UNIVERSITÄT KOBLENZ · LANDAU

Fachbereich 4: Informatik

Path Similarity Skeleton Graph Matching for 3D Objects

Diplomarbeit
zur Erlangung des Grades
Diplom-Informatikerin
im Studiengang Computervisualistik

vorgelegt von

Simone Schäfer

Betreuer: Dipl.-Inform. J. Hedrich, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau
Erstgutachter: Jun.-Prof. Dr. Ing. Marcin Grzegorzek, Institut für Bildinformatik, Departement Elektrotechnik und Informatik, Universität Siegen
Zweitgutachter: Prof. Dr.-Ing. Dietrich Paulus, Institut für Computervisualistik, Fachbereich Informatik, Universität Koblenz-Landau

Koblenz, im April 2011

Kurzfassung

In dieser Diplomarbeit wird ein skelettbasiertes Matching-Verfahren für 2D - Objekte vorgestellt. Zunächst werden aktuelle Ansätze zum Matchen von Objekten vorgestellt, anschließend werden die Grundlagen von skelettbasiertem Matching erklärt.

Ein skelettbasiertes Verfahren wurde im Rahmen dieser Arbeit gemäß des vorliegenden Original-Paper neu implementiert. Diese Implementierung wird anhand einer Ähnlichkeitssuche in drei Bild-Datenbanken evaluiert. Stärken und Schwächen des Verfahrens werden herausgearbeitet.

Des weiteren wird der vorgestellte Algorithmus auf Erweiterungen untersucht, die das Matchen von 3D-Objekten ermöglichen sollen. Im speziellen wird das Verfahren auf medizinische Daten angewendet: Pre- und postoperative CT-Aufnahmen der abdominalen Aorta eines Patienten vor und nach einer Operation werden miteinander verglichen. Problemfälle und Erweiterungsansätze für das Matchen von 3D-Objekten im Allgemeinen und von Blutgefäßen im Speziellen werden vorgestellt.

Abstract

In this diploma thesis a skeleton-based matching technique for 2D shapes is introduced. First, current approaches for the matching of shapes will be presented. The basics of skeleton-based matchings will be introduced.

In the context of this diploma thesis, a skeleton-based matching approach was implemented as presented in the original paper. This implementation is evaluated by performing a similarity search in three shape databases. Strengths and limitations of the approach are pointed out.

In addition, the introduced algorithm will be examined with respect to extending it towards matching of 3D objects. In particular, the approach is applied to medical data sets: Pre- and postoperative CT images of the abdominal aorta of one patient will be compared. Problems and approaches for matching of 3D objects in general and blood vessels in particular will be presented.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Vereinbarung der Arbeitsgruppe für Studien- und Abschlussarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden.	ja ☐	nein ☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.	ja ☐	nein ☐

Koblenz, den 7. April 2011

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The amount of image data is growing rapidly, supported by the increasing possibilities to save huge amount of data. So does the number of 3D models, as advanced scanning methods and higher processing power get less expensive and hence more widespread.

The problems stemming from this development are similar for both 2D and 3D data: The recording and consumption of such data gets more easy, but the complexity of searching through the data and establishing relations between these items complicates the access to the data. In comparison to well-tried data types like plain text documents, images and 3D-models are difficult to manage by software. For example, an image retrieval system enables the user to browse through a database of images. Searching in semi-structured data like text documents is less complex as search conditions can be controlled by syntactic means like keywords contained in the documents. Image searches have to happen by semantic aspects which are not explicitly known in an image [Sch06].

Similarity measures for 2D or 3D images are therefore an ongoing research topic. They are an important contribution to various applications, like multimedia retrieval, object classification or object recognition. Multimedia retrieval deals with the problem of similarity search in databases where no explicit similarity for the contained objects is known. Classification and recognition usually rely on a comparison of a query shape to a set of known prototype shapes.

Thus, comparing two shapes is a key component in the applications mentioned before. In order to compare two shapes, salient features of the two shapes are selected. The query shape's features are compared to the target shape's features based on a predefined similarity measure. That way, the features of one object can be mapped to the features of another object so that they are as similar as possible:

Correspondences are established between the objects' features. This process is also called *matching*. The matching of two 2D or 3D images is hence a key component in all of the applications mentioned before. The challenge is to find a meaningful similarity measure that captures most of the original shape's properties.

1.2 Document structure

This diploma thesis can be divided into two parts: First, the promising skeleton-based technique for matching two-dimensional objects presented in [BL08] will be introduced. The technique was reimplemented, evaluated and tested for limitations. Second, the possibilites to extend this technique for matching three-dimensional objects are examined. The approach is tested by matching blood vessel data.

This document is structured as follows: Chapter 2 gives an overview of existing work for matching techniques of 2D and 3D images. In particular, previous work for skeleton-based matching is examined. In chapter 3, the fundamentals of skeleton-based matching approaches are explained, including basic notations, skeletons and fundamentals of graph matching. Basic terms and notations will be introduced. An outline of the matching algorithm presented in [BL08] is described in chapter 4. In addition, the implementation of this algorithm is evaluated in several matching and retrieval experiments on three shape databases. Advantages and limitations of the algorithm are pointed out. In chapter 5 the problems for applying the algorithm for the matching of 3D data are highlighted. Some ideas on the extension of this approach to three-dimensional objects are introduced. Conclusions and future work are summarized in chapter 6.

Chapter 2

Overview of matching techniques

When comparing two shapes, a representation for the shapes is needed. Usually, the shape representation is a reduction of the complex information contained in a shape. The representation has to preserve most important geometrical and toplogical features of a shape while making it easy to apply matching algorithms to them. The effectiveness of an algorithm for matching two- or three-dimensional objects highly depends on the choice of representation for the object. Apart from color- and texture based techniques, an important group of algorithms analyze objects based on their shape. User surveys show that users, for example in the context of an image retrieval system, are generally more interested in matching by shape than by color or texture [SLV99].

Typical approaches for shape-based object matching techniques will be introduced in this chapter.

Point set representation One approach is the matching based on unorganized point sets. [CR03] state that points are the "most fundamental of all features". Two sets of points are mapped by finding a useful local descriptor, like color or location, then establishing a one-to-one correspondence between the two point sets based on the descriptors.

[BMP02] introduce as a shape descriptor the *shape context.* Within this approach the shape is represented by a finite set of boundary points, which are not required to be landmarks or salient points. For each of these boundary points, a coarse histogram containing the relative coordinates to the other points of the shape is computed. The goal is to find corresponding pairs of points in both shapes with shape contexts that have the highest similarity.

Typically, in approaches based on point sets, landmark points are extracted from the shape, and these feature points are matched [SP08]. One possibility for a local point descriptor is SIFT (*scale-invariant feature transform*) as proposed in

[Low04]. SIFT features are invariant to scale and rotation, and robust to global illumination changes.

The strength of landmark representations is that, depending on the choice of landmark, strong and non-ambiguous correspondences between the single elements can be established [SP08]. When used alone however, point based matching approaches tend to suffer from supoptimal conditions like noise or outlier elements [KG10, SK05] as they usually incorporate only local properties.

Boundary representation Another object matching approach is based on boundary representations. A classic way to matching curves is based upon the models of *snakes* as originally proposed in [KWT88]: In this model, a snake is a spline, conceivable as an elastic band, influenced by both internal image forces and external constraint forces, both trying to push the snake to model itself after the shape of an object. These forces could be caused by a user interface or can be detected automatically. The external force pushes the snake towards the correct location on the shape, usually salient features like edges or lines, while the internal force is used to align the shape with the contour.

[You98] use this idea and apply it directly to contour matching by estimating the costs to stretch or bend the query object's contour so that it best matches the target object's contour. The less energy needed to match the two contours to each other, the higher the two shapes' similarity is.

However, boundary representations cannot access shapes' interior. [SK05] states that most curve based techniques are not invariant to scale or rotation. Another problem is that deformable objects could lead to unsatisfying results when using curve based techniques, for example in case of articulated joints. For some applications, curve based representations lead to good results, mainly in limited domains like recognition of hand writing, but as the experiments in [SK05] demonstrate, most curve-based approaches show their limitations when dealing with overlapping object parts.

Skeleton representation Another approach for object representation, the *skeleton* or *medial axis*, gives access to both shape interiors and boundary properties. The skeleton is the set of points within a shape that are the center of a maximal inscribed disc within the shape, that touches the objects boundary in at least two points. Thus, a skeleton is the reduction of a shape to a thin line centered within the shape. A skeleton captures essential topology and shape information of the object in a simple form [LP09]. Skeletons hold information about the interior of an object as well as information about the object's outline. When objects are represented by skeletons, these skeletons are mostly reduced to a graph, and usually graph matching techniques can be applied. These are generally more computationally expensive than the matching of curves. Compared to curve based approaches, skeleton-based methods show their advantages in the matching of deformable objects as they are more robust to overlaps, deformations or misplaced object parts [SK05].

In general, skeletons offer different possiblities for matching approaches. Complete skeleton branches can be matched to the skeleton branches in the other skeleton. Another general approach is to match single salient skeleton points, usually junction nodes or end nodes, or both.

Several approaches [SKK04, SK96, KSSK00, KSK01] use a derived form of the skeleton, called the *shock graph* or *shock tree*. Shock graphs were first introduced in [SK96]. This derived representation differs from skeletons in that the contour information of a shape is incorporated.

[KSSK00, KSK01] match shock trees based on an edit-distance algorithm. The edit distance is computed by traversing the rooted shock tree, and edit operations like stretching or bending are applied to the traversed edges. The idea is to deform one skeleton branch in one skeleton to another branch, that is more similar to a specific branch in the other skeleton. Possible edit operations include operations that might change the graph topology, as for instance inserting, merging or deleting branches, and topology-preserving operations like stretching or compressing a branch. Each edit operation comes with a predefined edit cost. The correspondence between skeleton branches can then be found by finding the "cheapest" edit operation to transform one shock graph into another. The similarity of two shapes is defined as the sum of all edits costs. The skeleton needs to be converted to a rooted tree before matching which might lead to a loss of topological information.

The main idea of the shock graph based approach introduced in [SKK04] is to "treat each shape as a point in a shape space and define the distance between two shapes in terms of the minimum-cost deformation path connecting them" [SKK04]. In order to reduce the dimensionality of all possible deformations, the shape space is partitioned into shape cells, where each shape cell contains shapes having identical shock graph topology. In order to compute the similarity between two shapes, an edit-distance algorithm similar to [KSK01] is applied. Editing a query shape will lead to one or more transitions between the shape cells. Correspondences between two shapes can be established by observing these transitions.

[HHW04] match skeletons mainly based on their topological information. Two skeletons' branches are matched according to their connectivity within the skeleton, observing bifurcations at junction nodes in reference to the bifurcation angles between skeleton branches emanating from junction nodes. Furthermore, for object recognition, the object shape variations are incorporated by computing a Gaussian distribution on the distance values within the shape. This approach lacks flexibility for matching non-rigid objects: As the angle of branches at junction nodes differ if parts of the objects are moved, this is no reliable matching indicator for articulated joints. Moreover, junction nodes tend to get disarranged in moving objects [XWB09] and are thus not a reliable matching feature when used without other support matching indicators.

[DSK$^+$06, DSD09] deal with the problem that in noisy image data, due to errors for example in image aquisition or segmentation, one-to-one matchings are not always possible. This approach thus significantly differs from other approaches in the process of the final matching. In most matching approaches, a one-to-one correspondence is enforced. That is, each element in one skeleton has to be matched to exactly one element in the other skeleton. [DSK$^+$06] embed two skeleton graphs into the same space and map each node's attributes to a vector of masses. In this scenario, the matching of skeletons is not the matching of a graph anymore but the computation of the minimal flow from one weighted point set to another which can be computed by *Earth Mover's Distance* (*EMD*) [RTG98]. The advantage of EMD over other approaches is that it permits partial matchings instead of enforcing one-to-one-matchings.

Most of the existing approaches based on skeletons cannot deal with holes in the shapes which would lead to loops in the skeleton [BL08]. One advantage of the algorithm proposed in [BL08] (which will be described in detail later) over other approaches is hence that loops are no problem for this algorithm.

Furthermore, several approaches for the matching of 3D models based on skeletons have been proposed. In [CDS$^+$05] the distance transform value is assigned to each skeleton point which is used in the final matching process. Again, two skeletons are matched using the Earth Movers Distance which also permits partial matchings.

[BI04] propose a backtracking-based approach. The main idea is to find the largest common subgraph of two skeletons and compute the similarity between them. In order to find corresponding vertices and edges, the length and angles of branches at junction nodes are incorporated when computing the similarity between two subgraphs. This approach has exponential computation complexity. As the angles of branches in this approach are a significant measurement for similarity non-rigid objects are hard to match with this approach.

[SSGD03] match the vertices of two skeleton graphs. A so-called *signature* is assigned to each graph vertex. These signatures are vectors representing the structure of the underlying subgraph at this node, based on the eigenvalues of the subgraph's adjacency matrix. The similarity between two nodes is then defined by the distance between these signatures. Thus, this approach only takes local shape information into account.

Chapter 3

Foundation of skeleton-based matching

In the following chapter the fundamental concepts that form the foundation for this thesis will be described.

As mentioned in chapter 1, in this diploma thesis a matching technique will be presented and evaluated. In particular, the presented matching algorithm uses skeletons as a shape descriptor. Therefore, section 3.1 will summarize the fundamental concepts of skeletons in 2D and 3D images. The computation of the similarity between salient skeleton points will later be reduced to the problem of matching time series, that is, sequences of real numbers. Section 3.2 therefore introduces the main ideas of the matching of time series. As the final matching of salient skeleton points will be based on graph theory problems, the fundamentals of graph matching will be introduced in section 3.3. Finally, the basic concept of retrieval systems will be introduced in section 3.4, as the matching algorithm will be evaluated by a simple retrieval software.

3.1 Skeletons

Initially, the term *skeleton*, also called *medial axis*, has to be clarified. Skeletons are a shape descriptor for objects. Shape descriptors generally represent shapes in an abstracted way, reducing the content of the shape to facilitate further processing and analysis. Skeletons in particular are an “abstraction of objects, which contain both shape features and topological structures of original objects’ [BLL07]. First, skeletons will be introduced for 2D objects. Then, an introduction to 3D skeletons will be given.

3.1.1 Skeletons in 2D

For the first time, skeletons as a shape descriptor were mentioned 1967 in the work of Harry Blum [Blu67] about visual perception of shapes. According to Blum's idea, from each edge and each corner of an object, waves are spreading uniformally in all directions. These waves don't interfere with each other, and all waves propagate in the same speed. Two waves are cancelled as soon as they collide with each other.

One can imagine that the foreground pixels in a binary image are made of prairie grass. All shape boundary points are then set on fire simultaneously, and the fire fronts propagate inside the object at the same speed. After some time, two or more of these fire fronts will converge in a point which is then said to belong to the skeleton S of the shape. After a clash, the involved fire fronts are extinguished [Ogn92].

Another geometrical model of skeletons is to consider skeletons as the set of all interior points of an object, where each point is the center of the largest disc that exactly fits within the object boundary, called the largest inscribed disc. In particular, a disc Δ is said to be a maximal inscribed disc if the following conditions hold [Mai99]:

1. Δ is totally contained in the shape
2. There is no other disk totally contained in the shape which contains Δ

A maximal inscribed disk Δ centered at a skeleton point p touches the object boundary in at least two points.

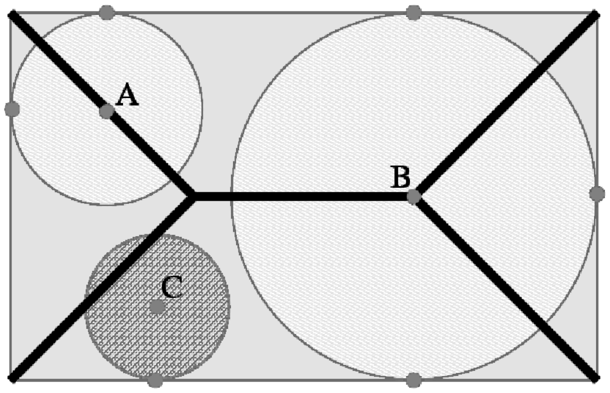


Figure 3.1: Rectangle and its skeleton, marked in black lines. A and B are skeleton points, as both are the center of a maximal inscribed disc, touching the boundary in at least two points. Image source: [Pal].

Figure 3.1 shows a simple example, a rectangle and its corresponding skeleton. Three points within the shape are highlighted: A, B and C. The maximal inscribed disc that center on them but fit within the shape is shown. The surrounding discs centered at A and B touch the object boundary in at least two points. Thus, A and B are skeleton points. C is not a skeleton point, as the maximal inscribed disc centered at C touches the boundary at only one point. From the skeleton's definition it can be concluded that the skeleton is always completely contained within the shape.

Both theoretical models - the propagation model and the model of maximal inscribed discs - are equivalent. The geometric consideration underlying both is that each skeleton point is equidistant to at least two boundary points.

Formally, and with the idea of maximal discs in mind, the skeleton S of a shape Ω with the boundary $\partial\Omega$ is defined as the set of all "points p in Ω that have at least two boundary points a, b at minimum distance of p" [Ren09]:

$$S = \{p \in \Omega \mid \exists a, b \in \partial\Omega, a \neq b, \|p - a\| = \|p - b\| = \mathrm{D}(p)\} \tag{3.1}$$

where $\mathrm{D} : \Omega \mapsto \mathbb{R}_+$ is the *distance transform*, "assigning to each object point the minimum distance to the boundary" [Ren09]. The boundary points a, b with the described properties are called the *feature points* of the skeleton point p.

Figure 3.1 shows that a skeleton is a connected set of digital arcs, in this example, straight lines. These curves are called *skeleton branches*. Each skeleton branch consists of a finite number of skeleton points. Skeleton points can be classified according to the number of their feature points. Skeleton points having exactly two feature points are called *connection points*. Point B in the figure is a so-called *junction point*, that is, a skeleton point where at least three skeleton branches meet. Junction points have at least three feature points, depending on the number of branches they are connecting. The maximal inscribed disc of an endpoint partly overlaps with the shape contour. Thus, the feature points form one contiguous set [Ren09].

For the purpose of image analysis, skeletons are often further simplified by creating the so-called *skeleton graph*. A skeleton graph is a reduction of the original skeleton to only end points and junction points, as these are the kind of points that hold the skeleton's topological information. Thus, the skeleton graph is created by removing all connection points and directly connecting the remaining points [YBYL07, BL08].

To use skeletons as a shape descriptor in digital image processing, the theoretical concepts of the skeleton have to be mapped to the discrete pixel space. A shape's skeleton is usually encoded in a binary image. Given a binary skeleton image, the skeleton is defined by the set of pixels labeled as foreground pixels. Generally, one foreground pixel represents one skeleton point. Henceforth, a pixel in the skeleton binary image labeled as foreground will be referred to as skeleton pixel, and the words skeleton point and skeleton pixel will be used interchangeably.

When processing a skeleton image, it is desirable to be able to identify the topological points, that is, end points and junction points, by the examanation of the 8-neighborhood of the skeleton pixel. For example, a skeleton pixel having exactly one adjacant skeleton pixel is an end point. The distinction between the two remaining types of skeleton points is a little more complex. Generally, a skeleton point where at least three skeleton branches meet is called junction point, and skeleton points that are neither end points nor junction points are called *connection points*. For the discrete case, one could conclude that junction points always have at least three adjacent skeleton pixel in the 8-neighborhood, while connection points have exactly two adjacent skeleton pixels. However, in practice, there are pixel constellations where connection points have more than two neighbors. An example is shown in figure 3.2. The left half of the image shows an example skele-

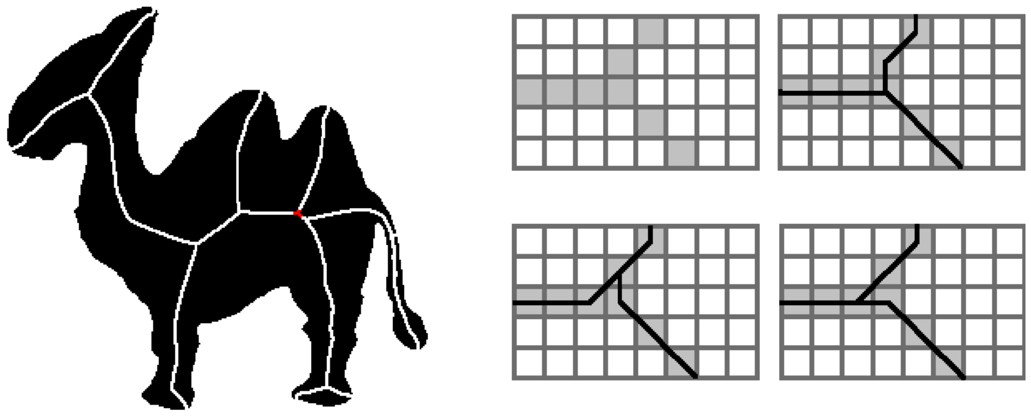


Figure 3.2: Junction points can not be identified unambiguously in the 8-neighborhood. As can be seen in the skeleton cut-outs in the right, in some constellations, the choice of junction points depends on the order in which pixels are traversed when creating the skeleton graph.

ton. The pixel rasters drawn in larger scale in the right half of the image show a cut-out of the skeleton in the left. As shown in the cut-outs, the choice of junction point depends on the order in which pixels are traversed when creating the skeleton graph. Obviously, in this pixel constellation three different pixels could be chosen as junction point. Thus, the identification of junction points requires more consideration. One possibility is to classify junction points as they are encountered while traversing the skeleton branches, while all neighboring nodes are classified as connection points. This would lead to random choices, as the choice of junction point highly depends on the starting point for the traversal of the skeleton points. This issue is not widely discussed in literature. [RJP00] propose to deal with this issue by assigning priorities to different types of connections of skeleton pixels. The idea is to trace the skeleton branch at potential junction points in a way, that edge-connected neighbors are traversed first. The skeleton point with the most edge-connected neighbors will be chosen as the junction point.

Skeleton properties An object's skeleton has the following properties, which can be derived from its formal definition [CS07, Ren09, Mai99]:

Centered By definition, skeletons are centered within the object. Furthermore, a skeleton S of an object F is totally contained in F. One problem with this condition is the discretization in image processing. Due to inaccuracies in the discrete pixel space, the skeleton might not be exactly centered. For example, in a rectangle with a height of an even number of pixels, there are two pixels that could be seen as center of the object, as shown in Figure 3.3.

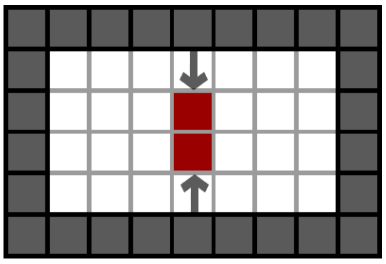


Figure 3.3: In the discrete pixel space, skeletons are not exactly defined. In this example, each square represents a pixel, grey pixels are boundary pixels. The clash of the two fire fronts implied by the two arrows would be between the pixels marked with red, which is not possible.

Thin The skeleton abstracts the original shape to a thin representation, and thus has one dimension less than the original shape. In the discrete two-dimensional case, the object is abstracted to a thin line. This means that the skeleton is exactly one pixel thick at each position.

Homotopic A skeleton preserves the topology of the original object, as was proven in [Lie04]. Simply said, "two objects have the same topology if they have the same number of components, tunnels and cavities" [CS07]. That is, the upper-level structures of an object Ω and its skeleton S can be mapped onto another by a continuous transformation [Mai99].

Reconstruction The structure of the skeleton alone is not sufficient for reconstruction of the original shape. To allow the reconstruction of the original shape from a skeleton, additional information has to be stored with the skeleton. If the distance of a skeleton point to its feature points is known, the original shape can be restored as it is known that the feature vectors ***pa*** and ***pb*** pointing from a skeleton point p to its feature points a and b are normal to the shape boundary [Ren09]. This structure is then called a *medial axis transform.* However, in practice, accurate reconstruction can get difficult due to inaccuracies in the pixel space, as shown in figure 3.3.

Uniqueness The medial axis transform is unique for different shapes. However, the skeleton is not: Two different objects can have the same skeleton. An example is shown in figure 3.4

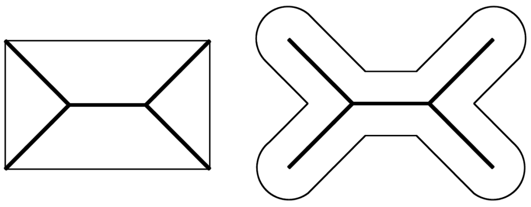


Figure 3.4: Two different shapes, having the same skeleton. Image source: [Pal]

Connected A skeleton is contiguous. No gaps or holes exist on any skeleton branch. In the discrete pixel space, this means that every skeleton point has at least two skeleton points in its 8-neighborhood, besides end points, which have exactly one skeleton neighbor pixel in the 8-neighborhood.

Transformation invariant As isometric transformations do not change the geometry of a shape, skeletons are invariant to isometric transformations like rotation, translation and uniform scale change.

Unstable The biggest weakness of skeletons is their instability towards noise. Figure 3.5 shows the skeletons of a bird and the influence of various deformations by noise.

The types of noise are classified as salt-and-pepper noise and boundary noise. Salt-and-pepper, as shown in figure 3.5b, is the most intrusive noise in the context of object skeletons. Removing one single pixel in the shape causes a change in topology. Boundary noise can cause additional branches, as shown in figure 3.5c and 3.5d.

Typically, these problems are dealt with by preprocessing the input image, i.e. with gaussian smoothing. Additional branches, occuring by boundary noise, can be eliminated after skeletonization. This instance is called *pruning*, which will be explained later.

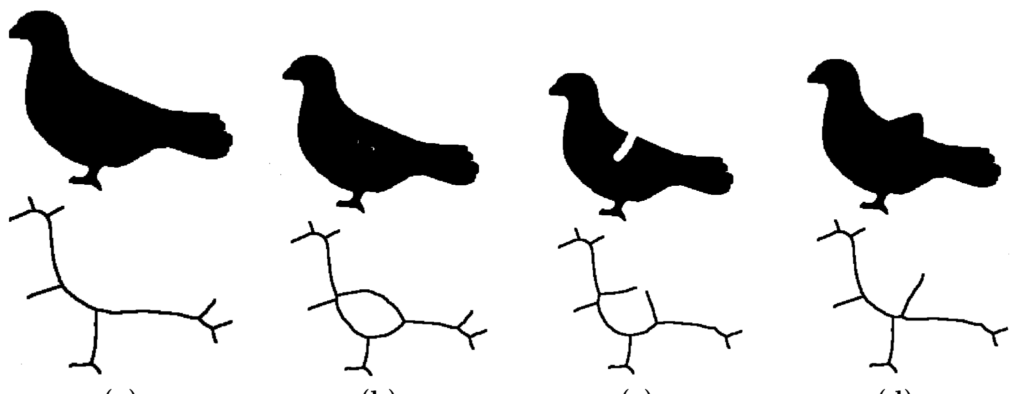
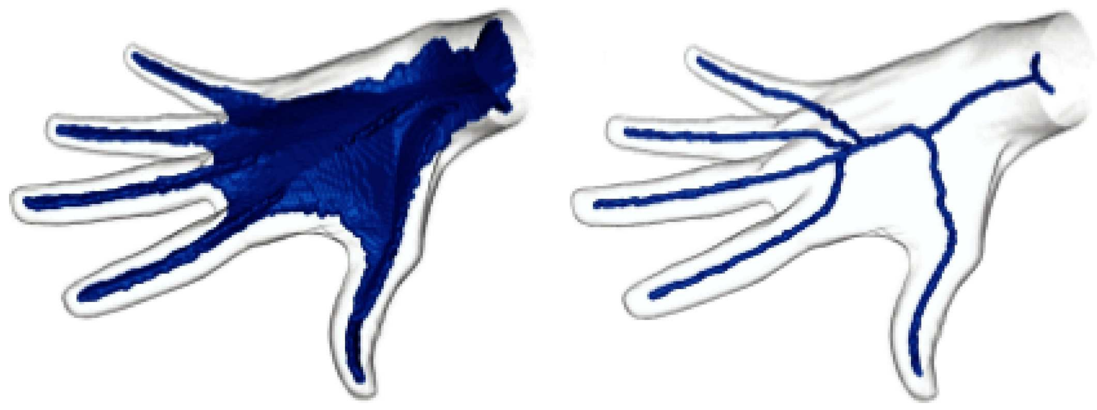


Figure 3.5: 3.5a: regular skeleton. 3.5b: skeleton distorted by noise within the shape. 3.5c and 3.5d: skeleton distorted by boundary noise.

3.1.2 Skeletons in 3D

Up to now, skeletons for two-dimensional objects were discussed. Skeletons in the three-dimensional space are quite similar in the main principles, but are more difficult to define.

In general, a skeleton is a reduction of dimensions of the original shape. For the two-dimensional case, this means that two-dimensional planes are mapped to a set of one-dimensional lines, as described earlier. In the three-dimensional case, a distinction between $S^{3,2}$ surface skeletons and $S^{3,1}$ curve skeletons is made.



(a) Surface skeleton **(b)** Curve skeleton

Figure 3.6: Figure 3.6a: Surface skeleton of a hand shape. Figure 3.6b: Curve skeleton of a hand shape. Image source: [Ren09]

The surface skeleton is the direct transfer of the Blum skeleton to three dimensional data and thus, has similar properties. The definition of a skeleton point is adjusted so it can apply to 3D: A skeleton point is not the center of a maximal inscribed disc, but the center of a maximal inscribed sphere. One can imagine that this direct transfer of the blum skeleton to 3D does not directly lead to curves, like in 2D, but to surfaces within the 3D shape. Thus, the surface skeleton is a set of manifolds, called *sheets* [Ren09]. These sheets can be 2D surfaces and curves. Curves occur mainly in object parts with tubular form, while surfaces can be expected in more flattened object parts. An example for a surface skeleton is shown in figure 3.6a. The figure shows that the surface skeleton has a 2D surface structure inside the palm of the hand, while the fingers contain curves, similar to the two-dimensional skeletons. .

In most applications, however, a further reduction of the object is desired. Curve skeletons are a reduction of three-dimensional shapes to only one dimensional curves. The problem with curve skeletons is that they are not well defined, and no mathematical definition has been formulated yet [DS06]. The problem lies hereby in the fact that the most desirable property of the curve skeleton - centeredness - is difficult to define mathematically for 3D curve skeletons. While for tubular shapes, like the fingers in the hand shape shown in figure 3.6a, the centered skeleton is comparatively easy to define and compute, for more complex shapes, it becomes less clear which points are "centered" within the shape.

Curve skeletons are conceptually related to the Blum skeletons, Thus, they share many properties. As curve skeletons lack of a formal definition, however, most properties are more explicit requirements rather than implied by their definition [Ren09].

Centered As mentioned before, this most prominent desired property of curve skeletons is hard to compute, and various approaches have been proposed to define the centeredness of the one-dimensional curve within the three-dimensional object. For example, one possibility is to restrict the curve skeleton to a subset of the surface skeleton [DS06].

Thin Like the two-dimensional Blum skeleton, the curve skeleton should be thin. For the discrete three-dimensional space this means the skeleton should be exactly one voxel thick.

Homotopic The curve skeleton should preserve the original shape's topology. In the three-dimensional space this means that each tunnel in the shape results in a loop in the curve skeleton.

Reconstruction As the curve skeleton preserves the geometry of the original shape to a lesser extent, the reconstruction of the shape based on the curve skeleton is generally not possible. Usually, the curve skeleton preserves only geometric information about salient features in the shape, as they are supposed to reach into all salient parts of the original shape and terminate at prominent points on the shape contour.

Connected Like the two-dimensional Blum skeleton, the curve skeleton is supposed to be connected. That is, there should be no missing voxels ("holes") in the skeleton branches.

3.1.3 Skeletonization in 2D

The process of extracting a skeleton from an object is called *skeletonization*. In the following, skeletonization algorithms are introduced, first for 2D and then also for 3D shapes.

There are in principle four main classes for skeletonization techniques: Grassfire simulations, thinning algorithms, algorithms based on Voronoi diagrams, and distance map based algorithms [BLL07, PSS^{+}03, Ren09].

Grassfire simulations Grassfire simulations actually try to implement the idea of skeletons as proposed by Blum, simulating fire fronts spreading from a shape's contour. Algorithms of this class are rather rare [OK95]. An early work is introduced in [Mon69], where each fire front is represented by a sequence of straight-line segments and arcs. In [LL92], active contours are used to model the fire fronts. The snakes in this approach are controlled by a 3D surface H, specified by a previously computed distance transform. A weight is assigned to the snake that makes it fall down on the slopes of the surface.

Thinning algorithms The general idea of thinning algorithms is to iterativily erase pixels from the shape boundary, until only a skeleton remains. A boundary pixel p is deleted depending on the configuration of the neighbor pixels of p. In each iteration, contour pixels of the shape are inspected for their topological relevance. The idea is to identify those pixels that are essential for representing the shape. Like in other morphological operators, the conditions that dictate whether a pixel can be deleted without changing the object's toplogy are usually encoded by structuring elements, also called kernel or template. A match of the template within the shape causes the center pixel to be deleted.

One of the difficulties in thinning algorithms is that they have to consider global properties of the shape, like connectedness, though they operate only in a 3×3 window. Another problem is to locate skeleton end points as these must not get deleted [DP81]. Thus, a pixel is deletable, if it is no end point, and if no connectivities in the original image are destroyed [LLS92]. That is, if two pixel p and q were connected in the original shape, and both pixels are not deleted during the process, they need to stay connected after the deletion. For example, the erosion shown in figure 3.7 is no valid thinning operation, as parts of the original shape get disconnected during the process.

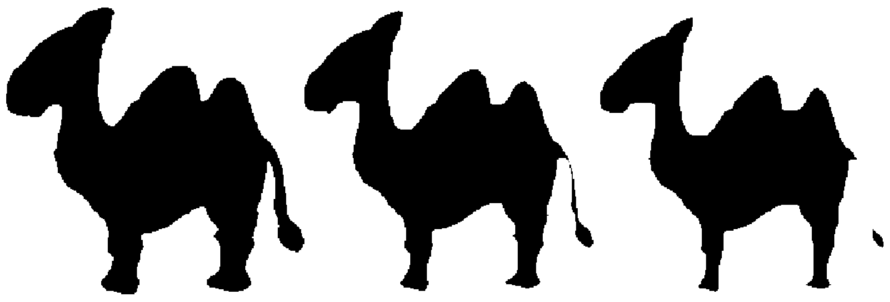


Figure 3.7: Parts of the object get disonnected in the reduction process. Thus, the erosion shown is no valid thinning operation.

The process of deleting pixels is repeated until no pixels can be deleted any more without violating the operator's conditions. Obviously, the result of the thinning process in sequential algorithms heavily relies on the order how pixels are removed. Thus, the result of an iteration depends not only on the result of the previous iteration, but also on the removed pixels in the current iteration. Parallel algorithms deal with this issue by inspecting multiple points for deletion. That way, in parallel algorithms the removal of pixels depend only on the result of the previous iteration. Figure 3.8 shows an example of the thinning process on a simple binary image.

Thinning algorithms usually guarantee connected skeletons. But mostly, the result is not perfectly thinned as not all unnecessary pixels can be deleted in the thinning process, and unimportant skeleton branches may remain. Thus, additional postprocessing methods are needed.

Voronoi diagrams Another class of skeletonization algorithms are techniques based on Voronoi diagrams. A Voronoi diagram paritions the given space in so called *Voronoi cells.* Usually, the input is a set of points in a plane, called generating points or *sites.* A Voronoi cell of a site is then the set of all points on the plane, that are closer to this site than to any other site on the plane. In the context of skeletonization, boundary points are used as sites. The Voronoi edges located completely outside the shape are discarded, as are edges that intersect with the shape boundary. All remaining Voronoi edges form the skeleton. Depending on the number of generating points, the skeleton resulting from Voronoi based approaches consist of more or less long straight lines. Figure 3.9 shows an

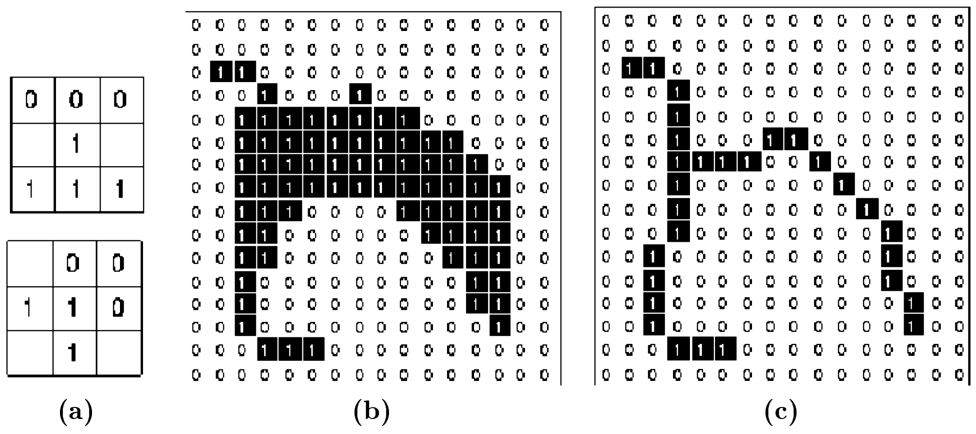


Figure 3.8: Figure 3.8a: Structuring elements. The image is thinned sequentially by the structuring element shown, and then with the remaining six 90 °rotations of the two elements in each iteration. This process is repeated until no pixels are removed any more. Figure 3.8b: Original shape. Figure 3.8c: Skeleton, after the thinning process. Image source: [FPWW04].

example of a skeleton, extracted from a Voronoi diagram. The most severe problem in Voronoi diagram based methods is the choice of generating points. The more sampling points were used, the more spurious branches are contained in the resulting skeleton. Besides, the approach is very sensitive to boundary noise. If the number of sample boundary points is to high, the generation of the Voronoi diagram becomes intensive in computational time as well as in memory usage. If too few sampling points are used, it becomes more likely that important boundary points are not taken into account for skeleton generation.

Distance maps The idea in distance map based approaches is to first compute the distance map for the input shape. Different distance functions have been used for that purpose. The euclidean distance map is neither trivial nor efficient to compute, but the euclidean distance can be substituted by simpler distance functions like chessboard or Manhattan distance. This usually leads to less complex computations, but reduces the distance precision. The skeleton can then be extracted from the computed distance map. The general idea is as follows: The distance map can be seen as a height map, and skeleton points can be found at local maxima. This might lead to single points on the distance map's peaks, which requires additional post processing methods to guarantee connected results. [Cha07] introduce an algorithm for finding the ridges in the distance map by creating a sign map for an object's distance map: The rows of the distance map are scanned from left to right. If a pixel has a higher value than its left neighbor, this pixel in the sign map is marked by a "+". If both pixels have the same value in the distance map, the

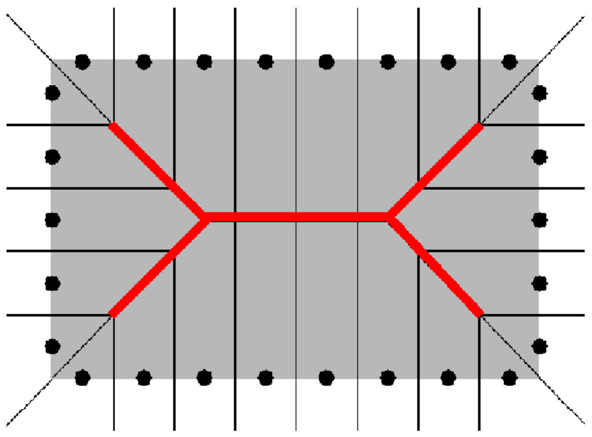


Figure 3.9: Example for a Voronoi diagram based skeleton. The original shape (a rectangle) is shown in grey. The black dots are sampled on the shape boundary as generating points. The corresponding Voronoi cells are indicated by black lines. Voronoi edges outside the shape, as well as Voronoi edges intersecting with the shape boundary are discared. The remaining edges form the skeleton, marked in red. Image source: [Pal].

current pixel is marked with a “0”, and it is marked with a “-” if the left neighbor has a higher value than the current pixel. The process is repeated for the columns of the distance map. The resulting sign maps are visualized in figure 3.10.

These sign maps are then analysed to find ridges. The most obvious indicator for a ridge is a “+-”-pattern in the sign map. To handle two neighboring points having the same distance to the object boundary, the “+0-” pattern is also included as an indicator for a ridge. With this technique, the result is still not connected, so the gaps have to be filled by tracing the maximum gradient paths around the found ridges.

Each of the skeletonization algorithms have their advantages and disadvantages. Neither of them can preserve all desired skeleton properties as discussed before. [CSM07] examine the classes of skeletonization algorithms with respect to their preservation of some of the desired properties. Table 3.1 shows an overview of their conclusions.

The examination is based on the general idea of the skeletonization algorithm classes. However, variations of these classes can have subtle differences in some of these properties. For example, the centeredness of the skeleton produced by a skeletonization algorithm based on Voronoi diagrams depends on the density of the sampling points.

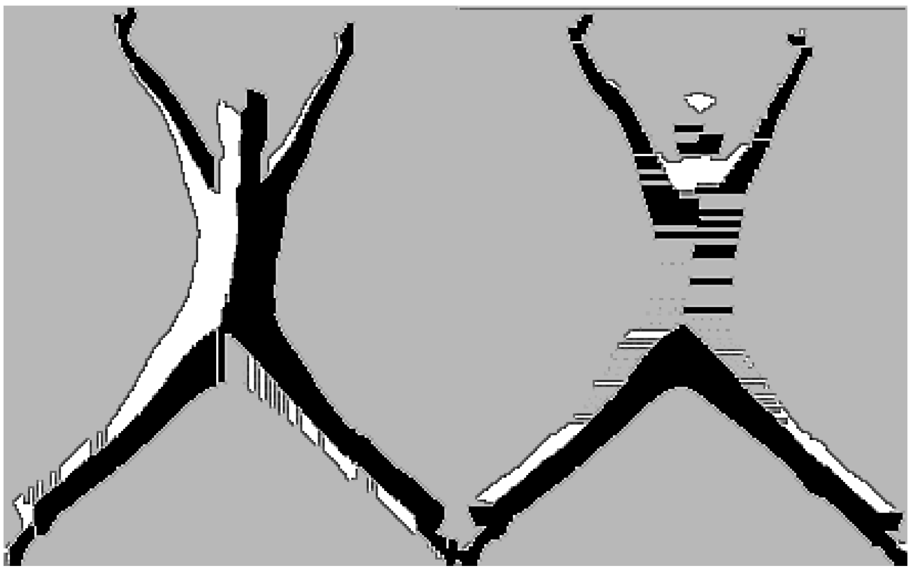


Figure 3.10: Both sign maps based on the euclidean distance map. The left figure shows the sign map for the image's rows, the right image shows the sign map for the image's columns. Image source: [LKMT]

3.1.4 Skeletonization in 3D

For 3D, skeletonization algorithms are based on similar concepts as the 2D algorithms. The skeletonization approaches in 3D can be classified into the same classes as the 2D algorithms: Grassfire simulations, thinning algorithms, algorithms based on Voronoi diagrams, and distance map based algorithms.

Grassfire simulations [QSO04] model the skeletonization process after the original idea of Blum and simulate the wave propagation within the shape. Where two wave fronts meet, the point is marked as a skeleton point. This simple approach leads to surface skeletons, as in complex objects, the waves are rarely expected to clash in one centerline. [SLSK07] generate curve skeletons by using a grassfire simulation: They use model deformations to observe competing fire fronts to extract multiple centered lines, approximations for the actual skeleton. By keeping track of the reconstruction of the original shape, the "best" centered axis is chosen from the set of generated lines.

Thinning algorithms The idea of thinning algorithms in 3D is similar to the thinning algorithms in 2D, namely to iteratively delete those voxels from the shape border that satisfy specific geometric and topological contraints. The thinning algorithms in 3D however can also be classified by the type of skeleton they produce.

	Thinning	**Voronoi**	**Distance Map**
Centered	-	-	✓
Thin	-	✓	✓
Homotopic	✓	✓	-
Reconstruction	-	-	-
Connected	✓	✓	-
Robust	-	-	-
Invariance	-	✓	✓

Table 3.1: An overview of the skeletonization algorithms and the properties they guarantee in the resulting skeletons.

Some algorithms extract the surface skeleton [Ber95, Pal08] while others directly compute the curve skeleton [PK98, XTP03, LG07]. Like in 2D, the difficulty is to preserve the global connectivity by only inspecting the $3 \times 3 \times 3$ neighborhood of a voxel. Usually, a voxel is tested against a set of template voxels to test if it can be removed without changing the topology. A schematic visualization of the thinning process is shown in figure 3.11.

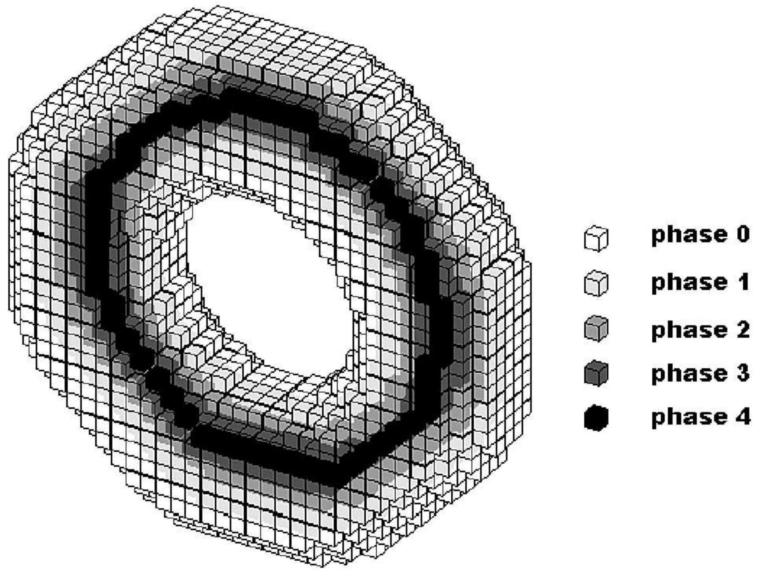


Figure 3.11: Thinning of a 3D shape. In each phase, voxels are erased from the shape border, until only a center line - indicated in this figure by the dark voxels - remains. Image source: [Pal].

Voronoi diagrams A skeletonization approach for 3D meshes based on Voronoi diagrams is introduced in [HBK01]. From the computed Voronoi cells, the Voronoi poles are extracted, resulting in a point cloud within the volume. Two Voronoi poles are then connected by an edge if the corresponding mesh vertices also were connected by an edge. Thus, each mesh triangle is mapped to a triangle of Voronoi poles, forming a surface skeleton within the shape.

Distance maps Another way to skeletonize 3D volumes uses the volume's distance transform as basis. [TW02] compute the skeleton by computing the 2D-distance transform for each axis-parallel 2D slice in the 3D volume. In the next step, the resulting three volumes are intersected voxel by voxel, and that way, the 3D centerline is obtained. [DWT06] present a distance map based skeletonization algorithm using the GPU. The volume's distance transform is computed and sampled in a 3D texture. The distance value is assigned to the depth channel of the voxel, and the skeleton can then be extracted using the Z-buffer depth test.

3.1.5 Skeleton pruning

Usually, the skeletons generated by the proposed skeletonization approaches contain spurious branches, mostly caused by boundary noise. Thus, the used skeletonization algorithm has to ensure stability of the skeleton. There are generally two main approaches to handle this problem: The first one is to preprocess the input shape, typically by smoothing. The idea is to eliminate noise before the skeleton is computed. The other possibility is to edit the skeleton during or after the skeletonization process and to erase unimportant skeleton branches based on a predefined importance measure. The process of finding and deleting spurious skeleton parts is called pruning. Generally, the pruning process can be integrated in the skeletonization process or it can be applied after the skeletonization process. Figure 3.12 shows an example for a skeleton before and after pruning.

The problem is to find a meaningful significance measure for skeleton points and skeleton branches, respectively. Spurious branches should get deleted completely, while branches holding important information about the object's geometry have to be preserved. An introduction to pruning techniques is presented in [SB98].

In [TH02] an area-based pruning method is proposed. The idea is to analyse if features on the object boundary that lead to additional skeleton branches are to be considered noise or significant features. For this purpose, the features are defined as the set of triangles associated with any subtree of the skeleton. An example is shown in figure 3.13. Features that cover an area smaller than a user-defined threshold are considered as noise, and thus, the skeleton branches caused by these features can be deleted.

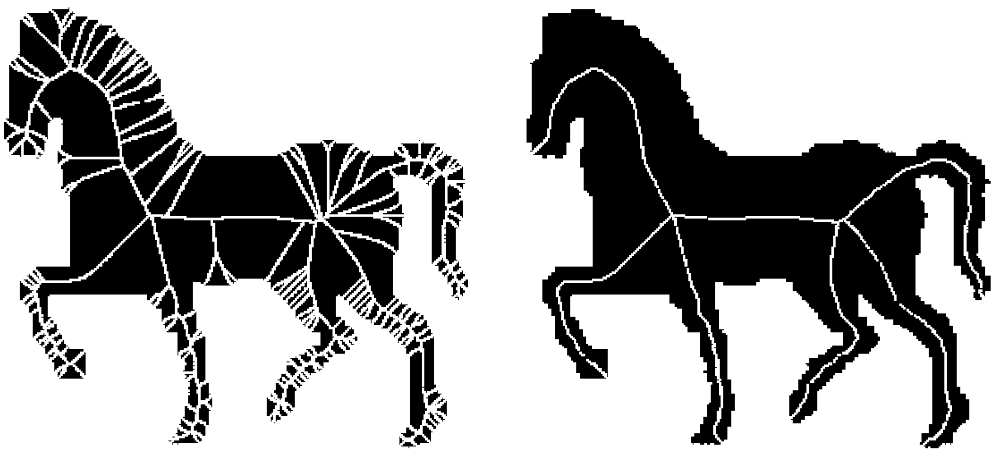


Figure 3.12: Left: skeleton containing spurious branches. Right: Skeleton after pruning. Image source: [BLL07]

[OI92] propose several relevance measures for skeleton points based on the boundary curve connecting a skeleton point's feature points. This approach is based on the fundamental observation that skeleton branches that lie deep within the object are less sensitive to boundary noise than the outer parts. The length of the shortest path from one feature point to the other along the boundary is considered to be an indicator for the importance of a skeleton point. If this distance is long, the concerned skeleton point is likely to lie deep inside the object and thus may not be removed. If the distance is below a threshold, this indicates that the skeleton point was caused by noise and can be removed. Connectivities are preserved in this approach, but pruning based on the residual function might lead to the loss of end nodes.

[SBH$^+$11] introduce a siginificance measure based on the bending potential ratio. The bending potential ratio is defined for a skeleton point p and its feature points q_1 and q_2. An isosceles triangle with the base $\overline{q_1 q_2}$ is defined with the additional vertex g such that

$$d(g, q_1) = d(g, q_2) = \frac{1}{2} l(q_1, q_2) \tag{3.2}$$

where $l(q_1, q_2)$ is the arc length between q_1 and q_2, measured on the boundary. g is then called the ghost point of the contour segment between q_1 and q_2. The bending potential ratio is defined

$$\epsilon(p, q_1, q_2) = \frac{h_g}{h_p} \tag{3.3}$$

where h_g is the height of triangle $q_1 g q_2$, and h_p ist the height of triangle $q_1 p q_2$.

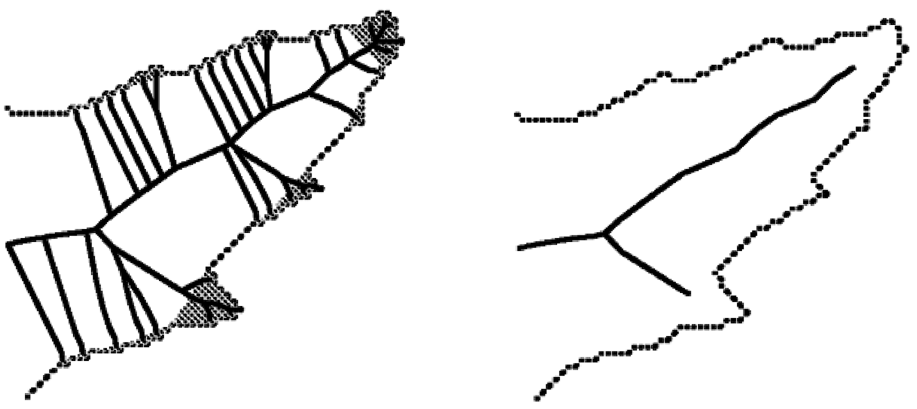


Figure 3.13: Area-based pruning. The left image shows the original, unpruned skeleton. The shaded regions in the right image are smaller in area than the given user-defined threshold. Image source: [TH02]

As h_g is influenced by the length of the contour segment, it holds information about the bending of the contour segment. The assumption is, the higher h_g is the more significant is the corresponding skeleton point p.

The bending potential ratio contains not only local information of the contour segment between q_1 and q_2, encoded by h_g, but also the context where it is located. If a skeleton point is located on a broad part of the object, it is more likely to be interpreted as a feature if only the arc length of the contour segment is taken into account. Therefore, h_p is included to avoid hastily considering skeleton points as features. Figure 3.14 shows a visualization of the bending potential ratio.

Since the skeleton to work on is the fundament for further analysis, the used skeletonization and pruning algorithm have to be chosen carefully. For the experiments in this diploma thesis, the pruning algorithm introduced in [BLL07] has been used, as it promises to preserve the shape topology, while removing most spurious branches. Besides, the applicability of this approach has been proven in [BL08]. The proposed technique is based on the observation that shapes usually can be segmented into a few regions or visual parts, and for each of these regions, only one skeleton branch is needed to represent it. The main idea of this pruning approach is therefore to remove all skeleton points whose feature points are located on the same contour segment. The difficulty is to partition the contour in a meaningful way. An algorithm for simplifying shapes called *Discrete Curve Evolution (DCE)* proposed in [LL99b] and [LL99a] is used for this purpose.

The fundamental observation of DCE is that, due to finite image resolution, a shape boundary in an image can be represented as a finite polygon. During the process of DCE, the polygon vertices with the smallest shape contributions are removed recursively from the shape boundary. Thus, in every evolution step, a

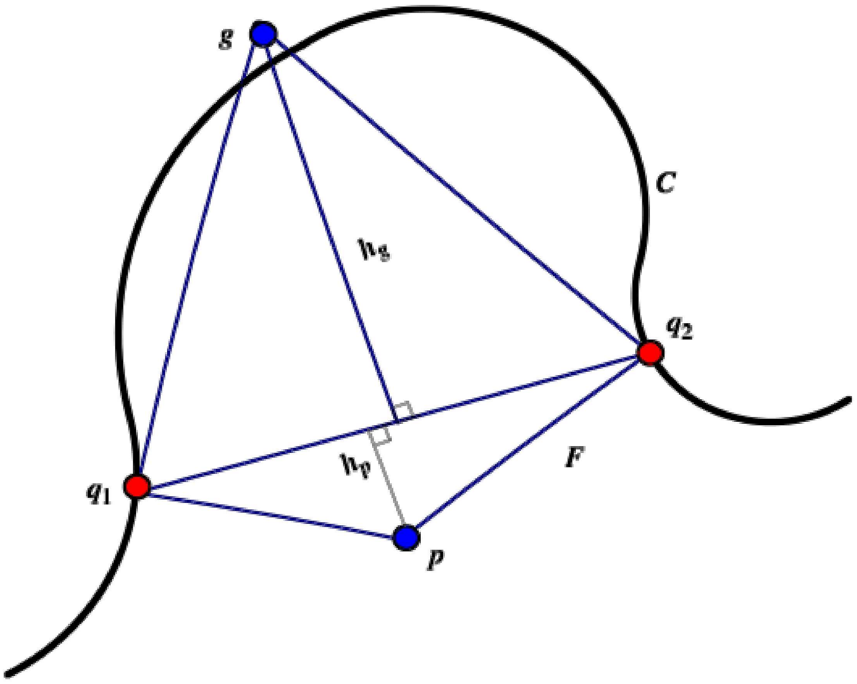


Figure 3.14: Definition of the bending potential ratio. p is the skeleton point to be examined for relevance, q_1 and q_2 are the feature points of p. Image source: [SBH+11]

polygon is obtained in which the less important vertices of the previous polygon are removed, until only a subset of boundary vertices remains that best represents the original shape.

The unanswered question remaining is what "important" means in this context. In [LL99b] a relevance measure for a vertex v, depending on v and its two neighbor vertices u, w in the polygon P^i of the current evolution process i, is introduced. Intuitively, the relevance measure takes into account the shape contribution of vertex v to the current polygon. Formally, it is given by

$$K(v,u,w) = K(\beta, l_1, l_2) = \frac{\beta l_1 l_2}{l_1 + l_2} \tag{3.4}$$

where β is the turn angle at v in P^i, l_1 is the length of the polygon edge $\overline{vu}$ and l_2 is the length of the polygon edge $\overline{vw}$. The assumption is, the higher the value of $K(u,v,w)$ is, the larger is the shape contribution of arc $\overline{uv} \cup \overline{vw}$ to the polygon in the current evolution step. The vertices with the lowest relevance value can be deleted.

The pruned skeleton is then computed with respect to the obtained DCE segments, while concave vertices are ignored. The deletion of a vertex v results in the deletion of a complete skeleton branch, namely of the branch ending at v. As the remaining boundary points remain at their position, the skeleton is not displaced by this process. Figure 3.15 shows an example for the simplification of the original shape. By deleting boundary points, also skeleton branches are deleted.

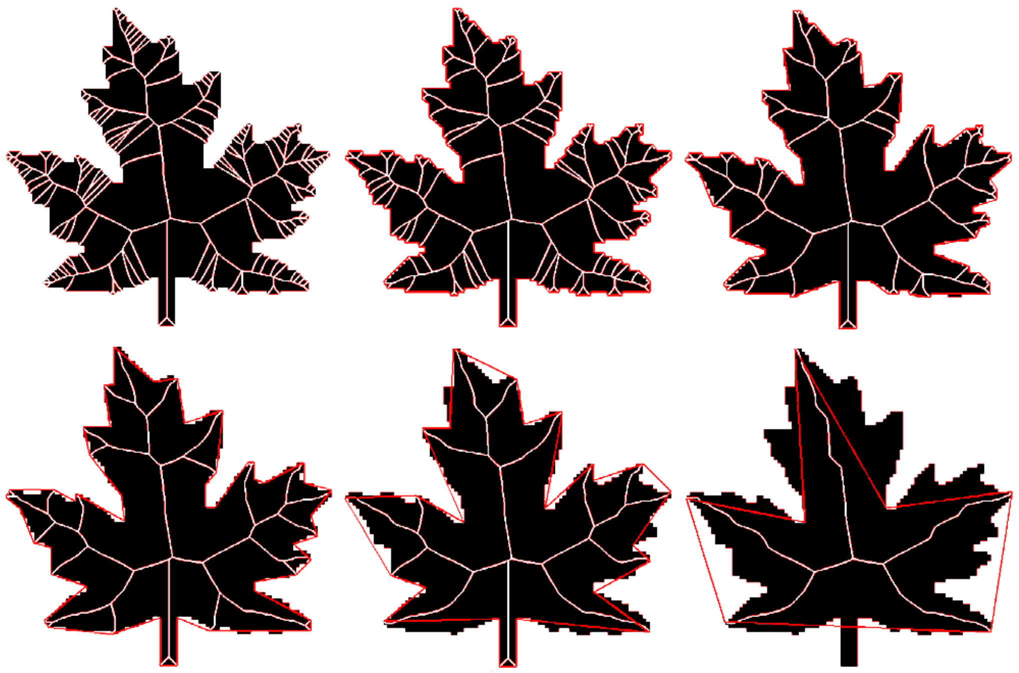


Figure 3.15: Skeleton pruning by contour partitioning. The original contour of the shape gets simplified in each evolution step. The simplified polygon is marked in red. The deletion of a vertex results in the deletion of the skeleton branch ending at this vertex. Image source: [BLL07]

3.2 Matching of time series

The concept of time series can be used in various applications, such as word recognition and tracking of moving objects in video sequences. As will be shown later, also a skeleton's end nodes can be treated as time series. Therefore, in this section the matching of time series is introduced.

A time series is a sequence of data points, that is, real numbers. One of the main problems when dealing with time series is that the time axis might be partially streched or compressed. For example in speech recognition, speakers might vary in their pronouncation of words especially in the extent of vocals.

Various approaches have been proposed to deal with the matching of time series. The general idea is usually to arrange both time series a and b in a matrix R, where each matrix cell R_{ij} holds the information about the cost to align the two elements a_i and b_j. The best alignment for a and b is then found by mapping elements of a to elements of b which is then the problem of finding the cheapest path through the matrix.

One problem when using *dynamic time warping* (DTW) [BC94] is that the exact beginning and ending of the time series have to be known for both sequences which is not always the case. *Minimal Variance Matching* (MVM) [LMW$^+$05] deals with this problem by treating the cost matrix R as a directed graph.

However, the problem remaining in both approaches is the dealing with outlier elements in one the two time series. An outlier element is an element in one of the two series that cannot be matched to any element in the other series. Outlier elements can have a significant effect on the overall alignment result. Every element of series a must correspond to some element of the series b, and vice versa. The *Longest Common Subsequence* (LCSS) [DGM97] approach deals with these outlier elements by aligning subsequences, found by moving a window over the two sequences. The performance of this approach depends largely upon the configuration of a manually set threshold.

An extension of DTW and MVM, respectively, that deals with the problem of outlier elements, is the *Optimal Subsequence Bijection* proposed in [LWKTM07]. As OSB also is used in the matching algorithm proposed in 4, this approach will now be explained in detail.

Given are two time series $a = (a_1, \ldots, a_m)$ and $b = (b_1, \ldots, b_n)$. "The goal of OSB is to find subsequences a' of a and b' of b such that a' best matches b'" [LWKTM07]. First, the two time series are arranged in a cost matrix R. The elements of a are charted in the matrix rows, the elements of b are charted in the matrix columns. The matrix cell R_{ij} contains the similarity value for a_i and b_j. There are no restrictions on the distance function used. One possibility is to use the difference between a_i and b_j:

$$r_{ij} = (b_j - a_i) \tag{3.5}$$

The idea is to find the least-value path through the matrix without going backwards, neither in the rows nor in the columns. As in *MVM*, the idea is to treat the cost matrix as a directed acyclic graph. The vertices of the graph are all index pairs $(i, j) \in \{1, \ldots, m\} \times \{1, \ldots, n\}$. Thus, one graph vertex represents one matrix cell. Two vertices (i, j) and (k, l) are connected by an edge if the following conditions hold:

1. $i + 1 \leq k$, that is, rows in the matrix can be skipped, but it is not allowed to go backwards

2. $j+1 \leq l$, that is, columns in the matrix can be skipped, but it is not allowed to go backwards.

The edge weight $w((i,j),(k,l))$ depends upon whether the two vertices (i,j) and (k,l) are directly adjacent in the matrix. In particular, the edge weight w is determined by

$$w((i,j),(k,l)) = \begin{cases} d(a_i, b_j), & \text{if } i+1 = k \text{ and } j+1 \leq l \\ k-i-1 \cdot \textit{jumpcost}, & \text{if } i+1 < k \text{ and } j+1 \leq l \end{cases} \tag{3.6}$$

The first definition covers the case that the two vertices represent two consecutive rows in the cost matrix. The distance between two elements $d(a_i, b_j)$ is defined by the entry in the cost matrix at index (i,j). There is no explicit penalty for skipping columns. The second condition holds if rows are skipped in the cost matrix. The value *jumpcost* in this definition is a constant used as a penalty for skipping rows in the matrix. This constant has to be chosen carefully, as too many elements might be skipped if the penalty is too low, but if *jumpcost* is too high, elements might be forced to align even if they do not correspond. [BL08] propose to compute *jumpcost* as

$$\textit{jumpcost} = \textit{mean}_i(\textit{min}_j(d(a_i, b_j))) + \textit{std}_i(\textit{min}_j(d(a_i, b_j))) \tag{3.7}$$

For every element a_i the closest element b_j is found. For all of these minimum distance values, the mean is computed, and the standard deviation for these minimum distance values is added.

Once the weighted graph is built, the problem of matching the two input time series can be solved by finding the shortest path through the graph, using the shortest path algorithm on a directed acyclic graph. Each path can start at r_{1j}, for $j = 1, \ldots, n-m$, that is, in the first row in the first $n-m$ columns. The end vertex for each path is restricted to r_{mj} with $j = n-m, \ldots, n$. For any correspondence f, the total distance between two time series is then defined by

$$d(a,b,f) = \frac{1}{m} \sum_{i=1}^{m} (d(a_i, b_{f(i)}))^2 \tag{3.8}$$

The following example, extended from [LWKTM07], will show the basic principles of the OSB function. Given are the two time series $a = (1,2,8,6,8)$ and $b = (1,2,9,3,3,5,9)$. First, the two time series are used to create the difference matrix R with $R_{ij} = b_j - a_i$:

$$\boldsymbol{R} = \begin{pmatrix} 0 & 1 & 8 & 2 & 2 & 4 & 8 \\ -1 & 0 & 7 & 1 & 1 & 3 & 7 \\ -7 & -6 & 1 & -5 & -5 & -3 & 1 \\ -5 & -4 & 3 & -3 & -3 & -1 & 3 \\ -7 & -6 & 1 & -5 & -5 & -3 & 1 \end{pmatrix} \tag{3.9}$$

Based on this matrix, the directed weighted graph can be constructed as desribed above: The elements in the matrix are represented by graph nodes. Nodes are connected by an edge if this edge would not mean that one can traverse the matrix in a wrong order, that is, going upwards in the rows or backwards in the column. To each edge a weight is assigned: if the edge connects two nodes (i, j) and (k, l) that represent two consecutive rows in the matrix, the edge weight is given by the value of matrix entry (k, l). If the edge connects two nodes that are at least two rows in the matrix apart from each other, the edge weight is given by jumpcost, multiplied by the number of skipped rows. To make it possible to skip also the border elements (that is, the first and last row and the first and last column), dummy rows and column are added to the beginning and end of the matrix. Figure 3.16 shows the edge creation for entry (0,0) in the matrix to the first few graph nodes, the remaining nodes are skipped in this figure to preserve a better overview.

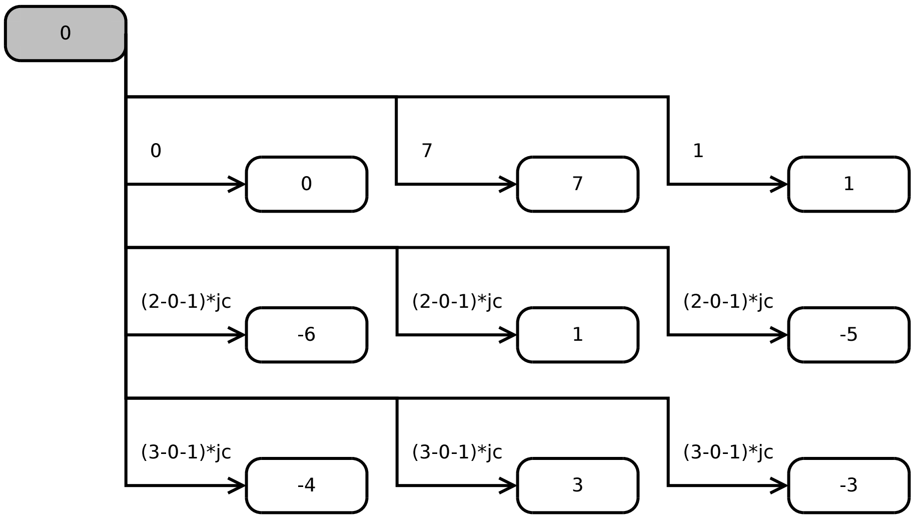


Figure 3.16: An example for the construction of a directed acyclic graph, based on the path distance matrix in 3.9. The graph nodes are the elements of the matrix. The figure shows the edge creation from the upper left entry (0,0) in the matrix to the nearest other graph nodes (the other nodes are not shown to ensure a clear arrangement). The abbreviation *jc* stand for the constant *jumpcost*. Observe that skipping columns in the matrix is not explicitly punished, in constrast to the skipping of rows in the matrix.

For all other nodes in the graph the procedure is the same as shown in figure 3.16. Once the graph is built, the shortest path algorithm on a directed acyclic graph can be applied to find the cheapest path through the matrix. The path found by this approach is highlighted in the following matrix:

$$\boldsymbol{R} = \begin{pmatrix} \colorbox{lightgray}{0} & 1 & 8 & 2 & 2 & 4 & 8 \\ -1 & \colorbox{lightgray}{0} & 7 & 1 & 1 & 3 & 7 \\ -7 & -6 & \colorbox{lightgray}{1} & -5 & -5 & -3 & 1 \\ -5 & -4 & 3 & -3 & -3 & \colorbox{lightgray}{-1} & 3 \\ -7 & -6 & 1 & -5 & -5 & -3 & \colorbox{lightgray}{1} \end{pmatrix} \tag{3.10}$$

The highlighted entries are the corresponding elements of the time series. In this example, no rows are skipped. Finally, the overall distance between the two sequences is computed as defined in equation 3.8.

3.3 Graph Matching

Many real-world scenarios such as networks, path finding problems or communication flows can be described by graphs. As will be shown later, also the matching of two shapes can be mapped to a graph matching problem. Thus, fundamentals of graph matching will be introduced in the following section. An important field of graph theory research, the assignment problem, will be discussed in section 3.3.2.

3.3.1 Terms and definitions

Formally, a graph is defined as follows [Die05, BM76, Jun07]: A graph G is an ordered pair $G = (V, E)$ of sets, where V is a non-empty, finite set of vertices, and a set of edges E where the elements of E are 2-element subsets of V, with $V \cap E = \emptyset$. Thus, a graph consists of vertices that are connected by edges. Two vertices u and v are connected if there is an edge $e \in E$ with $e = (u, v)$. If two vertices are connected by an edge, they are said to be *adjacent* or *neighbors*. A vertex in a graph is said to be incident with an edge if it is connected to another vertex by this particular edge.

The number of vertices in a graph is called its order, denoted by $\mid G \mid$. If all edges $e \in E$ are an unordered pair of vertices $u, v \in V$ the graph is said to be undirected. If all edges $e \in E$ are an ordered pair of vertices, the graph is said to be directed.

Edges can be defined uniquely by the two vertices they connect. However, additional information can be assigned to the edges, called weights or costs. A graph with weighted edges is called a weighted graph. Weighted graphs are used in many contexts, such as routing or matching problems.

A graph path is a non-empty graph such that $V = \{x_0, x_1, \ldots, x_k\}$ and $E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$. A path length in unweighted graphs is determined by the number of vertices visited in the path. In weighted graphs the path length is usually determined by the sum of the weights of the traversed edges.

Graphs can be classified according to their properties. One class of graphs is the r-partite graphs. A graph $G = (V, E)$ is called r-partite if the vertices of G can be partitioned into r classes such that each edge has its ends in different classes, with $r \in \mathbb{N}+$ and $r \geq 2$. That is, two vertices within the same class must not be adjacent.

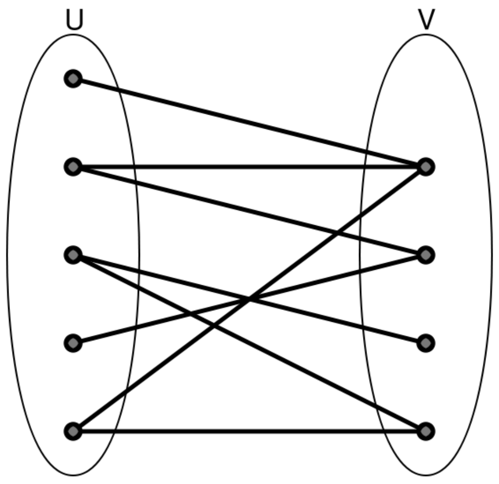


Figure 3.17: Example for a bipartite graph. Image source: [Wik11].

A special case of the r-partite graph is the bipartite graph with $r = 2$. In particular, it has been shown that the object matching in image processing can be mapped to the bipartite graph matching problem. Figure 3.17 shows an example for a bipartite graph.

3.3.2 The assignment problem

A matching M in the context of graphs, also called correspondence, is a subset of all edges in a graph G such that no edges share the same end nodes. In other words, all edges $e \in M$ are disjunct, meaning none of the edges are incident with the same vertices. If all vertices $v \in V$ are incident with an edge in M the matching is said to be perfect.

The term “matching” is now used in two different contexts. Henceforth, the term will be used for both matching of shape features, as mentioned in chapter 1, as well as for the matching in graph theory. The context should reveal the particular meaning.

A widely spreaded application of graph matching is the matching in a bipartite weighted graph. A matching M in a bipartite graph maps each vertex in one class to the vertices in another class. This problem is better known as the assignment problem. The matching cost $w(M)$ is then determined by the sum of all weights of the matching edges. A matching M is called a minimum weighted correspondence in a bipartite weighted graph G if $w(M) \leq w(M')$ for every possible matching in G.

Kuhn [Kuh55] introduces the assignment problem with the following scenario: Given is a set of persons and a set of jobs. Each person can be assigned to any job by a predefined cost. The goal is to assign exactly one person to exactly one job, such that the sum of all assignment costs is minimal. In other words, the minimum weighted correspondence for the bipartite graph consisting of one vertex set V_1, containing all persons, and one vertex set V_2, consisting of all jobs, is wanted.

Solving this problem by a brute-force algorithm would require computing the costs for each possible assignment. That would be $n!$ possible assignments, depending on the number of nodes to be matched, resulting in exponential runtime complexity. A solution to this problem with a better performance is the *Hungarian Algorithm*, originally proposed as *Hungarian Method* in 1955 by Harold W. Kuhn [Kuh55] and revised by James Munkres in 1957 [Mun57]. Since then, it is known as *Hungarian Algorithm* or *Kuhn-Munkres-Algorithm.*

Given is a cost matrix R with dimensions $n \times n$, with n being the number of persons and jobs to be assigned, respectively. The entries of R, r_{ij}, denote the assignment cost of person i for the job j. A set of elements of a matrix is called independent if neither of them are located in the same row or column. The goal is to find n independent elements so that the sum of the assignment costs is minimal.

The main approach of the algorithm is to iterativelly increase the number of zero elements in the matrix. In detail, the steps of the Hungarian algorithm are as follows [Cas10]:

Step 1 For each row in the cost matrix, find the row minimum entry. Substract this minimum from each entry in the row.

Step 2 For each column in the cost matrix, find the column minimum entry. Substract this minimum from each entry in the column. The matrix resulting from step 1 and step 2 in this algorithm is also called reduced matrix.

Step 3 Cover all zero elements in the matrix with the minimum number of vertical and horizontal lines possible.

Step 4 If the number of covered columns is equal to the number of columns in the matrix, the reduced matrix already contains a unique optimal assignment, and a solution is found. If the number of covered columns is less than the number of columns in the matrix, go to step 5.

Step 5 Find the minimum entry value in the cost matrix that has not been covered. That minimum is substracted from every uncovered number and is added to every number covered with two lines. Go back to step 3 and repeat until all columns are covered.

An example will show the usage of the Hungarian algorithm [Cas10]. Given is the cost matrix shown in the following table:

	Job A	Job B	Job C	Row minimum
Person A	4	2	8	2
Person B	4	3	7	3
Person C	3	1	6	1

Each entry in the cost matrix describes the cost to assign the person denoted in the row to the job denoted in the column. The goal is to find an assignment of each person to a job so that each person is assigned to exactly one job, and the assignment cost is minimal.

The row minima are charted in the last column. First the matrix has to be reduced, according to step 1 and 2 in the algorithm. First, for each row, the row minimum is substracted from each entry in the row, leading to the following matrix:

	Job A	Job B	Job C
Person A	2	0	6
Person B	1	0	4
Person C	2	0	5
Column minimum	1	0	4

The column minima are charted in the last row. In step 2 of the algorithm, the column minima are substracted from each entry in the column, resulting in the following cost matrix:

	Job A	Job B	Job C
Person A	1	0	2
Person B	0	0	0
Person C	1	0	1

This matrix is referred to as the reduced matrix. Now, the minimum number of lines is used to cover all zero elements in the reduced matrix. Vertical and horizontal lines are possible. There is no elegant solution for finding the optimal covering. In principle, a trial-and-error procedure has to be done.

	Job A	Job B	Job C
Person A	1	0	2
Person B	0	0	0
Person C	1	0	1

As one can see, the minimum number of lines needed to cover all lines is 2, which is smaller than the number of columns which is 3. Thus, no unique assignment has been found yet.

The minimum uncovered number m in the reduced matrix is 1. m is now substracted from every uncovered element in the matrix, and is added to every element covered twice, leading to the following matrix:

	Job A	Job B	Job C
Person A	0	0	1
Person B	0	1	0
Person C	0	0	0

Step 3 is then applied again to the resulting matrix: All zero elements in the matrix are covered with the minimum number of lines possible. In this case, 3 lines are needed.

	Job A	Job B	Job C
Person A	0	0	1
Person B	0	1	0
Person C	0	0	0

As the number of lines needed to cover all zero elements is equal to the number of columns in the matrix, unique assignments can be found in the matrix. As there is no row or column with just one zero elements in the matrix, one can start by choosing an abritary zero element. For example, *Person A* can be assigned to *Job A*, leaving Person B and C and Job B and C unassigned. As the only zero element for *Person B* is *Job C*, and the only zero elements for *Person C* is *Job B*, *Job C* is assigned to *Person B*, and *Job B* is assigned to *Person C*. The minimum weighted correspondence in this example thus is {(Person A, Job A), (Person B, Job C), (Person C, Job B)}. The total costs for this assignment can be computed by the sum of all matching costs in the original cost matrix.

In this example, the total matching cost is 12.

	Job A	Job B	Job C
Person A	④	2	8
Person B	4	3	⑦
Person C	3	①	6

Actually, in most scenarios, more than one solution is possible. Also in this example, three minimum weighted correspondences can be found. If as initial assignment the assignment (Person A, Job B) is chosen, both Person B and Person C can be assigned with minimal cost to the remaining to jobs, resulting also in a matching cost of 12.

3.4 Rating of retrieval systems

A system that performs a similarity search in a database for a query given by the user is called an information retrieval system [Sch06]. In order to evaluate the matching algorithm introduced in chapter 4, a simple retrieval system was implemented. The challenge in information retrieval is that the similarity search is performed under semantic aspects. In contrast to conventional data retrieval systems, for example in relational databases, the search conditions are not explicitly known. In data retrieval systems, all results are syntactically fully consistent with the query. Usually, the amount of result documents in a retrieval system is limited by the total amount of returned documents or by some threshold for the similarity value.

A common way to rate a retrieval system is the computation of precision and recall values. For this purpose, one distinguishes between relevant and irrelevant documents. Moreover, one does not take into account that the result list of returned documents is an ordered list [Sch06]. Each document is then classified in one of the following classes:

- *correct alarms (ca):* this class contains all documents that are relevant to the query, and the retrieval system rated them as relevant.
- *correct dismissals (cd):* this class contains all document that are not relevant to the query, and the retrieval system rated them as not relevant.
- *false alarms (fa):* this class contains all documents that are not relevant to the query, but the retrieval system rated them as relevant anyway.
- *false dismissals (fd):* this class contains all documents that are relevant to the query, but the retrieval system rated them as irrelevant.

The quality measure *precision* P_q describes to what extent the documents found by the retrieval system are relevant. The precision P_q is computed by

$$P_q = \frac{ca}{ca + fa} \tag{3.11}$$

The qualitiy measure *recall* R_q describes to what extent relevant documents in the database where found. The recall R_q is computed by

$$R_q = \frac{ca}{ca + fd} \tag{3.12}$$

Both quality measures always depend on a query q. The computation of both measures require a manual relevance rating for the documents by the user. Both precision and recall values lie between 0 and 1. The higher the value for precision and recall is, the "better" is the retrieval system . The "perfect" retrieval system would have a precision and recall value of 1 for any query. Usually, precision and recall are not only computed for one single query but for multiple queries. In order to evaluate a retrieval system the mean value for both precision and recall for multiple queries is computed.

Both precision and recall not only depend on the query, but also on the number of result documents. If the retrieval system always returns all documents stored in the underlying database, the recall value will always be 100 percent, while the precision value in this case would be very low [Sch06]. If only the document the most similar to the query is returned, the precision value is likely to be 100 percent, but the recall value is very low. A common way to deal with this behaviour is the combined *precision-recall-diagram* which treat the precision values as a function of the recall values, as introduced in [Sch06]. As more precision values are possible for each recall value, only the best precision value is displayed, all other values are ignored.

Chapter 4

Path Similarity Skeleton Graph Matching

In [BL08] an algorithm is presented to match two-dimensional object silhouettes based on their skeletons. In particular, one skeleton's end nodes are matched to the end nodes of another shape's skeleton. The skeleton end nodes are salient features of an object as they are important for holding the shape's geometry information: Skeleton branches are supposed to end in significant visual parts of the shape. Thus, it is a reasonable assumption that each end node can be mapped to its counterpart in the other skeleton. Besides, end nodes are interesting features as they are part of the shape contour as well as part of the skeleton.

The matching is done by comparing the similarity between the shortest paths between the end nodes. Correspondences between the end nodes are established. Furthermore, the total similarity between two shapes is calculated. A detailed description of the algorithm follows in the next section. The example of the two bird shapes in figure 4.1 will guide through the explanations. The implementation of the algorithm is evaluated in section 4.2.

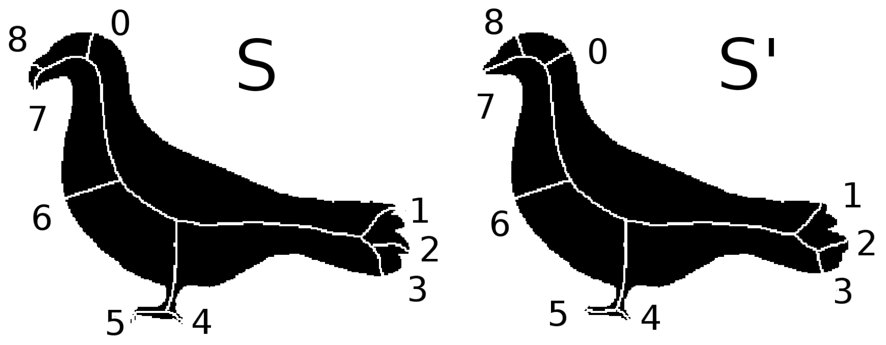


Figure 4.1: Example for two skeletons to be matched.

4.1 Algorithm outline

The input of the algorithm is four binary images: Two binary images containing the objects' silhouettes, and two binary images containing the objects' skeletons. The goal is to find correspondences between distinctive points of the two objects - the skeleton's end nodes -, as well as finding the total costs to match these two objects.

As a preparation, the skeletons are ordered by the number of their end nodes: the skeleton with fewer end nodes is referred to as the query skeleton, the skeleton with more end nodes is referred to as the target skeleton. In the example shown in 4.1, both skeletons S and S' have the same number of end nodes, so they don't have to be reordered. S is thus referred to as the query skeleton, while S' is referred to as the target skeleton.

The algorithm can be split into three steps, which are described in detail in the following sections.

4.1.1 Skeleton representation

The algorithm uses a special representation for skeletons that not only incorporates the skeleton characteristics, but also the shape's contour information.

A key concept in the algorithm is the usage of information about skeleton paths. A skeleton path $p(v_m, v_n)$ in a skeleton is the shortest path between a pair of end nodes v_m and v_n, with the limitation that all points on the shortest path have to be skeleton points. Figure 4.2 shows a series of images, displaying all skeleton paths emanating from one example end node.

The skeleton paths can be found by constructing a weighted skeleton graph for the skeleton. The edge weight for an egde connecting two vertices in the graph is defined by the length of the skeleton branch connecting the two corresponding skeleton points in the original skeleton. It is then possible to apply a shortest path algorithm, like Dijkstra's algorithm, on the resulting graph. [BL08] choose a special representation for the skeleton paths that does not only include information about the skeleton itself, but also information about the object contour. For this purpose, a skeleton path $p(v_m, v_n)$ is sampled with M equidistant points. Since all points on a skeleton path are skeleton points, they are the center of a maximal inscribed disc within the object contour. The radius of the maximal disc $R_{m,n}(t)$ at this point is obtained, for each point t of the sampled skeleton points. Thus, for each sample point, the distance to its feature point is known.

The distance of a point t to its feature points is not exactly calculated, but approximated by the distance transform $DT(t)$. Afterwards, the distance is nor-

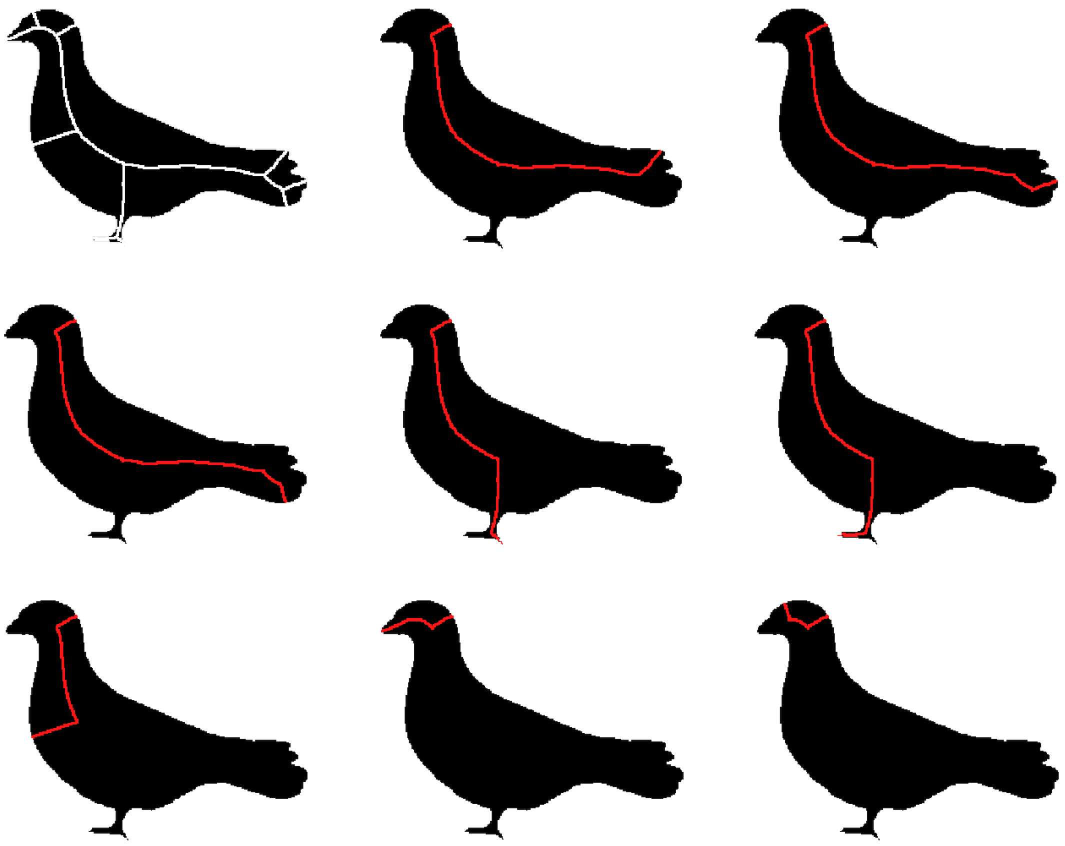


Figure 4.2: The whole skeleton (top left) and all skeleton paths emanating from end node 0, the end node in the birds's head

malized to make the method invariant to the scale. Finally, the distance is approximated and normalized as follows:

$$R_{m,n} = \frac{DT(t)}{\frac{1}{N_0}\sum_{i=1}^{N_0} DT(s)} \tag{4.1}$$

The quotient involves the distance transform values of all points within the object: N_0 is the number of pixels in the original shape, and s_i varies over all N_0 pixels in the shape. Thus, the average distance transform value of all object pixels is involved in the calculations. This step makes the method invariant to scale.

That way, an ordered list of M distance values is obtained for each skeleton path. All distance values are noted in a vector, called the path vector.

$$R_{m,n} = (R_{m,n}(t))_{t=1,2,\ldots,M} = (r_1, r_2, \ldots, r_m) \tag{4.2}$$

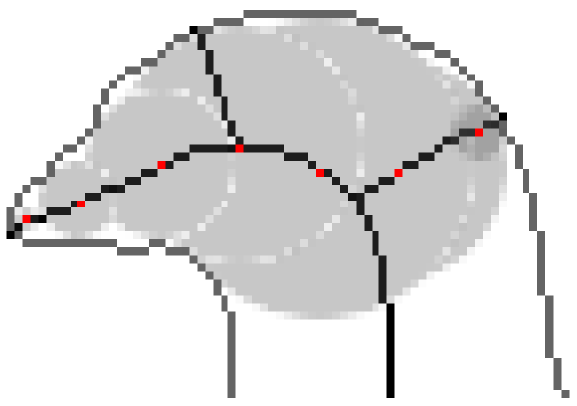


Figure 4.3: Sampling of a skeleton path. The sampling points are indicated in red. The distance to their feature points is indicated by the grey circles. For the skeleton path representation, the distance of the skeleton points to their feature points is measured and noted in the skeleton path vector. The sampling points are equidistant.

Figure 4.3 schematically shows the sampling of the skeleton path from end node 0 to end node 7 in skeleton S in figure 4.1. The resulting skeleton path vector is used to describe the skeleton path.

4.1.2 Dissimilarity between end nodes

In the previous step, a compact representation for skeleton paths was obtained, including additional contour information. This information about the skeleton paths will be used to express the similarity between two end nodes. In particular, the dissimilarity between the skeleton paths emanating from a pair of end nodes is used to compute the similarity between these two end nodes.

Skeleton Path Dissimilarity The remaining question to be dealt with is still what "dissimilarity" means here. The dissimilarity between two skeleton paths, called the path distance, is based on the obtained path vectors. Let r and r' be the path vectors for two skeleton paths $p(u,v)$ and $p(u',v')$. l and l' are the length of $p(u,v)$ and $p(u',v')$, respectively. In order to make the approach invariant to scale, the lengths are normalized. Finally, the path distance between the two skeleton paths is defined by

$$pd(p(u,v),p(u',v')) = \sum_{i=1}^{M} \frac{(r_i - r'_i)^2}{r_i + r'_i} + \alpha \frac{(l_i - l'_i)^2}{l_i + l'_i} \tag{4.3}$$

The motivation is that similar skeleton paths are expected to have consecutive skeleton points with similar radii in their maximal inscribed discs. The scaling factor in the denominator in this equation "weights the radii difference with respect to the radii values, that is, if both radii are large, their difference must be significant" [YBYL07]. This factor is motivated by the observation of human perception: Differences in thicker parts of an object must be more significant to be noticed by humans than differences in thin parts [XWB09].

It is necessary to involve the path lengths l and l' in the calculations, as the chosen path vector representation does not include the path length - each path vector has the same length, namely M, the fixed number of sample points. The importance of the skeleton path length can be weighted by an arbitrary weight factor $\alpha \in \mathbb{R}^+$. The higher α is, the more relevant the similarity of the path length becomes.

In the examples shown, it becomes clear that all skeleton paths emanating from the same end node are very similar in the beginning, as they share the first skeleton branch, but begin to vary as soon as the first junction node is passed. Furthermore, it might become obvious that similar end nodes in the two skeletons have similar paths emanating from them, while dissimilar end nodes have bigger differences in their emanating skeleton paths. This observation will be shown by referring to the two example birds in figure 4.1. Clearly, the end nodes in the beaks, marked with 7, would be a correct match. There is a pair of skeleton paths from end node 7 in both skeletons to one of the other end nodes in the skeletons that are similar to each other, according to the path distance definition in equation 4.3. For example, the skeleton path $p(7, 1)$ in skeleton S and skeleton path $p(7', 1')$ in skeleton S' are expected to have similar skeleton path vector entries.

Still the question is how the information in the skeleton path distances emanating from a pair of end nodes can be used for describing the similarity between two end nodes. The idea is to encode the information of all path distances for a pair of end nodes in one matrix, referred to as the path distance matrix. Such a path distance matrix can be defined for each combination of end nodes in the two skeletons. These path distance matrices describe the dissimilarity between two end nodes. They contain information about the path distances between all skeleton paths emanating from the two specific end nodes.

Let skeleton S with $K + 1$ end nodes and S' with $N + 1$ end nodes be the two skeletons to be matched, with $K \leq N$. For example, the matching costs for the end nodes v_{i0} from skeleton S and v'_{j0} from skeleton S' are wanted. First, the end nodes of both skeletons are ordered by traversing the object contour in clockwise direction, starting at v_{i0} in skeleton S and at v'_{j0} in S', respectively, resulting in a sorted list of end nodes for each skeleton: $\{v_{i0}, v_{i1}, \ldots, v_{iK}\}$ for S and $\{v'_{j0}, v'_{j1}, \ldots, v'_{jN}\}$ for S'.

The path distance matrix between two end nodes v_{i0} from skeleton S and v'_{j0} from skeleton S' is defined as

$$\boldsymbol{(v_{i0}, v'_{j0})} = \begin{pmatrix} pd(p(v_{i0}, v_{i1}), p(v'_{j0}, v'_{j1})) & \cdots & pd(p(v_{i0}, v_{i1}), p(v'_{j0}, v'_{jN})) \\ \vdots & \vdots & \vdots \\ pd(p(v_{i0}, v_{iK}), p(v'_{j0}, v'_{j1})) & \cdots & pd(p(v_{i0}, v_{iK}), p(v'_{j0}, v'_{jN})) \end{pmatrix} \tag{4.4}$$

The path distance matrix thus encodes information about the similarity of all skeleton paths emanating from the two skeleton end nodes to be matched. The assumption for the following computations is that similar end nodes are similar in the skeleton paths emanating from them.

For further computations, it is necessary to extract a scalar value from the path distance matrix to express the similarity between end nodes. Based on the path distance matrix, it is now possible to compute the total costs to match a pair of end nodes in the two skeletons. The problem of computing the similarity of two end nodes is in this approach considered as the problem of elastic matching of time series, that is, the skeleton's end nodes are treated as a time series. It is handled by *Optimal Subsequence Bijection (OSB)*, which was proposed in [LWKTM07] and already introduced in chapter 3.

As described earlier, the basic idea is to find the cheapest path through a given distance matrix. In the case of skeleton matching, the path distance matrix is used as the input cost matrix for OSB. The idea behind this is, that the more similar any of the emanating paths from two end nodes are, the more similar those end nodes are. OSB is order-preserving, going backwards is not allowed in the matrix, neither in the rows nor in the columns. Thus, the order of the end nodes still holds an important information about the contour as they previously were ordered by traversing the shape contour.

For example, the following matrix shows the path distance matrix between end node 0 in graph S and end node 0' in graph S' in figure 4.1.

$$\boldsymbol{pdm(0, 0')} = \begin{pmatrix} 0.41 & 31.75 & 31.82 & 45.38 & 47.36 & 48.0 & 48.34 & 2.75 \\ 26.03 & 0.17 & 2.01 & 8.28 & 6.59 & 6.61 & 6.31 & 18.4 \\ 26.05 & 1.03 & 0.26 & 13.09 & 6.74 & 7.07 & 7.84 & 17.57 \\ 38.32 & 8.56 & 13.49 & 0.12 & 8.35 & 6.94 & 4.33 & 31.72 \\ 40.74 & 6.59 & 6.85 & 8.04 & 0.19 & 0.18 & 1.21 & 31.36 \\ 41.09 & 6.71 & 7.67 & 6.19 & 0.94 & 0.35 & 0.28 & 32.01 \\ 40.87 & 6.29 & 7.97 & 4.7 & 1.57 & 0.81 & 0.12 & 32.04 \\ 1.64 & 18.47 & 19.91 & 27.88 & 30.93 & 31.04 & 30.74 & 1.14 \end{pmatrix} \tag{4.5}$$

Applying OSB to this path distance matrix leads to an alignment of the emanating skeleton paths from the two skeleton nodes with the possibility of skipping paths, if no corresponding path is found. This would mean that the end node of this path is not contained in the other sequence of end nodes. The cheapest path through the example matrix, found by this approach, is highlighted in the following matrix:

$$\boldsymbol{pdm(0,0')} = \begin{pmatrix} 0.41 & 31.75 & 31.82 & 45.38 & 47.36 & 48.0 & 48.34 & 2.75 \\ 26.03 & 0.17 & 2.01 & 8.28 & 6.59 & 6.61 & 6.31 & 18.4 \\ 26.05 & 1.03 & 0.26 & 13.09 & 6.74 & 7.07 & 7.84 & 17.57 \\ 38.32 & 8.56 & 13.49 & 0.12 & 8.35 & 6.94 & 4.33 & 31.72 \\ 40.74 & 6.59 & 6.85 & 8.04 & 0.19 & 0.18 & 1.21 & 31.36 \\ 41.09 & 6.71 & 7.67 & 6.19 & 0.94 & 0.35 & 0.28 & 32.01 \\ 40.87 & 6.29 & 7.97 & 4.7 & 1.57 & 0.81 & 0.12 & 32.04 \\ 1.64 & 18.47 & 19.91 & 27.88 & 30.93 & 31.04 & 30.74 & 1.14 \end{pmatrix} \tag{4.6}$$

In this example, the cheapest path is almost completely the diagonal from the left upper entry to the bottom right entry, but the last entry is skipped. The value in this entry is more expensive than skipping it: As there is a low-cost alignment in each row of the matrix, the constant value to punish skipping of rows is very low. If the two skeletons have end nodes that would not match any end node in the other skeleton, it is likely that more rows and columns in the matrix are skipped.

The total similarity, that is, the cost for matching the two nodes 0 in S and 0 in S' is computed based on all entries in the matrix visited by the shortest path: Each entry is squared, then added to the sum. Then, the squared root is computed for the complete sum and divided by the number of aligned elements. In this example, this would be

$$\frac{\sqrt{0.41^2 + 0.17^2 + 0.26^2 + 0.12^2 + 0.19^2 + 0.35^2 + 0.12^2}}{7} \approx 0.096 \tag{4.7}$$

4.1.3 Matching the end nodes

The steps as described in the previous paragraph are done for every combination of end nodes in the two skeletons. This way, for each combination of end nodes a scalar is obtained, expressing the similarity between the two end nodes.

The problem is to assign each end node in S to an end node in S' based on the matching costs found in the previous steps. For this purpose, all found cost values are stored in a cost matrix, containing all cost values for the two skeletons. The cost matrix $\boldsymbol{C(S,S')}$ for two skeletons is defined as

$$\boldsymbol{C(S,S')} = \begin{pmatrix} c(v_0, v'_0) & c(v_0, v'_1) & \dots & c(v_0, v'_N) \\ c(v_1, v'_0) & c(v_1, v'_1) & \dots & c(v_1, v'_N) \\ \vdots & \vdots & \vdots & \\ c(v_K, v'_0) & c(v_K, v'_1) & \dots & c(v_K, v'_N) \end{pmatrix} \tag{4.8}$$

Thus, each row j in the cost matrix contains the costs for matching the end node m_j in the target skeleton S' to each end node in the query skeleton S, while each column i in the matrix contains the costs for matching end node n_i in the target skeleton S' to each end node in the query skeleton S.

An example is given in the following matrix. This matrix shows the cost matrix for matching the two birds in figure 4.1.

$$\boldsymbol{C(S,S')} = \begin{pmatrix} 0.1 & 9.72 & 3.21 & 2.45 & 4.98 & 4.34 & 5.46 & 7.53 & 0.31 \\ 10.04 & 0.28 & 0.42 & 6.19 & 4.77 & 4.79 & 4.87 & 2.25 & 2.89 \\ 3.1 & 0.48 & 0.1 & 0.17 & 4.7 & 5.05 & 4.82 & 2.25 & 2.18 \\ 2.25 & 7.48 & 0.39 & 0.16 & 5.55 & 4.82 & 5.18 & 6.46 & 2.4 \\ 4.9 & 4.75 & 4.4 & 4.8 & 0.11 & 0.48 & 4.62 & 3.31 & 4.15 \\ 4.43 & 5.71 & 4.88 & 4.17 & 0.77 & 0.13 & 6.26 & 4.88 & 3.86 \\ 4.62 & 4.85 & 5.3 & 4.77 & 4.93 & 6.23 & 0.11 & 5.26 & 5.15 \\ 7.8 & 3.29 & 2.64 & 6.26 & 2.96 & 4.48 & 6.08 & 0.13 & 0.55 \\ 1.69 & 3.35 & 2.95 & 2.68 & 3.97 & 3.2 & 5.79 & 2.02 & 0.27 \end{pmatrix} \tag{4.9}$$

In this example, the value at index $(0,0)$ in the matrix is the costs to match the skeleton end node 0 in skeleton S and end node $0'$ in skeleton S'. This matrix, in turn, can be seen as input weight matrix for the Hungarian algorithm, as described in section 3.3.2. The problem of matching two objects is thus reduced to the classic assignment problem in a bipartite graph: The end nodes of skeleton S is one set of vertices V, and the end nodes of skeleton S' is another set of vertices V'. Both vertex sets can be seen as a partition in a bipartite graph. The cost of matching one vertex in V to a vertex in V' is given by the corresponding entry in the cost matrix. If the Hungarian algorithm is applied as shown in section 3.3.2, the minimum weight correspondence can be found. Figure 4.4 shows the found assignments, indicated by lines connecting two matched end nodes.

In a final step, the total cost to match the two skeletons can be computed by the sum of all edge weights.

4.1.4 Summary

In summary, the proposed algorithm can be split into the three following basic parts:

1. Getting a compact skeleton representation, based on the distance of the skeleton points to their feature point on the contour. These distances are stored in a vector.

2. Computing the costs to match one end node to another. This is done as follows:

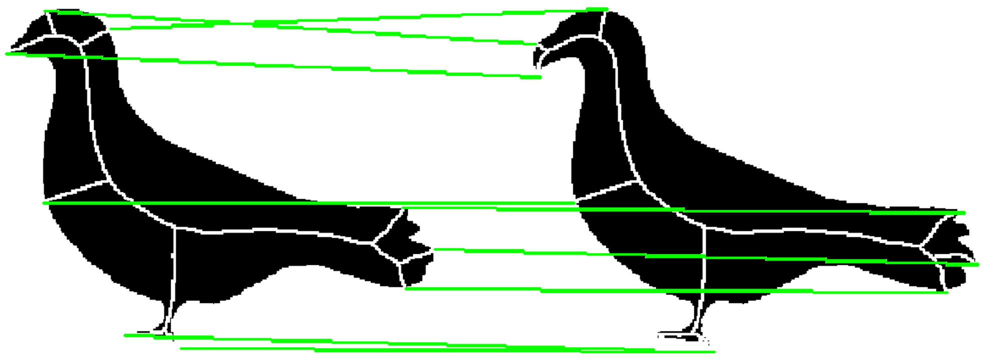


Figure 4.4: The matching result. Matched end nodes are connected by lines.

(a) A path distance matrix is created for two end nodes, encoding information about the dissimilarity between the skeleton paths emanating from those end nodes.

(b) OSB is applied to find the shortest path through the matrix to obtain the total costs to match these two nodes

This is done for all combinations of end nodes. The obtained values are stored in a matrix.

3. Matching the skeleton end nodes by applying the Hungarian algorithm to the cost matrix obtained in the previous step.

4.2 Experiments

In the context of this diploma thesis, the algorithm introduced in [BL08] was re-implemented. The experiments are split into two parts: First, some examples for end node matchings are shown. In the second part, the recognition performance of the method is shown by using images from three shape databases.

4.2.1 End node matching

First, the matching method was tested on several example images of non-rigid objects, in particular elephants, a subset of the animal dataset used in [BLT09]. Although the algorithm works well for various examples, it also has some limitations which will be summarized in section 4.2.1.

Figure 4.5 shows a very simple example. Correct matchings are indicated by green lines. Wrong matchings are indicated in red lines. In this example, all end

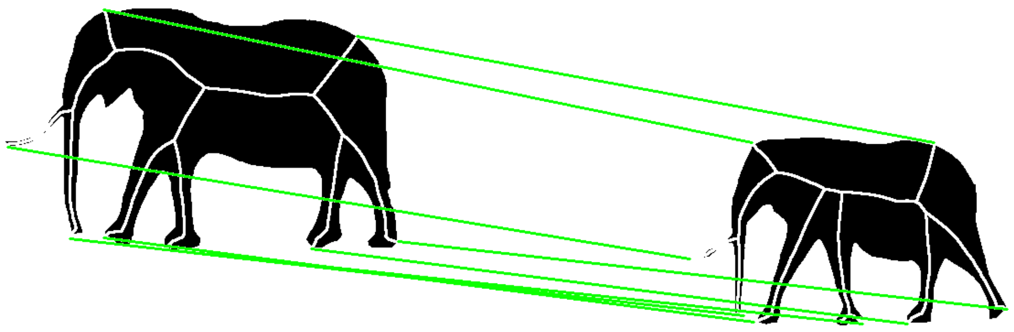


Figure 4.5: Example of a good matching result. Correct matchings are indicated by green lines. Wrong matchings are indicated by red lines. In this example, all end nodes have been matched correctly.

nodes have been matched correctly, that is, the end nodes are matched like a human would except it. In this case, both shapes are very similar to each other, varying only in few details, and one of the shapes is slightly smaller than the other.

Figure 4.6 shows a slightly more complex example. Again, the two shapes are quite similar, but overlaps occur due to the bending of the front legs. It can be observed that even though the two front legs are overlapping in the left image, the algorithm finds the correct correspondences between the front legs.

Figure 4.7 shows an example where the two shapes are flipped horizontally. As will be shown later, a simple alteration had to be made for the algorithm to make it invariant to rotation. The bending of the trunk does not have any effect on the matching result, as deformations in object parts that don't have any effect on the radii of the maximal inscribed discs are not punished in this approach. One mismatch has been found in this example, indicated by the red line. Nevertheless it can be observed that both end nodes in the two skeletons do not have any matching partner in the other respective skeleton. This kind of mismatch will be examined in section 4.2.1.

In the example shown in figure 4.8, one of the shapes is rotated, but most of the end nodes have still been matched correctly. However, the two end nodes in the trunk are swapped in the matching result. This is due to the fact that when traversing the contour they are encountered in reverse order. As the skeleton paths emanating from these two end nodes are very similar, they are swapped in the final matching result.

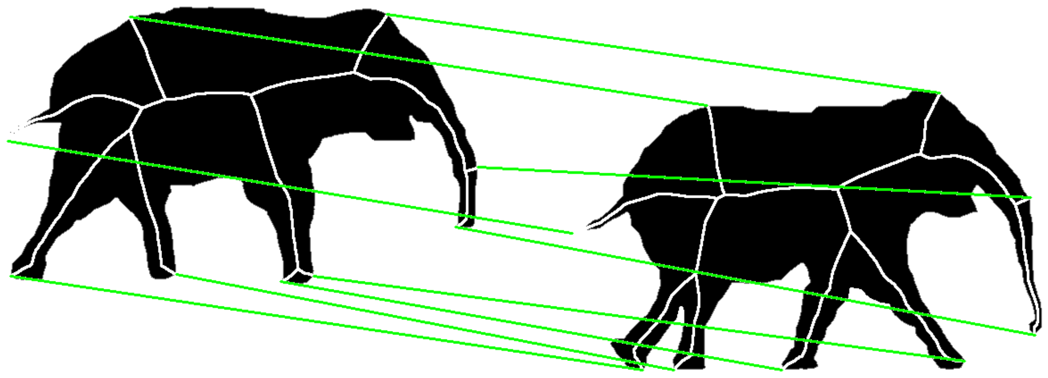


Figure 4.6: Example of a good matching result. All end nodes have been matched correctly, in spite of overlaps in the front legs.

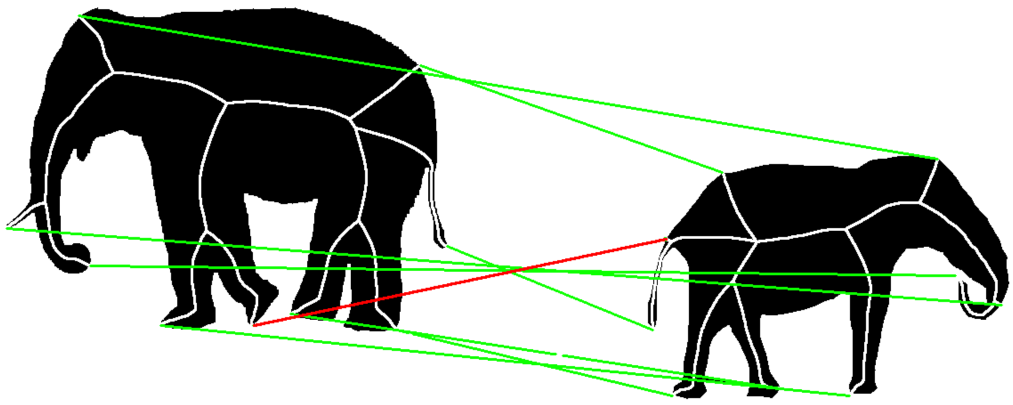


Figure 4.7: Example of a matching between shapes with bent object parts.

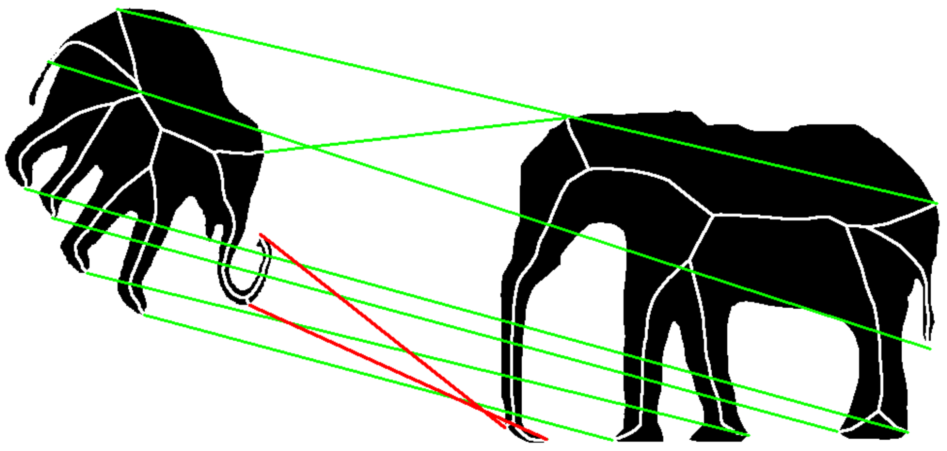


Figure 4.8: Example of a matching of rotated shapes. The two end nodes in the elephants' trunk are swapped in the matching, that is, if a human should establish the correspondences between the skeleton end nodes manually, he would match the two end nodes in the trunk the other way around. This is due to the fact that the skeleton paths emanating from these two end nodes in both skeletons are quite similar, and as in both skeletons the order for the two end nodes in the trunk is switched when traversing the contour, they get falsely matched.

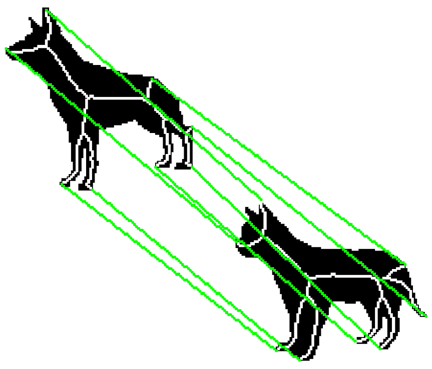


Figure 4.9: Matching between a dog and a cat.

Further matching experiments were performed with the Kimia99 database [SKK04], a subset of the MPEG-7 database [LLE00]. This database contains mostly non-rigid objects and several shapes with occlusions.

Figure 4.9 shows that establishing correspondences between two different animals, in this case a dog and a cat is possible as long as the two animals have similar characteristics.

Several experiments were performed to test the algorithm's performance in the presence of occlusion. Figure 4.10 shows the matching of two hand shapes. One of the hand shapes is partly occluded, so that the fingertips seem to be missing. All the same, the correspondences are established correctly as the characteristics of the two shapes remain similar.

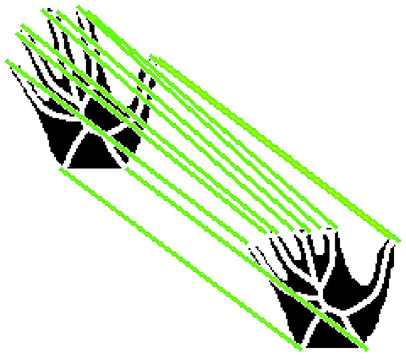


Figure 4.10: Matching in the presence of occlusions

Another type of occlusion is when not only object parts seem to be missing like in the previous example, but the occluding object is also included in the shape.

Figure 4.11 shows two examples of this scenario. Again, the two hand shapes are matched correctly.

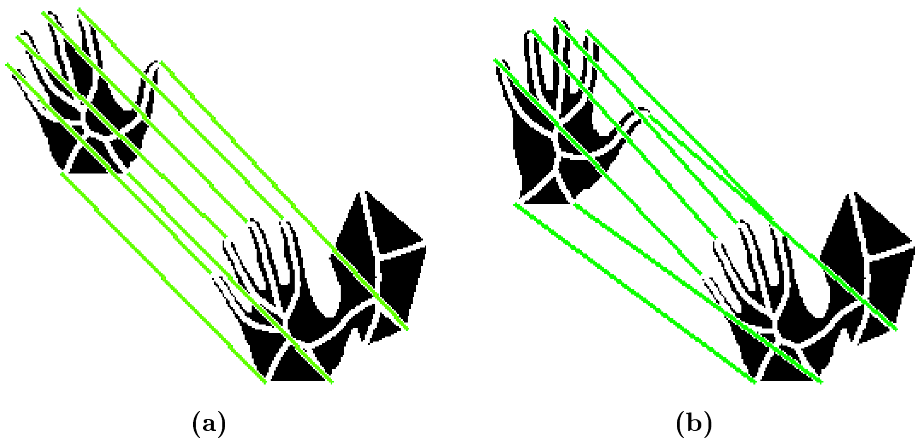


Figure 4.11: Matching in the presence of occlusions

In the original paper, two similar shapes were matched, but in their experiments, some correspondences could not be established correctly. This is assumed to be due to an implementation detail in the choice of parameters in the OSB function. In the implementation of the OSB function it is possible to determine how many rows are allowed be skipped in the matrix. It is assumed that a fixed value was chosen for the original implementation. For the new implementation in the context of this diploma thesis, a variable value was chosen, depending on the difference in the number of skeleton end nodes in the two skeletons, making the described approach more flexible in situations like the shown occlusions.

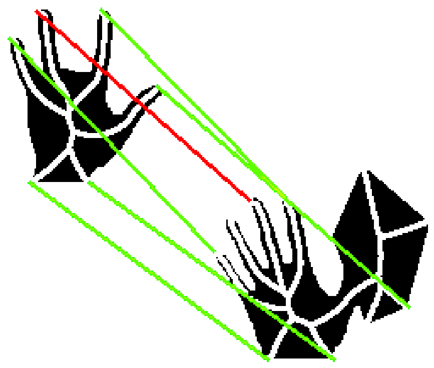


Figure 4.12: Matching in the presence of occlusions and deformations.

However, the absence of one finger in one of the shapes, caused by the deformation of the hand and the projection to the 2D plane, leads to one mismatch, as shown in figure 4.12. The skeleton paths in the fingers are quite similar. Hence, the order of occurance of the fingertips' end nodes is an important indicator for their similarity. As the order in this example is misleading, a mismatch occurs.

Limitations

As shown before, the algorithm has some limitations that will now be summarized and explained.

Flipped Images In the case of flipped images, that is the two shapes point in different directions, the correct matching costs for a pair of end nodes cannot be found. The weak point is the OSB function.

Consider for example the scenario where one shape should be matched to exactly the same shape, but flipped horizontally. The path distance matrix for two corresponding end nodes would in this case include a cheapest path from the upper right corner to the lower left corner, as shown in the following example matrix:

$$(\boldsymbol{v_{i0}}, \boldsymbol{v'_{j0}}) = \begin{pmatrix} \dots & 7 & 5 & 8 & 0 \\ \dots & 2 & 3 & 0 & 8 \\ \dots & 5 & 0 & 3 & 5 \\ \dots & 0 & 5 & 2 & 7 \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix} \tag{4.10}$$

The actual shortest path is highlighted in grey. As the OSB function does not allow to go backwards in the matrix, neither in the rows nor in the columns, this cheapest path can't be found, and the resulting matching costs are no reliable indicator for the similarity of two end nodes.

As a solution, the algorithm is applied twice: once for the original images, and once with one image flipped horizontally. From the resulting two match lists, the one with lesser matching costs will be chosen as the real matching. In most cases this works quite well. There are cases, however, in which this method fails to determine correctly if shapes are flipped or not.

For example, the shapes in figure 4.13 are oriented in opposite directions, but the matching costs for the second run with one of the images flipped leads to lower matching costs than the first run. Thus, in the algorithm the two shapes are assumed to be oriented in the same direction, which in the end leads to an unsatisfying matching.

Enforcing of 1-to-1-matching Another problem is caused by the fact that the algorithm requires a 1-to-1 matching between the two skeletons which is not always possible. This problem is closely related to the problem mentioned previously, that spurious branches can have a negative impact on the matching results.

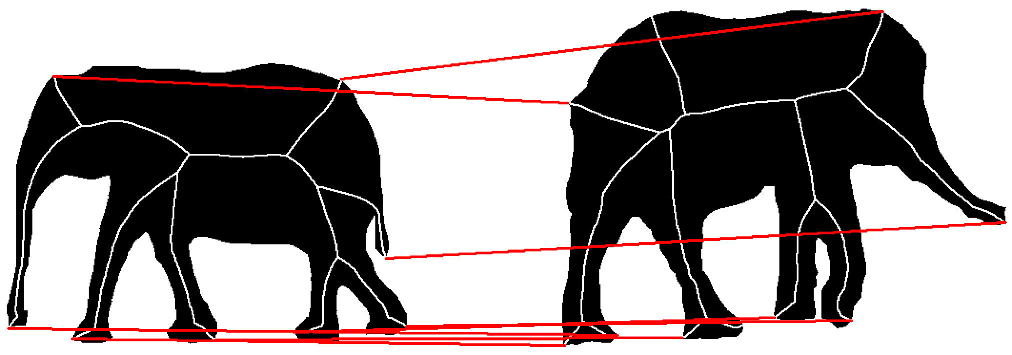


Figure 4.13: In some cases, the method fails to determine if shapes are flipped or not.

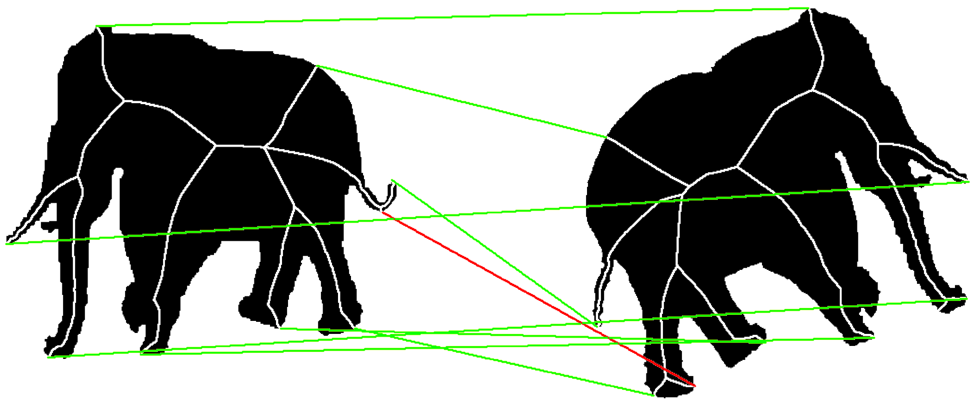


Figure 4.14: 1-to-1 matching is not always possible. In this example, all matchings have been found correctly, but the two remaining end nodes with no matching partner in the other skeleton are matched.

See for example figure 4.14, which shows an acceptable matching result for two elephant shapes. All correspondences have been found correctly, but both skeletons have one additional end node that has no matching partner in the other skeleton. As the Hungarian Algorithm forces all end nodes to find a matching partner in the other skeleton, the two remaining nodes are matched, even though they do not correspond. In fact, this is not only a limitation of this particular algorithm , but a problem of all matching algorithms that reduce the matching problem to a 1-to-1 matching in a bipartite graph. Using a different model than the matching of a bipartite graph for the final matching could be a solution to this problem. For example, the Earth Mover's Distance (EMD) [RTG00] also allows partial matchings. This could be a solution to better deal with noisy skeleton data.

Spurious branches This problem is related to the limitation described in the previous paragraph. The algorithm requires optimal skeletons, where each skeleton branch represents a significant visual part of the shape. If one of the skeletons contains spurious branches, this can have a negative impact on the matching result.

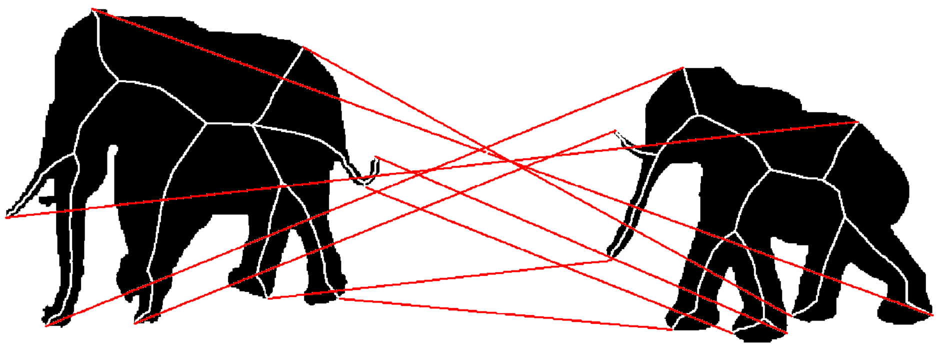


Figure 4.15: Spurious branches can have a deep impact on the matching result. The figure shows a matching between two elephant shapes, where no correct correspondences could be established at all. If one inspects the left elephant in detail, one can see that he has a spurious branch in the tail. Figure 4.16 shows a cut-out of the affected area.

Figure 4.15 shows the assignments between two elephant shapes. As can be seen, none of the found correspondences is correct. If the skeleton of the left elephant is inspected in detail, one can see that the left elephant has a spurious branch in the tail that does not represent a significant visual part of the object. The affected area is shown in figure 4.16 in detail.

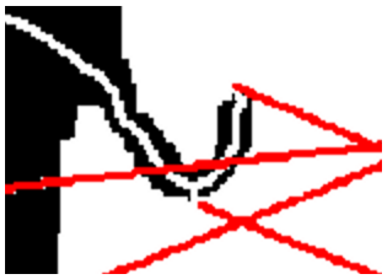


Figure 4.16: Cut-out of the elephant shape in figure 4.15. The elephant's tail contains one spurious branch that has a negative impact on the matching result.

If this branch is removed manually, the result is still not perfect, but the number of correct correspondences is now six of the eight possible matchings, as shown in figure 4.17.

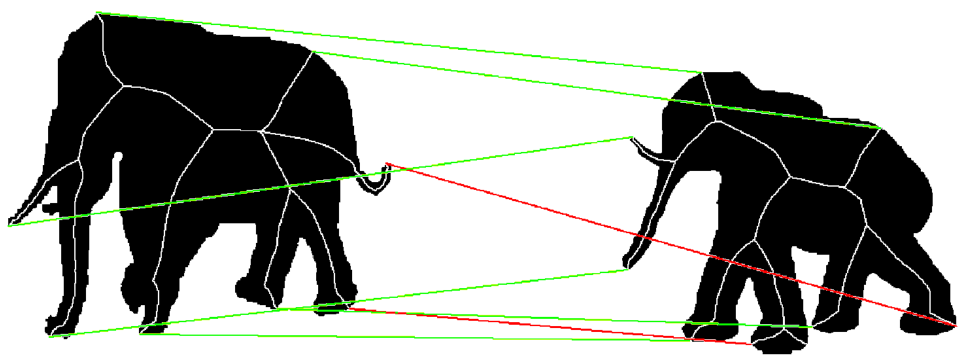


Figure 4.17: Once the spurious branch is removed manually, the matching result gets much better, reducing the number of wrong matches to two.

Thus, one has to make sure that the input skeletons do not contain spurious branches. Again, a possible solution could be to use an algorithm that also allows partial matchings for the final matching, for example the *Earth Mover's Distance* [RTG00].

4.2.2 Recognition performance

As the proposed method computes a value for the similarity of two shapes based on the end node correspondences, it can also be used for object recognition in a shape database. A very simple retrieval system was implemented to evaluate the matching algorithm. In this retrieval system, it is possible to enter a shape and its skeleton as a query, and the matching algorithm is applied to the query and all other shapes in the database. All shapes in the database are ordered according to their computed similarity to the query, with the highest similarity first.

As a first database, a subset of the Aslan and Tari shapes [AT05] is used. It contains eleven classes of shapes, and each class contains four images. One example for each of the classes is shown in table 4.1.

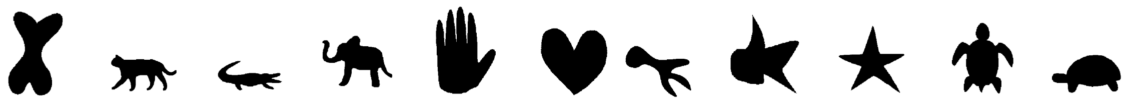


Table 4.1: Example shapes from the Aslan and Tari database [AT05].

The shapes' skeletons were obtained from the implementation of the *Discrete Curve Evolution* algorithm [BLL07] [1], with the parameters $\rho = 4$, $T_1 = 1$ and $number\_vertice = 15$.

In this first experiment, each of the shapes in the database was used as a query. The parameters used were $M = 50$ and $\alpha = 40$. All shapes in the database are ordered according to their similarity value, resulting in an ordered result list. As there are four images in each class, the query and the first three result shapes should be in the same class.

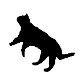
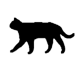
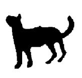
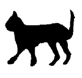
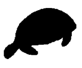
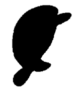
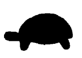
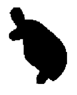
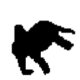
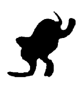
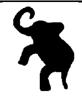
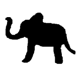
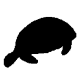
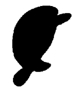
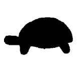
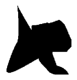
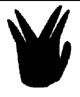
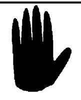
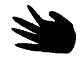
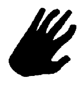
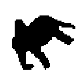
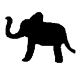
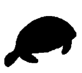
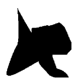
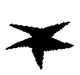
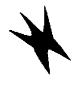
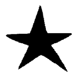
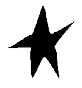
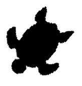
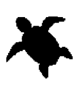
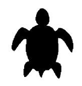
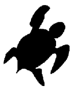
Query	1st	2nd	3rd	4th	5th	6th	7th
							
							
							
							

Table 4.2: Example queries in the Aslan and Tari database.

Table 4.2 shows four example queries in the Aslan and Tari database. In the left column, the query image is shown. From left to right, the computed similarity to the query image drops. In the shown examples, the first three search results are in the same class as the query image, that is, the result is optimal.

A common way to rate a retrieval system is the computation of precision and recall values. As the number of relevant documents is known for each query, this value can be used as the number of returned result documents in the similarity search. For this small database, the average precision for the number of three result documents is an interesting value in order to rate the retrieval system. The average is computed for all queries, that is, all images in the database have been used as a query. The precision is computed for each query in relation to the number of returned result documents, which in this case is 3. Under these conditions, the average precision for this retrieval system is 0.93. If one summarizes the number of correct shapes for all queries among the first three retrieval results, one obtains 43, 41, and 41. The perfect result for this database subset would be 44, 44, 44.

[1] available at http://sites.google.com/site/xiangbai/BaiSkeletonPruningDCE.zip

In order to analyze the limitations of the algorithm however, it is helpful to inspect the mismatches in detail, especially the queries where two or even three mismatches occured. Table 4.3 summarizes all queries to the shape database with wrong results. While the mismatches in the first three rows are easily explained

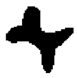
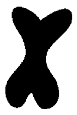
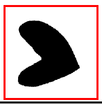
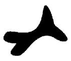
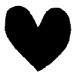
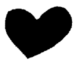
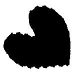
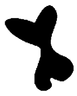
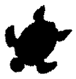
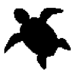
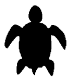
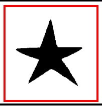
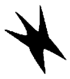
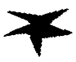
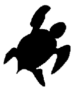
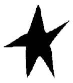
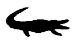
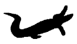
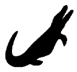
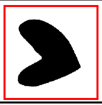
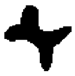
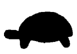
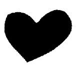
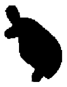
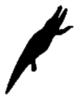
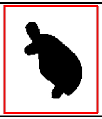
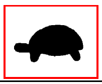
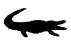
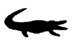
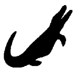
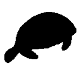
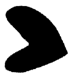
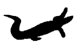
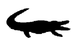
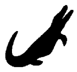
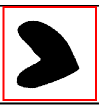
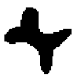
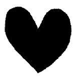
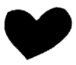
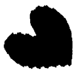
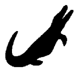
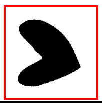
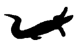
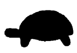
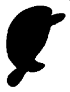
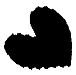
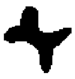
Query	1st	2nd	3rd	4th	5th	6th	7th	8th
								
								
								
								
								
								

Table 4.3: Mismatches in the Aslan and Tari database. Wrong results are highlighted in red.

as the wrong results and the query have quite similar shapes, the mismatches for the queries of class "crocodile" in the last four rows are somewhat surprising. The high matching costs between the four crocodile shapes become more obvious when inspecting the skeletons and their end node matching in detail.

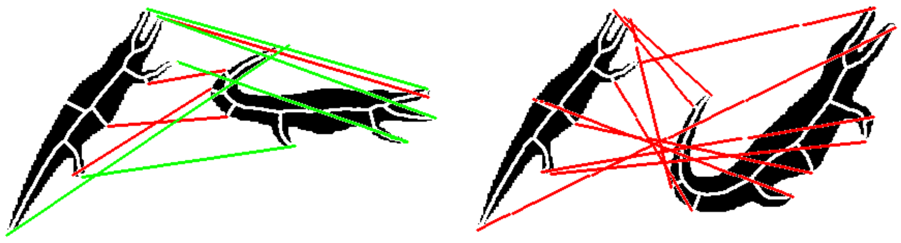


Figure 4.18: Matching with spurious branches, leading to bad similarity values.

Figure 4.18 shows two example assignments of the crocodile with the worst recognition result to two other crocodiles. The example shows that the problem is again that the crocodiles' skeletons have spurious branches and therefore end nodes that do not find a matching partner in the other skeleton. In the best case, all correspondences are established correctly, but the end node matching contains additional correspondences that were established due to the fact that the algorithm requires one-to-one correspondences, as shown in the left image in figure 4.18. Assignments like these have generally high matching costs, leading to distorted total similarity values. In the worst case however, the whole matching process gets corrupted by these additional branches, as shown in the right image in figure 4.18.

In order to support this assumption, additional experiments were performed. The most problematic skeletons, the crocodile skeletons, were manually pruned, and the whole experiment was repeated. One example for the manual pruning of the crocodile skeleton is shown in figure 4.19. With these alterations one obtains 44, 44, 41 for the number of correct shapes in the first, second and third result. The average precision for three returned result documents for each query in this altered shape database is 0.98.

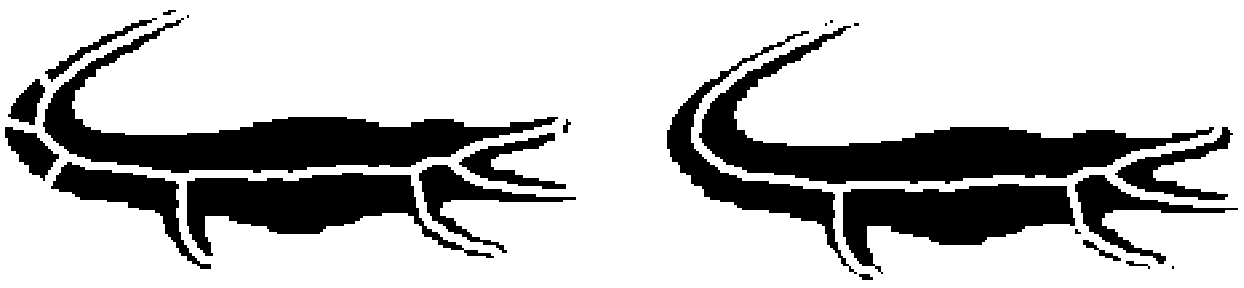


Figure 4.19: Crocodile skeleton before and after the manual pruning.

Further experiments were performed with a subset of 60 images of the kimia-99 shape database [SKK04]. This subset contains six classes with 10 images in each of them. Included are rigid as well as non-rigid objects. The challenge when using this database for experiments is that several of the contained shapes include partial occlusions. Examples of these occluded shapes already have been shown in section 4.1.3. Table 4.4 shows further example shapes from the kimia-99 database.

Again, each shape has been used as a query. Since each class consists of ten shapes, the first nine results in the similarity search should be in the same class as the query. The average precision in relation to 9 returned result documents is 0.84. Figure 4.20 shows the development of the average precision and recall values with an increasing number of result documents.

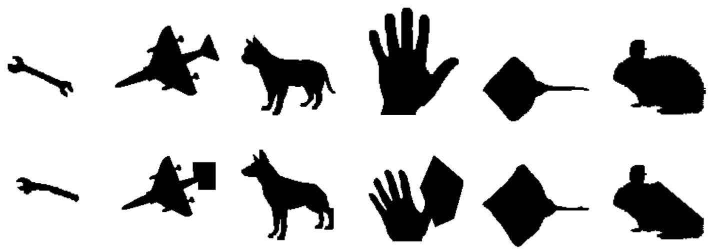


Table 4.4: Example shapes from the kimia-99 database [SKK04]

One important fact about this diagram is that the recall value does not reach 1.0, even though in the diagram, the average precision and recall values for eleven result documents is shown, and thus, takes two additional documents within account. The flat recall curve in the end indicates that in some classes the most dissimilar shapes are so dissimilar that they do not appear in the result list, even if the number of returned result documents is higher than the number of relevant documents. This corelates to the precision graph that drops drastically after the first eight result documents. This is also reflected if one summarized the number of correct result in the first nine result documents: one obtains 60, 58, 57, 54, 56, 51, 49, 45 and 22. Inspecting the mismatches in detail shows that even though in large part correct correspondences were found even for occluded shapes, the total similarity values for these occluded shapes are too high for the shapes to be recognized to belong to the same class as the query. Another aspect is again dealing with spurious branches in the skeletons.

The last experiments to evaluate the recognition performance were done with the kimia-216 shape database. This shape database consists of 18 categories with 12 images in each of them. Example shapes are shown in table 4.5.

Table 4.5: Example shapes from the kimia-216 database

The parameters used for these experiments were the same as for the previous experiments: The skeletons were computed by the *Discrete Curve Evolution* algorithm [BLL07], with the parameters $\rho = 4$, $T_1 = 1$ and $number\_vertice = 15$. Again, the parameters used were $M = 50$ and $\alpha = 40$.

Five example queries and the first eight most similar results are shown in table 4.6.

Each shape from the shape database has been used as a query. There are eleven relevant documents in the database for each query. The average precision value in

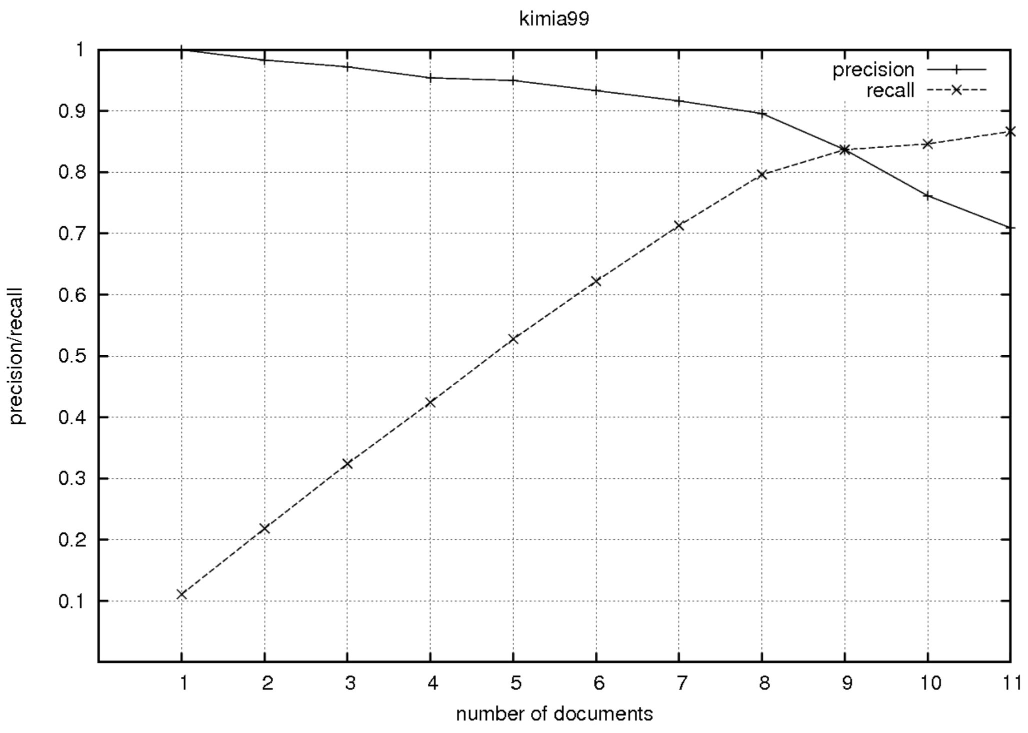


Figure 4.20: Average precision and recall development in the kimia-99 database with increasing number of result documents.

relation to eleven returned result documents is an interesting measurement for the quality of the retrieval system, which in this case is 0.81. Figure 4.21 shows the development of the average precision and recall values for an increasing number of returned result documents.

Table 4.7 summarizes the number of all correct shapes for the first eleven retrieval results in comparison to the values listed in the original paper.

Obviously, the results are not as good as in the original paper. The assumption is that the input skeletons play a significant role here. Several skeletons used in the experiments in the context of this thesis contain spurious branches which can have a profound impact on the quality of the end node matching, leading to distorted overall similarity values. This effect has been observed in many of the query results.

To verify this assumption, further experiments were performed on some of the more problematic classes. This time, some of the skeletons in the database were pruned manually so that each skeleton branch represents a significant visual part of the original shape. As the significant parts of shapes of the same class should be quite similar, the skeletons get more comparable. Using the manually pruned skeletons leads to better results in the performed queries. For example, the average

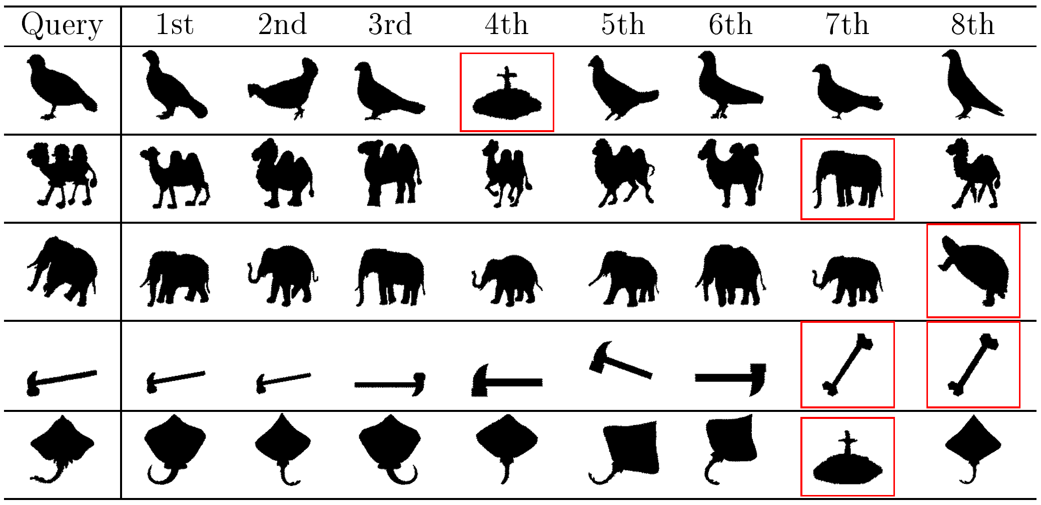


Table 4.6: Example queries on the Kimia-216 database. In the left column, the query shape is shown. From left to right, the eight most similar shapes in the database are shown. The similarity to the query drops from left to right.

	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th
Original paper	216	216	215	216	213	210	210	207	205	191	177
Own results	205	208	202	199	200	192	184	167	161	130	96

Table 4.7: Summarization of correct shapes in the 1st, 2nd,.. retrieval result.

precision for the queries from the “bird”-class went up from 0.69 to 0.81, the average precision for the queries from the “camel” class went up from 0.63 to 0.73

It can also be observed that the average precision value for the kimia-99 and kimia-216 database is worse than for the Aslan and Tari database. The reason for this partly lies with the composition of data in both databases. In the Aslan and Tari database, the algorithm’s performance for non-rigid shapes is mainly evaluated. Parts of the shapes are bent, but besides that, the shapes within a class are quite similar to each other.

The main challenge in the Kimia-99 database is that several shapes are occluded.

The Kimia-216 database contains non-rigid objects as well as rigid objects. The main problem with this database is that some of the classes are very similar in their shapes, while within some classes, there is a huge variety. Table 4.8 shows some examples of shapes that belong to different classes, but are very similar and thus, lead to several mismatches in the retrieval result.

In summary, the matching algorithm shows acceptable results in the retrieval experiments. Matching of non-rigid objects poses no problem as deformations

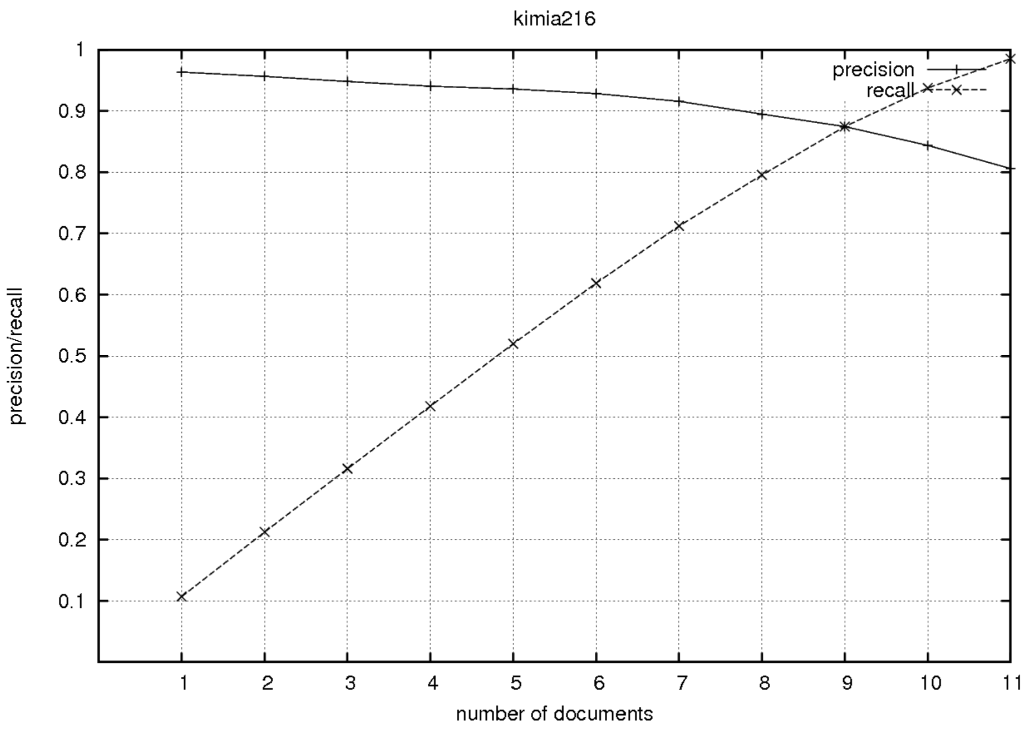


Figure 4.21: Average precision and recall development in the kimia216 database with increasing number of result documents.

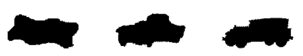
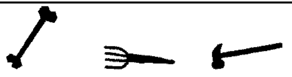
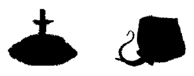
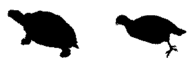
			

Table 4.8: Similar shapes in the Kimia-216 database from different classes.

that do not change the shape breadth are not punished. Suboptimal skeletons with spurious branches however can lead to severe mismatches. In the context of extending the algorithm into the third dimension some ideas, amongst others, were developped to deal with this problem. These ideas are introduced in the next chapter.

Chapter 5

Skeleton Graph Matching in 3D

In this chapter, the algorithm described in section 4 will be examined with respect to extending it into the third dimension. The algorithm will be applied to three-dimensional medical data. The goal is to compare pre- and postoperative blood vessel volumes.

5.1 Data origin

The idea is to compare two aorta images of the same patient before and after an EVAR procedure was performed. The aorta is

> "the main trunk of a series of vessels which convey the oxygenated blood to the tissues of the body for their nutrition. It commences at the upper part of the left ventricle, where it is about 3 cm in diameter, and after ascending for a short distance, arches backward and to the left side ...; it then descends within the thorax ..., passes into the abdominal cavity ..., and ends, considerably diminished in size (about 1.75 cm in diameter) ... by dividing into the right and left common iliac arteries. Hence it is described in several portions, the ascending aorta, the arch of the aorta, and the descending aorta, which last is again divided into the thoracic and abdominal aortæ." [GG73]

Figure 5.1 shows an overview of the arteries in the human body. The imaged region is located at the descending aorta, in particular, at the lower parts of the abdominal aorta (*aorta abdominalis*), at the aortic bifurcation (*bifurcatio aortae*), where the abdominal aorta divides into the right and left common iliac arteries [FS08]. The region is highlighted in grey in figure 5.1.

As mentioned before, the idea is to match two aorta volumes of the same patient, before and after surgery. The patient was suffering from *abdominal aortic*

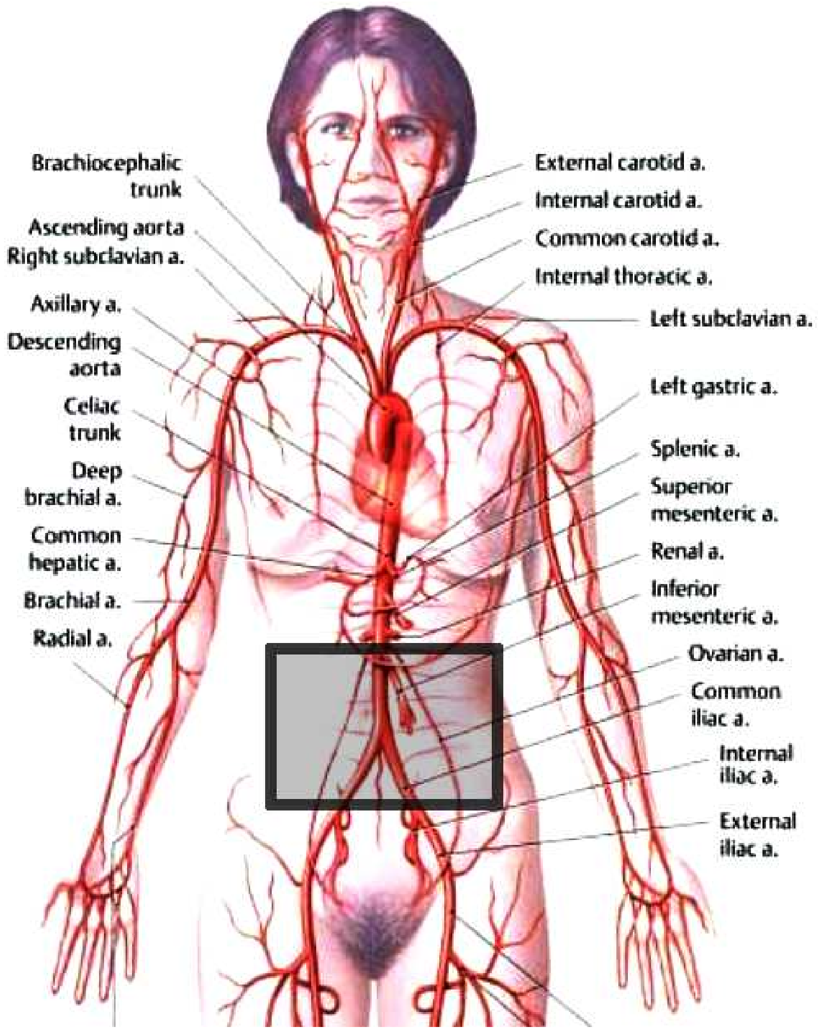


Figure 5.1: Schematic overview of the arteries in the human body. Image source: [SSS$^+$06].

aneurysm, AAA, before the procedure. An aneurysm is a blood-filled bulge in the wall of a blood vessel. With an increasing size of the aneurysm, the risk of rupture increases. The rupture of an aneurysm can lead to death [FS08]. One standard procedure to deal with AAA is *EVAR* (*endovascular aneurysm repair*) which carries a "reduced early morbidity rate and mortality compared to open operation for aneurysim repair" [RHL11].

Both data volumes (before and after EVAR) were gathered by CT scan. The interesting areas are then segmented and skeletonized using the built-in thinning filter from the Insight Toolkit (ITK) [1]. An example of the resulting volumes is shown in figure 5.2.

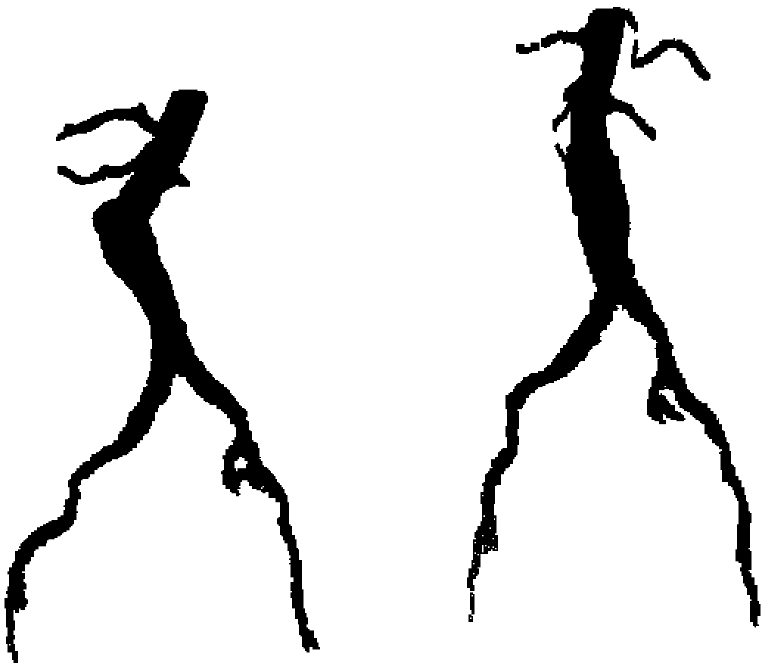


Figure 5.2: Pre- and postoperative blood vessel volumes. Left: Before surgery. Right: After surgery.

Though the main application of the 3D algorithm will be the matching of parts of aorta volumes it is desirable to develop a general extension of the algorithm in 3D that not only applies to this special kind of 3D data, but also to possible other scenarios.

5.2 Considerations about matching of 3D objects

Before introducing the alterations that were made to make the algorithm work in 3D, some general observations and expected problems will be listed in the following section.

The key concept of the algorithm is the skeleton path and the distance from the sample points to their feature points. In general, the idea can also be applied to 3D data as the 3D curve skeleton has (or at least should have) similar properties as the 2D Blum skeleton, like continuity and thickness of exactly one voxel, which makes the sampling of the path possible. As mentioned earlier, the centeredness

[1]http://www.itk.org/

of the 3D curve skeleton is difficult to define mathematically, which could have an impact on the results. In this particular application however this is less of a problem, as the blood vessels have a tubular form where the centeredness of the curve skeleton is easier to define.

Some problems in the 2D experiments stem from overlapping object parts caused by the projection to the 2D plane. For example, some animal shapes in the experiments seem to have only three legs, which makes the matching process difficult. There is no projection in 3D, and there are are no overlaps to be expected. This can have a positive impact on the matching results.

Though the general idea of the algorithm can be applied to 3D data, some problems are expected to occur. The biggest problem expected is that the algorithm requires the skeleton end nodes to be ordered, which happens when traversing the contour of the input shape. In 3D, traversing a contour is not possible. As the OSB function which extracts a scalar value from the path distance matrix is order-preserving, either this step has to be replaced by another approach that isn't, or a meaningful order for the end nodes in 3D has to be found.

Another expected problem results from the special application in the matching of blood vessels: The blood vessel volumes do not vary very much in their thickness. Thus, the distance from any skeleton point to the boundary is homogenous in every part of the volume.

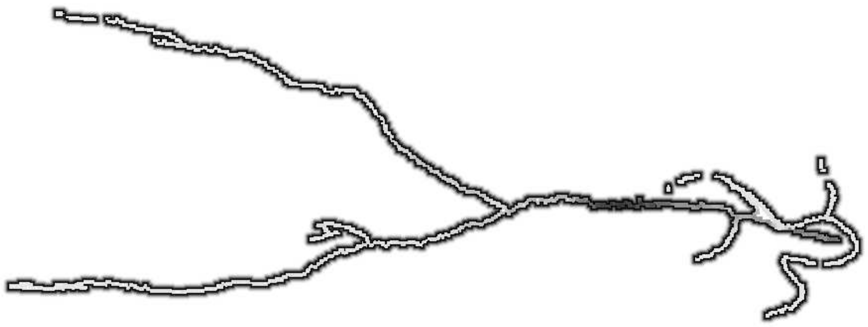


Figure 5.3: Skeleton with visualization of distance transform

Figure 5.3 shows a skeleton where the distance to the shape boundary is visualized. The darker the voxel's color, the broader the shape at this point is. As can be seen, the brightness values do not vary siginificantly over the skeleton. Just the area above the aortic bifurcation is considerably thicker than the rest of the shape.

The algorithm requires optimal skeletons, that is, each skeleton branch should represent a visual part of the object. As already shown in 4.2.1, spurious or missing branches can lead to unsatisfying matching results. The provided medical data however is quite noisy, resulting from inaccuracies during the image acquisition, the segmentation process and skeletonization. Some branches are shortened or cut off completely in one skeleton, but are still existent in the other skeleton. An example can be seen in figure 5.2: the shape is not connected, and neither is the skeleton, resulting in shortened branches. In the following discussion, the disconnected parts are simply ignored. Ignoring the disconnected parts can, however, lead to problems in the matching results.

Applying the 2D algorithm without any alterations and without any explicit ordering of the end nodes may lead to random matching results. In the case of blood vessel volume data this problem can be worked around, as the pictures are usually taken from a similar point of view, and the implicit ordering of the skeleton end nodes in their appearance in the 3D space can be used as an order: The volume's bounding box is scanned in the direction of x-, y- and z axis and end nodes are stored to an ordered list when they occur in the scans. Figure 5.4 shows the matching between two blood vessels. Corresponding end nodes are connected by a line, while the color of the line indicates true and wrong matchings, respectively.

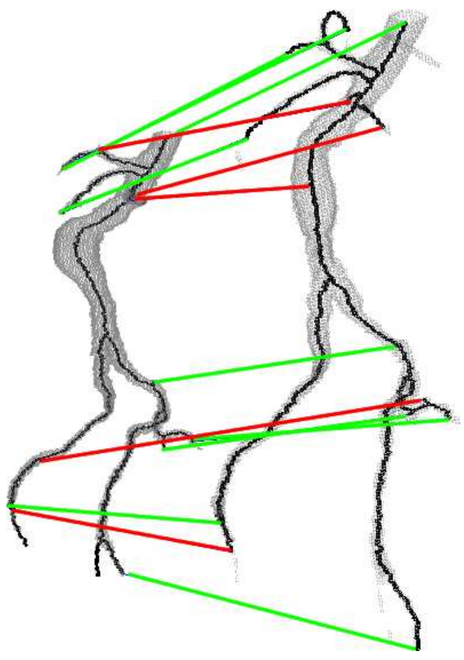


Figure 5.4: Applying the algorithm without any alterations to 3D volumes. The parameters used were $\alpha = 70$ and $M = 50$.

No alterations have been made to the algorithm in this example, and the implicit order of the end nodes in the bounding box is used. As can be seen in this example, 8 of 14 correspondences are correct assignments, but 6 of the found correspondences are mismatches. These mismatches result from the problems already mentioned. Considering that both skeletons contain end nodes that do not have a matching partner in the other this is an acceptable result, but a more robust approach is desired that also is reliable for general matching of 3D objects, even if the point of view differs for the two shapes. Furthermore, a solution for filtering the wrong matchings is desirable.

In summary, there are three main problems to deal with when applying the introduced algorithm to 3D:

- *OSB in 3D*: The OSB function requires ordered end nodes, but in 3D, they cannot be ordered by traversing the contour.
- *Similar disc radii*: The volumes do not vary much in their breadth. The disc radii are similar for each skeleton point.
- *Dealing with noise*: The skeletons are too noisy. The branches are shortened at different levels or cut off completely.

Some ideas and alterations have been developed to resolve or mitigate the problems pointed out above and thus, will lead to more robust matching results. These ideas will be shown in the next sections with two example volumes. More experiments with other datasets will follow in section 5.6.

5.3 OSB in 3D

As mentioned before, the main weak point in the algorithm for the matching of 3D data is the OSB function which requires ordered end nodes. The solution would be to replace OSB by another function that is not order preserving, or to find a meaningful order for the end nodes.

5.3.1 Replacing the OSB function

The OSB function is used to compute the similarity between two end nodes by aligning their emanating skeleton paths. In order to replace OSB one needs another function that is capable of finding optimal assignments in a given cost matrix, but without requiring any order. One possibility that comes to mind is the Hungarian Algorithm which also extracts matching costs for a given cost matrix, and thus can also be used for extracting the similarity value from the path distance matrix.

Figure 5.5 shows a matching between the two example shapes, with the OSB function replaced by the Hungarian algorithm.

The number of wrong correspondences in this example has increased to 8 out of 14. The result quality does not increase in other examples either. Thus, using the

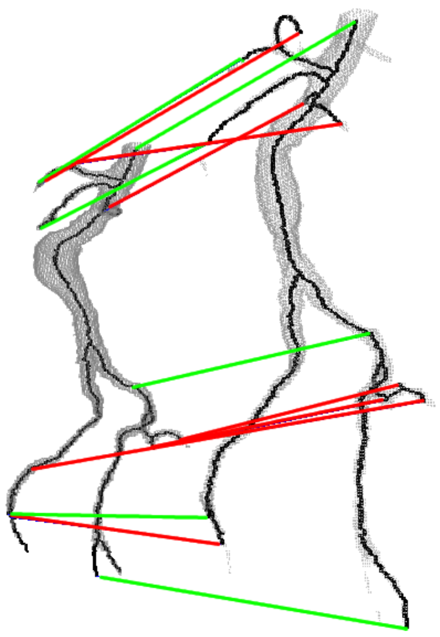


Figure 5.5: Applying the algorithm to 3D data, with replacing the OSB function by Hungarian algorithm. The parameters used were $\alpha = 70$ and $M = 50$.

Hungarian algorithm instead of the OSB function does not seem to be a reliable procedure. The problems are:

- The Hungarian algorithm is not capable of skipping elements. The CT scans however contain several spurious branches which makes skipping elements necessary.
- The weakness of the OSB function for the application in 3D is also its strength: By requiring ordered end nodes, the OSB function does not only incorporate the similarity of the emanating skeleton paths, but also contour information. When using the Hungarian algorithm instead, this information about the shape cannot be used anymore. Furthermore, the disc radii are very similar for each skeleton point, which leads to similar skeleton paths. Without the additional information about the shape contour the path distance alone is not a reliable similarity indicator.

5.3.2 Ordering the end nodes

As was shown in the previous section, replacing the OSB function by the Hungarian algorithm is not a reliable procedure for computing the similarity between end nodes. Hence, it is desirable to find an ordering of the end nodes in 3D to make it possible to use OSB also in 3D.

Obviously, it is not possible to order the end nodes by traversing the shape's contour, as was done in the two-dimensional case. For the aorta volumes the implicit order of the end nodes by their occurance in the 3D space can be used. As the point of view is very similar for both volumes, the end nodes are expected to occur in a similar order.

For other 3D images, this might not be a good solution as the 3D models are not always expected to be aligned. One possibility to order the end nodes is to order the end nodes by their distance to the start node.

Assuming that the similarity between two end nodes i and i' in two skeletons G and G' is wanted, the end nodes in skeleton G are then ordered so that the first end node i_0 in the list of ordered end nodes is the end node with the shortest distance to i. The distance is measured by the length of the skeleton path between i and i_0. The second end node is the end node i_1 with the second shortest distance to i, and so on. This again results in an ordered set of end nodes $(i_0, i_1, \ldots, i_M)$. The same procedure is applied to the end nodes in skeleton G'. Given this ordering for the end nodes, this set can be used for the creation of the path distance matrix, and, as a meaningful order is now at hand, an order-preserving function like OSB can be used for finding the similarity between a pair of end nodes.

Figure 5.6 shows an example for this procedure. The number of mismatches in the shown example was reduced to five mismatches out of 14.

However, this approach is sensitive to noisy input data, where corresponding skeleton branches do not have similar lengths, as was the case in all of the available data sets. The order of the end nodes in this case might not be the same for the two skeletons. Several experiments even showed that for most of the available aorta images the implicit order of the end nodes by their occurance in the 3D space is a more reliable ordering.

5.4 Similar disc radii

This issue does not stem from applying the algorithm to 3D data, but from the special application in matching blood vessel volumes. The problem is that the disc radii are quite similar for all skeleton points in this special kind of data. The disc radii are hence a less significant indicator for similar skeleton paths than in conventional test shapes. Due to the fact that the preoperative volume shows an aneurysm, the radii are even expected to vary slightly for corresponding paths. Moreover, three of the available nine test volumes are perforated due to segmentation errors. In these cases, the disc radii vary strongly even for corresponding paths.

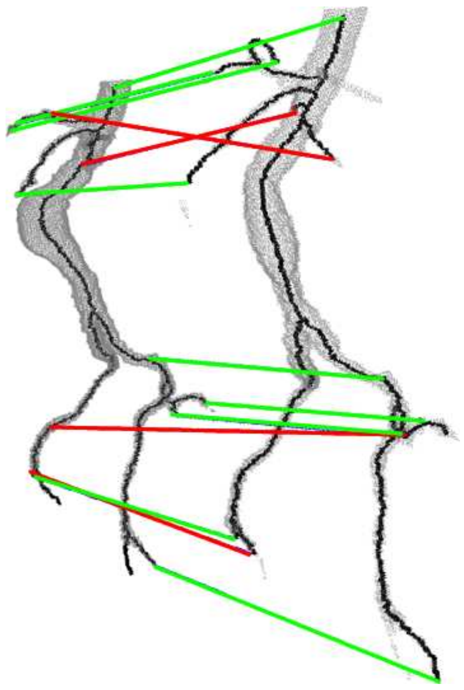


Figure 5.6: In this matching, the skeletons' end nodes are ordered by their distance to the start node, so that the OSB function can be used to extract the similarity costs for two end nodes.

Thus, an idea for a more reliable approach in this application would be to replace the path distance definition as defined in equation 4.3. In this definition, the skeleton path vectors of the radii of the maximal inscribed discs are the most important parameter for determining the similarity between two skeleton paths. The assumption is that for several cases, the path length is a more significant indicator for path similarity. Therefore, equation 4.3 is substituted with the following definition:

$$pd(p(u,v), p(u',v') = (l_i - l'_i)^2 \tag{5.1}$$

The equation has become much simpler: The part with the disc radii was removed completely. Instead, only the path lengths are incorporated. Thus, the weight factor α is not needed anymore. The denominator, $l_i + l'_i$, also has been removed from the equation, only the absolute difference between the path lengths is incorporated.

First experiments showed that in the perforated volumes this new path distance definition is a reliable alternative to the one introduced in the original paper. However, for most cases, the path length alone is not a reliable similarity measure as several branches are shortened at different levels.

5.5 Dealing with noise

As mentioned before, the skeletons of the blood vessel volumes contain spurious branches that are not contained in the other skeleton and thus, both skeletons contain end nodes that cannot be matched non-ambitigously to an end node in the other skeleton. Also, branches may be shortened or cut off in different levels in the two shapes, so that correct correspondences are difficult to find. This can have a negative impact on the matching results, as shown in the previous examples. In the available data sets, it is difficult to find a similarity measure for skeleton end nodes that leads to perfect matching results.

Thus, to improve the quality of the matching results, it is necessary to deal with this noise. In general, there are three possible approaches to deal with this problem:

1. *Preprocess the skeletons:* One possibility is to preprocess the skeletons so that at least some of the spurious branches are removed before the matching process.

2. *Skipping elements in the final matching:* An alteration in the matching process could be performed that allows the skipping of end nodes in the final matching.

3. *Apply filter to correspondences:* The third possibility is to filter the found correspondences after the matching process according to a predefined significance indicator.

All of these approaches will lead to results where not all end nodes are matched, but the total quality of the matching result will increase.

5.5.1 Preprocessing the skeletons

A relevance measure for skeleton branches in order to remove the spurious branches is desirable. However, preprocessing the skeletons before the matching process is difficult as no semantic information about the skeletons is known at this point. Thus, it is difficult to determine which branches should be removed before further computations.

However, it is possible to determine the significance of a skeleton branch based on its length. Shorter branches are generally less significant for the representation of the shape compared to longer branches. It is possible to eliminate these short branches: For each end node, the length of the emanating skeleton branch is determined. All branches that have a length smaller than a given threshold are deleted.

This procedure does not guarantee better results. In fact, the results can get even worse as it can lead to branches being removed in one skeleton, but the corresponding branches in the other skeleton are not. Finding the optimal threshold to get better results is a difficult task and not all spurious branches will be found. As can be seen in the previous examples, spurious branches are not always short enough that they would be eliminated in this process, without deleting other, more significant branches.

5.5.2 Skipping elements in the final matching

So far, the final matching was always performed by the Hungarian algorithm which enforces one-to-one correspondences. However, one-to-one correspondences are not always possible. An alteration to the final matching process is desirable that also allows the skipping of elements.

One possibility is to extend the Hungarian algorithm to also allow the skipping of elements in the cost matrix. The easiest way to do so is to add additional dummy rows and columns to the cost matrix, besides the ones that are needed to make the matrix square. These additional rows and columns can be seen as additional dummy nodes in both skeletons. These dummy nodes act like a buffer: If the assignment of two elements is too expensive, they will be matched to one of the dummy nodes instead.

Figure 5.7 shows a matching of the previous example volumes, with four additional nodes added before applying the Hungarian algorithm. As can be seen in this example, this alteration leads to additional mismatches in the lower part of the blood vessel, but the wrong matchings in the top were eliminated. All in all, the number of mismatches is reduced to two out of 10, but this also comes with the loss of correct matches that could have been found otherwise. The challenge with this approach is, however, to find the best number of additional dummy nodes. If the number is too small, mismatches will still occur. If the number is too high, possible correct matchings might not be found. The experiments showed that too few or too many additional nodes in the Hungarian algorithm can even lead to more mismatches.

Another more robust approach would be to substitute the Hungarian algorithm in the final matching for another matching algorithm that allows the skipping of elements. One possibility would be to replace the Hungarian algorithm by the OSB function. The problem again is that OSB requires a meaningful order of the end nodes. Otherwise, the cheapest path in the matrix cannot be found. The order already introduced for applying the OSB in the intermediate step to extract a similarity value from a path distance matrix can be used at this point: The distance from any end node to the start node can be used as order criterion. The remaining question is how to choose the “start node”. The solution is to apply OSB multiple

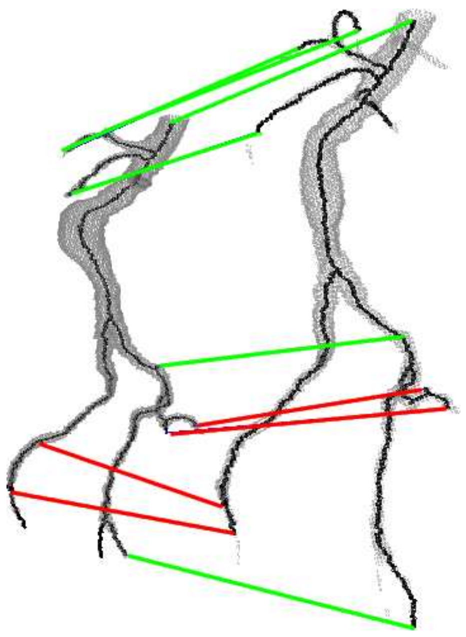


Figure 5.7: In this matching, four additional nodes were added before applying the Hungarian algorithm. The original path distance definition as described in [BL08] was used, with the parameters $\alpha = 70$ and $M = 50$.

times, once for each combination of end nodes. For each combination of end nodes i and i' in the two skeletons G and G' the following procedure is performed: The end nodes of skeleton G are ordered according to their distance to end node i, measured by the length of the skeleton path between the two skeleton end nodes. The same procedure is applied to end node i' and skeleton G'. This results in two ordered lists of skeleton end nodes $\{i, i_0, i_1, \ldots, i_M\}$ for G and $\{i', i'_0, i'_1, \ldots, i'_N\}$ for G'. Based on this order, a new cost matrix can be built as input matrix for the OSB function: The end nodes are charted in the cost matrix's rows and columns, respectively, according to their order in the ordered list. The cell (i_m, i'_n) in the cost matrix contains the assignment cost between the two end nodes i_m and i'_n. The OSB function can be applied to the resulting cost matrix to find corresponding end nodes while skipping elements that are too expensive to be aligned.

This procedure is done for each combination of end nodes in the two skeletons. The "best" matching, that is, the matching with the lowest total matching costs, is assumed to be the right matching.

Figure 5.8 shows an example for this procedure. The number of wrong correspondences has been reduced to four mismatches out of 13.

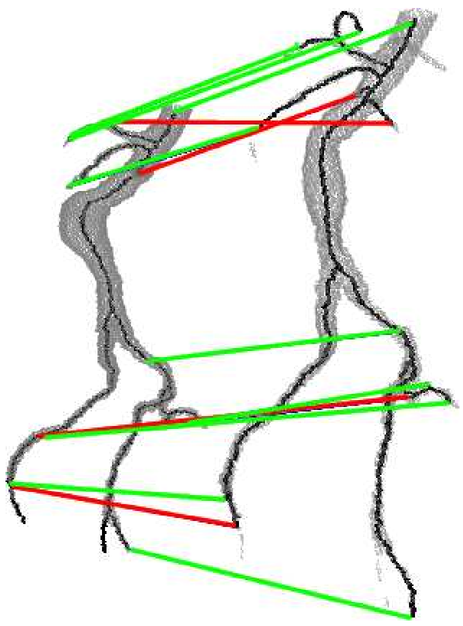


Figure 5.8: The original path distance definition was used in this example, with $\alpha = 70$ and $M = 50$. The OSB function has been used to compute the final matching.

5.5.3 Apply filter to correspondences

The previous examples have shown that it is difficult to define a similarity measure based on the skeleton paths when dealing with noisy data. Especially if both skeletons contain end nodes with no matching partner in the other skeleton optimal matchings (with few or even no mismatches) are not possible. The idea is now to filter the result, so that mismatches are deleted.

For example, the matching result for the scenario shown in figure 5.5 should be filtered. One solution is to filter the found matchings according to their matching costs. Figure 5.9 shows a graph with the matching costs. The curve's upward trend starting at $x = 12$ indicates that the mismatches indeed possess higher matching costs. Thus, the filtering of matchings with high matching costs could be a possible solution to eliminate mismatches.

The remaining question to be dealt with is how to find a threshold for the filtering. The first approach tested is to compute the mean of all matching costs, and then delete all matchings with a computed matching cost that is higher then the mean. Another threshold could be found by computing the median of all assignment costs. The first experiments have shown that in general, when using the median as a threshold, more matchings are eliminated than when using the mean. Thus, more mismatches are eliminated, but this comes at the cost that also correct matchings are deleted.

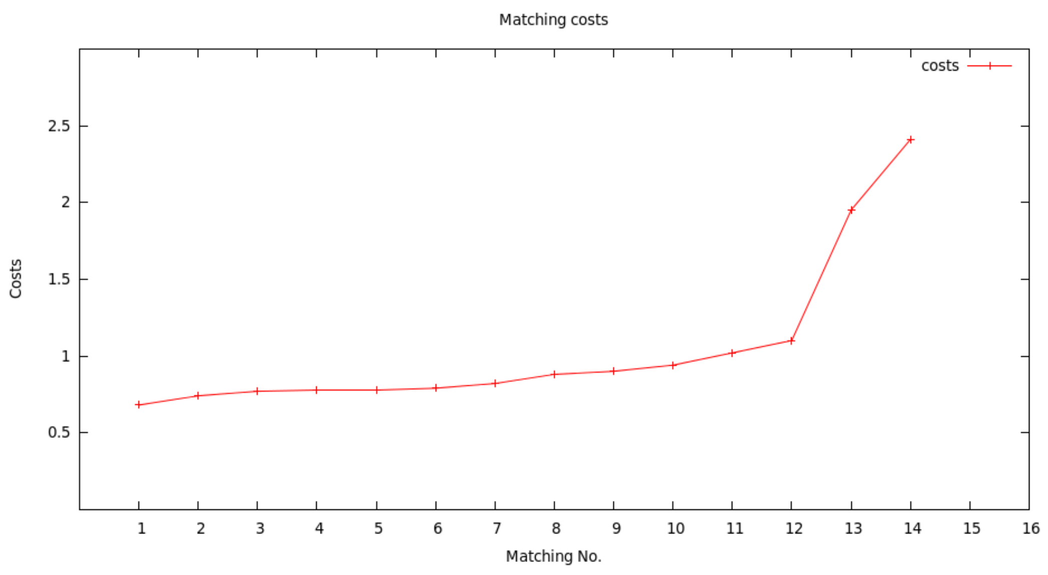


Figure 5.9: Graph showing the increasing matching costs.

Filtering by matching costs is a general approach and could also be applied to other 3D shapes as well. Another filter approach that can be used especially in medical data is the filtering by angle: As the two volumes are aligned to each other in the 3D space, the lines indicating correct assignments stand in a similar angle to each other. Thus, one can compute the average angle in x, y and z direction between these matching lines and filter those matchings whose matching lines differ too much from the average angle in any direction. First experiments have shown that after filtering the result with this simple approach only few matchings remain, in several cases even all matchings are deleted. Future work could thus include a better formula for filtering the matching results by angle.

5.6 Experiments

In the previous section, ideas were introduced to make the algorithm in [BL08] more robust for the matching of 3D data, especially the matching of aorta volumes. In summary, these ideas are:

- Order the end nodes by distance, or replacing OSB function by Hungarian algorithm
- Use a different definition for path distances
- Filter the found matchings

- Replace the Hungarian algorithm in the final matching with OSB.

The single approaches also can be combined with each other to achieve better results. The question is which combination of these ideas performs best on the available data sets. Different configurations in the algorithm can have a significant effect on the matching result.

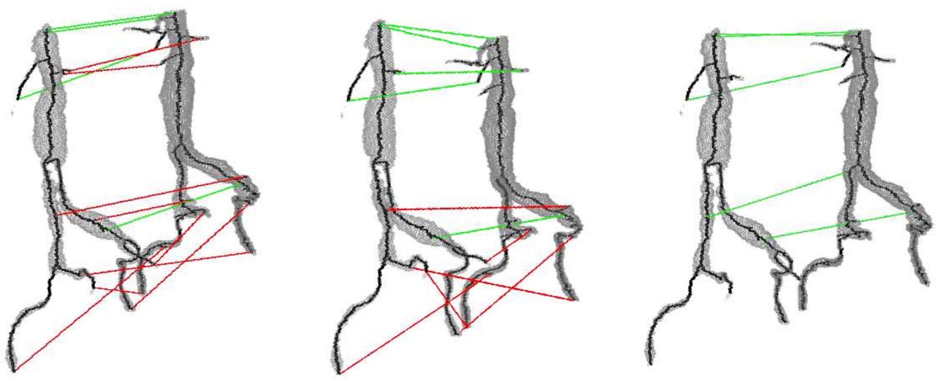


Figure 5.10: Different configurations in the final matching algorithm. Figure 5.10a: Hungarian Algorithm was used for the final matching. Figure 5.10b: Hungarian Algorithm was used for the final matching, and two dummy nodes were added. Figure 5.10b: OSB was used for the final matching.

For instance, figure 5.10 shows the impact of the final matching algorithm in one data set. Different configurations of the final matching were applied to one dataset. In this example, the configuration vary only in the final matching. In all configurations, no explicit order was used for the end nodes. The path distances are defined by the radii, but without denominator. The figure shows that less matchings are found in total when OSB is used instead of Hungarian algorithm for the final matching, as several assignments with high costs are skipped. This comes with the disadvantage that several correct matchings cannot be found, but most of the false matchings (in the shown example all of them) are filtered out.

Figure 5.11 shows the matching results in another dataset, with variations in the definition of the path distance. All remaining parameters stayed the same in the three setups: The end nodes were not explicitly ordered, OSB has been used to extract the similarity values of the path distance matrices and the final matching was also performed by applying OSB. By substituting the original path distance definition from the paper, the number of mismatches in this example could be reduced from four to two.

Of course, another important parameter influencing the matching result is the quality of the input skeletons. Figure 5.12 shows the image of the same aorta

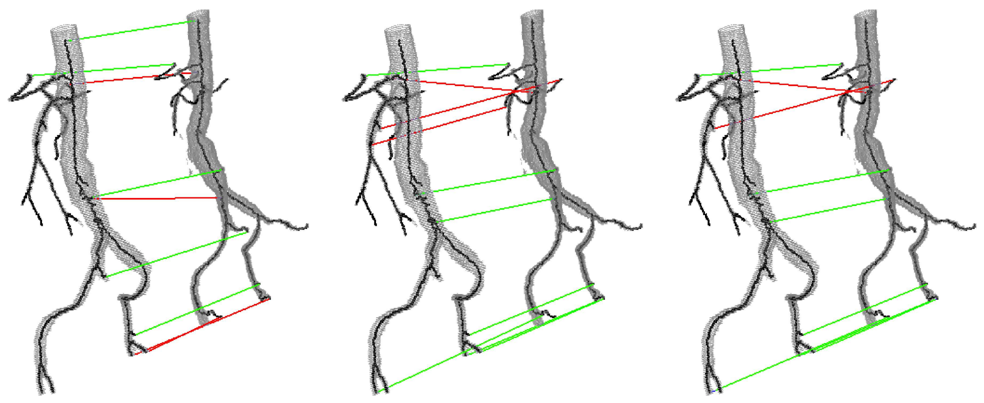


Figure 5.11: Different configurations of the path distance definition. Figure 5.11a: Original path distance from the paper. Figure 5.11b: Path distance by the path length, including the denominator proposed in the original paper. Figure 5.11c: Path distance by path length, without denominator.

data, segmented and skeletonized with different parameters. In both matching processes, the end nodes were not ordered. The path distances were determined by the path length only - due to the occuring holes in one of the volumes, the radii are not meaningful. The OSB function has been used to compute the matching costs between two end nodes, and OSB has been used for the final matching. The results were filtered by the mean of all matching costs. Spurious branches lead to three mismatches in figure 5.12a, while the same setup with the "cleaner" skeletons lead to no mismatches.

First experiments showed that the best approach is:

- The end nodes should not be ordered. In several cases the level of skeleton branch shortening is too different for both skeletons. Thus, the proposed order by length is more an approximation than a real order. The implicit order of the end nodes by their occurence in the 3D cube has shown to be a more reliable order than the one introduced before.

- OSB should be used to compute the similarity between two end nodes. The noisy data leads to several spurious branches in the skeletons and makes skipping skeleton elements necessary, which is not possible with the Hungarian algorithm.

- OSB should be used for the final matching. Due to spurious branches, it is necessary to be able to skip elements in the final matching process. Adding dummy nodes to the Hungarian algorithm also deals with this problem, but the number of nodes to be added varies depending on the data. OSB is more

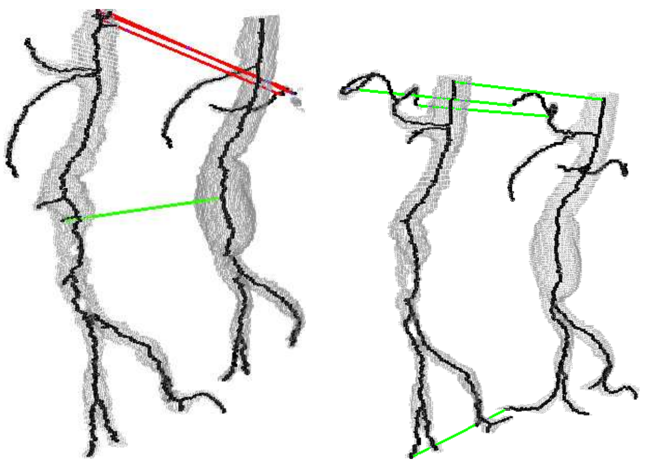


Figure 5.12: Impact of the skeleton quality on the matching result. The volumes shown are the result of the same image, but segmented with different parameters.

reliable for that purpose without manually setting additional parameters. Using OSB instead of the Hungarian algorithm usually leads to the loss of some correct matchings as well, but the overall quality of the matching result increases by being able to skip spurious branches.

The only parameter where no general solution could be found was the definition of the path distance. The choice of the path distance definition highly depends on the quality of the input skeleton, as shown in figure 5.13.

Unfortunately, the number of available data sets is very small. This makes a proper evaluation impossible. Only nine pairs of pre and post surgery blood vessel images are available in total, three of them contain distortions the algorithm can only handle by usage of alternative path distance definitions, as for example shown in figure 5.13. As this kind of issue is more a problem of the segmentation and skeletonization and the corresponding data is not really helpful in evaluating the algorithm, these data sets were left out in the experiments. All experiments performed in the context of this thesis are therefore only approximate values. It is not possible to determine the approach's performance properly with this little data.

The question is how the matching algorithm can be evaluated for the matching of aorta volumes. While the algorithm's performance for 2D shapes was tested by performing similarity searches in three shape databases, this procedure is not practical for the matching of aorta images. Instead, the correct matchings in each data set have been counted for this purpose. The definition of a "correct" matching is difficult in the context of noisy blood vessel data. As most of the skeletons

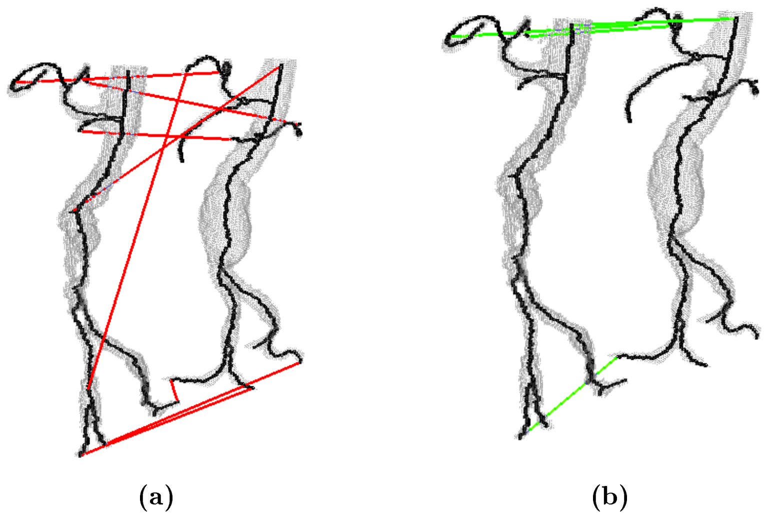


Figure 5.13: The impact of the path distance definition on the matching result. Obviously, in the shown example the path radii are no reliable indicator for path similarity: Due to the accrued holes in one of the volumes, the radii at the skeleton paths differ too much between the two skeletons. Figure 5.13a: The path distance is computed as in the original paper. Figure 5.13b: The path distance is computed by the difference in the path length.

contain shortened or cut off branches it is often not possible to determine correct or false matchings.

Therefore, the experiments are performed as follows: First, a ground truth is established by establishing correspondences between the end nodes in all data pairs manually. In this ground truth, only unambigitous end node correspondences were established in order to avoid distorted results. End nodes with no unique matching partner in the other skeleton are ignored as they would lead to distorted results. In the next step, the algorithm is applied to each pair of pre- and postoperative images with the parameters described above. The found end node matchings are compared to the ground truth. Matchings involving only end nodes that were not matched in the ground truth are ignored as they cannot be classified unambigitously. All found assignments are classified according to the following definitions:

- If the assignment involves at least one end node m that is contained in any ground truth assignment, this assignment is further examined:

  - If the assignments maps m to the same end node n in the respective other skeleton as in the ground truth, this assignment as classified as correct.
  - If the assignment does not map m to the same end node n in the respective other skeleton as in the ground truth, this assignment is classified as wrong.

- If the end node matching involves no end node that is contained in any ground truth matching, the matching is classified as noise.

The results for the six data pairs available are shown in table 5.1. The table

	ground truth	correct	wrong	noise
1	8	5	2	11
2	9	5	4	4
3	9	6	3	4
4	11	10	1	1
5	3	1	1	9
6	7	4	3	6
total	47	31	14	35

Table 5.1: Summary of correct and wrong matchings in the six pre- and postoperative data pairs. The single pairs are charted in the rows. The column "ground truth" indicates the total number of manually established correspondences in the ground truth. The last three columns chart the number of end node matchings found by the algorithm, classified by the characteristica described before.

shows that 31 of the 47 matchings established manually in the ground truth were found by the algorithm. In total, 14 mismatches were found, and 35 of the found matchings were classified as noise.

By additionally filtering the results the number of mismatches and noise can be reduced significantly, but this comes at the cost that also correct matchings are eliminated. In the filtered results, only 22 of the initial 47 ground truth matchings were found, but only seven matchings were classified as wrong, and only 12 matchings were classified as noise. This behaviour is shown in table 5.2.

If one examines the mismatches in detail, one realizes that mismatches occur mainly due to the following reasons:

- The skeleton paths generally are quite similar in the aorta volumes. This was already shown in figure 5.3. It is thus difficult to distinguish skeleton paths within one volume from each other, only based on the path radii. Hence, several end nodes get switched in the matching result because the skeleton

	ground truth	correct	wrong	noise
1	8	5	2	2
2	9	4	2	3
3	9	4	2	1
4	11	4	0	0
5	3	1	1	3
6	7	4	0	3
total	47	22	7	12

Table 5.2: Summary of correct and wrong matchings in the six pre- and postoperative data pairs. Correspondences with a matching cost higher than the mean of all correspondences are filtered out.

paths emanating from them are similar. The problem becomes even clearer if one inspects the application of the algorithm: The data to be matched are a blood vessel before a surgery - with an aneurysm - and after surgery - without an aneurysm. As an aneurysm is a bulge in the wall of a blood vessel, the radii are likely to differ between the two images.

- The blood vessel volumes contain several junction nodes from which similar short branches emanate. An example is shown in figure 5.14. The end nodes at those branches are very similar in their skeleton paths and thus, are likely to be switched.

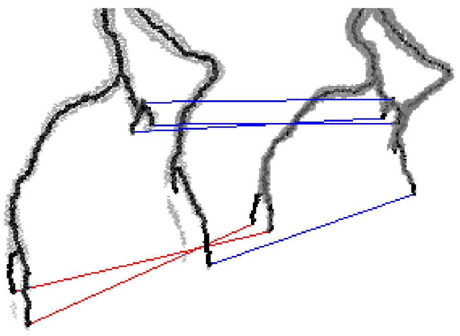


Figure 5.14: Example for a mismatch due to similar radii, highlighted in red. The two end nodes cannot be distinguished based on the path radii.

- Several end nodes have no obvious matching partner in the other skeleton. Due to that ambiguity and the fact that the path radii are quite similar for all skeleton points, it is likely that skeleton paths get switched in the matching process.

Chapter 6

Conclusion and future work

This diploma thesis consists of two parts: First, an existing skeleton based 2D shape matching algorithm was introduced and analyzed for strengths and limitations. Then, ideas have been developped to extend the described algorithm for the matching of 3D data.

The matching of salient features of a shape is a key component in several applications such as image retrieval or object recognition. The matching algorithm introduced in this thesis uses the skeleton as a shape descriptor. Skeletons are a reduction of the original shape: 2D objects can be abstracted to one-dimensional curves, 3D objects can be represented by planes or curves. Skeletons "contain both shape features and topological structures of original objects" [BLL07]. The goal of the algorithm is to match the end nodes of a shape's skeleton. The end nodes of a skeleton hold important geometric information about the shape. In addition, they are an interesting contour feature as they are part of the skeleton as well as part of the contour. The algorithm is based on a special skeleton representation that not only incorporates the skeleton structure, but also contour information of the shape, like width. The skeleton's topology is not explicitly incorporated. Instead, the similarity between the shortest paths connecting two end points are used to compute the similarity between two end points.

In the experiments for 2D, the algorithm showed its advantages when dealing with non-rigid objects and articulated joints. An average precision of 0.93 (0.98 with manually pruned skeletons, respectively) for the Aslan and Tari database shows that shape deformations that do not affect the skeleton topology or the path radii have no impact on the matching results. The experiments with the Kimia-99 and Kimia-216 database also showed acceptable results with an average precision of 0.84 and 0.81, respectively.

However, problems in the recognition performance occured when shapes of different classes were similar. In addition, overlaps had a negative impact on the recognition results. A severe limitation of the algorithm is that it requires optimal skeletons. The experiments showed that spurious branches in one of the skeletons lead to distorted matching results in several cases, affecting the object recognition performance when using the described matching algorithm in a retrieval system.

This sensitivity to noisy skeletons occuring in the 2D experiments poses also one of the problems in the matching of the 3D data. Another problem for the extension of the algorithm to match 3D data is that in 2D, the order of the end nodes as obtained by traversing the contour is an important matching indicator, holding important information about the shape. As it is not possible to traverse the contour in 3D, this important information about the object shape cannot be used anymore. First experiments showed that as an alternative idea the ordering of the skeleton end nodes by their distance to each other was not reliable enough when dealing with noisy data. In the context of the matching of CT images the implicit order of the end nodes by their occurance in the 3D cube posed a more reliable order criterion a sthe point of view in this type of image is the same, and the arrangement of the volume in the 3D cube can assumed to be similar.

In general, the matching of blood vessel volumes as done in the experiments, is a challenging problem for any matching algorithm. The skeletons contain several spurious branches with no obvious matching partner in the other skeleton. On top of that, all skeleton branches are quite similar to each other as the radii of the maximal inscribed discs hardly vary for any skeleton points. The sensitivity to spurious branches of the introduced algorithm, the lack of contour information and less significant skeleton path radii information thus are all expected to have a negative impact on the matching results.

Due to the lack of testing data no reliable experiments could be performed, but the first analysis showed that the algorithm is able to find correct matchings in optimal skeletons, but is quite prone to noisy data. In the first analysis, 31 of 47 matchings in the manually established ground truth could be found, but also 14 mismatches occured. The matching results are expected to improve if skeletons with less noise are used.

Noisy data poses a problem when choosing an optimal configuration for the algorithm: While in optimal skeletons, the original path distance definition usually leads to acceptable results, an alternative has to be used for several cases, for example, if holes occured in the volumes due to segmentation errors. Another problem is that due to noisy data, a filtering of the matching results is necessary. In the experiments, matchings were deleted from the result if their matching cost was higher than a given threshold, for example the mean of all found matching costs. The problem hereby is that due to the fact that the skeleton paths all are quite similar in this kind of data, the matching costs are not the optimal indicator for a wrong matching: Several of the wrong end node matchings could be eliminated by the introduced methods, but this comes with the cost that also several correct matchings are eliminated. In the filtered matching results, the mismatches were reduced to seven, but only 22 of the initial 47 ground truth matchings could be found.

A first step to a more reliable matching algorithm for blood vessels was done by replacing the Hungarian algorithm by applying the OSB function multiple times. That way, no one-to-one correspondence is enforced anymore which was one of the main problems when dealing with spurious skeleton branches. Using OSB instead of the Hungarian algorithm led too less mismatches, as the skipping of elements is possible. Thus, skipping elements in the final matching is an important step to improve the matching results. However, the OSB approach is limited by the fact that it requires a meaningful order of the end nodes. An alternative would be to combine the approach with other methods: Using a matching model that also allows for partial matching, but does not require an explicite order of the end nodes, like the *Earth Mover's distance* [DSK$^+$06], could lead to more robustness in this approach.

The introduced method is a generic approach that can be applied to any kind of data. For the challenging task of matching blood vessel volumes, further improvements are necessary. Single branches cannot be matched unambiguously only based on the path radii, as they are too similar and likely to be switched in the matching process. One possibility would be to further involve the skeleton's topology in the matching process. For example, the skeleton's junction nodes could be included in the matching process. In the original paper, the junction nodes were disregarded as the authors argued that junction nodes are not reliable in non-rigid objects. However, in the application of the matching of blood vessel volumes, the junction nodes also carry important information about the aorta. Though aorta volumes are, of course, non-rigid objects, they are usually hardly deformed in a CT scan. Xu et. al. [XWB09] present an approach to involve junction nodes in the matching process based on skeleton paths for 2D shapes. Another approach to incorporate the junction nodes in the matching process would be to involve the angles of the emanating branches at junction nodes, as they are expected to be similar for aorta images of the same patient.

Bibliography

[AT05] Aslan, Cagri ; Tari, Sibel: An Axis-Based Representation for Recognition. In: *Proceedings of the Tenth IEEE International Conference on Computer Vision*, 2005, S. 1339–1346

[BC94] Berndt, Donald J. ; Clifford, James: Using Dynamic Time Warping to Find Patterns in Time Series. In: *KDD Workshop*, 1994, S. 359–370

[Ber95] Bertrand, Gilles: A parallel thinning algorithm for medial surfaces. In: *Pattern Recognition Letters* 16 (1995), Nr. 9, S. 979–986

[BI04] Brennecke, Angelika ; Isenberg, Tobias: 3D Shape Matching Using Skeleton Graphs. In: *Simulation and Visualization*, 2004, S. 299–310

[BL08] Bai, Xiang ; Latecki, Longin J.: Path Similarity Skeleton Graph Matching. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30 (2008), S. 1282–1292

[BLL07] Bai, Xiang ; Latecki, Longin J. ; Liu, Wen-Yu: Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 29 (2007), S. 449–462

[BLT09] Bai, Xiang ; Liu, Wenyu ; Tu, Zhuowen: Integrating contour and skeleton for shape classification. Version: 2009. `http://sites.google.com/site/xiangbai/animaldataset`. IEEE Workshop on NORDIA, 2009. – Forschungsbericht

[Blu67] In: Blum, H.: *A transformation for extracting new descriptors of shape*. Bd. Models for the Perception of Speech and Visual For. MIT Press, 1967, S. pp. 362–380

[BM76] Bondy, J.A. ; Murty, U. S. R.: *Graph theory with applications*. London, UK : Macmillan, 1976

[BMP02] Belongie, Serge ; Malik, Jitendra ; Puzicha, Jan: Shape Matching and Object Recognition Using Shape Contexts. In: *IEEE Trans. Pattern Anal. Mach. Intell* 24 (2002), Nr. 4, S. 509–522

[Cas10] Castello, Beryl: *Introduction to Optimization.* Course Notes, 2010

[CDS$^+$05] Cornea, Nicu D. ; Demirci, M. F. ; Silver, Deborah ; Shokoufandeh, Ali ; Dickinson, Sven J. ; Kantor, Paul B.: 3D Object Retrieval using Many-to-many Matching of Curve Skeletons. In: *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005.* Washington, DC, USA : IEEE Computer Society, 2005, S. 368–373

[Cha07] Chang, Sukmoon: Extracting Skeletons from Distance Maps. In: *IJCSNS International Journal of Compute Science and Network Security* 7 (2007)

[CR03] Chen, Hui ; Rangarajan, Anand: A new point matching algorithm for non-rigid registration. In: *Computer Vision and Image Understanding* 89 (2003), Nr. 2-3, S. 114–141

[CS07] Cornea, Nicu D. ; Silver, Deborah: Curve-skeleton properties, applications, and algorithms. In: *IEEE Transactions on Visualization and Computer Graphics* 13 (2007), S. 530–548

[CSM07] Cornea, Nicu D. ; Silver, Deborah ; Min, Patrick: Curve-Skeleton Properties, Applications, and Algorithms. In: *IEEE Transactions on Visualization and Computer Graphics* 13 (2007), S. 530–548

[DGM97] Das, Gautam ; Gunopulos, Dimitrios ; Mannila, Heikki: Finding Similar Time Series. In: *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery.* London. UK : Springer London, 1997, S. 88–100

[Die05] Diestel, Reinhard: *Graph Theory (Graduate Texts in Mathematics).* Springer Verlag, 2005

[DP81] Davies, E.R. ; Plummer, A.P.N.: Thinning algorithms: A critique and a new methodology. In: *Pattern Recognition* 14 (1981), Nr. 1-6, S. 53 – 63. – 1980 Conference on Pattern Recognition

[DS06] Dey, Tamal K. ; Sun, Jian: Defining and computing curve-skeletons with medial geodesic function. In: *Proceedings of the fourth Eurographics symposium on Geometry processing.* Aire-la-Ville, Switzerland, Switzerland : Eurographics Association, 2006, S. 143–152

[DSD09] Demirci, M. F. ; Shokoufandehand, Ali ; Dickinson, Sven: Skeletal Shape Abstraction from Examples. In: *IEEE Trans. Pattern Anal. Mach. Intelligence* 31 (2009), Nr. 5, S. 944–952

[DSK$^+$06] Demirci, M. F. ; Shokoufandeh, Ali ; Keselman, Yakov ; Bretzner, Lars ; Dickinson, Sven J.: Object Recognition as Many-to-Many Feature Matching. In: *Int. J. Comput. Vision* 69 (2006), S. 203–222

[DWT06] Dortmont, M. A. M. M. ; Wetering, H. M. M. d. ; Tele, Alexandru: Skeletonization and Distance Transforms of 3D Volumes Using Graphics Hardware. In: *DGCI*, 2006, S. 617–629

[FPWW04] Fisher, Robert ; Perkins, Simon ; Walker, Ashley ; Wolfart, Erik: *Image Processing Learning Ressources.* `http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm`. Version: 2004

[FS08] Faller, Adolf ; Schünke, Michael: *Der Körper des Menschen. Einführung in Bau und Funktion.* 15. Thieme Flexible Taschenbücher, 2008

[GG73] Gray, Henry ; Goss, Charles M.: *Anatomy of the human body.* 29. Philadelphia : Lea & Febiger, 1973

[HBK01] Hisada, Masayuki ; Belyaev, Alexander G. ; Kunii, Tosiyasu L.: A 3D Voronoi-Based Skeleton and Associated Surface Features. In: *Pacific Conference on Computer Graphics and Applications* 0 (2001)

[HHW04] He, Lei ; Han, Chia Y. ; Wee, William G.: Graph matching for object recognition and recovery. In: *Pattern Recognition* 37 (2004), Nr. 7, S. 1557–1560

[Jun90] Jungnickel, Dieter: *Graphen, Netzwerke und Algorithmen.* Wissenschaftsverlag Mannheim/Wien/Zürich, 1990

[Jun07] Jungnickel, Dieter: *Graphs, Networks and Algorithms.* 3. Springer Publishing Company, 2007 [Jun90]

[KG10] Kim, Jaechul ; Grauman, Kristen: Asymmetric region-to-image matching for comparing images with generic object categories. In: *CVPR*, 2010, S. 2344–2351

[KSK01] Klein, Philip N. ; Sebastian, Thomas B. ; Kimia, Benjamin B.: Shape matching using edit-distance: an implementation. In: *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete* *algorithms.* Philadelphia, PA, USA : Society for Industrial and Applied Mathematic, 2001, S. 781–790

[KSSK00] KLEIN, Philip ; SRIKANTA, Tirthapura ; SHARVIT, Daniel ; KIMIA, Ben: A tree-edit-distance algorithm for comparing simple, closed shapes. In: *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms.* Philadelphia, PA, USA : Society for Industrial and Applied Mathematic, 2000, S. 696–704

[Kuh55] KUHN, Harold W.: The Hungarian Method for the assignment problem. In: *Naval Research Logistics Quarterly* 2 (1955), S. 83–97

[KWT88] KASS, Michael ; WITKIN, Andrew ; TERZOPOULOS, Demetri: Snakes: Active contour models. In: *International Journal of Computer Vision* 1 (1988), Nr. 4, S. 321–331

[LG07] LOHOU, Christophe ; GILLES, Bertrand: Two symmetrical thinning algorithms for 3D binary images, based on P-simple points. In: *Pattern Recognition* 40 (2007), S. 2301–2314

[Lie04] LIEUTIER, Andre: Any open bounded subset of Rn has the same homotopy type as its medial axis. In: *Comput.-Aided Des.* 36 (2004), S. 1029–1046

[LKMT] LARSEN, Jeppe V. ; KNUDSEN, Lars ; MADSEN, Rasmus K. ; TAKLE, Christian M.: *Skeletonization using distance transform.* `http://www.cvmt.dk/education/teaching/f10/MED8/CV/Stud/838.pdf`

[LL92] LEYMARIE, Frederic ; LEVINE, Martin D.: Simulating the Grassfire Transform Using an Active Contour Model. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14 (1992), S. 56–75

[LL99a] LATECKI, Longin J. ; LAKÄMPER, Rolf: Convexity rule for shape decomposition based on discrete contour evolution. In: *Comput. Vis. Image Underst.* 73 (1999), S. 441–454

[LL99b] LATECKI, Longin J. ; LAKÄMPER, Rolf: Polygon Evolution by Vertex Deletion. In: *Proceedings of the Second International Conference on Scale-Space Theories in Computer Vision.* London, UK : Springer Verlag, 1999, S. 398–409

[LLE00] LATECKI, L. J. ; LAKÄMPER, Rolf ; ECKHARDT, T.: Shape descriptors for non-rigid shapes with a single closed contour. In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* Bd. 1, 2000, S. 424–429 vol.1

[LLS92] Lam, Lousia ; Lee, Seong-Whan ; Suen, Ching Y.: Thinning Methodologies-A Comprehensive Survey. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14 (1992), S. 869–885

[LMW$^+$05] Latecki, Longin J. ; Megalooikonomou, Vasileios ; Wang, Qiang ; Lakaemper, Rolf ; Ratanamahatana, C. A. ; Keogh, E.: Partial Elastic Matching of Time Series. In: *Proceedings of the Fifth IEEE International Conference on Data Mining.* Washington, DC, USA : IEEE Computer Society, 2005, S. 701–704

[Low04] Lowe, David G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision* 60 (2004), Nr. 2, S. 91–110

[LP09] Lakshmi, J. K. ; Punithavalli, M.: A Survey on Skeletons in Digital Image Processing. In: *Proceedings of the International Conference on Digital Image Processing.* Washington, DC, USA : IEEE Computer Society, 2009, S. 260–269

[LWKTM07] Latecki, Longin J. ; Wang, Qiang ; Koknar-Tezel, Suzan ; Megalooikonomou, Vasileios: Optimal Subsequence Bijection. In: *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining.* Washington, DC, USA : IEEE Computer Society, 2007, S. 565–570

[Mai99] Maillet, S. M.: *Binary Digital Image Processing: A Discrete Approach.* Academic Press, 1999

[Mon69] Montanari, Ugo: Continuous Skeletons from Digitized Images. In: *J. ACM* 16 (1969), S. 534–549

[Mun57] Munkres, James: Algorithms for the Assignment and Transportation Problems. In: *Journal of the Society of Industrial and Applied Mathematics* 5 (1957), S. 32–38

[Ogn92] Ogniewicz, Robert L.: *Discrete Voronoi Skeletons*, Swiss Federal Institute of Technology Zurich, Diss., 1992

[OI92] Ogniewicz, R. ; Ilg, M.: Voronoi Skeletons: Theory and Applications. In: *CVPR92*, 1992, S. 63–69

[OK95] Ogniewicz, R.L. ; Kübler, O.: Hierarchic Voronoi skeletons. In: *Pattern Recognition* 28 (1995), Nr. 3, S. 343–359

[Pal] Palágyi, Kálmán: *Skeletonization.* `http://www.inf.u-szeged.hu/~palagyi/skel/skel.html`

[Pal08] Palagyi, Kalman: A 3D fully parallel surface-thinning algorithm. In: *Theoretical Computer Science* 406 (2008), S. 119–135

[PK98] Palagyi, Kalman ; Kuba, Attila: A 3D 6-subiteration thinning algorithm for extracting medial lines. In: *Pattern Recogn. Lett.* 19 (1998), S. 613–627

[PSS+03] Pizer, Stephen M. ; Siddiqi, Kaleem ; Szekely, Gabor ; Damon, James N. ; Zucker, Steven W.: Multiscale Medial Loci and Their Properties. In: *Int. J. Comput. Vision* 55 (2003), S. 155–179

[QSO04] Quadros, W.R. ; Shimada, K. ; Owen, S. J.: 3D discrete skeleton generation by wave propagation on PR-octree for finite element mesh sizing. In: *Proceedings of the ninth ACM symposium on Solid modeling and applications*, 2004, S. 327–332

[Ren09] Reniers, Dennie: *Skeletonization and segmentation of binary voxel shapes*, Technische Universiteit Eindhoven, Diss., 2009

[RHL11] Rückert, Ralph ; Hepp, Wolfgang ; Luther, Bernd: Chirurgie der abdominalen und thorakalen Aorta. In: *Chirurgie der abdominalen und thorakalen Aorta.* Berliner Gefäßchirurgie, 2011

[RJP00] Reinders, Freek ; Jacobson, Melvin E. D. ; Post, Frits H.: Skeleton Graph Generation for Feature Shape Description. In: *IEEE TCVG Symposium on Visualization*, Springer Verlag, 2000, S. 73–82

[RTG98] Rubner, Yossi ; Tomasi, Carlo ; Guibas, Leonidas J.: A Metric for Distributions with Applications to Image Databases. In: *Proceedings of the Sixth International Conference on Computer Vision.* Washington, DC, USA : IEEE Computer Society, 1998, S. 59–

[RTG00] Rubner, Yossi ; Tomasi, Carlo ; Guibas, Leonidas J.: The Earth Mover's Distance as a Metric for Image Retrieval. In: *Int. J. Comput. Vision* 40 (2000), S. 99–121

[SB98] Shaked, Doron ; Bruckstein, Alfred M.: Pruning medial axes. In: *Comput. Vis. Image Underst.* 69 (1998), S. 156–169

[SBH+11] Shen, Wei ; Bai, Xiang ; Hu, Rong ; Wang, Hongyuan ; Latecki, Longin J.: Skeleton growing and pruning with bending potential ratio. In: *Pattern Recognition* 44 (2011), S. 196–209

[Sch06] SCHMITT, Ingo: *Ähnlichkeitssuche in Multimedia-Datenbanken. Retrieval, Suchalgorithmen und Anfragebehandlung.* München, 2006

[SK96] SIDDIQI, K. ; KIMIA, B.B.: A shock grammar for recognition. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 0 (1996), S. 507

[SK05] SEBASTIAN, Thomas B. ; KIMIA, Benjamin B.: Curves vs. skeletons in object recognition. In: *Signal Processing* 85 (2005), Nr. 2, S. 247–263

[SKK04] SEBASTIAN, Thomas B. ; KLEIN, Philip N. ; KIMIA, Benjamin B.: Recognition of Shapes by Editing Their Shock Graphs. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (2004), Nr. 5, S. 550–571

[SLSK07] SHARF, Andrei ; LEWINER, Thomas ; SHAMIR, Ariel ; KOBBELT, Leif: On-the-fly Curve-skeleton Computation for 3D Shapes. In: *Computer Graphics Forum, (Proceedings Eurographics 2007)* 26 (2007), Nr. 3, S. 323–328

[SLV99] SCHOMAKER, Lambert ; LEAU, Edward de ; VUURPIJL, Louis: Using Pen-Based Outlines for Object-Based Annotation and Image-Based Queries. In: *VISUAL*, 1999, S. 585–592

[SP08] SIDDIQI, Kalem ; PIZER, Stephen: *Medial Representations: Mathematics, Algorithms and Applications.* Springer Publishing Company, 2008

[SSGD03] SUNDAR, H. ; SILVER, Deborah ; GAGVANI, Nikhil ; DICKINSON, Sven J.: Skeleton Based Shape Matching and Retrieval. In: *Shape Modeling International*, 2003, S. 130–142, 290

[SSS$^+$06] SCHÜNKE, Michael ; SCHULTE, Erik ; SCHUMACHER, Udo ; LAMPERTI, Edward D. ; ROSS, Lawrance M.: *Thieme Atlas of Anatomy: General Anatomy and Musculoskeletal System.* Thieme Flexible Taschenbücher, 2006

[TH02] TAM, Roger ; HEIDRICH, Wolfgang: Feature-Preserving Medial Axis Noise Removal. In: *Proceedings of the European Conference on Computer Vision (ECCV)*, 2002

[TW02] TELEA, Alexandru ; WIJK, Jarke J.: An augmented Fast Marching Method for computing skeletons and centerlines. In: *Proceedings of the symposium on Data Visualisation 2002*, 2002, S. 251–ff

[Wik11] WIKIPEDIA: *Bipartite graph — Wikipedia, The Free Encyclopedia.* `http://en.wikipedia.org/w/index.php?title=Bipartite_graph&oldid=411125443`. Version: 2011. – [Online; accessed 9-March-2011]

[XTP03] XIE, Wenjie ; THOMPSON, Robert P. ; PERUCCHIO, Renato: A topology-preserving parallel 3D thinning algorithm for extracting the curve skeleton. In: *Pattern Recognition* 36 (2003), Nr. 7, S. 1529–1544

[XWB09] XU, Yao ; WENYU, Liu ; BAI, Xiang: Skeleton Graph Matching Based on Critical Points Using Path Similarity. In: *ACCV (3)*, 2009, S. 456–465

[YBYL07] YANG, Xingwei ; BAI, Xiang ; YU, Deguang ; LATECKI, Longin J.: Shape Classification Based on Skeleton Path Similarity. In: *Energy Minimization Methods in Computer Vision and Pattern Recognition* Bd. 4679. Springer Verlag, 2007, S. 375–386

[You98] YOUNES, Laurent: Computable Elastic Distances between Shapes. In: *SIAM J. Appl. Math* 58 (1998), S. 565–586