Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2003-008
AITR-2003-016

August 8, 2003

Representation and Detection of Shapes in Images

Pedro F. Felzenszwalb

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Representation and Detection of Shapes in Images

by

Pedro F. Felzenszwalb

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2003

© Massachusetts Institute of Technology 2003.
All rights reserved.

Certified by: W. Eric L. Grimson
Bernard Gordon Professor of Medical Engineering
Thesis Supervisor

Accepted by: Arthur C. Smith
Chairman, Department Committee on Graduate Students

Representation and Detection of Shapes in Images

by

Pedro F. Felzenszwalb

Submitted to the Department of Electrical Engineering and Computer Science on July 12, 2003, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

We present a set of techniques that can be used to represent and detect shapes in images. Our methods revolve around a particular shape representation based on the description of objects using triangulated polygons. This representation is similar to the medial axis transform and has important properties from a computational perspective. The first problem we consider is the detection of non-rigid objects in images using deformable models. We present an efficient algorithm to solve this problem in a wide range of situations, and show examples in both natural and medical images. We also consider the problem of learning an accurate non-rigid shape model for a class of objects from examples. We show how to learn good models while constraining them to the form required by the detection algorithm. Finally, we consider the problem of low-level image segmentation and grouping. We describe a stochastic grammar that generates arbitrary triangulated polygons while capturing Gestalt principles of shape regularity. This grammar is used as a prior model over random shapes in a low level algorithm that detects objects in images.

Thesis Supervisor: W. Eric L. Grimson
Title: Bernard Gordon Professor of Medical Engineering

Acknowledgments

I want to thank my advisor, Eric Grimson, for all of his help. Eric always let me do whatever I wanted, and sometimes believed in my ideas more than I did. I also want to thank Dan Huttenlocher, who introduced me to computer vision and has been a great friend since then.

Many people at MIT have helped me during the last few years. I want to thank my thesis committee, Bill Freeman, Leslie Kaelbling and Tomas Lozano-Perez and everyone from the welg-medical group.

I want to thank my friends and family for their support and encouragement. Specially my parents who have always inspired me. I want to thank Carly for making everything so wonderful.

This work was supported in part by NSF-ITR award 0085836 and DARPA contract N00014-00-1-0907.

Contents

1 Introduction **9**
1.1 Shape Representations 13

2 Triangulated Polygons **17**
2.1 Chordal Graphs and k-trees 21
2.2 Shape of Landmark Data 22
2.3 Shape of Triangulated Polygons 23

3 Deformable Template Matching **27**
3.1 Related Work 28
3.2 Matching 28
3.3 Energy Function 30
3.4 Algorithm 30
3.5 Experimental Results 33

4 Learning Deformable Template Models **43**
4.1 Statistical framework 43
4.2 Procrustes Mean Shape 45
4.3 Deformable Templates 46
4.4 Experimental Results 50

5 Random shapes **59**
5.1 Shape Grammar 60
5.2 Sampling Shapes From Images 66
5.3 Experimental Results 70

6 Conclusion **75**
6.1 Future Work 75

Bibliography **77**

Chapter 1

Introduction

The study of shape is a recurring theme in computer vision. For example, shape is one of the main sources of information that can be used for object recognition. In medical image analysis, geometrical models of anatomical structures play an important role in automatic tissue segmentation. The shape of an organ can also be used to diagnose diseases. In a completely different setting, shape plays an important role in the perception of optical illusions (we tend to see particular shapes) and this can be used to explain how our visual system interprets the ambiguous and incomplete information available in an image.

Characterizing the shape of a specific rigid object is not a particularly hard problem, although using the shape information to solve perceptual tasks is not easy. The shape representation problem becomes more difficult when we try to characterize generic object classes such as shoes, cars, elephants and so on. No two objects in a class have the same exact shape, but it seems clear that we can classify objects in such classes using only geometrical information. One of the major challenges in generic object recognition is specifying which are the important geometrical features of a particular object class, and which features are irrelevant.

The shape of an object can be described in many ways and different representations may be appropriate for different tasks. Theoretically two descriptions might be equivalent in the sense that one can be constructed from the other, but in practice each representation will make explicit different properties of objects. From an algorithmic perspective, it is important to consider how a shape representation can be used in a computational theory of vision. For example, we want to be able to quickly find and identify the objects in an image. Many of the theories

that have been developed for shape analysis and interpretation do not lead to practical algorithms. Simulated annealing and other generic optimization methods are often used as a black box by the computer vision community, yet these are inherently slow computations.

One of our main goals is to develop techniques that can be used to represent and detect relatively generic objects. In particular we emphasize the importance of efficient algorithms for image analysis. The techniques we describe in this thesis revolve around a particular shape representation, based on the description of objects using triangulated polygons. As we show in the next chapter, the triangles that decompose a polygon without holes are connected together in a tree structure, and this has important algorithmic consequences. By picking a particular triangulation for the polygons we obtain decompositions of objects into meaningful parts. This yields a discrete representation that is closely related to Blum's medial axis transform [6]. Moreover, the triangulated polygons allow us to describe complex shapes using simple building blocks (the triangles).

We consider two quite different shape detection problems. In the first problem the goal is to find the location of a deformable shape in an image. This problem is important for the recognition of non-rigid objects. In particular, objects in many generic classes can be described as deformed versions of an ideal two-dimensional template. In this setting, the location of a deformable shape is given by a continuous map from a template to an image. Figure 1.1 illustrates how we use a deformable template to detect a particular anatomical structure in an MR image. We will show how triangulated polygons provide rich models for deformable shapes. These models can capture both boundary and interior information of an object and the object can be deformed in an intuitive way. Equally important, we present an efficient algorithm to find the optimal location for a deformable shape in an image. In contrast, previous methods that take into account the interior of deformable templates are too slow for practical use or rely on local search techniques to perform detection. The local search techniques require initialization near the correct answer, while our algorithm finds the optimal location for the object without such information.

Initially we use templates that are constructed from a single picture of an object. In this case the deformation model for the template is relatively generic. We then describe how we can learn a deformable template model for a class of objects by observing multiple instances of the objects in the class. In this case we obtain a deformation model that can be quite specific, capturing which parts of the template are deformable and which parts are mostly rigid. Figure 1.2 illustrates

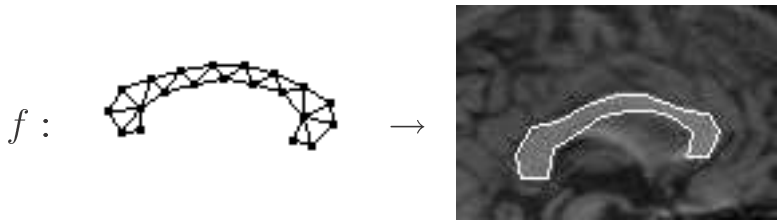


Figure 1.1: The function f maps a template to an image, characterizing the location and shape of a non-rigid object in the image.

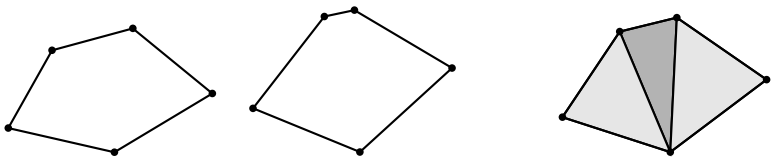


Figure 1.2: Two examples of a deformable object (a and b), and the learned model (c). The color of each triangle in the model indicates how much their shapes vary across different examples (darker triangles vary more).

the learning procedure for a simple object. The learning method is extremely useful for constructing good models of different object classes in an easy and automatic way.

The second main problem we consider falls into the category of low-level segmentation and grouping. The goal of a segmentation algorithm is to identify parts of an image that are likely to correspond to individual objects. This task is different from the shape detection problem described above because the segmentation algorithm is not searching for a specific object, as it does not know in advance which objects are present in the scene. We approach this problem in a new way, providing a system that generates multiple hypotheses for possible objects in the image. The output of our system is not meant to be a final answer to an application, but the first step in solving a perceptual problem. For example, each hypothesis could be matched against a database of known objects to establish their identities.

To find unknown objects in images, a segmentation algorithm must make assumptions about the appearance of typical objects. The Gestalt psychologists identified a number of laws that guide the grouping of to-

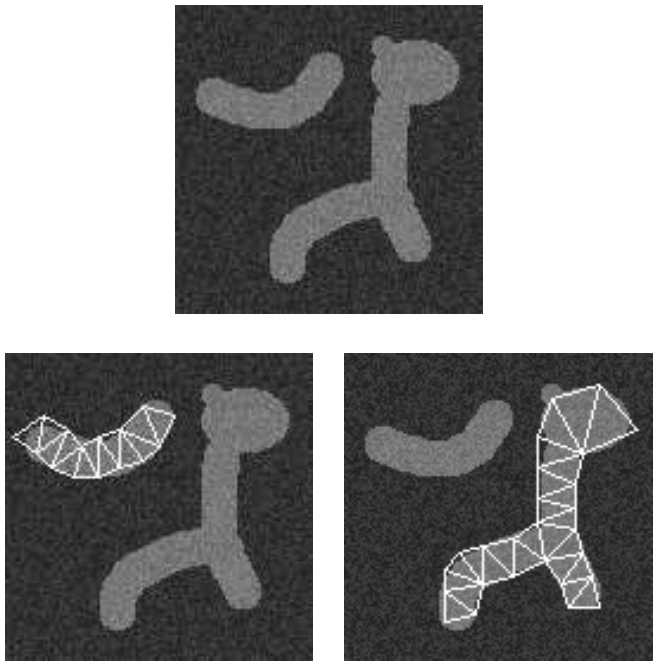


Figure 1.3: An input image and two different hypothesis for the shapes in the image as generated by the low-level detection algorithm.

kens by the human visual system. For example, some of the most important grouping cues are that object boundaries tend to be continuous and smooth almost everywhere. We will describe a stochastic grammar that generates triangulated polygons and captures these and other grouping cues including contour closure and self similarities. The grammar defines a distribution over random shapes where shapes that look natural have higher probability of being created. Together with a model of how images are generated we obtain a posterior distribution over random shapes in an image. By sampling shapes from this distribution we obtain hypotheses for the objects in a scene. Each hypothesis looks natural and their boundaries align with places where the image has high gradient magnitude. Figure 1.3 illustrates some of the hypotheses generated for a simple image.

In the next section we review some of the shape representations that have been previously used in computer vision. In Chapter 2 we describe how we can represent objects using triangulated polygons and study a number of properties of this representation. In Chapter 3 we show

how triangulated polygons can be used to model deformable shapes and how these models can be used to efficiently detect the location of non-rigid objects in images. In Chapter 4 we show how we can learn deformable template models from multiple examples of objects in a class. In Chapter 5 we describe the shape grammar that generates triangulated polygons and its application in the shape detection system that performs low-level segmentation. In the last chapter we summarize our results and describe some ideas for future work.

1.1 Shape Representations

The geometric properties of any specific rigid object are relatively well understood. We know how three-dimensional features such as corners or edges project into images, and there are a number of methods that can be used to represent rigid shapes and locate their projections. Some techniques, such as the alignment method [23], use explicit three-dimensional representations. Other techniques, such as linear combination of views [40], capture the appearance of three-dimensional shapes using a small number of two-dimensional pictures. These and similar techniques assume that all shape variation comes from the viewpoint dependency of two-dimensional images.

A number of representations describe objects in terms of a small set of generic parts. This includes representations based on generalized cylinders [29] and geons [5]. These approaches are appealing because they provide symbolic descriptions of objects, and make semantic information explicit. A shortcoming is that some objects do not have a clear decomposition into generic parts. For example, what are the parts that make up a shoe? Another problem is that we do not know how to extract generic parts from images in a robust way. On the other hand, models known as pictorial structures [15, 14] have been successfully used to characterize and detect objects that are described by a small number of rigid parts connected in a deformable configuration. In the pictorial structure approach, generic parts are not extracted from images on their own, the whole object model is used at once. Our shape representation is similar to the pictorial structure models in that objects are represented by a number parts connected together. When matching a triangulated polygon model to an image we take the same approach as the pictorial structures techniques, considering the whole model at once instead of trying to detect the generic parts individually.

We can represent objects in terms of their boundaries, which for two-dimensional objects are curves, and for three-dimensional objects

are surfaces. Such representations are commonly used both in the context of image segmentation and object recognition. One example is a popular technique for image segmentation known as active contour models or snakes [26]. Boundary models are also used for generic object recognition. For example, the work in [2] describes how we can measure similarity between objects in terms of the amount of stretching and bending necessary to turn the boundary of one object into the boundary of another one. Similarly, in [19] deformable boundary models were used to represent generic objects. In this case the models were used to detect instances of the generic class in noisy images. One problem with deformable boundary models is that they do not capture well how the interior of objects deforms.

Blum introduced a representation known as the medial axis transform [6] that is now widely used. The medial axis of an object is defined as the set of centers of all maximally inscribed disks (disks that are contained inside the object but not contained in any other such disk). The medial axis transform is the medial axis together with the radius of each maximal disk. For two-dimensional objects the medial axis is one-dimensional and if the shape has no holes the medial axis has no loops. The tree structure is appealing from a computational point of view. The medial axis captures local symmetries of an object and provides a natural decomposition of the object into parts (corresponding to branches in the one-dimensional structure). A closely related representation known as the shock graph [37] makes even more information explicit. In general, representations based on the medial axis seem well suited to capture the geometry of generic objects. A model of how the shock graph deforms and changes structure was presented in [34]. As shown in [33], medial axis models can better capture natural deformations of objects when compared to boundary models. Another example of generic object recognition using a representation related to the medial axis transform is described in [43]. Our triangulated polygon representation is closely related to the medial axis transform. In particular our models capture local symmetries and provide natural decompositions of shapes into parts.

Statistical shape theory [38, 12] can be used to study objects that are defined by a set of labeled landmarks, where each landmark marks the locations of an important object feature. In this scenario, the space of possible object shapes can be described using differential geometry, leading to a natural notion of distance between shapes. Moreover, with this approach we can characterize generic object classes using probability distributions in shape space. In computer vision these techniques became popular with the active shape models [7]. While active shape

models can capture the typical variation in shape among a class of objects fairly well, the problem of detecting objects in images using these models is hard and so far we need to resort to local search methods. These methods perform well as long as a good initial guess for the location of the target object is available, but they do not tend to work without such information. As demonstrated in [1] the object detection problem can be solved efficiently if we constrain the distribution over shapes to be of a particular form, in terms of decomposable graphs. This is a promising approach and is closely related to how we represent deformable shapes. As shown in the next chapter, a triangulation of a polygon yields a natural decomposable graph.

Chapter 2

Triangulated Polygons

In this chapter we describe our representation of objects using triangulated polygons. Intuitively, a polygonal boundary is used to approximate the actual boundary of an object, and a triangulation provides a decomposition of the object into parts. Some examples are shown in Figure 2.1. As we will see below, this representation has many nice properties. It captures perceptually important features of an object and its structure can be exploited by efficient computational mechanisms.

We assume that objects are connected subsets of $\mathbb{R}^2$ whose boundaries are smooth except at a finite number of points. Also, we only consider objects without holes (their boundaries are simple closed curves). In this case an object can be approximated to any desired precision by a *simple polygon*, which is a polygon without holes. A triangulation of a polygon P is a decomposition of P into triangles, defined by *diagonals*.

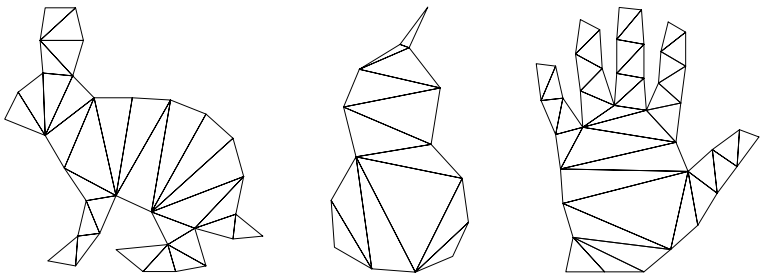


Figure 2.1: Rabbit, pear, and hand represented by triangulated polygons. The polygonal boundaries represent the outlines, while the triangulations decompose the objects into parts.

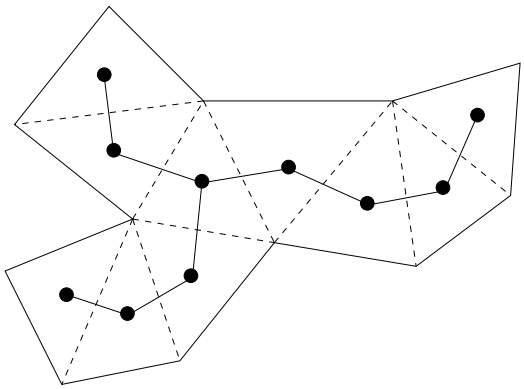


Figure 2.2: A triangulated polygon and its dual graph. If the polygon is simple the dual graph is a tree.

Each diagonal is a line segment that connects two vertices of P and lies in the interior of the polygon. Moreover, no two diagonals cross. We know (see [11]) that every simple polygon can be triangulated, and any triangulation of a polygon with n vertices consists of exactly $n-2$ triangles.

There is a natural graph structure associated with a triangulated polygon. The nodes of the graph are the polygon vertices, and the edges include both the polygon boundary and the diagonals in the triangulation. Any triangulation of a simple polygon has an important property which will play a key role in the rest of our work. First, if T is a triangulated polygon we denote its dual graph by G_T. The nodes of G_T correspond to the triangles in T, and two nodes are connected when the corresponding triangles share an edge. Figure 2.2 illustrates a triangulated polygon and its dual graph.

Proposition 2.1. *If T is a triangulation of a simple polygon, then its dual graph G_T is always a tree.*

Proof. To see this, just note that each diagonal in T cuts the polygon into two disjoint parts, so removing an edge from G_T disconnects the graph. The dual graph would have cycles if the polygon had holes, and this is why we consider only simple polygons. □

Note that every tree has a leaf, and a leaf in G_T corresponds to a triangle with some polygon vertex v that is not in any other triangle. If we delete v and its triangle from T we obtain a new triangulated polygon. Repeating this procedure we get an order of elimination for

the vertices and triangles of T. The order is such that when eliminating the i-th vertex, it is in exactly one triangle of the current triangulated polygon. If we consider the graph structure defined by T, this ordering is a *perfect elimination scheme* for the vertices of the graph. Graphs which admit perfect elimination schemes form an important class which we discuss in Section 2.1.

In principle there are many possible ways to triangulate a polygon but if we use a particular class of triangulations known as constrained Delaunay triangulations (CDT) we obtain a representation that is closely related to the medial axis transform. A good introduction to the Delaunay triangulation can be found in [3]. To define the CDT we need to introduce the notion of *visibility*. Given a polygon P, we say that a point a is visible to a point b if the line segment ab lies entirely inside the polygon. Similarly, a is visible to the line segment bc if it is visible to some point on bc.

Definition 2.1. *For a polygon P, the* constrained Delaunay graph *contains the edge ab if and only if a is visible to b and there is a circle through a and b that contains no vertex c visible to ab.*

If no four vertices are collinear then this definition yields the (unique) CDT of the polygon. Otherwise we call any triangulation obtained by adding diagonals to the constrained Delaunay graph a CDT. A constrained Delaunay triangulation can be computed efficiently and yields a decomposition of the polygon into meaningful parts. We can see the relationship between the CDT and the medial axis transform by considering what happens as the distance between neighboring polygon vertices decreases. In the limit the circles that define the Delaunay graph are inscribed in the polygon, and correspond to the disks that define the medial axis.

By looking at Figure 2.1 we can see how the diagonals in a CDT decompose the objects into natural parts. For example, there is a diagonal separating each finger from the rest of the hand and a diagonal separating each leg of the rabbit from the body. It is known (see [22, 36]) that a natural way to decompose objects into parts is to split them at places where the boundary has curvature minima. This is because joining two parts together usually creates such minima. Figure 2.3 illustrates how the CDT always includes diagonals that split a limb from the rest of an object and diagonals that cut objects at points where they are "pinched". In both cases the diagonals connect pairs of boundary points with locally minimal curvature.

There are three possible types of triangles in a triangulated polygon, corresponding to nodes of different degrees in the dual graph. The

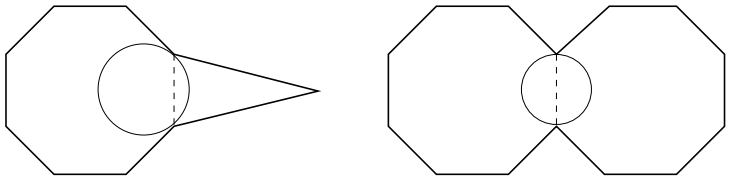


Figure 2.3: Limbs and pinch points are naturally represented in the CDT. These pictures show the defining circle for a diagonal that corresponds to a limb boundary and a diagonal that corresponds to a pinch point.

Figure 2.4: Different triangle types. The first triangle type corresponds to ends of branches, the second type corresponds to branches and necks while the last corresponds to connections between multiple branches and necks.

three types are shown in Figure 2.4, where solid edges are part of the polygon boundary, and dotted edges are diagonals in the triangulation. Sequences of triangles of the second type form branches (or necks) of a shape. Triangles of the first type correspond to ends of branches, and triangles of the third type connect multiple branches together. We can see how in Figure 2.1 each finger in the hand and both rabbit legs are formed by sequences of triangles of the second type and end with a triangle of the first type.

Counting the number of triangles of each type in a triangulated polygon gives very coarse geometric information that may be useful for generic object recognition and classification. For example, the number of junction triangles gives a measure of the "branchiness" of an object. Some care must be taken because different triangulations of the same polygon will have different structure. If the polygon vertices are in general position then the CDT is unique and we can count the number of triangles of each type in this triangulation. Another source of coarse geometric information is to consider the structure of the dual graph of a triangulated polygon. Blum studied such information in the context

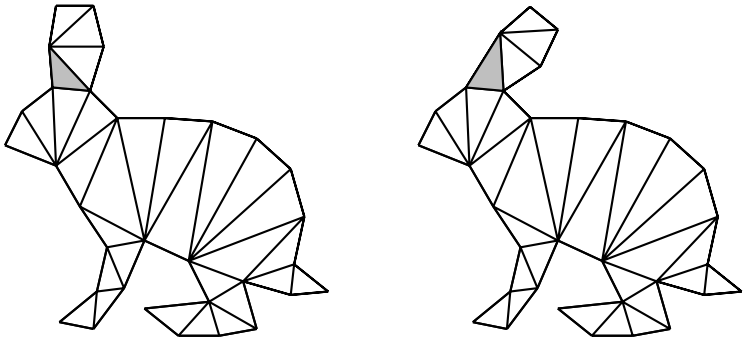


Figure 2.5: The rabbit ear can be bent by changing the shape of a single triangle (painted gray).

of the medial axis transform, and recently the combinatorial structure of the shock graph has also been used for generic object recognition.

A nice property of triangulated polygons is that the triangulations give a simple and intuitive way to non-rigidly deform the objects. For example, the rabbit's ear in Figure 2.5 can be bent by changing the shape of a single triangle. In the next chapter we will use this idea to detect non-rigid objects in images. The exact relationship between the shape of a triangulated polygon and the shape of each of its triangles will be clarified in Section 2.3.

2.1 Chordal Graphs and k-trees

Chordal graphs and k-trees play an important role in our work, so we review some of their properties here (see [16] and [10] for more details).

Definition 2.2. *A graph is* chordal *if every cycle of length at least 4 has a chord (a chord is an edge between non-consecutive vertices of the cycle).*

Chordal graphs are also known as *triangulated* or *decomposable* graphs. These graphs are important because many problems that are hard to solve for arbitrary graphs can be efficiently solved in this restricted class. Most algorithms for chordal graphs rely on the following characterization. Recall that a *clique* in a graph is a set of nodes where there is an edge between each pair of them. If G is a graph and S is a subset of its nodes, then the graph *induced* by S consists of S itself and all edges from G that connect pairs of nodes in S. A vertex of a

graph is called *simplicial* if its neighbors form a clique. Every chordal graph has a simplicial vertex, and actually something stronger is true.

Definition 2.3. *Let G be a graph and $\sigma = (v_1, \ldots, v_n)$ be an ordering of its vertices. The ordering is a* perfect elimination scheme *if each v_i is a simplicial vertex of the subgraph induced by $\{v_i, \ldots, v_n\}$.*

It turns out that G is chordal if and only if G admits a perfect elimination scheme. Moreover, there are efficient algorithms to find a perfect elimination scheme for a chordal graph. We have already shown that the graph structure of a triangulated simple polygon admits a perfect elimination scheme. In the next chapter we will use this fact to detect deformable objects in images using dynamic programming. Triangulations of simple polygons actually belong to a well-known subclass of chordal graphs.

Definition 2.4. *A clique on $k+1$ vertices is a k-tree. Given any k-tree T on n vertices, we can construct a k-tree on $n+1$ vertices by adding a new vertex to T which is made adjacent to each vertex of some k-clique.*

Every maximal clique in a k-tree has size $k+1$ and a k-tree can be thought of as a set of k-dimensional simplices connected along $(k-1)$-dimensional faces. With this interpretation k-trees are acyclic simplicial complexes. The class of graphs corresponding to triangulations of simple polygons are called *planar 2-trees*. These are exactly the 2-trees where at most two triangles meet at any particular edge.

2.2 Shape of Landmark Data

In this section we review some basic concepts from statistical shape theory. First we define what we mean by the shape of an object that lives in an Euclidean space. The following definition is from [12]:

Definition 2.5. Shape *is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.*

This makes the shape of an object invariant to Euclidean similarity transformations. In particular, two objects have the same shape if one can be translated, scaled and rotated to exactly match the other one. Figure 2.6 illustrates different objects that have the same shape.

To describe shape we consider the location of a finite number of points on each object. Points that mark the location of important object features are called *landmarks*. For example, to describe the shape of a maple leaf we might choose landmarks as illustrated in Figure 2.7.

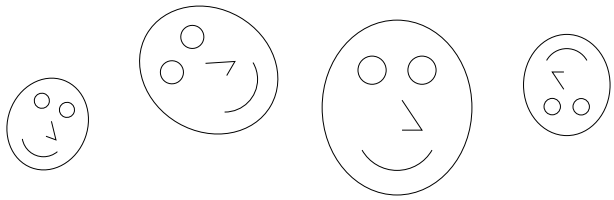


Figure 2.6: Different objects with the same shape.

We will always assume that landmarks are labeled so that each one corresponds to a particular object feature. In the case of a triangulated polygon we can take the vertices as the landmarks. Note that a polygon is fully determined by the location of its vertices. In general the set of landmarks on an object provides a partial description of the object.

Definition 2.6. *A* configuration *is the set of landmarks on a particular object. The configuration of an object with k landmarks in m dimensions is given by a $k \times m$ matrix X, where the i-th row of X gives the coordinates of the i-th landmark.*

The configuration space is the set of all possible configurations for an object and it usually equals $\mathbb{R}^{km}$ minus some possible singularities (we may want to exclude configurations where landmarks coincide). Let $\sim$ denote the equivalence relation defined by $X \sim Y$ when X is related to Y by a similarity transformation. The space of possible shapes is the set of equivalence classes defined by $\sim$. We denote the shape of X by $[X]$ and we say that a shape is *degenerate* if it is the shape of a configuration with coinciding landmarks.

2.3 Shape of Triangulated Polygons

Suppose we have an object with n labeled landmarks in $\mathbb{R}^2$. The object is described by a $n \times 2$ configuration matrix X, and its shape is given by $[X]$. We will show that if we eliminate some singularities, $[X]$ is determined by the shapes of the triangles in a 2-tree over the landmarks, and each triangle can have an arbitrary non-degenerate shape. This means that by fixing a 2-tree over the landmarks we obtain a decomposition of the object into parts (the triangles) with shapes that are independent of each other. In particular, a triangulation of a polygon defines a 2-tree, so we can represent the shape of a triangulated polygon by

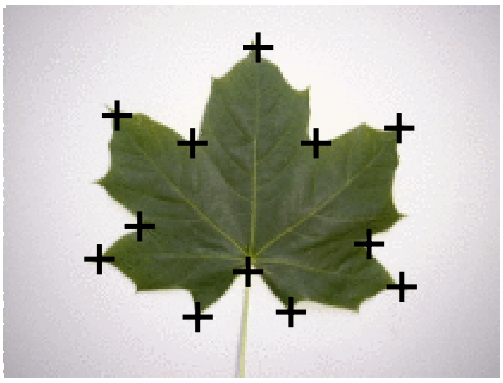


Figure 2.7: The red crosses indicate landmarks marking some of the important features of a maple leaf.

describing the graph structure of the triangulation and specifying the shape of each triangle.

Let T be a 2-tree with vertices $(v_1, \ldots, v_n)$. The 2-tree defines a set of triangles (the 3 cliques in the graph), and we will use $(v_i, v_j, v_k) \in T$ to denote that three particular vertices form a triangle in T. We denote by X_i the i-th row of X. Similarly X_{ijk} is the sub-matrix obtained by selecting rows i, j and k of X. Consider configurations X for n landmarks in $\mathbb{R}^2$ where X_i, X_j and X_k are all different for each triangle $(v_i, v_j, v_k) \in T$. Clearly, X defines a non-degenerate shape $[X_{ijk}]$ for each triangle $(v_i, v_j, v_k) \in T$. In fact, $[X]$ defines the shape of each triangle because $X \sim Y$ implies $X_{ijk} \sim Y_{ijk}$. The following theorem shows a converse to this statement.

Theorem 2.1. *For a fixed 2-tree T, and non-degenerate shapes $s(i, j, k)$ for each triangle of T, there is a unique shape $[X]$ such that $[X_{ijk}] = s(i, j, k)$.*

Proof. If $n = 3$ we only have one triangle and the result is trivial. Now suppose $n > 3$. Let v_i be a simplicial vertex of T. We know v_i is in exactly one triangle, say with v_j and v_k. Let T' be the 2-tree obtained by deleting v_i from T, and X' the matrix X without the i-th row. By induction we can assume $[X']$ is defined by the shapes of the triangles in T'. For fixed v_j and v_k, each position of v_i gives a different shape for the triangle (i, j, k). Moreover, by varying v_i we can obtain any triangle shape. So X is defined by X' and the shape $s(i, j, k)$. □

This result is important because it allows us to describe the shape space for a triangulated polygon in terms of shape spaces for triangles. Let M be a space of triangles. There are two canonical choices for M, either Kendall's or Bookstein's space of triangles [38]. A triangulated polygon on n vertices has $n-2$ triangles and we can take its shape space to be $M_1 \times M_2 \times \cdots \times M_{n-2}$. The metric structure of M induces a canonical metric on the cross product space. Figure 2.5 shows how an object can be deformed by changing the shape of a single triangle. The figure on the left is a point in shape space $(x_1, x_2, \ldots, x_{n-2})$, where $x_i \in M_i$. The figure on the right has the same coordinates except for one of the x_j that changes.

Chapter 3

Deformable Template Matching

In this chapter we address the problem of detecting non-rigid objects in images. Our approach falls within the framework of deformable template matching, where one wants to find a non-rigid transformation that maps a model to the image. Figure 3.1 illustrates the situation, where we have a template and a non-rigid map that indicates how the template is deformed to align with the target object in an image.

An energy function associates a cost with each potential transformation of the model, and we want to find a transformation with the lowest possible cost. Typically the energy function is a sum of two terms: the data term attracts the deformed model toward salient image features, while another term penalizes large deformations of the model. Most of the existing non-rigid matching techniques require initialization near the final solution or are too slow for practical use. This is because the

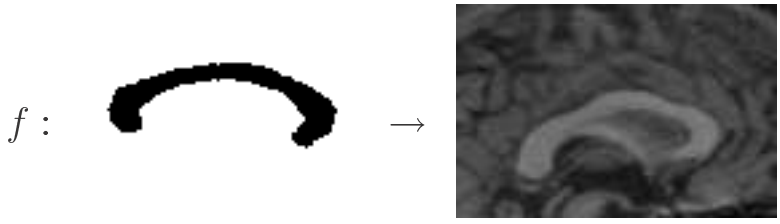


Figure 3.1: The function f maps a template to an image, characterizing the location and shape of a non-rigid object.

number of possible transformations of a template is usually very large. In contrast, we present an algorithm that can quickly find a global optimal non-rigid transformation without any initialization. The search over transformations is done efficiently by exploiting special properties of triangulated polygons and their deformations.

We consider energy functions that can be written as a sum of terms, one for each triangle in a triangulated polygon template. This type of energy function is quite general, and can be used to represent a wide range of deformable template models, including models that represent the boundary and the internal structure of a two-dimensional non-rigid object. Even when we use an energy function with a data term that depends only on the boundary of a shape we take into account region information when measuring shape deformation. In this way we obtain more realistic models of deformation than are possible using only boundary models.

Our experimental results illustrate the robustness of our method, showing accurate detection of non-rigid objects even in highly cluttered scenes. We show results both on medical and natural images, demonstrating the wide applicability of these techniques.

3.1 Related Work

The basic idea of matching a deformable model to an image goes back to Fischler and Elschlager [15] and Widrow [41]. More recently, Grenander [18] introduced a framework that provides a very general setting to represent deformable objects. Other influential models were presented in [25] and [7]. A few efficient and provably good matching algorithms have been developed for restricted sets of deformable models. For example, in [9] a dynamic programming algorithm was used to detect open deformable contours in images. Dynamic programming was also used in [1] to match models consisting of a number of landmarks with positions constrained by a decomposable graphical model. Efficient algorithms also exist for the related problem of computing a non-rigid match between two pre-segmented objects (such as [2] and [34]).

3.2 Matching

Let P be a simple polygon corresponding to an object template. An embedding of P in the plane is defined by a continuous function $f : P \rightarrow \mathbb{R}^2$, where f is defined over both the boundary and interior of the polygon. We consider a set of embeddings that are extensions of maps $g : V \rightarrow \mathbb{R}^2$,

where V are the vertices of P. Let T be a triangulation of P. The triangulation gives a natural extension of g to all of the polygon as a piecewise affine map f. The function f sends each triangle $(v_1, v_2, v_3) \in T$ to the triangle $(g(v_1), g(v_2), g(v_3))$ using linear interpolation. In this way, f restricted to each triangle $t \in T$ is an affine map f_t. To see that f is well defined (and continuous) just note that if two triangles $a, b \in T$ touch, then f_a and f_b agree along the intersection of a and b. What may seem surprising is that all embeddings which map each triangle according to an affine transformation are extensions of some g. This follows from the fact that an affine transformation is defined by the image of three non-collinear points.

We define an energy function that assigns a cost to each map g, relative to an image I. The matching problem is to find g with minimum energy (corresponding to the best location for the deformable template in the image). Consider energy functions with the following structural form:

$$E(g, I) = \sum_{(v_i, v_j, v_k) \in T} c_{ijk}(g(v_i), g(v_j), g(v_k), I). \tag{3.1}$$

Each term c_{ijk} should take into account the shape of the embedded triangle and the image data covered by the embedding. For the experiments in this chapter we use a simple energy function similar to typical deformable template matching costs. For each triangle t, a deformation cost measures how far the corresponding affine map f_t is from a similarity transformation. This makes our models invariant to translations, rotations and uniform scalings, which is important for detecting objects from arbitrary viewpoints. A data cost attracts the boundary of the embedded polygon to image locations with high gradient magnitude. In particular, we expect the target object to have different intensity or color from the background, and the intensity gradient should be roughly perpendicular to the object boundary. More details are given in the next section.

While the implementation described here uses a fairly simple energy function, the formulation can handle richer concepts. For example, the deformation costs could be tuned for individual triangles, taking into account that different parts of the shape may be more flexible than others. This involves selecting different costs c_{ijk} for each triangle in T. In fact, in the next chapter we describe a method that learns deformation parameters from training data. Also, the data costs could take into account the whole area covered by the embedded polygon. For example, if we have a grayscale or color texture map associated with a model we can use the correlation between the deformed texture map and the image to obtain a data cost.

3.3 Energy Function

In our framework, each triangle in a template is mapped to the image plane by an affine transformation. In matrix form, we can write the affine transformation as $h(x) = Ax + a$. We restrict our attention to transformations which preserve orientation ($\det(A) > 0$). This ensures that the corresponding embedding f is locally one-to-one. Let α and β be the singular values of A. The transformation h takes a unit circle to an ellipse with major and minor axes of length α and β. The value $\log(\alpha/\beta)$ is called the *log-anisotropy* of h and is commonly used as a measure of how far h is from a similarity transform (see [38]). We use the log-anisotropy measure to assign a deformation cost for each affine map (and let the cost be infinity if the affine map is not orientation preserving). The deformation costs are combined with a data cost that attracts the template boundary to locations in the image that have high gradient magnitude,

$$E(g, I) = \sum_{t \in T} \mathrm{def}(f_t)^2 - \lambda \int_{\partial P} \frac{\|(\nabla I \circ f)(s) \times f'(s)\|}{\|f'(s)\|}\, ds,$$

where $\mathrm{def}(f_t)$ is the log-anisotropy of f_t. The term $\|(\nabla I \circ f)(s) \times f'(s)\|$ is the component of the image gradient that is perpendicular to the shape boundary at $f(s)$. We divide the gradient term in the integral by $\|f'(s)\|$ to make the energy scale invariant. Note that the integral can be broken up into an integral for each edge in the polygon. This allows us to write the energy function in the form of equation (3.1), where the cost for each triangle will be the deformation cost plus one integral term for each boundary edge that belongs to the triangle.

3.4 Algorithm

The matching problem is to find a map $g : V \to \mathbb{R}^2$ with lowest possible energy. The only approximation we make is to consider a finite set of possible locations for each polygon vertex. Let $\mathcal{G} \subset \mathbb{R}^2$ be a grid of locations in the image. For example, each location in the grid could correspond to an image pixel. Normally we use a coarser grid, with about 50×50 locations independent of the image size. In the discrete setting g maps each vertex v_i to a location $l_i \in \mathcal{G}$. For a polygon with n vertices, the number of different such maps is $|\mathcal{G}|^n$. The form of the energy function in equation (3.1) is quite general, and we depend on the structure of T to be able to find an optimal g efficiently. Our

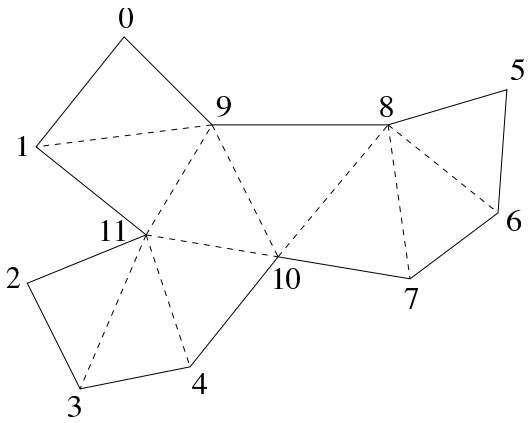


Figure 3.2: Perfect elimination scheme for the vertices of a triangulated simple polygon.

matching algorithm finds an optimal map in time $O(n|\mathcal{G}|^3)$, which is exponentially better than just trying all possible maps.

As described in Chapter 2, there is a nice order of elimination for the vertices and triangles of a triangulated simple polygon (a perfect elimination scheme). The order is such that when eliminating the i-th vertex, it is in exactly one triangle of the current triangulated polygon. Figure 3.2 shows a triangulated polygon with vertices labeled by their order in a perfect elimination scheme. A perfect elimination order can be computed in time linear in the number of polygon vertices using one of the algorithms in [16].

The matching algorithm works by sequentially eliminating the vertices of T using the perfect elimination scheme. This is an instance of a well known dynamic programming technique (see [4]). After eliminating $v_1, \ldots, v_{i-1}$, vertex v_i is in exactly one triangle, say with nodes v_j and v_k. The two nodes v_j and v_k are the parents of v_i, which we indicate by letting $p[i].a = j$ and $p[i].b = k$. We compute the cost of the best placement for v_i as a function of the locations for v_j and v_k. This cost is stored in $V[j,k](l_j, l_k)$. When we get to the last two vertices we can solve for their best location and trace back to find the best location of the other vertices, as is typical in dynamic programming.

Algorithm *Match*(I)
($*$ Find the best embedding of a shape in an image $*$)
1. **for** $i = 1$ **to** $n - 2$
2. ($*$ Eliminate the i-th vertex $*$)
3. $j \leftarrow p[i].a$
4. $k \leftarrow p[i].b$
5. **for** each pair of locations l_j and l_k in $\mathcal{G}$
6. $V[j,k](l_j, l_k) \leftarrow \min_{l_i \in \mathcal{G}} c_{ijk}(l_i, l_j, l_k, I) + V[i,j](l_i, l_j) + V[i,k](l_i, l_k)$
7. Pick l_{n-1} and l_n minimizing $V[n-1, n]$ and trace back to obtain the other optimal locations.

This algorithm runs in $O(nm^3)$ time and uses $O(nm^2)$ space, where n is the number of vertices in the polygon and m is the number of possible locations for each vertex. In practice we can speed up the algorithm by noting that given positions l_j and l_k for the parents of the i-th vertex there is a unique similarity transformation taking v_j and v_k to the respective locations. This similarity transformation defines an ideal location for v_i. We only need to consider locations for v_i that are near this ideal location, because locations that are far introduce too much deformation in the model. With this optimization the running time of the algorithm is essentially $O(nm^2)$.

Note that in line 7 of the matching algorithm each entry in $V[n-1, n]$ corresponds to the quality of an optimal embedding for the deformable shape given particular locations for v_{n-1} and v_n. We can detect multiple instances of a shape in a single image by finding peaks in $V[n-1, n]$. To generate multiple detections we can trace back from each peak that has a value above some given fixed threshold.

3.5 Experimental Results

We present experimental results of our matching algorithm on both medical and natural images. In each case we used a binary picture of the target object to build a triangulated polygon template. From the binary picture we computed a polygonal approximation of the object and then we computed the CDT of the resulting polygon. For the matching results shown here we used a grid of 60×60 possible locations in the image for the vertices of the polygon. The matching algorithm took approximately five minutes in each image when running on a standard 1Ghz Pentium III machine.

Corpus callosum: Figure 3.3 shows a model for the corpus callosum generated from a manually segmented brain MRI. The best match of the model to several new images is shown in Figure 3.4. Note how these images have very low contrast, and the shape of the corpus callosum varies considerably across them. We are able to reliably locate the boundary of the corpus callosum in each case. The quality of our results is similar to the quality of results obtained using the best available methods for model based segmentation of medical images (such as [28]). The main advantage of our method is that it does not require any initialization.

Maple leaves: Figure 3.5 shows a model for maple leaves, constructed from a binary silhouette. The best match of the model to a few images is shown in Figure 3.6. The leaves in each image are different, and the viewing direction varies. Note how our method can handle the variation in shape even in the presence of occlusion and clutter. In particular, the last image shows how we can "hallucinate" the location of a large occluded part of the leaf. Techniques that rely on local search to find non-rigid transformations tend to fail on cluttered images because they get stuck on local optimum solutions as we demonstrate below. Since our models are invariant to similarity transformations we can detect the target object independent of of its position, scale and orientation. Figure 3.7 demonstrates detections at very different scales.

To check the performance of our algorithm on inputs with low signal to noise ratio we corrupted one of the leaf images with random Gaussian noise. Figure 3.8 shows the corrupted image with increasing amounts of noise (corresponding to $\sigma = 50$, 150 and 250) and the matching results for each input. We can identify the approximate location of the leaf even when it is barely visible. In Figure 3.9 we demonstrate how our matching algorithm can be used to detect multiple instances of an

Figure 3.3: A model for the corpus callosum generated from a binary picture. The picture comes from a manually segmented MRI.

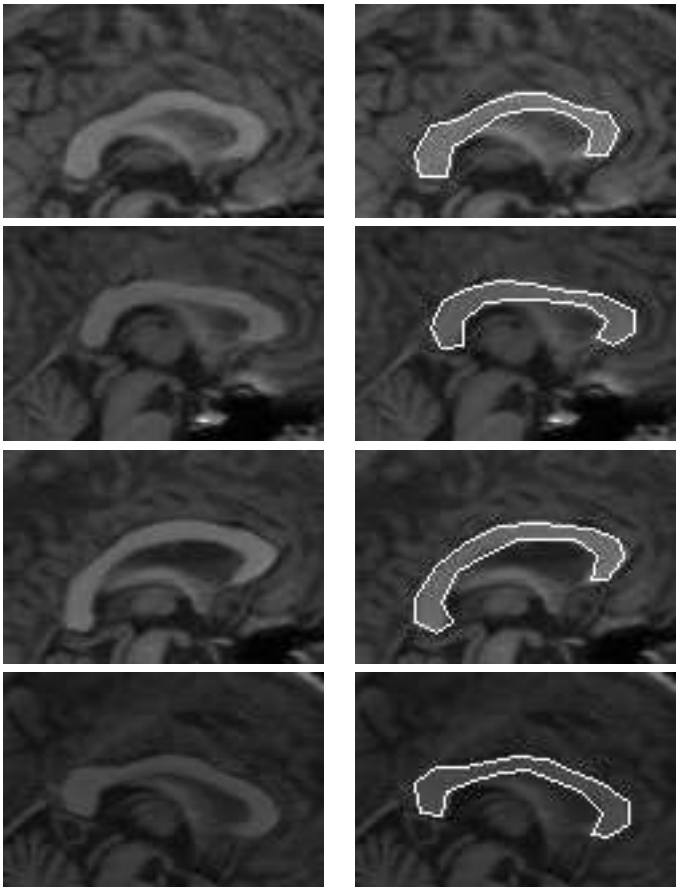


Figure 3.4: Matching the corpus callosum model to different images.

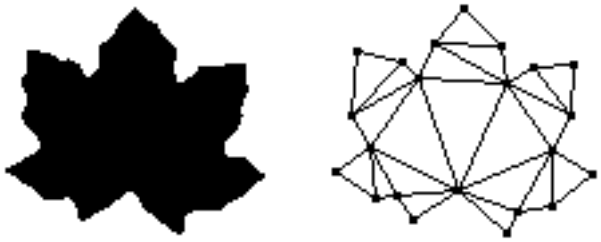


Figure 3.5: A model for maple leaves generated from a binary picture.

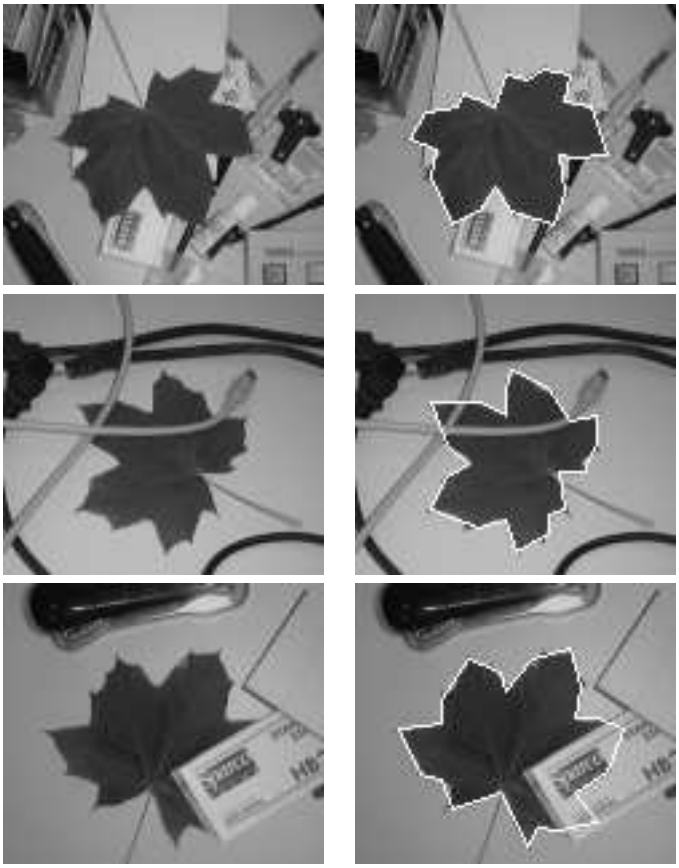


Figure 3.6: Matching the leaf model to different images.

object in an image. As discussed in the last section we simply selected peaks in $V[n-1,n]$ with value above a pre-determined threshold to generate each detection.

We can see that our matching algorithm performs well in hard situations. In contrast, local search methods tend to fail in such cases. To illustrate this we obtained a public implementation of a popular technique known as active appearance models [39]. Every local search technique depends heavily on initialization, so we show results of matching using different initialization parameters on a fixed image. Figure 3.10 shows some cases where the local search method worked well, while Figure 3.11 shows cases where the method did not work. It is clear that good initialization parameters are vital for obtaining reliable results with a local search technique. These experiments illustrate the advantage of global methods.

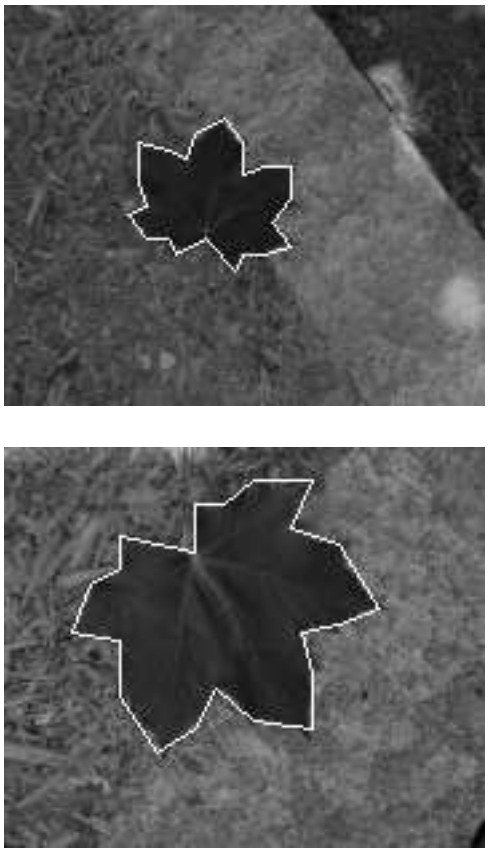


Figure 3.7: Our models are invariant to similarity transformations so we can detect deformable objects at arbitrary scale, position and orientation.

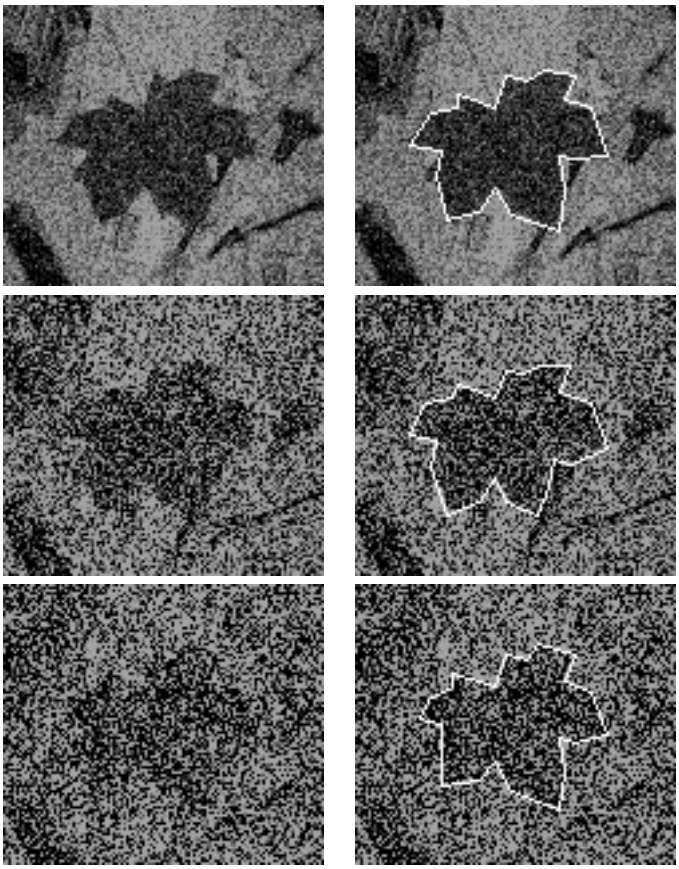


Figure 3.8: Matching in an image corrupted by increasing amounts of Gaussian noise.

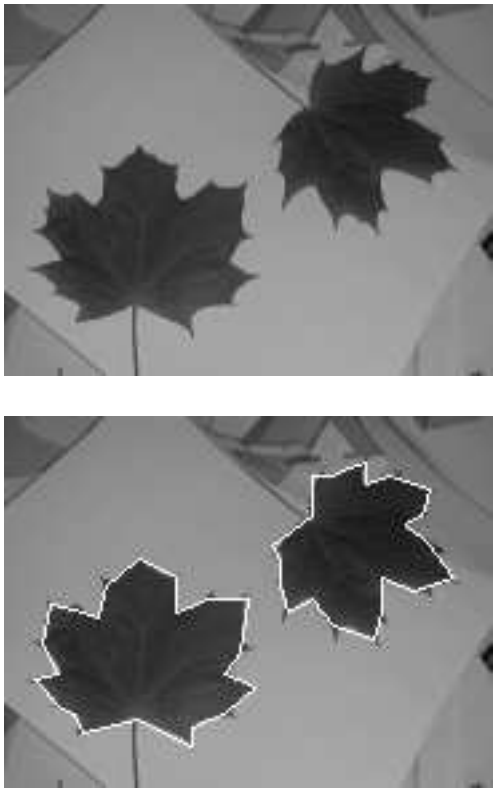
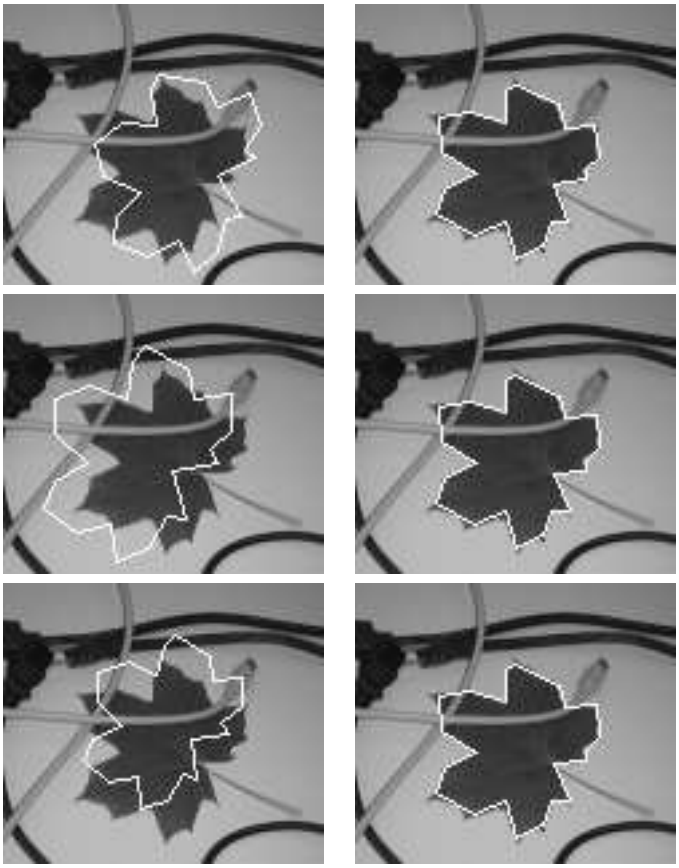
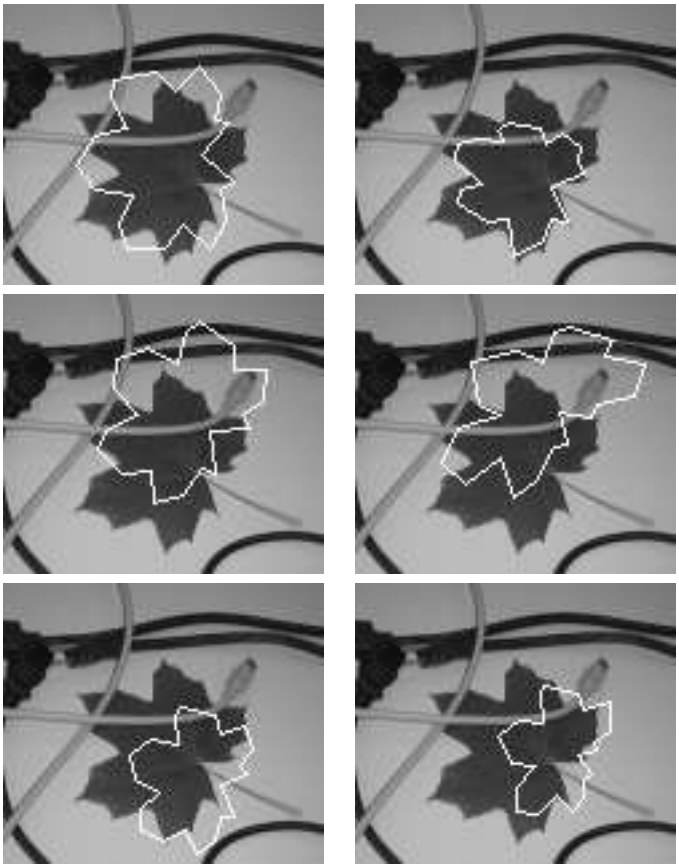


Figure 3.9: Detection of multiple leaves in one image.



Initial pose Final match

Figure 3.10: Cases where a local search method worked well. Each row corresponds to an experiment, the first image gives the initialization parameters and the second image shows the matching results.



Initial pose Final match

Figure 3.11: Cases where the local search method did not work. When the initialization parameters are far from the correct solution the method can give arbitrary results.

Chapter 4

Learning Deformable Template Models

In this chapter we describe how we can learn deformable shape models from examples. Intuitively the learning problem is the following. We are given a number of examples for the shape of an object, each of which is a polygon on a fixed number of vertices. Moreover, the vertices of each example are in correspondence with each other. We want to find a triangulated model that can be easily deformed into each of the examples. Each triangle in the triangulated model should have an ideal shape and a parameter that controls how much it is allowed to deform. Figure 4.1 illustrates the learning procedure. Each triangle in the model is shown in its ideal shape, and the triangles are color coded, indicating how much they are allowed to deform.

There are other techniques such as active shape models [7] that can capture the typical shape variation for a class of objects. Our models are unique in that they can capture objects that deform by large amounts. Moreover, we concentrate on models that can be used by the efficient matching algorithm described in the last chapter.

4.1 Statistical framework

We want to model the shape of a polygon on n vertices. Each instance of the polygon is given by a $n \times 2$ configuration matrix X where the i-th row gives the location of the i-th polygon vertex. The deformable object detection problem can be posed in a statistical framework in the following way. Given an image I, we want to find the location for the

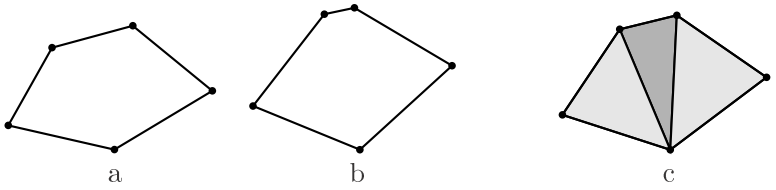


Figure 4.1: Two examples of a deformable object (a and b), and the learned model (c). The color of each triangle in the model indicates how much their shapes vary across different examples (darker triangles vary more).

object that has highest probability of being its true position. Using Bayes' law, the optimal location for the object is defined as,

$$X^* = \arg\max_X p(X|I) = \arg\max_X p(I|X)p(X),$$

where $p(I|X)$ is normally called the likelihood model, and $p(X)$ is a prior distribution over configurations. The likelihood model encodes the image formation process. For example, the image tends to have high gradient near the object boundary. The prior distribution $p(X)$ encodes which configurations the object is likely to assume.

The matching problem from the last chapter can be cast in this statistical framework, by considering the energy we were minimizing as the negative logarithm of the posterior $p(X|I)$ (up to an additive constant). Previously we defined the location of a deformable template by a map from the vertices of the template to the image plane $g: V \to \mathbb{R}^2$. In this setting, the configuration of the object in the image is given by $X_i = g(v_i)$. The energy function we used for the matching experiments was defined in Section 3.3,

$$E(g, I) = \sum_{t \in T} \mathrm{def}(f_t)^2 - \lambda \int_{\partial P} \frac{\|(\nabla I \circ f)(s) \times f'(s)\|}{\|f'(s)\|}\, ds.$$

The first term in the energy is a cost for deforming the template, and corresponds to the negative logarithm of $p(X)$. The second term in the energy encourages the shape boundary to align with areas in the image with high gradient, and corresponds to the negative logarithm of $p(I|X)$.

The choice of the deformation costs in the energy function above is somewhat arbitrary. The learning problem we address is to find a prior distribution $\hat{p}(X)$ that approximates the true one by observing

random examples of the object. This will allow us to have more specific deformation models for different objects. By restricting $\hat{p}(X)$ to a class of functions that can be expressed in a particular form we can use the learned model to detect shapes in images using the techniques from the last chapter.

4.2 Procrustes Mean Shape

Before describing how we learn deformable shape models we review a standard method to estimate the mean shape of landmark data known as generalized Procrustes analysis (see [12] or [17] for more details). This is very similar to the learning procedure used for active shape models.

In two-dimensions the analysis is simplified using complex arithmetic. We identify a landmark $(x, y) \in \mathbb{R}^2$ with $x+iy \in \mathbb{C}$. In this case a configuration is given by a vector of complex numbers. Note that two configurations $x, y \in \mathbb{C}^n$ have the same shape when $y = c1_n + \beta e^{i\theta} x$, where 1_n is the vector of n ones, c is a complex number corresponding to a translation, $\beta > 0$ corresponds to a change of scale and θ corresponds to a rotation.

Say we have a set of random configurations $\{x_1, \ldots, x_m\}$. Suppose each sample is obtained from a mean configuration μ by a small perturbation ϵ_i and a similarity transformation,

$$x_i = c_i 1_n + \beta_i e^{i\theta}(\mu + \epsilon_i).$$

The perturbations account both for measurement error and variation in the shape of a population. We assume that the perturbations are independent and identically distributed according to a spherical Gaussian distribution. In this case a maximum likelihood estimate of the mean shape μ can be obtained using Procrustes analysis. First we translate the samples so that ${x_j}^* 1_k = 0$, where y^* denotes the transpose of the complex conjugate of y. This does not change the shape of the configurations and simplifies the analysis.

Definition 4.1. *The full Procrustes mean shape $[\hat{\mu}]$ is obtained by the following minimization over a mean configuration μ,*

$$\hat{\mu} = \operatorname*{argmin}_{||\mu||=1} \min_{\beta_j, \theta_j} \sum_{j=1}^{m} ||\mu - \beta_j e^{i\theta_j} x_j||^2.$$

The mean shape defined by $[\hat{\mu}]$ is also the maximum likelihood estimate of the modal shape under various distributions in shape space.

The mean configuration can be computed efficiently by solving a complex eigenvector problem. The solution is given by the eigenvector corresponding to the largest eigenvalue of the complex sum of squares and products matrix,

$$S = \sum_{j=1}^{m} x_j x_j^* / (x_j^* x_j).$$

Once we have computed the mean configuration, the optimal alignment between each configuration and the mean can be easily computed,

$$z_j = x_j^* \hat{\mu} x_j / (x_j^* x_j).$$

These are called the full Procrustes coordinates of the configurations. For each configuration we define the Procrustes residuals by $r_j = z_j - \hat{\mu}$. The sum of squares of the residuals gives an overall measure of shape variability. More detailed analysis of the variability can be performed using principal component analysis as discussed in [7].

Definition 4.2. *An overall measure of shape variability is given by the* root mean square *of the residuals,*

$$\mathrm{RMS}(\hat{\mu}) = \sqrt{\frac{1}{n} \sum_{j=1}^{m} ||r_j||^2}.$$

One problem with Procrustes analysis comes from the assumption that objects can be approximately aligned using similarity transformations. In practice this means that these techniques are useful to model objects that are almost rigid. In the next section we will relax this assumption using models based on triangulated polygons. Our techniques can be used to represent objects that are almost rigid locally, but can have large global deformations.

4.3 Deformable Templates

Now we turn to the problem of learning a deformable template model for an object. We need to estimate a prior distribution for the configuration of the object, and the distribution should be in a form that can be used by our matching algorithm. The matching algorithm can handle arbitrary energy functions that are a sum of costs, one for each triangle in a 2-tree over the landmarks in the model,

$$E(X, I) = \sum_{(v_i, v_j, v_k) \in T} c_{ijk}(X_i, X_j, X_k, I).$$

The energy function corresponds to the negative log of the posterior $p(X|I) \propto p(I|X)p(X)$. Thus, one requirement for the prior distribution is that its negative logarithm be a sum of costs, one for each triangle in a 2-tree. This is the form we would obtain by assuming that the shapes of each triangle in the model are independent. As described in Section 2.3 the shape of a set of landmarks is defined by a 2-tree over the landmarks and the shapes of each triangle in the 2-tree. This means that if we take the product of priors for the shape of each triangle in a 2-tree we do indeed get a distribution over the shape of the whole object.

The energy function also needs to take into account the image data, which is the contribution coming from the likelihood model $p(I|X)$. At a minimum, the likelihood model should depend on the boundary of the deformable object. This implies that the 2-tree that we choose for the model should include all the edges corresponding to the polygon boundary. If we want the likelihood to depend on the interior of the object then we need the 2-tree to be a planar triangulation of each polygon. In this way the interior of the object will be decomposed into parts by the triangles in the model. For now we require only that the 2-tree include the polygon boundary.

Let $E = \{X^1, \ldots, X^m\}$ be a set of random configurations for a polygon on n vertices and let T be a 2-tree over the polygon vertices. To define the prior over configurations let

$$p_T(X) = \prod_{(v_i, v_j, v_k) \in T} p_{ijk}(X_{ijk}),$$

where X_{ijk} is the sub-configuration of X containing the landmarks i, j and k. We use the model from the last section for each triangle in T. That is, for each triangle $(v_i, v_j, v_k) \in T$ we assume that the sub-configurations $\{X^1_{ijk}, \ldots, X^m_{ijk}\}$ were obtained from a mean triangle μ_{ijk} by a spherical Gaussian perturbation and an arbitrary similarity transformation.

When T is fixed we can estimate the prior $\hat{p}_T(X)$ using generalized Procrustes analysis for each triangle in the model. The Procrustes residuals give an estimate for the standard deviation of the perturbation vector for each triangle. Triangles with small residuals correspond to areas of the model that are almost rigid, while triangles with large residuals correspond to parts of the model that tend to deform. Note how this procedure does not assume that the polygons can be accurately aligned to each other using similarity transformations. The alignment only needs to be relatively accurate for each set of corresponding triangles in the samples. Intuitively, our models makes sense under an

assumption that objects are *locally almost rigid*.

Now we consider the case where no triangulation is given a priori. In this case the learning algorithm has to automatically select a common triangulation for the polygons. The choice can be made using a maximum-likelihood principle. Let $\mathcal{T}$ be the set of 2-trees that have the n-cycle $(v_1, \ldots, v_n, v_1)$ as a subgraph. These are the 2-trees that contain all edges corresponding to the boundary of the deformable object. The following theorem makes it possible for us to efficiently select an optimal model.

Theorem 4.1. *Elements of $\mathcal{T}$ correspond to triangulations of a convex n-gon with boundary vertices labeled by $(v_1, \ldots, v_n)$.*

Proof. First, if we have a triangulation of a convex n-gon, the boundary edges form the cycle $(x_1, \ldots, x_n, x_1)$, and we know that any triangulation of a simple polygon is a 2-tree.

Now suppose we have a 2-tree T with the special cycle. If $n = 3$ the 2-tree is a triangle and we are done. We proceed by induction. Let x_i be a simplicial vertex of T. This vertex must be connected to x_{i-1} and x_{i+1} by the cycle condition. Since x_i is simplicial, it is not connected to any other vertices, and x_{i-1} is connected to x_{i+1}. Removing x_i from T we obtain a 2-tree T' over the remaining $n-1$ vertices, with the cycle $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ in T'. By induction, T' is a triangulation of the convex $(n-1)$-gon induced by the remaining vertices. Adding the triangle (x_{i-1}, x_i, x_{i+1}) we obtain a triangulation of the convex n-gon. □

This correspondence implies that when searching for the graphical structure of our model we can search over triangulations of a convex n-gon. A well known algorithm (see [8]) can be used to find an optimal triangulation for a convex polygon, where the cost of the triangulation is a sum of costs, one for each triangle. Now it remains to see that this can be used to solve our problem, and what the costs for each triangle should be.

The maximum-likelihood estimate for the model is defined as,

$$\hat{T} = \operatorname*{argmax}_{T \in \mathcal{T}} \prod_{l=1}^{m} \prod_{(v_i, v_j, v_k) \in T} \hat{p}_{ijk}(X_{ijk}^l),$$

where $\hat{p}_{ijk}$ is the maximum likelihood estimate for the shape prior of triangle (v_i, v_j, v_k). By taking the negative logarithm of this equation we can express the optimal 2-tree as the one minimizing a sum of costs

for each triangle,

$$\hat{T} = \operatorname*{argmin}_{T \in \mathcal{T}} \sum_{(v_i, v_j, v_k) \in T} c_{ijk}, \quad \text{where } c_{ijk} = -\sum_{l=1}^{m} \log \hat{p}_{ijk}(X_{ijk}^l).$$

With some algebra we can see that the costs are equal to the log of the root mean square of the Procrustes residuals up to multiplicative and additive constants. These constants do not affect the solution for the optimal 2-tree so we can take $c_{ijk} = \log \mathrm{RMS}(\hat{\mu}_{ijk})$. Because of the logarithmic dependence on the residuals, the maximum-likelihood solution for the deformable model tends to concentrate all of the variation in shape to as few triangles as possible. Intuitively, the learning procedure should select models that are rigid almost everywhere, keeping the deformations localized.

Recall that we may want to enforce that the learned model yield a planar triangulation for each input polygon. If this is the case we just need to set $c_{ijk} = \infty$ for each triangle that is not in the interior of some polygon. The planarity requirement may not be satisfiable as not every set of polygons have a common planar triangulation. If this is the case the optimal triangulation with the modified c_{ijk} will have infinite cost.

We do not have to restrict the prior model for the shape of each triangle to the ones learned with Procrustes analysis. The method just described works with any choice of prior for the triangle shapes. Whatever form we choose for the prior distribution of a triangle, we just need to compute the costs c_{ijk} accordingly. Then the optimal triangulation is selected with the same algorithm. Note that the optimal triangulation may be different for different choices of priors. A number of different shape priors for landmark data that could be used here are described in [12].

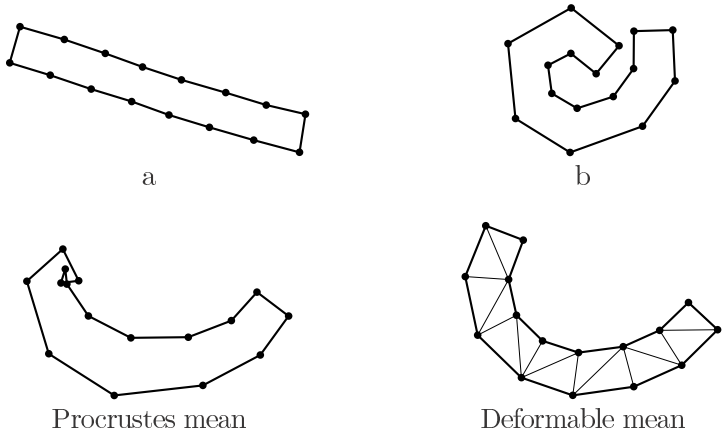


Figure 4.2: Comparing Procrustes and deformable mean shapes. Two input polygons are shown in (a) and (b). Procrustes analysis breaks down when the samples can not be aligned using similarity transformations.

4.4 Experimental Results

One problem with Procrustes analysis (and active shape models) is the assumption that objects from a population can be approximately aligned using similarity transformations. When this assumption breaks, the Procrustes mean shape can be quite bad as shown in Figure 4.2. The same figure shows a model learned with our technique, which only assumes that the object is locally almost rigid. The mean shape computed by our method is exactly what we expect from a deformable average of the two objects shown.

Hand model: We obtained a database with 40 pictures of hands. Each picture was annotated with the location of 56 landmarks located along the boundary of the hand. We can consider the landmarks as the vertices of polygons that approximate the boundary of each hand. Some examples of the objects in this data set are shown in Figure 4.3. The model selected by our learning algorithm is shown in Figure 4.4. The graphical structure of the model is somewhat surprising, we expected something more like a Delauney triangulation. But this structure is optimal in the maximum likelihood sense, given the class of distributions we are using for the shape of each triangle. To see that the learned

model does indeed capture the typical variation in the shape of the hands we generated the random samples from $\hat{p}(X)$. The random samples are shown in Figure 4.5. Note how the shape variation among the samples is similar to the variability present in the training set.

Maple Leaf model: We also trained a model to capture the shapes of maple leafs. In this case we had 34 example pictures, each annotated with the location of curvature extrema along the boundaries of each leave. Some examples of the objects in this data set are shown in Figure 4.6. For this experiment we enforced the constraint that the learned model be planar. The model selected is shown in Figure 4.7. Random samples from the estimated shape prior are shown in Figure 4.8. Again we see that our model seems to capture the typical shapes of maple leaves well.

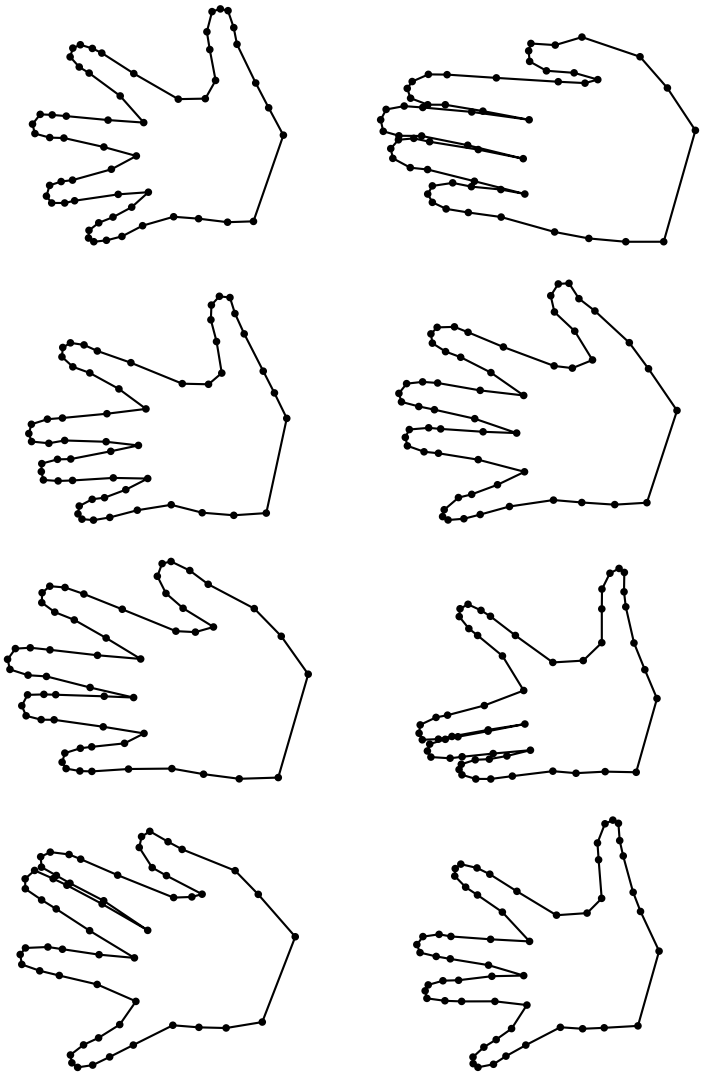


Figure 4.3: A few of a total of 40 samples of hands.

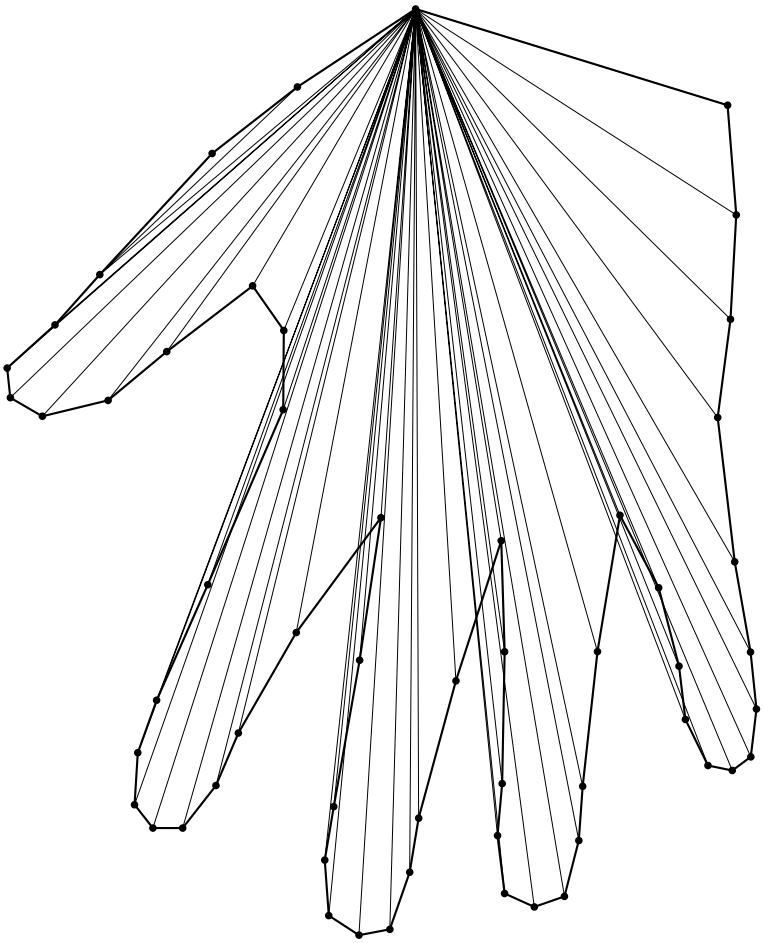


Figure 4.4: Deformable model learned for the hand data (without the planarity constraint).

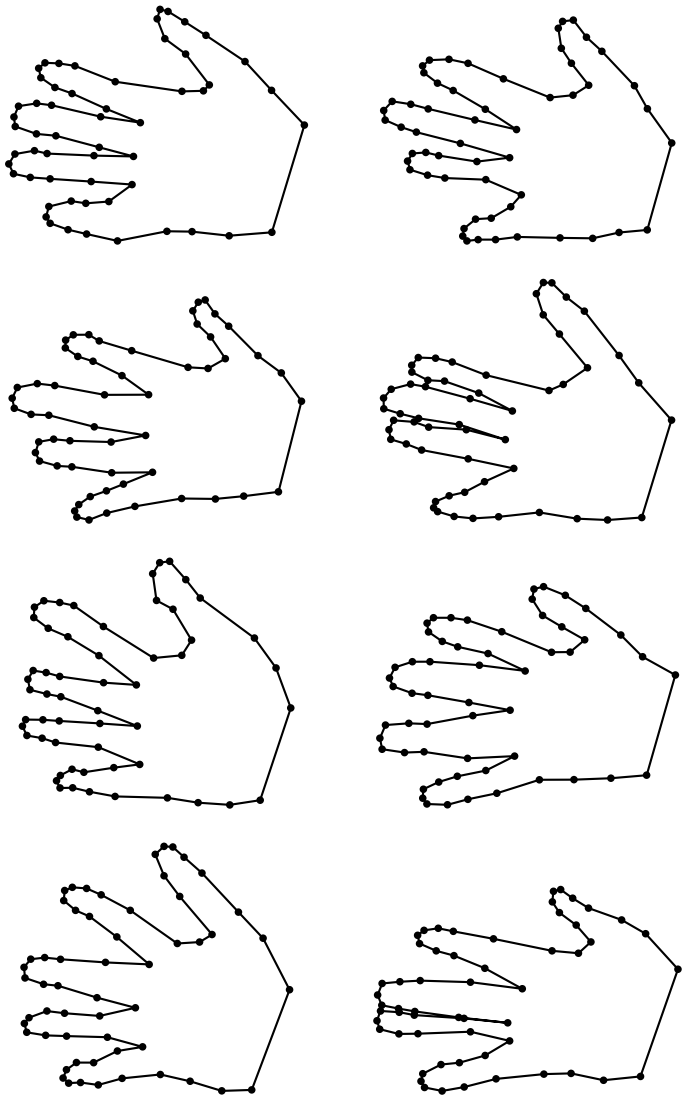


Figure 4.5: Random samples from the prior model for the hands.

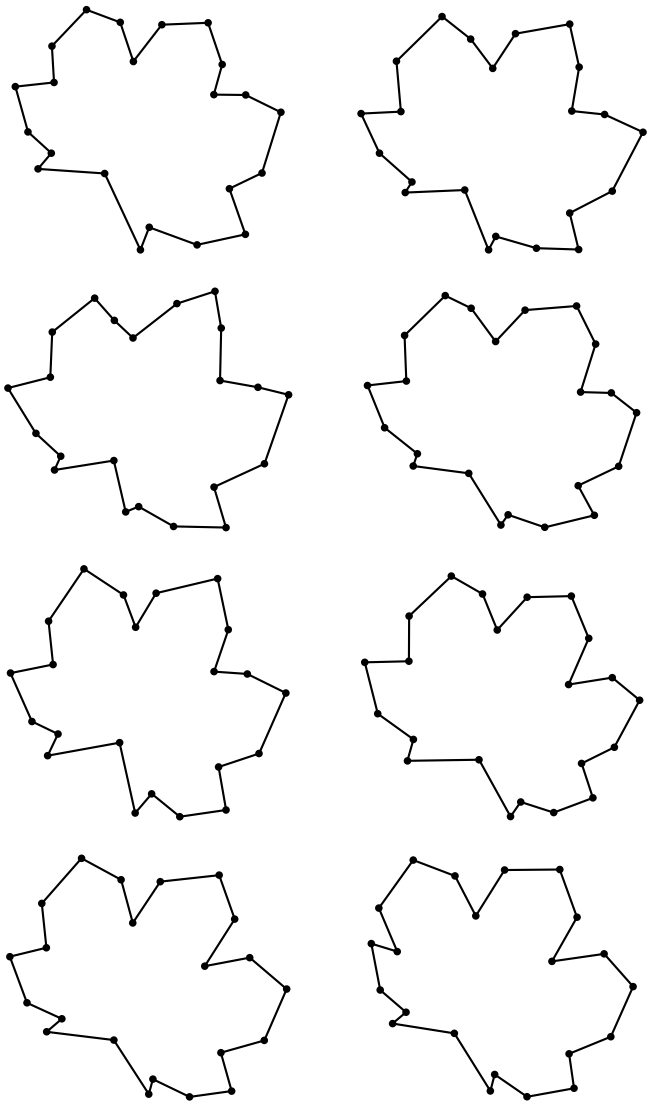


Figure 4.6: A few of a total of 34 maple leaves.

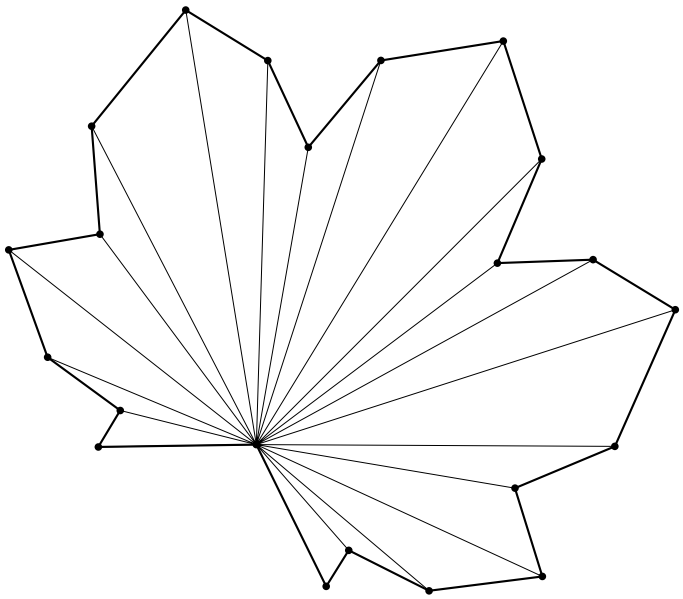


Figure 4.7: Deformable model learned for the maple leaf data (with the planarity constraint).

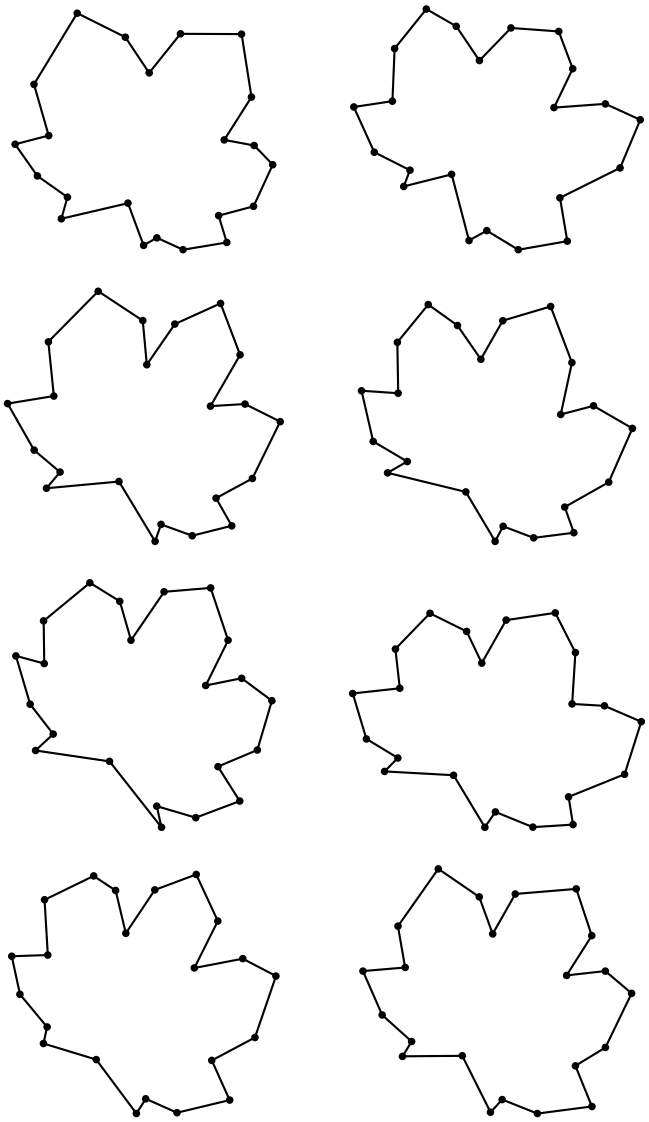


Figure 4.8: Random samples from the prior model for the leaves.

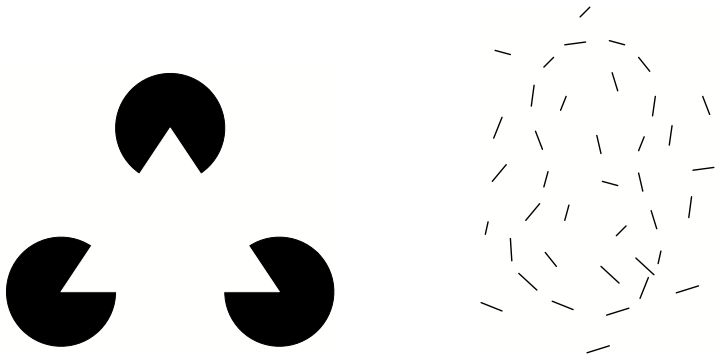
Chapter 5

Random shapes

In this chapter we consider the problem of detecting objects in images without knowing their identities in advance. In computer vision this is the goal of a low-level segmentation or grouping algorithm. In Chapter 3 we were able to detect objects in very noisy images because we had strong prior knowledge about the shapes of the objects we were looking for. One of the main challenges we face now is to build a prior model for shapes that is strong enough to interpret the ambiguous information in an image, but generic enough to capture most of the objects in a typical scene.

It seems clear that the shapes we usually encounter are not completely random, they have certain regularities. For example, shapes formed by random walks are too irregular when compared to the shape of a typical object. The regularities of natural shapes can be used to infer the locations of the objects in an image. Many theories in perceptual science suggest that our visual system favors the perception of some shapes over others. This is illustrated by the pictures in Figure 5.1. The Gestalt psychologists identified certain properties which guide the grouping of tokens such as edges and corners in our visual system. Some of the strong grouping cues are: proximity, continuity, collinearity, cocircularity, parallelism, symmetry and closure.

Intuitively, the tokens in an image should be grouped into regular shapes. This idea has been previously studied in computer vision (for example [35], [31], [30], [24], [27] and [42]). We propose a method in which a pre-attentive process searches the image for regular shapes to generate object hypotheses. These hypotheses must then be processed further in a way that depends on the perceptual task at hand. For example, each hypothesis could be matched against a database of known



(a) (b)

Figure 5.1: (a) In this picture we are inclined to see a triangle even though it is not really there. (b) Our visual system automatically groups some of the edges into a "peanut" shape.

objects to establish identities. Our algorithm works by sampling shapes from a particular distribution that depends on an input image. The distribution is constructed so that shapes with high probability look natural, and their boundaries align with areas of the image that have high gradient magnitude. Our notion of natural shapes is motivated by the gestalt grouping laws.

We start by defining a stochastic grammar that generates random triangulated polygons. This grammar can be tuned to capture many of the Gestalt grouping laws. For example, with the correct choice of parameters the random shapes generated tend to have smooth boundary and a nice decomposition into elongated parts. In this way we obtain a generic but rich prior model for the objects in an image. We combine this prior with a likelihood model that defines the probability of observing an image given the presence of a particular shape in the scene. These two distributions together define a posterior distribution over random shapes in a scene. Samples from the posterior provide good hypotheses for the objects in an image.

5.1 Shape Grammar

As described in Chapter 2, if T is a triangulation of a simple polygon, then its dual graph G_T is a tree. Figure 5.2 shows a triangulated polygon and its dual graph. There are three possible types of triangles in T, corresponding to nodes of different degrees in G_T. The three

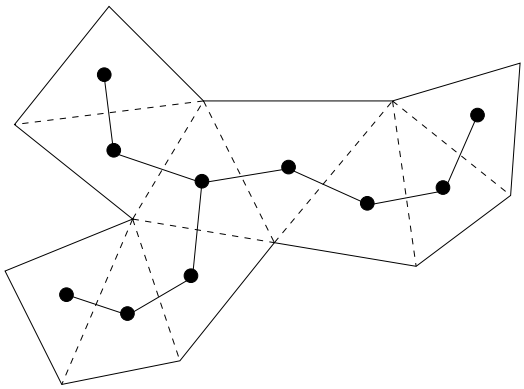


Figure 5.2: A triangulated polygon and its dual graph. If the polygon is simple the dual graph is a tree.

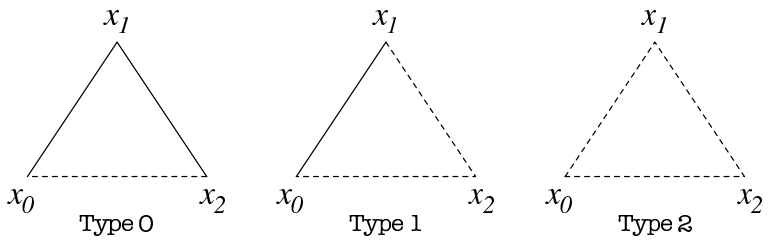


Figure 5.3: Different triangle types in a triangulated polygon. The types corresponds to nodes of different degrees in the dual graph.

triangle types are shown in Figure 5.3, where solid edges are part of the polygon boundary, and dotted edges are diagonals in the triangulation. For the rest of this chapter we will use a particular labeling of the triangle vertices as shown in this picture. In this way, a triangle is defined by its type and the location of its vertices $x_0, x_1, x_2 \in \mathbb{R}^2$. Sequences of triangles of the second type form branches (or necks) of the shape. Triangles of the first type correspond to ends of branches, and triangles of the third type connect multiple branches together.

A procedure to generate triangulated polygons is given by the following growth process. Initially a seed triangle is selected from one of the three possible types. Then each dotted edge of the seed grows into a new triangle. Growth continues along newly created dotted edges

until all branches end by growing a triangle of the first type with only one dotted edge. A similar process for growing combinatorial structures known as n-clusters is described in [21]. The growth process can be made stochastic by defining a few probability distributions. Let a triangle of type i be selected (initially or during growth) with probability t_i. As an example, imagine picking t_1 to be the largest value. This would encourage growth of shapes with long branches. Similarly, t_2 will control the number of branches in the shape. Besides the three parameters t_i, we also have a distribution that controls the shape of each triangle created. The probability that a shape $[X]$ is selected for a triangle of type i equals $s_i([X])$.

The growth process can be characterized by a stochastic context free grammar (or a tree automaton, see [13]). The grammar not only generates triangulated polygons, it also generates objects with overlapping parts as illustrated in Figure 5.4. To be precise, the grammar generates planar 2-trees (see Section 2.1) that are embedded in the plane by a map that is not necessarily one to one. Let $t = \{0, 1, 2\}$ be the three possible triangle types. There are two types of symbols in the grammar. The possible triangles created during growth are elements of $t \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2$, corresponding to a type and the location of the vertices. There are also symbols corresponding to edges that need to grow, and these are elements of $\mathbb{R}^2 \times \mathbb{R}^2$. The edges are oriented so the grammar remembers to which side of the edge the next triangle should grow. Figure 5.5 illustrates the production rules for the grammar. Note that there are two different rules to grow a triangle of type one, corresponding to a choice of how the new triangle is glued to the edge that is growing. We simply let both choices have equal probability.

To understand the effect of the t_i, consider the dual graph of a triangulated polygon generated by the stochastic process just described. The growth of the dual graph starts in a root node that has one, two or three children with probability t_0, t_1 and t_2 respectively. Now each child of the root grows according to a Galton-Watson process (see [20]), where we start with a single node that has i children with probability t_i and the children themselves reproduce with the same distribution.

An important parameter of the Galton-Watson process is the expected number of children for each node or Malthusian parameter, which we denote by m. In our process, $m = t_1 + 2t_2$. When $m < 1$ the probability that the growth process eventually terminates is one. From now on we will always assume that $m < 1$, which is equivalent to saying $t_2 < t_0$. In this case we can define a distribution over triangulated polygons by letting $p(T)$ be proportional to the probability that the grammar would generate T.

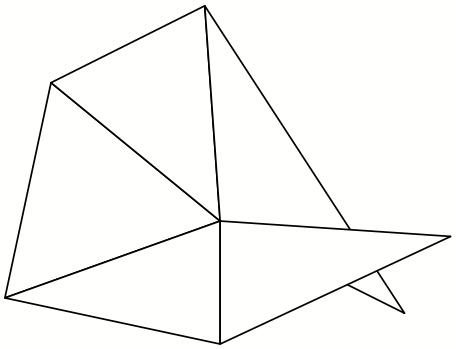


Figure 5.4: The grammar can generate objects that cross over themselves, this never happens in a triangulated polygon.

Let e, b and j be random variables corresponding to the number of end, branch and junction triangles in a random shape, and $n = e + b + j$ is the overall number of triangles. For the Galton-Watson process (which corresponds to growth from each child of the root) we can compute the expected number of nodes generated, which we denote by x,

$$x = 1 + (x)t_1 + (2x)t_2 \;\Rightarrow\; x = 1/(t_0 - t_2).$$

The number of nodes in the dual graph is obtained as one node for the root plus the number of nodes in the subtrees rooted at each children of the root. So the expected value of n is,

$$E(n) = 1 + (x)t_0 + (2x)t_1 + (3x)t_2.$$

Substituting for x we get,

$$E(n) = \frac{2}{t_0 - t_2}. \tag{5.1}$$

Similarly we can compute the expected number of junction triangles in a random shape. The number of junction triangles is interesting because it gives a measure of the complexity of the shape, in particular it is a measure of the number of parts (limbs, necks, etc). For the Galton-Watson process, let y be the expected number of nodes with degree 3 (two children),

$$y = (y)t_1 + (1 + 2y)t_2 \;\Rightarrow\; y = t_2/(t_0 - t_2).$$

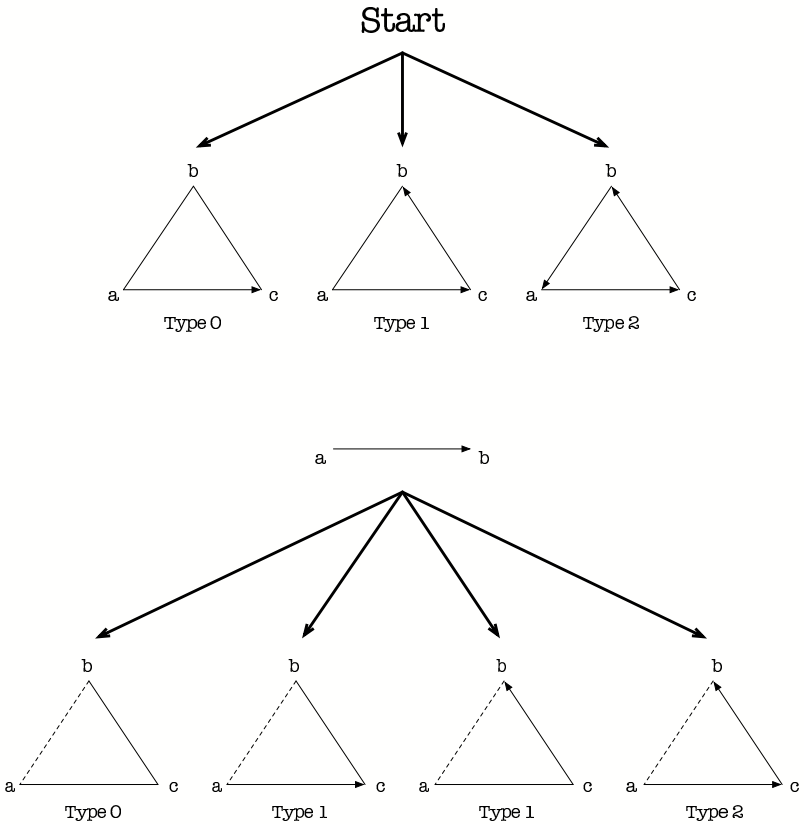


Figure 5.5: Production rules for the context free grammar. The variables a, b and c correspond to locations in the plane. The three variables are selected in a production from the start symbol, but only c is selected in a production from an edge.

Now we compute the expected number of junction triangles in the whole shape. The number of junction triangles equals the number of such triangles in each subtree of the root plus one if the root itself is a junction triangle,

$$E(j) = (y)t_0 + (2y)t_1 + (1+3y)t_2.$$

Substituting for y we get,

$$E(j) = \frac{2t_2}{t_0 - t_2}. \tag{5.2}$$

Equations 5.1 and 5.2 give us intuition to the effect of using different parameters. Also, these equations show that the three parameters t_0, t_1 and t_2 are uniquely defined by the expected number of triangles and the expected number of junction triangles in a random shape. We can compute the t_i corresponding to any pair $E(n)$ and $E(j)$ such that $E(n) \geq 2$ and $E(n) \geq 2E(j) + 2$. These requirements are necessary for consistency, since our growth process always creates at least two triangles and the number of triangles is always at least twice the number of junction triangles plus two.

$$\begin{aligned} t_0 &= (2 + E(j))/E(n), \\ t_1 &= 1 - (2E(j) + 2)/E(n), \\ t_2 &= E(j)/E(n). \end{aligned}$$

While the t_i control the combinatorial structure of the random shapes their geometry is highly dependent on the choice of shape for each triangle. The triangle shapes are chosen according to distributions that depend on the triangle type. As an example we can define,

$$s_i([X]) \propto e^{-k_i \operatorname{def}(X_i, X)^2},$$

where X_i is an ideal triangle of type i and $\operatorname{def}(X_i, X)$ is the log-anisotropy of the map taking X_i to X as defined in Chapter 3. The constant k_i controls how much the shapes are allowed to vary. For the experiments in this chapter we chose both X_0 and X_2 to be equilateral triangles and X_1 to be isosceles, with a smaller side corresponding to the polygon boundary edge. This choice for X_1 generates shapes that tend to have smooth boundaries. Figure 5.6 shows what happens when we connect multiple triangles of type one with alternating or similar orientations.

Figure 5.7 shows some random shapes generated by the random process with $E(n) = 20$, $E(j) = 1$, and the choice for $s_i([X])$ described

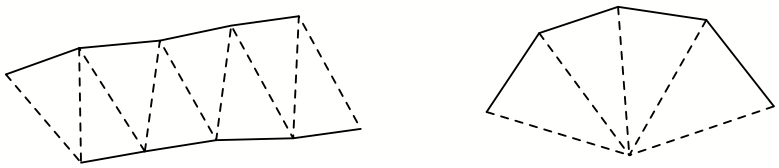


Figure 5.6: Connecting multiple neck triangles in alternating orientations to form an elongated shape, and with the same orientation to form a bend. If the neck triangles tend to be isosceles and thin than the shape boundary tends to be smooth.

above. Note how the shapes have a nice decomposition into parts, and each part has an elongated structure, with smooth boundaries almost everywhere. These examples illustrate some of the gestalt principles captured by our shape grammar. In the next section we will show how the grammar can be used for low-level grouping and segmentation.

5.2 Sampling Shapes From Images

Now we describe how the prior on random shapes defined by the stochastic grammar can be combined with a likelihood model to yield a posterior $p(T|I)$ over triangulated polygons in an image. We then show how to sample from the posterior using a dynamic programming procedure. These samples provide good hypothesis for the objects in a scene.

The shape grammar generates triangulated polygons rooted at a particular triangle r where growth starts. Let T_r denote a rooted shape. Recall that each triangle created during growth is an element of $t \times \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2$, corresponding to a type and the location of the vertices. We assume that the likelihood $p(I|T_r)$ factors into a product of terms, one for each triangle and it does not depend on the choice of the root,

$$p(I|T) = \prod_{(i,a,b,c) \in T} \pi_i(a, b, c, I).$$

For example, we expect the image to have high gradient perpendicular to the boundary of the objects in the scene. So we can use a function similar to the data term defined in Chapter 3,

$$P(I|T) \propto \exp\left(\lambda \int_{\partial P} \|(\nabla I \circ f)(s) \times f'(s)\| \, ds\right),$$

where ∂P is the boundary of the polygon defined by T, and $f(s)$ is a parametrization of the boundary by arclength. This function can be

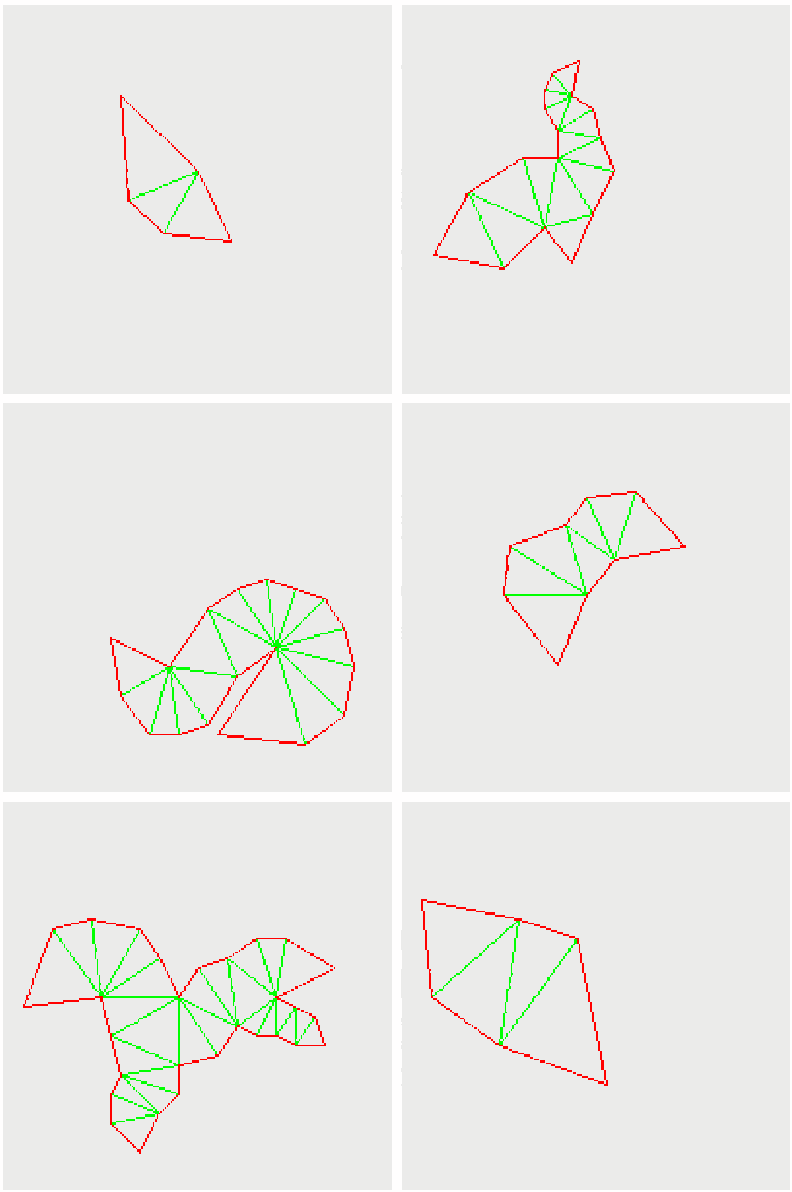


Figure 5.7: Random shapes generated by the stochastic grammar.

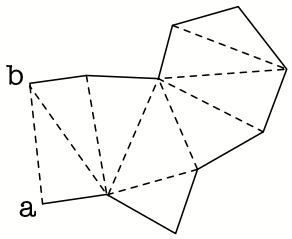


Figure 5.8: A partial shape generated from the edge (a, b).

written as a product of terms corresponding to the triangles in T. In this case each term takes into account the piece of the polygon boundary that belongs to a particular triangle.

Using Bayes' law we can write the posterior distribution for rooted shapes given an observed image as,

$$p(T_r|I) \propto p(T_r)p(I|T).$$

There are two approximations we need to make to be able to sample from the posterior distribution efficiently. We consider only shapes where the depth of the dual graph is bounded by a constant d (the depth of a rooted graph is the maximum distance from a leaf to the root). This should not be a problem since shapes with too many triangles have low prior probability. Moreover, the running time of our sampling algorithm is linear in d, so we can let this constant be relatively large. We also only consider shapes where the location of each vertex is constrained to lie on a grid $\mathcal{G}$, as opposed to an arbitrary location in the plane.

To sample from the posterior we first pick a root triangle, then pick the triangles connected to the root and so on. The root should be selected according to its the marginal distribution,

$$p(r|I) = \sum_{T_r} p(T_r|I). \tag{5.3}$$

Note that the sum is over all shapes rooted at r, and with the depth of the dual graph bounded by d. Computing this marginal is possible because the triangles in a shape are connected together in a tree structure.

Let $T_{(a,b)}$ denote a partial shape generated from an edge (a, b) as illustrated in Figure 5.8. We denote the probability that the grammar would generate $T_{(a,b)}$ starting from the edge (a, b) by $p(T_{(a,b)})$. The

posterior probability of $T_{(a,b)}$ given an image is given by,

$$p(T_{(a,b)}|I) \propto p(T_{(a,b)}) \prod_{(i,a,b,c)\in T_{(a,b)}} \pi_i(a,b,c,I).$$

We define the following quantities in analogy to the forward and backward weights of a Hidden Markov Model (see [32]),

$$V_j(a,b) = \sum_{T_{(a,b)}} p(T_{(a,b)}|I),$$

where the sum is taken over all partial shapes with depth at most j. Here we measure depth by imagining the root to be a triangle that would be immediately before the edge (a,b). The quantities $V_j(a,b)$ can be computed recursively using a dynamic programming procedure,

$$\begin{aligned}
V_0(a,b) &= 0, \\
V_j(a,b) &= t_0 \sum_c s_0([b,c,a])\, \pi_0(b,c,a,I) + \\
&\quad (t_1/2) \sum_c s_1([b,c,a])\, \pi_1(b,c,a,I)\, V_{j-1}(a,c) + \\
&\quad (t_1/2) \sum_c s_1([c,a,b])\, \pi_1(c,a,b,I)\, V_{j-1}(c,b) + \\
&\quad t_2 \sum_c s_2([b,c,a])\, \pi_2(b,c,a,I)\, V_{j-1}(a,c)\, V_{i-1}(c,b).
\end{aligned}$$

Now, depending on the type of the root triangle we can rewrite the marginal distribution in equation 5.3 as,

$$\begin{aligned}
p((0,a,b,c)|I) &\propto t_0\, s_0([a,b,c])\, V_d(a,c), \\
p((1,a,b,c)|I) &\propto t_1\, s_1([a,b,c])\, V_d(a,c)\, V_d(c,b), \\
p((2,a,b,c)|I) &\propto t_2\, s_2([a,b,c])\, V_d(a,c)\, V_d(c,b)\, V_d(b,a).
\end{aligned}$$

The equations above provide a way to sample the root triangle from its marginal distribution. The running time necessary for computing all the $V_j(a,b)$ and the marginal distribution for the root triangle is $O(d|\mathcal{G}|^3)$. Once we compute these quantities we can obtain multiple samples for the root very fast (its just a discrete distribution). After choosing $r = (i,a,b,c)$ we need to sample the triangles connected to the root. We then sample the triangles that are at distance two from the root, and so on. When sampling a triangle at distance j from the root, we have an edge (a,b) that is growing. We need to sample

the location c of another vertex and a triangle type according to the following distribution:

$$\begin{aligned}
p((0,b,c,a)|I,(a,b)) &\propto t_0\, s_0([b,c,a]),\\
p((1,b,c,a)|I,(a,b)) &\propto (t_1/2)\, s_1([b,c,a])\, V_{d-j}(a,c),\\
p((1,c,a,b)|I,(a,b)) &\propto (t_1/2)\, s_1([c,a,b])\, V_{d-j}(c,b),\\
p((2,b,c,a)|I,(a,b)) &\propto t_1\, s_1([b,c,a])\, V_j(a,c)\, V_{d-j}(c,b).
\end{aligned}$$

In each case we simply compute the discrete distribution and then sample from it. Note that for a triangle at depth d the only choices with non-zero probability will have type zero, as $V_0(a,b) = 0$.

5.3 Experimental Results

For the experiments in this section we used a grid of 40×40 locations for the vertices of the shapes. We used the likelihood model defined in the last section, and the same grammar parameters used to generate the random shapes in Figure 5.7.

Figure 5.9 shows some of the random samples generated from a synthetic image with one object. Most of the samples correspond to a good interpretation of the scene. Note how we obtain multiple similar samples, they just correspond to different representations of the object in terms of triangulated polygons. In Figure 5.10 we show random samples from another synthetic image. In this case the samples are split between the two objects in the image. Again, the samples give a good interpretation of this scene.

Figures 5.11 and 5.12 illustrate samples obtained from real pictures. For the mushroom image, we obtained different samples corresponding to competing interpretations of the scene. In one case the whole mushroom is considered an object, with in another case the stem comes out on its own.

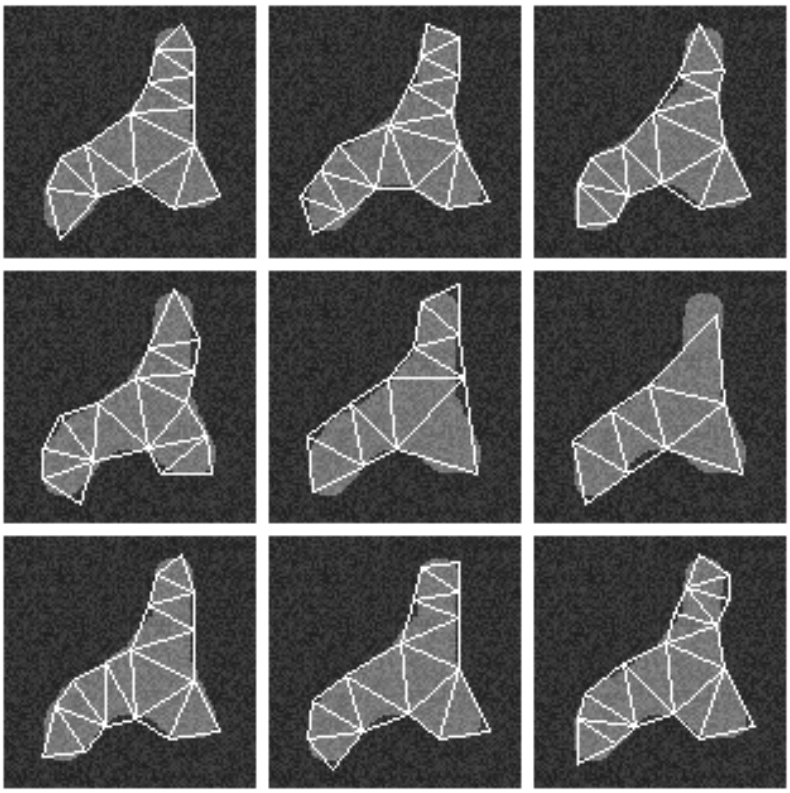


Figure 5.9: Some of the random shapes from an image with one object.

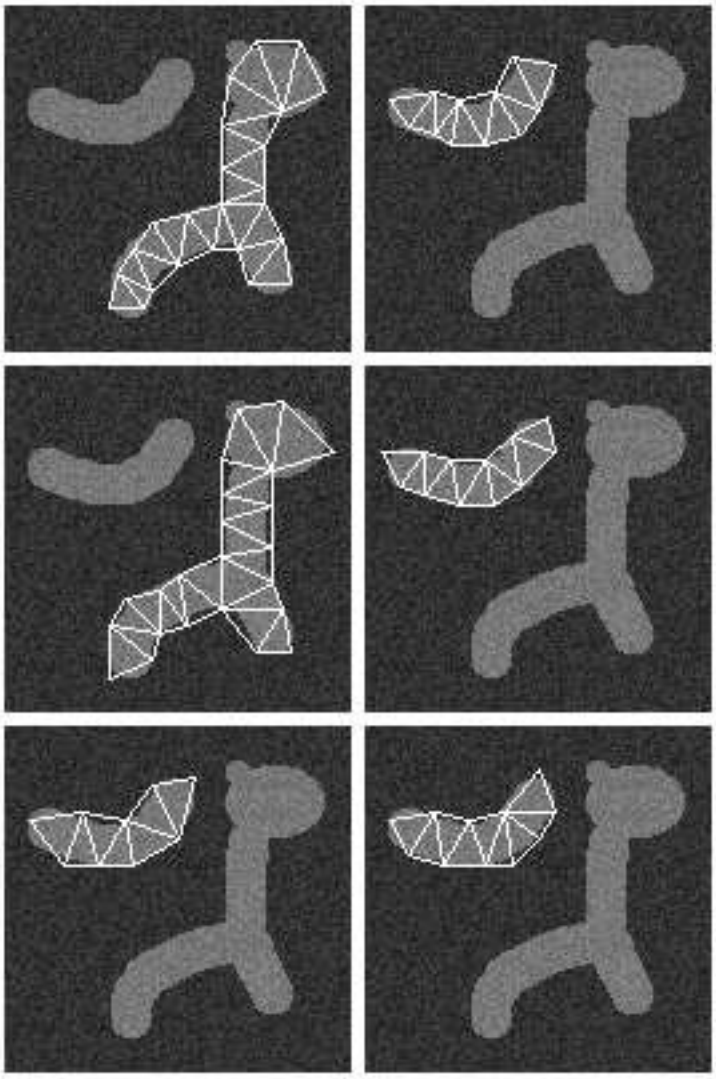


Figure 5.10: Some of the random shapes from an image with two objects. Both objects are found among the hypothesis generated.

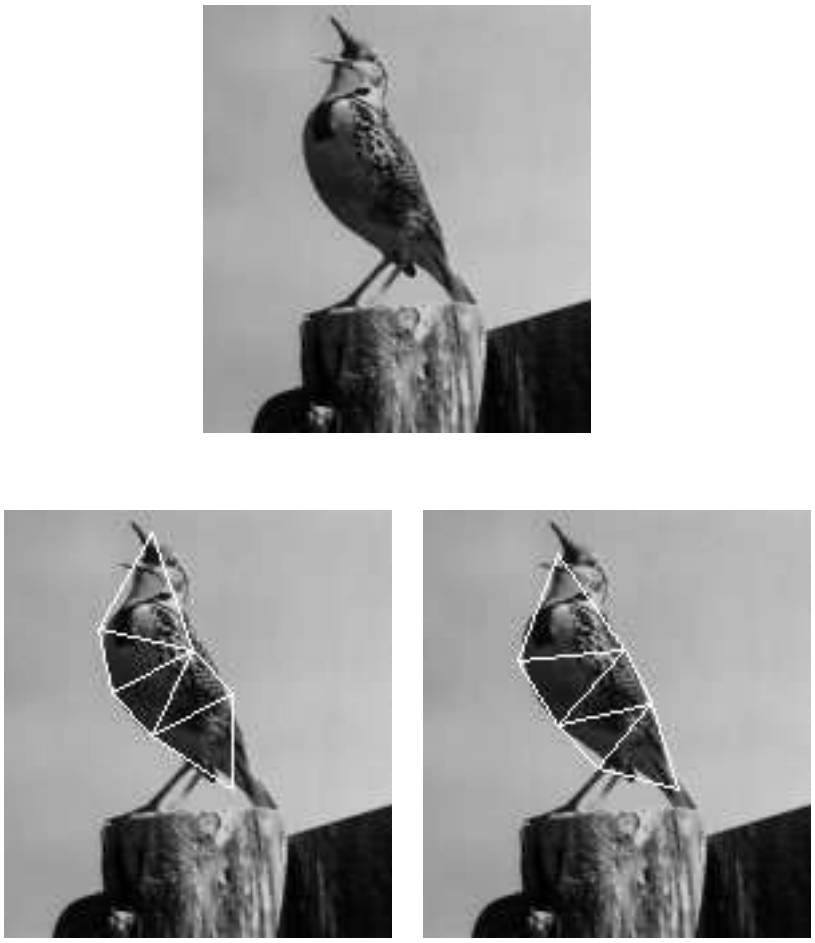


Figure 5.11: Samples from an image of a bird.

Figure 5.12: Samples from an image of a mushroom.

Chapter 6

Conclusion

In this thesis we have described how triangulated polygons can be used to represent planar shapes. This representation is closely related to the medial axis transform and provides a natural decomposition of arbitrary planar shapes into simple parts. The triangles that decompose a polygon without holes are connected together in a tree structure, and this has important computational consequences. For example, we have shown how triangulated polygon models can be used to represent and detect non-rigid objects in images. Our detection algorithm efficiently computes a global optimal solution to the deformable template matching problem and has a wide range of applications. We have also considered the problem of learning accurate non-rigid shape models for a class of objects. Our learning method computes good models while constraining them to be in the form required by the detection algorithm. Finally, we described a stochastic grammar that generates arbitrary triangulated polygons. The grammar captures many of the Gestalt principles of shape regularity and can be used as a prior model to detect objects in images.

6.1 Future Work

One important question is whether the techniques described in this thesis can be extended to three dimensions. This would be useful for medical image analysis, where we would like to detect structures in volumetric images. Unfortunately, obvious ways to extend the triangulated polygon models will not work. For example, we could consider decompositions of three dimensional objects into tetrahedra. But even

for objects without holes the tetrahedra will not necessarily be connected together in a tree structure.

Our matching procedure allows the triangles in a model to deform, but keeps the structure of the model fixed. It would be interesting to allow the model structure to change. This is important if we want to compute matchings between objects that have a different number of vertices or different triangulations. We could use some of the ideas in [34] to study how to move in the space of triangulated polygons.

The shape grammar described in the last chapter can be made more specific by using higher level constructs. We could have non-terminal symbols in the grammar that correspond to things like "elongated branch", "pinch point" and "bend". These symbols would then produce arrangements of triangles, constrained to follow a general structure but not with fixed geometry. Using these higher level constructs we might be able to build grammars that are tuned to generate particular classes of objects.

We would like to explore how to perform recognition of triangulated polygons. With the techniques from the last chapter we are able to generate hypotheses for the objects in a scene. It would be nice if we could recognize the identities of the objects based on their representation as triangulated polygons. For example, the number of triangles of each type gives a coarse description for a triangulated polygon that may be useful for recognition.

Bibliography

[1] Y. Amit and A. Kong. Graphical templates for model registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):225–236, 1996.

[2] R. Basri, L. Costa, D. Geiger, and D.W. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:2365–2385, 1998.

[3] M.W. Bern and D. Eppstein. Mesh generation and optimal triangulation. In Du and Hwang, editors, *Computing in Euclidean Geometry*, number 4 in Lecture Notes Series on Computing, pages 47–123. World Scientific, second edition, 1995.

[4] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

[5] I. Biederman. Recognition by components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987.

[6] H. Blum. Biological shape and visual science. *Theoretical Biology*, 38:205–287, 1973.

[7] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models: Their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.

[8] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1989.

[9] J. Coughlan, A. Yuille, C. English, and D. Snow. Efficient deformable template detection and localization without user initialization. *Computer Vision and Image Understanding*, 78(3):303–319, 2000.

[10] R.G. Cowell, A.P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag, 1999.

[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry*. Springer-Verlag, 1997.

[12] I.L. Dryden and K.V. Mardia. *Statistical Shape Analysis*. John Wiley & Sons, 1998.

[13] Clarence A. Ellis. Probabilistic tree automata. In *Proceedings of the Second Annual ACM Symposium on Theory of Computing*, pages 198–205, 1970.

[14] P.F. Felzenszwalb and D.P. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 66–73, 2000.

[15] M.A. Fischler and R.A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973.

[16] M.C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980.

[17] C. Goodall. Procrustes methods in the statistical analysis of shape. *Journal of the Royal Statistical Society, Series B*, 52(2):285–339, 1991.

[18] U. Grenander. *Elements of Pattern Theory*. Johns Hopkins University Press, 1996.

[19] U. Grenander, Y. Chow, and D.M. Keenan. *HANDS: A Pattern-Theoretic Study of Biological Shapes*. Springer-Verlag, 1991.

[20] M. Habib, C. McDiarmid, J. Ramirez-Alfonsin, and B. Reed. *Probabilistic Methods for Algorithmic Discrete Mathematics*. Springer-Verlag, 1998.

[21] F. Harary, Palmer E.M., and R.C. Read. On the cell-growth problem for arbitrary polygons. *Discrete Mathematics*, 11:371–389, 1975.

[22] D.D. Hoffman and W.A. Richards. Parts of recognition. *Cognition*, 18:65–96, 1985.

[23] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[24] D.W. Jacobs. Robust and efficient detection of salient convex groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1):23–37, 1996.

[25] A.K. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):267–278, 1996.

[26] M. Kass, A.P. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[27] M.S. Lee and G. Medioni. Grouping ., -, –¿, 0, into regions, curves, and junctions. *CVIU*, 76(1):54–69, 1999.

[28] M. Leventon, W. Grimson, and O. Faugeras. Statistical shape influence in geodesic active contours. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 316–323, 2000.

[29] D. Marr and H.K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London, Series B, Biological Sciences*, 200:269–294, 1978.

[30] D. Mumford. Elastica and computer vision. In *Algebraic Geometry and Its applications*, pages 491–506. Springer-Verlag, 1994.

[31] M. Nitzberg and D. Mumford. The 2.1-d sketch. In *IEEE International Conference on Computer Vision*, pages 138–144, 1990.

[32] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), February 1989.

[33] Thomas B. Sebastian and Benjamin B. Kimia. Curves vs skeletons in object recognition. In *IEEE International Conference of Image Processing*, 2001.

[34] T.S. Sebastian, P.N. Klein, and B.B. Kimia. Recognition of shapes by editing shock graphs. In *IEEE International Conference on Computer Vision*, pages 755–762, 2001.

[35] A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *IEEE International Conference on Computer Vision*, pages 321–327, 1988.

[36] K. Siddiqi and B.B. Kimia. Parts of visual form: Computational aspects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):239–251, 1995.

[37] K. Siddiqi and B.B. Kimia. A shock grammar for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 507–513, 1996.

[38] C.G. Small. *The Statistical Theory of Shape*. Springer-Verlag, 1996.

[39] M. B. Stegmann, B. K. Ersbøll, and R. Larsen. FAME - a flexible appearance modelling environment. *IEEE Transactions on Medical Imaging (to appear)*, May 2003.

[40] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):992–1005, 1991.

[41] B. Widrow. The rubber mask technique. *Pattern Recognition*, 5(3):174–211, 1973.

[42] S.C. Zhu. Embedding gestalt laws in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1170–1187, 1999.

[43] S.C. Zhu and A.L. Yuille. Forms: A flexible object recognition and modelling system. *International Journal of Computer Vision*, 20(3):187–212, 1996.