)$:

- Point features: In this representation, each descriptor v_i is a SURF feature and the weight assigned to a superpixel vertex is the sum of the visual word weights for all local features located within that superpixel: $w(v) = \sum_{x_i \in v} w^v_{c_i}$.
- Shape features: In this case, each superpixel is mapped to a single shape descriptor described in Section 2.2.3. Thus, we get the vertex weight $w(v) = w^v_{c_i}$, where c_i is the single visual word associated with that superpixel's shape descriptor.

To avoid background regions incorrectly getting included into the result we consider edge weights between pairs of adjacent superpixels based on saliency measures of the wartershed arcs. Since we would like to compute the edge weights by ranking contours within a region just like we did for the vertex weights we introduce a bag-of-contour-strengths histogram vector that captures the statistics of the internal contours within an object. Intuitively, the contours between an object and its background are expected to be highly salient. Therefore, the weights should be learned in such a way that the scores of segmentations, that cross object boundaries, decrease.

Formally, the distribution of contour strengths $h^e(R)$ for an object region R is a histogram of L bins, where each bin represents a given range in the contour saliency measure. Having learned the edge weights w^e as described in the structured SVM learning framework [17], we can now extend the region ranking score in Equation 8.11 as follows:

$$f'(R) = \sum_{i=1}^{N} w^v_{c_i} - \sum_{j=1}^{M} w^e_{s_j} \tag{8.20}$$

where $s_j \in [1, L]$ is the bin index of $h^e(R)$ into which the contour strength of the j^{th} contour within the region falls and M denotes the total number of contours in the region R. Note that by subtracting the contour term $f'(R)$ returns lower scores for regions crossing strong object boundaries, thereby helping to exclude background regions from the optimal solution.

Having constructed a region graph from the input image our goal now is to find a subgraph R maximizing the region ranking score $f'(R)$. The best-scoring subgraph identifies the most likely region for the object of interest. This problem can be transformed into the Prize-Collecting Steiner Tree problem (PCST), which is defined as follows:

Definition 1 *PCST PROBLEM: Given a connected undirected vertex and edge weighted graph $G = (V, E, c, p)$ with vertex profits $p : V \to \mathbb{R}^{\geq 0}$ and edge costs $c : E \to \mathbb{R}^{\geq 0}$, find a connected subgraph $T = (V_T \subseteq V, E_T \subseteq E)$ of G that maximizes the profit:*

$$P(T) = \sum_{v \in V_T} p(v) - \sum_{e \in E_T} c(e) \tag{8.21}$$

Note that in the PCST problem both vertex profits and edge costs must be positive while our region ranking scores give both positive and negative values. To circumvent this issue we map $p(v) = w^v(v) - \mu$ and $c(e) = w^e(e) - \mu$, where μ being the minimum of both vertex and edge weights. The advantage of transforming the region graph into the PCST problem is that we can use a mathematical programming approach, in this case the branch-and-cut algorithm [18], to efficiently solve this problem and find the optimal solution. Section 5.1 discusses this problem in more detail. Figure 8.6 shows the entire pipeline of this region graph approach.

4.2 Contour Graph

In this section, we present a method to construct a directed contour grouping graph [19] based on which salient contours can be extracted from the otherwise 2D image clutter. Firstly, the output of an edge detector (e.g. gPb) is thresholded to obtain a discrete set of edge segments called edgels. A directed graph $G = (V, E, W)$ is then defined as follows. Graph nodes V correspond to all edgels. Since the edge orientation is ambiguous up to π, we duplicate every edgel into two copies i and $\bar{i}$ with opposite directions θ and $\theta + \pi$. Graph edges E include all the pairs of edgels within some distance r_e: $E = \{(i, j) : \|(x_i, y_i) - (x_j, y_j)\| \leq r_e\}$. Since every edgel is directed we connect each edgel i only to the neighbors in its direction. Graph weights W measure directed collinearity using the elastic energy between neighboring edgels, which describes how much bending is needed to complete a curve between i and j:

$$W_{ij} = e^{-(1-cos(|\phi_i|+|\phi_j|))/\sigma^2} \text{ if } i \to j \tag{8.22}$$

Here $i \to j$ means that j is in forward direction of i and σ is the scaling factor. $W_{ij} \geq 0$ implies that $W_{ji} = 0$. ϕ_i and ϕ_j denote the turning angles of i and j w.r.t. the line connecting them as shown in Figure 8.7(c).

In this graph, an ideal curve leads to two chains while random clutter produces fragmented clusters in the graph. Our goal is to detect such topological differences and extract 1D topological structures only.

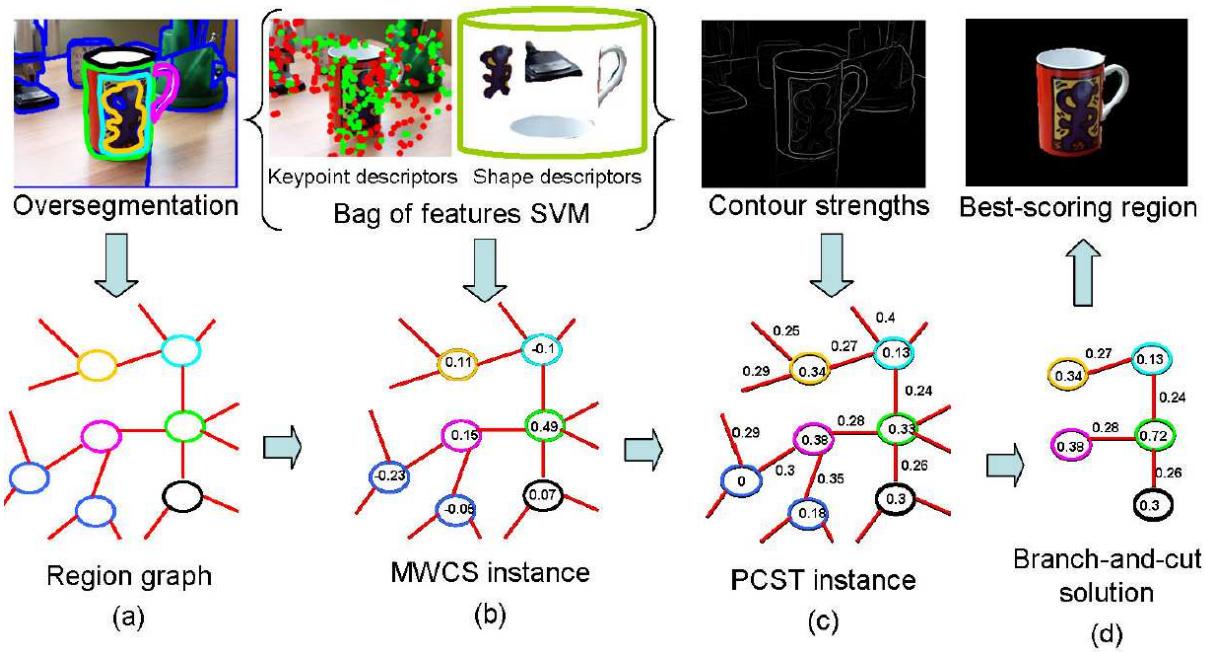


Figure 8.6: Region graph pipeline. (a) We oversegment the test image and construct a region-graph. (b) A region's node weight is its contribution to the classifier response. The optimal contiguous set of regions is equivalent to the maximum-weight connected subgraph problem (MWCS) on the vertex weighted region-graph. (c) We incorporate class-specific inter-region contour cues by adding edge costs leading to the PCST problem. (d) The best scoring region is obtained by efficiently solving the PCST instance with a branch-and-cut algorithm. Image source [9].

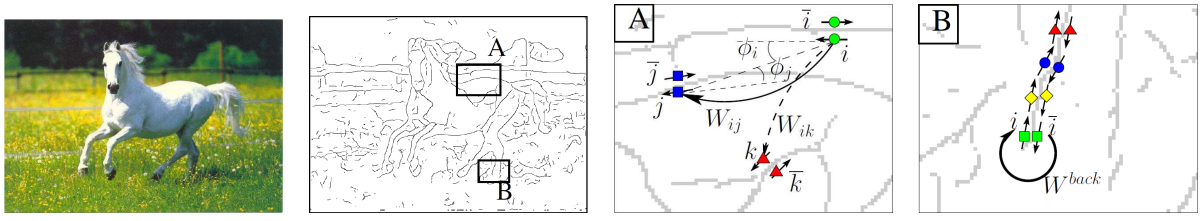


Figure 8.7: Directed graph for contour grouping. (a) Input image. (b) Edge map extracted from Pb. (c) Zoom-in view of graph connection in window A. Each edge node is duplicated in two opposite orientations. Oriented nodes are connected according to elastic energy and their orientation consistency. Here $W_{ij} \gg W_{ik}$. Salient contours form a 1D topological chain or cycle in this graph. (d) In window B, adding W^{back} to duplicated nodes i and $\bar{i}$ turns a topological chain into a cycle. Image source: [19].

To simplify the topological classification task and reduce the search to only cyclic structures we transform two duplicated chains into a cycle by adding a small amount of connection W^{back} between the duplicated nodes i and $\bar{i}$. For open contours, W^{back} connects the termination points back to the opposite direction to create a cycle as illustrated in Figure 8.7(d).

A contour (C, O) is defined by a set of vertices $C \subseteq V$ and a function $O : C \rightarrow \{1, ..., |C|\}$ which specifies a unique ordering of these vertices. To represent a contour (C, O) we must encode nodes which are part of the contour as well as the ordering of these points. This is accomplished by using a circular embedding where each node of the contour is mapped to a point on a circle about the origin in the complex plane and all other points are mapped to the origin. In this way, each point is represented as complex number

$$x_j = r_j e^{i\theta_j} \tag{8.23}$$

with $r_j = 1$ if $j \in C$ and 0 otherwise, and $\theta_j = O(j)\delta$ specifies the ordering with $\delta = \frac{2\pi}{|C|}$ is a phase step. The radius r_j of each point encodes whether it is part of the contour.

4.3 Markov Random Fields

Markov Random Fields (MRF) are undirected graphical models, which formalize and visualize the structure of a probabilistic model through a graph. In MRFs, the nodes correspond to a set of random variables $\{x_1, ..., x_n\}$, the edges represent soft constraints between them and the graph as a whole can be understood as a joint distribution $p(x)$ over the set of random variables. It is advantageous to express the joint distribution $p(x)$ as a product of functions defined over sets of variables that are local to the graph since this so-called factorized representation conveys interesting information about the properties of the class of distributions that the graph represents. We therefore need to define the appropriate notion of locality through a graphical concept called a clique. A clique is defined as a subset of nodes in a graph such that there exists a link between all pairs of nodes in the subset. In other words, the set of nodes in a clique is fully connected. Furthermore, a maximal clique is a clique such that it is not possible to include any other nodes from the graph in the set without it ceasing to be a clique. Figure 8.8(a) shows an example of cliques. Given a clique C, we denote the set of variables in that clique by x_C. Then the joint distribution

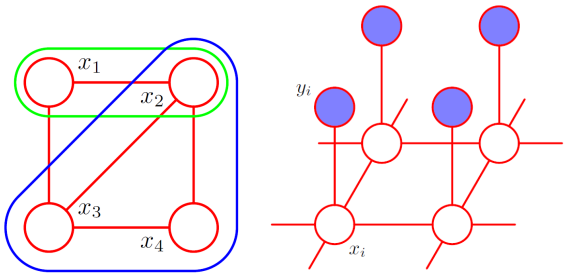


Figure 8.8: (a) A four-node undirected graph showing a clique (in green) and a maximal clique (in blue). Image source: [20]. (b) An undirected graphical model representing a Markov random field in which x_i is a variable denoting the hidden true state of pixel i and y_i denotes the corresponding value of pixel i in the observed image data. Image source: [20].

is written as a product of potential functions $\psi_C(x_C)$ over the maximal cliques of the graph

$$p(x) = \frac{1}{Z} \prod_C \psi_C(x_C) \tag{8.24}$$

Here the quantity Z, sometimes called the partition function, is a normalization constant and is given by

$$Z = \sum_x \prod_C \psi_C(x_C) \tag{8.25}$$

which ensures that the distribution $p(x)$ given by Equation 8.24 is correctly normalized. By considering only potential functions, which satisfy $\psi_C(x_C) \geq 0$, we ensure that $p(x) \geq 0$. It is convenient to express $\psi_C(x_C)$ as exponential functions so that

$$\psi_C(x_C) = \exp\left(-E(x_C)\right) \tag{8.26}$$

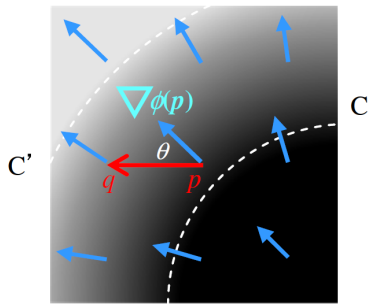


Figure 8.9: Gradient vectors of $\phi(p)$ and an angle θ between a gradient vector of $\phi(p)$ and a vector connecting points p and q. Image source: [21].

where $E(x_C)$ is called an energy function. The joint distribution is defined as the product of potentials and so the total energy is obtained by adding the energies of each of the maximal cliques.

In the field of image processing MRFs are used to capture for each pixel the correlation between the hidden state variable and the observed image data. The spatial coherent of the image content can also be modeled using the neighborhood relationships between pixel state variables as shown in Figure 8.8(b).

4.3.1 Shape Energy For the image segmentation problem our goal is to minimize the energy function associated with a MRF constructed from the training data and the observed input image. Particularly, given a set of labels $L = \{0, ..., n\}$ we need to find a labelling configuration $A = (A_1, ..., A_{|P|})$ for a set of voxels P that minimizes an energy $E(A)$ given by

$$E(A) = \lambda R(A) + B(A) = \lambda \sum_{p \in P} R_p(A_p) + \sum_{(p,q) \in \mathcal{N}} B_{p,q} \delta_{A_p \neq A_q} \tag{8.27}$$

where the set $\mathcal{N}$ is a collection of neighboring voxel pairs, the function δ is 1 if $A_p \neq A_q$ and 0 otherwise and λ is a balancing parameter. There are two types of energy terms in Equation 8.27. The first term $R_p(A_p)$ is a data term which expresses a penalty for assigning label A_p to voxel p. Normally, we use the negative log likelihood of the gray value for this term. The second term $B_{p,q}$ is a boundary term which punishes the assignment of labels A_p and A_q to the two neighboring voxels p and q. In the context of lung segmentation Shimizu and Nakagomi [21] [11] proposed a novel energy function incorporating shape energy term $S_{p,q}$ based on multiple shape priors learned with the statistical shape model described in Section 3.3:

$$E(A) = \lambda \sum_{p \in P} \{R_p(A_p) + NB_p(A_p)\} + \sum_{(p,q) \in \mathcal{N}} \{B_{p,q} + S_{p,q}\} \delta_{A_p \neq A_q} \tag{8.28}$$

$$S_{p,q} = \min \left(\sqrt{\frac{1 - \cos(\theta_{A_p})}{2}}, \sqrt{\frac{1 - \cos(\theta_{A_q})}{2}} \right) \tag{8.29}$$

where θ_{A_p} represents an angle between a vector connecting voxels p and q and a gradient vector of a signed distance $\phi_{A_p}(p)$ from the boundary of a shape corresponding to a label $A_p \in L$ as shown in Figure 8.9. $NB_p(A_p)$ is defined by the distance from the dorsal ribs. This multiple shape MRF problem can be solved using the combination of fusion move and min-cut algorithms explained in Section 5.3.

5 Graph Cut Algorithms

This part will concentrate on some of the state-of-the-art graph-cut algorithms, which can find a cut that separates the optimal subgraph from those kinds of graphs constructed in Section 4.

5.1 Prize-Collecting Steiner Tree

In this section, we discuss a method to solve the Prize-Collecting Steiner Tree (PCST) problem [18] of finding a connected subgraph $T = (V_T, E_T)$ that maximizes the profit function defined in Section 4.1.

This maximization problem can be transformed into the problem of finding a subtree $T = (V_T, E_T)$ that minimizes the following function:

$$GW(T) = \sum_{v \notin V_T} p(v) + \sum_{e \in E_T} c(e) \tag{8.30}$$

where $GW(T)$ is the cost function for the subtree T. Here, $p(v)$ is interpreted as penalty for not connecting a vertex v. Furthermore, we can formulate this problem by means of an integer linear program (ILP) by transforming the reduced graph $G' = (V', E', c', p')$ resulting from the application of preprocessing into a directed edge-weighted graph $G_{SA} = (V_{SA}, A_{SA}, c'')$, which is called Steiner arborescence. The vertex set $V_{SA} = V' \cup \{r\}$ contains the vertices of the input graph G' and an artificial root vertex r. The arc set A_{SA} contains two directed arcs (i, j) and (j, i) for each edge $(i, j) \in E'$ plus a set of arcs from the root r to the customer vertices $R_{SA} = \{i \in V' | p'_i > 0\}$, which are the vertices having positive profit. The cost vector c'' is defined as follows:

$$c''_{ij} = \begin{cases} c'_{ij} - p'_j & \forall (i,j) \in A_{SA}, i \neq r \\ -p'_j & \forall (r,j) \in A_{SA}. \end{cases} \tag{8.31}$$

To model the presence or absence of each vertex or edge in the solution T_{SA} we introduce variable vectors $x \in \{0,1\}^{|A_{SA}|}$ and $y \in \{0,1\}^{|V_{SA}|-1}$ with the following interpretation:

$$x_{ij} = \begin{cases} 1 & (i,j) \in T_{SA} \\ 0 & \text{otherwise} \end{cases} \forall (i,j) \in A_{SA}, \quad y_i = \begin{cases} 1 & i \in T_{SA} \\ 0 & \text{otherwise} \end{cases} \forall i \in V_{SA}, i \neq r. \tag{8.32}$$

Given a set of vertices $S \subset V_{SA}$ and its complement $\bar{S} = V_{SA} \backslash S$ we define two directed cuts: $\delta^+(S) = \big\{(i,j) | i \in S, j \in \bar{S}\big\}$ and $\delta^-(S) = \big\{(i,j) | i \in \bar{S}, j \in S\big\}$. We also write $x(A) = \sum_{ij \in A} x_{ij}$ for any subset of arcs $A \subset A_{SA}$. The corresponding ILP model then reads as follows:

$$\text{(CUT)} \qquad \min \sum_{ij \in A_{SA}} c''_{ij} x_{ij} + \sum_{i \in V_{SA}} p'_i \tag{8.33}$$

$$\text{subject to} \qquad \sum_{ji \in A_{SA}} x_{ji} = y_i \qquad \forall i \in V_{SA} \backslash \{r\} \tag{8.34}$$

$$x(\delta^-(S)) \geq y_k \qquad k \in S, r \notin S, \forall S \subset V_{SA} \tag{8.35}$$

$$\sum_{ri \in A_{SA}} x_{ri} = 1 \tag{8.36}$$

$$x_{ij}, y_i \in \{0,1\} \qquad \forall (i,j) \in A_{SA}, \forall i \in V_{SA} \backslash \{r\} \tag{8.37}$$

Note that the profits from the vertices of the solution have been absorbed into the cost formulation. Therefore, in Equation 8.33 we only penalize those vertices not getting included into the solution. The condition in Equation 8.34 is required to ensure a tree structure of the solution. The cut constraints in Equation 8.35 guarantee that for each vertex v in the solution there must be a directed path from r to v. The constraint in Equation 8.36 is used to ensure the existence and uniqueness of a connection to the root.

In order to create a bijection between arborescence and PCST solutions we introduce the so-called asymmetry constraints:

$$x_{rj} \leq 1 - y_i, \forall i < j, i \in R \tag{8.38}$$

These inequalities assure that for each PCST solution the customer vertex adjacent to the root is the one with the smallest index. Note furthermore that in every non-customer vertex, which is not a branching vertex in the Steiner arborescence, the in-degree and the out-degree must be equal, whereas in a branching non-customer vertex the in-degree is always less than the outgoing degree. Thus, we have the following flow-balance constraints:

$$\sum_{ji \in A_{SA}} x_{ji} \leq \sum_{ij \in A_{SA}} x_{ij}, \qquad \forall i \notin R, i \neq r. \tag{8.39}$$

This ILP problem can be solved efficiently in practice with a branch-and-cut algorithm.

5.1.1 Branch-and-Cut Algorithm The branch-and-cut algorithm is a very successful technique for solving a wide variety of integer programming problems to optimality. In essence, it is a combination of a cutting plane method with a branch-and-bound algorithm. These methods work by solving a sequence of linear programming relaxations of the integer programming problem. Cutting plane methods improve the relaxation of the problem to more closely approximate the integer programming problem and Branch-and-bound algorithms break the problem into sub-problems based on the domain of its variables and proceed by pruning the branches of sub-problems having optimal solutions worse than the bounding value.

We consider a general ILP problem with n variables and m constraints stated as follows

$$\min \left\{ c^T x : Ax \geq b, x \in \mathbb{Z}_+^n \right\} \tag{8.40}$$

where $x \in \mathbb{Z}_+^n$ is a vector of integer decision variables, $c \in \mathbb{Z}^n$ is the objective function vector, $b \in \mathbb{Z}^m$ is the right hand side vector and $A \in \mathbb{Z}^{m \times n}$ is the matrix of constraint coefficients.

The LP-relaxation (CUT) is obtained by replacing the integrality requirements by their corresponding real-value constraints:

$$\min \left\{ c^T x : Ax \geq b, x \in \mathbb{R}_+^n \right\} \tag{8.41}$$

Its feasible region is the polyhedron:

$$P = \left\{ x \in \mathbb{R}_+^n : Ax \geq b \right\} \tag{8.42}$$

The integral hull is the convex hull of the set of integer solutions, i.e., the polyhedron:

$$P_I = \text{conv} \left\{ x \in \mathbb{Z}_+^n : Ax \geq b \right\} \tag{8.43}$$

We define a cutting plane as a linear inequality that is valid for P_I but not for P as shown in Figure 5.1.1. Following the tutorial of Mitchell [22] the branch-and-cut algorithm is outlined as follows:

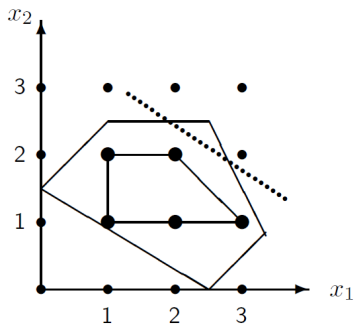


Figure 8.10: A cutting plane cuts through the polygon P while trying to get as close to P_I as possible.

1. Initialization: Denote the initial integer programming problem by ILP^0 and set the active nodes to be $L = \left\{ ILP^0 \right\}$. Set the upper bound to be $\bar{z} = +\infty$ and set the lower bound $\underline{z}_l = -\infty$ for the problem $l \in L$.

2. Termination: If $L = \emptyset$, then the solution x^* which yielded the incumbent objective value $\bar{z}$ is optimal. If no such x^* exists (i.e., $\bar{z} = +\infty$) then ILP is infeasible.

3. Problem selection: Select and delete a problem ILP^l from L.

4. Relaxation: Solve the linear programming relaxation of ILP^l. If the relaxation is infeasible, set $\underline{z}_l = +\infty$ and go to Step 6. Let $\underline{z}_l$ denote the optimal objective value of the relaxation if it is finite and let x^{lR} be an optimal solution; otherwise set $\underline{z}_l = -\infty$.

5. Add cutting planes: If desired, search for cutting planes that are violated by x^{lR}; if any are found, add them to the relaxation and return to Step 4.

6. Fathoming and Pruning:

(a) If $\underline{z}_l \geq \bar{z}$ go to Step 2.

(b) If $\underline{z}_l < \bar{z}$ and x^{lR} is integral feasible, update $\bar{z} = \underline{z}_l$, delete from L all problems with $\underline{z}_l \geq \bar{z}$, and go to Step 2.

7. Partitioning: Let $\left\{S^{lj}\right\}_{j=1}^{j=k}$ be a partition of the constraint set S^l of problem ILP^l. Add problems $\left\{ILP^{lj}\right\}_{j=1}^{j=k}$ to L, where ILP^{lj} is ILP^l with feasible region restricted to S^{lj} and $\underline{z}_{lj}$ for $j = 1, ..., k$ is set to the value of $\underline{z}_l$ for the parent problem l. Go to Step 2.

5.2 Contour Cut

Given a weighted directed graph $G = (V, E, W)$ a random walk on G is a Markov process with transition matrix $P = D^{-1}W$, where $D = \text{diag}(\sum_u W_{vu})$ is a diagonal matrix with the row-sums of W along the diagonal. Let π be the unique left eigenvector such that $\pi P = \pi$. The row-vector π corresponds to the stationary distribution of the random walk. This means:

$$\pi_u = \sum_{v, v \to u} \pi_v P_{vu}, \tag{8.44}$$

that is, the probability of finding the random walk at u is the sum of all the incoming probabilities from vertices v that have a directed edges to u.

A circulation on a directed graph G is defined as follows.

Definition 2 *A matrix $F \in (\mathbb{R}^+ \cup \{0\})^{|V| \times |V|}$ that assigns each directed edge to a non-negative value is called a circulation if*

$$\sum_{u, u \to v} F_{uv} = \sum_{w, v \to w} F_{vw} \tag{8.45}$$

for each vertex v.

One interpretation of a circulation is a flow in the graph. The flow at each vertex must be conserved, hence, the flow in is equal to the flow out. It is easy to show that $F = \Pi P$ is a circulation in the graph G, where $\Pi = \text{diag}(\pi)$. This is because the conservation property follows directly from the stationarity property of π:

$$\sum_{u, u \to v} F_{uv} = \sum_{u, u \to v} \pi_u P_{uv} = \pi_v \cdot 1 = \sum_{w, v \to w} \pi_v P_{vw} = \sum_{w, v \to w} F_{vw} \tag{8.46}$$

This circulation is advantageous because if W is symmetric then the directed versions of cut and volume that we will define reduce to the original undirected versions [23].

5.2.1 Contour Cut Cost We define the external cut of a contour (C, O) to measure its separation from the rest of the graph, $V \backslash C$:

$$\text{Ecut}(C) = \sum_{i \in C, j \notin C} F_{ij} \tag{8.47}$$

The internal cut is used to measure the entanglement caused by graph edges within the contour that violate the ordering O. Intuitively, the internal cut measures how much the contour deviates from an ideal one-dimensional contour toward a 2-dimensional clique. Let $k \in \mathbb{Z}^+$ be the width of the contour. Nodes $i, j \in C$ with $|O(i) - O(j)| > k$ are beyond the width of the contour, hence they are included into the internal cut:

$$\text{Icut}(C, O) = \sum_{\{i,j\} \subseteq C, |O(i) - O(j)| > k} F_{ij} \tag{8.48}$$

Having defined the external and internal cuts, we can now define the cost function for the contour cut as follows:

$$\text{Ccut}(C, O) = \frac{\text{Icut}(C, O) + \text{Ecut}(C)}{\text{Vol}(C)} \tag{8.49}$$

where $\text{Vol}(C) = \sum_{i \in C, j \in E} F_{ij}$ is the sum of the weights of all edges incident with the contour. This cost function will be small for contours having small internal and external cuts.

The internal and external cuts can be encoded with respect to the circular embedding. Given a the circular embedding of a contour, $x \in \mathbb{C}^{|C|}$, the external cut is:

$$\text{Ecut}(x) = \sum_{(i,j)\in E: r_i \neq 0, r_j = 0} F_{ij} = \sum_{(i,j)\in E} F_{ij} r_i (1 - r_j) \tag{8.50}$$

Instead of using the hard-bound of k for the internal cut, a soft version of the internal cut is applied by using the cosine function:

$$\text{Icut}(x) = \sum_{\{i,j\}\subseteq C} F_{ij} r_i r_j (1 - \cos(\theta_j - \theta_i - \delta)) \tag{8.51}$$

The volume of the contour is defined as

$$\text{Vol}(x) = \sum_{(i,j)\in E} F_{ij} r_i \tag{8.52}$$

The contour cut in terms of the circular embedding x is then

$$Ccut(x) \quad = \quad \frac{Icut(x) + Ecut(x)}{\text{Vol}(x)} = ... = \frac{x^*(\Pi - H(\delta))x}{x^*\Pi x} = R_{\Pi - H(\delta),\Pi}(x), \tag{8.53}$$

where $H(\delta) = \frac{Fe^{-i\delta} + F^T e^{i\delta}}{2}$ and $R_{\Pi - H(\delta),\Pi}(x)$ is the generalized Rayleigh quotient. It follows that the problem of minimizing $Ccut(x) = R_{\Pi - H(\delta),\Pi}(x)$ is equivalent to maximizing $R_{H(\delta),\Pi}(x)$.

5.2.2 Computational Solution In order to minimize the contour cut, we would like to solve

$$\max_x \frac{x^* H(\delta) x}{x^* \Pi x} \tag{8.54}$$

$$\text{such that} \qquad x_i = r_i e^{i\theta_i}, \qquad r_i \in \{0, 1\}, \qquad \theta_i = O(i)\delta,$$

where we seek not just the global maximum but all critical points, which correspond to different contours. By requiring r_i and θ_i to take on discrete values the problem is computationally infeasible since it requires searching not only over an exponential number of subsets of vertices but also over orderings on each of these sets. We relax the problem by allowing x to take on arbitrary complex values: $x \in \mathbb{C}^{|C|}$. Note that we now must maximize over δ in addition to x. The main result for solving the problem is based on the following theorem:

Theorem 5.1 *The critical points of the relaxed contour cut problem*

$$\max_{x,\delta} \frac{x^* H(\delta) x}{x^* \Pi x} \qquad s.t. \qquad x_i \in \mathbb{C} \tag{8.55}$$

can be found by searching over δ and finding the eigenvectors of the corresponding matrices $\Pi^{-1} H(\delta)$; any eigenvectors for which $x^* H(\delta) x = |x^* F x|$ are critical points with respect to both x and δ.

The proof for this theorem can be found in [10]. In practice it might be difficult to search over all values of δ with a small enough step size such that $x^* H(\delta) x = |x^* F x|$ to a high enough precision. Instead a different algorithm is used based on the observation that as δ changes, the eigenvalues of $H(\delta)$ rarely cross. To determine the critical points with respect to δ, we make the assumption that the k^{th} largest eigenvalue of $H(\delta)$ remains the k^{th} largest over all values of δ, in which case it suffices to directly find the local maxima of the k^{th} eigenvalue over all δ. This leads to the following algorithm:

5.3 Graph-Cut

Given an undirected graph constructed from Section 4.3 we convert it to a directed weighted graph by introducing terminal nodes corresponding to the set of labels that can be assigned to pixels. For the two-class problem there are two terminal nodes called the source s and the sink t. For each node in the original graph we insert a directed edge from the source to it and another directed edge from that node to the sink with the edge weights being the data term in Equation 8.28 evaluated at the label values of

Algorithm 1 Contour cut

Calculate matrices P and Π from W.

for $\delta = \delta_{min}$ *to* δ_{max} **do**

$$H(\delta) \leftarrow \frac{\Pi P e^{-i\delta} + P^T \Pi e^{i\delta}}{2}$$

Solve $\Pi^{-1} H(\delta) x = \lambda x$ for top k eigenvectors.

end for

Find local maxima of each λ_i over all δ. The associated x's are critical points with respect to both x and δ.

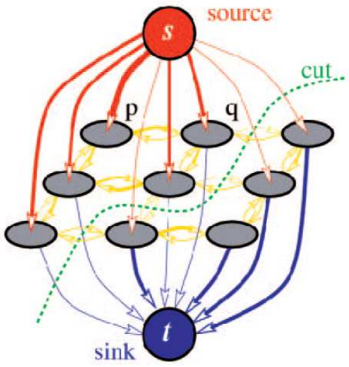


Figure 8.11: Example of a directed capacitated graph. Edge costs are reflected by their thickness. Image source: [24].

the source or the sink accordingly. Each existing edge from the original graph is replaced by 2 directed edges with the weights being the boundary term. Figure 8.11 shows an example of a two-terminal graph that can be used to minimize the energy function as in Equation 8.28 on a 3×3 image with two labels.

An s/t cut C on a graph with two terminals is a partitioning of the nodes in the graph into two disjoint subsets S and T such that the source s is in S and the sink t is in T. The cost of a cut $C = \{S, T\}$ is defined as the sum of the costs of boundary edges (p, q), where $p \in S$ and $q \in T$. One of the fundamental results in combinatorial optimization is that the minimum s/t cut problem can be solved by finding a maximum flow from the source s to the sink t. Loosely speaking, maximum flow is the maximum amount of water that can be sent from the source to the sink by interpreting graph edges as directed pipes with capacities equal to edge weights.

The max-flow problem can be solved in polynomial time by using the augmenting paths-based algorithm, which pushes flow along non-saturated paths from the source to the sink until the maximum flow in the graph G is reached. A typical augmenting path algorithm stores information about the distribution of the current $s \rightarrow t$ flow f among the edges of G using a residual graph G_f. The topology of G_f is identical to G, but the capacity of an edge in G_f reflects the residual capacity of the same edge in G given the amount of flow already in the edge. At the initialization, there is no flow from the source to the sink ($f = 0$) and edge capacities in the residual graph G_0 are equal to the original capacities in G. At each new iteration, the algorithm finds the shortest $s \rightarrow t$ path along non-saturated edges of the residual graph. If a path is found, then the algorithm augments it by pushing the maximum possible flow df that saturates at least one of the edges in the path. The residual capacities of edges in the path are reduced by df while the residual capacities of the reverse edges are increased by df. Each augmentation increases the total flow from the source to the sink $f = f + df$. The maximum flow is reached when any $s \rightarrow t$ path crosses at least one saturated edge in the residual graph G_f.

Note that a minimum $s - t$ cut in G yields a global minimum of the energy E. However, the global minimum can only be reached if the energy function satisfies the submodular property:

$$E(s, s) + E(t, t) \leq E(s, t) + E(t, s) \tag{8.56}$$

where s, t are the labels of the corresponding source or sink. In the non-submodular case, Kolmogorov and Rother propose a quadratic pseudo-boolean optimization method (QPBO) which produce optimal solution based on the roof duality and the probing methods. More detail can be found in [25].

5.3.1 Multi-Shape Graph-Cuts In order to solve the multi-label problem posed in Section 4.3 we use an approximate algorithm called fusion move [26], which iteratively solves a series of problems between a set of current labels and a set of proposed labels. In essence, we define a fusion move operator which selects labels from either of the two input solutions such that the energy of the output solution is minimized. This is done by using the QPBO min-cut algorithm.

Let X^{cur} be a collection of current labels and X^{pro} be a set of proposed labels. We divided the current labels into object labels and background labels. First, the algorithm proposes an object label l against voxels whose current labels X^{cur} are background and proposes the same labels as current for other voxels ($X^{pro} \leftarrow X^{cur}$). Then we apply the fusion move operator on X^{cur} and X^{pro} to find the combinatorial optimal solution of labels or shape. This step can reduce false negatives but cannot deal with false positives. The second step focuses on false positives based on the same analogy. It proposes a background label l against voxels whose current labels X^{cur} are object. To compute the shape energy $S_{p,q}$ in the case of a background proposal, we extend the set of labels L to $L = \{1, -1, 2, -2, ..., n, -n\}$, where the absolute value of a label refers to the number of a shape and its sign represents the state of a voxel. Here, positive labels represent object labels and negative labels represent background labels. The fusion move operator can again find combinatorial optimal solutions of labels. We iterate a pair of object proposals and background proposals by changing a label or a shape. The algorithm stops the iteration process when no label changes or when it reaches the maximum number of iterations defined by the user. The method is summarized in Algorithm 2.

Algorithm 2 Multi-shape graph-cuts

Require: a set of labels L, X^{cur}
Ensure: segmented regions
  Initialize $X^{cur} = \{-1, ..., -1\}$
  repeat
    for each $l \in L$ **do**
      Generate a set of labels X^{pro} where either l or an element of X^{cur} is assigned at each voxel
      $X^{cur} \leftarrow X^{cur} \odot X^{pro}$ $\odot$: fusion move operator [26]
    end for
  until Convergence: no label changes or reaches maximum number of iterations

6 Applications

This section is devoted to illustrate some applications of the algorithms described above to real-world image segmentation as well as in the field of medical image processing. The results are often compared using the precision vs. recall graph. Precision is the ratio of the number of true positive members to the total number of members predicted as positive:

$$\text{Precision} = \frac{tp}{tp + fp} \tag{8.57}$$

where tp is the number of true positive members and fp is the number of false positive members. Recall is the ratio of the number of true positive members to the actual number of positive members:

$$\text{Recall} = \frac{tp}{tp + fn} \tag{8.58}$$

where fn is the number of actually positive members predicted as negative members. The higher the precision vs. recall graph the better the result is. An *average precision* (AP) over all recall range is a single number also used to compare the results.

6.1 Efficient Region Search for Object Detection

In this section we describe the efficient region search (ERS) system developed by Vijayanarasimhan and Grauman [9]. Given training images in which the spatial extent of the object of interest is marked at the pixel level an additive classifier is trained to distinguish that object category from any other. Given a new image we oversegment it into subregions, extract the image features associated with each subregion

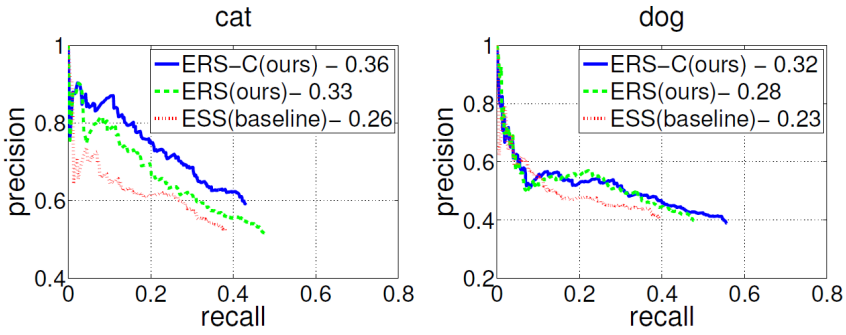


Figure 8.12: Detection accuracy on PASCAL VOC 07 objects. Numbers in legends report mean AP.

and pre-compute its resulting contribution to the classifier response. This yields a vertex-weighted region-graph, where the weights are the components of the classifier score. To incorporate inter-region contour cues we introduce learned edge costs into the graph leading to the PCST problem, which can be solved with a branch-and-cut algorithm. The resulting detection method rapidly determines the region within the image that yields the maximal classifier response.

The ERS system is compared with the two most relevant existing methods: The state-of-the-art efficient subwindow search (ESS) method [15], which efficiently provides the best result possible with bounding box search and the global connectivity conditional random field (CRF) model of [27], which efficiently provides an approximate region-based solution. Three datasets are used to test the system: The PASCAL VOC 2007, the ETHZ Shapes and the PASCAL VOC 2008. For the PASCAL VOC 2007, 659 cat images and 839 dog images are used to test the ERS system. 50% of the data are used for training and 50% are used for testing. The ETHZ Shape dataset consists of 255 images of five shape categories (Applelogos, Bottles, Mugs, Giraffes, Swans). Half the examples per category are used for training, the rest for testing. All 20 categories of 10057 images in the PASCAL VOC 2008 segmentation dataset are used to compare the ERS system with the CRF model.

For point features the system uses SURF extracted at Canny edge points and quantizes the training points into $K = 1000$ visual words using K-means. To construct the region-graph 100 regions per image are extracted by using the technique in [12]. For shape features, each region is described with a 4×4 spatial grid scaled to the region size, where each cell bins the normalized gPb and Canny edge responses according to their orientations. To collect responses at multiple scales, the responses from blurring the gPb map with Gaussians of two scales ($\sigma = 5, 10$) are also collected.

Figure 8.12 compares the accuracy of ERS versus ESS on the PASCAL VOC 2007 dataset. For both categories, maximizing the classifier response over the region-graph (ERS) yields much better accuracy than rectangular windows (ESS). In addition, including the edge costs (ERS-C) further boosts precision. Even under the PASCAL bounding box metric, the ERS method is about 70%more accurate than ESS (mean AP of 27.2 for ERS, compared to 16.0 for ESS).

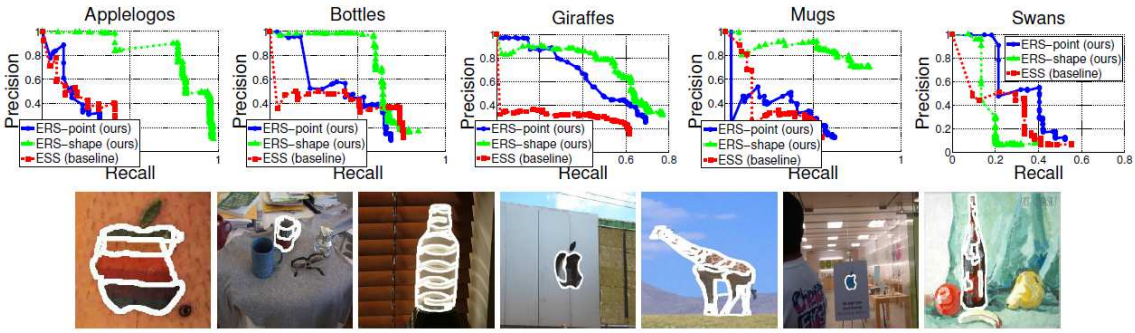


Figure 8.13: Detection accuracy (top row) and example detections by the ERS method (bottom row) on the ETHZ categories. Image source: [9].

Figure 8.13 shows the results on ETHZ where the ERS method is applied using either the point features (blue curves) or the shape features (green curves). ESS is only applicable to the point features

(red curves). Overall, the mean AP of ERS is from 9% to 90% better than ESS using the same features. Using shape features, we see dramatic gains for almost all objects, since the ETHZ objects have parts well-described by their shape (e.g., mug handle, bottle neck). Again, even with the bounding box metric the ERS approach is 19% better than ESS (30.1 vs. 25.3 mean AP).

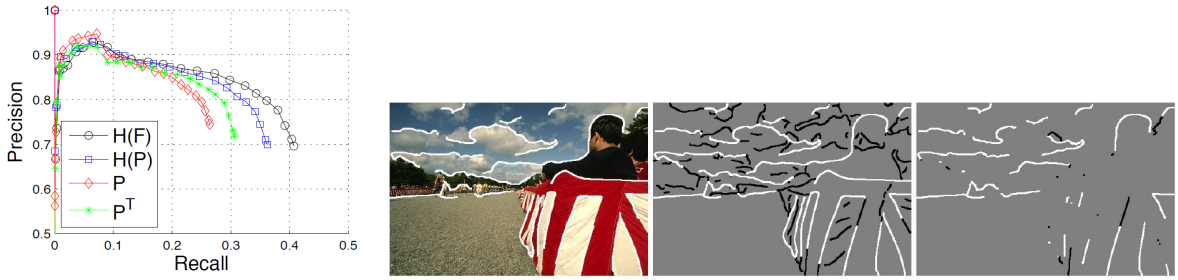
Finally, the ERS method is compared with the CRF method on the PASCAL VOC 2008 dataset based on the amount of overlapping with the ground truths criterion. The ERS approach outperforms the baseline on 17 out of 20 categories and improves the mean overlap accuracy (0.274 vs. 0.228).

Regarding the computation time on average the ERS method converges in 0.29 seconds, which is similar to ESS. The longest time taken for any single test image was 5.8 secs.

6.2 Salient Contour Detection

In this section we demonstrate the result from the contour cut algorithm developed by Kennedy et al. [10]. The contour cut approach first identifies edge segments (known as edgels) from the output of a contour detector, it then constructs a contour grouping graph whose vertices are the edgels duplicated in opposite directions and edges connecting the edgels within a small radius. Cycles are formed by connecting the duplicated nodes. The contour cut cost function is formulated based on the internal and external cuts normalized by the volume of the contour. The minimization of the contour cut cost can be formulated as a Hermitian eigenvalue problem as stated in Theorem 5.1.

The algorithm is tested on the Berkeley Segmentation dataset consisting of 300 images, of which 200 images are used for training and 100 images are used for testing. We compare the algorithm, which finds the exact critical points of Equation 8.55 by searching over all δ to the algorithms that give approximations by finding the right eigenvectors of P and the left eigenvectors of P scaled by Π^{-1}. The algorithms we compare are $H(F)$, $H(P)$, P, and P^T, where the name designates which matrix is used in the eigenvector computation. Specifically, $H(F)$ is the algorithm, which finds the exact critical points of Equation 8.55 using the circulation matrix $F = \Pi P$; $H(P)$ is the exact algorithm using P rather than F; P is the approximation given by the right eigenvectors of P; P^T is the approximation given by the left eigenvectors of P and scaled by Π^{-1}. Note that P is the algorithm given by Zhu [19]. We also compare between using the full graph and clustering the nodes to $n = 1000$ clusters using normalized cut [28] and running the algorithms on this reduced graph. A comparison between algorithms is shown in Figure 8.14(a).



(a) (b)

Figure 8.14: (a) Comparison of algorithms for the Berkeley Segmentation dataset with $n = 1000$, top 20 contours. (b) Results of $H(F)$ on various images using an aggregated graph with $n = 1000$ vertices. The top 20 contours, as ranked by the cost function (Equation), are plotted. Observe that the contour cut algorithm ($H(F)$) finds significantly longer contours than Zhu′s algorithm (P). Left: image with extracted contours. Middle: all thresholded Pb edges (black) and contours (white). Right: difference between $H(F)$ and P. Contours only $H(F)$ found are white and contours only P found are black. Image source [10].

Using a graph of size $n = 1000$, the precision values are similar for low recall, but for higher recall the algorithm $H(F)$ outperforms the other algorithms. Because all algorithms are producing the same number of contours, this indicates that $H(F)$ is producing longer contours, which is a better result. Figures 8.14(b) shows that $H(F)$ finds notably longer contours than P.

Figure 8.15 shows a comparison between the contour cut algorithm and the Pb contour detector. Since the contour cut algorithm begins with Pb as an input it will converge to Pb as more contours are used. The contour cut algorithm outperforms Pb in the low-recall range and so is able to pick out the best salient contours in an image.

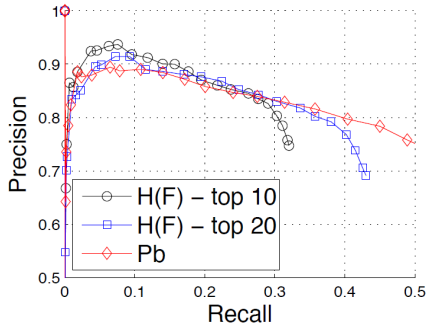


Figure 8.15: Comparison of the contour cut algorithm to the *Pb* algorithm on the Berkeley Segmentation dataset.

6.3 Multi-Shape Graph-Cut for Lung Segmentation

This section presents results from the multi-shape graph-cuts algorithm proposed by Nakagomi et al. [11] that can solve the problem of a combination of multiple shape priors. First, patient-specific shape priors are estimated by combining a statistical shape model and a pre-segmented result that is obtained by thresholding and morphological operations. Shapes similar to a pre-segmentation result are extracted from an eigen shape space. Particularly, the top n shapes with a minimum distance between a shape in an eigen shape space and a pre-segmented region are selected as the shape priors and used in the proposed multi-shape graph-cuts Algorithm 2.

The algorithm is tested on the dataset of 97 cases with pulmonary diseases, such as lung carcinoma. These cases include both non-contrast and contrast CT volumes (image size: 512 × 512 × 204-561[voxel], pixel size: 0.625-0.741[mm/pixel], slice spacing: 0.5-1.0 [mm], bits stored: 16 [bit]). The algorithm uses a 26-neighborhood system for the graph-cuts. 49 CT images are used for training and 48 for testing. The algorithm is compared with the single-shape graph-cuts approach.

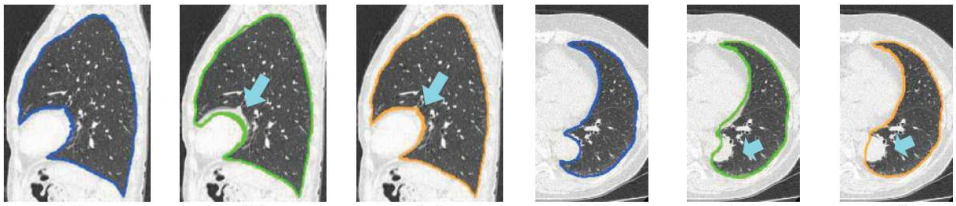


Figure 8.16: Examples of segmentation results. (a) and (d): true boundaries, (b) and (e): single-shape graph-cuts, (c) and (f): multi-shape graph-cuts. Image source: [11].

Figure 8.16 (a)-(c) show cases in which the single-shape graph-cuts were inferior to the proposed method as a result of false positives of the heart region, as indicated by an arrow. In order to evaluate the results quantitatively we compare average distances between an extracted surface and a manually delineated surface. The average distance of this case is improved from 0.835 to 0.523 [mm]. Figure 8.16 (d)-(f) show failure cases, using both methods, in which the average distances are 0.862 [mm] for the single-shape graph-cuts and 0.732 [mm] for the multi-shape graph-cuts.

Figure 8.17 shows the average and standard deviation of the average distance over all testing data, which was improved from 0.719 ± 0.309 to 0.587 ± 0.176 [mm]. Figure 6.3 presents the shape priors of Figure 8.16 (d)-(f). Since the shape of the aorta region of this case is unique and differs from those of statistical shape model training data it was impossible to estimate the shape correctly. Consequently, there was no shape among the set of proposed shapes that resembled the aorta. The failure of Figure 8.16 (e) and (f) could be explained by the above reasons.

The computational time of the multi-shape graph-cut algorithm was approximately 30 min per CT volume (Intel(R) Core(TM) i7 3.07 GHz × 2).

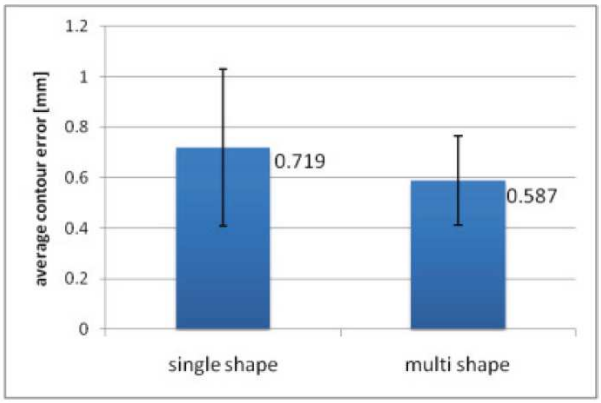


Figure 8.17: Performance index of single-shape graph-cuts and multi-shape graph-cuts: average and standard deviation of testing 48 clinical cases.

Figure 8.18: Shape priors used for the case in Figure 8.16 (d)-(f).

7 Conclusion

Throughout the paper, we introduced two state-of-the-art methods for extracting features: One for local features and the other for shape features. These features are used together with a linear support vector machine to build a region graph to solve for the sub-graph having the best region score which often corresponds to the object in an image. We also showed how a contour graph is built based on the output of the gPb contour detector. Then, the contour cut algorithm was applied on the contour graph to successfully detect salient contours in a image. Finally, we showed a method to construct a Markov random field from an image and used the multi-shape graph-cut algorithm to segment the lungs from a CT image with high accuracy.

We have seen that both the efficient region search system and the contour cut algorithm heavily depend on the output of an edge detector therefore it is obvious that the performance of the edge detector can affect the accuracies of these systems. The multi-shape graph-cut algorithm also depends on the shape priors. Hence, given the wrong assumptions about the shape of the object we will end up at the wrong resulting shape similar to the shape priors. This leads to the problem of tuning the parameters such that the shape priors can partly influence the result. Depending on the goal each system is addressing an appropriate graph construction algorithm is needed. The contour cut algorithm tries to improve the result of an edge detector and it can spot out the most salient contours from the image, as the consequence edge segments are suitable candidates for graph construction. The purpose of the efficient region search system is to detect and segment an object, hence object fragments detected by an edge detector together with the water shed algorithm are appropriate to build the graph. On the other hand, the multi-shape graph-cut algorithm operates on the pixel/voxel, level which are the smallest image elements. In fact, both the contour cut and the efficient region search system can be formulated on the pixel level but this will rapidly slow down the efficiencies. Furthermore, since the edge detector performs well enough in the real-world images we can safely rely on it. Finally, for each type of graph constructed by a different system, we need to use a different graph-cut algorithm to find the optimal solution thanks to the advancement in graph theory.

Literatur

[1] Mortensen, Eric N., and William A. Barrett: Intelligent scissors for image composition. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. ACM, 1995.

[2] Chan, Tony F and Vese, Luminita A: Active contours without edges. Image Processing, IEEE Transactions on 10.2 (2001): 266–277.

[3] Fripp, Jurgen and Crozier, Stuart and Warfield, Simon and Ourselin, Sébastien: Automatic segmentation of articular cartilage in magnetic resonance images of the knee. Medical Image Computing and Computer-Assisted Intervention–MICCAI 2007 (2007): 186–194.

[4] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool: Surf: Speeded up robust features. Computer Vision–ECCV 2006 (2006): 404–417.

[5] Gu, C. and Lim, J.J. and Arbeláez, P. and Malik, J.: Recognition using regions. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 1030–1037. IEEE, 2009.

[6] Malik, Jitendra, Serge Belongie, Thomas Leung, and Jianbo Shi: Contour and texture analysis for image segmentation. International Journal of Computer Vision 43, no. 1 (2001): 7–27.

[7] Martin, David R., Charless C. Fowlkes, and Jitendra Malik: Learning to detect natural image boundaries using local brightness, color, and texture cues. Pattern Analysis and Machine Intelligence, IEEE Transactions on 26, no. 5 (2004): 530–549.

[8] Maire, M. and Arbeláez, P. and Fowlkes, C. and Malik, J.: Using contours to detect and localize junctions in natural images. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1–8. IEEE, 2008.

[9] Vijayanarasimhan, Sudheendra, and Kristen Grauman: Efficient region search for object detection. Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011.

[10] Kennedy, Ryan, Jean Gallier, and Jianbo Shi: Contour cut: identifying salient contours in images by solving a Hermitian eigenvalue problem. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 2065–2072. IEEE, 2011.

[11] Nakagomi, Keita, Akinobu Shimizu, Hidefumi Kobatake, Masahiro Yakami, Koji Fujimoto, and Kaori Togashi: Multi-shape graph-cuts and its application to lung segmentation from a chest CT volume.

[12] Arbeláez, P. and Maire, M. and Fowlkes, C. and Malik, J.: From contours to regions: An empirical evaluation. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pp. 2294–2301. IEEE, 2009.

[13] Cortes, Corinna, and Vladimir Vapnik: Support–vector networks. Machine learning 20, no. 3 (1995): 273–297.

[14] Burges, Christopher JC: A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery 2, no. 2 (1998): 121–167.

[15] Lampert, Christoph H., Matthew B. Blaschko, and Thomas Hofmann: Beyond sliding windows: Object localization by efficient subwindow search. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pp. 1–8. IEEE, 2008.

[16] Leventon, Michael E., W. Eric L. Grimson, and Olivier Faugeras: Statistical shape influence in geodesic active contours. In Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on, vol. 1, pp. 316–323. IEEE, 2000.

[17] Tsochantaridis, Ioannis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6, no. 2 (2006): 1453.

[18] Ljubić, I. and Weiskircher, R. and Pferschy, U. and Klau, G.W. and Mutzel, P. and Fischetti, M.: An algorithmic framework for the exact solution of the prize–collecting Steiner tree problem. Mathematical Programming 105, no. 2 (2006): 427–449.

[19] Zhu, Qihui, Gang Song, and Jianbo Shi: Untangling cycles for contour grouping. In Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pp. 1–8. IEEE, 2007.

[20] Bishop, Christopher M.: Pattern recognition and machine learning. Vol. 4. No. 4. New York: springer, 2006.

[21] Shimizu, Akinobu, Keita Nakagomi, Takuya Narihira, Hidefumi Kobatake, Shigeru Nawano, Kenji Shinozaki, Koich Ishizu, and Kaori Togashi: Automated segmentation of 3D CT images based on statistical atlas and graph cuts. Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging (2011): 214–223.

[22] Mitchell, John E: Branch-and-cut algorithms for combinatorial optimization problems. Handbook of Applied Optimization (2002): 65–77.

[23] Gleich, David: Hierarchical directed spectral graph partitioning. (2006).

[24] Boykov, Yuri, and Vladimir Kolmogorov: An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. Pattern Analysis and Machine Intelligence, IEEE Transactions on 26.9 (2004): 1124–1137.

[25] Kolmogorov, Vladimir, and Carsten Rother: Minimizing nonsubmodular functions with graph cuts-a review. Pattern Analysis and Machine Intelligence, IEEE Transactions on 29.7 (2007): 1274–1279.

[26] Lempitsky, Victor, Carsten Rother, Stefan Roth, and Andrew Blake: Fusion moves for markov random field optimization. Pattern Analysis and Machine Intelligence, IEEE Transactions on 32, no. 8 (2010): 1392–1405.

[27] Nowozin, Sebastian, and Christoph H. Lampert: Global connectivity potentials for random field models. Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[28] Shi, Jianbo, and Jitendra Malik: Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on 22, no. 8 (2000): 888–905.

[29] Leung, Thomas, and Jitendra Malik: Contour continuity in region based image segmentation. Computer Vision–ECCV'98 (1998): 544–559.

Perceptual Based Depth-Ordering

Tobias Funke

Contents

Abstract

This work introduces two different approaches for rendering semitransparent volumes. The first presented approach tries to enhance humans depth perception by using psychological findings in the field of human visual perception. This approach thereby tries to avoid introducing new lighting features (e.g. halos) to minimize redundant image information and to avoid occlusion. In contrast to that, the second approach tries to increase the perception of the depth ordering by introducing feature halos and shadows to the rendered volumes. This work compares both of these approaches taking into account advantages and disadvantages of perception as well as computation times and real-time ability.

Keywords: volume rendering, depth perception, transparency, halo, illustrative visualization

1 Motivation

The field of volume rendering is a part of nowadays computer science . The main idea of volume rendering is to generate 2D images from 3D volume information. The volume data on which the projected 2D image is based on may be derived from several sources and fields of use. Beside artificial data that was created on a computer e.g. component planing in preproduction processes, techniques of volume rendering are often used for analytic purpose. Movement of fluids or gases is one example for analytic usage. One very important field of use is medicine. Todays diagnostic processes create a huge amount of visual data like CT, MRT or Sonar scan images. Much of the output that is produced by CT- and MRT-Scans is stored in sets of 2D-Images. This type of data may be sufficient for some diagnostic purposes but it is often hard to perceive coherences within the data that is based on three dimensional structures. In order to ease and increase the use of these sets of 2D-Images a three dimensional volume is processed. Since the 2D data is normally filled with opaque color- or grey-values it is often necessary to derive a semitransparent view for the volume in order to not only show the surface of an object. To generate different volumes from the two dimensional dataset, the data often contains density- ,intensity- or other comparable values, depending on the data source. With these values it is possible to detect coherent structures that have similar density values. Later in the process a color and a transparency level can be assigned to each object. With the transparency the user is on the one hand able to look *trough* different image-layers in a single view but on the other hand a problem is introduced regarding the depth perception of a given object. Since especially in medical purpose many volumes may overlap or even intersect (e.g. bones, muscles, veins) it is important for the user to perceive each object in a correct depth order. To achieve this ordering there are different approaches to compute a 3D representation and the projected 2D image from given data. Although it is not possible to implement the depth perception unambiguous, todays approaches try to ease the perception. This paper will introduce two different approaches of volume rendering that intend to increase the correctness of the perceived depth order of several volumes. The two approaches are later compared regarding their usability for different cases like static images, videos or real time usability.

1.1 Foundations of Volume Rendering

In general volume rendering describes a transfer function $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ that maps information of a three dimensional space to an image plane consiting of pixels with scalar values e.g colors that can be displayed on a screen. To render the three dimensional volume there are different approaches. They can be divided into direct and the indirect volume-rendering. One indirect rendering approach is to extract the iso-surfaces of a given volume. An iso-surface is a plane that consists of voxels with similar density values. This representation for example can be calculated with the marching cubes algorithm [1]. The surfaces are then rendered onto the screen. Since the indirect rendering needs to calculate this alternative representation of the volume data at first, it is more time consuming then direct volume rendering. Therefore in medical applications often the direct rendering is used. This also allows the visualization of semi transparent objects and their interior. In the following section only the direct rendering process is described. In contrast to typical rendering of polygonal objects, direct volume rendering does not only take information of the surface of an object into account. While polygonal rendering is normally limited to the rendering of polygonal surfaces that face the users point of view volume rendering is used to display three dimensional objects with their interior. Therefore not only the rendering itself works in a different way but also the data structure that is rendered looks different. Since not only the surfaces are rendered it is necessary to have information about every voxel inside a volume. In many cases the data set even provides the same

information for every voxel in the whole scene. These information are often density- or intensity values that come up from diagnostic measurements. The results of these measurements are often stored in sets of two dimensional slices. These slices contain for every pixel a measured value e.g. the measured magnetic resonance of a body part represented as scalar value. In a rendering process these values somehow have to be mapped to color- or grey-values to be displayed on the screen. The first step of this process is to find a mapping for each voxel. The mapping should map the density- or intensity-value of a voxel to a color- and an opacity-value. This can be done by an easy filter function. The properties and the configuration of the filter can thereby strongly vary. Dependent on the purpose the filter can be used to extract for example bones, vessels or skin. In this way the voxels in the scene that do not belong to a structure that should be visualized (e.g. air) can be filtered out. Typically the filter functions can be described as Gaussian- or triangle-distributions. Since the image data comes from physical proceedings and due to the natural deviation of the body parts that are examined the density-/intensity-values of structures like bones are not fix values but can be defined as a range of values with different probabilities. This guarantees a robust mapping and in the further progress useful results. Figure 9.1 shows an Gaussian-filter that maps density or intensity values to opacity values. The parameters of the filter functions e.g. the peak of each distribution as well as the width can often be manipulated on the fly in the user interface. In this way the filters can be adapted to specific needs.

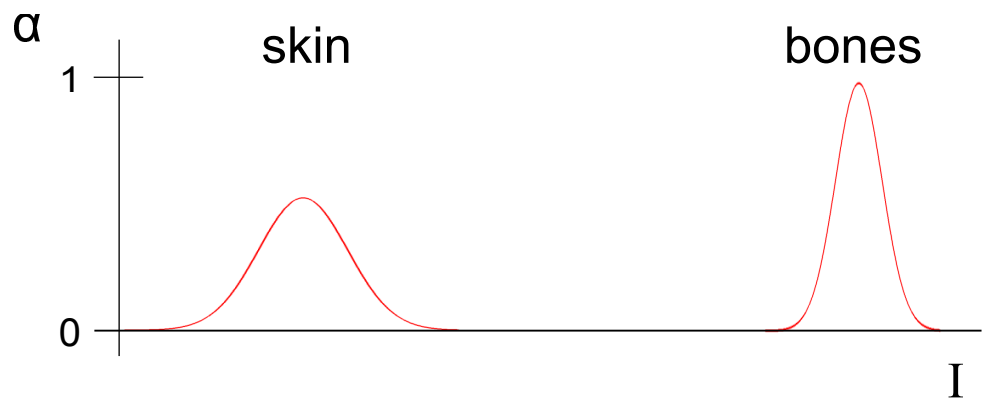


Figure 9.1: A simple density filter that assigns opacity values alpha to a range of density values I with a Gaussian distribution. The higher the alpha-value the more opaque a voxel gets. For color values similar filters can be assigned.

After assigning color- and grey-values as well as opacity values to the voxels of a dataset it is possible to compute a two dimensional image from the dataset. Therefore in many approaches the technique of *ray tracing* is used. Thereby a ray is from the virtual camera position through a pixel of the screen is cast into the dataset. On its path through the volume the ray integrates color- and opacity values, generating a single value for the color and a value for the opacity to be displayed on the respective screen pixel. To calculate the correct color that should be displayed on the screen is is necessary to understand the rendering integral. Each voxel of the volume is assumed to emit a fix amount of light with a defined color this light accumulates on the path of a ray since the voxels that are intersected emits light. Therefore the rendering integral runs over all the voxels along a ray. Furhtermore the voxels influence the perceived color of the voxels lying behind by damping the emitted light of other pixels, therefore also a damping factor has to be defined to calculate the screen pixels color. The damping can be described as

$$damp = e^{-\int_0^s o(R(x))dx}, \text{ where } o(R(x)) \text{ is the opacity at the position } x \text{ on the Ray } R \tag{9.1}$$

The continuous representation of the whole calculation for one ray/pixel is then

$$C = \int_0^\infty c(R(s)) \cdot damp, \text{ where } c(R(x)) \text{ is the color at the position } s \text{ on the Ray } R \tag{9.2}$$

Since this approach is not suitable for discrete data like a voxel set, the equation has to be transfered to a discrete representation.

$$C = \sum_{i=0}^{k} (c_j \cdot \prod_{j=0}^{i-1} e^{-o_j}) \tag{9.3}$$

For algorithmic use this equation is further simplified but the general concept does not change. [3]

2 Perceptually Based Depth-Ordering

The volume rendering method that is introduced in this paragraph is based on the idea that psychological aspects of human perception can be used to create unambiguous visual depth information in 2D images [2]. The processing of the 3D image data is thereby focused on semitransparent structures. As explained in the introduction, the overlapping and intersecting of these structures and the presentation of their correct depth order is one of the most challenging problems in the field of volume rendering. In contrast to many other approaches this approach tries to avoid introducing new visual features (e.g. specular highlights, halos or shadows)to the image data. Since artificially introduced features can't generate additional information that was not in the data before, added features are always redundant. Furthermore, each additional feature has a chance to occlude important information that was visible in the original image data. This is true for halos at the edge of a feature as well as for shadows that are casted from one structure onto the surface of another feature. To increase the depth perception without new features it is necessary to manipulate and improve existing features. Therefore the improvement has to be done by only adjusting color- and transparency-levels of the structures that should be displayed. The presented approach is mainly based on two different depth-perception models that are also introduced in this chapter. The first model is the Adelson-Anandan-Anderson's *X-Junction* model [4] and the *Transmittance Anchoring Principle* (TAP) [5]. Both models describe the human perception of depth ordering of semitransparent objects. Both models are used to predict how the depth ordering of two overlapping regions in an image is most probably perceived by a user. With this information the image is then optimized with regard to the psychological findings to use the users psychology to improve the perceptive results. Beside the psychological models, a third component is used to evaluate and iterate the models' output and to find the most supporting visualization. This third component, that is described in this chapter, contains an energy function that measures the effectiveness of the perceived depth and weights the output of both models as well as the original image.

2.1 Edge detection

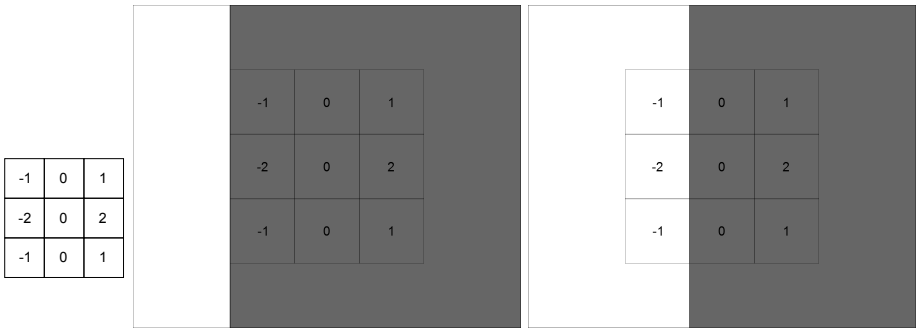
To be able to apply the concept of the X-Junctions and the TAP (see section 2.2, 2.3) it is necessary to find suitable regions within the image. This preprocessing step is done to decrease the computational effort that would emerge if both concepts would be executed for each pixel of the image. The X-Junction model that should be applied in the first step works on intersecting edges of overlapping image features. To find these edges a common practice in computer graphics the so called *edge detection* is used on the image. In a two dimensional image an edge can be described as a rapid change of color- or grey-values between neighboring pixels. The edge detection algorithm that was used in this approach is the *canny edge detector* [6]. To guaranty a robust edge detection the algorithm at first filters the image to smooth the grey values. This step is done to minimize errors that can occur due to interferences in the image data. The filter is a square matrix that calculates a new color- or grey-value for each pixel taking neighboring pixels into account. Dependent on the size of the matrix the filter works more robust or more precise. A big matrix decreases the interference errors but may also lead to missed detections of small edges. A small matrix detects more edges but is less robust. Typically the matrices are Gaussian filters. After the first filtering step for each pixel a separate convolution in x- and y-direction is calculated. Figure 9.2 shows a 3x3 convolution matrix. This matrix is applied to a pixel of the image and its neighbors. Figure 9.2 (b) shows the matrix applied to a smooth surface. Assuming all white pixel to have a color value zero and all grey pixels to have a color value one the result of the convolution can be calculated as sum of the pixels inside the matrix with the respective factors.

$$(-1) \cdot 1 + (-2) \cdot 1 + (-1) \cdot 1 + 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 + 2 \cdot 1 + 1 \cdot 1 = 0 \tag{9.4}$$

In the same way a value for the center pixel of Figure 9.2 c) is calculated.

$$(-1) \cdot 0 + (-2) \cdot 0 + (-1) \cdot 0 + 0 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 + 2 \cdot 1 + 1 \cdot 1 = 4 \tag{9.5}$$

In this way a high value indicates that a pixel lies on an edge while a low value indicates that a pixel is most likely within a plane. Since the example shows a perfect edge with a high contrast it can be imagined that the result does not always match the case of the maximum- or minimum-value. For results in between a boundary value can be defined to decide if a pixel is on an edge or not. The example that was shown in Fig. 9.2 describes the convolution of pixels in Y-direction. Accordingly the convolution also has to be done in X-direction to not only detect vertical edges. The two results are then combined in one edge image where only the edges are left. The example shows that in this way edges can be detected with a small computational effort (only additions and multiplications of integer values).



(a) 3x3 convolution matrix (b) Convolution matrix applied to a pixel within a smooth surface (c) Convolution matrix applied to a pixel on an edge

Figure 9.2: A convolution matrix is applied to different pixel regions

2.2 X-Junctions

The X-Junction model that is used in this volume rendering approach, is a psychological model of human depth perception. The implementation mimics the behavior of the psychological model and tries to calculate the perceived depth order of two overlapping semitransparent features of a 2D-image. Since the X-Junction model only calculates the depth perception for a certain point of the 2D-image and its neighboring pixels it is possible to have many overlapping features processed in a single image. As it can be derived from its name, the X-Junction models is applied to two dimensional junctions in a certain image. Therefore it is necessary to find the junctions in the image. The junctions are basically intersecting outlines of image features. To find intersecting outlines of different features in the image is is useful to first process a filtering of the image. As it is described in chapter 2.1 a robust edge detection algorithm is applied to the image. In this way the calculation of the actual junction points gets much faster. Since after the filtering only pixels on the filtered edges have to be examined further, there is a huge amount of saved computation time in comparison to an algorithm without pre filtering. In this case each pixel of the image would have been examined if it is a junction point or not. To find the actual junction points the pixels on the edges are checked for abnormalities. Therefore at each edge-pixel the surrounding pixels are subdivided into four areas. For each area a color value is calculated. The resulting color values are then compared to each other. If two color values correspond to each other the pixel is most likely no junction. If all four regions diverge in their color values the pixel on the edge is with a certain probability a junction point. With the differences between the color values it is then possible to find local extrema along an edge. These extrema are then selected as junction points.

If a junction is found the area around the junction is again divided into four segments. Afterwards for each segment the algorithm has to determine the segments luminance. With the luminance values of the four segments it is then possible to calculate an order for these elements. The description of the different cases of luminance values is based on Figure 9.3. Figure 9.3 shows a junction point where the edges of the image features I and II intersect. The four regions that are chosen for the differentiation are a, b, c and d. Once the luminance-order of the segments is calculate the X-Junction model distinguishes different constellations of the luminance-order to determine the perceived depth order. The order of the luminance values of the four segments is in Fig. 9.4 and Fig. 9.5 denoted with the black arrows. These arrows also give name to the different configurations.

The A-Configuration that is shown in Figure 9.4 (a) shows the ordering of the luminance values of the four areas around the junction point. Although the areas c and d (according to Figure 9.3) have

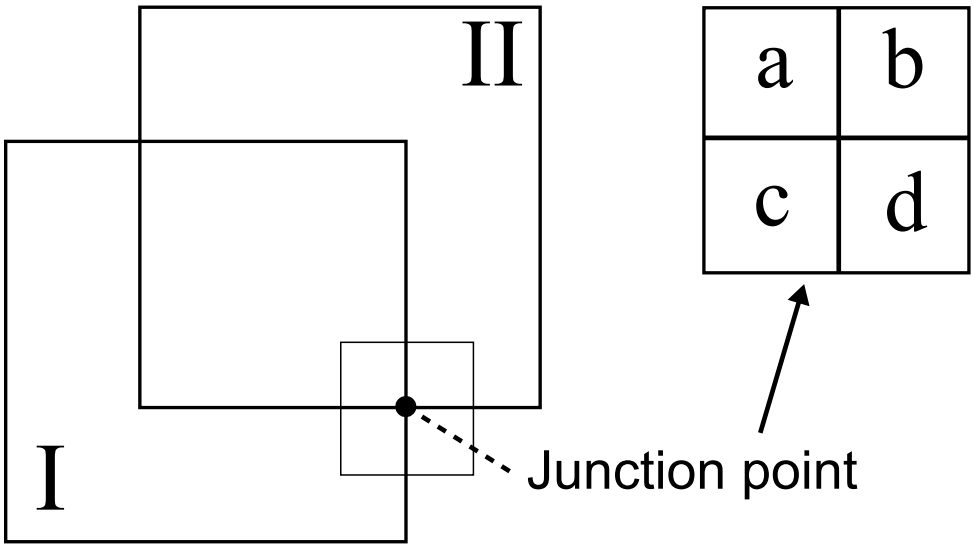
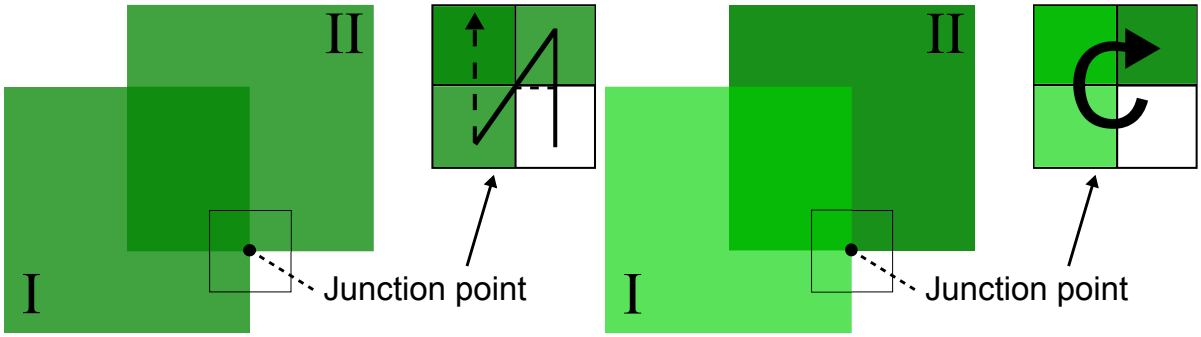


Figure 9.3: The edges of two overlapping image features (I ,II) form a X-Junction point. The are around the junction is divided into four areas (a, b, c, d) to allow further calculation.



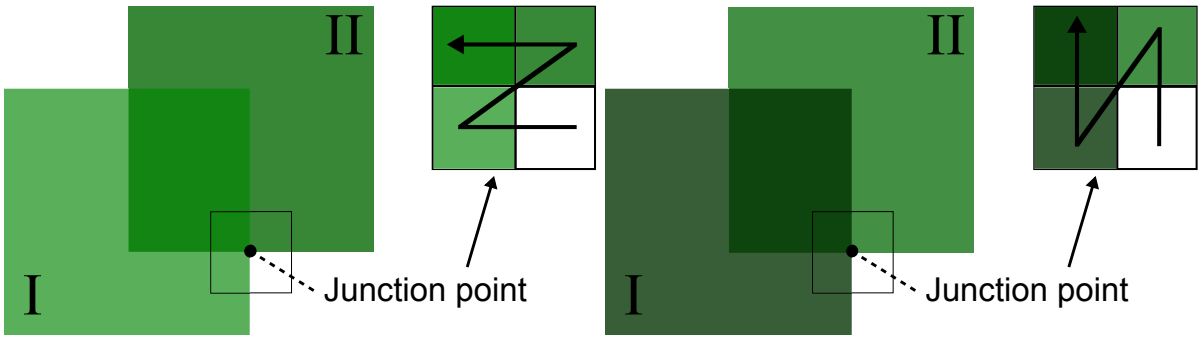
(a) (b)

Figure 9.4: Comparison of A- and C-Configurations at an X-Junction point (based on [2])

the same luminance values and therefore the ascending ordering of the luminance values could also be $d < c < b < a$ this configuration is defined as A-Configuration. To differentiate between a Z- and an A-Configuration it is defined that a configuration is an A-Configuration if two areas have the same value else it is called a Z-Configuration. The A-Configuration describes an ambiguous situation. The viewer can not determine which one of the features I and II is in the foreground and which is in the background. Both orderings could be predicted. In contrast to that the C-Configuration shows an unambiguous case of the ordering. The image feature I is overlapping image feature II. The ordering of the luminance values of the four areas can be denoted as $d < c < a < b$.

The Z-Configuration (Fig. 9.5) is different to the other configurations. Z-Configurations can support the visual perception of the depth order on the one hand but also can mislead to a wrong perception. Similar to the C-Configuration shown in Fig. 9.4 (b) the correct Z-Configuration supports the correct depth perception although the correct Z-Configuration (Fig. 9.5 (a)) is more ambiguous then the C-Configuration. The incorrect Z-Configuration shown in Figure 9.5 (b) even leads to a wrong perception of the depth ordering. The arrows that form a 90 degrees turned Z lead to the perception that feature II is overlapping feature I although the ordering is the opposite way.

As it is shown in Figure 9.4 and 9.5 the different configurations may lead to different perceived depth orderings. Although the X-Junction model is based on an psychological effect and therefore is not able to exactly determine how the depth ordering of to objects is perceived, it is necessary to compute the most likely perceived order of the objects. Only if the algorithm can make a decision if an object is perceived in



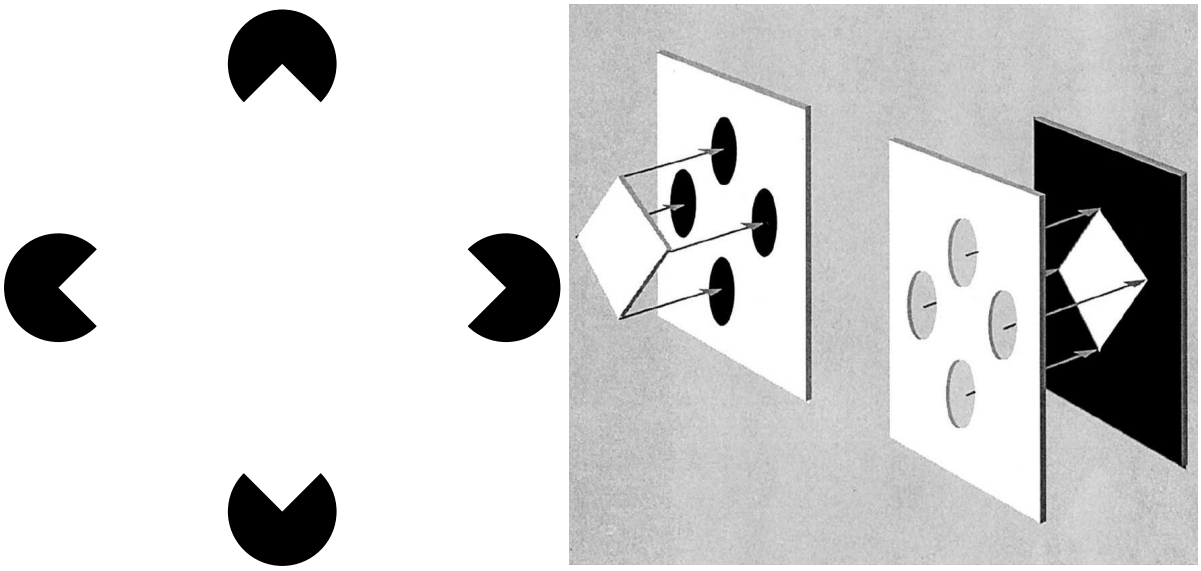
(a) (b)

Figure 9.5: Correct and incorrect Z-Configuration a an X-Junction point (based on [2])

the front or in the back it is possible to amplify the effect or even to change the perceived order regarding the real depth order.

2.3 Transmittance Anchoring Principle

The second model that is used in this rendering approach is the Transmittance Anchoring Principle. This principle is also based on psychological findings. The idea of the TAP is that the perceived depth of two objects is dependent on the contrast of these objects. The object that has the highest contrast is perceived to have the highest distance to the users point of view. With this idea being implemented in the rendering approach it is possible to detect mismatches between perceived and actual depth ordering. It also may support hypotheses of the X-Junction model or even calculate a depth order where the X-Junction model was not able to find an order.



(a) (b)

Figure 9.6: Transmittance Anchoring Principle example (taken from [7])

Figure 9.6 shows an easy example of TAP. In Figure 9.6 (a) the structure will most likely be perceived as a white square covering four black circles. Figure 9.6 (b) shows that there are different ways to create a two dimensional image shown in (a) with overlapping structures in tree dimensions. The TAP is a very easy model for depth perception and therefore also fast to calculate. Therefore the idea of the TAP is used in the algorithm to support or even correct the calculated depth order that comes up from the calculations regarding the X-Junction model. Remembering the different configurations of the X-Junction model (see Section 2.2) the TAP model can help in two cases. Although the Z-Configurations are divided

into a correct and an incorrect configuration an algorithm is not able to decide which of the two cases is accurate in a given situation without additional depth information. The TAP can therefore be used to calculate the perceived depth order of the overlapping features and in this way help to determine an improvement for the luminance values of the image features. For the C-Configuration as well as for the A-Configuration (see Fig. 9.4) the TAP is not suitable. For a given C-Configuration no improvement has to be done and therefore the TAP would just increase the computational effort that is needed. For an A-Configuration where the luminance values of both image features are the same the TAP is not able to calculate the perceived depth ordering.

2.4 Combination of X-Junctions and TAP

As it was presented in Section 2.2, the X-Junction model is designed for depth ordering decisions of two overlapping objects at a time. Since the model is just able to apply rules for four different areas around a junction it is not possible to detect perceived depth ordering for more than two objects. Obviously the X-Junction model can not always be used to change perceived depth ordering in complex scenes. As soon as the number of overlapping objects increases, the color and luminance adaption for each object that are based on the X-Junction calculations may be problematic. The perceptive improvement of one X-Junction and the two related objects may lead to worsening of other X-Junctions that are also related to the same two objects. Therefore it might be impossible to find a perfect configuration of all the objects. At this point the TAP model that was presented can be used to decrease ambiguity. To enhance the performance of the system, the TAP is in this approach only used as an additional perception test if the X-Junction model fails to calculate an unambiguous configuration. In cases where the X-Junction model is able to calculate an unambiguous depth perception, the TAP is not used for further calculations.

2.5 Algorithm

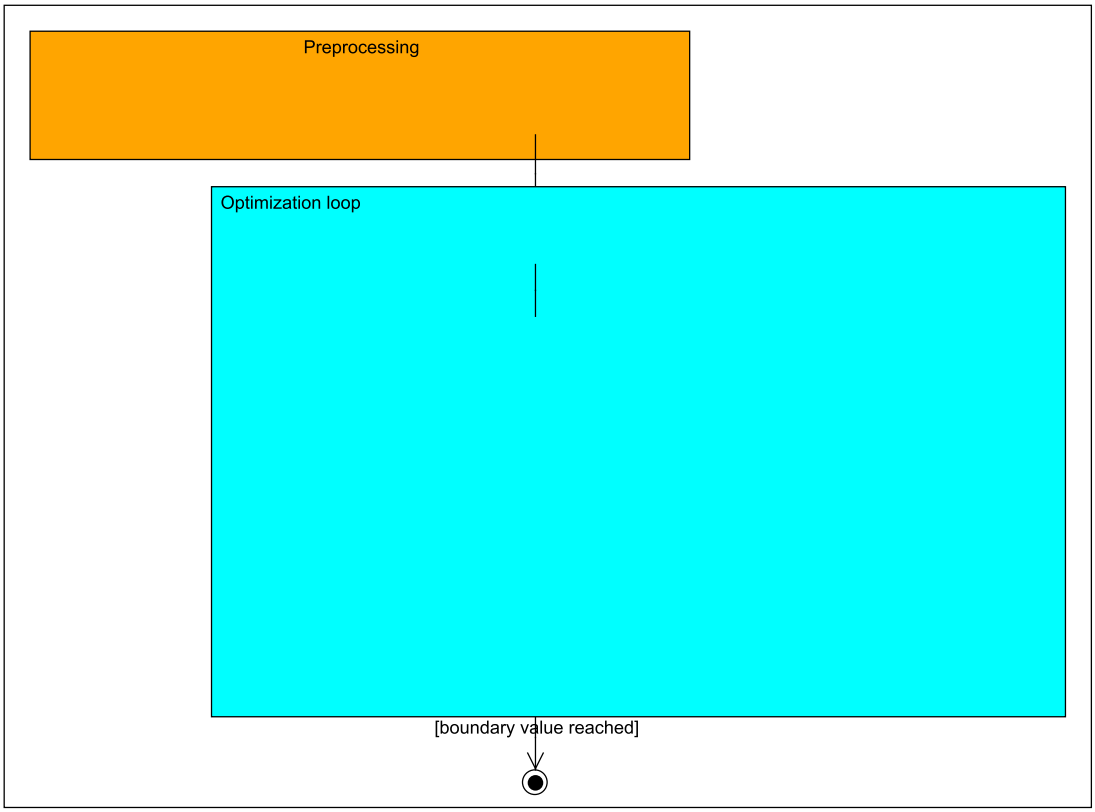


Figure 9.7: Complete Algorithm of the perceptual based rendering approach. The preprocessing step is only executed once while the optimization is an iterative process.

Since both psychological models that were presented are only concepts for analyzing a given image, we need to transfer these information to an optimization algorithm to actually increase the images' quality. Since the X-Junction model is primarily used to calculate the perceived depth and the algorithm uses an iterative process to increase the image-quality, the junctions that are needed for this model are computed once in a preprocessing step. Since the junction points do not change their position during the optimization it is not necessary to compute them in every iteration step. In this way the overhead of the function is minimized to increase computation speed. Nevertheless the X-Junction model has to recompute the perceived depth for every junction point in every iteration step. For complex scenes where many overlapping objects occur, this iteration slows down the algorithm. This has to be taken into account if this approach is compared to other volume rendering approaches.

2.6 Energy Function

The main goal of the algorithm that was presented in [2] was to increase the quality of a given input image. Therefore the first step of an optimization is to define the term *quality* in the context of depth-perception and to calculate this quality for a given image. In order to calculate the quality of an image, the approach that is presented introduces an energy function with three different parts that are described in the following sections.

2.6.1 X-Junction Energy Function The first part of the energy function takes the X-Junction model as well as the TAP model into account and rates the different configurations. As it was described in chapter 2.2 there are several possible configurations for X-Junctions. The configurations that describe the luminance relations between the four neighboring regions around an X-Junction thereby have to be separated according to their effect on the quality of the image. The quality of the image in this context only concerns the quality of the depth perception. In this context an image with a high quality is defined as an image where the perceived depth information has a high accordance to the real depth values. Misleading or ambiguous images or image parts have a low quality. To measure the quality of an image it is important to measure the congruence of the displayed and the actual depth relations. With the classification that was done in chapter 2.2 the configuration may correspond to one of the three cases that were described in 2.2. Since it is necessary to distinguish if a configuration is correct (regarding the actual depth relation) the three cases are further split. In this way each configuration matches one of the new cases:

- *A-Configuration*: As it was described in chapter 2.2 the A-Configuration leads to an ambiguous depth perception. Therefore it is not decidable which of the two overlapping objects I and II is in the front and which is in the back. For the energy function that measures the quality of the depth perception of a given image this case does not provide any benefit but neither does it provide wrong information.
- *Correct Z-Configuration*: A correct Z-Configuration as depicted in Fig. 9.5 (a) is the second ambiguous case of the different configurations. Although a Z-Configuration provides more information about the depth ordering of two objects it still is not unambiguous. A correct Z-Configuration still provides benefit for a given image and therefore also for the energy function.
- *Incorrect Z-Configuration*: The last ambiguous case that can occur in the given setup is the incorrect Z-Configuration. Similar to the correct Z-Configuration the incorrect configuration does provide depth ordering information at a given X-Junction point. As the name describes it for the incorrect configuration theses provided information do not correlate to the actual depth ordering. Instead of increasing the image quality this configuration leads to wrong perception of the depth ordering and therefore should be penalized in the optimization process.
- *Correct C-Configuration*: In contrast to all the other configurations the C-Configuration provides unambiguous information about the depth ordering of two intersecting image features (see Fig. 9.4 (b)). The C-Configuration therefore is the configuration with the highest benefit for the image quality.
- *Incorrect C-Configuration*: Although this configuration is part of the X-Junction model it can not occur in the context of this algorithm. Therefore this configuration is not considered in the energy function.

For the quality-function the different cases are ordered by their benefit for the images quality. Since it is in the later process easier to minimize a given function (towards zero) then to find a maximum, the configuration with the highest benefit for the image is assigned to the lowest value. Obviously the incorrect and therefore misleading configurations should be penalized. Thus the incorrect Z-configuration has the highest value in the energy function. The incorrect C-configuration that would also lead to a wrong depth perception can not occur in semitransparent volume rendering. The configuration with the next highest value is the A-configuration. The A-configuration is always ambiguous and therefore gets a higher value than the correct Z-configuration. This configuration is not unambiguous but it still supports a depth ordering and therefore increases the images quality. If the contrast of the regions is further improved it is possible to generate a C-configuration from a Z-configuration. The C-configuration is the best case. It is according to the X-Junction theorem an unambiguous way of displaying a correct depth order. Although this configuration is rated best it is still necessary to check if the color differences are high enough to be perceived by human users. With this information a function is set up to calculate numeric values. Since the differences in luminance values of the four areas are used to calculate the value of the energy function these values are defined as *l(a)*, *l(b)*, *l(c)* and *l(d)* for the four areas a, b, c and d (as depicted in Fig. 9.3). The energy function itself is divided into two parts to allow an easy decision between the different cases that were described above. The first part of the function ranges from 0.5 to 1 and is used to measure the values of the Z-Configurations and the A-Configuration. Since the A-Configuration can be seen as transition between the incorrect and the correct Z-Configuration its value is fixed to 0.75. Figure 9.5 shows that for the decision if a Z-Configuration is correct only *l(b)* and *l(c)* are needed. This also explains the transition position of the A-Configuration since this configuration only occurs if *l(b)=l(c)*. Therefore the first part of the energy function is defined as

$$E_{d_1} = \frac{3}{4} + \frac{1}{4} tanh(-d_1) \tag{9.6}$$

with

$$d_1 = l(b) - l(c) \tag{9.7}$$

For the A-Configuration case the equation is equal to 0.75 since d_1 is equals to zero. The second part of the energy function only refers to the correct C-Configuration. To measure the quality of this configuration it is necessary to consider all four luminance values. To calculate the distance only the neighboring regions (within the luminance order) are compared. With this regards the second part of the energy function is described as

$$E_{d_2} = \frac{1}{2} \cdot tanh(\frac{1}{d_2} - 1) \tag{9.8}$$

with

$$d_2 = \sqrt{\frac{||l(d) - l(c)||^2 + ||l(c) - l(b)||^2 + ||l(b) - l(a)||^2}{3}} \tag{9.9}$$

Since only one case can occur at a time the resulting energy term for one X-Junction is

$$e_d(i) = \begin{cases} e_{d_1}, & \text{if } l(a) >= l(b), \\ e_{d_2}, & \text{otherwise} \end{cases} \tag{9.10}$$

In this way the quality of each junction can be rated to determine a resulting quality value for the whole image

$$E_d = \frac{1}{n_j} \cdot \sum_{i=1}^{n_j} e_d(i), \text{ with } n_j \text{ being the number of all X-Junctions in the image} \tag{9.11}$$

2.6.2 Transparency Energy Function To avoid an *over-optimization* of a given image where image features become nearly invisible during the optimization a second part of the energy function is used to measure transparency differences before and after the optimization. The perceived transparency of overlapping features is thereby calculated as

$$\alpha = \frac{p - q}{a - b}, \text{ with } p = \alpha a + (1 - \alpha)t \text{ and } q = \alpha b + (1 - \alpha)t \tag{9.12}$$

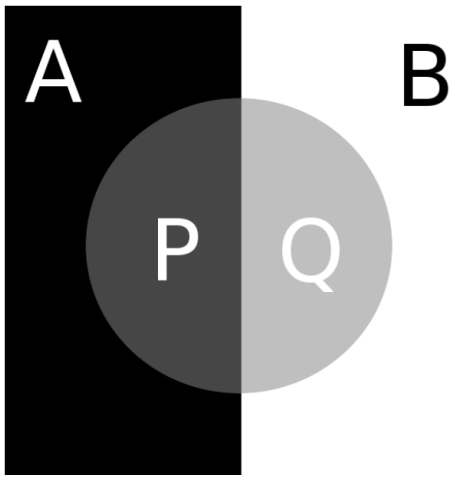


Figure 9.8: Two neighboring regions A and B with a circular overlaying region (taken from [2]).

In this equation a,b,p and q represent the luminance values of the different areas depicted in Fig. 9.8. Furthermore t is the luminance value of the layer T. To calculate how the perceived transparency has changed during the optimization process α is calculated for the input image as well as for the optimized image. The energy term that takes the difference into account is therefore defined as:

$$E_t = \frac{1}{n} \cdot \sum_{i=0}^{n-1} ||\alpha_i^{'} - \alpha_i|| \tag{9.13}$$

Used in an optimization process, this term helps to keep the transparency of a feature close to its original transparency value. Since high differences result in high values of this energy-term they are therefore punished in the minimization.

2.6.3 Image Faithfulness Energy Function The third part of the energy function regards the faithfulness of the optimization. This means that this term is used to benefit optimization results that are close to the original input image and punishes outputs that strongly differ. Although the second part of the energy function has a similar functionality the third part takes the whole image into account instead of only judging transparency values around feature points. To compare both, the input image and the optimized image, both images are converted into greyscale images. To measure the overall entropy the probability distribution *p(x)* regarding the grey value $x \in [0, 255]$ of the histogram of the input image X is needed. Furthermore the joint distribution *p(x,y)* is needed. It describes the two dimensional joint histogram for the two images X and Y after converting them to greyscale images. With these distributions the energy term is defined as:

$$E_e = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \frac{p(x)}{p(x,y)} \tag{9.14}$$

With the probability distributions $p(x)$ and $p(x,y)$ this energy term increases for differences between the histograms of both images. If both images are equal and therefore their histograms are also equal the term $log \frac{p(x)}{p(x,y)}$ becomes zero as well as the whole sum.

2.6.4 Combination With the help of the three energy terms E_d, E_t and E_e the combined energy function E can be defined as:

$$E = \sum w_d \cdot E_d + w_t \cdot E_t + w_e \cdot E_e \tag{9.15}$$

The weights w_d, w_t and w_e are defined by the user and determine how each term should be weighted in the energy function. This weighting therefore also influences the optimization process. As it was described above the first energy term increases the distances between the luminance values of regions around a junction point. It is necessary to also use the second and third part of the energy function to maintain the

features of a given image and to furthermore preserve visibility of the features of the input image. After the optimization with the first part of the energy function some areas of the image may become nearly invisible. Since the regions are optimized to increase the contrast between these regions the visibility may suffer from these optimizations if the luminance value of one region gets to high or to low. The second part of the energy function is able to solve this problem but still does not preserve the faithfulness of the input image (initial image and optimized image differ very strong). The third part is used to preserve the optimization on the one hand but to keep the resulting image close to the original image on the other hand. The authors of [2] showed the influence of the different parts of the energy function in Fig. 9.9. For their results they chose the weights as: $w_d = 3$, $w_t = 1$ and $w_e = 1$.

Figure 9.9: The first image shows the initial image with three overlapping layers A, B and C. The second image shows the optimization result only using the first part of the energy function. In the third image the second part of the energy function is included in the optimization process. The fourth image shows the optimized image using the complete energy function (taken from [2]).

2.7 Transfer-Function

With the energy function it is possible to calculate the quality of an image and its optimized luminance values regarding the depth perception. All terms of the energy function are designed in a way that the best quality results in the smallest values (greater zero). Therefore the optimization problem in this approach can be transfered to a minimization problem. Due to the huge amount of parameters that influence the energy function an optimal solution is very difficult to compute. These parameters consist of the parameters for each part of the energy function: $x_d = (l(a), l(b), l(c), l(d)), x_t = (a, b, p, q)$ and $x_e = (p(x), p(x, y))$. Therefore the minimization process is only implemented as an approximation. To find a good solution a nonlinear conjugate gradient method is used with a transfer-function defined as:

$$T(x) = \sum_i \alpha_i G_{\mu_i, \sigma_i}(x), \text{ where x is a vector of the input parameters } x_d, x_t, x_e. \tag{9.16}$$

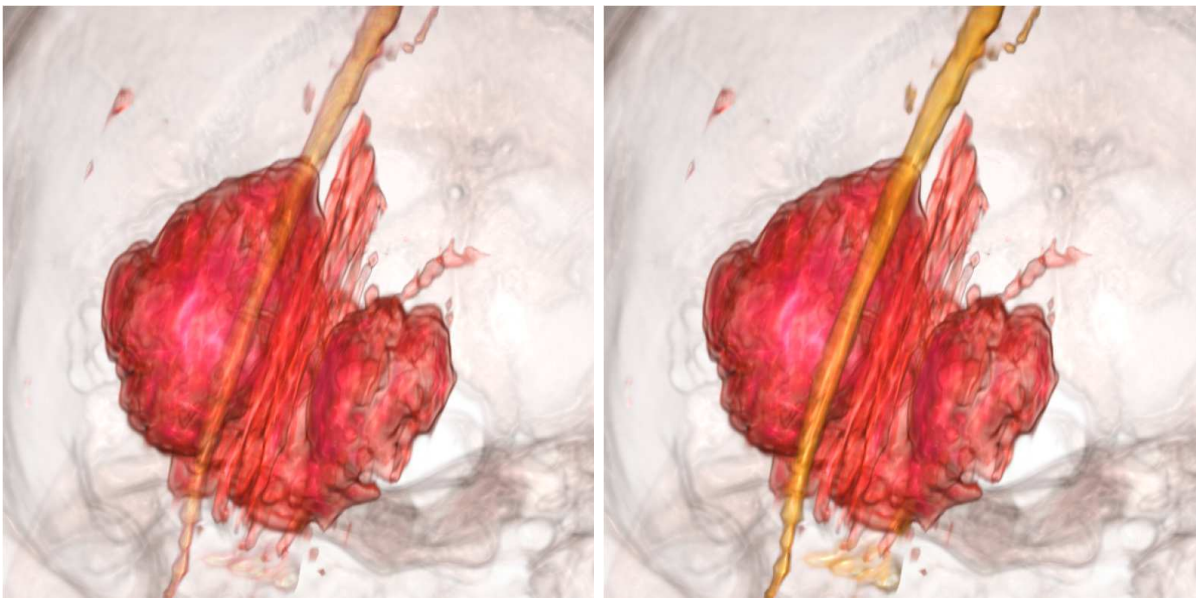
This function consists of several Gaussians with μ being the mean and σ being the standard deviation of the Gaussian function. The Gaussians that are used in the transfer-function are also weighted with factors α_i. The actual search for a minimum consist of iterative steps. In each step a search direction is set as the steepest descent. With this robust method it is possible to find a minimum by iteratively following the steepest descent.

2.8 Results

The results that are created with the rendering approach show an enhanced depth perception of overlapping semi-transparent features, see Fig. 9.10. To apply the presented rendering algorithm to test datasets the following hardware setup was used:

- Intel Core(TM)2 Quad
- NVIDIA GTX 280 GPU

To test the algorithms computational effort the algorithm was applied to different dataset. The data set that was used to generate the results in Fig. 9.10 had a resolution of $(256 \times 256 \times 60)$. Other tests were applied to sets with $(128 \times 128 \times 128)$ and $(600 \times 600 \times 600)$ voxels. The time that was needed to render



(a) Rendered volume without optimization (b) Rendered volume with optimized depth-perception

Figure 9.10: Comparison of normal rendering and optimized rendering of a brain tumor volume [2]

the image data with the enhanced luminance and opacity values ranged from 5 to 20 seconds depending on the size of the test datasets [2]. This result shows one big disadvantage of the presented approach. Even in the minimal time of five seconds the approach is far away from real time rendering. Therefore the rendering approach is only useful for static image data or pre-rendered image sequences. Another disadvantage of this approach is described by the authors. If a three dimensional object is rotated by a user, an image sequence emerges. Within this image sequence objects in the foreground and objects in the background may shift relatively due to the changed perspective. Since the adaption of the color and opacity values is done separately for each image, the optimized output of two sequential images can differ in both adjusted values. During a movement the shift between foreground and background may lead to the introduction of new and the disappearance of existing X-Junctions. This causes the optimization algorithm to handle the same area in a different way from one picture to another.
Nevertheless the approach shows how characteristics of human perception can be used to increase the quality of depth-perception of rendered volumes. On static images, this approach still can be very useful to enhance the quality with only slight adaptations.

3 Enhancing depth-perception with volumetric halos

In contrast to the volume-rendering approach that was presented in Section 2, this approach is based on additional visual cues to support the depth-perception in semi transparent volumes. The term *halo* thereby describes a range of different illustrative features. An example of a halo is shown in Fig. 9.11. On the one hand halos can be used to generate opaque outlines to clearly separate features from their background on the other hand they can be generated as smooth outlines that are just strong enough to support the users depth-perception. While clear outlines are used in computer games or in descriptional images, smooth halos that do not occlude objects in the background with opaque outlines are more suitable regarding the optimization of depth perception. The main idea of the approach that is presented here is to generate halos that can create visual separation between objects in the foreground and objects in the back. Allthough the halos are not physically correct shadows that are cast from one object onto another objects surface the visual effect is very close to a shadow. In this way it gets easier for the user do distinguish the correct depth order of overlapping objects. In Comparison to the concept of changing luminance and transparency values to achieve a better depth perception the halos don't change the appearance of existing objects but introduce new features. Since both appendages of volume rendering should be compared in this paper, the descriptions and examples of the halo based approach are only introduced regarding semi transparent halos. Since opaque halos are, as explained, not sufficient for most

(a) Rendered skull with small halos (b) Rendered skull with strong halos on the outlines

Figure 9.11: Three dimensional model of a skull with halo enhancement [8]

of the needs in medical fields of use.
In order to create halos for an optimized depth perception, different steps of the algorithm that is presented in this chapter need to be executed. In the seeding process the initial positions of the halos are calculated. Aftherwards the halos are generated and then mapped into the volume data before the volume is rendered. In this chapter the different steps of the algorithm are presented as well as results that were generated with this rendering approach.

3.1 Seeding and generating halos

The seeding of the halos is the first step that needs to be done to find locations where the halos are created. This first step approximately corresponds to the edge and X-Junction detection in the first presented approach. The difference between both steps is the dimension of the functions. While the edge detection was in Section 2 done in a preprocessing step and is a function from $\mathbb{R}^2 \rightarrow \mathbb{R}$ the halo seeding is a function from $\mathbb{R}^3 \rightarrow \mathbb{R}$. Since the whole process of the second approach is embedded in the regular rendering of the scene there is no preprocessing needed. This keeps the halo rendering flexible. The halo seeding is implemented as vertex operation. The seeding function takes the view direction $\mathbf{v} \in \mathbb{R}^3$ from the camera position to a voxel and the gradient $\mathbf{g} \in \mathbb{R}^3$ of a voxel to calculate the angle between v and g. For all gradient vectors that are orthogonal (within a certain threshold) to the viewing direction a halo seed point is created. To keep the function more flexible, additional parameters can be used to bound the halos to certain areas or to define the initial intensity values of a halo seed-point. The intensity values of the seed-points are in the later progress used to generate the halos with different color- and opacity values. All points that are not seed points have intensity value 0.
In order to generate a smooth halo, after creating the seed points the halo is spread to its neighboring points. The spreading is done in an iterative process. For each processing step the neighboring voxels of a seed point are also set as seed points with the respective intensity value of the original seed point. In order to define a smooth blending in and out of the halo volume for each step the new generated halo volume is interpolated with the original halo volume as well as with the volume from the previous step. In this way the halo is not generated as a uniform outline but as a smooth transition from the object to its background. In order to terminate the process and limit the spreading, the number of iteration steps is set to a fixed value. The larger this number gets, the larger the halos get. But with a raising

step-number for the spreading also the computational effort raises. For the example images the algorithm was executed with a number of four iterations.

3.2 Mapping and composing halos

To generate the desired visual cues in a given image. The halos that were created in an iterative seeding process have to be mapped to the existing image data. Furthermore the intensity values of the halos have to be mapped to color and opacity values in order generate a useful image. The correlation between intensity values on the one hand and color and opacity values on the other hand, is done by a profile function $P(i)$ where i is the intesity value of a halo seed-point. With i the function calculates a color and an opacity value. Which colors are mapped to which intensity value and also to which opacity can be adjusted by the user. To map the halo volumes to the actual image data, the rendering approach offers two possibilities. For the first case, the *emissive halos* the halos that were adjusted in the previous step are simply integrated as set of voxels into the volume image data. The actual rendering of the halos is then done by the normal rendering pipeline that also renders the rest of the scene.
The second case is slightly more complex. To minimize the intrusive effects of the halos to the rendered image the halos are only integrated into the scene if they occlude objects in the back. Therefore a halo map is generated to detect the regions where the halos occlude objects in the background. This visual result of this case is similar to shadows that are casted from objects in the front onto objects in the back. But since the halos are independent three dimensional structures they do not vary in their appearance regarding the distance to the occluded object. Therefore there is no additional light rendering necessary. The visual effect for the user is still an enhanced depth-perception.

3.3 Algorithm

The halo based rendering approach is composed of different stages from the halo seeding and generation described in Section 3.1 to mapping and composing the halo volumes and the image volume described in Section 3.2. Since volume rendering in a medical field of use is often needed in real time applications, the computational effort is an important factor for the quality of an algorithm. To increase the speed of the rendering, the whole algorithm that is presented in this chapter is embedded in a normal OpenGL rendering pipeline. Since OpenGL offers the possibility to assign the results of a rendering pass to multiple targets, the algorithm can be embedded very effectively. In contrast to a pre- or post processing step the embedded algorithm avoids redundant accesses to the image data. The sequential process of the Algorithm is presented in Figure 9.12.

3.4 Results

In order to test the rendering approach respective to the visual results as well as to the computational effort the algorithm was executed with different datasets. The hardware that was used for the testing is described as:

- AMD Athlon 64 X2 Dual 4600+
- NVidia GeForce 8800 GTX

With the described hardware-setup a reference rendering benchmark was done. The test scene was thereby rendered during a 360 degree rotation. This rotation was performed along each of the three coordinate-axis. After the rendering was done an average frame rate was calculated as reference for the rendering approach with halos. For comparison the test scenario was not changed and the algorithm was performed with four iteration steps at the halo generation. In contrast to the reference frame rate of 29.34 the algorithm achieved an average frame rate of 10.26 fps [2]. Although this frame rate is only about one third of the reference frame rate is is still high enough to perform a satisfying frame rate for a direct user interaction.
In order to present comparable results, the images presented in this chapter only show a selection among all the results that were created. Figure 9.13 shows the different visualizations for the image data of a wrist without halos and with two possible configurations of halo rendering. The image that was rendered with emissive halos shows a very strong illustration of the depth relations. In contrast to the image without halos the depth perception was significantly increased. But the image also comes along with the

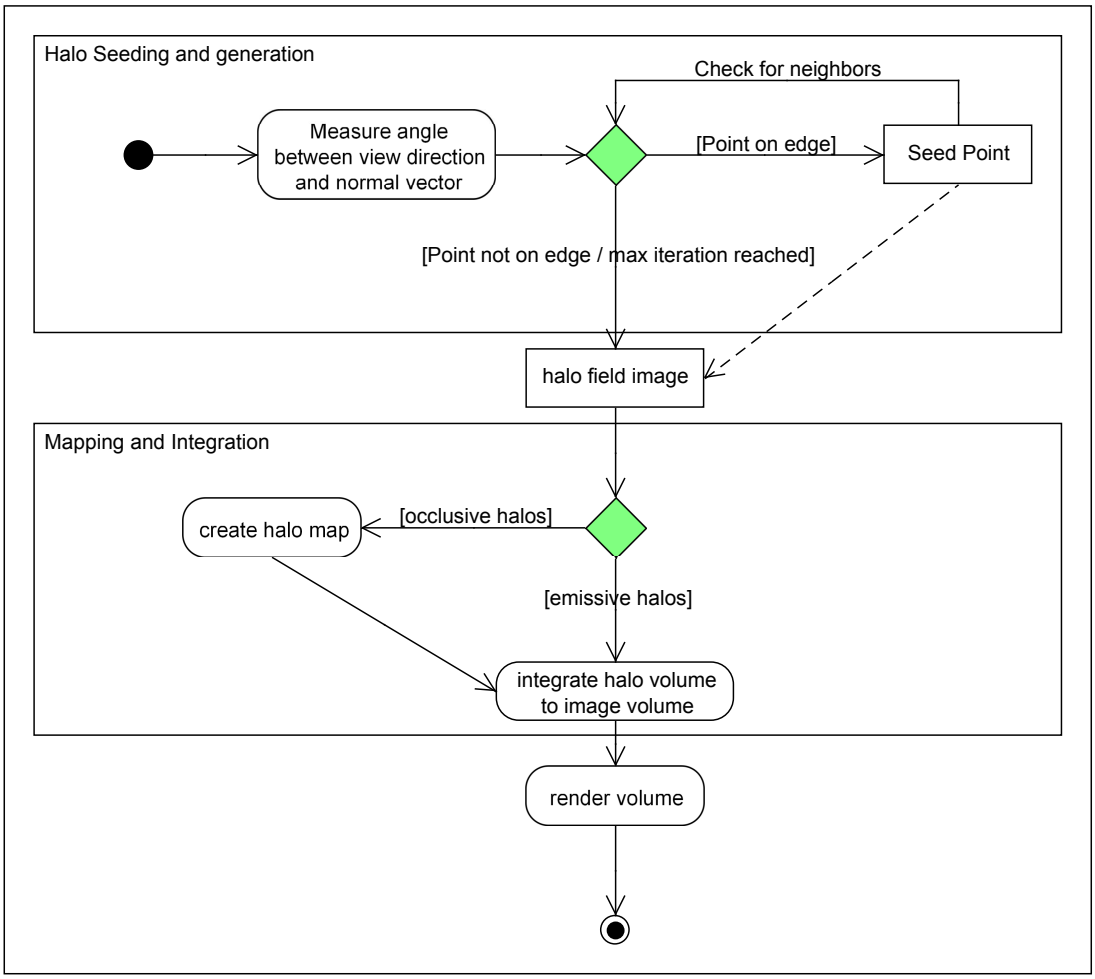
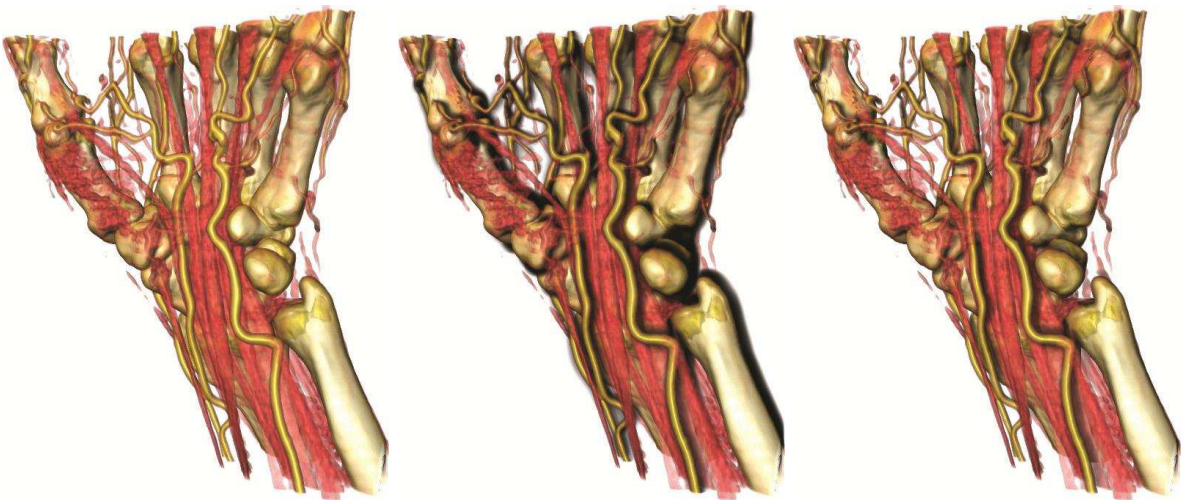


Figure 9.12: Complete Algorithm of the halo based rendering approach. The seeding and halo generation 3.1 is combined in a first step. Afterwards the halos are mapped and integrated to the image data (see 3.2) before the final image is rendered.

disadvantage of occlusion. Especially the bones on the right sight of the image suffer from self occlusion. Since the bone structures direction in some regions goes from the front to the back, the angle between view direction and the normals of the bones surface is close to orthogonal (within a certain threshold) and therefore halo seed points are placed. Along the bones structure the halo volume accumulates its occlusive character and therefore occludes the farther parts of the bone. Since this bone structure only occludes itself in a very small area, the occlusive halo mapping achieves better results. The halo volumes are only mapped to the regions where the occlusion takes place and therefore most of the halo volume that is generated for this special bone structure is not included in the final image. The depth perception in the other parts is still very good in contrast to the original image. For medical purposes the emissive rendering suits the requirement of unambiguous depth perception. Furthermore the occlusion that appears after introducing halos is minimized since many halos are not even mapped to the final image. Another reason is the semi transparent appearance of the halos that allows the user to still recognize features in the back that are occluded by the halos.

4 Comparison

Although the psychological based approach of enhancing the depth perception avoids introducing new visual features, the benefit in its todays state is questionable. One of the advantages that was presented by the authors is that the approach does not introduce additional occlusion, which is true for static images.



(a) Rendering of wrist including muscles, bones and vessel structures

(b) Rendered wrist with emissive halos

(c) Rendered wrist with occlusive halos. As it was described in 3.2, the halos are only mapped where objects in the front occlude objects in the background

Figure 9.13: Comparison of different render configurations

As soon as more than one picture is used for analysis or diagnosis of the visual data, the occlusion on one image is no longer important since features are visible in other pictures. This is also true for sequences of images. For diagnostical purpose it is always necessary not to rely on a single image. One big advantage of volume based rendering is in contrast to classical 2D-visualization that the data can be perceived as three dimensional objects. This kind of data representation intends user interaction (moving camera position, zooming in/out) and therefore more than one picture. One problem that comes along with the idea of user interaction can also occur in picture sequences that are rendered with this approach: As described by the authors the presented method is not designed for image sequences but for single images. The approach extracts the junctions of two objects individually for each image. This may lead to inconsistency in the displayed colors as well as in the displayed transparencies. Since junctions between two objects may disappear from one image of a given sequence to another, the colors of the affected structures may no longer be adjusted in the second image. This problem is also likely to occur if new junctions become visible through the shift of the perspective between two images.
In contrast to that the halo based approach offers a consistent performance on single images as well as for image sequences since the halo volumes are not based on a single intersection point but on several seed points. One disadvantage of the halo based rendering was pointed out in [2]. It regards the occlusion that may occur by integrating halos as new features into an existing image dataset. The results that are presented in Fig. 9.13 show that halos in fact may lead to occlusion. Figure 9.13 (a) shows, how the accumulation of the halo volumes density leads to an occlusion of image features that were visible before the rendering process. Figure 9.13 (b) shows that these occlusions can be minimized by adjusting the parameters of the rendering function. Since the first approach that was presented adjusts opacity values of existing features it seems possible that the adjusting of multiple semi transparent features can also lead to occlusion if the opacities of overlapping features sum up. To investigate an objective comparison of both approaches concerning the occlusion it would be necessary to test both algorithms with the same data. This is also true for the computational effort that is needed for the calculations. Since both approaches were developed independently and there was no normalized test data that was used for both approaches, the comparison of the improved depth-perception can only be done with the results that are shown in 9.10 and 9.13.

4.1 Future Work

As the authors of the first rendering approach [2] describe the problems regarding sequential images could be solved in the future by creating rendered sample images within the sequence. The rendered samples

that have a certain distance to each other (sequentially) could then be interpolated to generate smooth transitions between sequential images. Since it is likely for a good configuration in one image to be also a good configuration in the previous and following image this could decrease the computational effort for the rendering of sequences. Furthermore this could increase the visual appearance of a given sequence since heavy leaps between the opacity of one feature could no longer occur.
Although this was a possibility to enhance the algorithms behavior on image sequences, the idea of interpolating between two (or more) images of a sequence still does not work for real time applications since only the present and past image are known. For a sufficient interpolation the future image would also have to be known. The problems for the real time rendering on the one hand and the quality of the enhanced depth perception show that the rendering approach is not useful in practice today. Although the user study that was accomplished shows a benefit of this approach to the perceived depth-ordering, further studies have to show how depth-perception of non artificial data can be improved. On the computational side of this approach it is also necessary to heavily decrease the computational effort in order to provide usability for applications in practice.

The halo based approach shows very good results in the depth-perception. Since the approach was implemented very flexible it is possible to find a good configuration of the different variable parameters. To adjust these parameters for a special purpose like medical image rendering or even special configurations for several body parts could be done in future user studies. In this way the approach was able to offer a wide range of applications. Although the approach is still embedded in the rendering pipeline there are some optimizations like the free space skipping that was proposed by the authors to increase the speed and the frame rate of this approach.

Literatur

[1] Lorensen, William E. and Cline, Harvey E.: Marching cubes: A high resolution 3D surface construction algorithm SIGGRAPH Comput. Graph. vol. 21, no. 4, pp. 163-169, July 1987

[2] L. Zheng, Y. Wu, and K. Ma.: Perceptually Based Depth-Ordering Enhancement for Direct Volume Rendering. IEEE Transactions on visualization and computer graphics, June 2012

[3] T. Akenine-Moeller, E. Haines and N. Hoffman Real-Time Rendering Third Edition Peters, Wellesley, July, 2008

[4] E.H. Adelson and P. An: Ordinal characteristics of transparency Proceedings of AAAI workshop on Qualitative Vision pp. 77-81, 1990

[5] B.L. Anderson A theory of illusory lightness and transparency in monocular and binocular images: the role of junctions Perception vol. 7, no. 1, pp. 419-453, 1997

[6] J.F. Canny: A Computational Approach to Edge Detection Artificial Intelligence Laboratory, Massachusetts Insitute of Technology, Pattern Analysis and Machine Intelligence vol. PAMI-8 pp. 679-698, November 1986

[7] B.L. Anderson The Role of Occlusion in the Perception of Depth, Lightness and Opacity Massachusetts Institute of Technology, Psychological Review, vol. 110, no. 4, pp. 785-801, 2003

[8] S. Bruckner, and M.E. Gröller.: Enhancing Depth-Perception with Flexible Volumetric Halos. Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria, TR-186-2-07-04, April 2007