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

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

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

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

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

Hauptseminar im Wintersemester 2011/12

Medizinische Bildverarbeitung

Volume 7, Band 1
ISSN 1860-8906
ISBN 978-3-9813213-4-0

Aachener Schriften zur Medizinischen Informatik

Aachener Schriften zur Medizinischen Informatik
ISSN 1860-8906
ISBN 978-3-9813213-4-0

Herausgeber: Institut für Medizinische Informatik der RWTH Aachen
Pauwelsstr. 30
D-52074 Aachen

Geschäftsführender Direktor: Universitätsprofessor Dr. Dr. Klaus Spitzer

Preface

With a history of more than 20 years the Seminar Medical Image Processing has been established as a core seminar of higher education for students of Computer Science, Media Informatics, Software Systems Engineering, also supporting recently established master programs for international students.

Hence this year we turned the seminar to English language. All presentations were performed in English and again, the seminar was subdivided into a practical and a theoretical part co-supervised by Professor Ney and Professor Rossmanith, respectively.

Medical image processing is still an emergently growing field of research. The topics that have been selected this year were taken from recent academic literature. After handing over the student one or two papers, they were asked to join a specific course on scientific literature retrieval allowing them to find secondary literature autonomously. Equipped with these material, the students were asked to report on their topic. Here, all local advisors emphasized the fact that the students have established a suitable critical distance to the authors of their initial papers; not confusing the authors' and their own work. Furthermore, the question was to detect and emphasize weak points in the algorithms or evaluations presented by the original authors of the papers.

All six students that ended their seminar by their presentation successfully completed the entire course. Since we are now performing a corresponding lecturer in English language, that aims at providing baseline information in medical image processing, visualization and analysis, we assume that our Seminar Medical Image Processing – although existent now for more than 20 years – will have a bride future with steadily increasing number of interested students.

Aachen im April 2012

T. Deserno

Inhalt

Hauptseminar Medizinische Bildverarbeitung

Vorträge der Blockveranstaltung im Sommersemster 2012

praktischer Teil/Prof. Ney, 24.01.2012

14:00-14:45	Roubust Tracking	Referent: *Ramya Nagarajan*
14:45-15:00	Diskussion	Betreuer: *Jens Forster*
15:00-15:45	Active Shape Models (for Medical Images Segmentation)	Referent: *Tonima Mukherjee*
15:45-16:00	Diskussion	Betreuer: *Yannick Gweth*
16:00-16:45	Near-Realtime Stereo Vision	Referent: *Martin Lang*
16:45-17:00	Diskussion	Betreuer: *Harald Hanselmann*
17:00-17:45	Contour based shape matching (for object detection and recognition)	Referent: *Ludger Steens*
17:45-18:00	Diskussion	Betreuer: *Harald Hanselmann*

theoretischer Teil/Prof. Rossmanith, 26.01.2012

08:30-09:15	An optimization problem in virtual endoscopy	Referent: *Safoura R. Lakani*
09:15-09:30	Diskussion	Betreuer: *Felix Reidl*
09:30-10:55	Dynamic Programming Algorithms for Efficiently Compution Cosegmentations between Biological Images	Referent: *Archana Kumari* Betreuer: *Alexander Langer*
10:15-10:30	Diskussion	

Seminar Report on Dynamic Programming Algorithms for Efficiently Computing Cosegmentations between Biological Images

Archana Kumari

Inhalt

Zusammenfassung

In field of biomedical image analysis, there has been a lot of work which uses the idea of comparing trees. Although many of them showed promising results, but they were not very suitable due to their complexity. [18] proposes two dynamic programming algorithms for efficiently computing cosegmentations between component trees representing individual time frames using the so-called tree-assignments. The first algorithm introduces a restricted version of test data, which reduces on the complexity and still gives good results. The second one proposes some weight computations between so-called component trees that can be applied to obtain certain cosegmentations in bioimaging applications. The authors have verified the results over a large collection of image data imposing the restrictions by dynamic programming formulations. Their experiments prove that this model outperforms the state-of-the-art approaches.

Keywords: *Component Trees, Tree-Constrained Bipartite Matching problem, Cosegmentation of Image Pairs,Tree Edit Distance*

1 Introduction

The use of medical images has become a key point in field of medical diagnosis. Cell imaging is one among the many imaging techniques used in interpretation of medical images. There are various technologies available for 3D study of cell images. Studying cell motility has become an important factor in understanding numerous biological processes, driven by the rapid development of bio-imaging technology. This involves a lot of comparison tasks, for which trees are used. In recent past, there has been a lot of work on the comparison of RNA structures. There are several ways to model RNA structures, like trees, stochastic context-free grammars (SCFGs) etc. Trees are used extensively for this purpose. There are many approaches proposed to ascertain the similarity between different RNA structures(represented using trees or SCFGs). The authors in [13], have used tree comparisons for comparing multiple RNA secondary structures. They typically compute the distance between two trees. A special type of tree called Component tree is commonly used for image filtering and segmentation applications [4]. One of the proposed methods to deal with background inhomogeneities and objects of varying intensities, is to pick local thresholds in a hierarchical representation of all possible thresholds of an image. The component trees are compared by solving the so-called tree assignment problem, a natural generalization of bipartite matchings and the associated assignment problem [15]. Comparing component trees by computing tree assignments yields a cosegmentation of two images, for example for cell tracking it yields cosegmentations between two time frames in a video sequence. The authors in [10], have used topological alignments to link segmentations of two consecutive frames in the video sequence. For bioimaging applications, generalization of the classical bipartite matching problem appears, which is also referred as the Tree-Constrained Bipartite Matching problem. Recently some authors have shown this as an NP-hard problem [2]. However, another approach using integer linear programming works good for small data set [10, 15]. Still, for the real time application, the size of data obstructs the performance. The constrained versions of the problem which can be solved in polynomial time is of high relevance in practice. [18] proposes a restricted version of the problem that is solved in polynomial time using a dynamic programming formulation. They have use the same restriction which was used in context of tree edit distance [1] by Philip Bille. [18] extends the class of tree alignments and distance measures surveyed in [1]. For bioimaging applications, component trees can be used to model fluorescence-based images. An image can be filtered using component trees. Filtering the tree is a decision process which classifies nodes into those that are to be removed and those that are to be preserved [6]. The obtained component tree requires near-linear time using a union-find data structure [11].

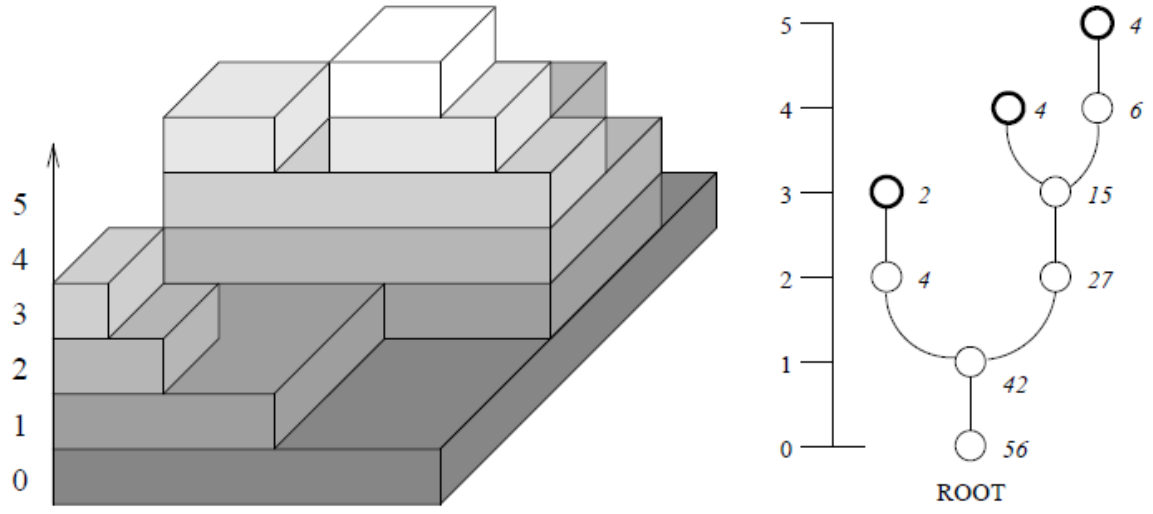
In [15], The authors aim to find maximum-weighted sets of pairwise compatible assignments between vertices of two component trees, which is termed as the Jaccard index. A naive approach to this problem would need at least quadratic time to compute the weights between all pairs of vertices. [18] proposes an optimal dynamic programming algorithm, with run time proportional to the number of pixels in the image plus the number of non-zero overlap weights. In an approach similar to the cosegmentations introduced in [15], Mattes et al. [9] also used the idea of computing assignments between vertices of component trees. The authors apply a density based clustering method in order to obtain a tree which classifies the points on which the gray level function is defined. Secondly, They use the identification of the hierarchical representations of the two images to guide the image matching or to define a distance between the images for object recognition. The exact constraints on the matching results remain uncharacterized

in their approach. In [15], the authors use the tree assignments to obtain the cosegmentations in context of cell tracking. They have microscopic video sequence of N time frames. They need to solve the $N-1$ cosegmentation instances. In [18] they have introduced the restriction to the three-point condition imposed by the dynamic programming approach. To assess the efficiency of this approach, the authors have evaluated their approach using the same data set as in [15]. Interestingly, there is also theoretical evidence that the three-point condition does not restrict meaningful results as long as background inhomogeneity (a common phenomenon in fluorescence microscopy) is similar among the cosegmented images.

2 Key Terms

2.1 Component Trees

Introduced to image processing by Jones [6], the component tree is a representation of a gray-level image that contains information about each image component and the links that exist between components at sequential gray-levels in the image. The authors define a component tree as a set of nodes connected by a set of edges. Each node in the tree represents a particular component in the gray-level image. The node is an abstract representation of the component that may be anything from a list of all the pixels in the component to a single attribute value such as component area. An example of a component tree is illustrated in Fig.1(a), corresponding to the small gray-level image shown in Fig.1(b). For every component in that image, there is a corresponding node in the tree, represented by the circles in the figure. The root of the tree is shown at the bottom of the figure and the leaves of the tree are indicated by the circles in bold at the top of the figure. A branch is defined as the shortest sequence of linked nodes from any given leaf down to the root of the tree. There are three branches in the tree, corresponding to the three leaves. The lines drawn between the nodes are the edges and show the links between components in the image. As an example of the attribute values that can be stored in the nodes, component area is shown in italics to the right of each node.



(a) (b)

Abb. 1.1: *The component tree. (a) Perspective view of an example image. (b) Corresponding component tree.*

The process of filtering and segmenting an image using a component tree can be been divided into the three stages: (1) Constructing the tree from the image; (2) Filtering the tree (specifically, using an attribute signature); (3) Mapping the filtered tree to an output image.

2.2 Tree-Constrained Bipartite Matching problem

The Tree-Constrained Bipartite Matching (TCBM) problem is a natural generalization of the classical maximum matching problem in bipartite graphs. It arises naturally in the computational analysis of live cell video data. Studying cell motility using live cell video data helps understand important biological

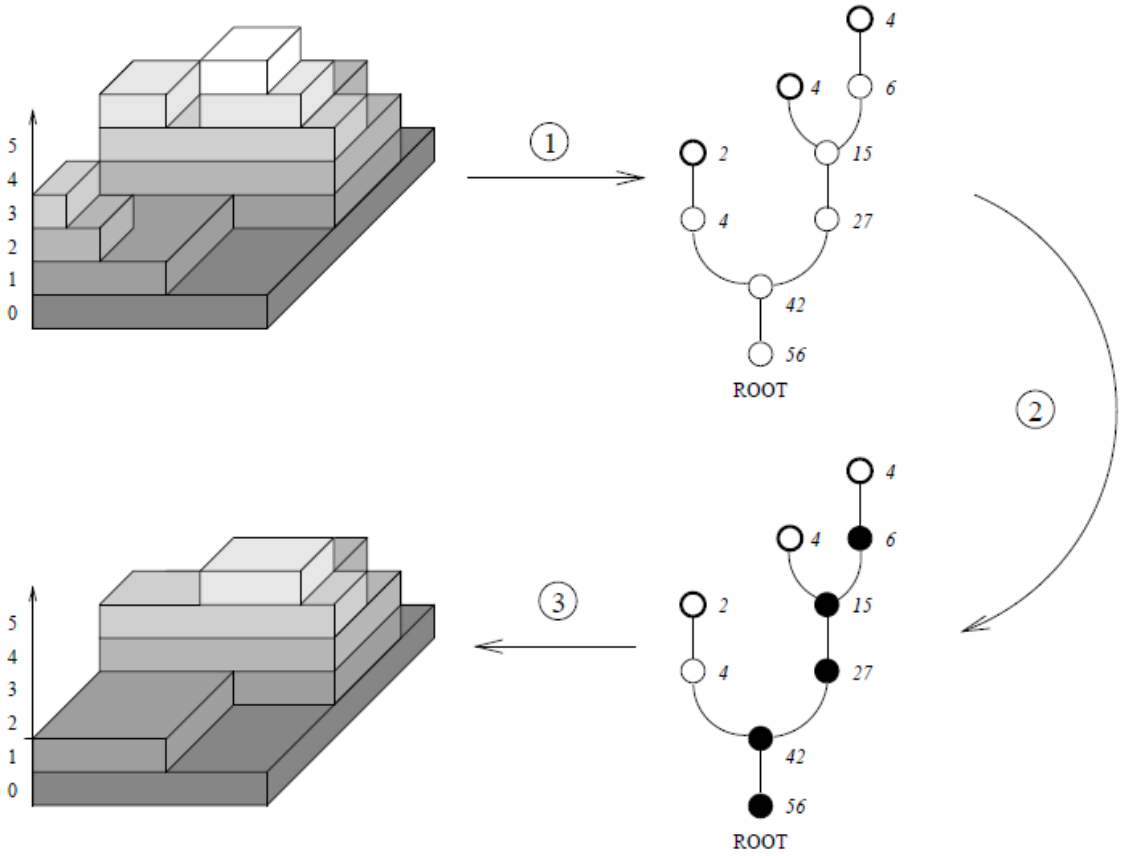


Abb. 1.2: The three stages of filtering and segmenting an image using a component tree.

processes, such as tissue repair, the analysis of drug performance, and immune system responses. Segmentation based methods for cell tracking typically follow a two stage approach: The goal of the first detection step is to identify individual cells in each frame of the video independently. In a second step, the linkage of consecutive frames, and thus the tracking of a cell, is achieved by assigning cells identified in one frame to cells identified in the next frame. However, limited contrast and noise in the video sequence often leads to over-segmentation in the first stage: a single cell is comprised of several segments. A major challenge in this application domain is therefore the ability to distinguish biological cell division from over-segmentation.

The input of TCBM consists of a weighted bipartite graph $G = (V_1, V_2, E)$ and two rooted trees T_1 and T_2. The vertex set of T_i is v_i for $i = 1, 2$. The objective is to find a maximum weight matching such that the matched vertices in each tree are not comparable; that is, if $u, v \in V_i$ are matched then u cannot be v's ancestor or vice-versa. Fig.2 illustrates the definition.

2.3 Cosegmentation of Image Pairs

Rother et al. [17], introduce the term cosegmentation which denotes the task of segmenting (Differentiating the subdivisions of an organism or part) simultaneously the common parts of an image pair. The authors have presented a generative model for cosegmentation. To differentiate the background from the subjects, it is essential to incorporate some form of differentiation of parts of images, so that comparison can be based on those parts of an image pair which are shared in common. In that way, a similarity between subjects can be scored highly, without unreasonable dilution by differences in backgrounds. Another approach can be to capture the similarities in the background scenes of a pair of images, despite the subjects being unrelated. In [17], the authors have used an approach called integrated region matching, in which images are subjected to mean-shift segmentation, and then a simple similarity measure records the similarity of paired regions, in a search over both segmented images. However, the choice of paired regions takes no account of object coherence, and so cannot properly take account of the distinction between subject and background. The authors have addressed this shortcoming by jointly cosegmenting

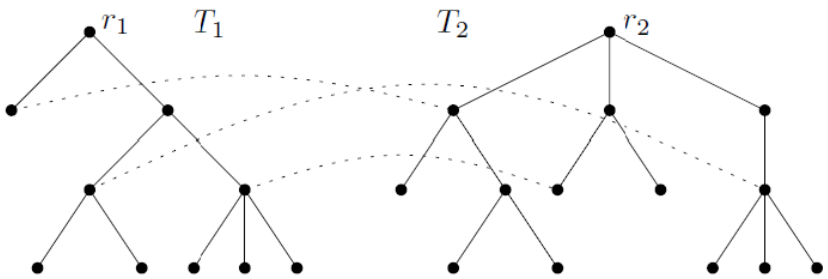


Abb. 1.3: *Example of a feasible tree-constrained bipartite matching. For each matched pair of vertices, indicated by dotted lines, neither of their descendants are matched.*

the image pair using a proper MRF coherence prior and a histogram matching cost, and then compare either subject or background. Next step is to find a coherent image region with given target histogram. They have defined coherence via MRF priors and solve the problem with iterated graph cuts, called trust region graph cuts.

2.4 Tree Edit Distance

The edit distance between two trees is defined by considering the minimum cost edit operations (local operations of deleting, inserting, and relabeling nodes required to turn one tree into another) sequence that transforms one tree to another. Formally, the edit distance between T_1 and T_2 is defined as

$$D(T_1, T_2) = \min_S \ \{\gamma(S) \mid S \textit{ is an edit operation sequence taking } T_1 \textit{ to } T_2.\}$$

3 Background

Cell tracking plays vital role in various analytical processes like understanding cell cycle, neuronal division and migration, immune response, development of cancer etc. Based on phase-contrast, confocal or two-photon microscopy, such live cell imaging protocols are now commonly established and corresponding equipment is commonly available. This arises the need for computational methods to analyze the motility of cells captured using the time-lapse microscopy. In the whole process, the basic task is to identify the individual cells and to track their identities over time. The process of cell tracking faces various challenges because of unpredictable cell behavior like cells devisions, cell entering and leaving the display area. It gets even more complicated by background inhomogeneity, for instance due to uneven illumination and cells touching each other. In [15], Xiao et al. have introduced a novel algorithm for cell tracking that allows to track cells, in particular zebrafish microglia, in 3D two-photon image sequences over time. Their algorithm can be considered as a broad generalization of thresholding methods. Some previous methods differentiate the background and foreground using a threshold intensity. The pixel intensities below threshold are treated as background while the pixel intensities above this threshold are treated as foreground. There are some other existing methods (such as locally adaptive thresholding etc) to deal with background inhomogeneities and objects of varying intensities. The authors in [15], have proposed highly systematic way of picking local thresholds in a hierarchical representation of all possible thresholds of an image, the so called component tree. They have compared the component trees of consecutive time frames by solving the tree assignment problem. Comparing component trees by computing tree assignments yields a cosegmentation of two images, which is of high relevance for cell tracking, cosegmentations between two time frames in a video sequence etc. They use their own approach for cosegmentation, which is quite different from the original histogram comparing approach. Xiao et el. [15] find an optimal tree assignment to identify overlapping regions in two images.
Their cell tracking algorithm in [15], takes as input the sequence of images, pruning parameters and the singlenode cutoff. The output is a sequence of segmented images. It goes through the following steps:

1. Compute component tree for all the input images.
2. Prune the component tree using the pruning parameters and the singlenode cutoff.

3. For each consecutive set of trees, compute the tree assignment.
4. Use the consecutive tree assignments to obtain two segmentations of image and compute consensus segmentation from the consecutive tree assignments.
5. Compute a maximum-weighted bipartite matching between the consecutive segments
6. Assign random color to each segment in the first calculated segment and use the same color to same segment in the next computed segments to track segment (cell) identities over time..

Figure 4 illustrates the basic steps of this cell tracking algorithm.

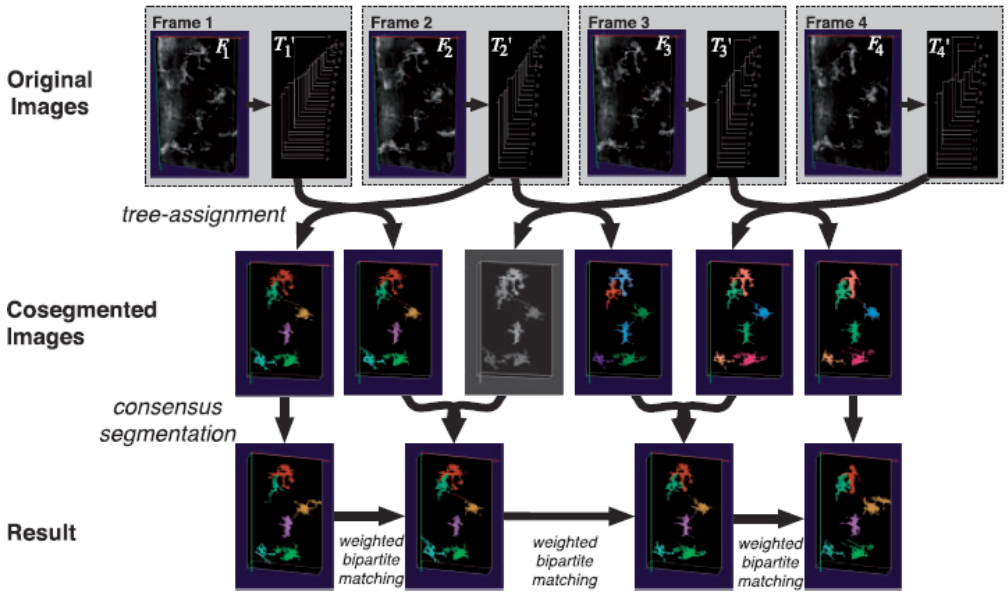


Abb. 1.4: *Overview of complete cell tracking algorithm*

4 Problem Formulation

The authors in [18] have used the approach of tree assignments along with some other constraints. This section presents the idea used by them in [18].
General tree assignments : Let S and T denote two rooted unordered trees, with vertices U and V, respectively. [18], deals with tree assignments between two trees, which are sets $M = \{(u_1, v_1), \ldots, (u_k, v_k)\} \subset U \times V$ such that for any two distinct indices $1 \leq i, j \leq k$, neither u_i is an ancestor/descendant of u_j nor v_i is an ancestor/descendant of v_j. The authors in [18], refer to the set of all possible assignments between S and T as $match(S, T)$. Given a weighting function $w : U \times V \to \mathbf{R}_{\geq 0}$ that assigns a score $w(u, v)$.

Whenever u is matched with v they assign a weight $W(M) := \sum_{u,v \in M} w_{u,v}$ M. The above setup defines their tree assignment problem (in [18]), which is to find the maximum weighted tree assignment, given S, T and w. The tree assignment problem is a generalization of the maximum weighted bipartite matching problem.
Constrained tree assignments: The tree assignment problem can be modelled and solved using integer linear programming [15]. Recently this problem has been shown to be NP-hard [2]. The authors have considered the constrained tree assignment inspired by the constrained tree edit distance [16]. They found that the constrained version of the tree assignment problem is solvable in polynomial time like tree edit distance problem. They named the contraint as the three-point condition because this contraint introduces a restriction on the topology of trees with three leaves. They use recursions for the constrained tree edit distance to solve the restricted tree assignment problem.
The three-point condition involves the lowest common ancestor of two vertices a and b in a tree, which is denoted by $lca(a, b)$. A match $M \in match(S, T)$ is considered as *constrained tree assignment* if for any three assignments (a_1, a_2), (b_1, b_2), $(c_1, c_2) \in M$ the following holds,

$$lca(a_1, b_1) = lca(a_1, c_1) \Rightarrow lca(a_2, b_2) = lca(a_2, c_2)$$

The three-point condition is applied to establish the idea that the topology of the two induced subtrees are identical for any three pairs of vertices in a tree assignment. This implies that distinct subtrees in one tree are assigned to distinct subtrees in the other tree.
The authors have used a short hand notation, $cmatch(S,T)$, which refers to the set of all constrained tree assignments between S and T. Corresponding to the tree assignment problem, the *constrained tree assignment* problem finds the maximum weighted constrained tree assignment between two trees based on a weight function w.

Component Trees: The authors in [18], have used Component trees in their image analysis setting. They have computed these component trees $I: P \to \{0,\ldots,q-1\}$, where P is the set of all image coordinates and q is the number of gray values. For a threshold $0 \leq \theta < q$, the pixel set $\{p \in P | I(P) \geq \theta\}$ falls apart into connected components (e.g. with regard to the 4-neighborhood or 8-neighborhood of pixels in a 2D image). The connected components for all possible thresholds $0,\ldots,q-1$ are hierarchically ordered, defining the component tree in which each vertex represents one connected component. In a component tree, each vertex v is associated with a (connected) set of pixels, we denote this set by $\beta(v)$.
Leong et al. [7] introduces an interpretation of constrained tree assignments between component trees involving background inhomogeneities that are of high relevance in microscopic images. This interpretation indicates that the restrictions imposed on constrained tree assignments does not affect results as long as background inhomogeneity (which can be dealt with using rolling-ball-type algorithms [14]) is consistent among the two underlying images, as illustrated in Fig.5.

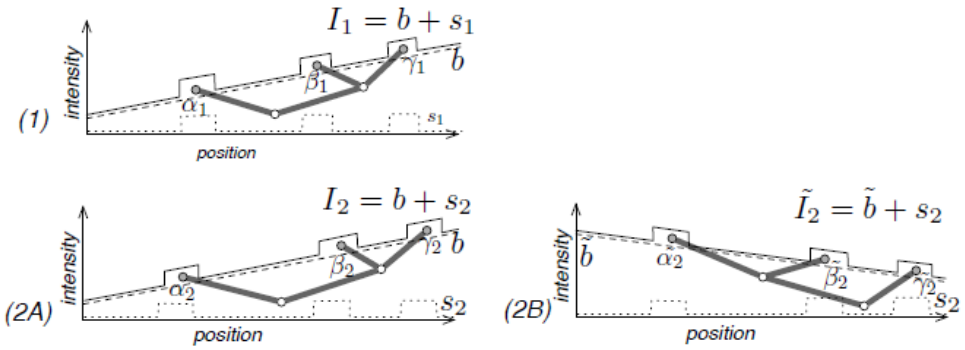


Abb. 1.5: *Relationship between the three-point condition and background inhomogeneity in images: (1) Often, a biological image such as I_1 contains an inhomogeneous background b in addition to the actual (e.g. fluorescence) signal s_1, which contains three objects that are represented by vertices α_1, β_1, γ_1 in the corresponding component tree. (2A) Given a second image $I_2 = b + s_2$ with same background b and similar objects represented by vertices α_2, β_2, γ_2, a tree assignment involving the three assignments (α_1,α_2), (β_1,β_2), (γ_1,γ_2) respects the three-point condition. 2B) If, however, there is an image I_2 with same signal s_2 but completely different background $\tilde{b}$, an assignment involving the three assignments (α_1,$\tilde{\alpha}_2$), (β_1,$\tilde{\beta}_2$), (γ_1,$\tilde{\gamma}_2$) will violate the three-point condition.*

Calculating overlap weights: Computing the vertex weights between two given component trees S and T is also an important part of obtaining cosegmentation. For simplicity, the authors have used the same symbol β for the vertex to pixel mappings of S and T. So, for vertex $u \in S$, $\beta(u)$ refers to a pixel set in image corresponding to S and for vertex v in T refers to a pixel set in image underlying T. A weighting function to be used for cosegmentation, as introduced in [15], is the Jaccard index:

$$w(u,v) = |\beta(u) \cap \beta(v)| / |\beta(u) \cup \beta(v)|$$

Before solving either the constrained or the unconstrained tree assignment problem, it is necessary to compute the weights between all pairs of vertices.

5 A Quasi-Linear Time Algorithm for Computing Overlap Weights

The time complexity in weight calculation between the pairs of vertices is relatively higher than the complexity of the rest steps. In practice, this step limits the performance of the entire process. Optimization

of this step will enhance the performance of overall cosegmentation process in terms of time. This section presents the algorithm introduced the authors in [18], to compute overlap weights. They propose a linear time dynamic programming algorithm for calculating overlap weights by systematically utilizing the inclusion relationship of a node and its children in a component tree. The authors in [18], have described the component tree T as $T = (V, E, \alpha, \beta)$ where V is the set of vertices in tree T, and α, β are two mappings from vertices to sets of pixels in an image. For each vertex v, $\beta(v)$ is the connected area in image, which is associated with v, while α(v) is the connected area of v excluding the connect areas of the children of v. The pixels in $\alpha(v)$ belong exclusively to v and not to any of its children. In this setup v_i denotes the i^{th} child of vertex v and $C(v)$ denotes the set of children of v. They have formulated it as eq.(1).

$$\beta(v_i) \cap \beta(v_j) = \phi \ \forall v_i \neq v_j \in C(v) \tag{1.1}$$

Hence, $\beta(v)$ can be decomposed as follows:

$$\beta(v) = \alpha(v) \cup \bigcup_{v_i \in C(v)} \beta(v_i)$$

That means $\beta(v)$ can be partitioned into $\alpha(v)$ and $\beta(v_i)$ where $v_i \in C(v)$. The authors have proposed this as the guiding principle of the dynamic programing algorithm for weight calculation. They consider $S = (V_S, E_S, \alpha, \beta)$ to be the first component tree and $S = (V_T, E_T, \alpha, \beta)$ to be the second component tree. Next they compute the Jaccard index between each vertex $u \in S$ and vertex $v \in T$,which is defined as the eq.(2) below.

$$w(u, v) = |\beta(u) \cap \beta(v)| / |\beta(u) \cup \beta(v)| \tag{1.2}$$

They re-write the above equation as the eq.(3) below.

$$w(u, v) = \frac{|\beta(u) \cap \beta(v)|}{|\beta(u)| + |\beta(v)| - |\beta(u) \cap \beta(v)|} \tag{1.3}$$

They rephrase the weight calculation between vertex $u \in S$ and vertex $v \in T$ as calculating intersections between $\beta(u)$ and $\beta(v)$. As a shortcut notation, they define cardinality of the intersection of $\beta(u)$ and $\beta(v)$ as

$$\beta\beta(u, v) = |\beta(u) \cap \beta(v)|$$

Similarly,

$$\alpha\beta(u, v) = |\alpha(u) \cap \beta(v)|$$

$$\alpha\alpha(u, v) = |\alpha(u) \cap \alpha(v)|$$

After decomposing $\beta(v)$ into $\alpha(v)$ and $\beta(v_i)$, they split $\beta\beta(u, v)$ as

$$\beta\beta(u, v) = \alpha\beta(u, v) + \sum_{u_i \in C(u)} \beta\beta(u_i, v)$$

While $\alpha\beta$(u,v) is split as

$$\alpha\beta(u, v) = \alpha\alpha(u, v) + \sum_{v_i \in C(v)} \beta\beta(u, v_i)$$

For the proposed dynamic programming algorithm, the authors have used three dynamic programming tables $\alpha\alpha$, $\alpha\beta$, and $\beta\beta$. Based on the dependency relationship between them, the authors compute them in the following order:

$$\alpha\alpha(u, v) \rightarrow \alpha\beta(u, v) \rightarrow \beta\beta(u, v)$$

The authors have assumed P as the set of all pixels in an image, and $T = (V, E, \alpha, \beta)$ as its component tree. Then, $\alpha(v)$ is a partition of P.

$$P = \bigcup_{u \in V} \alpha(v) \tag{1.4}$$

Using the above equation the authors define a reverse mapping $\alpha^{-1} : P \rightarrow V$ is defined, which identifies for each pixel $p \in P$, the unique vertex v that satisfies $p \in \alpha(v)$. Using this reverse mapping, they have calculated all $\alpha\alpha(u, v)$ in time $O(|P|)$. Based on the above recurrence relations, both $\beta\beta(u, v)$

and $\alpha\beta(u, v)$ were calculated in a dynamic programming fashion by postorder traversal of the component trees (see Algorithm 1). This led to a time complexity of $O(|S| \cdot |T|)$.
Worst-case time complexity: For the whole weight calculation process, they first compute all $\alpha\alpha$ weights, then the $\alpha\beta$ weights, and finally the $\beta\beta$ weights, which yields a total running time of $O(|P| + |S| \cdot |T|)$. From eq.(4), we get

$$|V| = \frac{|P|}{avg_{v \in V}(|\alpha(v)|)}$$

$|V| \leq |P|$ always holds, since $avg_{v \in V}(|\alpha(v)|) \geq 1$.

For large images ($|P| \approx 10^6$), such as the ones considered in Section 5, the full component tree contains about 10^5 vertices. However, after pruning, the tree size decreases dramatically, to typically much less than 500, as observed in [15]. Usually, for large images, they have $avg_{v \in V}(|\alpha(v)|) \geq |V|$, so $|V|^2 << |P|$. Under these circumstances, the total running time, dominated by the number of pixels, is quasi-linear w.r.t $|P|$.

6 Dynamic Programming Algorithm for Constrained Tree Assignment

Constrained tree assignments can be considered to be a special case of the constrained edit distance [16], where assigning node u to node v is equivalent to changing the label of node u to the label of node v. The cost of changing node u to v is $w(u, v)$ and the rest of the operations have zero cost. In addition,once node u is changed to v, the descendants of u and v need not be considered anymore. Hence, the proposed dynamic programming algorithm in [18], is a simplification of the dynamic programming algorithm for computing constrained edit distance between unordered labeled trees [16]. They assume T_u to be the tree rooted at node u and F_v to be the forest of the subtrees of u. They refer to T_v as a subtree of another tree S if vertex v belongs to some tree S. $M(T_u, T_v)$ is the optimal assignment between the two trees T_u and T_v and $W(T_u, T_v)$ is the score of the optimal assignment. They have introduced the following to establish the recurrence relation lemmas for W.
Lemma 1.

$$W(T_u, T_v) = max \begin{cases} W(u, T_v) \\ W(T_u, v) \\ W(F_u, F_v) \end{cases}$$

$$W(u, T_v) = max \begin{cases} w(u, v) \\ max_{y \in C(v)} w(u, T_y) \end{cases}$$

$$W(T_u, v) = max \begin{cases} w(u, v) \\ max_{x \in C(u)} w(T_x, v) \end{cases}$$

Algorithm 1: Computing all overlap weights between component trees S and T

Compute post-order enumerations of the vertices in S and T as $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ respectively
{$\alpha\alpha$ weights calculation}
Initialize all $\alpha\alpha(u, v)$ to 0

for each pixel $p \in P$ **do**
  $u := \alpha_1^{-1}(p)$
  $v := \alpha_2^{-1}(p)$
  increase $\alpha\alpha(u, v)$ by one
end for
{$\alpha\beta$ weights calculation}
for $i := 1 \rightarrow n$ **do**
  for $j := 1 \rightarrow m$ **do**
    Initialize $\alpha\beta(u_i, v_j)$ to $\alpha\alpha(u_i, v_j)$
    for each child $c \in \mathrm{C}(v_j)$ **do**
      $\alpha\beta(u_i, v_j) := \alpha\beta(u_u, v_j) + \alpha\beta(u_i, c)$
    end for
  end for

end for
{$\beta\beta$ and Jaccard index calculation}
for $i := 1 \rightarrow n$ **do**
for $j := 1 \rightarrow m$ **do**
Initialize $\beta\beta(u_i, v_j)$ to $\alpha\beta(u_i, v_j)$
for each child $c \in C(u_i)$ **do**
$\beta\beta(u_i, v_j) := \beta\beta(u_u, v_j) + \beta\beta(c, v_j)$
{ $|\beta(u)|$ and $|\beta(v)|$ are tracked during the building of the component trees}
$w(u_i, v_j) := \beta\beta(u_i, v_j)/(|\beta(u_i)| + |\beta(v_j)| - \beta\beta(u_i, v_j))$
end for
end for
end for

Proof: Consider the nodes u and v, there are three possible cases: (1) $u \in M$, (2) $v \in M$, and (3) $u \notin M$ and $v \notin M$.
Case 1: ($u \in M$): Node u is matched to some node in T_v. In order to maximize the objective function node, u must be matched to some node x in T_v that maximizes $w(x, v)$.
Case 2: ($v \in M$). Similar to case 1.
Case 3: ($u \notin M$ and $v \notin M$). Since both u and v are not in M, we can remove them and find an optimal assignment between the remaining forests, F_u and F_v.

Algorithm 2: Computing the optimal assignment between two trees S and T

Compute post-order enumerations of the vertices in S and T as $u_1, \ldots, u_n$ and $v_1, \ldots, v_m$ respectively
{Matching a node u_i in S to all subtrees of T}
for $i := 1 \rightarrow n$ **do**
for $j := 1 \rightarrow m$ **do**
$W(u_i, T_{v_j}) = w(u_i, v_j)$
for each child $c \in C(v_j)$ **do**
$W(u_i, T_{v_j}) := \max(W(u_i, T_{v_j}), w(u_i, T_c))$
end for
end for
end for
{Matching a node v_j in T to all subtrees of S}
for $j := 1 \rightarrow m$ **do**
for $i := 1 \rightarrow n$ **do**
$W(u_i, T_{v_j}) = w(u_i, v_j)$
for each child $c \in C(v_j)$ **do**
$W(T_{u_i}, v_j) := \max(W(T_{u_i}, v_j), w(T_c, v_j))$
end for
end for
end for
{Matching a subtree of S to a subtree of T}
for $i := 1 \rightarrow n$ **do**
for $j := 1 \rightarrow m$ **do**
Construct a weighted bipartite graph $G = (U \cup V, E)$, where $U = F_{u_i}$, $V = F_{v_j}$, and
$E = \{(T_x, T_y, W(T_x, T_y)) | T_x \in F_{u_i} and T_y \in F_{v_j}\}$
$M_G = MaxWeightedBipartiteMatching(G)$
$W(T_{u_i}, T_{v_j}) := \max(W(u_i, T_{v_j}), W(T_{u_i}, v_j), \sum_{(T_x, T_y) \in M_G} W(T_x, T_y))$
end for
end for

Lemma 2: Let $\tilde{M}$ be the set of all possible matchings between the trees in F_u and the trees in F_v.

Then,

$$W(F_u, F_v) = \max_{M \in \tilde{M}} \sum_{(T_x, T_y) \in M} W(T_x, T_y)$$

Proof: The key property of the three-point condition is that distinct trees in F_u is assigned to distinct trees in F_v, hence it suffices to consider one-to-one matchings between the trees in F_u and F_v. Since the aim is to maximize the final assignments, the authors in [18], maximize over all possible one-to-one matchings between trees in F_u and trees in F_v. This is precisely the *maximum weighted bipartite matching* problem where the two partite sets are the trees in F_u and F_v respectively and the weight between two trees is given by W.

In [18], they first compute $W(u, T_v)$ and $W(v, T_u)$ for all possible pair of nodes u and v, where u is in S and v is in T using dynamic programming. Then next step is to compute $W(T_u, T_v)$ by solving a maximum weighted bipartite matching problem and combining that with the results from the previous step. The pseudo code for the whole algorithm is listed in Algorithm 2.

Worst-case time complexity: The number of subproblems in W is $O(|S| \cdot |T|)$ and except for the computation of the maximum weighted bipartite matching, each subproblem can be solved by taking the maximum of a fixed number of cases. Therefore, the bottleneck in this algorithm is in the computation of maximum weighted bipartite matching. The worst-case time complexity for computing the maximum weighted bipartite matching on a graph with n vertices and m edges is $O(n(m+n \log n))$ [3], thus the worst case time complexity for computing the optimal assignment between F_u and F_v is $O((n_u + n_v)(n_u n_v + (n_u + n_v) \log(n_u + n_v)))$, where n_u is the number of children of node u and n_v is the number of children of node v. Hence, the worst-case time complexity of this algorithm is

$$\begin{aligned}
&\sum_{u \in U} \sum_{v \in V} C \;\; X \;\; (n_u + n_v)(n_u n_v + (n_u + n_v) \log(n_u + n_v)) \\
&\leq \sum_{u \in U} \sum_{v \in V} D \;\; \cdot \;\; (n_u n_v) D \log D(n_u + n_v) \\
&= C(D(|S| \cdot |T|) + D \log D(|S| \cdot |T|)) \\
&\leq 2C((|S| \cdot |T|) D \log D) \\
&= O((|S| \cdot |T|)(deg(S)) + deg(T)) \log(deg(S) + deg(T))
\end{aligned}$$

where $deg(S)$ is the maximum degree of a node in S and $D = deg(S) + deg(T)$.

7 Results

The authors in [18], aim to introduce an algorithm for the application in [15]. They used a pair of fluorescence based images, taken in two consecutive time frames, to identify the regions that corresponds to cells. The approach used in [18] is to convert each image into their corresponding component trees and then perform tree assignment to identify similar segments in the two images. The authors in [18], use dynamic programming algorithms for solving the constrained tree assignment problems while in [15], the authors have used the Integer Linear Programming (ILP) approach for the same task.
The authors have evaluated their dynamic programming algorithm for solving the constrained tree assignment against the ILP approach using the same synthetic and real images which were used in [15]. Real images display in-vivo time-lapse recordings of zebrafish brain with a green fluorescent protein expressed specific to microglia (neural immune cells); synthetic images contain ellipsoid objects perturbed by noise and different types of background inhomogeneity. Both synthetic and real images are three-dimensional involving around $200 \times 200 \times 40$ voxels each. The running times and the scores obtained by the constrained tree assignments was compared against the unconstrained tree assignment to evaluate whether the constraint affect solutions that are relevant in practice. As the quasi-linear time algorithm for computing overlap weights has been implemented, although not described, in [15], its running time was not further compared to a naive approach.

Running time of unconstrained versus constrained tree assignments: In the first experiment, the authors compared the running time of the integer linear programming approach from [15] with the dynamic programming approach for the constrained tree assignment as described in Section 4. As shown in Figure 6, their DP approach help them to solve substantially larger instances within a few seconds compared to the ILP approach. The running time of the ILP solver increases much more rapidly, especially when the number of vertices exceeds 2000.

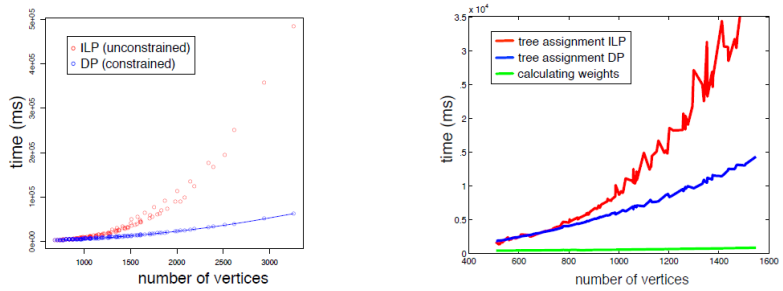


Abb. 1.6: *Running time of integer-linear programming approach vs. dynamic programming approach.* The number of vertices is the sum of the size of the two trees that were aligned. Instances of different size were derived from the microglia dataset from [15] by applying different pruning parameters for the component trees, again following [15]. The running time for computing weights becomes negligibly small using the quasi-linear time algorithm introduced in [18].

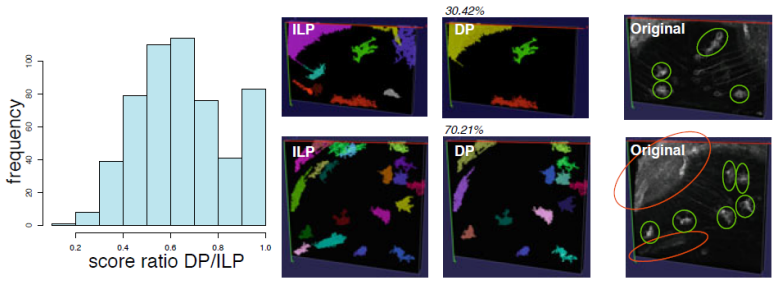


Abb. 1.7: *Left Part.* Histogram of score ratios between dynamic programming approach and integer linear programming approach. The majority of instances achieve a score of more than 60% of the unconstrained version and can be expected to be reliable. *Right Part. (Top.)* Example of a dataset where the constrained version (middle) achieves a score of only 30.42% of the unconstrained version (left), thus missing relevant segments representing microglia indicated by green circles in the original image (right). (Bottom.) Example of instance where the constrained version achieves 70.21% of the unconstrained version's score. All microglia are identified equally by both versions (green circles on the right), while the segments missed represent segments from background noise or unspecific expression of the fuorescent marker (red circles). (Visualized using v3d [12].)

Scores of unconstrained versus constrained tree assignments: To quantify how the three-point condition affects the score of solutions (and thus the quality results), in [18], the authors compared the scores obtained using Algorithm 2 versus the scores obtained from the unconstrained tree assignment using the integer linear programming approach from [15], utilizing both synthetic data and real microscopic images of zebrafish brain from [15]. The results displayed in Fig.7 indicate that while a small number of constrained scores achieve less than 30% of the unconstrained ones and can be considered to possibly loose critical segments, the majority of instances achieves a score of around 60% of the unconstrained score. Therefore, most of the results obtained using the constrained version are useful in practice. The score ratio is virtually constant across different signal-to-noise ratios in synthetic images with homogeneous background noise (data not shown).

8 Conclusion and Future Work

[18] introduces dynamic programming approaches to the tree assignment problem, which is of importance in bioimaging applications. From a theoretical point of view, constraining assignments to the three-point condition helps them to design a fast polynomial-time algorithm. The authors have evaluated the practical implications of this, demonstrating that the dynamic programming approach enables them to solve large instances within a few seconds. Comparing the resulting scores with the unconstrained version suggests that solving the constrained version is sufficient in many cases. This dynamic programming algorithm will be made available in a future release of the ct3d software package for cell tracking and cosegmentation

applications.
From an algorithmic point of view, investigating other variations of tree assignments is interesting from both theoretical and practical perspectives. The more relaxed condition introduced by Lu et al [8]. for the less-constrained edit distance [8] can also be applied to tree assignment in future. In that case, the dynamic programming algorithm from Jiang et al. [5] can be adapted, for alignment of unordered rooted trees, to solve the less-constrained tree assignment problem. The drawback is that the algorithm has running time exponential in the degree of the trees. The trade-off between the running time and the quality of the solution can also be explored. While beyond the scope of this more algorithmically focused contribution, a more detailed evaluation of how restricting to the three-point condition affects results on both synthetic and real biological image data is desirable.

Literaturverzeichnis

[1] Bille, Philip. *A survey on tree edit distance and related problems.* Theor. Comput. Sci., Volume 337, issue 1-3, June, 2005.

[2] Canzar, Stefan and Elbassioni, Khaled and Klau, Gunnar W. and Mestre, Julín. *On tree-constrained matchings and generalizations.* Proceedings of the 38th international colloquium conference on Automata, languages and programming - Volume Part I, series ICALP'11, pages 98-109, 2011.

[3] Fredman, Michael L. and Tarjan, Robert Endre. *Fibonacci heaps and their uses in improved network optimization algorithms.* J. ACM, volume 34, issue 3, pages 596-615 July, 1987.

[4] Tao Jiang, Guohui Lin, Bin Ma and Kaizhong Zhang. *A General Edit Distance between RNA Structures.* Journal of Computational Biology. pages 371-388, volume 9, number 2, 2002.

[5] Tao Jiang and Lusheng Wang and Kaizhong Zhang. *Alignment of trees - an alternative to tree edit.* Theoretical Computer Science. volume 143, number 1, pages 137-148. 1995.

[6] R. Jones. *Component trees for image filtering and segmentation.* Proceedings of the 1997 IEEE Workshop on Nonlinear Signal and Image Processing, Mackinac Island, September, 1997.

[7] Leong, F. J. W-M and Brady, M. and McGee, J. O'd. *Correction of uneven illumination (vignetting) in digital microscopy images* Journal of Clinical Pathology, number 8, pages 619-621, volume 56, August, 2003.

[8] Lu, Chin and Su, Zheng-Yao and Tang, Chuan. *A New Measure of Edit Distance between Labeled Trees.* National Center for High-Performance Computing 19-136 Hsinchu Taiwan 300 R.O.C., Computing and Combinatorics, Lecture Notes in Computer Science, pages pages 338-348, volume 2108, 2001.

[9] Mattes, Julian and Richard, Mathieu and Demongeot, Jacques. *Tree Representation for Image Matching and Object Recognition.* Proceedings of the 8th International Conference on Discrete Geometry for Computer Imagery, pages 298-312, series DCGI '99, 1999.

[10] Mosig, Axel and Jager, Stefan and Wang, Chaofeng and Nath, Sumit and Ersoy, Ilker and Palaniappan, Kannappan and Chen, Su-Shing. *Tracking cells in Life Cell Imaging videos using topological alignments.* Algorithms for molecular biology : AMB, volume 4, July, 2009.

[11] Najman, L. and Couprie, M. *Building the Component Tree in Quasi-Linear Time.* Image Processing, IEEE Transactions on, volume 15, number 11, pages 3531-3539, Nov. 2009.

[12] Li, Anan and Gong, Hui and Zhang, Bin and Wang, Qingdi and Yan, Cheng and Wu, Jingpeng and Liu, Qian and Zeng, Shaoqun and Luo, Qingming. *Micro-Optical Sectioning Tomography to Obtain a High-Resolution Atlas of the Mouse Brain.* volume 330, number 6009, pages 1404-1408, 2010.

[13] Bruce A. Shapiro and Kaizhong Zhang. *Comparing multiple RNA secondary structures using tree comparisons.* Computer Applications in the Biosciences. pages 309-318, 1990.

[14] Sternberg, S. R. *Biomedical Image Processing*, Computer, volume 16, issue 1, pages 22-34, January, 1983.

[15] Xiao, Hang and Li, Ying and Du, Jiulin and Mosig, Axel, *Ct3d: tracking microglia motility in 3D using a novel cosegmentation approach.* volume 27, number 4, pages 564-571, 2011.

[16] Zhang, Kaizhong. *A constrained edit distance between unordered labeled trees.* Algorithmica, pages 205-222, volume 15, issue 3, 1996.

[17] Rother, Carsten and Minka, Tom and Blake, Andrew and Kolmogorov, Vladimir. *Cosegmentation of Image Pairs by Histogram Matching - Incorporating a Global Constraint into MRFs.* Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1, pages 993-1000, 2006.

[18] Xiao, Hang and Zhang, Melvin and Mosig, Axel and Leong, Hon Wai. *Dynamic programming algorithms for efficiently computing cosegmentations between biological images* Proceedings of the 11th international conference on Algorithms in bioinformatics, series WABIÍ1, pages 339-350, 2011.

Object Detection by Shape Matching

Ludger Steens

Inhalt

Zusammenfassung

This report compares two recent publications that deal with the problem of object detection by shape matching: "From Partial Shape Matching to Robust Global Shape Similarity" by Ma et al. and "Global Contour Shape Matching" by Schindler et al. Both approaches are explained in detail and their advantages and disadvantages are discussed.

Keywords: shape matching, object detection, medical image processing

1 Introduction

Object detection in computer vision and medical image processing is the task of finding the objects in an image that belong to a specific class. Even though this is an easy problem for humans, for computers it is generally very hard. An algorithm has to decide if an image contains an object based on some template or model of this object. If the image contains the object the algorithm needs to find the position and dimension of the object in the image. The problems that may arise include noise or clutter in the image, objects that are only partly visible due to occlusion, and objects that are hard to separate from the background. These problems can either lead to not finding the object or detecting an object in a region that actually does not contain it (a false positive). The model or template that specifies the class of objects that should be found can either be a set of example images or some description of the features of the model. The features can include various information such as color, texture or geometric information.

This report focuses on object detection by shape matching. Shape matching uses only geometric information to compare objects. In the case of images these are the edges of an object. Objects can be detected by comparing the template object with the geometric information found in the image. In chapter 2 the basic concept of shape matching is explained in detail and some applications of shape matching and object detection in the context of medical image processing are presented. Chapter 3 and 4 present two recent publications that use shape matching for object detection: "From Partial Shape Matching through Local Deformation to Robust Global Shape Similarity for Object Detection" [1] and "Object Detection by Global Contour Shape" [2]. The last chapter compares the two approaches by explaining some of its strength and weaknesses.

2 Motivation and Fundamentals

2.1 The Basic Workflow of Shape Matching

There exist a lot of shape matching algorithms, each with a different approach to solve the problem of finding objects by their shape. Although they are quite different in detail, most of them share a common workflow. The first thing that needs to be defined is the template shape that specifies the class of objects that should be found. Template shapes can, for example, be learned though a set of sample images or they are given by the user in the form of a binary image. In either way they are stored in the form of a shape descriptor. Shape descriptors are data structures that model the most important features of a shape. Compared to a simple set of points (i.e. pixels) they have the advantage that they are usually independent from a specific image resolution. They should in particular be scale, translation and rotation invariant. This makes it easier to compare them with shapes found in an image.

Having defined a template shape, it is used to find similar shapes in a set of input images. First for each of these images the edges are extracted. Since only the shape of the objects is important other information like the texture of the object is not needed. Most edge detectors apply some filter to smooth the image and then compute the first or second order derivative of the resulting image. Due to occlusion or other artifacts, some parts of an edge may be missing. Several edge detection algorithms are capable of linking edges to fill small gaps caused by these artifacts. The quality of the edge extraction directly influences the result of the object detection. If some edges of the object are missing or the edges contain too much noise it is harder to find the object.

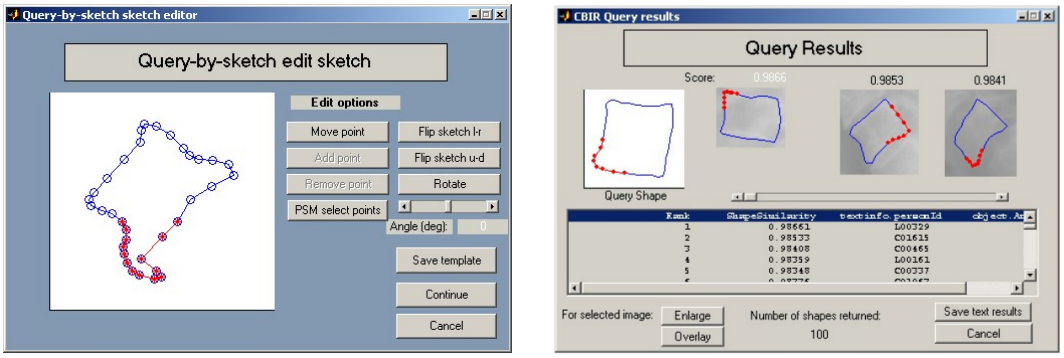
The next step in shape matching is to measure the similarity between the shape descriptor of the template shape and a set of edges from the input image. Therefore the edges are often stored in the form of the shape descriptor and a matching function is defined that measures the similarity to the template shape. The result of the matching function determines if the edges represent the object being sought. It is mostly inefficient to compare the template shape to all possible set of edges. Some kind of heuristic is used to find the potentially best sets of edges.

2.2 Medical Applications of Shape Matching

The object detection and shape matching mechanism described in the last section is most useful when searching for objects in large image collections. In medical applications there are often content-based image retrieval (CBIR) systems or computer-aided diagnosis (CAD) systems that contain lots of medical images. CBIR systems are used to retrieve images form a database based on some search parameters. They are for example used to find images that containing a specific pathology to help doctors in diagnosing this pathology. In addition to that they can be used for medical research or training medical students. CAD systems even go one step further by automatically diagnosing certain pathologies. They also work with large databases that contain images of various pathologies. The challenging task with these systems is to find exactly the images that the user wants to find. Indexing all images with keywords is time consuming and requires medical experts. Usually specific features of the images are automatically extracted and these features are used to compare images. Features can include information about color, texture, intensity or shape. However, sometimes shape is the only useful feature.

A use case where only shape information is used for automatic image retrieval can be found in [3] and [4]. The authors describe CBIR system for a large database of spine X-ray images. The National Institutes of Health of the United States maintains a collection of 17000 spine X-ray images that have been collected during a national survey. The problem with X-ray images is that they are grey value, have low contrast and contain little in terms of texture. The only useful information that can be extracted is the shape of the vertebrae. However, shape reliably describes various pathologies. For all images in the database the shape of the vertebrae was automatically extracted and stored together with the images in the database. The authors developed a tool where the user can sketch the shape of the vertebra and the system automatically searches for images that contain vertebrae with similar shapes. Figure 2.1 shows the user interface of the system.

In most cases only parts of the vertebra boundary are relevant for a specific pathology. One example of such a pathology that is mentioned by the authors is anterior osteophytes (AO) which are bone spurs that develop on the front of the vertebrae. They are usually caused by shrinking of the intervertebral discs and the resulting bone-on-bone friction of the vertebrae. To reliably find vertebra shapes that contain AO, the search should be restricted to the parts of the shape that represent the characteristic properties of AO. Therefore the authors developed a partial matching system where the user can select the part of the query shape that should be used during the search. In figure 2.1 this part is indicated by red dots.



(a) (b)

Abb. 2.1: (a) The user interface to sketch a query shape and to select the region of interest. (b) The user interface to show the shapes that have been found. (from [3])

3 From Partial Shape Matching to Robust Global Shape Similarity

The approach to object detection described by Ma et al. [1] is based on partial shape matching which gives the basis for global object detection. Partial shape matching means that only parts of the template contour are matched with edges in the image. For each edge in the image the algorithm tries to find a part on the template contour that is similar to the edge. The alignment of the edges found in this step

is compared to the alignment of the corresponding parts of the template object to find global detection hypothesis. An object is detected if the correspondence of the edges and the alignment is similar enough to the template object.

As input the algorithm expects a set of edges extracted from an image (section 3.1). The shape descriptor that is used to model the template object and the edges is based on the direction and distance between the points of the described shape. This is explained in detail in section 3.2. The matching function used for partial matching is presented in section 3.3. From the partial matches a weighted graph is constructed where certain subgraphs define the detection hypothesis (section 3.4). In this step the alignment of the partial matches is analyzed. In the last step these detection hypothesis are evaluated and ranked.

3.1 Edge Detection

Abb. 2.2: An image (a) with the output of an edge detector (b). All pixels that have been linked are shown in the same color. (from [1] with slight modifications)

The output of a standard edge detector is used as the basis for object detection. Figure 2.2 shows an example of an image together with the output of the edge detection process. To understand the problems and difficulties of edge detection the steps of the Canny edge detector [6] are roughly described in the following. The Canny edge detector is widely-used and in addition to that it is mentioned by Ma et al. as one possible edge extraction operator for their algorithm. First the input image is transformed into a grey value image. To reduce noise in the grey value image it is smoothed with a Gaussian filter. Edges are characterized by large changes of intensity between neighboring pixels. These changes are extracted by computing the first discrete derivative of the image in x- and y-direction. From these two derivatives the direction and magnitude of the gradient can be computed. Potential edges that are more than one pixel wide are then thinned to one pixel by setting all gradient magnitudes to zero (i.e. black) that do not represent a local maximum. The result of this step is again a grey value image where the intensity represents the magnitude of the gradient. The last step of the Canny edge detector is to decide whether a pixel with an intensity greater than zero really belongs to an edge. Starting with pixels that have a high intensity, the gradient direction computed earlier is used to trace edges and remove pixels that fall below a threshold. This results in a binary image where each pixel either represents an edge or no edge. One additional step that can be performed is linking edge pixels by assigning a unique value to all edge pixels that belong to one edge. The overall output of the edge detection process is thus a set of edges where each edge consists of a list of pixels that belong to this edge.

The Canny edge detector has a number of parameters that influence the result. The Gauss filter removes noise. But setting the size of the filter too high can have the result that small sharp objects are not detected. Setting the threshold used in the last step too high may result in not detecting some important edges. Setting it too low will detect small irrelevant edges.

The authors of [1] identified three challenges for object detection that are caused by suboptimal edge detection:

1. Edges or parts of edges that belong to the object may be missing in the image. This can, for example, happen if only a part of the object is visible or the edge detection algorithm was not able to detect these edges. For example in figure 2.2(b) the lower parts of the legs are missing.

2. If the edge extracted by the edge detector contains holes the edge linking algorithm may not be able to link all edge pixels to one edge. The result is that one edge of the template object is broken into several pieces in the image.

3. Some edges of the object may be wrongly connected to edges of the background. In figure 2.2(b) the yellow edge that belongs to the swan's neck contains also the reflection of the neck in the water.

The approach in [1] is designed to handle the three difficulties. The simple approach to match the whole template object with a whole edge in the image does not work if one of the problems occurs. To overcome the difficulties 1. and 2., the algorithm allows that only parts of the template object must match with an edge in the image. Problem number 3 makes the situation even worse. It implies that only part of an edge may belong to the object. Thus only part of an edge must match with part of the template object contour.

3.2 Shape Descriptor

The shape descriptor that is used is based on the ideas of "shape context" [5] which is one of the most popular descriptors to model the shape of an object. To describe a shape, shape context only needs a set of points sampled from the contour of the object. This makes the descriptor very flexible, because it can handle arbitrary edges and not only the outline of an object. For each sample point, the vectors to all other points are computed. The length and angles of these vectors can be viewed as the shape of the object relative to their originating point. They define the direction and the distance to all other points. Both quantities are placed inside histogram bins to reduce the amount of data. The histograms of all points together describe the shape of the object. To achieve invariance to scaling all length values are divided by the median distance of all point pairs before they are placed in the histogram. Furthermore, all distances are measured in logarithmic space to make the descriptor more sensitive to the position of nearby points. The descriptor is by construction invariant to translation because there are no absolute positions involved.

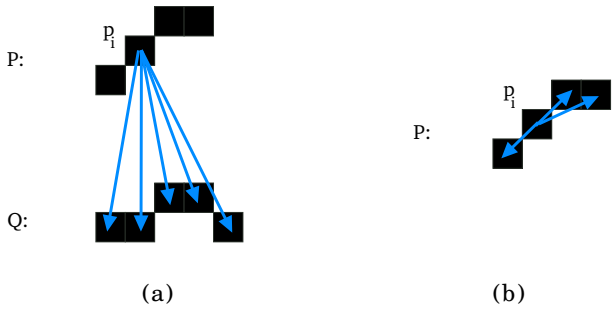


Abb. 2.3: (a) Illustration of the shape descriptor for two contour fragments. For each point p_i of P the distance and direction to all points of Q are computed. Likewise for each point of Q the distance and direction to all points of P are computed. (b) Illustration of the shape descriptor of one fragment. For each point p_i of P the distance and direction to all points of P are computed

The main difference between shape context and the descriptor used in [1] is that no histograms are computed. Instead all length and direction values are stored in matrices. In addition to that no points are sampled from the contour. Distance and direction is computed either between all points of one contour fragment (i.e. edge) or between the points of two contour fragments (see figure 2.3). The shape descriptor for one contour fragment is used in the partial matching step and the shape descriptor for two fragments is used in the global matching step. Given two contour fragments $P = \{p_1, ..., p_m\}$ and $Q = \{q_i, ..., q_n\}$, the matrix $D^{(P,Q)} \in \mathbb{R}^{mxn}$ stores all distances between points in P to points in Q and the matrix $\Theta^{(P,Q)} \in \mathbb{R}^{mxn}$ stores all directions from points in P to points in Q. So for two points p_i and q_j, $D^{(P,Q)}(i,j)$ contains the distance between the two points and $\Theta^{(P,Q)}(i,j)$ contains the orientation of the vector from p_i to q_j. The distance is again defined in the logarithmic space to make the descriptor more sensitive for near points:

$$D^{(P,Q)}(i,j) = \log\left(1 + \|\vec{p_i} - \vec{q_j}\|_2\right) \tag{2.1}$$

One is added to the euclidean distance to ensure that the matrix only contains positive values. To describe just one contour fragment, the definition of both matrices stays the same. $D^{(P,P)}$ contains the distance and $\Theta^{(P,P)}$ the directions of all points in P to all other points in P. The shape descriptor is used to describe both, the template contour and the edges in the input image. The observation made above for translation invariance regarding shape context are also valid for this descriptor. Scale invariance is achieved during comparison of shape descriptors (see section 3.3.1). However, the descriptor is not rotation invariant. The directions are defined in an arbitrary but fixed coordinate system and if the shape is rotated the descriptor changes. As later can be seen rotation is also not considered when comparing descriptors. This can be seen as a drawback of the approach presented in this chapter.

3.3 Matching Function

3.3.1 Similarity of two Shape Descriptors To measure the similarity of two descriptors, two affinity matrices are defined: one for the distances and one for the directions. The affinity matrix $A_D(P,Q,T,U)$ is used to compare the distance values of $D^{(P,Q)}$ with the values of $D^{(T,U)}$. Thus it models the similarity in the distance values of the contour fragment configuration (P,Q) to the contour fragment configuration (T,U). To compare the two contour fragment configurations, P must have the same number of points as T and Q must have the same number of points as U. The definition of the distance affinity matrix is the following:

$$A_D^{(P,Q,T,U)}(i,j) = \exp\left(-\frac{\left(D^{(P,Q)}(i,j) - D^{(T,U)}(i,j)\right)^2}{\left(D^{(P,Q)}(i,j)\sigma\right)^2}\right) \tag{2.2}$$

By dividing the distance difference by the distance of the first descriptor, the relative instead of the absolute difference is computed. This makes the comparison scale invariant. σ is used to control the tolerance of the distance comparison. A larger σ results in a larger affinity. Obviously a large distance leads to a small value in the affinity matrix and the other way around.

The affinity matrix for the directions is defined in a similar way as the one for the distances:

$$A_\Theta^{(P,Q,T,U)}(i,j) = \exp\left(-\frac{\left(\Theta^{(P,Q)}(i,j) - \Theta^{(T,U)}(i,j)\right)^2}{\delta^2}\right) \tag{2.3}$$

Here δ controls the tolerance of angle differences. The affinity matrix for the overall similarity is the sum of the distance and direction affinity matrices:

$$A^{(P,Q,T,U)} = A_D^{(P,Q,T,U)} + A_\Theta^{(P,Q,T,U)} \tag{2.4}$$

The similarity $\Psi(P,Q,T,U)$ between the two contour fragment configurations is now defined as the average value of the elements in the affinity matrix. All values in $A^{(P,Q,T,U)}$ are summed up and divided by the total number of values. If only two fragments are compared, the definitions from above can be used as well. For the two fragments P and T the affinity matrix is given by $A^{(P,P,T,T)}$ and the similarity is denoted by $\Psi(P,T)$.

3.3.2 Matching Edge Fragments To Template Contour Fragments With the affinity matrix defined above edges in the image can be compared with the template contour. However, as stated in section 3.1 only parts of the template contour must match with parts of edges. So for an edge the part that matches best with a part of a model fragment has to be found. Given an edge e_k which consists of the points $\{q_1,...,q_n\}$ and the template contour $\mathcal{M}$ with the points $\{p_1,...,p_m\}$, the question is at what point does the best match start and how long is it. One part of the edge is denoted by $e_k(j,l) = \{q_j,..,q_{j+l-1}\}$ and similarly one part of the template is denoted by $\mathcal{M}(i,l) = \{p_i,..,p_{i+l-1}\}$ (see figure 2.4). Obviously both parts need to have the same length in order to compare them. To find the best match, the values of the variables i,j and l need to be chosen in a way that maximizes the affinity of $\mathcal{M}(i,l)$ with $e_k(j,l)$. In addition to that the template shape needs to be compared to more than one edge and not just to a fixed edge e_k. To formalize the problem, the four dimensional matrix Γ is constructed:

$$\Gamma(i,j,l,k) = \Psi(\mathcal{M}(i,l), e_k(j,l)) \tag{2.5}$$

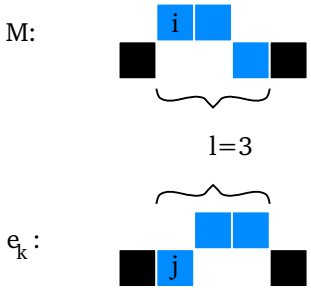


Abb. 2.4: The parameters that define a partial match between the template contour $\mathcal{M}$ and and edge e_k. i is the starting point on the template shape and j the starting point on the edge. l defines the length of the parts that should be compared.

Remember that Ψ uses the affinity matrices to measure the similarity of two fragments and thus Γ returns the similarity of a part of the template contour to a part of the edge k. Good matches between the template and the edges are given by finding maxima of Γ. The first observation that one can make is that for given starting points i and j and a specific edge k, only the length l is relevant that gives the best matching. It is unlikely (however not impossible) that a length l that does not belong to the best partial matching leads to a better global matching. Using only the best length reduces the dimension of the matrix and is thus a first optimization step. Based on this observation two functions are defined:

$$S(i,j,k) = \max_l \Gamma(i,j,l,k) \tag{2.6}$$

$$G(i,j,k) = \arg\max_l \Gamma(i,j,l,k) \tag{2.7}$$

Given i, j and k, the function G returns the length l that maximizes the similarity and S computes this similarity value. One additional improvement must be made regarding the definition of the matching. If the length is small only a few points are compared which gives no reliable information about the shape similarity. For example if $l = 1$ only one point is compared which always leads to the maximal similarity. Therefore $\Gamma(i,j,l,k)$ is set to zero if l is smaller than a specific threshold.

3.4 Finding the Best Matching Contours in the Image

In the last section parts of the template contour were matched with the edges in the image. To detect objects, this partial matching must be transformed to a global one. Based on the local matching, edge fragments need to be found whose alignment is similar to the alignment of the corresponding template fragments. Given two partial matches $(e_m, \mathcal{M}_1)$ and $(e_n, \mathcal{M}_2)$ between edges (or parts of edges) and template fragments, it has to be checked if the configuration of e_m and e_n is compatible to the configuration of $\mathcal{M}_1$ and $\mathcal{M}_2$ (see figure 2.5). Here configuration means the direction and relative distance between the fragments. If enough edges are found where the overall configuration is similar to the template object, an object is detected.

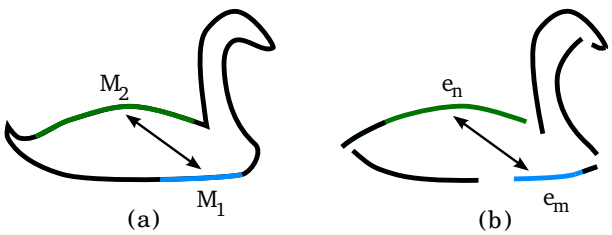


Abb. 2.5: To find a global matching, the configuration of partial matches must be compared within the template object (a) and between the edges of the image (b).

3.4.1 Transformation to a Graph Problem The task of finding good global matches is transformed to a graph problem. A weighted graph, the matching graph, is constructed, where each vertex corresponds to a partial match $G(i, j, k)$. The graph is complete, i.e. each vertex is connected with all other vertices. The edge weight between two vertices is defined as the similarity between the configuration that is created by the two partial matches.

Given two vertices $v_1 = G(i_1, j_1, m)$ and $v_2 = G(i_2, j_2, n)$ which correspond to the partial matchings $(\mathcal{M}(i_1, l_1),\ e_m(j_i, l_1))$ respectively $(\mathcal{M}(i_2, l_2),\ e_n(j_2, l_2))$, the edge weight between the vertices is defined as:

$$A(1,2) = \Psi\left(\mathcal{M}(i_1, l_1),\ \mathcal{M}(i_2, l_2),\ e_m(j_i, l_1),\ e_n(j_2, l_2)\right) \tag{2.8}$$

The edge weight is equal to the similarity of the configuration $(\mathcal{M}(i_1, l_1),\ \mathcal{M}(i_2, l_2))$ to the configuration $(e_m(j_i, l_1),\ e_n(j_2, l_2))$. A is actually a matrix where each element specifies the weight of the edge between two vertices. The graph can now be used to find detection hypothesis by finding clusters in the graph. A cluster is a subgraph where the vertices have high internal affinity and low affinity to vertices outside of the subgraph [13]. By adding additional vertices to the cluster, the average affinity is decreased. A cluster can thus be seen as a maximal clique in a weighted graph. The precise definition is a little bit complicated and can be found in [13]. In the case of the matching graph, a cluster is a subgraph where the similarity (i.e. edge weights) between the partial matches (i.e. vertices) is maximal.

To reduce the complexity of the graph, a few adjustments are made. For each point i of the template shape only the K best matches are included as vertices in the graph. This reduces the number of vertices in the graph but also limits the number of objects that can be detected per image. If there is more than one match for an edge, the distance between the edge parts must be approximately the same as the distance between the two corresponding parts of the template object. If the distance difference is too large, the edge weight between the two matches is set to zero. This sparsifies the edge weight matrix A. In addition to the two simplifications from above, the complexity of the graph can further be reduced by observing that a part of the template fragment can only correspond to one edge fragment. It is impossible that several edges in the image correspond to the same part of the template contour within one detection hypotheses. If two local matchings overlap too much on the template shape, the edge weight between the two corresponding vertices is set to zero. This way both vertices cannot be part of the same detection hypothesis.

To find clusters in the graph, the following optimization problem has to be solved:

$$\begin{aligned} &\text{maximize} \quad f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x} \\ &\text{subject to} \quad x \in \left\{\mathbf{x} \in \mathbb{R}^{M \cdot K} : \mathbf{x} \geq 0 \text{ and } \|\mathbf{x}\|_1 = 1\right\} \end{aligned}$$

Here, M denotes the number of points of the template shape and A is the matrix containing the edge weights. A cluster is defined by a vector $\mathbf{x}$ that maximizes the above equation. All vertices v with > 0 belong to the cluster with $\mathbf{x}_v$ being the v-th element of $\mathbf{x}$. Details about the procedure and a proof that the solution of the optimization problem defines clusters can be found in [13].

3.4.2 Evaluation of Detection Hypothesis The detection hypothesis found with the procedure above may vary in their quality. Not all may actually be similar enough to the template model and some parts of the template shape may be missing in the partial matches. To evaluate and score a detection hypothesis, the template object is transformed into the image based on the transformation that is induced by the partial matches of the detection hypothesis. Each detection hypothesis consists of a set of partial matches that establish a correspondence between points of the template contour and image edge points. These correspondences define an affine transformation between template contour points and image points. However, not all points in the template contour belong to a match and thus have no corresponding image points. The subset of template points that have a matching with an edge point are denoted $\mathcal{M}_a$. All other template points are denoted $\mathcal{M}_b$. The transformation that is defined by mapping the points in M_a to their corresponding edge points is denoted $T(x)$. The task is to extend $T(x)$ to include the points in M_b.

The general idea is to estimate the transformation of a point $x \in \mathcal{M}_b$ by close points $N(x)$ that are in $\mathcal{M}_a$. Close means no spacial distance but the connection of points. The reason for this definition is that points that are connected are likely to have the same affine transformation. The transformation for x is chosen in a way that the accumulated square distance to the transformed points of $N(x)$ is minimized.

Based on this transformation each detection hypothesis is assigned a score and all hypothesis are ranked. The score or confidence of a hypothesis is build from two aspects:

$$S(T(\mathcal{M})) = \Psi(\mathcal{M}, T(\mathcal{M})) \cdot \Psi(T(\mathcal{M}), T'(\mathcal{M})) \tag{2.9}$$

The first part measures how well the original template shape corresponds to the transformed one. The second part measures if the transformed image corresponds with the edges in the image. The points in $\mathcal{M}_a$ all have corresponding edge points E_a. For all points in $\mathcal{M}_b$ it is tried to find corresponding edge points based on $T(x)$. Therefore the tangent direction is computed for all edge points and for all points in $T(x)$ with $x \in \mathcal{M}_b$. Based on the position and the tangent direction for each transformed point $x \in \mathcal{M}_b$ the closest edge point is found. $T'(\mathcal{M})$ contains all these closest edge points together with the points in E_a.

3.5 Experimental Results

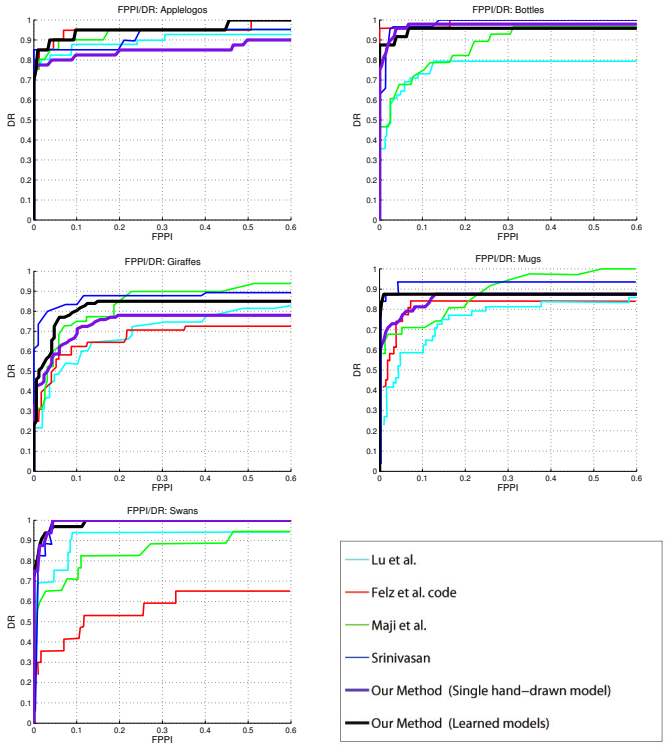


Abb. 2.6: Comparision of detection results for the ETHZ data set (from [1]).

Ma et al. compared their approach to object detection with four other approaches: Lu et al. [10] from 2009, Felzenszwalb et al. [8] from 2008, Maji et al. [11] from 2009 and Srinivasan [14] from 2010. As test data, the ETHZ data set [9] is used which consists of 255 images that contain five different classes of objects: swans, giraffes, bottles, mugs and the Apple logo. The template objects for the algorithm of Ma et al. were specified in two different ways. First a hand drawn image was used for all five object classes. In a second run for each class the first half of the images that contain an object of the class is used for training. The outlines of the objects in the images were used to obtain several template shapes for each class. The second half of the images is then used to test the object detection algorithm. Figure 2.6 and table 1 show the result of the comparison. Results are given as ratio of detection rate to false positive per image (FPPI). The detection rate specifies the relative number of objects that were detected. To decide if an object is detected correctly, the bounding boxes of the object detected by the algorithm and the bounding box of the object in the image are compared. If the intersection of both bounding boxes over the union of the bounding boxes is greater than 50%, a detection is considered as correct. A false positive occurs when an algorithm detects an object at a position were actually no object is present. The FPPI is

	0.3 FPPI		0.4 FPPI	
	Swan	Apple Logo	Swan	Apple Logo
Ma et al. [1]	100%	92%	100%	92%
Srinivasan et al. [14]	100%	95%	100%	95%
Maji et al. [11]	88.2%	95%	88.2%	95%
Felzenszwalb et al. [8]	58.8%	95%	64.7%	95%
Lu et al. [10]	93.8%	90%	93.8%	90%

Tab. 1: The detection rate of some object classes for 0.3 and 0.4 false positives per image (data from [1]).

thus the average number of wrong detections per image. Usually object detection algorithms exhibit some parameters that influence the tolerance of shape comparison. In section 3.3.1 the two parameters σ and δ where used to control the tolerance of distance differences respectively the tolerance of tangent angel differences. Raising both values leads to more detected objects. However, the number of false positives rises as well.

The results indicate that the approach of Ma et al. performs fairly well compared to the other approaches. On some classes such as the mugs and giraffes other algorithms perform better whereas Ma et al.'s approach gives the best results for the swan object class. Template shapes that have been learned almost always outperform the hand drawn ones.

4 Global Contour Shape Matching

The publication "Object detection by Global Contour Shape" [2] presents a shape matching algorithm based only on the outline of objects. The template shape consists only of the silhouette of an object. No edges inside the object are considered and objects that have more than one closed contour cannot be handled directly. The template shape can thus be seen as a deformed circle.

The contour of the template shape is approximated by a closed polygon where the vertices are sampled from the contour. The shape descriptor is simply the sequence of tangent angles of this polygon. The details of the shape descriptor are presented in section 4.1. The algorithm works with segmented images in which multiple adjacent pixels are grouped together. These groups are called super-pixels and they are also described by the shape descriptor. Segmentation is explained in section 4.2. The similarity between the template shape and one or more super-pixels is computed by comparing the two shape descriptors. Two parameters influence the similarity: stretching of vertices and tangent angle difference. The details of the matching function which measures the similarity are given is section 4.3. The sets of super-pixel that best match the template shape are found by +using a greedy search algorithm (section 4.4). This chapter concludes with some experimental results of object detection in section 4.5.

4.1 Shape Descriptor

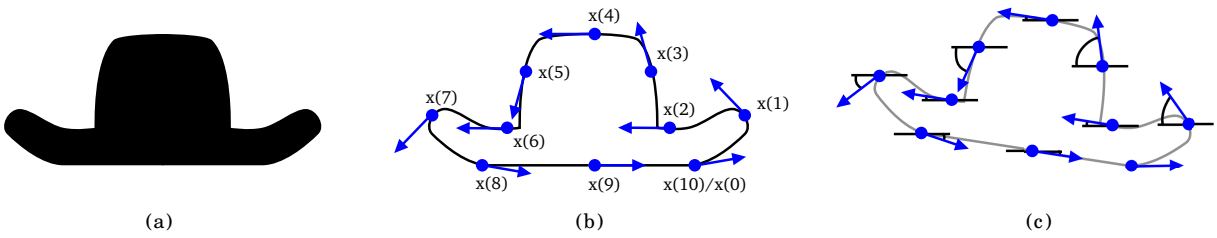


Abb. 2.7: The process of creating a shape descriptor for an object. (a) The shape of the object as a binary image. (b) The contour is approximated by 10 vertices together with their tangent vectors. (c) The tangent vector of the first vertex is aligned with the x-axis.

The shape descriptor must be able to describe the contour of an object. As noted above, edges inside the object are not considered. The idea is to represent an object by the tangent angles of a fixed number of points sampled from the contour.

Figure 2.7 illustrates the steps necessary to compute the shape descriptor. In figure 2.7(a) the object (a hat) is represented as a binary image where the contour is defined by the boundary pixels of the hat. The first step to compute the shape descriptor is to approximate the contour by a fixed number of vertices that are evenly distributed on the contour (figure 2.7(b)). Evenly distributed means that the distance on the contour curve between two vertices is always the same. The polygon defined by these vertices approximates the contour. If the number of vertices is denoted by N, the set of vertices X is defined by $X = \{x(u), u = 0...N-1\}$. To make the following definitions easier, an additional vertex $x(N)$ is defined which is the same as the first vertex: $x(N) = x(0)$. For each vertex $x(u)$ the tangent vector $\dot{x}(u)$ is computed. The tangent vector is used to calculate the tangent angle, which is the angle between the tangent and the x-axis. To make the shape descriptor independent from the rotation of the object in the image, the tangent vector of the first vertex is aligned with the x-axis (Figure 2.7(c)). This way, the tangent angle belonging to the first vertex is always zero. These normalized tangent angles are denoted by $x'(u)$ and the shape descriptor X' is the sequence of all normalized tangent angles: $X' = \{x'(u), u = 0...N-1\}$. To reduce the noise that the tangent angles may contain they are smoothed. For each tangent angle the average of the tangent angle itself and the tangent angels of the neighbouring points is computed.

The shape descriptor defined here is translation and scale invariant. It is translation invariant because it does not contain any absolute positions but the relative positions between the vertices. It is scale invariant because it uses a fixed number of vertices. The number of vertices is equal no matter if the object is large (i.e. contains many pixels) or small (i.e. contains few pixels) in the image. Later , when comparing two shape descriptors, the choice of the first vertex will change to make the comparison rotation invariant.

4.2 Edge Detection

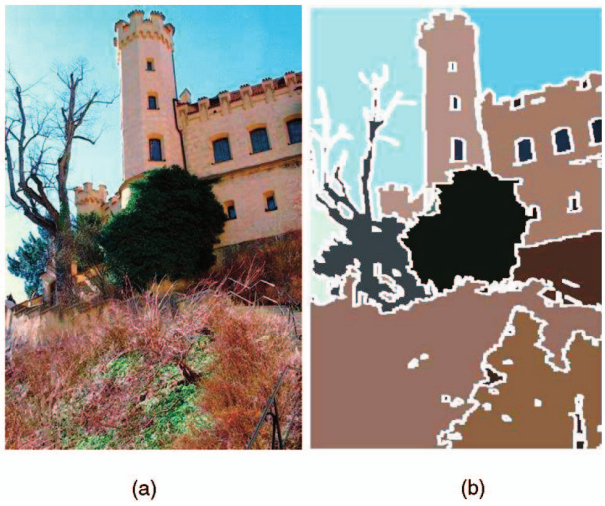


Abb. 2.8: An image (left) with a possible segmentation (right) [12]

To detect edges in the input image a segmentation algorithm is used instead of a “classical” edge detection algorithm such as the one described in section 3.1. Image segmentation groups neighboring pixels based on their similarity. The resulting groups are called segments or super-pixels. The similarity of pixels can depend on different factors such as color, intensity or texture. The goal of segmentation is to group pixels together that belong to the same object. The result is a set of super-pixels that together cover the whole image. Figure 2.8 shows an image and the result of segmentation. Each segment is separated by a white border.

Image segmentation has the property that every segment is defined by a closed contour. This makes it particularly useful for the shape matching algorithm presented in this chapter which works exclusively with closed contours. A segment or a set of adjacent segments can directly be described by the shape descriptor defined above. Moreover, image segmentation avoids the problem of spurious edges, wrongly connected edges and edges that are not connected by mistake. However, image segmentation has some problems on its own [12]. Under-segmentation describes the effect, that one segment covers more than

one object. An ideal segmentation would divide the segment into multiple smaller segments. The opposite of under-segmentation is over-segmentation which describes the effect that one object is covered by more than one segment. The third problem, a hybrid between under-segmentation and over-segmentation, is that one segment covers *parts* of multiple objects. The problem of over-segmentation is not a severe one for the object detection algorithm presented in this chapter. Later multiple segments are "merged" and the result is compared to the template shape (section 4.4).

The segmentation approach used in [2] is "Statistical Region Merging" presented in [12]. The segmentation algorithm is configured to rather produce over-segmentation than under-segmentation.

4.3 Matching Function

This section describes how two shape descriptors are compared. One shape descriptor defines the template object and the other one defines one or more segments of the input image. To compare how well the contour of one shape matches the contour of the other the authors of [2] use a technique called "nonlinear elastic matching distance with stretch cost R" (NEM_R). NEM_R matches two contours by associating the points of one contour with the points of the other. The similarity of the contours then depends on two factors or costs: a stretching factor that is influenced by the way the points are associated and a distance factor that measures the difference of two associated points. The best matching is the one that minimizes both factors and it defines how similar the two contours are.

The next sections explains the details of matching two contours. In section 4.3.2 the NEM_R is described in general and section 4.3.3 contains a description of the variant used in [2].

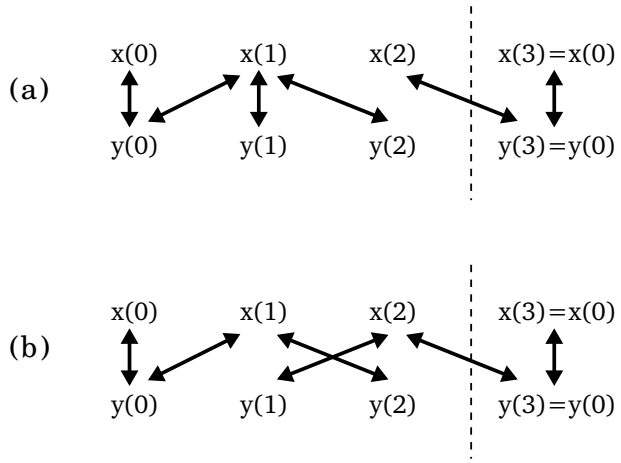


Abb. 2.9: Examples for the matching between two contours $X = x(u)$ and $Y = y(v)$ with three vertices each. (a) shows a valid matching whereas (b) is an invalid matching.

4.3.1 Matching of Contours To be able to measure the similarity of two shapes they are first matched by associating the vertices of each shape with vertices of the other shape. Given two sets of vertices $X = \{x(u)\}$ and $Y = \{y(v)\}$ with N vertices each, all vertices in X are matched to at least one vertex in Y. It is possible that a vertex from one shape is associated with more than one vertex of the other shape. However, it is important that the order of the vertices is maintained by the matching: There is no $i < j$ such that $x(i)$ matches with a vertex $y(i')$ and $x(j)$ matches with a vertex $y(j')$ with $i' > j'$ [7]. Figure 2.9 show two examples of associations. The matching defined in (a) is valid whereas the matching in (b) is not valid: If $x(1)$ is associated with $y(2)$ then $x(2)$ cannot be associated with $y(1)$. If a matching is visualized the way it is done in figure 2.9 there must not be any association arrows that cross.

A matching $\mathcal{V}$ between two contours X and Y is formally defined by:

$$\mathcal{V}(X,Y) = \{\langle u_i \Leftrightarrow v_i \rangle, i = 1...K_{\mathcal{V}}\} = \{\langle 0 \Leftrightarrow 0 \rangle, ..., \langle u_i \Leftrightarrow v_i \rangle, ..., \langle N \Leftrightarrow N \rangle\}. \tag{2.10}$$

$K_\mathcal{V}$ is the number of associations between points (excluding $\langle N \Leftrightarrow N \rangle$) and it holds that $N \leq K_\mathcal{V} \leq 2N$ if the matching is valid. The example in figure 2.9(a) can be formally defined by:

$$\begin{aligned}\mathcal{V}(X,Y) &= \{\langle u_1 \Leftrightarrow v_1 \rangle, \langle u_2 \Leftrightarrow v_2 \rangle, \langle u_3 \Leftrightarrow v_3 \rangle, \langle u_4 \Leftrightarrow v_4 \rangle, \langle u_5 \Leftrightarrow v_5 \rangle, \langle u_6 \Leftrightarrow v_6 \rangle\} \\ &= \{\langle 0 \Leftrightarrow 0 \rangle, \langle 1 \Leftrightarrow 0 \rangle, \langle 1 \Leftrightarrow 1 \rangle, \langle 1 \Leftrightarrow 2 \rangle, \langle 2 \Leftrightarrow 3 \rangle, \langle 3 \Leftrightarrow 3 \rangle\}.\end{aligned}$$

The above definition of $\mathcal{V}$ makes the assumption that the first vertex of X always matches with the first vertex of Y. This is no real restriction because later the numbering of the vertices is shifted (see section 4.3.4).

4.3.2 Nonlinear Elastic Matching Distance with Stretch Cost R The NEM_R measures the distance or dissimilarity of two contours based on two factors: a stretching cost and a distance cost [7]. Both cost functions are defined for a specific matching of the contours. The matching that minimizes the sum of both costs defines the best matching and thus the distance of the two contours.

The stretching cost function measures the local stretching that is created by the matching. If each vertex of a shape X is associated with exactly one vertex of a shape Y there is no stretching. Stretching occurs when one vertex of X is associated with more than one vertex of Y and vice versa. An association $\langle u_i \Leftrightarrow v_i \rangle$ is called a *stretch-association* if either $u_{i-1} = u_i \wedge v_{i-1} = v_i - 1$ or $u_{i-1} = u_i - 1 \wedge v_{i-1} = v_i$. Informally a strech-association is an association where one of the vertices is also associated to a vertex prior in the sequence. For example in figure 2.9(a) the association $\langle u_2 \Leftrightarrow v_2 \rangle = \langle 1 \Leftrightarrow 0 \rangle$ is a stretch-association because of the association $\langle u_1 \Leftrightarrow v_1 \rangle = \langle 0 \Leftrightarrow 0 \rangle$ ($u_{2-1} = u_2 - 1 = 0$ and $v_{2-1} = v_2 = 0$). In fact all associations but $\langle 2 \Leftrightarrow 3 \rangle$ are stretch-associations. The stretching cost function S assigns the cost R to all stretching-associations and zero to all other associations:

$$S(u_i, v_i) = \begin{cases} R & \text{if} \quad (u_{i-1} = u_i \wedge v_{i-1} = v_i - 1) \text{ or } (u_{i-1} = u_i - 1 \wedge v_{i-1} = v_i) \\ 0 & \text{else} \end{cases} \tag{2.11}$$

This definition assumes that only valid matches are performed. R defines how much stretching is penalized and should obviously be greater than zero.

The distance cost function D measures the dissimilarity of two associated points. With the shape descriptor defined above this would be the difference of the tangent angles. A simple choice whould be to take the absolute value of the difference between the angles:

$$D(u_i, v_i) = |x'(u_i) - y'(v_i)| \tag{2.12}$$

Other choices for D are to take the square or the cosine of the difference. In any way the distance is higher the more the tangent angles differ.

The cost for a matching $\mathcal{V}$ is the sum of both the stretching cost and the distance cost over all associations:

$$cost(\mathcal{V}) = \sum_{i=1}^{K_\mathcal{V}} S(u_i, v_i) + \sum_{i=1}^{K_\mathcal{V}} D(u_i, v_i) \tag{2.13}$$

Of all possible matchings only the one with minimum cost (i.e. the highest similarity) is relevant. Finally the definition of the NEM_R is given by the following optimization problem:

$$NEM_R(X,Y) = \min_{\mathcal{V}} \{cost(\mathcal{V})\} \tag{2.14}$$

The above definition is not one hundred percent correct because in the last section one association was fixed for a match $\mathcal{V}$. The first vertex of X always matches with the first vertex of Y. This issue can be solved by shifting the numbering of the vertices of one contour. The NEM_R is then defined as the minimum of all matches of all rotations. When the choice of the first vertex changes by rotation, the tangent angles need to be re-normalized so that the first tangent again aligns with the x-axis and the corresponding tangent angle is zero.

4.3.3 Stretching and Distance as Probabilities In [2] the cost of a matching is expressed by a probability function that specifies how likely it is that a certain matching is right. The problem of finding

the best matching is then solved by finding the matching with the highest probability. This problem can be transformed to a NEM_R equation.

Given a matching of two contours, the probability that an association between two points is right is again divided into the two components local stretching and distance: $P(u_i \Leftrightarrow v_i) = P_S(u_i \Leftrightarrow v_i)P_D(u_i \Leftrightarrow v_i)$. P_S and P_D are probability functions but they serve the same purpose as the corresponding functions in the last section: they measure local stretching respectively tangent angle difference. The probability function that accounts for stretching is defined by:

$$P_S(u_i \Leftrightarrow v_i) = \frac{1}{1+e^{-E}} e^{-S(u_i,v_i)} \tag{2.15}$$

The definition for $S(u_i, v_i)$ is the same as the one in equation 2.11 except that the constant for the stretching cost is called E instead of R. So E should be a positive number that defines how much stretching is penalized. P_S defines the probability for a matching based on the local stretching. If $\langle u_i \Leftrightarrow v_i \rangle$ is a stretching-association then $S(u_i, v_i) = E$ and thus the probability P_S is low. If $\langle u_i \Leftrightarrow v_i \rangle$ is not a stretching-association the probability P_S is high.

The distance part of the probability that a matching between two points is right is defined by:

$$P_D(u_i \Leftrightarrow v_i) = \frac{1}{H(B)} e^{-\frac{1}{B} \cdot D(u_i,v_i)} \tag{2.16}$$

The distance between points is again measured by the difference of the tangent angles and $D(u_i, v_i)$ can be defined as in equation 2.12. As stated in the last section there are multiple choices for D that change the type of distribution that P_D defines. For example:

- $D(u_i, v_i) = |x'(u_i) - y'(v_i)|$ creates a truncated Laplace distribution.
- $D(u_i, v_i) = (x'(u_i) - y'(v_i))^2$ creates a truncated Gaussian distribution.
- $D(u_i, v_i) = 1 - \cos(x'(u_i) - y'(v_i))$ creates a von Mises distribution.

B specifies the second moment of the distribution and $H(B)$ is a constant that ensures that P_S integrates to 1 and thus actually represents a probability distribution. Later it can be seen that B actually influences the penalty for stretching.

The probability that a whole matching is correct is given by the product over the probability of all point associations:

$$P(\mathcal{V}) = \prod_{i=1}^{K_\mathcal{V}} P_D(u_i \Leftrightarrow v_i) \prod_{i=1}^{K_\mathcal{V}} P_S(u_i \Leftrightarrow v_i) \tag{2.17}$$

What needs to be done, is to find the matching with the highest probability, i.e. the best matching. This can be done by computing the minimum of the function $C(\mathcal{V}) = -ln(P(\mathcal{V}))$. The logarithm is monotonically increasing and thus has no influence on the maximum of $P(\mathcal{V})$. However, taking the logarithm has the advantage that the products in $P(\mathcal{V})$ turn into sums and the exponential functions inside P_S and P_D are removed. By multiplying with minus one the maximization problem is turned to a minimization problem. After a few transformations of $C(\mathcal{V})$ the best matching is defined by:

$$\tilde{C}(X,Y) = \min_{\mathcal{V}} \left\{ \sum_{i=1}^{K_\mathcal{V}} D(u_i, v_i) + 2(K_\mathcal{V} - N) \cdot \frac{1}{2} B(Q + 2E) \right\} \tag{2.18}$$
$$\text{with} \quad Q = -\ln\left(H(B)(1 + e^{-E})\right)$$

This function has the form of a "nonlinear elastic matching distance with stretch cost R" with the stretch-cost $R = \frac{1}{2}B(Q+2E)$ (see equation 2.14). The reason that there is no summation over the stretching costs of all point matches is that there are exactly $2(K_\mathcal{V} - N)$ stretching-associations for a matching $\mathcal{V}$. Imagine a matching with N associations (i.e. $K_\mathcal{V} = N$). There are obviously no stretching-associations. Each association that is added to that matching creates exactly two stretching-associations. Non stretching-associations can be ignored because they have zero stretching cost.

```
Input: Shape descriptors {x'(u)} and {y'(v)}
Output: ShapeDistance

allocate D_{N×N}, R_{N×N} ;
// compute vertex dissimilarities
for i = 0 to N do
    for j = 0 to N do
        d_{i,j} ← D(u_i, v_i) ;
    end
end
// compute cumulative matching cost
r_{0,0} ← d_{0,0} ;
for i = 1 to N do
    r_{i,0} ← r_{i−1,0} + d_{i,0} + R ;
    r_{0,i} ← r_{0,i−1} + d_{0,i} + R ;
end
for i = 1 to N do
    for j = 1 to N do
        h_{10} ← r_{i−1,j} + R ;
        h_{01} ← r_{i,j−1} + R ;
        h_{11} ← r_{i−1,j−1} ;
        r_{i,j} ← min(h_{10}, h_{01}, h_{11}) + d_{i,j} ;
    end
end
ShapeDistance ← r_{N,N} ;
```

Abb. 2.10: Algorithm to compute the matching with the lowest cost via dynamic programming (from [2]).

4.3.4 Solving the Nonlinear Elastic Matching function To solve the optimization problem defined by the NEM_R equation mentioned above, the problem can be converted to a path-search problem on a directed graph. Dynamic programming is then used to solve the path-search problem. Figure 2.10 shows the algorithm that is used.

Each vertex of the directed graph corresponds to an association $\langle u_i \Leftrightarrow v_i \rangle$ of two points from the contours. Assuming that the first point of the template contour matches with the first point of the contour in the image, the task is to find the path with the lowest cumulative cost from $\langle 0 \Leftrightarrow 0 \rangle$ to $\langle N \Leftrightarrow N \rangle$. To calculate the cost of a path, there are two aspects to consider: the distance cost and the stretching cost. The distance cost, which measures the tangent angle difference, can be computed separately for each point association. It can thus be attached directly to a vertex of the graph. In figure 2.10 the dissimilarity matrix $\mathcal{D}$ contains the distances for all possible point associations.

To calculate the stretching cost from one vertex to another, there are three cases to consider. From a vertex $\langle i \Leftrightarrow j \rangle$ that matches the points i and j there are the following continuations possible:

1. Associate the points $i + 1$ and $j + 1$. This obviously causes no stretching.
2. Associate the points i and $j + 1$. This creates a stretching-association, because the second contour is locally stretched.
3. Associate the points $i + 1$ and j. This creates a stretching-association, because the first contour is locally stretched.

	0.2 FPPI			0.4 FPPI		
	Swan	Apple Logo	Average	Swan	Apple Logo	Average
NEM_R	94%	86 %	89%	97%	88%	92%
Ferrari et al. [9]	73%	52%	61%	94%	73%	82%
Chamfer	15%	26%	21%	24%	30%	28%

Tab. 2: The detection rate of some object classes for 0.2 and 0.4 false positives per image (data from [2]).

In the last two cases the stretching cost R needs to be added to the path. In figure 2.10 the variables h_{10}, h_{01} and h_{11} are used to calculate the cost for the three cases. The matrix $\mathcal{R}$ is used to iteratively compute the lowest cumulative cost for all paths from $\langle 0 \Leftrightarrow 0 \rangle$ to $\langle N \Leftrightarrow N \rangle$. The computation of $\mathcal{R}$ follows the standard approach of dynamic programming: $\mathcal{R}$ is computed from top to bottom and the element r_{ij} contains the lowest possible cost from the vertex $\langle 0 \Leftrightarrow 0 \rangle$ to the vertex $\langle i \Leftrightarrow j \rangle$ and at the end r_{NN} contains the lowest possible cost for the whole path. To get rid of the assumption that the first points of both contours must match, an additional loop has to be added to the algorithm, which rotates the indices of the points for one contour.

It can be observed that the algorithm has a time complexity of $O(N^2)$ because of the two nested loops that compute the cumulative matching cost matrix $\mathcal{R}$. If the loop for the rotation of the indices is added, the complexity is in $O(N^3)$. It is thus important that the number of sample points is not too high.

4.4 Finding the best matching Contours in the Segmented Image

In the last section a way to compute the similarity of two contours was presented. One contour belongs to the template shape and the other one has to be from the segmented image. Remember that a segment or super-pixel is defined by a closed contour. The template shape can thus be compared not only to one super-pixel but also to a group of neighboring super-pixels that have a closed contour together.

A simple approach to find matching contours in the image would be to compare the template shape to each element of the combinatorial set of super-pixels. Since the cardinality of this set rises rapidly with the number of super-pixels, this approach is not feasible in practice. In [2] a greedy method is used to find groups of super-pixels with a good matching. Figure 2.11 shows the algorithm that is used. Each super-pixel is successively chosen as the start point for the search and selected as an object candidate. Starting with one super-pixel, the group of super-pixels that represent an object candidate is iteratively increased as long as the distance to the template object can be minimized. In each step all neighboring super-pixels are merged in turn to the candidate object and the one that most decreases the distance to the template object is added to the group. If no neighboring super-pixel can be found that can be used to decrease the distance, the search stops. To compare contours, the NEM_R function is used, which is defined by the algorithm in figure 2.10. In a last step, all candidates are compared and if two candidates are found that overlap more than a specific threshold, the one with the greater distance is removed. This ensures that there are no two candidates that actually represent the same object.

The greedy approach has a number of advantages: it is fast and requires no parameter tuning since it does not depend on specific properties of the template object or the image. However, the big disadvantage is that it only finds local minima. If the distance to the template object can be reduced by adding two neighboring super-pixel simultaneously but not successively, the algorithm will not recognize this.

4.5 Experimental Results

Schindler et al. tested the object detection algorithm presented in this chapter with different sets of images and classes of objects. Table 2 lists the result achieved with the ETHZ data set [9]. NEM_R denotes the approach of Schindler et al. which is compared to Ferrari et al.'s approach [9] from 2006 and Chamfer matching, which is a fairly old method for object detection. The template object is defined by a hand drawn sketch for each class. A comparison to the results in section 3.5 indicate that the approach of Ma et al. performs better than the approach of Schindler et al. However, a direct comparison of experimental values is always difficult because slight changes in the test procedure can lead to different results. For example the quality of the hand drawn template object clearly influences the detection quality. Furthermore, Schindler et al. use a slightly different definition of object detection. Again the bounding box of the object detected by the algorithm and the bounding box of the object in the image are compared.

```
Input: object template, list of super-pixels, detection threshold, overlap threshold
Output: detected objects

foreach SuperPixel do
    // greedily maximise similarity
    set object candidate: Candidate ← SuperPixel ;
    set Distance ← NEM_R(Candidate, Template) ;
    while Distance decreases do
        set Neighbours ← adjacent Super-pixels of Candidate ;
        foreach Neighbour do
            compute NEM_R(Candidate ∪ Neighbour, Template) ;
        end
        find Neighbour_i which most decreases the Distance ;
        set Candidate ← Candidate ∪ Neighbour_i ;
    end
    // only keep candidate if similarity is high
    if NEM_R(Candidate, Template) > DetectionThreshold then
        delete Candidate ;
    end
end
// non-minima suppression in candidate list
sort Candidates in descending order of Distance ;
for i = 1 to #Candidates do
    for j = i + 1 to #Candidates do
        if Overlap(Candidate_i, Candidate_j) > OverlapThreshold then
            remove Candidate_j ;
        end
    end
end
// return detected objects
Detections ← Candidates ;
```

Abb. 2.11: Algorithm to find the best matching contours in a segmented image (from [2]).

An object is detected, if the area of the intersection of both bounding boxes divided by the area of the larger bounding box is greater than 20%.

Abb. 2.12: The template shape of a swan (left) and and a false object detection with this template shape (from [2] with changes).

Abb. 2.13: Two cases where image segmentation fails due to low contrast (from [2])

There are several reasons why object detection may fail with the algorithm presented in this chapter. Schindler et al. listed three of the most common problems. Since their approach is based on global matching, the whole contour of the template object is compared with the edges in the image at once. This leads to problems if only part of the object is visible. Usually such objects are not detected by the algorithm. In addition to that the elastic matching procedure allows some deformations that are implausible. Figure 2.12 shows a case where a swan was detected although the contour in the image is clearly different from to the template contour. A severe problem for the algorithm are images that cannot be segmented correctly. Figure 2.13 shows two cases where the segmentation fails due to low image contrast. Large parts of the object contour are missing and the object detection algorithm is unable to locate the objects.

5 Comparison and Evaluation

In this report two approaches for shape matching were presented: "From Partial Shape Matching through Local Deformation to Robust Global Shape Similarity for Object Detection" by Ma et al. and "Object Detection by Global Contour Shape" by Schindler et al. Both approaches seem to perform well based on the experimental results the authors provided. However, they differ strongly in detail which leads to certain advantages and disadvantages for each of them.

Clearly the main difference is the restrictions that they impose on the template shapes. The algorithm by Schindler et al. is only capable of handling template shapes consisting of the silhouette of an object whereas Ma et al.'s approach can handle all kinds of shapes. Not all shapes are clearly defined by their silhouette. For example the shape of the letter "B " can only be described by taking into account the two holes inside of it. Ma et al.'s approach is more flexible by allowing template objects that consist just of a bunch of points that may be connected.

One main drawback of Ma et al.'s shape descriptor and matching function is the fact that the matching is not rotation invariant. Thus objects that have a different rotation compared to the template object cannot be detected. This advantage can be compensated by using multiple template objects with different rotations. However, this clearly increases the runtime of the algorithm which makes it unusable for large sets of images such as the one presented in the case study in section 2.2. A more intelligent approach would be to first align edge fragments before computing the shape descriptor to compare them. Schinder et al.'s matching is rotation invariant and both approaches are translation and scale invariant.

Ma et al.'s approach is based on partial matching which leads to several advantages. If an object is only partly visible because of occlusion or because only part of the object is contained in the image, partial matching is still able to detect the object whereas the global matching of Schinder et al. has difficulties with these kinds of objects. In addition to that the transformation defined in section 3.4.2 can be used to complete objects with missing parts by transforming the template shape into the image. Partial matching is also useful in some applications such as the image retrieval system described in section 2.2.

Regarding the performance, both works do not contain any information about the computational complexity of the algorithms. When detecting objects in a large set of images, runtime may be a critical factor. Schindler et al. report that their detection process takes on average 6.5 seconds for an average image size of 480x320 pixels and 13.9 seconds for an average image size of 640x480 pixels for specific test data.

Literatur

[1] Tianyang Ma and L.J. Latecki. From partial shape matching through local deformation to robust global shape similarity for object detection. In *Computer Vision and Pattern Recognition (CVPR),*

2011 IEEE Conference on, pages 1441 –1448, june 2011.

[2] Konrad Schindler and David Suter. Object detection by global contour shape. *Pattern Recognition*, 41(12):3736 – 3748, 2008.

[3] Sameer Antani, Xiaoqian Xu, L. Rodney Long, and George R. Thoma. Partial shape matching for cbir of spine x-ray images. In *Storage and Retrieval Methods and Applications for Multimedia 2004, 20 January 2004, San Jose, CA, USA*, volume 5307 of *SPIE Proceedings*, pages 1–8, 2004.

[4] Xiaoqian Xu, Dah-Jye Lee, S. Antani, and L.R. Long. A spine x-ray image retrieval system using partial shape matching. *Information Technology in Biomedicine, IEEE Transactions on*, 12(1):100 –108, jan. 2008.

[5] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *In NIPS*, pages 831–837, 2000.

[6] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8:679–698, June 1986.

[7] Ronald Fagin and Larry Stockmeyer. Relaxing the triangle inequality in pattern matching. *International Journal of Computer Vision*, 30:219–231, 1998.

[8] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1 –8, june 2008.

[9] Vittorio Ferrari, Tinne Tuytelaars, and Luc Van Gool. Object detection by contour segment networks. In *Computer Vision - ECCV 2006*, volume 3953 of *Lecture Notes in Computer Science*, pages 14–28. Springer Berlin / Heidelberg, 2006.

[10] ChengEn Lu, L.J. Latecki, N. Adluru, Xingwei Yang, and Haibin Ling. Shape guided contour grouping with particle filters. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2288 –2295, 29 2009-oct. 2 2009.

[11] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1038 –1045, june 2009.

[12] Richard Nock and Frank Nielsen. Statistical region merging. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26:1452–1458, 2004.

[13] Massimiliano Pavan, Marcello Pelillo, and Senior Member. Dominant sets and pairwise clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI*, 29:2007, 2007.

[14] P. Srinivasan, Qihui Zhu, and Jianbo Shi. Many-to-one contour matching for describing and discriminating object shape. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1673 –1680, june 2010.

Near-Realtime Stereo Vision

Martin Lang

Inhalt

Zusammenfassung

This seminar paper gives a short introduction into computational stereo vision systems. Stereo vision systems are extracting depth informations out of two different images. This paper describes two different state of the art systems with near-realtime performance to give an impression of different approaches for extracting the depth information. The first system is the current top performer of the Middlebury stereo vision benchmark and the second one is a generic filter based framework. Furthermore this paper will shortly describe a few medical applications to show the variety of different possible application areas in medical practice.

Keywords: Stereo Vision, Near-Realtime, Depth-Information, Medical Applications

1 Introduction

Stereo vision is the process of extracting depth information out of two images. Like the eyes in human vision, two camera-objectives which are placed next to each other capture the same scene at the same time. The resulting two half images show the same scene but in two different angles (see Figure 3.1).

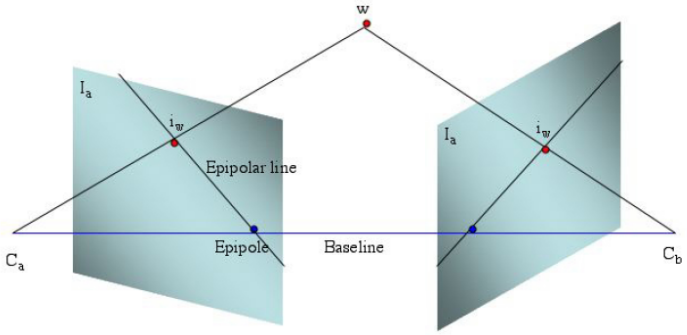


Abb. 3.1: Two cameras are each taking a picture out of two different angles [1]

Here are the two cameras C_a and C_b connected with a line. This is called the baseline. The points where the baseline intersects the images are called the epipoles. The lines which connect the epipoles with the image-points of an object in the scene are called the epipolar lines. Depending on the distance of the objects in the scene to the cameras, its projections on the half images are shifted along the epipolar lines. This distance between the corresponding image-points in both images along the epipolar line is called *disparity*. The disparity is high, if the object is close to the cameras and low if the object is far away. From now on we will only consider rectified images. Rectified images are post processed images in which vertical disparities have been removed. In order to do this, they are transformed in a way that the epipolar lines of the two images are parallel to the x-axis (shown in Figure 3.2). Now the disparity can be calculated as the difference of the x-coordinates of both image-points.

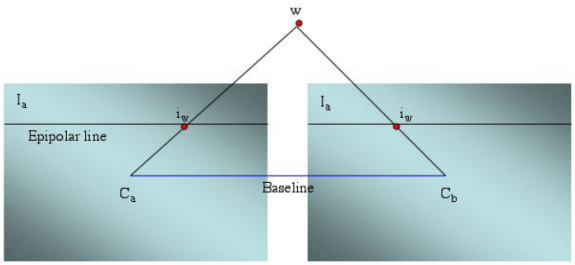


Abb. 3.2: Two rectified images with parallel epipolar lines to remove vertical disparities [1]

In order to get depth information out of these two half images it is helpful to compute the disparities. After that, it is possible to create one spatial image (*disparity map*) out of these disparities to show depth information. Here the brightness of the objects shows the relative distance (see Figure 3.3).

(a) (b) (c)

Abb. 3.3: Dataset *tsukuba* from the Middlebury dataset [2] with (a) left, (b) right half-image and (c) corresponding disparity map (based on the left image)

To calculate the disparities in computational stereo vision it must be computed first, which pixels belong together. This is called the *correspondence problem*. Here the rectified images simplify the correspondence problem enormously because corresponding pixels can only be in the same pixel row. However, the correspondence problem is the major problem in computational stereo vision.

According to Scharstein et al. [2], general stereo vision systems usually follow a certain pattern to handle this correspondence problem and calculate the disparity map:

- *Matching Cost Computation:*
  This is the initialization step. To find two corresponding pixels a similarity measure called the *costs* is defined. A simple example for a cost measure would be the color difference. The costs indicate the correspondence quality of two pixels - the lower the costs, the better they are matching. In this initialization step each pixel gets a cost value for each possible corresponding pixel.

- *Cost Aggregation:*
  The lowest initial costs of two pixels are no reliable indication that they belong together. To get more reliable costs it is better to use context to context mapping instead of pixel to pixel comparison. To obtain this, the pixels in the surrounding area are included if they are belong to same object. The disparities of that context are aggregated and normalized by the size of the support region.

- *Disparity Computation and Optimization:*
  This is the actual matching step. In this step the disparity results are created and optimized by using the cost values. After this step each pixel has its final disparity and a disparity map can be created.

- *Refinement:*
  Here the disparities are improved to get a smoother image. This can be obtained by removing discontinuities in the disparity map.

Although stereo correspondence is one of the most investigated topics in computer vision [2] there are still challenges such as the balance between accuracy and efficiency. On the one hand the stereo vision systems should create very accurate results which often result in complicated and expansive algorithms. On the other hand they should compute the results very fast - in best case with near-realtime performance. For example the implementation with CUDA (Nvidia programming interface) can achieve very efficient systems because a graphic card can process tasks with high parallelism much faster than a CPU. However, not all accurate algorithms are parallizable. In these cases, there is a trade-off between computation performance and accuracy.

Another problem is the occurrence of ambiguities. If there is for example a large texture-less region with many pixels of the similar color, the matching costs are similar if they are depending on the color difference. Therefore many disparities have low costs and it is difficult to identify the correct correspondence.

Furthermore, every time an objects is placed behind another one it comes to occlusions. Because of the two different angles of the cameras a part of the scene is visible from the angle of the first camera and occluded from the angle of the second camera. In these cases there is no corresponding pixels in the second image.

These are challenges a good stereo vision framework has to handle.

In the following chapters I want to give a short introduction in state of the art computational stereo vision systems with near-realtime performance. For this purpose I will describe two different frameworks: The first one is a very accurate framework created by Mei et al. [3] using an AD-Census distance measure with cross-based cost aggregation and scanline optimization. The second one is a generic and simple filter-based solution by Rhemann et al. [4].

2 AD-Census

In this chapter I will present a state of the art stereo vision framework by Mei et al. [3]. This framework is a collection of several steps, designed to be suitable for graphics hardware with near-realtime performance. According to Scharstein the framework can be split up into the following steps:

- *Matching Cost Computation*
  Two different cost measures are combined: the absolute difference (AD) and the census cost measure. Mei et al. combine these two models to consider both the color intensity (absolute difference) and the structure of surrounding pixels (census) for the matching process.

- *Cost Aggregation:*
  The initial costs are aggregated for each pixel including a local context to reduce the ambiguities and noise in the first cost volume. The aim is to get as much context of the same object as possible. For this purpose the authors use a cross-based cost aggregation algorithm that allows a flexible support region for each pixel. In this way the environment is not defined by a fixed window, it is a variable support region trying to adapt to the object of that pixel. Here the assumption is that pixels of the same object have a similar color. This aggregation achieves a context to context mapping instead of pixel to pixel correspondence.

- *Disparity Computation and Optimization:*
  In this step the disparities for each pixel are computed based on the aggregated cost volume. For that the authors are using a scanline optimization process. This process creates a smoothed disparity image by penalizing neighbours with disparity changes along horizontal and vertical scanlines and is based on dynamic programming. In that way the resulting disparity map contains less noise and looks smoother.

- *Refinement:*
  In the post processing step Mei et al. implemented a multi-step disparity refinement to improve the previous created disparity results by outlier handling.

2.1 AD-Census Cost Initialization

To match the pixels within the left and right images Mei et al. create a *cost volume* C that maps a pixel $p = (x, y)$ and a disparity d to a cost value $C(p, d)$. Based on the assumption that objects can only be shifted horizontally, the costs $C(p, d)$ evaluate how well a pixel p in the left and a pixel $pd = (x - d, y)$ in the right half image are matching. A lower cost value $C(p, d)$ means a high probability that these pixels correspond. This framework combines two different measures for cost initialization C_{AD} and C_{Census}.

- *Absolute Difference:*
  The absolute difference cost volume C_{AD} computes the costs by comparing the color of two pixels p and pd assuming corresponding pixels in the two images should have nearly the same color. Due to lightning effects exactly the same color can not be expected. Therefore C_{AD} is defined as the average color difference in the RGB channels:

$$C_{AD}(p, d) = \frac{1}{3} \sum_{i \in \{R,G,B\}} |LeftImage_i(p) - RightImage_i(pd)| \tag{3.1}$$

- *Census:*
  The census cost measure evaluates the matching costs p and pd by comparing the local structure of surrounding pixels in a small window. In order to be able to encode the structures of p and pd in two 64-bit strings the authors use a window of the size 9×7 (=63 bits). The census costs $C_{Census}(p, d)$ are defined as the Hamming distance of these two bit strings. The Hamming distance measures the difference of two strings by comparing the character on each position and is defined as the number of different meanderings. For two bit-strings the Hamming distance can be easily computed by counting the 1 bits after a xor operation. A possible encoding for the local structure would be this: The pixels in the window are numbered. This number is the position in the bit string where the pixel is encoded. If a pixel has a lower intensity than p, its position in the bit string is 1 and 0 otherwise. Figure 3.4 demonstrates the idea in a smaller 3×3 window.

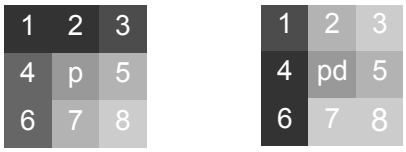


Abb. 3.4: The local structures of p and pd could be encoded as En_p="11110100" and En_{pd}="10010100". The result of the xor operation is "01100000" and the Hamming distance equals 2.

The AD-Census combined cost volume C is then computed as:

$$C(p, d) = \rho(C_{Census}(p, d), \lambda_{Census}) + \rho(C_{AD}(p, d), \lambda_{AD}) \tag{3.2}$$

where ρ is a normalization function that maps the different cost measures to a range of [0,1] and limits the influence of outliers by the parameter λ . In order to do that, ρ does not consider outliers with higher costs than the thresholds λ_{Census} and λ_{AD}. The idea of this combination is to reduce the disadvantages of the single methods and the resulting mismatches. The AD method can not handle large textureless regions because the color intensities are similar in these areas. The Census method instead is problematic for repetitive local structures because it cannot differentiate between objects with similar structures. The combination of them can consider both, the structure and the color and reduce the total mismatches (see Figure 3.5).

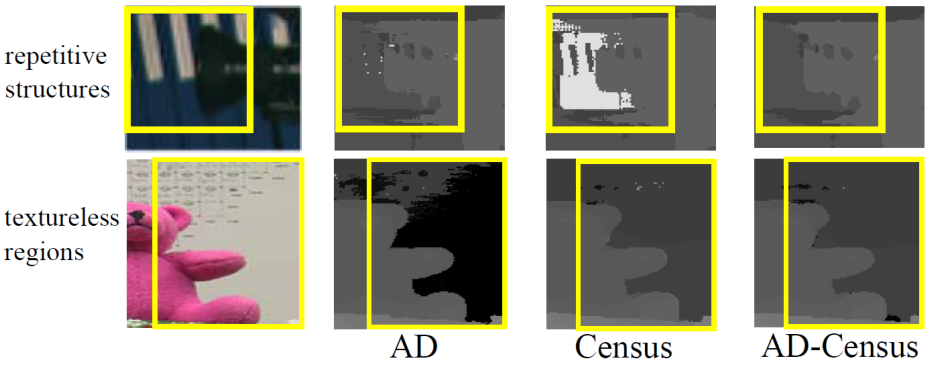


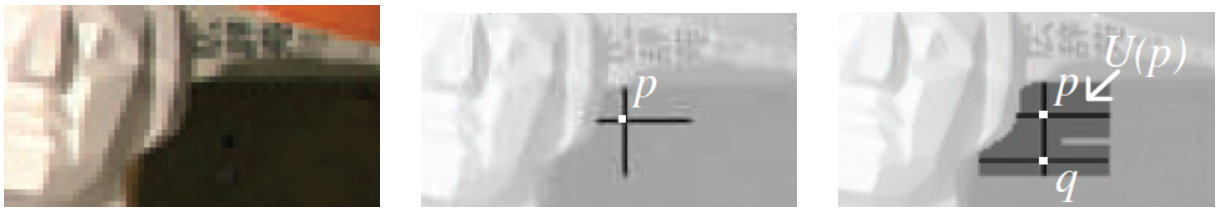
Abb. 3.5: Resulting errors by [3] of repetitive structures and textureless regions (after cross-based aggregation)

Because each of these steps is independent from the costs of other pixels, it should be noticed that these initialization steps can be performed in parallel for each pixel and each disparity.

2.2 Cross-based Cost Aggregation

The previously computed cost volume can contain matching ambiguities and noise. To reduce them Mei et al. use an improved version of cross-based cost aggregation proposed by Zhang et al. [5]. This method

wants to aggregate as much context as possible, but only of the same object. For this purpose a variable *support region* is created for each pixel that includes the neighbouring pixels with similar color. The advantage of a flexible support region is that it can adapt to the environment and the shapes of the scene (shown in Figure 3.6) instead of having a constant size. For example the support region of a pixel within many similar colored pixels is much bigger than the support region of a pixel neighboured with only a few similar colored pixels.



(a) (b) (c)

Abb. 3.6: The support region of pixel p adapted to the object [5]: (a) original image, (b) created cross at pixel p and (c) resulting support region. Here can be seen that the left and upper arms stop at color-changes and the right and bottom arms stop when distance threshold L_1 is reached.

- *Creating the support region:*
  To create the support region every pixel p gets a left, right, upper and bottom arm (shown in Figure 3.7(a)), which are forming a cross. Each arm of the cross has flexible length. To get a support region with similar colored neighboured pixels, the arm will include each pixel p_1 in that direction until a pixel is reached that does not fulfill one of these criteria:

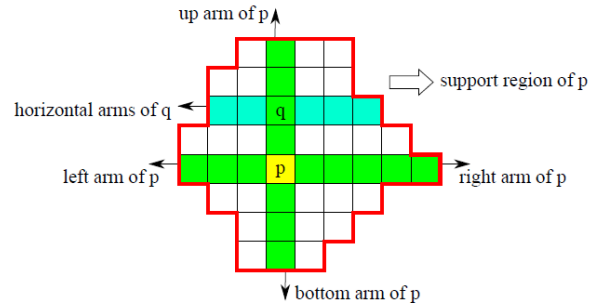
  1. The color of p_1 differs from p by less than color difference threshold τ_1.
     That means that only pixels with a similar color are included. Here the color difference is defined by $\max_{i\in\{R,G,B\}} |I_i(p_1) - I_i(p)|$.
  2. The color differs from p_1's predecessor (on that arm) by less than τ_1.
     This criterium avoids that the arm runs across the edges of the image.
  3. The distance $|p_1 - p|$ is less than the distance threshold L_1.
     This criterium limits the maximum length of the arms.
  4. If the distance $|p_1 - p|$ is more than L_2 the color difference from p must be less than τ_2.
     That means long arms are only allowed if the color difference is very small. Here τ_2 is a much stricter threshold and $L_2 < L_1$.

  With the similar proceeding each pixel on the vertical arm of p (q for example) creates its horizontal arms. All pixels included in the arms result in the support region of p.

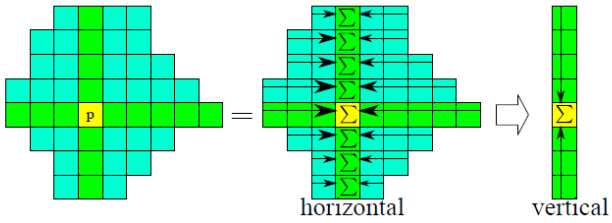
- *Aggregating the Costs:*
  Now the aim is to define the costs for matching the context of p to the context of pd. Therefore the costs for disparity d are aggregated for all pixels which are in the support reagion of p and its corresponding pixel is in the support region of pixel pd. To efficiently sum up the costs of all pixels within the support region, the costs of all pixels are summed up horizontally and then vertically (shown in Figure 3.7(b)). Therefore 1 D integral images can be used. In that way the stored sums can be reused to calculate the sums - instead of calculating the same costs many times. These aggregated costs are normalized by the size of the support region and result in the final costs of pixel p for the disparity d.

To obtain better aggregation costs Mei et al. repeat this aggregation by four iterations. That means the aggregated costs of the previous aggregation are used as new initial costs for the next iteration. But in the second and forth iteration the iteration directions are switched. That means they first calculate and sum up the vertical arms (on the horizontal arms of p) and then aggregated them horizontally. This leads to a different support region in the second and fourth iteration.

Thus they computed an improved cost volume (called C_1) containing the aggregated costs. These costs are smoother and contains less outliers.



(a) Cross Construction



(b) Cross Aggregation

Abb. 3.7: Cross-based cost aggregation scheme by [3]

2.3 Scanline Optimization

This is the real matching process taking the previous created cost volume C_1 to compute the resulting disparities. Pixel-wise cost calculation even with aggregated costs is generally ambiguous and wrong matches can easily have lower costs than correct ones [6]. Therefore a scanline optimization method based on Hirschmüller's semi-global matcher [6] with smoothness constraints will perform this step to create a smoother image by penalizing changes of neighbouring disparities.

The idea is to define a function which sums up the costs for all disparities in a disparity image, adding penalties for disparity changes. Now a smooth disparity image can be calculated by finding the image which minimizes the costs.

But because this problem is NP-complete [7], Hirschmüller simplifies this by minimizing the costs along individual scanline directions r to pixel p. The simplified problem can be calculated in polynomial time because it is only one-dimensional and only smooths the disparities along one direction. This minimization is done via dynamic programming for each direction vector r:

$$
\begin{aligned}
C_r(p,d) \quad = \quad & C_1(p,d) && (3.3)\\
& + \min(C_r(p-r,d), C_r(p-r,d\pm 1)+P_1, \min_k C_r(p-r,k)+P_2) && (3.4)\\
& - \min_k C_r(p-r,k) && (3.5)
\end{aligned}
$$

Here P_1, P_2 are the penalizing parameters with $P_1 \leq P_2$, which are set depending on the color difference of the left and right half images.
This function implies three parts:

Term 3.3: Adding the costs of the previous cost volume

Term 3.4: Adding the minimal costs over the previous pixel in that direction with:

- the same disparity
- nearly the same disparity (± 1) penalized by P_1
- the minimum of all disparities heavily penalized by P_2

Term 3.5: Subtracting the minimum of all disparities. This does not change the result of the minimal path because it is equal for all disparities and should just slow down the growth.

Unlike Hirschmüller, Mei et al. only use four scanline directions (two horizontal and two vertical ones). At the end the final cost volume C_2 can be computed at the average of the scanline costs:

$$C_2(p,d) = \frac{1}{4}\sum_r C_r(p,d) \tag{3.6}$$

Finally the disparity with the minimal costs is selected for the disparity image D:

$$D(p) = \arg\min_d C_2(p,d) \tag{3.7}$$

2.4 Multi-Step Disparity Refinement

For the post-processing Mei et al. use a multi-step disparity refinement which tries to handle outliers contained in the disparity image in four steps. In fact the disparity map is not only computed once. For outlier detection this computation is done twice: one time from the left to the right half image with the resulting disparity map D_L. And one time from the right to the left half image with the resulting disparity map D_R. To detect the outliers they compare the left and right disparity map. p is detected as outlier, if

$$D_L(p) \neq D_R(p - (D_L(p), 0)) \tag{3.8}$$

This means the algorithm computes different disparities for corresponding pixels and therefore the disparities are not consistent. Additionally the authors differentiate between *occlusions* and *mismatches*. This is done by comparing the epipolar-line $e_p(d) = (x_p - d, p_y$ of an outlier with $D_R(p)$. If the two functions intersect, the outlier is marked as occlusion. Otherwise the outlier is marked as mismatch.

The detected outliers p_{out} are handled by the following four consecutive steps:

1. *Iterative Region Voting:*
   Here the support region of the previous cost aggregation step is reused to "vote" for a disparity of pixel p_{out}. If there are enough reliable votes (votes of non-outlier pixels), pixel p_{out} gets the most frequent disparity d^* in its support region and is not marked as outlier any more.

2. *Interpolation:*
   If p_{out} is an occlusion, the nearest reliable pixel with the lowest disparity is selected for interpolation, because p_{out} is probably occluded by this pixel. If p_{out} is a mismatch, the pixel with the lowest color difference is selected instead.

3. *Depth Discontinuity:*
   First all edges in the disparity image are detected. If $D(p)$ is on a disparity edge, we compare it with one pixel p_1 on the one and p_2 on the other side of the edge. If one of them has lower matching costs than p, $D(p)$ is replaced by $D(p_1)$ or $D(p_2)$.

4. *Sub-pixel Enhancement:*
   In this last refinement step the outliers caused by discrete disparity levels are handled. The problem is that the disparity is discrete in the disparity map but can be continuous in reality. To handle this problem a quadratic polynomial interpolation is used to calculate a disparity between $d+1$ and $d-1$. Afterwards a 3×3 median filter is used to get the final disparities.

As you can see, this framework is performing a lot of different steps. Here complex aggregation and optimization methods are used to calculate the disparity map. Afterwards a multi step refinement process optimized for stereo vision is used to improve the disparity map. With the next framework, I will show a more general framework which only uses a general filtering process and less refinement steps.

3 Fast Cost-Volume Filtering

In this chapter I will present the state of the art stereo vision framework by Rhemann et al. [4]. This framework is based on a filtering process which can be additionally applied to many different computer vision tasks such as optical flow and image segmentation. The four steps of Scharsteins schema are realized in the following way:

- *Matching Cost Computation:*
  In this filter based framework, the cost initialization is obtained by the truncated absolute difference cost model.

- *Cost Aggregation:*
  For cost aggregation (which is the central point of this framework) they use a filtering process which can smooth the disparity image but preserves the edges of the reference image (see Figure 3.8). For this purpose they applied a filter proposed by He et al. [8].

- *Disparity Computation and Optimization:*
  After that the cost computation is obtained in a simple winner-takes-all manner by taking the disparities with the lowest costs for each pixel.

- *Refinement:*
  In the refinement process they implemented two steps. First they replace the occluded pixels by the spatially closest non-occluded pixels. After that they remove artifacts (which can unintentional be generated in this previous step) by a median filter.

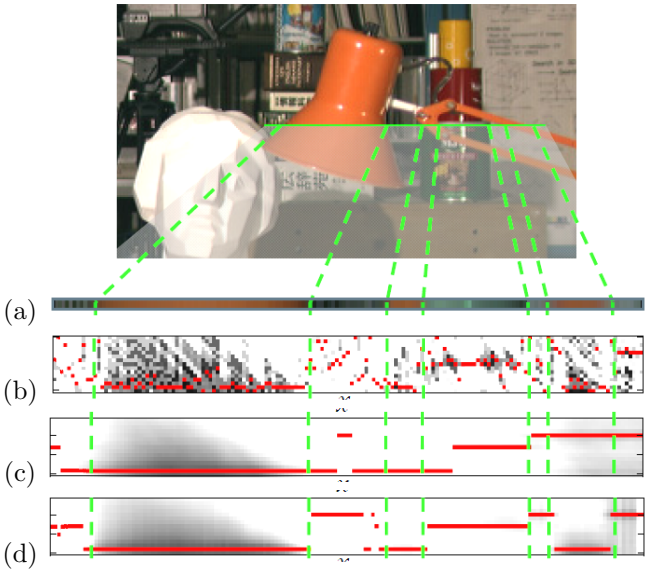


Abb. 3.8: This graphics [4] show different cost volumes of the pixels at the green line. Here are the guidance image (a), the cost volume (b) with (white/black/red : hight/low/lowest costs), a smoothed cost volume (c) and a smoothed and edge preserved cost volume filtered by the guided filter (d)

Even though the proposed method can be implemented on a CPU efficiently, the authors decided to create a GPU based framework to reach even better performance.

3.1 Cost Initialization

For the initial cost volume $C(p,d)$ Rhemann et al. adopt a cost model which considers the absolute difference of the color (similar to Mei et al. cf. equation 3.2) and the gradient at the matching points. Their initial cost value is defined by:

$$\begin{aligned} C(p,d) &= (1-\alpha)\cdot\min(\|I_p - I'_{pd}\|, \lambda_1) + \\ &\quad \alpha\cdot\min(\|\nabla_x I_p - \nabla_x I'_{pd}\|, \lambda_2) \end{aligned} \tag{3.9}$$

with the truncation thresholds λ_1 and λ_2. Here I_p is a 3×1 color vector of the one half image at position p and I'_{pd} the color vector of the other half image at the position p shifted by d. ∇_x is the gradient in x-direction and α a parameter to influence the balancing of color and gradient difference.

3.2 Cost-Volume Filtering

The filter process is the central point of this framework and the second step in Scharsteins schema. Here the previously created cost volume $C(p, d)$ is divided into slices for each disparity d. Each of these slices is filtered by a filter proposed by He et al. [8]. This is the aggregation step. Here the costs of pixel p at disparity d are aggregated with the costs of all pixels q in the same slice (with the same disparity d in the cost volume) depending on their weights $w_{p,q}$. This aggregation of each slice leads to a smoother image.

$$C'(p, d) = \sum_q w_{p,q} \cdot C(q, d) \tag{3.10}$$

With a guided filter it is possible to preserve the edges in the original image while smoothing. This is obtained by the weights $w_{p,q}$. To understand the idea I will explain how it works with gray-scale images first. Here the weights are defined as the sum of all square windows that includes p and q:

$$w_{p,q} = \frac{1}{|w|^2} \sum_{k:(p,q)\in w_k} (1 + \frac{(I_p - \mu_k)(I_q - \mu_k)}{\sigma_k^2 + \epsilon}). \tag{3.11}$$

Here μ_k is the mean, σ_k the variance, I_p and I_q are the gray scales of pixel p and q. Furthermore w_k is a square window w with the center pixel k, $|w|$ is the number of pixels in w and ϵ is a smoothing parameter.

Hereby the edge-preserving is obtained by the sign of the numerator. If p and q are on the same side of an edge, $(I_p - \mu_k)$ and $(I_q - \mu_k)$ have the same sign. Thus their product and the numerator are positive. If they are on different sides of an edge, they have got different signs and the numerator is negative. Hence, the term $(1 + \frac{(I_p-\mu_k)(I_q-\mu_k)}{\sigma_k^2+\epsilon})$ is large if p and q are on the same side of an edge and their weight $w_{p,q}$ is high. The term and the weight $w_{p,q}$ are low if they are on different sides. Through this approach the aggregation in (3.10) takes only an average of the pixels on the same side of an edge (in the guidance image) whilst the pixels separated by an edge are nearly disregarded. The amount of smoothing can be influenced by the smoothing parameter ϵ. If the smoothing parameter is high, the denominator is high and the absolute value of the fraction is lower. This leads to more similar weights and a higher amount of smoothing with less edge-preserving. Similarly to the grey-scale image weights, the weights for colored images can be defined by:

$$w_{p,q} = \frac{1}{|w|^2} \sum_{k:(p,q)\in w_k} (1 + (I_p - \mu_k)^T (\Sigma_k + \epsilon U)^{-1} (I_q - \mu_k)) \tag{3.12}$$

Here I_p and I_q are μ_k 3×1 color vectors (RGB values) and the covariance matrix σ_k and the identity matrix U are of the dimensions 3×3.

In this framework the third step according to Scharsteins schema is held really simple. Here the authors do not perform any optimization and the disparities for the disparity map are selected (similar to Mei et al.) in a winner-takes-all manner over all disparities of a pixel p. So the disparity with the lowest cost is set as the disparity:

$$D(p) = \arg\min_d C'(p, d) \tag{3.13}$$

3.3 Refinement

Similar to Mei et al this approach uses a consistency check (cf. Equation 3.8) to detect outliers, which then are processed in the following way:

1. *Replacement:*
   Here the disparities of all occluded pixels are replaced with the disparity of the spatially closest non-occluded pixel on the same scanline (horizontal). Because this line-wise method can easily generate artifacts (streaks on the image) a post-processing method is necessary afterwards.

2. *Post-Processing:*
   To remove these streak-like artifacts a median filter is used. A median filter creates a rank order of all pixels in a window around a pixel p and selects the one in the middle of the ranking (the median) as new disparity for p. As filter weights they use those of the bilateral filter, which is edge preserving, too.

Tab. 1: Results of the middlebury stereo vision benchmark [9].

Name	Rank	Percentage of all bad pixels in					Average time	fps
		tsukuba	teddy	venus	cones	Average		
ADCensus	1	1.48	0.25	6.22	7.25	3.80	59 ms	17
Costfilter	16	1.85	0.39	11.8	8.24	5.57	65 ms	15

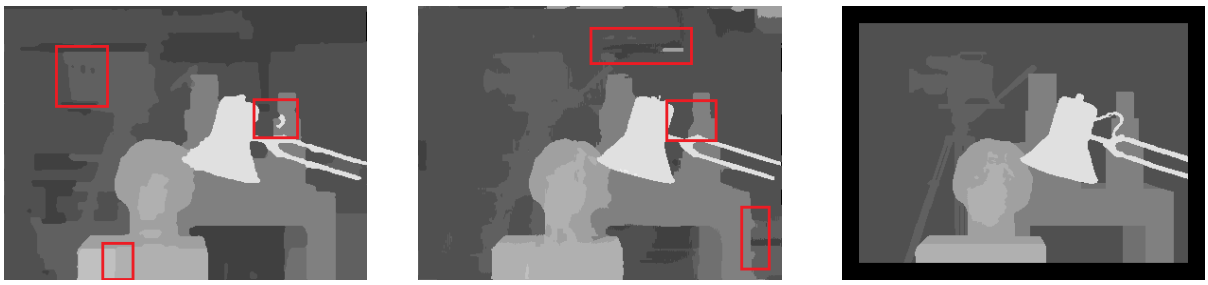
4 Comparison and Results

To evaluate their results, both frameworks tested their results on the Middlebury stereo vision benchmark created by Scharstein [9]. This benchmark takes the resulting disparity maps of four image pairs tsukuba, venus, teddy and cones (see Figure 3.9) and calculates the errors in the disparity maps. The image pairs have different resolutions (384×288, 434×383, 450×375, 450×375) and the disparity maps must be scaled by different disparity ranges (0..19, 0..19, 0..59, 0..59).

(a) tsukuba (b) venus (c) teddy (d) cones

Abb. 3.9: These are the four left images of the image pairs [2, 10] used for the Middlebury stereo benchmark.

In Figure 3.10 the resulting disparity maps of the two frameworks are shown for the tsukuba image pair. Here it can be seen that the results are close to the groundtruth disparity map, but still have some wrong disparities (example error regions are marked red). Here it can be noticed that the edges in the ADCensus disparity map are more clear than the one of the Costfilter disparity map.



(a) ADCensus (b) Costfilter (c) grountruth

Abb. 3.10: These are the resulting disparity maps calculated by ADCensus and Costfilter and the groundtruth disparity map for the dataset tsukuba (see Figure 3.3).

In the benchmark, the framework by Mei et al. currently ranks first (These are the ranks at the moment of this seminar paper (12. Juli 2012) and can change over time.) with a 3.80 average percent of all bad pixels of all datasets. The framework of Rhemann et al. is currently ranked 16^{th}. It reached 5.57 average percent of all bad pixels. As you can see in Table 3.10, the framework of Mei et al. obtains better results in every category. But being ranked 16^{th} in over a hundred stereo systems, the one by Rhemann et al. is still one of the top accurate systems. Notice that the benchmark just counts errors over a certain threshold (here: absolute disparity error which of is greater than 1). If the difference is smaller, the error is not counted. Here completely wrong disparities are not evaluated worse than an error hardly over that threshold. In that way the benchmark can only give a limited evaluation.

Additionally this benchmark does not consider the processing speed of the frameworks. Here, the framework of Mei et al. needs an average processing time of 59 ms (mean over 16 ms, 32 ms, 95 ms, 94 ms

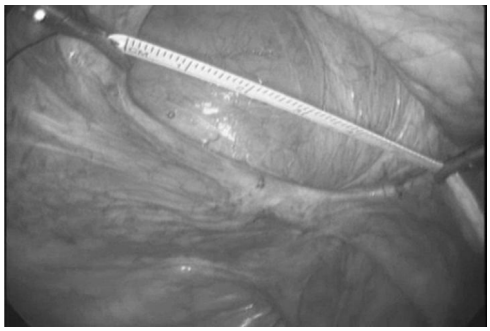


Abb. 3.11: In this current technique a tape is used for measurements. This task could be performed by a stereo endoscope without a tape [12]

for the different datasets [3]) for creating a final disparity map. Rhemann et al. need a comparable average runtime of 65 ms [4]. These average processing times correspond to an average frequencies of 17 and 15 fps. This mean the results both work with near-realtime performance. Unfortunately the measurements are not performed on the same machines and hence are not directly comparable.

An important difference between these two frameworks is the approach. The first system is exclusively created and optimized for stereo vision tasks and contains a lot of optimization and refinement steps. The second one is a more general filter framework which is designed to be suitable for different vision tasks. It uses basically a filter and performs much less refinement steps. However ADCensus does not need a longer processing time due to the parralel implementation on the GPU and Costfilter obtains a comparable accuracy.

5 Medical Applications

Stereo vision systems with near-realtime performance can be applied in many various application areas. In this chapter I will present a couple of different medical applications. In these examples stereo vision is used for measurements, navigation or the detection of nearby obstacles.

A big difference in practice is the fact that the captured images are not post-processed like the ones in the middlebury dataset. Therefore the calibration of the two cameras is an important additional task.

5.1 Stereo Endoscopy for Minimally Invasive Surgeries

The fist example of a medical application is the stereo endoscope in minimally invasive surgeries. Minimally invasive surgeries (MIS) are surgeries with only a small cut. In that way the skin and the tissue will be less damaged. As a direct consequence of that, there should be less complications after the surgery and the time of convalescence should be shorter than after an open surgery.

In the MIS endoscopes are inserted through the small cut to watch inside the patient and allow the entry for medical instruments. But this endoscope only allows a smaller field of view than possible in open surgeries. Additionally, during the MIS the surgeon only has 2-dimensional informations in the pictures. Therefore the size of an object can not be calculated exactly by the images and this is one reason for surgical accidents [11].

Lim et al. [11] and Field et al. [12] propose methods, which use a stereo endoscope in combination with near-realtime stereo vision systems to create three-dimensional informations during the endoscopy. This depth information can help the surgeon during the procedure. The three-dimensional positions can be used for measurements - for example calculating a tumor size. Moreover, the 3-dimensional coordinates can be reused to plan the path of a medical roboter for the actual operation.

In that way the stereo-vision can help during minimally invasive surgeries and reduce the risk of accidents.

5.2 Navigation of a Robotic Wheelchair

Another field in which stereo vision system can be used is the navigation of medical aid devices. Bailey et al. [13] for example develop a robotic wheelchair for handicapped persons who are not able to control

the wheelchair by themselves. This wheelchair should be capable to navigate through unknown locations automatically. For example it should be able to find a room by the room number in an unknown building. For the navigation of this chair a near-realtime stereo vision system is in use and enables the wheelchair

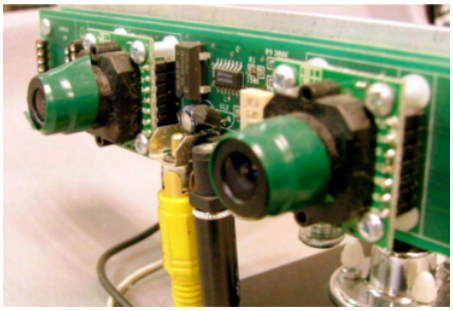


Abb. 3.12: Robotic wheelchair with stereo vision systems for navigation [13]. Here the camera is placed at the seat for testing. In practice the camera would be placed above the legs of the person.

to drive automatically and safely. The captured pictures are used to create a map of the current building. This is called simultaneous localization and mapping (SLAM). The range information for this mapping are normally created by active vision systems like laser-range finders or a sonar. But laser range finders are expensive and sonars produce an audible clicking sound that could bother the person over time [13]. Stereo vision do not have these disadvantages and can do the same work as well. Furthermore the wheelchair contains a robotic arm to open doors and press elevator buttons. Here, the stereo-camera helps to find the position of the door knob or button. Additionally, in contrast to a sonar the pictures of the stereo-camera can be reused to detect cues such as room numbers to derive which is the right direction to the desired room.

5.3 The vOICe - Seeing for the Blind

The last example I want to present is the project “the vOICe” [14]. This project is an open ecosystem for distributed research, development and deployment and involves hundreds of people around the world. The aim of this project is to help totally blind people to see with sound. For this purpose they created a head-mounted camera which can encode pictures in sound. To encode a picture, they scan the picture

Abb. 3.13: A head-mounted camera like this Vozix Wrap 920AR [15] could be used to create the stereo images.

every second from left to right with a scanline. Brighter areas are encoded with louder sounds, areas in top of the picture with higher sounds. There is a lot of practice necessary to decode these sounds afterwards and extract image-like information. Then it is possible to detect and even identify objects. 3.14.

Recently, a support for stereo vision extension is developed [16]. This stereo vision extension should enable blinds to more mobility. While orientation can be obtained with only one single camera, this stereo camera can additionally calculate depth information. This improves the mobility because nearby objects and obstacles can be detected. The stereo vision system can create a disparity map in near-realtime and this disparity map is directly encoded in a similar way. Here a louder sound means that there is a nearby object. An example is given in Figure

(a) (b) (c)

Abb. 3.14: Result of a suitable stereo camera [16] with (a) left, (b) right eye image and (c) corresponding disparity map. Here you can clearly detect the tree as obstacle. The scanline moves one time per second from left to right over the map while the pixels on the scanline are encoded in sound.

6 Conclusion

This seminar paper gave a short introduction to computational stereo vision. It presented two frameworks to show different ways of calculating a spatial, three-dimensional image out of two images representing the left and right view of the scene.

We have seen that the stereo vision system bei Mei et al. [3] which is created and optimized for stereo vision tasks obtained better results then the one of Rhemann et al. [4] . On the other hand the one by Rhemann et al. obtained good results as well with only a generic simple filtering process and much less refinement steps. Despite these near-realtime stereo vision systems can achieve a high accuracy in test conditions the situation in practice is different. The shown frameworks are tested on a rectified image pair of a test set. The depth calculation of a captured video in the real world is a much more challenging task. Here, additional problems like the calibration of the two cameras and vibrations need have to be taken into account. Hence, stereo vision is still a current research topic.

Nevertheless, there are many application areas where stereo vision can be used. A few medical applications are presented in this seminar paper to give an idea of the different application areas like navigation, measurements and so forth. There are additionally a lot of different application fields apart from medicine. This will assure that the research and employment of near-realtime stereo vision will still be a topic of much interest in the future.

Literatur

[1] A. Dankers, N. Barnes and E. Zelinsky: *Active Vision - Rectification and Depth Mapping,* Australian Conference on Robotics and Automation, 2004.

[2] D. Scharstein and R. Szeliski: *A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms,* International Journal of Computer Vision, 2002; 47(1-3): 7-42.

[3] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang: *On Building an Accurate Stereo Matching System on Graphics Hardware,* IEEE International Conference on Computer Vision Workshops, 2011; 467-474.

[4] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz: *Fast Cost-Volume Filtering for Visual Correspondence and Beyond,* IEEE Conference on Computer Vision and Pattern Recognition, 2011; 3017-3024.

[5] K. Zhang, J. Lu, and G. Lafruit: *Cross-based Local Stereo Matching Using Orthogonal Integral Images,* IEEE Transactions on Circuits and Systems for Video Technology, 2009; 19(7): 1073-1079.

[6] H. Hirschmüller: *Stereo Processing by Semiglobal Matching and Mutual Information,* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008; 39(2): 328-341.

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

[8] K. He, J. Sun, and X.Tang: *Guided Image Filtering,* In Proceedings of the 11th European Conference on Computer Vision, 2010; 1-14.

[9] D. Scharstein and R. Szeliski: *Middlebury Stereo Evaluation - Version 2,* 2010 [cited 2012 January 26] Available from http://vision.middlebury.edu/stereo/eval/.

[10] D. Scharstein and R. Szeliski: *High-Accuracy Stereo Depth Maps Using Structured Light,* IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003; 1: 195-202.

[11] J. Lim; D. Woo; H. Chun; B. Keum; J. Hyun; Y. Kim; M. Lim: *Tumor Size Measurement and Feature Point Extraction Using an Endoscope,* IEEE International Conference on Control Automation and Systems, 2010; 105-108.

[12] M. Field, D. Clarke, S. Strup, MD, and W.B. Seales: *Stereo Endoscopy as a 3-D Measurement Tool,* Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2009; 5748-5751.

[13] M. Bailey, A. Chanler, B. Maxwell, M. Micire, K. Tsui and H. Yanco: *Development of Vision-Based Navigation for a Robotic Wheelchair,* IEEE International Conference on Rehabilitation Robotics, 2007; 951-957.

[14] P. Meijer: *Augmented Reality for the Totally Blind,* 2011 [cited 2012 January 23]. Available from: http://www.seeingwithsound.com.

[15] Vuzix Corporation: *Vuzix - View the Future,* 2011 [cited 2012 January 23] Available from: http://www.vuzix.com/consumer/products_wrap920ar.html.

[16] P. Meijer: *Stereoscopic Vision for the Blind,* 2011 [cited 2012 January 23]. Available from: http://www.seeingwithsound.com/binocular.htm.

Robust tracking for the Seminar Medical Image Processing im Wintersemester 2011/2012

Ramya Nagarajan

Inhalt

Zusammenfassung

Object Tracking is an interesting tracking task in automatic video analysis. It has wide applications in industries and in analysing medical images. This seminar work discourses three robust algorithms proposed by A.Adam, E.Rivilin and I.Shimshoni in [2], H.Grabner, C. Leistner and H.Bischof in [10], B.Babenko, M.H.Yang and S.Belongie in [3]. The algorithms were tested on video sequences by Babenko et al in their paper. A tracking method that achieves robustness for all scenarios is still an open question. The methodologies used, the performance results and the pros and cons of each algorithm are discussed in this paper.

Keywords: Object tracking, FragTracker, SemiBoost, Semi- Supervised learning, Multiple Instance learning, appearance models, drifting, occlusion, Robust tracker.

1 Introduction

In the medical field, object tracking techniques greatly assist medical experts during their analysis. In sports medicine, for curing the athlete in case of injuries during the event, tracking of the victim‘s body parts during the event would show the cause of the accident that would help therapists in the treatment. Similarly, for checking the rate of recovery in body parts after surgeries, automatic tracking of repaired parts in medical images makes the task easier and accurate. This brings out the concern for robustness in tracking algorithms. Object tracking is a challenging task as the condition of the object in frame sequences is neither easily known in advance nor predictable. The object could get partially or fully occluded, lighting conditions may change, noise in the video sequence and fast movement of the object are issues that algorithms should handle.

The primary concepts in object tracking are explained in brief. The shape of target object is represented as a centroid point or set of points, geometric shapes, by their contour, skeletal models, as in [19]. The appearance of the object can be represented as a template that remains static or as probability densities either parametric like Gaussian or non parametric like histograms. The FragTracker algorithm in [2] uses template representation while the Semi- Supervised Online Boosting tracker [10] and Multiple instance learning (MIL) tracker [3] use probabilities of features to define the appearance of an object. A tracker finds the location of an object in every frame and estimates correspondence with the located object from the previous frames. Differences in establishing the correspondences lead to diversity in tracking algorithms.

Object tracking is also tackled as a classification task in recent research. A classification model for the object is generated to discriminate it from the background. Samples of the target object are used for training the classifier either off- line, or in some cases the learning process still continues during tracking. The classifiers used are either discriminative or generative. Woodley et al in [18] have incorporated both classifier types in their tracker. Selecting good feature that discriminates one class from another effects the performance of the classifier greatly. The feature could be color, edge, or texture that represents either the object or the background depending on the classifier.

A tracker comprises of three parts, the appearance model, the motion model and the search strategy [3]. The appearance model represents the likely appearance of the object and helps calculating the most likely location of the object in next frame. The appearance model is either static as in the FragTracker algorithm or adaptive as in Semi- Supervised Online Boosting tracker and MIL tracker. The motion model describes the motion of the object over time. The search strategy scans the image (the frame where the object is to be located) to find out the most probable location of the object.

The FragTracker algorithm handles occlusion well, but because of a fixed appearance model, it fails in tracking an object that frequently changes appearance. The Semi- Supervised Online Boosting tracker uses an adaptive appearance model, hence handles appearance changes over time. The MIL tracker is shown efficient during steady appearance changes and partial occlusions.

This paper is structured as follows, Section 2 highlights the current state of art in object tracking. Following that in Section 3, the three robust algorithms are discussed. Section 4 includes the experimental

comparisons and results from [3]. Finally, in Section 5 the discussion on advantages and drawbacks of each algorithm is added.

2 State of the Art

The tracking methodology is categorised into point, kernel or Silhouette by Yilmaz, Alper et al. in [19]. A point tracking considers the object as a point [6]. In this case, the motion of the object towards the next frame is only translational. The kernel tracking uses a rectangular template to represent the object. Motion of the object is a parametric transformation such as rotation, affine, translation. The three algorithms discussed in this paper belong to this type. The silhouette tracking uses the information encoded inside the object region. The information is represented as edge maps that are tracked using contour evolution [11].

The kernel based trackers either uses a template matching strategy or use classifiers to locate object in the new frame. In [7], a weighted histogram obtained from the object‘s kernel is compared to the histogram of similar kernel around a hypothesized object position in the image (target frame). The corresponding kernel in the image is selected based on the mean- shift approach that aims to maximize the appearance similarity between the histograms in each iteration. For the second type, the appearance model of either object [4] or that of both object and background is used to discriminate the object (the positive class) from background (the negative class) [17]. The Online Boosting classifier [9] learn from samples of the image and use boosting to combine them to generate a classifier. In [17], Woodley et al suggest to use a generative classifier as a fixed prior model to guide the classifier to achieve robustness during occlusions.

Static appearance model is called as template models. In [13], a static model is defined for the background which is used in background subtraction to obtain the foreground object while tracking. For adapting the object‘s appearance model, the authors [1] describe using EM algorithm to change the model‘s parameters. Choosing the positive and negative samples in every frame for training the classifier is the next concern. The discriminator should not over-fit as well learn from less number of class samples. Supervised learning methods require labelled training samples for each class. Co-training [5], trains two classifiers with different sets of labelled data and the features used by these classifiers are independent. After training, each classifier assigns label for a unlabelled data that becomes the training set for the other classifier. The authors claim that co training can provide a very accurate classification rule. A Semi-Supervised boosting classifier implemented in [16], uses labelled samples either from the first frame or are given a priori and the unlabelled instances are extracted from subsequent frames during tracking. The unlabelled examples along with a supervised algorithm increases the accuracy of the overall classifier. This SemiBoost framework handles occlusions well. Viola et al suggested to learn the classifier with multiple instances of object and background rather than a single positive and negative instance from each frame. They implemented the MIL concept to train the classifier for face recognition [15]. A collection of positive and negative instances are represented in a bag which is labelled respectively. Babenko et al [3], has combined the concept of MIL with boosting and used it with online feature selection for object tracking and has shown results for its efficiency.

3 Robust Tracking Algorithms

A robust tracker algorithm should address to all possible scenario the object might face. When the tracker uses a static appearance model, it fails in tracking when the target changes appearance, during partial occlusions or when the illumination changes. Failure occurs in these situations as the tracker would still try to locate an object that is similar to the appearance model. On the other hand, a tracker that uses an adaptive appearance model faces drifting. Drifting is the effect of error accumulated by the tracker in every frame during tracking. A robust algorithm could either choose to have a fixed training data set that does not lead it to drifting or update the training samples on-line after every frame so as to avoid problems because of appearance changes. The following subsections discuss the three robust algorithms from [2],[10] and [3].

3.1 Robust Fragments- based Tracking [2]

Fragment tracker (FragTracker) is a region based approach using histograms, inspired by the mean- shift procedure [7]. FragTracker uses a template object as its static appearance model. The template object

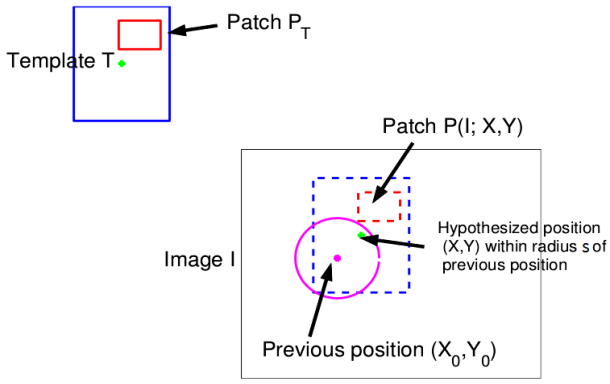


Abb. 4.1: Template and image patches representation. The pink circle in right image shows the search range s. Image source: Robust Fragments-based tracking using the Integral Histogram [2].

is represented as multiple fragments or patches. Every patch votes for its possible position in the new frame. Votes from all patches of the template object are combined to predict the location of the object in image I (current frame). Using region histograms in tracking, give rise to issues such as loss of spatial information and occlusion. The FragTracker deals with these problems in the algorithm.

Notations in Algorithm Refer Figure 4.1. The target object is represented as a rectangular template which is denoted by its center (x, y). Multiple rectangular sub- patches are cropped around the center. A patch from the template is denoted as P_T. The current frame is denoted by I. For instance, if a patch P_T is at distance (dx, dy) from template center then it is represented as (dx, dy, h, w) where h and w are half height and half width of the rectangular patch. Let the position of object in previous frame be (X_0, Y_0) and the search radius be s. In the frame I, a hypothesized point (X, Y) within the neighbourhood of (X_0, Y_0) (within the radius s around (X_0, Y_0), as the object probably is located here) is chosen. A patch $P_{I;(X,Y)}$ whose center is at distance (dx, dy) from the point (X, Y) is the corresponding patch for P_T. Patch $P_{I;(X,Y)}$ is denoted as $(X + dx, Y + dy, h, w)$.

Algorithm Patches P_T and $P_{I;(X,Y)}$ are represented as integral histogram [17]. Histograms with integral image data structure can represent high dimensions and integral image at point (x, y) constitutes sum of pixels contained within rectangular region bounded by top- left corner of the image and point (x, y). The purpose of using integral histogram is to achieve robustness. The two histograms from patch P_T and $P_{I;(X,Y)}$ are compared using a similarity measure. The distance $d(P_{I;(X,Y)}, P_T)$ denotes how similar both the patches are, the more similar they are, the more possible is for hypothesized position (X, Y) to be object's new location. The similarity value is the vote map for patch P_T as in Eq (1). The vote map of patch P_T describes its possible position in target frame I. For every patch P_T, a vote- map is generated by computing distance between patch P_T and its corresponding patch $P_{I;(X,Y)}$. Similarly vote maps for multiple patches around the template centre are generated.

$$V_{P_T}(X, Y) = d(P_{I;(X,Y)}, P_T) \tag{4.1}$$

Every patch is represented as either gray- scale or color histogram. Using a single patch in [7] loses spatial information of the object. In FragTrack algorithm, multiple sub- patches that maintain spatial relationship between them are used. The similarity between histograms is computed using metrics such as chi- square statistic as in Eq (2), or simply considering two histograms as vectors and use Euclidean distance to get the similarity. A naive approach considers the histogram as vectors and use norm over their differences.

$$\chi^2(h_{P_T}, h_{P_{I;X,Y}}) = \frac{1}{2}\sum_{k=1}^{K}\frac{\left[h_{P_T}(k) - h_{P_{I;X,Y}}(k)\right]^2}{h_{P_T}(k) + h_{P_{I;X,Y}}(k)} \tag{4.2}$$

where h_{P_T} and $h_{P_{I;(X,Y)}}$ are the histograms and K is number of bins in the histogram.

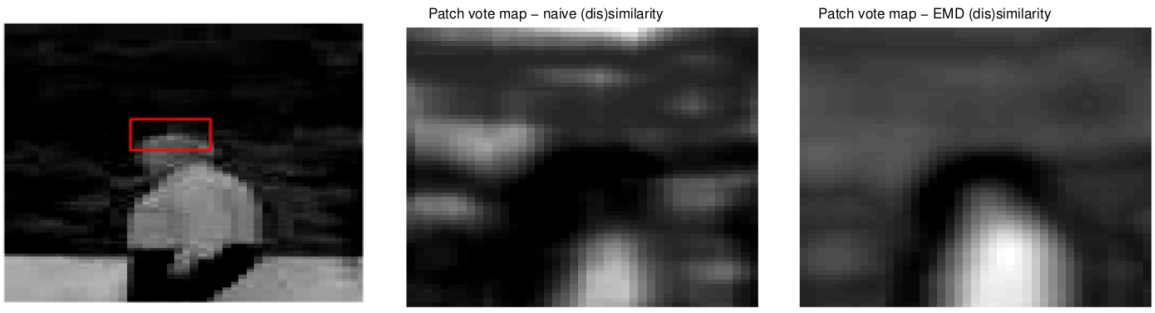


Abb. 4.2: Vote maps for example patch in red rectangular box. Image two is for naive based approach and third image is with the EMD measure. Image source: Robust Fragments-based tracking using the Integral Histogram [2].

Earth Mover‘s distance is another similarity measure introduced in [20]. This approach measures how dissimilar two histograms are by computing the distance cost needed to convert one histogram into another i.e how much probability has to be transported from one bin to another to make the histograms similar. Eq (3), computes the dissimilarity value.

$$EMD(h_{P_T}, h_{P_{I;X,Y}}) = \frac{\sum_{p=1}^{K}\sum_{q=1}^{K} d_{pq} f_{pq}}{\sum_{p=1}^{K}\sum_{q=1}^{K} f_{pq}} \tag{4.3}$$

where d_{pq} is ground distance between the two histograms h_{P_T} and $h_{P_{I;(X,Y)}}$. f_{pq} is flow between two histograms. p and q represents every bin in the histograms.
Figure 4.2 explains advantage of using EMD measure over the naive approach. The first image shows the example patch. The second and third images are vote maps for the same example patch. EMD is dissimilarity measure, so the lower (darker) the vote, more likely is the position of the patch. The naive surface has noise and is more blurred while the EMD is less blurred and smoothed. The experimental set up for this observation is as follows. Gray- scale image, with 16 bins for naive measure and 10 bins for EMD approach were used. Patches were taken from the region 30 pixels above or below upto 20 pixels to left or right from the example patch. Every sub- patch P_T around template centre (x, y) contributes its vote for the object‘s hypothesized position (X, Y). The vote maps from all template patches are combined to validate the object‘s new position.

The combined result is obtained for all possible hypothesized points around (X_0, Y_0). However, a single patch $P_{I;(X,Y)}$ that is occluded in frame I, shows drastic variation in similarity when compared with a static template patch P_T. The vote from this patch P_T, is the outlier and it gives a wrong estimate that would increase (dissimilarity measure) or decrease (similarity measure) the final result significantly. This effect is similar when the object undergoes appearance or pose change in the current frame. The algorithm uses Least Median of Squares type estimator that handles these outlier data. The vote maps from all patches around (x, y) are ordered and the combined vote $C(x, y)$ is obtained as in Eq (4).

$$C(x, y) = Q\text{‘}th \ value \ in \ sorted \ \{V_P(x, y)|\ patches \ P\} \tag{4.4}$$

For example if EMD metrics are used to compute dissimilarity between the patches, the set $V_P(x, y)$ when sorted in ascending order, ranges the patches that are minimally dissimilar (similar) to most dissimilar with its corresponding patches. The Eq (4) chooses Q‘th smallest score from the set, as a representative. Here, Q is maximal number of patches that is expected to give inlier value. For example, Q is 25% when it is assumed that every occlusion leaves (25%) of the target object visible i.e at least a quarter of the patches are visible and can be considered for valid voting.

In mean- shift approach during histogram generation, a kernel function assigns lower weight for pixels (in a patch) that are far away from object‘s center. These are the pixels which either belong in the background or would possibly give an outlier value. The FragTrack algorithm uses a weighted integral histogram data structure for this concern. Image I is represented as R. An inner region R_1 is marked that is closer to the image‘s center than the borders of region R. The pixel points falling between R and

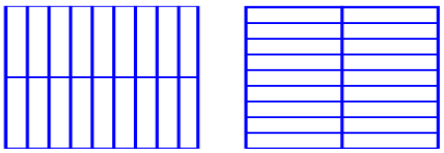


Abb. 4.3: Vertical and Horizontal patches used. Image source: Robust Fragments-based tracking using the Integral Histogram [2].

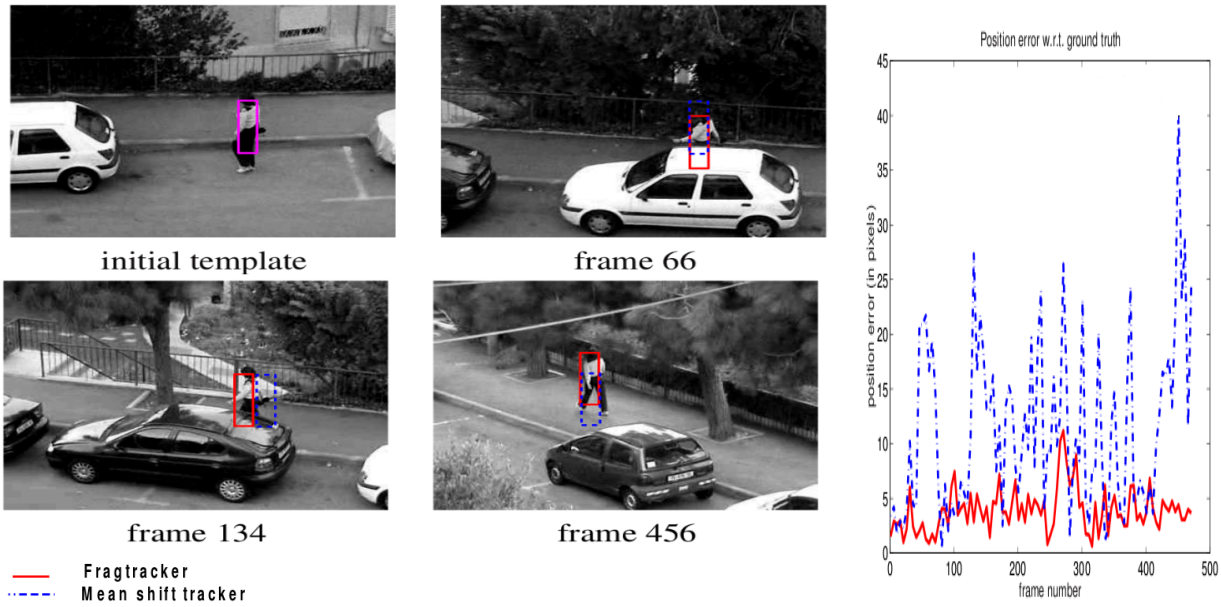


Abb. 4.4: First picture is the sequence of a female pedestrian walker. The second picture is the result graph of position error plotted against the frame number. Red plottings represents FragTracker and dashed blue is for mean shift tracker. Image source: Robust Fragments-based tracking using the Integral Histogram [2].

R_1 i.e $R - R_1$ are given less weight. Likewise an inner region R_2 is considered for R_1, so weight of pixels from $(R - R_1)$ $<$ weight of pixels from $(R_1 - R_2)$ $<$ weight of pixels from inside R_2.

Scale tracking is still an unsolved problem in this algorithm, however attempts were made to use a scale factor *scale* in the frame I to denote the hypothesis position (X, Y). If scale is added to hypothesis, corresponding image patch for the template patch is scaled in displacement vector, height and width by *scale* and is denoted as $P_{I;(X,Y,scale)}$. The template border is increased or decreased by 10% to check for that position and scale which gives a lowest score.

Experimental results Gray- scale images were used, the histogram contained 16 bins. 36 arbitrary patches were used as in Figure 4.3. The search radius s was 7 pixels. Value of Q was set to 25. The evaluation metric used was position error with respect to the ground truth that was marked every fifth frame. There were 6 experiment sequences from author's own and CAVIER database, as mentioned in their paper. The video frame speed is $5FPS$.

In the left image of Figure 4.4, in Frame 66 the object is occluded. FragTracker retraces the object in Frame 134 because of its support from upper part fragments while the mean- shift tracker fails as it uses only a single patch. Looking at the graph in the right side, at Frame 134 the position error for mean-shift tracker is around 27 pixels while only 5 pixels for FragTracker. In Frame 456, as the mean shift tracker does not maintain spatial information of the patches, it drifts away from the object. Whereas the FragTracker that considers spatial relationship between patches, recognizes that bright patch should be at upper part and dark patch at the lower part. Refer the graph for high position error by mean- shift tracker for that frame. The FragTracker could not handle out of plane rotation of the object as in Frame

134, when she enters the pavement. A slight drifting is also visible in Frame 456 because of the same effect.

3.2 Semi- Supervised Online Boosting for Robust Tracking [10]

This algorithm uses a different approach to track objects compared to the FragTracker. A Discriminative classifier is incorporated to distinguish the object from the background in every frame. For training a classifier, a supervised learning algorithm uses labelled data, while an unsupervised learner uses unlabelled data samples. Semi- supervised learning uses labelled samples to create a hypothesis function and unlabelled samples for improving the accuracy of the hypothesis. The discriminative classifier used in this approach learns the labelled samples using AdaBoost.

Algorithm 1 Adaboost

Input: Labelled training set $X_L = \{(f_1, l_1), (f_2, l_2), .., (f_{|X_L|}, l_{|X_L|})\}$ with $l_i = +1, -1$ where $i = 1, .., |X_L|$ and associated sample weights $W = \{w_1, w_2, .., w_{X_L}\}$

1: For iteration 1, Initialize weights $w_i^{(1)} = \frac{1}{|X_L|}$,
$W^{(1)} = \{w_1^{(1)}, w_2^{(1)}, .., w_{X_L}^{(1)}\}$ and train a weak classifier $h_m(f)$ using weighted data samples. Then proceed and continue from Step 4.

2: For iterations $m = 2, .., M$

3: Train a weak classifier $h_m(f)$ using weighted data samples, with current available weighting co-efficient $W^{(m-1)}$.

4: Estimate weighted error of this classifier on X_L,
$error_m = \frac{\sum_{i=1}^{|X_L|} w_i^{(m)} I(h_m(f) \neq l_i)}{\sum_{i=1}^{|X_L|} w_i^{(m)}}$.
where $I(A) = \begin{cases} 1 & \text{if A is true} \\ 0 & \text{if A is false} \end{cases}$

5: Calculate weighting co- efficient α_m for $h_m(f)$,
$\alpha_m = \frac{1}{2} \ln \left\{ \frac{1 - error_m}{error_m} \right\}$.

6: Update weighting co-efficient for data, $w_i^{(m+1)} = w_i^{(m)} \exp\{\alpha_m I(h_m(f_i) \neq l_i)\}$

Strong classifier $H(f) = sign\left(\sum_{i=1}^{|X_L|} \alpha_m h_m(f)\right)$

AdaBoost algorithm and Notations The input is labelled training dataset $X_L = \{(f_1, l_1), (f_2, l_2), .., (f_{|X_L|}, l_{|X_L|})\}$ where f_i is data point (feature), $l_i = +1, -1$ and $i = 1, .., |X_L|$. AdaBoost [8], produces a strong classifier $H(f)$ as a combination of weak classifiers $h_m(f)$ where $m = 1, .., M$, M is number of weak classifiers. Refer Algorithm 1. For the labelled training set X_L, $p(f_i)$ is the probability distribution for sample data set f_i. A weak classifier $h_m(f)$ discriminates data f with error less than 50%. All data samples are classified by series of weak classifiers till they are correctly classified. The weight of samples are updated, as in step (6), weights are increased for misclassified sample and are passed on to the next classifier. Based on α, from Step (3), probability distribution of f_i is changed.

Boosting can be used in feature selection. A strong classifier $H(f)$ can be combination of feature selectors where each feature selector is a weak classifier representing a feature. $H(f) = \{h_1^{sel}, h_2^{sel}, .., h_n^{sel}\}$. Every h_n^{sel} has a choice of m weak classifiers. With training data, each selector picks up a feature from the pool, and each selectors constitutes of $\{h_1(f), ..h_M(f)\}$ weak classifiers. Every selector $h_n^{sel}(f)$ becomes that weak classifier that gives minimum error in classifying f as in Eq (5).

$$h^{sel}(f) = arg \min_m \ error(h_m(f)) \tag{4.5}$$

The error of the selector is computed as, $error = \frac{\lambda^w}{\lambda^w + \lambda^c}$, where λ^w and λ^w are sum of correctly and incorrectly classified samples. Based on error of the selector, weight of the selector α_n and importance λ of the sample is increased and is passed onto next selector h_{n+1}^{sel}.

The Semi- supervised boosting algorithm uses labelled as well as unlabelled data samples for learning. Equations in AdaBoost algorithm are defined for supervised learners, but for SemiBoost tracker, while

computing the error of the weak classifier, all the data samples should contribute for the loss function. The extended loss function for the weak classifier $h_m(f)$, and the corresponding weight co-efficient for the selector h_n^{sel} is calculated as in Eq (6) and Eq (7).

$$h_n^{sel} = arg\min_{h_m}\left(\frac{1}{|X^L|}\sum_{f\in X^L, h_m(f)\neq l} w_m(f,l) - \frac{1}{|X^U|}\sum_{f\in X^U}(p_m(f)-q_m(f))\alpha_m h_m(f)\right) \tag{4.6}$$

$$\alpha_n = \frac{1}{4}\ln\left(\frac{\frac{1}{|X^U|}\left(\sum_{f\in X^U, h_n^{sel}(f)=1} p_n(f) + \sum_{f\in X^U, h_n^{sel}(f)=-1} q_n(f)\right) + \frac{1}{|X^L|}\sum_{f\in X^L, h_n^{sel}(f)=l} w_n(f,l)}{\frac{1}{|X^U|}\left(\sum_{f\in X^U, h_n^{sel}(f)=1} q_n(f) + \sum_{f\in X^U, h_n^{sel}(f)=-1} p_n(f)\right) + \frac{1}{|X^L|}\sum_{f\in X^L, h_n^{sel}(f)\neq l} w_n(f,l)}\right) \tag{4.7}$$

In Eq (6), the weight of labelled sample $w_m(f,l) = e^{-2H_{n-1}(f)}$. For the unlabelled data samples, the confidence measure of belonging to positive and negative class is given by $p_n(x)$ and $q_n(x)$ respectively. Refer Eq (8) and Eq (9) for formula. Both the labelled sample (first part of equation) and unlabelled samples (second part) contribute to error of classifier. $X^+ = \{\langle f_j, l_j\rangle | f_j \in X^L, l_j = 1\}$ is set of positive samples where f_j is input feature and l_j is class label of the feature and similarly, $X^- = \{\langle f_j, l_j\rangle | f_j \in X^L, l_j = -1\}$ is set with negative samples.

$$p_n(f) = e^{-2H_{n-1}(f)}\frac{1}{|X^L|}\sum_{f_j\in X^+} S(f,f_j) + \frac{1}{|X^U|}\sum_{f_j\in X^U} S(f,f_j)e^{H_{n-1}(f_j)-H_{n-1}(f)} \tag{4.8}$$

$$q_n(f) = e^{2H_{n-1}(f)}\frac{1}{|X^L|}\sum_{f_j\in X^-} S(f,f_j) + \frac{1}{|X^U|}\sum_{f_j\in X^U} S(f,f_j)e^{H_{n-1}(f)-H_{n-1}(f_j)} \tag{4.9}$$

In Eq (8), $S(f,f_j)$ computes similarity between the unlabelled feature data f and labelled samples f_j. A pseudo label and pseudo weight for the unlabelled data point is assigned as $z_n(f) = sign(p_n(f) - q_n(f))$ and $\lambda_n = |p_n(f) - q_n(f)|$.
For computing $p_n(f)$ and $q_n(f)$, following approximations are made. When $X^{|U|} \to \infty$, second term in Eq (4) and (5) becomes 0. Hence considering the first term, similarity between the new unlabelled data and labelled members of positive and negative class is to be computed. This can be done by a classifier $H^{sim}(f_j, f_v)$ that learns $S(f_j, f_v)$. A discriminative prior classifier $H^P(f)$ is built from labelled samples that distinguishes the positive class from the negative class. $H^P(f)$ is assumed equivalent to the classifier that describes the positive class. The new formulae for $p_n(f)$ and $q_n(f)$ are as in Eq (10) and (11) and are used to compute the pseudo label $\tilde{z}_n(f) = \tilde{p}_n(f) - \tilde{q}_n(f)$.

$$\tilde{p}_n(f) \approx \exp\{-H_{n-1}(f)\}\sum_{f_j\in X^+} S(f_j,f_v) \approx \exp\{-H_{n-1}(f)\}H^P(f) \approx \frac{\exp\{-H_{n-1}(f)\}\exp\{H^P(f)\}}{\exp\{H^P(f)\}+\exp\{-H^P(f)\}} \tag{4.10}$$

$$\tilde{q}_n(f) \approx \exp\{H_{n-1}(f)\}\sum_{f_j\in X^-} S(f_j,f_v) \approx \exp\{H_{n-1}(f)\}(1-H^P(f)) \approx \frac{\exp\{H_{n-1}(f)\}\exp\{-H^P(f)\}}{\exp\{H^P(f)\}+\exp\{-H^P(f)\}} \tag{4.11}$$

Semi-Supervised On-line boosting algorithm The tracker uses an adaptive appearance model such that it can handle appearance changes. The variations in the object's appearance is learnt online during the tracking process. The SemiBoost online adaptive tracker uses labelled data for building a prior classifier $H^P(f)$ and uses unlabelled data to construct an Online Boosting classifier $H(f)$. Locating the object is a combined decision of both the prior and Online boosting classifier. This is followed by updating the classifiers with samples from the current frame I. The samples for the prior classifier are taken from the first frame. In the successive frames, samples cut from object's position become the positive instance and sample chosen randomly from object's background belong to the negative class. Refer Figure 4.5 for a pictorial representation of SemiBoost tracker. Pseudo labels are assigned to unlabelled data which are used again for training the classifiers.

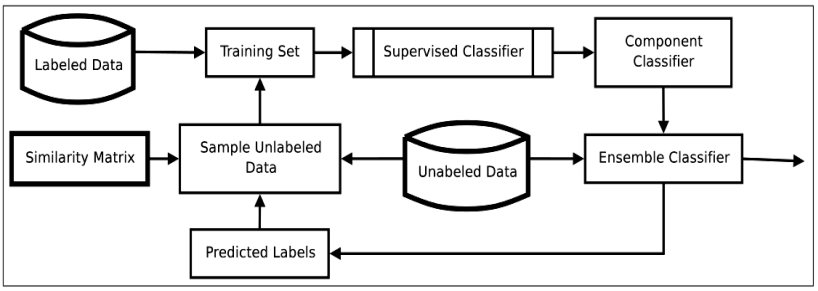


Abb. 4.5: An overview of Semiboost tracker. Image source: SemiBoost: Boosting for Semi-supervised Learning [16].

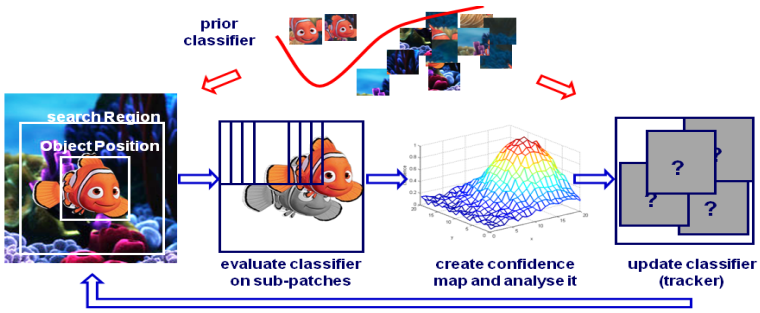


Abb. 4.6: Robust object tracking using SemiBoost Online classifier. Image source: Semi- Supervised Online Boosting for robust tracking [10]

Refer Figure 4.6. A fixed prior classifier from labelled samples and initial position of object in time t is considered. In frame $t+1$, the classifier exhaustively evaluates many possible positions in surrounding search region. Confidence map is obtained based on Eq (12). Here $H(f)$ means a classifier, which according to the algorithm is combination of both prior and Online Boosting classifier. The local maximum from the confidence map is chosen, the corresponding image sub- patch is the location of the object. Patches are randomly selected on and around the object.

$$H(f) = \frac{1}{2}\log\left(\frac{P(l=1|f)}{P(l=-1|f)}\right) \tag{4.12}$$

Consider Algorithm 2 that shows feature selection procedure using On-line Semi- Supervised boosting classifier. Assume, features are randomly assigned to selectors from the pool. Step (4) assigns label and importance for the labelled sample. The importance is obtained from previous selectors H_{n-1}. Pseudo label and importance for unlabelled sample is set in Step (6). In Step (9), for online classifier, weak classifier of the selectors are updated based on new training sets, and the error after the testing is calculated from Step (11) to (14). If the weak classifier identifies the label (or pseudo label) correctly then the importance of the sample adds to $\lambda^c_{n,m}$, else to $\lambda^w_{n,m}$. These two notations represent weight of the classifier based on how correct and incorrect they classify respectively. Step (16), calculates error of the classifier. The weak classifier with least error is chosen as the selector in Step (19). A weak classifier with $error = 0$ or $error > \frac{1}{2}$ is ignored i.e neither a classifier that is already over- fitted nor a classifier with error greater than 50% are of interest. Finally, in Step (23), weight co- efficient of the classifier is calculated. A strong online classifier is obtained from combinations of all N selectors. In Step (6) for assigning a pseudo label, the decision of either the prior classifier $H^P(f)$ or online classifier is considered based on their confidence value for f.

Drifting is a possible problem in adaptive learning. This is handled by the Online SemiBoost tracker as follows. In SemiBoost tracker, unlabelled samples taken out from region around object's new position updates the classifier. During decision making, even if the online classifier drifts away from the object due to over- fitting, the prior classifier monitors the decision. If the prior has high confidence score for the sample than the online classifier, then the tracker takes the result only from the prior thereby drifting from object is reduced.

Algorithm 2 On-line Semi-supervised Boosting for feature selection

Require: : training samples (labelled or unlabelled)
Require: : prior classifier H^P (can be initialised by training on X^L)
Require: : strong classifier H (initialised randomly)
Require: : weights $\lambda^c_{n,m}, \lambda^w_{n,m}$ (initialised with 1)
for $i = 1, .., |X|$ **do** {for all data samples, labelled and unlabelled}
  for $n = 1, 2, .., N$ **do** {for all selectors}
    if $f \in X^L$ **then**
      $l_{in} = l, \lambda_{in} = \exp(-yH_{in-1}(f))$ {l_{in} is label for sample i from selector n and λ_{in} is importance for sample i by selector n }
    else
      $l_{in} = sign(p(f) - q(f)), \lambda_{in} = |p(f) - q(f)|$
    end if
    for $m = 1, 2, .., M$ **do** {M is the iterations}
      $h_{n,m} = update(h_{n,m}, \langle f, l \rangle, \lambda)$ {update all weak classifier, $h_{n,m}$ is weak classifier with feature n under m^{th} iteration}
      {estimate errors}
      if $h^{weak}_{n,m}(f) = l$ **then**
        $\lambda^c_{n,m} = \lambda^c_{n,m} + \lambda_n$
      else
        $\lambda^w_{n,m} = \lambda^w_{n,m} + \lambda_n$
      end if
      $error_{n,m} = \frac{\lambda^w_{n,m}}{\lambda^c_{n,m} + \lambda^w_{n,m}}$
    end for
    {choose weak classifier with lowest error}
    $m^+ = arg\min_m(error_{n,m}, error_n = error_{n,m^+}, h^{sel}_n = h_{n,m^+})$
    if $error_n = 0$ or $error_n > \frac{1}{2}$ **then**
      exit
    end if
    $\alpha_n = \frac{1}{2} \ln\left(\frac{1-error_n}{error_n}\right)$ {calculate voting weight}
  end for
end for

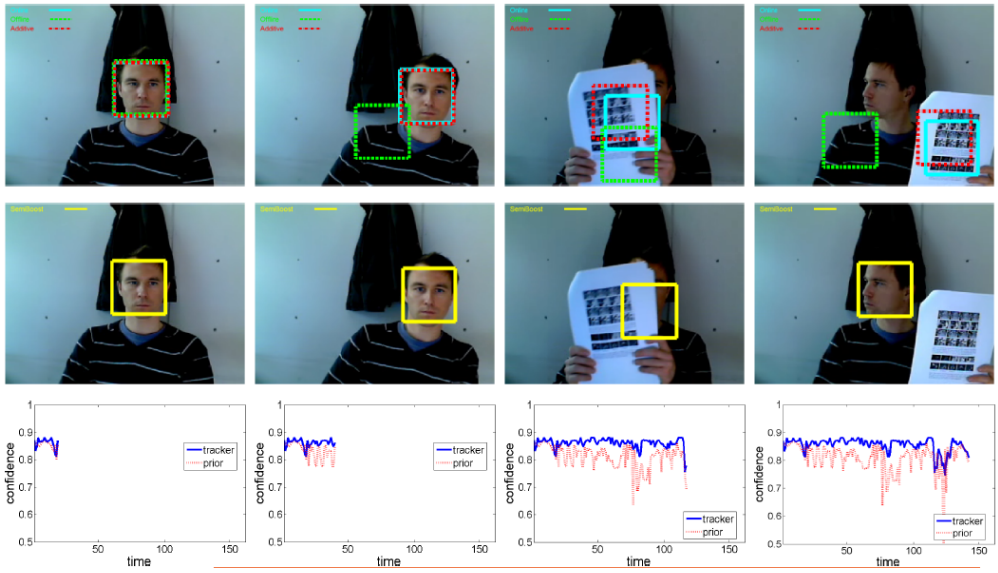


Abb. 4.7: Comparing four trackers. The cyan colour is the on-line boosting tracker, the red for the prior classifier, the green is the heuristic combination of both the above tracker as $0.5(H^P(x) + H(x))$. The yellow in second row is the Online SemiBoost tracker. The graph in last row is confidence values for On-line SemiBoost and Prior classifier measured against frames numbers. Image source: Semi- Supervised Online Boosting for robust tracking [10]

Experimental results The Online SemiBoost tracker was compared in performance with on-line boosting, prior classifier and heuristic combination of both, separately under various appearance changes. Image features are represented as Haar- like features [14]. A strong classifier comprises of 25 selectors ($N = 25$) and 50 weak classifiers each. Experiments were conducted on a 2.0 GHz PC with 2 GB RAM, with tracking speed of 25fps. The evaluation metric used was confidence values of the tracked objects. In Figure 4.7 notice in second frame that the prior classifier and the Online Boosting tracker performs well similar to SemiBoost tracker. But the heuristic combination of the first two trackers fails. When the object is occluded in third frame, the online boosting classifier learns from the occluded object and drifts away from the object in fourth frame. Similarly as the prior could not handle appearance change, it does not track the object correctly. While, the Online SemiBoost tracker uses the prior as reference and updates from previous frame to correctly locate the object in current frame. Last row is the graph showing confidence values of object measured for SemiBoost tracker and the prior while tracking.

3.3 Object Tracking with Online Multiple Instance Learning [3]

The Online SemiBoost tracker learns online with patches from the object and the background. The adaptive capability of the tracker helps it handle appearance changes. However, when the tracker is exposed to significant appearance changes, it is not shown to succeed. Even as several positive instances are given for learning, the classifier might face difficulty in selecting the best feature. Viola et al suggested using multiple instances learning (MIL) [15] to help the classifier in selecting the best feature. In MIL approach, examples are presented in labelled bags. An instance from the object is considered positive, whereas an instance taken from the background is negative. A bag containing at least one positive instance is labelled positive. A bag contains multiple examples. When the tracker learns from samples represented as bags, the classifier is more flexible to choose. With the bag, classifier is shown to handle situations with appearance change well as it has chance to select one prominent discriminative feature from the pool, that suits all instances in the bag. The learning algorithm should determine, which example in positive instance is correct.
Refer to Algorithm 3. The object‘s position in previous frame loc^*_{t-1} is known. In the new frame rectangular patches within a distance s pixels around the object‘s old position are cropped out as in Figure 4.8. The location of the image patch f is given as $loc(f)$. In Step (2), the classifier computes $p(l = +1|f)$ for every image patch $f \in X^s$. The f for which $p(l = +1|f)$ is high, i.e the local maximum of the distribution gives the current location of the object loc^*_t as in Step (3). In Step (4), samples are taken for

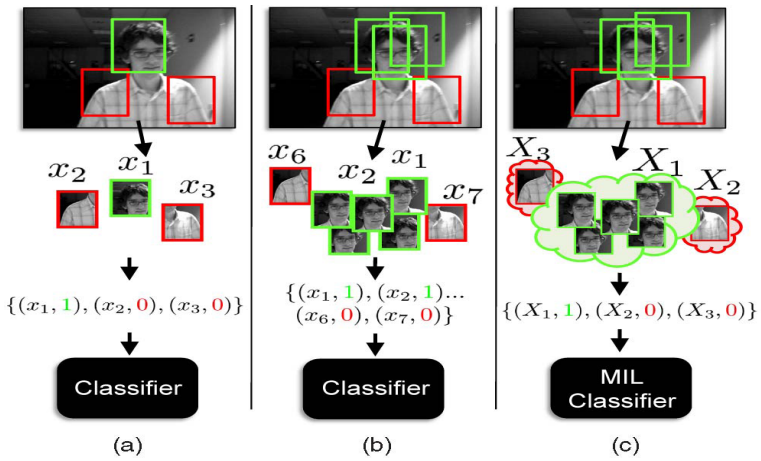


Abb. 4.8: Updating a discriminative appearance model: (a) Single patch to update the classifier. The patch captured may not cover the object completely. The system fails in handling appearance changes. (b) Several positive patches to update the classifier. Finding the best feature is difficult in this case. The system cannot handle significant appearance changes. (c) MIL approach of using a single positive bag with multiple patches for positive samples. Image source: Robust Object Tracking with Online Multiple Instance Learning [3].

updating the classifier. Positive bag contains patches cropped out within radius r pixels, with $r < s$ and the negative patches are taken out within radius β pixels.

The training set is $\{(X_1, l_1), (X_2, l_2), .., (X_I, l_I)\}$, where I is number of training sets and $i = 1, 2, ...I$. $X_i = \{(f_{i1}, l_i), (f_{i2}, l_i), .., (f_{iU}, l_i)\}$, where $U = |X_i|$ and $u = 1, ...U$. The labels of instances inside the bag is given the label of the bag. From Step (2) in Algorithm 3, the classifier calculates $p(l|f)$, hence labels for instances inside a bag are needed. Algorithm 4 explains online multiple instance learning using a boosting framework. The feature pool contains A features out of which K features are selected for the classifier. The A features are updated based on training data as in step (1). A strong classifier has K weak classifiers where each represents a discriminative feature. Step (3) is repeated to select K weak classifiers from pool of A classifiers using Steps (5) to (10). In Step (5), the classifier H_{iu} for $i = 1, 2, ...I$ and $u = 1, ., U$, computes the instance probability p_{iu}, by sequence of weak classifiers $h_a(f_{iu})$ for $a = 1, .., A$ using a sigmoid function, where $\sigma(B) = \frac{1}{1+\exp\{-B\}}$; B is just a parameter. Bag probability p_i is computed with all of its instance probabilities as in Step (6) using the Noisy-OR (NOR) model as in [15]. The classifier that maximises the log likelihood of the bag is chosen. Step (7) calculates that. The classifier learns all instances inside a positive bag as positive, this is suboptimal as there can be negative instances inside the same bag. The weak classifier k with $k = 1, ..K$ that maximises L^a is chosen in Step (9), (10) and is joined to the strong classifierH_{k-1}. The bags obtained during tracking, adds to the training set for updating the feature pool and weak classifiers in the strong classifier. In order to prevent the weak classifier from overfitting to new examples, the algorithm uses online weak classifiers that also retains information about previously seen data. Figure 4.9 illustrates the OMB algorithm.

Algorithm 3 MILtrack [15]

Input: Input Video frame number k

1: Crop out a set of image patches, $X^s = f : ||loc(f) - loc^*_{t-1}|| < s$ and compute feature vectors.
2: Use MIL classifier to estimate $p(l = +1|f)$ for $f \in X^s$.
3: Update tracker location $loc^*_t = loc(arg\max_{f \in X^s} p(l|f))$.
4: Crop out two sets of image patches $X^r = \{f : ||loc(f) - loc^*_t|| < r\}$ and $X^{r,\beta} = \{f : r < ||loc(f) - loc^*_t|| < \beta\}$.
5: Update MIL appearance model with one positive bag X^r and $|X^{r,\beta}|$ negative bags, each containing a single image patch from the set $X^{r,\beta}$.

The classifier learns online by updating the parameters of the weak classifiers. In their system, the authors represented the weak classifier by a Gaussian distribution. It has to update with four parameters

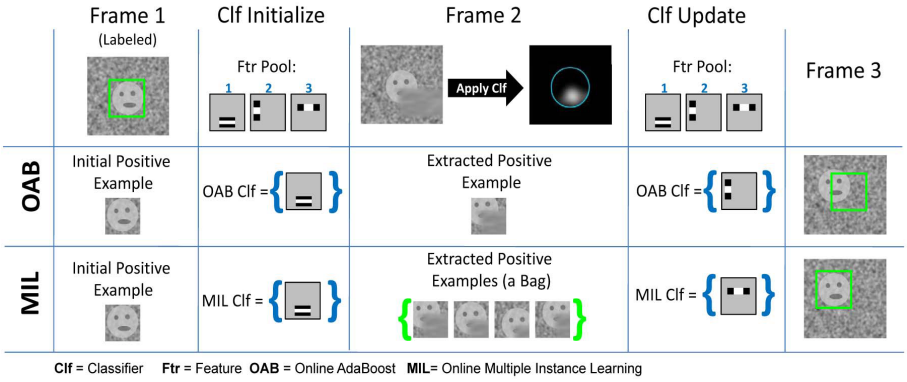


Abb. 4.9: The object is selected in first frame. Positive and negative patches around the object are extracted. For both OAB and MIL classifiers, a single and same feature is picked, $k = 1$. In Frame 2, when the object is occluded, both classifiers fail. The appearance model used before can no longer be used to track if this occlusion persists. OAB updates its classifier with a single positive example. It chooses one feature k that commonly represents samples in the past and current patch. The MIL tracker extracts many positive instances from the occluded patch and from its neighbourhood. The best feature that suits all four patches (say, the mouth) is selected from the pool to be the discriminator. In Frame 3, MIL tracks correctly and OAB drifts away. Image source: Robust Object Tracking with Online Multiple Instance Learning [3].

Algorithm 4 Online MILBoost(OMB)

Input: Data set $\{X_i, l_i\}_{i=1}^{I}$, where $X_i = \{f_{i1}, f_{i2}, ., f_{iU}\}, l_i \in \{-1, +1\}$.
For $u = 1, .., U$, $l_{iu} = l_i$.

Update all A weak classifiers in the pool with data f_{iu}, l_u.
Initialise strong classifier $H_{iu} = 0$ for all i, u.
for $k = 1, 2, .., K$ **do**
  for $a = 1, 2, .., A$ **do**
    $p_{iu}^a = \sigma(H_{iu} + h_k(f_{iu}))$ where p_{iu}^a is the instance probability.
    $p_i^a = 1 - \prod_u (1 - p_{iu}^a)$ where p_i^a is the bag probability.
    $\boldsymbol{L}^a = \sum_i (l_u \log(p_i^a) + (1 - l_u) \log(1 - p_i^a))$ where $\boldsymbol{L}$ is the likelihood function.
  end for
  $a^* = arg \max_a \boldsymbol{L}^a$
  $h_k(f) \leftarrow h_{a^*}(f)$
  $H_{iu}(f) = H_{iu}(f) + h_k(f)$
end for

Output: Classifier $H(f) = \sum_k h_k(f)$, where $p(l|f) = \sigma(H(f))$, and sigmoid function $\sigma(H(f)) = \frac{1}{1+\exp\{-H(f)\}}$

$(\mu_{+1}, \sigma_{+1}, \mu_{-1}, \sigma_{-1})$ using the unlabeled samples obtained online.

$$h_k(f) = \log\left[\frac{p_t(l=+1|f)}{p_t(l=-1|f)}\right] \tag{4.13}$$

p_t here is probability at current time frame t. Bayes formula, Eq (14) is used to compute $p_t(l=+1|f)$, after removing the consistent term $p(f,l)$ in the denominator.

$$p_t(l=+1|f) = \frac{p_t(f|l=+1)p(l=+1)}{p(f,l)} \tag{4.14}$$

$p_t(f|l=+1) \sim N(\mu_+1, \sigma_+1)$ and it is assumed that $p(y=1) = p(y=0)$. The mean and standard deviation of the distribution is updated using Eq (15) and (16).

$$\mu_1 \leftarrow \gamma\mu_{+1} + (1-\gamma)\frac{1}{|X_j|}\sum_{i;l_i=+1} f_u. \tag{4.15}$$

$$\sigma_1 \leftarrow \gamma\sigma_{+1} + (1-\gamma)\sqrt{\frac{1}{|X_j|}\sum_{i;y_i=+1}(f_u - \mu_{+1})^2} \tag{4.16}$$

The learning rate of the classifier γ with $(0 < \gamma < 1)$ is included in the above equations.

MIL algorithm for tracking scaled objects is also defined. The effective tracking of scaled objects by MIL tracker similar to Incremental visual tracker (IVT) is experimented and shown in the paper [3]. A parameter λ_{scale} is used as scale space step size. While cropping out patches in current frame for locating object, patches of current scale l_t^{scale} as well as patches with one scale step larger and smaller $l_t^{scale} \pm \lambda_{scale}$ are cropped. From these patches, the one with maximum likelihood is the object‘s new location.

4 Experimental comparisons and results

The trackers discussed in this seminar work are experimentally compared by Babenko et al in [3]. The experiment was conducted using 9 sequences, three were publicly available videos and the others were author‘s own clips. All videos posed challenges that brought out the efficiency of each algorithm. The common issues tested were robustness of the tracker against partial and full occlusions, background illumination changes and appearance changes.

Following are the parameter settings for all videos and trackers. The search radius $s = 35$ pixels, $r = 1$ pixel for OAB (that results in 1 patch) and 4 pixel for MIL tracker (resulting in 45 positive patches), $\beta = 50$ pixels (giving 65 negative patches). $A = 250$ out of which $K = 50$ features are selected, $\gamma = 0.85$. Ground truth was set at every 5 frames, for the other frames interpolation was done. The evaluation metric used was *center location error* and *precision plots*. The precision value of the classifier, gives the percentage of frames for which the predicted object location was within a threshold distance pixels from the manually set ground truth i.e if the object‘s location is predicted within 20 pixels from object‘s actual position, then that classification is still not considered as error.

Four challenging video snapshots are included in this seminar paper. The Occluded Face clip (Figure 10) that shows partial occlusion, the Cola Can sequence (Figure 11) that exhibits illumination changes, the Surfer video (Figure 12) needs tracking of an object that faces occlusion and appearance change. The Coupon Book clip (Figure 13) had appearance changes, in which after every 50 frames a page in the book is folded and an unfolded coupon book is introduced for distraction.

From the Table 4, the Online SemiBoost tracker has shown good result during occlusion scenarios. For instance in the Occluded Face clip, in 97% of frames the object is predicted at a location within a threshold of 20 pixels. The FragTracker has even performed well, for the same example with the precision of 0.95. MIL tracker handles this situation with precision of 0.43, still not better compared to the SemiBoost tracker. In case of coupon book that shows appearance change, MIL tracker outperforms SemiBoost and FragTracker when an unfolded book is introduced. With the surfer example, the SemiBoost tracker performs best followed by MIL tracker while the FragTracker completely lost the object from Frame 206.

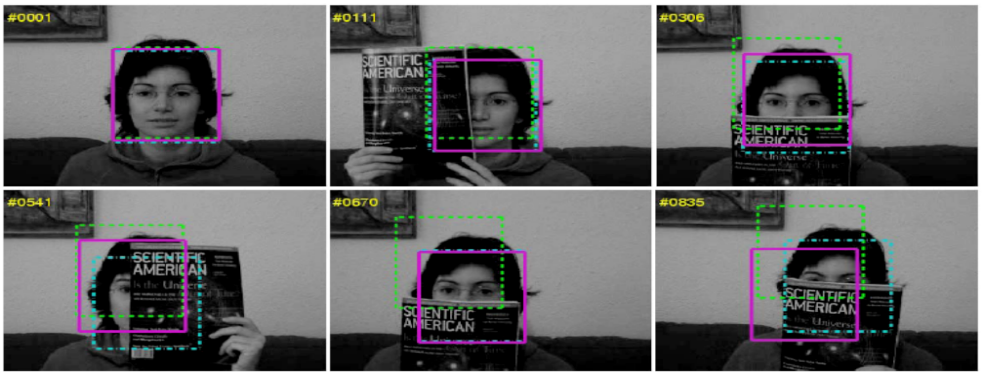


Abb. 4.10: Initial template marks the target object in Frame 0001. The OAB tracker is shown in dashed green, FragTracker in fine- dashed blue, SemiBoost tracker in dotted red and MIL tracker in pink. Refer Table 4 for measured precision values. The SemiBoost tracker handles occlusion well. This is because the tracker is trained online from unlabelled samples that are obtained in every current frame. The FragTracker also shows precision in 95 frames, as it uses a model (part- based) that uses combined votes from multiple patches before locating the object. The MIL tracker shows a precision of 0.43. It fails in competing with other two trackers as it uses an adaptive appearance model and adapts faster to changes in the new frames. Thereby it identifies the target in Frame 0835 based on discriminative feature obtained from Frame 0670 and consecutively drifts away from object. Image source: Robust Object Tracking with Online Multiple Instance Learning [3].

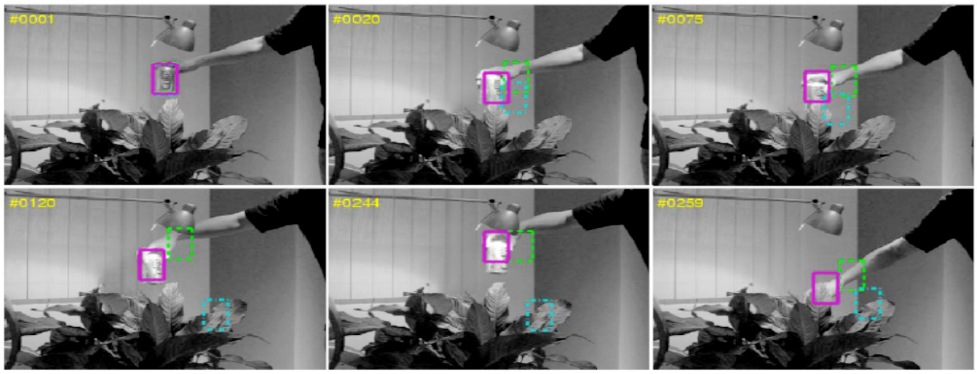


Abb. 4.11: Initial template marks the object to track in Frame 0001. The video sequence is challenging for the trackers as they need to locate an object that reflects light (specular object). The OAB tracker is shown in dashed green, FragTracker in fine- dashed blue and MIL tracker in pink. From the Table 4, the SemiBoost tracker has shown precision in 0.78% frames as it learns from the illumination change in Frame 0020. The tracker manages to track the target effectively in consecutive frames and at the same time learns in parallel. The MIL tracker shows a precision value of 0.55. It handles illumination changes well because of its adaptive appearance model and learning from the background instances of the target. The FragTracker could not handle the illumination change. It tries to track the target that looks similar as the template in Frame 0001 near its previous position and completely drifts away from object. Image source: Robust Object Tracking with Online Multiple Instance Learning [3].

Video Clip	OAB	SemiBoost	FragTracker	MILTrack
Occluded Face	0.22	0.97	0.95	0.43
Surfer	0.51	0.96	0.28	0.93
Cola Can	0.45	0.78	0.14	0.55
Coupon Book	0.67	0.37	0.41	0.69
Sylvester	0.64	0.69	0.86	0.90
David Indoor	0.16	0.46	0.45	0.52
Occluded Face 2	0.61	0.60	0.44	0.60
Tiger 1	0.48	0.44	0.28	0.81
Tiger 2	0.51	0.30	0.22	0.83

Tab. 1: Precision at a threshold of 20 pixels. Measurement: Every 5^{th} frame was labelled the remaining frames interpolated. The first and second best precision value are shown in bold. Source: Table from Robust Object Tracking with Online Multiple Instance Learning [3]

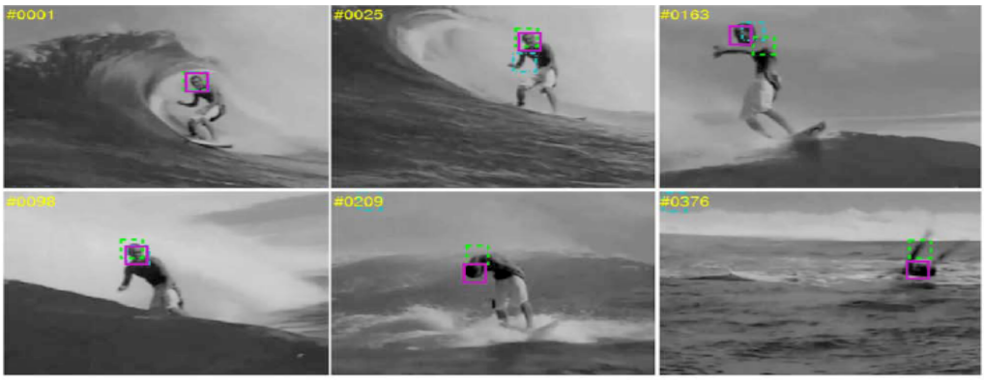


Abb. 4.12: Initial template marks the object to track in Frame 0001. The OAB tracker is shown in dashed green, FragTracker in fine- dashed blue and MIL tracker in pink. From the Table 4, SemiBoost tracker shows precision of 0.96 in tracking the object location. The MIL tracker also holds a precision value of 0.93. The algorithm tracks the target effectively in seen in Frame 0163 when there are changes in orientation and scale of the object. From Frame 0209, the Fragtracker loses its track on the target as the surfer bends. The system fails because of significant position change of the target object. The advantage of using adaptive appearance model is seen in Frame 0376 when the OAB tracker still manages to be close to the target. The MIL tracker identifies the location of the object in Frame 0376 with the help of multiple instances obtained from the object and the background in Frame 0209. Image source: Robust Object Tracking with Online Multiple Instance Learning [3].

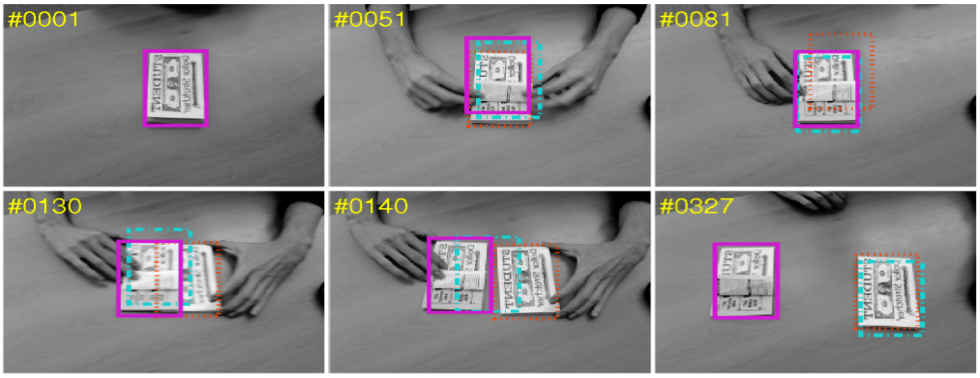


Abb. 4.13: Initial template marks the object to track in Frame 0001. The SemiBoost tracker is shown in dotted red, FragTracker in fine- dashed blue and MIL tracker in pink. This video clip is hard as it challenges the trackers to handle substantial appearance change. The page of the book is folded in Frame 0051. A new unfolded book is introduced in Frame 0130 to confuse the trackers. The MIL tracker copes up well with the appearance change from Frame 0051 till Frame 0327. The FragTracker still locates the book in Frame 0081 as the target is closer to its previous location. Patches taken from the upper part of the template object adds positive score for object‘s new position in Frame 0081. However when the new book is brought in Frame 0130, the static appearance model of FragTracker diverts the tracker to the new book thereby makes the tracker lose its track over the old unfolded book. In case of the SemiBoost tracker, in Frame 0130 the confidence measure for unfolded book will be higher because of the prior classifier than confidence value for folded book by Online Boosting classifier. So, the SemiBoost tracker scores only 0.37 precision value in this clip. Image source: Robust Object Tracking with Online Multiple Instance Learning [3].

The other clips experimented are also included in the table. The screen shots of tracking in these videos are shown in the author‘s paper [3]. The Occluded Face 2 video sequence shows appearance change as well as occlusion. The FragTracker is not designed to handle appearance change and hence shows precision only in 0.44% frames. The Sylvester and David Indoors exhibit lighting, scale as well as pose changes. The MIL tracker identifies the target object best in this clip. The Tiger 1 and tiger 2 sequences has frequent occlusions and blurred image because of fast motion of object. This video also had out of plane rotation of the tiger toy. MIL tracker tracked the object location with precision of more than 0.80 in both cases. The video tracking of all these sequences are available in Babenko‘s website [3].

5 Conclusion

Object tracking becomes challenging when the object moves faster relative to the frame rate, when it changes appearance and orientation over time. Three robust tracking systems were discussed in this paper. Every algorithm is defined to handle specific issue such as occlusion, appearance change, but fail to maintain robustness in other scenarios. The FragTracker algorithm is efficient during occlusions as it uses spatial relationships between template patches while tracking. However when the appearance of object changes, FragTracker loses the object as it uses a static appearance model. Objects that makes out of plane rotations are not traced well by the FragTracker. The Semi- Supervised Online boosting tracker handles occlusion very well, however loses object or drifts away completely on significant appearance changes. It makes use of unlabelled samples cropped from the scene to increase accuracy of the supervised learning based prior classifier. An Online boosting classifier learns from the unlabelled samples and final classification is the combined result of both the prior and OAB classifiers. The MIL tracker is useful during significant appearance changes and partial occlusion. It has good definition for tracking scaled object and has shown to perform well equal to the IVT trackers [12] that is specific for tracking scaled objects. The drawback for MIL tracker is that the system learns from wrong examples when the object remains occluded for longer time, and may lose the object completely. It takes considerably more time to re- trace the object. To conclude from the result in Table 4, the on-line SemiBoost tracker performs well in more than 50% of the examples, yet fails to efficiently handle the object‘s appearance change. The choice of one from those three trackers is highly based on the environment they should work in. However in real time, the circumstances the target object would face is not known in advance. A robust tracking algorithm that is efficient against all the problems discussed, is still an open issue.

Literaturverzeichnis

[1] D. J. Fleet A. D. Jepson and T. F. El-Maraghi. Robust on-line appearance models for visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 25, pages 1296– 1311, 2001.

[2] E.Rivlin A.Adam and I.Shimshoni. Robust fragments- based tracking using the integral histogram. In *IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 798– 805, 2006.

[3] Boris Babenko and Ming-Hsuan Yang Serge Belongie. Robust object tracking with online multiple instance learning. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, volume 33, pages 1619– 1632, 2011.

[4] A.O. Balan and M.J. Black. An adaptive appearance model approach for model-based articulated object tracking. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*, volume 1, pages 758– 765, 2006.

[5] A. Blum and T Mitchell. Combining labelled and unlabelled data with co-training. In *11th Annual Conference on Computational Learning Theory*, pages 92– 100, 1998.

[6] T. Broida and R Chellappa. Estimation of object motion parameters from noisy images. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 8, pages 90– 99, 1986.

[7] D. Comaniciu and P Meer. Mean shift analysis and applications. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1197–1203, 1999.

[8] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Computer and System Sciences*, pages 23– 37, 1997.

[9] H. Grabner and H Bischof. On-line boosting and vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 260– 267, 2006.

[10] C.Leistner H.Grabner and H.Bischof. Semi- supervised online boosting for robust tracking. In *Proceedings of the 10th European Conference on Computer Vision (ECCV): Part I*, pages 234– 247, 2008.

[11] M. Isard and A Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[12] R.S.Lin M.H.Yang J.Lim, D.Ross. Incremental learning for visual tracking. In *Advances in Neural Information Processing Systems 17*, pages 793– 800. MIT Press, 2004.

[13] M.Isard and J.MacCormick. Bramble: A bayesian multiple-blob tracker. In *Eight Proceedings IEEE International Conference on Computer Vision, (ICCV)*, volume 2, pages 34–41, 2001.

[14] H. Tao P. Dollar, Z. Tu and S. Belongie. Feature mining for image classification. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[15] J.C. Platt P. Viola and C. Zhang. Multiple instance boosting for object detection. In *Proceedings Neural Information Processing Systems*, pages 1417– 1424, 2005.

[16] A.K.Jain P.K.Mallapragada, J.Rong and Y.Liu. Semiboost: Boosting for semi-supervised learning. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, volume 31, pages 2000– 2014, 2009.

[17] Fatih Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 829– 836, 2005.

[18] B.Stenger T.Woodley and R.Cipolla. Tracking using online feature selection and a local generative model. In *Proceedings of British Machine Vision Conference (BMVC)*, 2007.

[19] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Computing Survey*, 38(4):13, 2006.

[20] C.Tomasi Y.Rubner and L.J. Guibas. The earth mover‘s distance as a metric for image retrieval. *International journal of Computer Vision*, 40, 2000.

Finding Central Path in Virtual Colonoscopy

Safoura Rezapour Lakani

Inhalt

Zusammenfassung

The seminar report concentrates on an optimization problem to find central path in virtual colonoscopy. Finding the central path is an useful method, that allows user to navigate through the colon image.

Keywords: Virtual colonoscopy, Central path

1 Introduction

This seminar report concentrates on an improvement in virtual endoscopy for complex anatomy such as colon. It focuses on finding the central path of it.
A central path is a path that passes through the center of the colon. It is useful because it provides an appropriate visualization for the user. Additionally a path that goes through the center of the object is more roboust to false branches.

2 Vitual Endoscopy for Colon

2.1 3D Model Construction

At the first step, the algorithm generates a 3D image data volume skeleton of a colon from CT scan. Which is based on the segmentation of the taken image. This step is done to make the graph construction of the image easier.

2.2 Thinning

To make the skeleton of the image, there is one step for thining. To be more precise, the ideal skeleton is the one that does not have branches since that can easily determines the central path. This is not the case in most of the time because there are always artifacts (as extra branches) which is caused by segmentation. And even with finer segmentations, these artifacts can not be completely removed. So there is a need for thining methods.

2.3 Removing False Branches

In this step a skeletal graph will be made. A skeletal graph is a connected graph which has just positive weights. In the skeletal graph, a vertex will be considered as a point where either of following two conditions will be satisfied. The first condition is that three or more branches join in the specified point. The second one is that the point will be the endpoint of the colon which means just one branch joins.
Since the central path lies at the center of the colon lumen which has quite large diameter, the weights should be determined in such a way that shows this property. The best way to imagine this property is to consider a ball that pass through this pipe. The radius of this ball can be chosen as the weight of the corresponding edge.

2.4 Smooth Representation

In this step of the algorithm, the aim is to make the representation of the central path smoother. To be more precise, the aim is to get more connected representation. Since later on the representation should be used to find the path, and that makes it easeier when there is smooth representation. Smoothing has been done by applying the B-Splines.

3 Finding the Central Path

Having the skeletal graph with given start and end points, the aim is to find a path between them such that its minimum weight is maximum among all the other paths and it does not pass through narrow passage as much as possible.
The algorithm tries to find the largest weight in each step. However, it causes two problems that should be considered.

3.1 Algorithm

In the algorithm snippet, d[u] represents the flow from s to u, $\Pi[u]$ indicates the predecessor of u in the current path, l[u] represents the number of edges from s to u in the current path through the Π attributes an a[u] indicates the average weightof the current path from s to u.

1. for each vertex$u \in V$ do
$d[u] \leftarrow -1, \Pi[u] \leftarrow NIL, l[u] \leftarrow 0, a[u] \leftarrow 0$
endfor
$d[s] \leftarrow 0, S \leftarrow NIL, Q \leftarrow V$
2. while $Q \neq NIL$ do
$u \leftarrow EXTRACTMAX(Q)$
if $d[u] \neq -1$ then $S \leftarrow S \cup u$ else goto step 3
for each vertex $v \in Adj[u]$ do
if $(d[u] < min\{d[u], w(u,0)\})$ then
$d[v] \leftarrow min\{d[u], w(u,v)\}$
$\Pi[v] \leftarrow u, l[v] \leftarrow l[u] + 1$
$a[v] \leftarrow \frac{(l[u].a[u]+w(u,v))}{l[v]}$
if($\Pi[v] \neq NIL$ and min $\{d[u], w(u,v)\} \geq d[v]$) then
if($\frac{l[u].a[u]+w(u,v)}{l[u]+1} > a[v]$) then
$\Pi[v] \leftarrow u, l[v] \leftarrow l[u] + 1$
$a[v] \leftarrow \frac{(l[u].a[u]+w(u,v))}{l[v]}$
endfor
endwhile
3. if $t \in S$ then output the path from s to t using the Π attributes
else there is no path from s to t

The algorithm initialized by giving negative flow to all the vertices apart from the start node. The flow of the start node will be set to zero. In the algorithm snippet l[u] shows the flow of the node u.
The algorithm tries to find a node that has the maximum flow by using EXTRACTMAX, which is the start node at the beginning. Then it checks the neighbors. For each neighbor, if it has not been visited before, its flow will be set accordingly, otherwise if the current path has the flow at least as large as the old one and it has got the strictly larger average weight than the old path is flipped over to the current path.
The foundation behind considering the flow of the path in each iteration is when a false branch has larger weight than true branches on a particular segment. This problem arises from the segmentation limitation. The better segmentation and thinning algorithms are more time consuming. So, to solve the problem, the maximum flow of a path should be considered. It means, if a vertex is in more than one paths, the one that has the largest weight to the specified vertex should be selected.
So, a simple path between two vertices will be selected, if the specified path has the maximum flow in comparison to other paths going through the vertices. To be more precise, if we consider a path p as $p = (v_0, v_1, ..., v_k)$, the flow of p will be described as following;

$$f(p) = min(w(v_{i-1}, v_i) : 1 \leqq i \leqq k) \tag{5.1}$$

Then p will be considered as the largest path between start node u and end node v, if for all the other paths from u to v namely $p^{'}$ the following two conditions will be satisfied,
$f(p^{'} \leqq f(p))$
if there is a vertex shared by both paths, then $f(u \rightsquigarrow^{p^{'}} x) \leqq f(u \rightsquigarrow^{p} x)$ The reason to consider the average weight is when there are more than one largest path between start and end points. This happens when for example a colon image contains some segments that are narrower than other segments. Then the narrowest segment may also include in the central path, if after this narrow segment, there will be a larger segment. In this case a false path may also have the same flow as the flow of the central path that contains narrow segment. That is because the maximum flow of all paths are already small.
In the ideal situation, if at any points in the path, all false branches have smaller weights than the true branch, then the central path has the heaviest average weight.

$$W(p) = \sum_{i=0}^{k-1} \frac{w(v_i, v_{i+1})}{k} \tag{5.2}$$

A heaviest path may contain an edge with very small weight. Hence, the aim is to find a path that has the heaviest weight among the largest paths. To this aim the critical path of the skeletal graph will be considered. The critical path from u to v is a the largest path between them and for all other paths the average weight of them will be less than or equal to the average path of p.

3.2 Complexity

The complexity of the algorithm depends on how the maximum path will be extracted is either $O(n^2)$ or $O((n + |E|)logn)$, which $|E|$ shows the number of edges, if $|E| = O(n)$ the algorithm can run in $O(nlog(n))$ time. There are two ways of implementation based on usual priority queue or binary heap based priority queue, which will be counted in following.

3.2.1 Priority Queue Based Implementation If the algorithm will be implemented by priority queue to examine each vertices, there is initially V-S , vertices in the queue, such that S shows the number of starting nodes in each step of the algorithm. Extacting the maximum from the queue takes O(|V|-|S|) because it is implemented based on priority and so it can be linearly obtained. This step must be done maximum for each vertices, so the total complexity till this general step is $O(|V|^2)$. Since there is maximum |E| adjancies for each vertices, and the maximum in each neighborhood can be extracted based on priority queue with O(1), the complexity of whole the algorithm will be $O(|V|^2 + |E|)$ which yields $O(|V|^2)$.

3.2.2 Binary Heap in Priority Queue If the priority queue will be implemented by binary heap, extracting the maximum in each step takes O(log(V)). Since this step must be done for each vertices, there are |V| of this execution. And since in each step all the neighbors should be examined, the total complexity of the algorithm will be O((|V|+|E|)log(|V|)). If |E| = O(|V|) , the total complexity will be O(|V|log|V|). This case will be happened since in a skeleton graph, |E|=O(|V|) .
There is also better implementation of priority queue based on Fibonacci heap, which yields O(|V|log|V|+|E|).

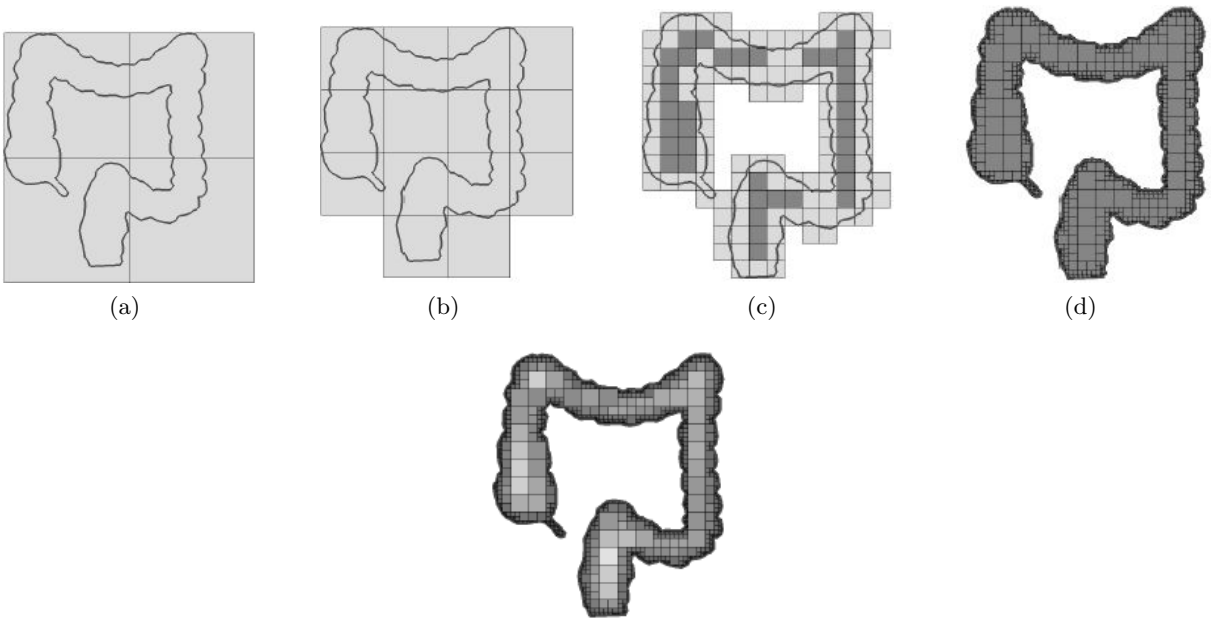
4 Other Methods

4.1 An Efficient Central Path Algorithm for Virtual Navigation

In [PCK] , path computation is done fisrt by subdividing the image. In this method first the image will be hierachically subdivided and for each block the distance from boundary will be computed. This will be later used to compute a path that stays away from boundary as much as possible. Finally, the central path will be computed.

4.1.1 Subdivision At the beginning, there is just one box surrounding the object of interest. Then the subdivision proceeds in such a way that the blocks which are completely either inside or outside the object will not be divided anymore. At the end of the subdivision, the blocks that lie on the boundary are the size of one. However, subdivision stops at the user defined threshold.

4.1.2 Distance from Boudary Estimation In this step the distance of each subdivided block from boundary are computed that later on will be used for finding the path which is far from boundary. To this aim, a graph kind of structure will be assumed. In this graph, there will be edge between each two blocks which intersect in at least one point. The length of this edge will be the summation of two blocks sizes, and if a block intersects with boundary, the length of the associated edge will be as the size of the block itself.



(a) (b) (c) (d)

(e)

Abb. 5.1: Hierachical Subdivision and DFB Fields
[PCK]

Distance from boundary for each block will be assigned as the shortest path between each block and boundary. To be more precise, it will be the path between the specified block and nearest boundary point. This path will be computed using Dijkstra single source shortest path algorithm. Figures 5.1 shows the procedure. The darker blocks have less distance to the boundary.

4.1.3 Computing Central Path After establishing the distance from boundaries, user specifies the source and destination points. First, weights will be assigned to each edge. The weight for the edge between blocks b_1 and b_2 will be defined as following:

$$w(b_1, b_2) = \frac{1}{dfb(b_1)} + \frac{1}{dfb(b_2)} \tag{5.3}$$

where dfb shows distance from boundary.
In the next step, the blocks that source and target points are will be identified. Then Dijkestra algorithm will be utilized to compute the shortest path between source and destination according to the computed weights. The answer to the user query, is the path $s, v_0, v_1, ..., v_l, t$ where v_i is the center of the block. Figure 5.2 shows path computation procedure.

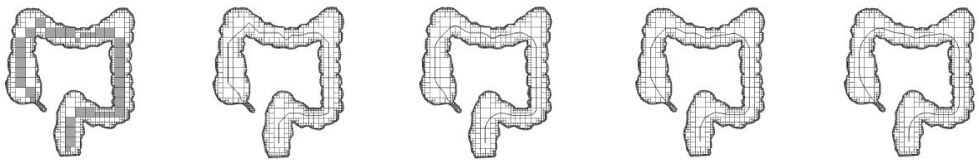


Abb. 5.2: Path Computation
[PCK]

4.1.4 Progressive Path Computation In central path computation, the entire 3D object will be subdivided in order to find central path between specified source and destination points. However in many cases, entire subdivision is not necessary. That happens when the source and destination are close to each

other in comparison to the size of the object.

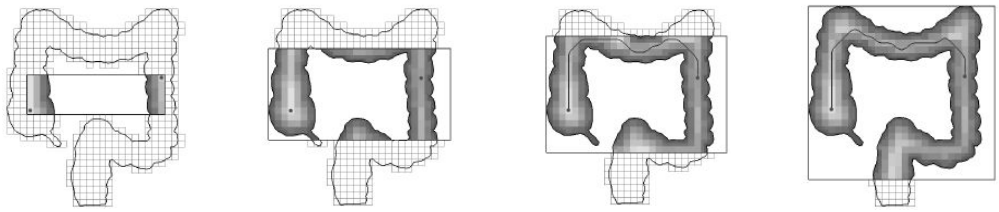


Abb. 5.3: Progressive Path Computation
[PCK]

In this method, the smallest box surrounding source and destination will be considered. The subdivision will be done in this region of interest and distance from boundary and later on central path will be computed. In the next iteration the ROI will be expanded. This process continues either if we do not have any central path or if the difference between path lengths in subsequent iterations will be greater than a specified threshold. Figure 5.3 shows how path will be computed iteratively.

4.2 Fast Extraction of Minimal Paths in 3D Images and Applications to Virtual Endoscopy

In [TD01] approach, the problem of finding central path is formulated as contour energy minimization. In this approach, the energy is defined as ;

$$E(C) = int_{\gamma}\left(w + P(C(s))\right) ds \tag{5.4}$$

In this equation, C shows the curve, P is the potential, w is the constant to control the smoothness of the contour. γ shows the boundary of the integral which is between 0 and the length of the curve.
The surface of minimal action between each point p_0 and p will be defines as following;

$$U(p) = inf_{A_{p_0,p}} E(C) = inf_{A_{p_0,p}} p\ (C(s))ds \tag{5.5}$$

$$p\ = p + w \tag{5.6}$$

A shows all the paths between specified points.

4.2.1 Fast Marching Algorithm The algorithm will be started by considering three sets of points, namely alive, trial and far points. Alive points are the points for which action values are computed and will not be changed. Trial points are those for which the estimation of minimal action is given and are in the neighborhood of the specified point. finally, far points are those that the minimal action is not yet estimated for them.
At the beginning, the estimated action - U - is 0. And for the set of trials the trial point which gives the minimum action will be considered as the alive node.

Abb. 5.4: Partial front propagation with colliding
[TD01]

This procedure will be done for all the points. To find out the path between start and target point,

gradient descent method from target node will be applied, till it reaches to the stat point or one of its closest neighbors.
The other improvement in [TD01] approach, is to compute the path simultaneously between start and target node. In this way, the minimal action from start point and the minimal action from end point will be computed; the conflict node between them is the stop point, in which the computed paths will be joined together. In this method, the computation will parallelized and it can also be divided in two processors. Figure 4.2.1 shows the digital subtracted angiography image. Colliding path technique is applied in this image.

Abb. 5.5: Comparing classic and centered path
[TD01]

To find the path which is near the center and far from the boundaries as much as possible, another potential will be defined as the difference of distance from nearest boundary point and a pre defined threshold. If this distance is less, the potential will be considered as zero. This step will be done after finding the minimal energy based on the original potential. Figure 5.5 left shows the two possible paths in the image. The middle figure is the classic path and the right one is the centered path by utilizing the new potential.

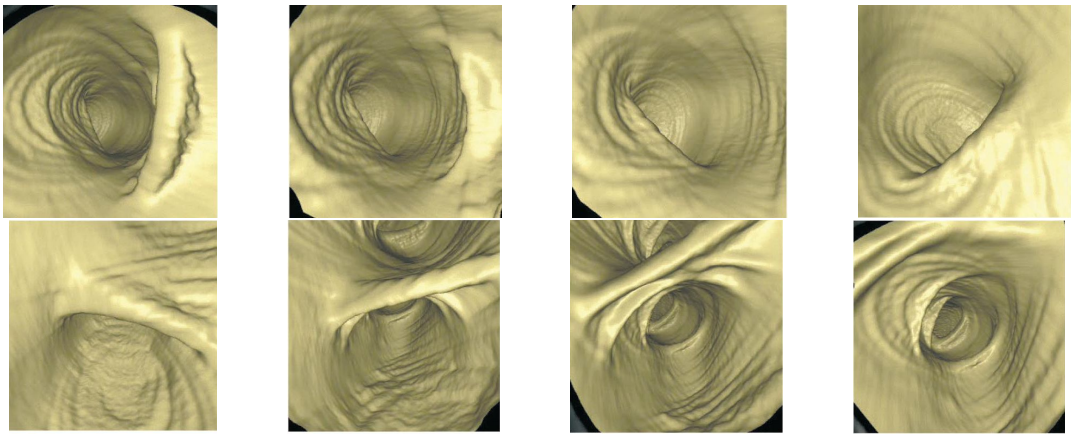


Abb. 5.6: Virtual endoscopy in the colon
[TD01]

5 Conclusion

5.1 Summary

In this seminar report the method for finding central path in virtual endoscopy especially in colon endoscopy was reviewed. In [JW] , the image will be skeletonized and based on that a graph will be created. The graph has either three or one branches. The weights of the edges are computed based on the radial thinning of it. The algorithm aims to find the path which its maximum flow is minimum among other paths. In this method the average weight is also considered in order to prevent from the paths that go through large weights related to false branches.

In [TD01] the problem formulated as active contour energy minimization, by defining the potential for the specified curve of the 3D image and minimizing the energy of the curve based on the snake model. The other concept in this method is the minimal action between points which is defined as the potential of the each path in the curve related to points. The algorithm consists of two tracks. At the first track, the minimal actions will be computed started from the initial point. In the second track which will be started from the endpoint, the best path will be computed based on gradient descent of the minimal actions. In [PCK] central path estimation is divided to hierarchical process, in which the image will be subdivided into blocks and the distance from boundary will be computed for each block. Giving start and end points, the algorithm tries to find a path between blocks by giving block centers as the result.

5.2 Comparison of Methods

In [PCK] the number of subdivision is not determined and it is an user specified parameter, however [JW] is not user dependent and it makes it better approach in this context.
In [TD01] whole of the image is used for computation, however, in [JW] just the skeleton image is utilized which is more efficient. Although, in [TD01] , there is an efficient approach to find the central path simultaneously between start and end point, which causes less computation time.

Literaturverzeichnis

[JW] Yaorong Ge Jie Wang. An optimization problem in virtual endoscopy.

[PCK] Deepak Sethi Parag Chaudhuri, Rohit Khandekar and Prem Kalra. An Efficient Central Path Algorithm for Virtual Navigation.

[TD01] Laurent D.Cohenb Thomas Deschamps. Fast extraction of minimal paths in 3D images and applications to virtual endoscopy. 2001.

Active Shape Models for Medical Images Segmentation

Tonima Mukherjee

Inhalt

1 Introduction

The following report discusses about the application of Active Shape models on flexible objects, here being the 2D image of the heart obtained during echo cardiogram. A number of rigid models exist for the location of objects that do not vary in appearance. However, in the field of Medical imaging, the objects vary over time and hence active shape models are used. The Active shape model consists of a flexible shape template which describes on how the points on the object (heart) can vary. It involves the construction of appropriate model by learning the variability based on changes of parameters in labelled images. We discuss the building of model for the heart and also discuss upon a way to construct the model for a 3D image of heart.
Model based vision enables the recognition of objects which do not vary in appearance. However, in practical situations there is a necessity for the models not to be rigid as certain objects in the medical field such as the shape of the internal organs like brain, abdomen and heart changes over a period of time and varies from person to person. A number of flexible models or deformable templates have been proposed to recognize the object. However, these existing models lose the model specificity. The term model specificity stands for the deformation of a model in accordance to the class of objects it represents. Feature extraction of structures from medical images is complex. The organs in the human body vary in shape, appearance and size. It is essential for the medical diagnosis that these organs be recognised properly in terms of the shape, size and appearance. This helps the doctors to notice any malfunctioning. The proper functioning of the Left ventricle is an indicator of the overall cardiac function, hence being important for cardiovascular diagnosis. The accurate location of the left ventricle boundary leads to the calculation of ventricular volume, motion and cardiac output. There is a lack of flexible template models for matching such complex biological structures like heart. The search strategies used to locate instances of the template within the image are also problem specific. Automated localization of these biological parts can be less labour intensive and can provide a more accurate image interpretation. However, the varying features of the anatomical parts and imperfect scanning procedures hinder this process. In the following report, we make use of the Point Distribution Model where the labelled points indicate the position. It gives the option of choosing the values of the parameters so that the model fits best to the image. In this report, Section 1 discusses the background and the previous work. In the next section, we discuss the need for a good model. In the following section, an overview of the Active Shape Model is provided. In the section following this, the Point distribution model is elaborated upon. The fifth section elaborates on an example model relating the heart. The next section explains the Active Shape Model. The seventh section gives a detailed account of obtaining the Initial guess of the parameters. The following section explains the use of this model to locate the ventricle in a 2-D image of heart followed by the discussion. The last section provides the conclusions.

2 Background

A number of flexible models have been used which use parameters to control the shape of the object. Hinton, Williams and Revow [9] describe a spline snake which has certain control points in the home locations. Deformation is achieved by moving the points away from the home locations. This models the average shape of the object but however fails to define the models of the shape variation. The model deforms in shape, but not within the legal constraints defined by the training set.

2.1 Rigid Models

The rigid models are based on the geometric properties of the visible surface of the objects [15]. A 2 dimensional model is usually created from a set of prototype objects which are viewed from different angles. The feature extraction involves edges, lines and corners and matching is done by sampling. These models are not suitable for the objects that vary in appearance over time as a suitable prototype can not be created. Hence, it cannot be applied to biological parts like heart as they vary individual to individual and also over different stages of the heart cycle.

2.2 Flexible Models

Detection of features using deformable templates has been discussed by Yuille *et al.* [16] and by Lipson *et al.* [13] where templates have certain parameters thus giving knowledge about the shape of the objects.

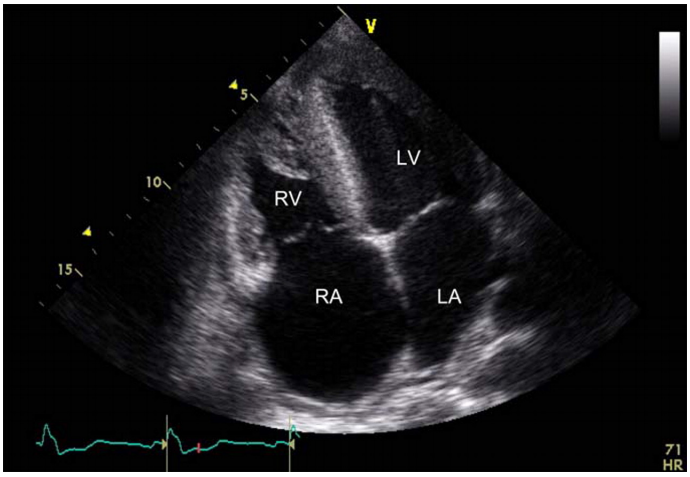


Abb. 6.1: Echo cardiogram of Heart Source: BMJ Case Reports-Google images

They have various degrees of freedom and can change with respect to the scale, orientation and rotation. A continuous variation of the parameters provides the good fit to the image. The peaks and valleys in the image intensity are noted and the sudden change in the image intensity are the features to which the template is compared. The energy function is defined. The template interacts with the image evaluating this energy function and modifying the parameters accordingly to until the least energy function is obtained. The minimum energy function means the best fit. The parameters of the template are updated by the steepest descent. However, these models are domain specific, that is hand crafted and cannot be generalized to other applications. Templates have to be created for every structure individually.

2.2.1 Finite Element Models These deformable models are considered to be clay with elastic properties as described by Pentland and Sclaroff [2]. The desired shape can be obtained by pushing, pulling the elastic material. The vibration modes of a particular shape is obtained. For example, the lowest frequency modes are the modes of translation and rotation. The next lowest mode are for the body deformations leaving the centre of mass and rotation fixed. Shapes having similar dimensions are of lower order modes. For example, for the human head, the first thirty lower order modes are used which stand for twisting, shearing, bending and scaling. These deformation modes are used as they are most stable to noise. The mode values are iteratively compared to that of the models to find a match.
An alternative model is suggested by Terzopoulos and Metaxas [5]. The parametrized model is based on superquadrics. However, this model does not support enough degrees of freedom to elaborate on the finer surface details. Also, this process requires a lot of iterations making it slower.

2.2.2 Active Contour Model Active contour models help in finding the continuous edge in an image by using stiff elastic strings called snakes. Kass *et al.* [12] describes these snakes being the deformable model slither while minimizing the energy are also known as energy minimizing spline curves. They are attracted towards edges, lines of images based on the external forces. From the initial state, the contour moves under the influence of the local forces, here being the gradient derived from the image until equilibrium with the contour's internal forces, elasticity and stiffness is reached. The snakes take the shape in conformance to the nearest neighbour. The snake is attracted to an edge where the energy is at minimum and locks to it. The internal forces make the contour smooth and the external forces deflect the snake towards a high gradient region of the image, that is the edge. Karaolani *et al.* [10, 11] provides an alternative approach to energy minimization.
In the above, the Active Contour Model performs the image search based on gradient climbing technique. However, an alternative is provided by Hinton *et al.* [9] where in the image is marked with a number of control points. In our example, 96 points are marked on heart and are initially in their home locations. The deformations in the iteration process is achieved by moving the control points away from these home locations. Moreover, the modes of shape variation are completely dependent on the number of control points and their initial positions. In case of the echo cardiogram of the heart, the model obtained differs depending upon the number of control points chosen initially.

3 Need for a Good Model

Image search and recognition has three issues [15]:

1. There exists a problem to identify which features need to be extracted to identify the object and their spatial relationship.

2. We encounter the problem of the way in which the features need to be combined to make a model such that it is sufficient to recognize all the objects of the particular class.

3. The technique to identify how the matching has to be done between the model and the image.

4 Overview of Active Shape Model

The question to be addressed is why there is a need for a model when a number of models with deformable templates exist. The need for an Active Shape model arises from the fact that the earlier models do not address the problem of model specificity. The model should deform to represent an object which is part of the class to be recognized.
Given an object, its boundaries are represented by a set of points. In the training set, the points are put in the same locations. The next step involves the alignment of these points to reduce the variance in the distance between the equivalent points. Based on this, the Point Distribution Model is obtained. This model provides the average positions of the points. The parameters needed to control the modes of variation of the training set are also obtained.
Interpretation is done by choosing the appropriate values of these parameters to obtain the best fit in terms of scaling, orientation and shape. A rough guess can be obtained by comparing the image to the model instance.

5 Point Distribution Model

The reason why the Point Distribution Model is chosen is because it captures the variability in the shape of the object (the heart) and also their spatial relationships. The construction involves labelling of the points indicating position on the object. A set of these points form the object description upon which the model is to be trained. The principle eigenvectors of the matrix give the modes of variation of the training set. The resultant model consists of small set of parameters acting as weights. These weights when manipulated give new instances of the object (heart). Here, we obtain a Point Distribution Model of the left ventricle, the septum and the mitral valve [3].

5.1 Labelling the Object

As mentioned earlier, the model is represented by a set of points. Each shape in the training set is labelled either representing the boundary or the internal parts. For example in Figure 1, the echo cardiogram of the heart is labelled by 96 points on the boundary walls of the Left Ventricle, the Right Ventricle (the septum) and the Atrium (mitral valve). Comparison of the points when the shape varies is the key.
The landmark points are defined where the analysis of biological organs are effective by recording of the geometric locations of these landmark points [3]. They have the same locations in every form of study of the data. The landmark points are defined as follows:

1. Points labelling the application-dependent parts of an object like the sharp corner of a boundary.

2. Points labelling the application-independent parts like the highest point of an object in a particular direction.

3. Points which label the other parts.

For the echo cardiogram as shown in Figure 2, the points labelled at the apex of the Left ventricle, the lower two extremes of the Left Ventricle can be easily identified and hence are the points confirming

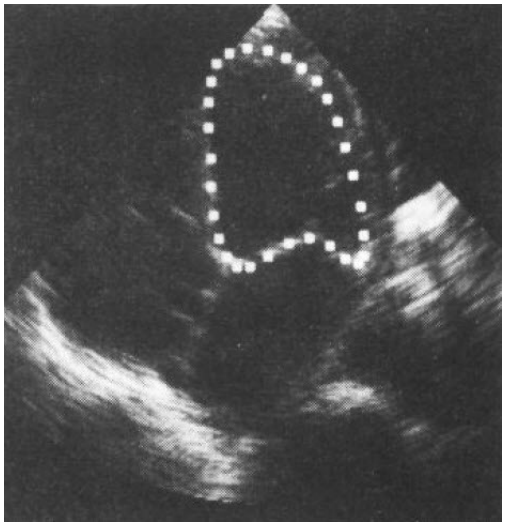


Abb. 6.2: Labelled points on boundary of Left Ventricle [1]

to the first type. The points labelled on the boundaries of the Left Ventricle along the septum on the right and the mitral valve below are equidistant and hence of the type 3. All the points are necessary to describe the boundary of the shape in detail. If sufficient number of type 3 points are labelled, they can help describe the curves with a greater accuracy and hence are more computationally efficient and effective.

5.2 Alignment of the Shapes

The alignment of the labelled points is necessary to compare the points among the different shapes. Alignment involves rotation, scaling and translation in order to minimize the weighted sum of the square of the distances.
Let us consider the alignment of a pair of shapes.

x_i : A vector describing n points of the ith shape in the set

$$x_i = (x_{i0}, y_{i0}, x_{i1}, y_{i1}, ..., x_{in-1}, y_{in-1})^T$$

$M(s, \theta)[x]$: Rotation by θ and scaling by s

For two shapes x_i, x_j, θ_j, s_j, translation (t_{xj}, t_{yj}) mapping on x_i on $M(s_j, \theta_j)[x_j] + t_j$ are chosen to minimize the weighted sum.

$$E_j = (x_i - M(s_j, \theta_j)[x_j] - t_j)^T W (x_i - M(s_j, \theta_j)[x_j] - t_j)$$

where $t_j = (t_{xj}, t_{yj}, ..., t_{xj}, t_{yj})^T$

W : Diagonal matrix of the weights for each point

$$M(s, \theta) \begin{bmatrix} x_{jk} \\ y_{jk} \end{bmatrix} = \begin{pmatrix} (scos\theta)x_{jk} - (ssin\theta)y_{jk} \\ (ssin\theta)x_{jk} + (scos\theta)y_{jk} \end{pmatrix}$$

The weight matrix

$$W_k = \left(\sum_{l=0}^{n-1} V_{R_{kl}} \right)^{-1}$$

R_{kl} : Distance between the points k and l in a shape

$V_{R_{kl}}$: Variance in this distance over various shapes

W_k : Weight for k^{th} point

For the points which move the least with respect to the the other points in a shape, a large weight is assigned, thus giving higher priority. Lower weights are allotted to the points which move a lot. Following is the algorithm to align a set of N shapes [3]:

- Align each shape to the first shape in the set by rotation, scaling and translation.
- **Repeat**
  - Calculate the mean shape from the aligned shapes.
  - Normalize the orientation, scale and origin of the current mean to suitable defaults.
  - Realign every shape with the current mean.
- **Until** the process converges.

Normalization is needed to ensure that the algorithm converges. It is to be noted that the mean shape is normalized and the rest of the shapes are aligned to this rather than every shape being normalized. Consider the normalization of every shape based on scale such that the distance between the two points is one unit. Artificial correlations might be forced on the set distorting the model. If alignment is with respect to the mean, then every shape will have similar scale, that of mean and the landmark points are chosen which match the mean. Testing of the convergence condition is by calculating the average difference between the transformation required to align each shape to the mean shape and identity transformation.

5.3 Capturing Data from the aligned shape set

It is known that every aligned shape is represented by a single point, vector x_i in a $2n$ dimensional space. For N shapes, the cloud of N points occur. These points lie in a region called the Allowable Shape Domain [3], where in the points indicate the shape and the size of the region. Every point within domain gives set of landmarks whose shape is similar to those in the original training set.
For N aligned shapes, the mean shape $\bar{x}$ which is the centre of the ellipsoidal Allowable Shape domain is

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$$

For every shape, the deviation from the mean is calculated dx_i.

$$dx_i = x_i - \bar{x}$$

Next, the $2n \times 2n$ covariance matrix S is calculated

$$S = \frac{1}{N} \sum_{i=1}^{N} dx_i dx_i^T$$
$$S_{p_k} = \lambda_k p_k$$

where

λ_k : kth eigenvalue of $S, \lambda_k \geq \lambda_{k+1}$

p_k for $k = 1, ..., 2n$ are the principal axes of the ellipsoid

$$p_k^T p_k = 1$$

The most significant modes of variation in the variables are used to derive the covariance matrix as the longest axes of the ellipsoid are described by the eigenvectors corresponding to the largest eigenvalues. Small number of modes t are required to explain most of the variation. t can be calculated by choosing the least number of modes so that the sum of the variances account for the total variance of all the variables.

$$\lambda_T = \sum_{i=1}^{2n} \lambda_k$$

where

λ_T : Total variance of all variables

The kth eigenvector acts on point l by moving it parallel to (dx_{kl}, dy_{kl}) along a vector obtained from the l^{th} pair of elements in p_k.

$$p_k^T = (dx_{k0}, dy_{k0}, ..., dx_{kl}, dy_{kl}, ..., dx_{kn-1}, dy_{kn-1})$$

Any shape in the training set is approximated using the mean shape and weighted sum of the deviations obtained from the t modes.

$$x = \bar{x} + Pb$$

where

$\bar{x}$: The mean shape

$P = (p_1 p_2 ..... p_l)$ is the matrix of the first t eigenvectors

$b = (b_1 b_2 ..... b_l)^T$ is the vector of weights

New shapes can be generated by varying the parameters b_k however within the limits so that the shapes are consistent to the ones in the training set. The parameters are linearly independent.

The eigenvectors are orthogonal [4]

$$p^T p = I$$

Hence,

$$b = p^T (x - \bar{x})$$

The parameter limits are of the order of

$-3\sqrt{\lambda_k} \leq b_k \leq 3\sqrt{\lambda_k}$ as most of the population lies within the three standard deviations of the mean.

The parameters can also be chosen based on the some distance measures like the Mahalanobis distance (D_{max}) from the mean is less than the desired value of D_{max}.

$$D_m^2 = \sum_{k=1} l \left(\frac{{b_k}^2}{\lambda_k} \right) \leq D_{max}^2$$

6 Example Model

Consider Figure 3 providing some samples from the set of 66 heart chamber boundaries,each being represented by 96 points. The variation is ensured because of the training set is obtained from different individuals and at different stages of the cardiac cycle. The points represent the boundary of the left ventricle, the right ventricle and part of the atrium. Figure 4 shows the reconstructed shapes after the modification of the parameters [4].
Parameter $b1$ modifies the width, $b2$ modifies the wall separating the left and right ventricle, parameters $b3, b4$ change the shape of the mitral valve (between the Left ventricle and atrium).

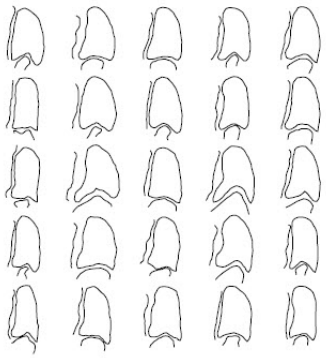


Abb. 6.3: Sample of Heart Ventricle Shapes [4]

7 Active Shape Model

Active shape models are an instance of the Point Distribution Model which deforms to improve their fit to the image. It helps in improving the correspondence between the model and image data. The shape constraints remain the same during the iterative deformation process. The process begins with putting the instance of the model on the image and deciding upon the deformation required for each point. Next, recompute the translation, scale and orientation and apply these changes. The residual deformation indicates the need to update the model parameters which control the shape. The resultant model best fits the image of the heart provided.

To find the model representing images, the following procedure is used.

1. A Hypothesis is made, giving the control points.

2.Each of the hypothesis is refined and the best is chosen.

Consider the instance of the model [3]

$$X = M(s, \theta)[x] + X_c$$

where

X is in the image frame created from x in coordinate frame.

$$X_c = (X_c, Y_c, X_c, Y_c, ..., X_c, Y_c)^T$$

where
(X_c, Y_c) : Position of the centre of the model
$M(s, \theta)[\quad]$: Rotation by θ and scaling by s

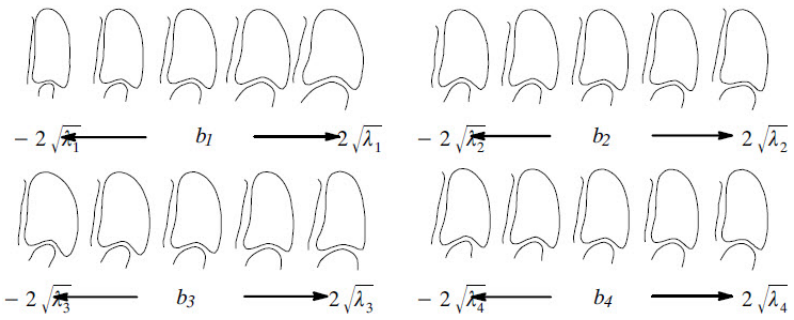


Abb. 6.4: Effect of the Modification of the Parameters [4]

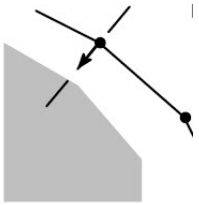


Abb. 6.5: Suggested Movement of a Point along normal to boundary in a direction where Profile Model best fits profile obtained from image [4]

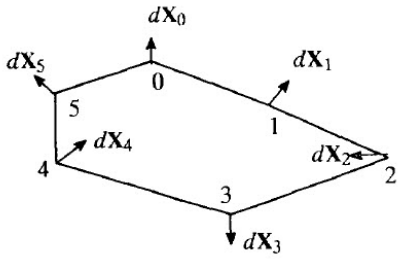


Abb. 6.6: Adjustment to set of points [3]

We need to find the shape parameters such that the model best fits the image structure. The aim is to find an X using an iterative process based on a rough estimate. The iterative process is as follows;

- Put the current X on image of interest and calculate the displacement required.
- Transform these elements to shape and scale parameters.
- Update the parameters.Constraint on the shape parameters ensures the model specificity.

7.1 Calculation of the movement of the Model Point

A profile model for each point is built. This is compared to the profile sampled from the image g. The model at point d pixels is

$$f_{prof}(d) = (h(d) - \bar{g})^T S g^{-1} (h(d) - \bar{g})$$

where

$h(d)$: Sub interval of g of length n_p pixels centred at d.

Lower the value of f_{prof} , better the fit.

The adjustment of the model points representing the boundaries to the image boundary is along the normal to the boundary proportional to the maximum edge strength along the normal as shown in Figure 5 .
These set of adjustments are denoted by vector dX as seen in Figure 6 where

$$dX = (dX_0, dY_0, ..., dX_{n-1}, dY_{n-1})^T$$

Consider d_{best} to be the distance along the sampled profile from model point to the point of the best fit [4].

$$\begin{array}{cc} |dX| = 0 & \text{if } |d_{best}| \leq \delta \\ |dX| = 0.5 d_{best} & \text{if } \delta < |d_{best}| < d_{max} \\ |dX| = 0.5 d_{max} & \text{if } d_{max} \leq |d_{best}| \end{array}$$

Here, δ is considered to be 0.5 pixels and d_{max} is considered to be 8 pixels.

7.2 Computation of changes in Shape and pose parameters

In order to move the points from current location X to $X + dX$, the pose and shape parameters need to be modified. For the current centre (X_c, Y_c) the transformation (dX_c, dY_c) needs to be found. Similarly, the rotation $d\theta$ and scaling factor $(1 + ds)$ needs to be found to achieve the best fit [3].
Once these are modified, deformation of the shape can be achieved by calculating dx in local coordinate frame.
That is,

$$X = M(s, \theta)[x] + X_c$$

Applying the adjustments,

$$M(s(1 + ds), (\theta + d\theta))[x + dx] + (X_c + dX_c) = (X + dX)$$

Thus,

$$M(s(1 + ds), (\theta + d\theta))[x + dx] = (M(s, \theta)[x] + X_c + dX) - (X_c + dX_c)$$

We get,

$$dx = M((s(1 + ds))^{-1}, -(\theta + d\theta))[M(s, \theta)[x] + dX - dX_c] - x$$

since $M^{-1}(s, \theta)[\quad] = M(s^{-1}, -\theta)[\quad]$

We transform dx to model parameter space, db to ensure shape constraint.

$$x{=}\bar{x}{+}Pb$$
$$x + dx \approx \bar{x} + P(b + db)$$

We know that ,

$$x = \bar{x} + Pb$$

Substituting this in above, we get

$$dx \approx P(db)$$

So,

$$db = P^T dx$$

Since $P^T = p^{-1}$

7.3 Choice of weights for parameters

We need to decrease the weights of the points which are away from the current model points than the average, including the outliers [4].

$$W_i = \left(\frac{1}{2+|dX_i|^2} \right)$$

7.4 Updation of Parameters

From all the equations above, the changes in $dX_c, dY_c, d\theta, ds$ and db can be calculated [3]. The iterative updation of the pose parameters $X_c, Y_c, d\theta, ds$ and shape parameter db is as follows,

$$X_c \to X_c + w_t dX_c$$
$$Y_c \to Y_c + w_t dY_c$$
$$\theta \to \theta + w_\theta d\theta$$
$$s \to s(1 + w_s ds)$$
$$b \to b + w_b db$$

Here, w_t, w_θ, w_s are scalar weights and w_b is the diagonal matrix of weights, one for each mode. Model specificity is achieved by con-straining the values of b_k. For Example,

$$D_m < D_{max}$$

where

D_{max} : Mahalanobis distance and has a value of 3.0

That is,
b should lie within the hyperellipsoid. If it lies outside, rescaling is required.

$$b_k \to b_k.(\tfrac{D_{max}}{D_m})$$

where $k = (1, 2, ..., t)$

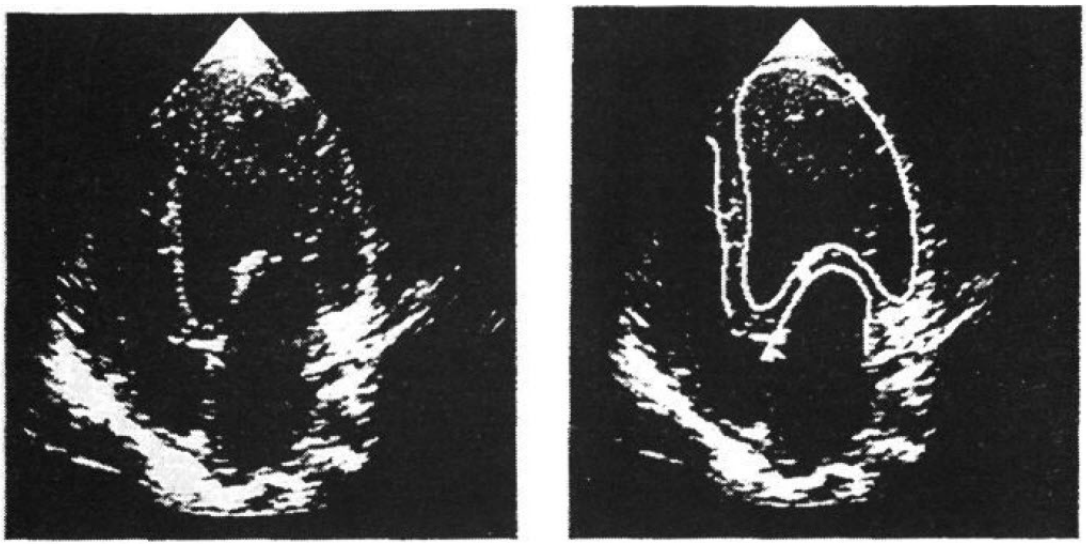


Abb. 6.7: Figuration of the Left Ventricle, Septum and Atrium [6]

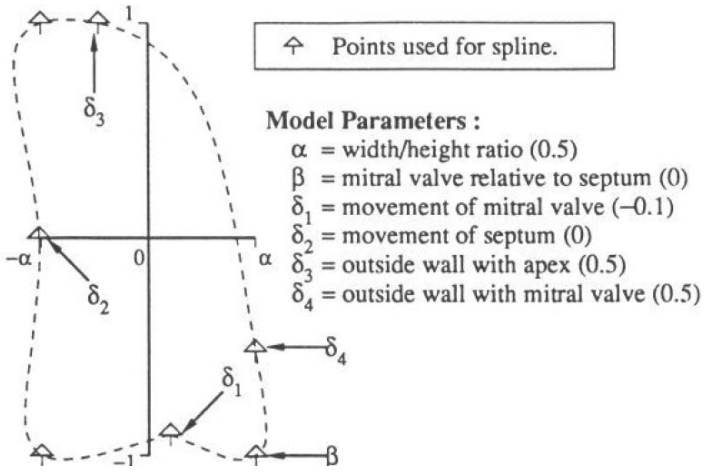


Abb. 6.8: The Left Ventricle Model[1]

8 Initial Guess

Stasm is a C++ software library used to find the features. Given the training set, it returns the positions of the features [14]. The Initial guess of the parameters for shape,scale and orientation is explained in [1]. The parameters need to be chosen so that the best fit is obtained. Here, the Genetic Algorithm search is used. Let us consider an echo cardiogram of heart and the objective is to locate the boundary of the Left Ventricle, as in Figure 7.

During the iterative process, further on for Active Shape Model creation, the following hurdles are encountered.

- Noise
- Occlusion of the boundary by other internal structures
- Poor quality of the echo cardiogram images

For the creation of the model, the initial following parameters are defined. As shown in Figure 8, the parameter α denotes width to height ratio. Thus, the point $(-\alpha, 1)$ represents the apex and $(-\alpha, -1)$ shows the connection of the mitral valve to the atrium.

9 Practical Example

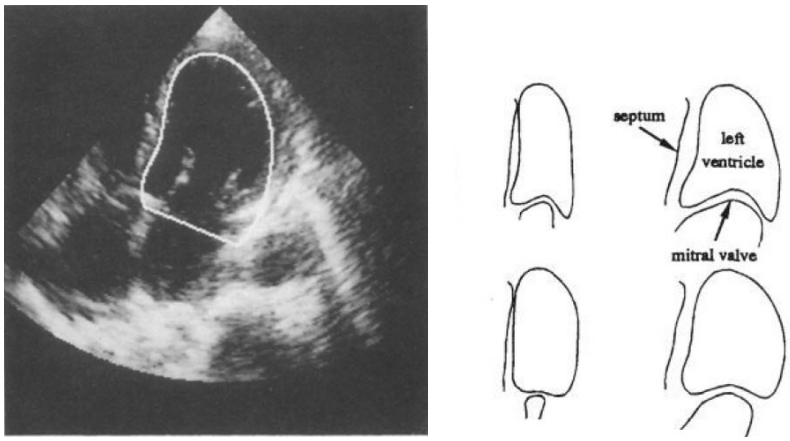
Parameter β represents the angle at which the mitral valve is to the septum. δ_1 governs the movement of the mitral valve. δ_2 governs the position of the control point at the centre of the septum. δ_3 governs the shape by calculating the horizontal displacement from the apex. δ_4 defines the vertical displacement from the mitral valve.

The active shape model and Genetic Algorithm are combined where in the set of parameters to get the best model are first obtained by the GA approach which is later refined by the Active shape Model. This leads to rapid convergence and obtaining the best solution. GA search involves searching the parameter space to achieve the objective function, here being the best fit. A population of solutions is obtained. The iterative process ensures the production of new generation of solutions. The selective breeding is applied to get the best shape and pose parameters [7].
Increased specificity of the model can be seen where the automatic location of the Left Ventricle, mitral valve and septum are seen. Figure 9 a) enables the best objective match. The aim is to locate the boundary of the left ventricle as shown in Figure 9 b) . Echo cardiogram time sequences are used for the Point Distribution model. From each sequence, two images are selected- left ventricle in the most contracted and expanded form. The left ventricle boundaries are labelled. Four points on each boundary are placed at the apex and the mitral valve. The intermediate points are placed at equidistance making it 18 points in total. 6 modes of variation are identified in left ventricle shape of this training set. The first mode varies the width of the model. The second mode varies the position of the lower part of the ventricle. The third and fourth modes perform other deformations of the boundary. The strongest edge is elected for the new position of the point during image search.

Figure 10 a) shows an echo cardiogram. Figure 10 b) indicates the initial placement of 96 points as described in the example model. Set of pose parameters are chosen and all the shape parameters set to 0. Figure 10 c) shows the Active Shape Model after 20 iterations. Figure 10 d) shows a more accurate and refined Active Shape Model after 80 iterations. After 200 iterations, a model gives the best fit to the data as seen in Figure 10 e). 12 degrees of freedom are present. Adjustments are made based on calculation using the strongest edge in the image [3]. All missing data with respect to boundary are obtained based on the knowledge of expected shape and the information from the areas where the ventricle wall is seen.

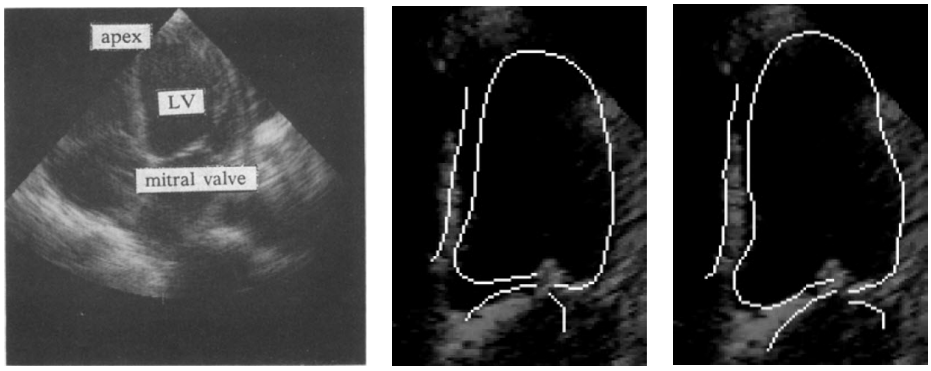
10 Discussion

A number of issues arise and there exist a number of areas where there is a scope for improvement. The issues to be discussed include Clutter and noise, the shape of the allowable shape domain and the placement of the landmark points [3]. The further research areas include the generation of the active shape model for three dimensional images.

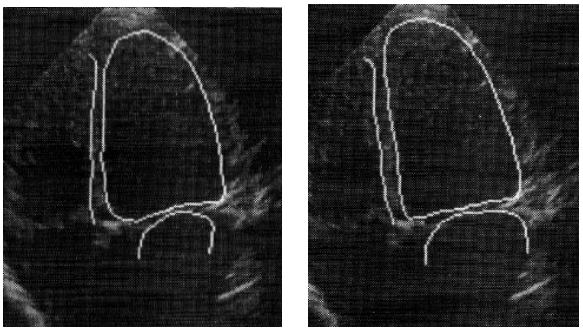


(a) Best Fit (b) Anatomy of Heart

Abb. 6.9: Locating the LV [7, 6]



(a) Echo cardiogram (b) Initial positions (c) After 20 iterations



(d) After 80 iterations (e) After 200 iterations

Abb. 6.10: Iterative superimposition of example model on echo cardiogram [7, 4, 3]

10.1 Accurate placement of Landmark Points

The landmark points need to be placed properly in the initial state otherwise, the noise terms are added leading to the points not being aptly located. This leads to alignment issues. The alignment of all the points in the training set need to be similar to allow the comparison of the equivalent points.
It is required that the examples to create the model have variety of shapes else, the modes of variation describing the small amount of differences might get truncated.

10.2 Models for Object with multiple sub parts

In our example of the echo cardiogram of heart, the 96 points are placed not over a single part but over three different internal parts, which being the Left Ventricle, the Atrium and the septum. The connectivity between the points marking these parts is essential in determination of the normal at every point.

10.3 Allowable Shape Domain

The Allowable Shape domain is considered to be an ellipsoid. However, if there is a large rotation in the training set, non linear relationships between the landmark points arise giving the $2n$ Dimensional space a banana like shape. There is a requirement of a more general Allowable Shape domain.
Locating objects in Active Shape model is based on initial estimates of scale, position and orientation. Calculation of these estimates depends on how cluttered the image is and the usefulness of the model.

10.4 Clutter

The example model for heart shows that Active Shape Model works with images consisting of missing data .In presence of clutter, noise and occlusion, the model boundary might join on a wrong image edge. The shape constraints ensures however the model specificity.

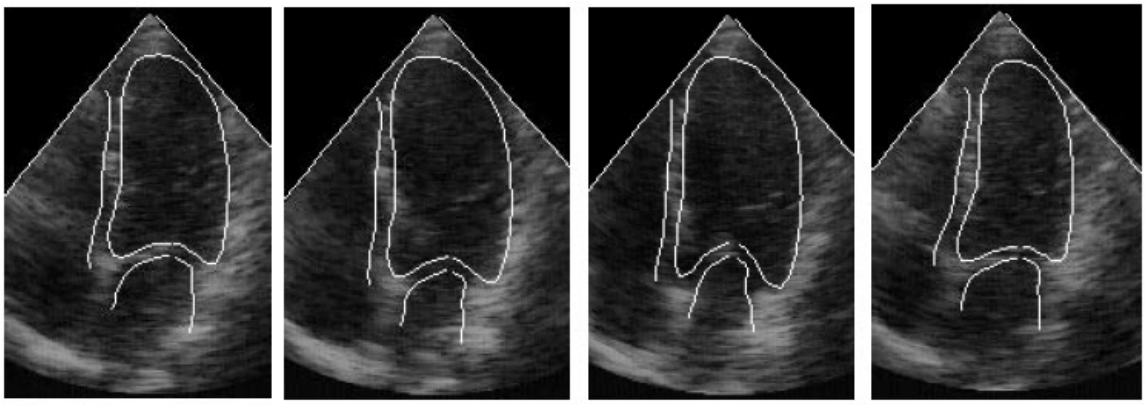


Abb. 6.11: Tracking the Left Ventricle [4]

10.5 Updation of Model Parameters

Adjustments and movement is based on the detection of the strong edge. A potential map is used, where the model points try to move to the likely locations.

10.6 Advantages and Disadvantages

The Active Shape Model ensures model specificity, that is the model is in tandem with the shape of the given image of the object which is achieved by adding the shape constraints. The Active Shape model is highly robust to noise and clutter.
On the other hand, the creation of the Active shape Model is complex when compared to the Finite element model or the active Contour model. This complexity arises from the need to mark all the edges in the training set with the correct interpretation.

10.7 Extension to 3 dimensional images

Active Shape Models can be applied to 3D images for which initially a 3D Point distribution Model is required [8]. In our example of the heart, the object is initially labelled with landmark points in 3 dimensional system (x_0, y_0, z_0). The points are marked on the surface of the object. For labelling, the object is sliced into segments.

The position for the slice operation required can be obtained by using the Genetic Algorithm [4, 6]. Consider the Figure 11,

- GA is applied to each slice to locate the left Ventricle.
- Refinement is done using Active Shape Model.
- The segmentation obtained is applied to the next slice providing the initial Left Ventricle position.
- The pose and the shape parameters, obtained from guessing, of the previous slice is compared to those of the next slice to get the new Left Ventricle position.
- The above two steps are repeated for each slice.

11 Conclusion

In this report, a brief overview over the Point distribution Model is provided which involves the construction of the model from the training set of aptly annotated images. A number of parameters, modes of variation are given to show the changes in shape. Model specificity is ensured by applying constraints on these parameters.
Active Shape Model uses the Point distribution Model as a part of the iterative process for image search. It is highly robust to noise, clutter because of the limitations of model instances.

We consider the heart shown in an echo cardiogram and model the internal parts for image search. We also discuss an important area demanding further research of 3 dimensional images. The Active shape model can be used to track sequences of images slicing the 3 dimensional image of heart and applying the iterative process to obtain the Active Shape Model.

Literaturverzeichnis

[1] Hill A and Taylor CJ. Model- based image interpretation using genetic algorithms, 1992.

[2] Pentland A and Sclaroff S. Closed-form solutions for physically based modeling and recognition. *IEEE TRANS. PATTERN ANAL. MACH. INTELL*, 13:715–729, 1991.

[3] Cooper DH Cootes TF, Taylor CJ and Graham J. Active shape models- their training and application. *COMPUTER VISION AND IMAGE UNDERSTANDING*, 61(1):38–59, 1995.

[4] Taylor CJ Cootes TF, Hill A and Haslam J. The use of active shape models for locating structures in medical images. *IMAGE AND VISION COMPUTING*, 12(6):355–366, 1994.

[5] Terzopoulos D and Metaxas D. Dynamic 3d models with local and global deformations: Deformable superquadrics. *IEEE TRANS. PATTERN ANAL. MACH. INTELL*, 13:703–714, 1991.

[6] Cootes TF Hill A and Taylor CJ. A genetic system for image interpretation using flexible templates. In *THIRD BRITISH MACHINE VISION CONFERENCE*, pages 276–285. Springer-Verlag, 1992.

[7] Taylor CJ Hill A and Cootes T. Object recognition by flexible template matching using genetic algorithms. In Ed. Sandini G, editor, *PROCEEDINGS, EUROPEAN CONFERENCE ON COMPUTER VISION*, pages 852–856, Berlin/New York, 1992. Springer-Verlag.

[8] Thornham A Hill A and Taylor CJ. Model-based interpretation of 3d medical images. In Ed. Illingworth J, editor, *PROCEEDINGS, FOURTH BRITISH MACHINE VISION CONFERENCE*, pages 339–348. BMVA Press, 1993.

[9] Williams CKI Hinton GE and Revow M. Adaptive elastic models for hand-printed character recognition. In Hanson SJ Moody JE and Eds. Lippmann RP, editors, *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 4*, pages 512–519, San Mateo,CA, 1992. Morgan Kauffmann.

[10] Baker KD Karaolani P, Sullivan GD and Baines MJ. A finite element method for deformable models. *PROCEEDINGS OF FIFTH ALVEY VISION CONFERENCE*, pages 73–78, 1989.

[11] Sullivan GD Karaolani P and Baker KD. Active contours using finite elements to control local scale. In *BRITISH MACHINE VISION CONFERENCE*, pages 481–487. Springer-Verlag, 1992.

[12] Witkin A Kass M and Terzopoulos D. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988.

[13] O'Keeffe D Cavanaugh J Taaffe J Lipson P, Yuille Al and Rosenthal D. Deformable templates for feature extraction from medical images. In Ed. Faugeras O, editor, *PROCEEDINGS OF THE FIRST EUROPEAN CONFERENCE ON COMPUTER VISION*, pages 413–417, Berlin/New York, 1990. Springer-Verlag.

[14] S. Milborrow and F. Nicolls. Locating facial features with an extended active shape model. *ECCV*, 2008.

[15] Chin RT and Dyer CR. Model-based recognition in robot vision. *ACM COMPUTING SURVEYS*, 18:67–108, 1986.

[16] Cohen DS Yuille AL and Hallinan P. Feature extraction from faces using deformable templates. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 8:99–112, 1992.