Representation and Detection of Deformable Shapes

Pedro F. Felzenszwalb
Department of Computer Science
The University of Chicago
Chicago, IL 60637
pff@cs.uchicago.edu

July 30, 2004

Abstract

We describe some techniques that can be used to represent and detect deformable shapes in images. The main difficulty with deformable template models is the very large or infinite number of possible non-rigid transformations of the templates. This makes the problem of finding an optimal match of a deformable template to an image incredibly hard. Using a new representation for deformable shapes we show how to efficiently find a global optimal solution to the non-rigid matching problem. The representation is based on the description of objects using triangulated polygons. Our matching algorithm can minimize a large class of energy functions, making it applicable to a wide range of problems. We present experimental results of detecting shapes in medical images and images of natural scenes. Our method does not depend on initialization and is very robust, yielding good matches even in images with high clutter. We also consider the problem of learning a non-rigid shape model for a class of objects from examples. We show how to learn good models while constraining them to be in the form required by the matching algorithm.

Keywords: Shape representation, Object recognition, Deformable templates, Chordal graphs, Dynamic programming.

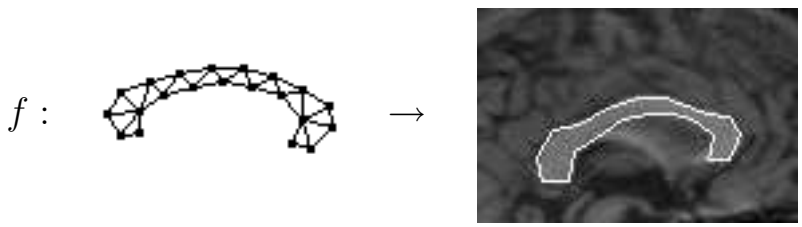


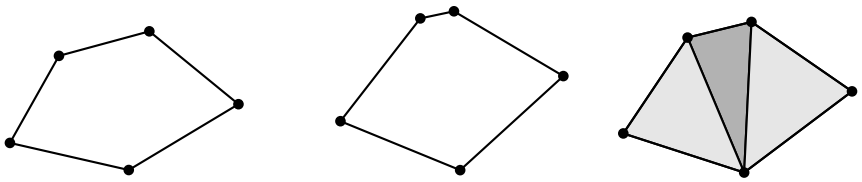
Figure 1: The function f maps a template to the image plane, characterizing the location and shape of a non-rigid object.

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.

Our main goal is to develop techniques that can be used to represent and detect relatively generic objects in images. The techniques we present here revolve around a particular shape representation, based on the description of objects using triangulated polygons. Triangulated polygons allow us to describe complex shapes using simple building blocks. As we show in the next section, 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 closely related to Blum's medial axis transform [6].

In this paper we concentrate on the task of finding the location of a deformable shape in an image. This problem is important for the recognition of non-rigid objects. Moreover, objects in many generic classes can be described as deformed versions of an ideal template. In this setting, the location of an object is given by a continuous map from a template to an image. Figure 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 can be deformed in an intuitive way. Equally important, we present an efficient algorithm for finding the optimal location of a deformable shape in an image. In contrast, previous methods that take into account the interior of deformable objects are too



(a) examples (b) model

Figure 2: Two examples of a deformable object (a), and the learned model (b). The color of each triangle in the model indicates how much their shapes vary across different examples (darker triangles vary more).

slow for practical use or rely on local search techniques to perform detection. These 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 to 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 2 illustrates the learning procedure for a simple object. The learning method is useful for constructing good models of different object classes in a semi-automatic way.

Below we briefly review some of the shape representations that have been previously used in computer vision. In Section 2 we describe how we can represent objects using triangulated polygons and some of the important properties of this representation. In Section 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 Section 4 we show how we can learn a deformable model from multiple examples of the objects in a class.

1.1 Shape Representations

The geometric properties of rigid objects are 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 [36], 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 [27] and geons [5]. These approaches are appealing because they provide symbolic descriptions of objects. 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 based on pictorial structures (e.g. [14, 13]) have been successfully used to characterize and detect objects that are described by a small number of simple parts connected in a deformable configuration. In this approach, generic parts are not extracted from images on their own, the whole object model is used at once. Our representation is similar in that objects are represented by a number parts connected together. When matching a triangulated polygon to an image we also consider 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 [25, 15]. Boundary models are also used for generic object recognition. Grenander et al. [20] pioneered the use of Markov models to represent the boundaries of non-rigid objects. They demonstrated how these models can be used to detect objects in noisy images. 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. 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 [32]

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 [30]. As shown in [29], medial axis models can better capture natural deformations of objects when compared to boundary models. An example of generic object recognition using a representation related to the medial axis transform is described in [38]. Our triangulated polygon representation is closely related to the medial axis transform. We obtain a discrete version of the medial axis similar to the representation based on self-similarities described in [16]. In particular our models capture local symmetries and provide natural decompositions of shapes into parts.

Statistical shape theory [33, 12] has been used to study objects that are defined by a set of labeled landmarks. In this scenario, the space of possible landmark configurations has led to natural notions of distances 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 active shape models [8]. While active shape models can capture the typical variation in shape among certain classes 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 shown 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 described in the next section, a triangulation of a simple polygon yields a natural decomposable graph that can be used for modeling shape.

2 Triangulated Polygons

In this section we describe how objects can be represented by 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 3. 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

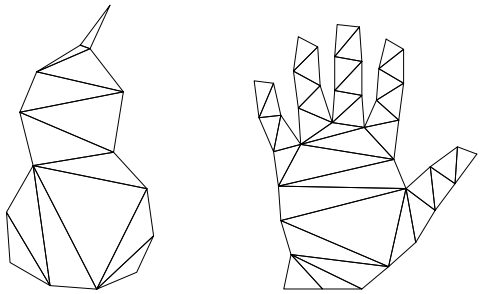


Figure 3: Pear, and hand represented by triangulated polygons. The polygonal boundaries represent the outlines, while the triangulations decompose the objects into parts.

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*. 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 are two natural graphs associated with a triangulated polygon. Let T be a triangulation of a polygon P. The graphical structure of the triangulation is denoted by G_T. The nodes of G_T correspond to the polygon vertices while the edges correspond to the polygon boundary and the diagonals in the triangulation. The dual graph of the triangulation is denoted by D_T. The nodes of D_T correspond to the triangles in T and two nodes are connected when the corresponding triangles share an edge. These two graphs are illustrated in Figure 4.

It is well known that the dual graph of a triangulated simple polygon is a tree. Note that every tree has a leaf, and a leaf in the dual graph 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 the triangulation, 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

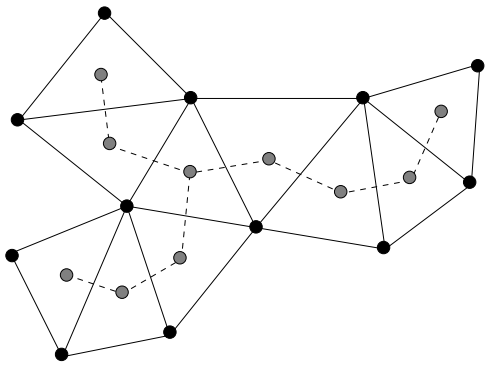


Figure 4: The graphical structure of the triangulation, G_T, is shown with black nodes and solid edges while the dual graph, D_T, is shown with gray nodes and dotted edges.

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 1. *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. In general the diagonals in a Delaunay triangulation will connect vertices that are approximately locally symmetric.

By looking at Figure 3 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. A natural way to decompose objects into parts is to split them at places where the boundary has curvature minima (see [22, 31]). This is because joining two parts together usually creates such minima. Figure 5 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.

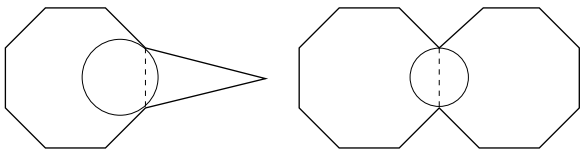


Figure 5: Limbs and pinch points are naturally represented in the CDT. These pictures show the defining circle for a diagonal that corresponds to a limb boundary and a diagonal that corresponds to a pinch point.

Figure 6: 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.

There are three possible types of triangles in a triangulated polygon, corresponding to nodes of different degrees in the dual graph. The three types are shown in Figure 6, 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 3 each finger in the hand is formed by sequences of triangles of the second type and end with a triangle of the first type.

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 7 can be bent by changing the shape of a single triangle. In the next section 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. 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 class. Most algorithms for chordal graphs rely on the

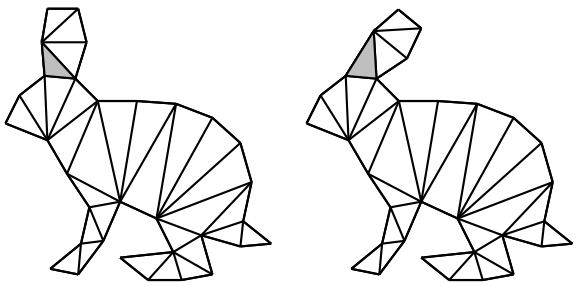


Figure 7: The rabbit ear can be bent by changing the shape of a single triangle.

following characterization. Let $G = (V, E)$ be a graph. Recall that a *clique* is a set of nodes where there is an edge between each pair of them. If S is a subset of V, then the graph *induced* by S consists of S itself and all edges in E that connect pairs of nodes in S. A vertex is called *simplicial* if its neighbors form a clique. We say that a graph is *chordal* if it admits a perfect elimination scheme as defined below.

Definition 2. *Let $G = (V, E)$ 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 $S = \{v_i, \ldots, v_n\}$.*

There are several efficient algorithms that can be used to compute a perfect elimination scheme for a chordal graph (see [17]). The graphical structure of a triangulated simple polygon belongs to a well-known subclass of chordal graphs known as 2-*trees*.

Definition 3. *A clique on $k + 1$ vertices is a k-tree. Given any k-tree G on n vertices, we can construct a k-tree on $n + 1$ vertices by adding a new vertex to G 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. A simplicial vertex in a k-tree is always in a single $k + 1$-clique. Moreover, if we delete a simplicial vertex from a k-tree on more than $k + 1$ nodes we obtain another k-tree. These and other characterizations of k-trees are described in [28] and [21].

2.2 Shape of Landmark Data

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

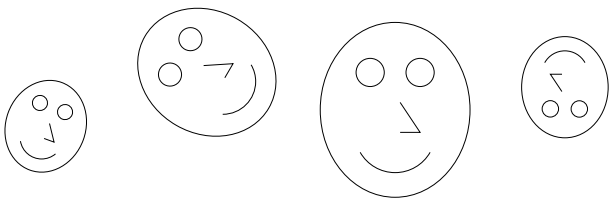


Figure 8: Different objects with the same shape.

Definition 4. 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. (i.e. two objects have the same shape if one can be translated, scaled and rotated to exactly match the other one). Figure 8 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*. We assume that landmarks are labeled so that each one corresponds to a particular object feature. In the case of a polygon we can take the vertices as the landmarks. Note that a polygon is fully determined by the location of its vertices. In general a set of landmarks provides a partial description of an object.

Definition 5. *A* configuration *is the set of landmarks on a particular object. The configuration of an object with n landmarks in m dimensions is given by a $n \times m$ 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 $\mathbb{R}^{nm}$ minus some possible singularities (we may want to exclude configurations where landmarks coincide). Let $X \sim Y$ denote that X is related to Y by a similarity transformation. The space of possible shapes is the set of equivalence classes defined by this relation. 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 describing the graph structure of the triangulation and specifying the shape of each triangle.

Let G 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 $(v_i, v_j, v_k) \in G$ to denote that three particular vertices form a triangle in G. 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 $(v_i, v_j, v_k) \in G$. Clearly, X defines a non-degenerate shape $[X_{ijk}]$ for each triangle $(v_i, v_j, v_k) \in G$. 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 1. *Given a 2-tree G, and non-degenerate shapes $s(i, j, k)$ for each triangle of G, 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 G. We know that v_i is in exactly one triangle, say with v_j and v_k. Let G' be the 2-tree obtained by deleting v_i from G, 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 G'. 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 shape for this triangle. So X is defined by X' and the shape $s(i, j, k)$. □

This result allows us to describe the shape space for a triangulated polygon in terms of shape spaces for triangles. Let M be a space of triangle shapes. There are two canonical choices for M, either Kendall's or Bookstein's space of triangles [33]. A triangulated polygon on n vertices has $n - 2$ triangles and we can take its shape space to be $M_1 \times \cdots \times M_{n-2}$, where we label the triangles in some arbitrary order. The metric structure of M induces a metric on the cross product space. Figure 7 shows how an object can be deformed by changing the shape of a single triangle. In this case the figure on the left can be seen as a point $(x_1, \ldots, x_{n-2})$ in our shape space, where the coordinate $x_i \in M_i$ parametrizes the shape of the i-the triangle. The figure on the right has the same coordinates except for one of the x_i that changed.

3 Deformable Template Matching

In this section we show how triangulated polygons can be used to detect 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 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 correct solution or are too slow for practical use. This is because the number of possible non-rigid transformations of a template is very large. In contrast, we present an algorithm that can quickly find a globally 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. This type of energy function is quite general, and can be used to represent a wide range of deformable models, including ones that capture both the boundary and the internal structure of a two dimensional 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 images and images of natural scenes, demonstrating the wide applicability of our technique.

The basic idea of matching a deformable model to an image goes back to the pictorial structures work of Fischler and Elschlager [14], the rubber mask technique of Widrow [37] and the pattern theory framework of Grenander [19]. Other important models are described in [24] and [8]. A few efficient and provably good matching algorithms have been developed for very restricted sets of models. For example, in [10] a dynamic programming algorithm was used to detect open deformable curves in images. Dynamic programming was also used in [1] to match models consisting of a number of sparse 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 [30]).

3.1 Matching

Let P be a simple polygon corresponding to an object template. An embedding of P in the image plane is defined by a continuous function $f : P \to \mathbb{R}^2$, where f is defined over both the boundary and the interior of the polygon. We consider a set of embeddings that are extensions of maps $g : V \to \mathbb{R}^2$, 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 a triangle $(g(v_1), g(v_2), g(v_3))$ in the image 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 if the following form:

$$E(g, I) = \sum_{(v_i, v_j, v_k) \in T} c_{ijk}(g(v_i), g(v_j), g(v_k), I). \tag{1}$$

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 section we use a simple energy function similar to other 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 below.

While our implementation 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 Section 4 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.

3.2 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 global 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 a measure of how far h is from a similarity transform. This quantity has also been used by Bookstein [7] to measure distance between triangle shapes. We use the log-anisotropy measure to assign a deformation cost for each affine map (and let the cost be infinity if the 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 this term by $\|f'(s)\|$ to make the energy scale invariant. The integral can be broken up into an integral for each boundary edge in the polygon. This allows us to write the energy function in the form of equation (1), where the cost for each triangle will be a deformation cost plus one integral term for each boundary edge that belongs to the triangle.

Note that for a deformation cost $\mathrm{def}(f_t)$ to be zero, f_t must be a similarity transformation. Thus the global map f has zero deformation cost exactly when it preserves the shape of each triangle. In this case Theorem 1 implies that the shape of the object does not change and f must be a global similarity transformation. If each f_t is close to a similarity transformation the global deformation cost will be small. In this case the shape of the object is preserved locally but the overall shape can change quite a bit. For example, an elongated branch of a triangulated polygon could twist into a spiral even if each triangle shape does not change much. Sometimes this is a good property, but other times it could be a problem. Since the energy function is a sum of costs per triangle we can not explicitly capture relationships between distant parts of an object. For example we can not enforce symmetries between different branches of a deformable object.

3.3 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 60×60 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 $|\mathcal{G}|^n$. Our matching algorithm finds an optimal map in time $O(n|\mathcal{G}|^3)$, which is exponentially better than just trying all possible maps.

The algorithm uses a technique known as non-serial dynamic programming (see [4] and [1]). Typical applications of dynamic programming relies on a chain structure. Non-serial dynamic programming generalizes the standard technique to certain problems defined on decomposable graphs that do not have large cliques. As described in Section 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 9 shows a triangulated polygon with vertices labeled by their order in a perfect elimination scheme. Such an order can be computed in time linear in the number of polygon vertices using one of the algorithms in [17]. Note that in general there are several valid elimination orders and the matching algorithm described here works with any of them.

The algorithm works by sequentially eliminating the vertices and triangles of a triangulated polygon. As an illustration lets consider how we would eliminate v_1 when matching the example in Figure 9 to an image. This vertex is in a single triangle (with v_2 and v_{10}), so its location in the image, $g(v_1)$, contributes to a single term in an energy function of the form in equation (1). Thus we can compute an optimal location for v_1 as a function of a pair of particular locations for v_2 and v_{10}. Now the quality of the best location for v_1 can be associated with the placements of v_2 and v_{10}. At this point we remove v_1 and its triangle from the model and solve the matching problem for a smaller triangulated polygon. Intuitively the energy function is updated so that the cost for the triangle (v_1, v_2, v_{10}) is taken into account with the placement of v_2 and v_{10} alone.

Let $(v_1, \ldots, v_n)$ be a perfect elimination scheme for the vertices of the triangulated polygon. 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 $V[j,k](l_j, l_k)$. At this point we can associate the costs

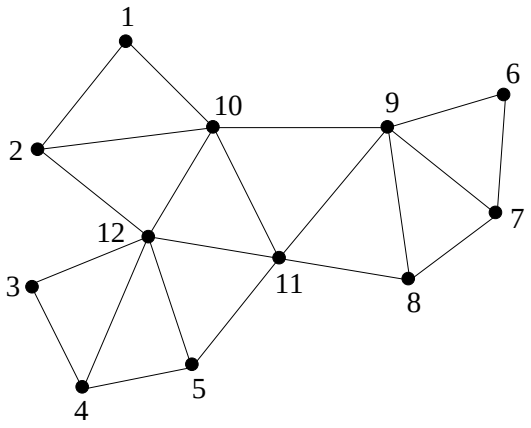


Figure 9: Perfect elimination scheme for the vertices of a triangulated simple polygon.

in $V[j,k]$ with the placement of vertices v_j and v_k and forget about vertex v_i. When we get to the last two vertices we can solve for their best locations and trace back to find the best locations of the other vertices, as is typical in dynamic programming. Pseudocode for this procedure is show in in Algorithm 1.

Algorithm *Match*(I)

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.

Algorithm 1: Find the best embedding of a shape in an image.

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 heuristic the running time of the algorithm is essentially $O(nm^2)$.

Note that in line 7 of the algorithm each entry in $V[n-1,n]$ gives the cost 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 an image by finding local minima in $V[n-1,n]$. We simply trace back from each local minima that has value below a fixed threshold.

3.4 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 computed a Delaunay triangulation 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 when running on a 1Ghz Pentium III machine.

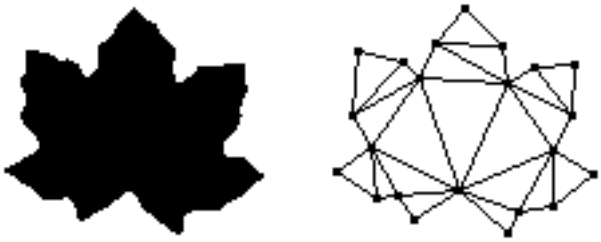
Figure 10(a) shows a model for the corpus callosum generated from a manually segmented brain MRI. The best match of the model to several images is shown in Figure 11. Note how these images have very low contrast, and the shape of the corpus callosum varies considerably across them. We are able to reliably 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 [26]). The main advantage of our method is that it does not require any initialization.

Figure 10(b) 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 12. 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 automatically "hallucinate" the location of a large occluded part of the leaf.

Since our models are invariant to similarity transformations we can detect the target object independent of of its position, scale and orientation. Figure 13(a) demonstrates detections at very different scales. In Figure 13(b) we demonstrate how we can detect multiple instances of an object in an image. As discussed in the last section we simply selected local minima in $V[n-1,n]$ with value below a pre-determined threshold to generate each detection.

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 14 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

(a) corpus callosum



(b) maple leaf

Figure 10: Models for the corpus callosum (a) and maple leaves (b) generated from binary pictures (see text for description).

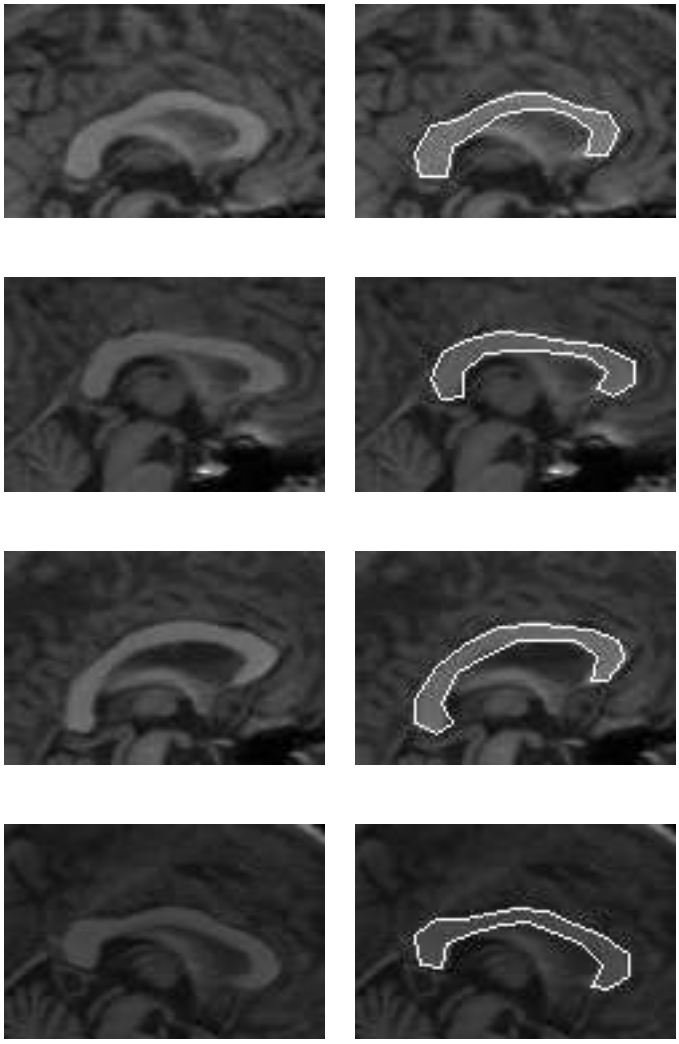


Figure 11: Matching the corpus callosum model to different images.

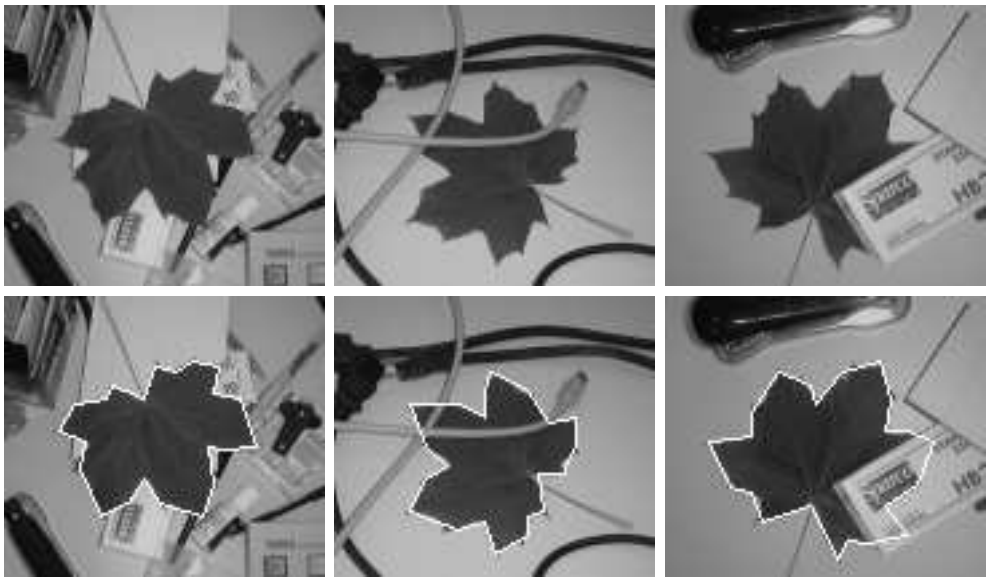


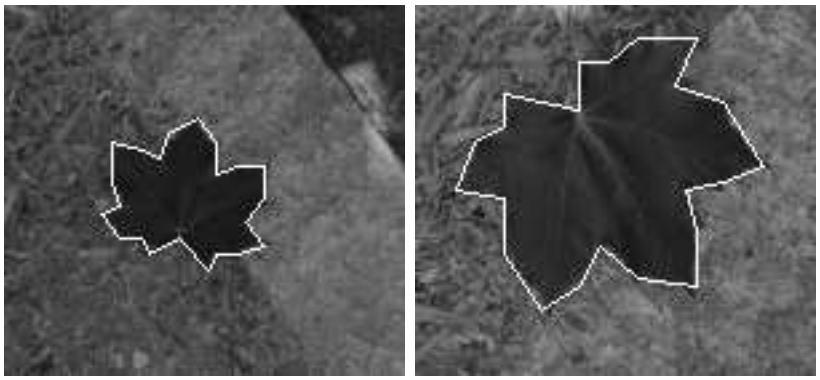
Figure 12: Matching the maple leaf model to different images.

when it is barely visible.

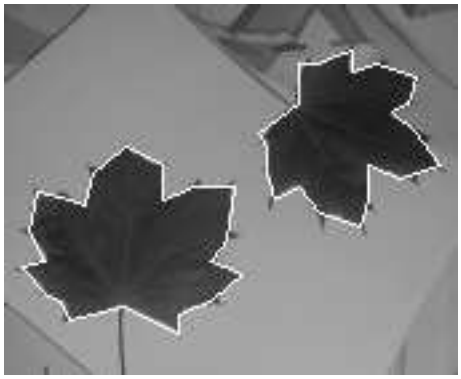
Our matching algorithm performs well in situations where local search techniques tend to fail. To illustrate this we used a public implementation of a local search method known as active appearance models [34]. Every local search technique depends on initialization, so we show results of matching using different initialization parameters on a fixed image. Figure 15 illustrates typical results obtained with active appearance models. It is clear that good initialization parameters are necessary to obtain a good match. This experiment illustrates the advantage of global methods.

4 Learning Deformable Template Models

Now we consider how to learn a deformable shape model for a class of objects 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 model should have an ideal shape and a parameter that controls how much it is allowed to deform. Figure 2 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. Below we describe how the shape detection problem can be cast in a statistical



(a)



(b)

Figure 13: Our models are invariant to similarity transformations so we can detect deformable objects at arbitrary scales, positions and orientations (a). Detection of multiple objects in one image is illustrated in (b).

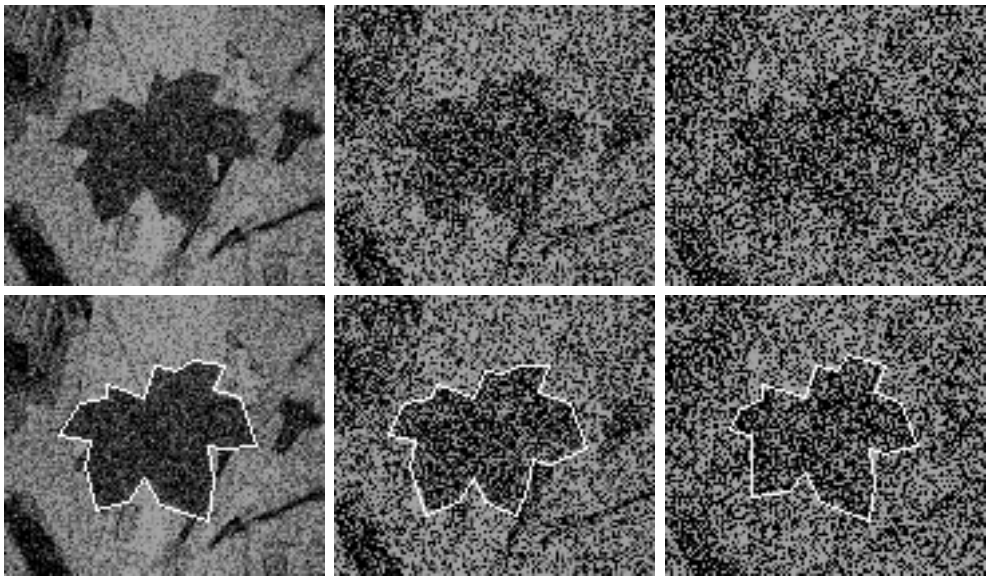


Figure 14: Matching to an image corrupted by increasing amounts noise.

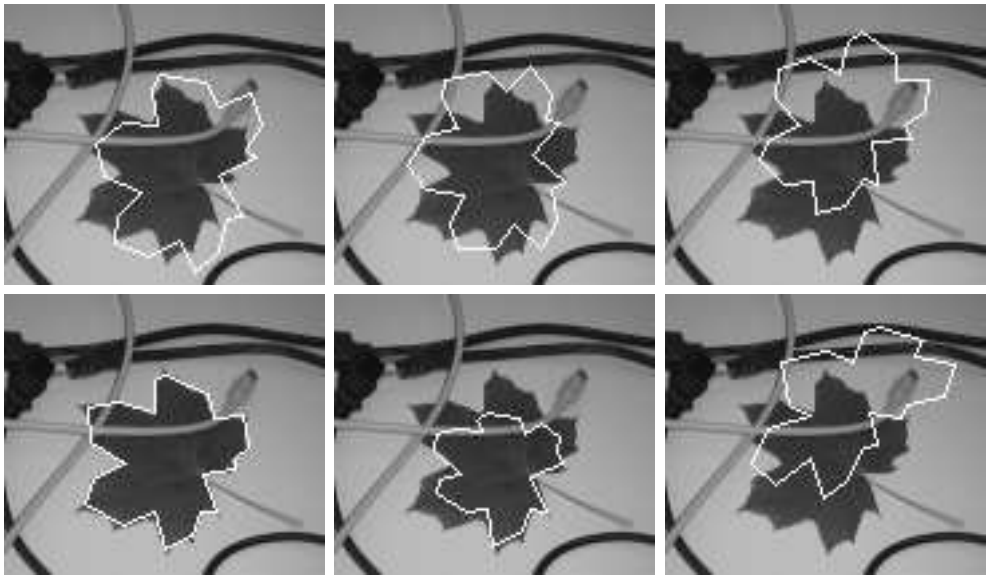


Figure 15: Results of a local search method with different initialization parameters. The first row shows the initialization and the second row shows the resulting match in each case.

setting, this allows us to derive the learning procedure in terms of maximum likelihood estimation.

There are other techniques for learning the typical shape variation among objects in a population. Our models are unique in that they can characterize objects that deform by large amounts. Moreover, we concentrate on models that can be used by the efficient matching algorithm described in the last section.

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 look for a location for the object that has highest probability of being its true position. Using Bayes' law, the optimal location 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 section 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 3.2,

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

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 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 examples of the object. This will allow us to have more specific deformation

models. By restricting $\hat{p}(X)$ to a particular form we can use the learned model to detect shapes in images using our matching algorithm.

4.2 Procrustes Analysis

Before describing our learning technique we review a standard method for estimating the mean shape and typical variation of landmark data known as generalized Procrustes analysis. See [12] or [18] for more details.

Say we have a set of random configurations $\{X^1, \ldots, X^m\}$ obtained from a mean μ by a small perturbation and a similarity transformation, $X^i \sim \mu + \epsilon^i$. Assuming the perturbations are independent and distributed according to a spherical Gaussian distribution a maximum likelihood estimate of the mean can be obtained as follows,

$$\hat{\mu} = \operatorname*{argmin}_{||\mu||=1} \min_{Y^i \sim X^i} \sum_{i=1}^{m} ||\mu - Y^i||^2.$$

We can efficiently compute $\hat{\mu}$ by solving a complex eigenvector problem. The optimal alignment, Y^i, between each configuration and the mean can also be computed efficiently. For each configuration we define the Procrustes residuals by $R^i = Y^i - \hat{\mu}$. Detailed analysis of the residuals can be performed using principal component analysis as discussed in [8]. The sum of squares of the residuals gives an estimate of the overall shape variability,

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

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 triangulated models.

4.3 Triangulated Models

We represent an object using a 2-tree over its vertices, and assume that the shapes of the triangles in the model are independent. Theorem 1 implies that this makes sense. Note that we only assume independence between the *shapes* of the triangles. Their configurations are dependent as certain triangles share vertices.

Recall that our matching algorithm can minimize energy functions that are sums of costs, one for each triangle in a 2-tree. As discussed above the energy function can be seen as the

negative logarithm of the posterior distribution $p(X|I) \propto p(I|X)p(X)$. The assumption that the shapes of the triangles in the model are independent ensures that the prior, $p(X)$, is in the right form. At a minimum, the image likelihood, $p(I|X)$, should depend on the boundary of the object. This implies that the 2-tree we choose for the model should include the edges corresponding to the object boundary. If we want the likelihood to depend on the interior of the object we need the 2-tree to be a planar triangulation of each example.

Let $\{X^1, \ldots, X^m\}$ be a set of configurations for a polygon on n vertices and let T be a 2-tree over the polygon vertices. Our prior over configurations is given by,

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

where X_{ijk} is the sub-configuration of X containing landmarks i, j and k. When T is fixed we can obtain an estimate for the prior, $\hat{p}_T(X)$, by using generalized Procrustes analysis separately for each triangle in the model. That is, $\hat{p}_{ijk}(X_{ijk})$ is estimated from the sub-configurations $\{X^1_{ijk}, \ldots, X^m_{ijk}\}$. The Procrustes residuals give a measure of how much each triangle in the model deforms across the different examples. 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 global similarity transformations. The alignment only needs to be relatively accurate for each set of corresponding triangles in the examples. Intuitively, our models makes sense under an assumption that objects are *locally almost rigid*.

Now we consider the case where no 2-tree is given a priori. In this case 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 the edges in the object boundary. The following theorem makes it possible to efficiently select an optimal graph.

Theorem 2. *Graphs in $\mathcal{T}$ correspond to triangulations of a convex n-gon with boundary vertices sequentially 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 $(v_1, \ldots, v_n, v_1)$, and we know that any triangulation of a simple polygon is a 2-tree.

Now suppose we have a convex n-gon and 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 v_i be a simplicial vertex of T. This vertex must be connected to v_{i-1} and v_{i+1} by the cycle condition. Since v_i is simplicial, it is not connected to any other vertices, and v_{i-1} is connected to v_{i+1}.

Removing v_i from T we obtain a 2-tree T' over the remaining $n-1$ vertices, with the cycle $(v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ in T'. By induction, T' is a triangulation of the convex $(n-1)$-gon induced by the remaining vertices. Adding the triangle (v_{i-1}, v_i, v_{i+1}) back we obtain a triangulation of the original convex n-gon. □

This correspondence implies that when searching for the graphical structure of the model we can search over triangulations of a convex n-gon. A well known algorithm (see [9]) can be used to find an optimal triangulation for a convex polygon, where the cost of the triangulation is a sum of arbitrary costs, one for each triangle. 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 2-tree defining the model structure is,

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

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,

$$\hat{T} = \operatorname*{argmin}_{T \in \mathcal{T}} \sum_{(v_i, v_j, v_k) \in T} c_{ijk},$$

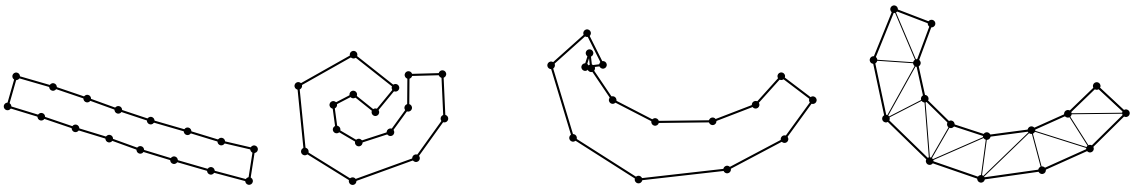
where,

$$c_{ijk} = -\sum_{l=1}^{m} \log \hat{p}_{ijk}(X^l_{ijk}).$$

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 model tends to concentrate all of the variation in shape to as few triangles as possible. The learning procedure will pick 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. This would cause the optimal triangulation with the modified c_{ijk} to have infinite cost.

We do not have to use Procrustes analysis to estimate the shape prior of each triangle in the model. Whatever form we choose for the prior distribution of a triangle shape, we just



(a) examples (b) procrustes mean (c) our model

Figure 16: Comparing the procrustes mean and triangulated model for the two input polygons shown in (a). Procrustes analysis breaks down when the samples can not be aligned using similarity transformations (b), while our model is good (c).

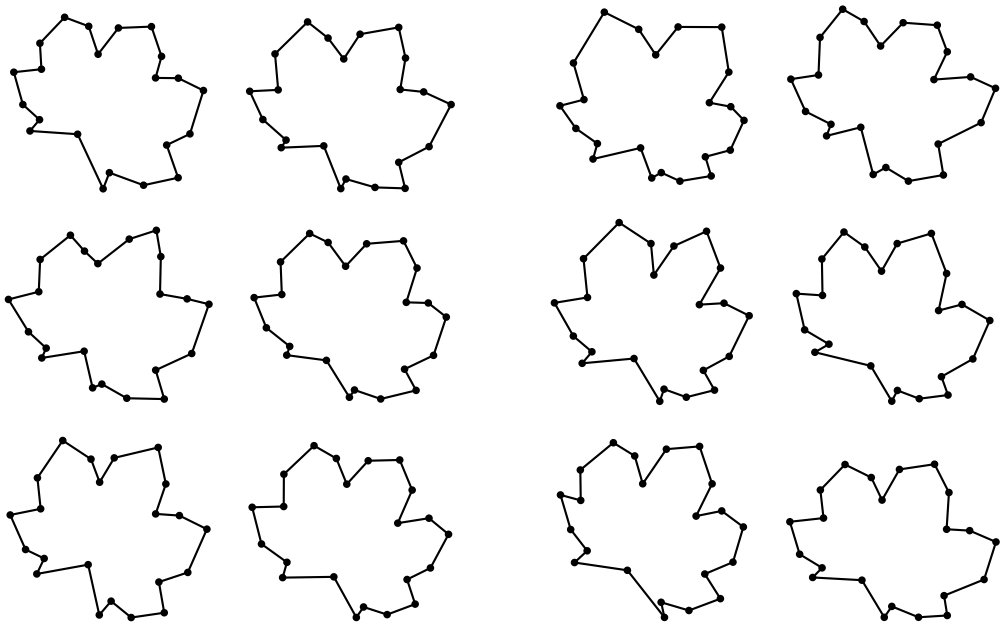
need to be able to estimate $\hat{p}_{ijk}$ and compute the costs c_{ijk} accordingly. Then the optimal triangulation is selected with the same algorithm. Note that the optimal triangulation may be different for different choices of priors. A number of different shape priors for landmark data that could be used here are described in [12].

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 16. 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.

We trained a model to capture the shapes of maple leaves. In this case we used 34 example pictures, each annotated with the location of landmarks corresponding to curvature extrema along the boundary of the leaves. Some examples of the objects in this data set are shown in Figure 17(a). To see how well the learned model captures the typical shape variation among the leaves we can generate random samples from $\hat{p}(X)$. Some of the random samples are shown in Figure 17(b). Note how the shape variation among the samples is similar to the variability present in the training set.

We also considered a more challenging problem, of modeling the shape of a set of hands. In this case we used a database with 40 pictures of hands from [35] as the input to the learning procedure. Each picture is annotated with the location of 56 landmarks along the boundary of the hand. Some examples of the objects in this data set are shown in Figure 18(a), while Figure 18(b) shows random samples from the learned prior. The random samples capture



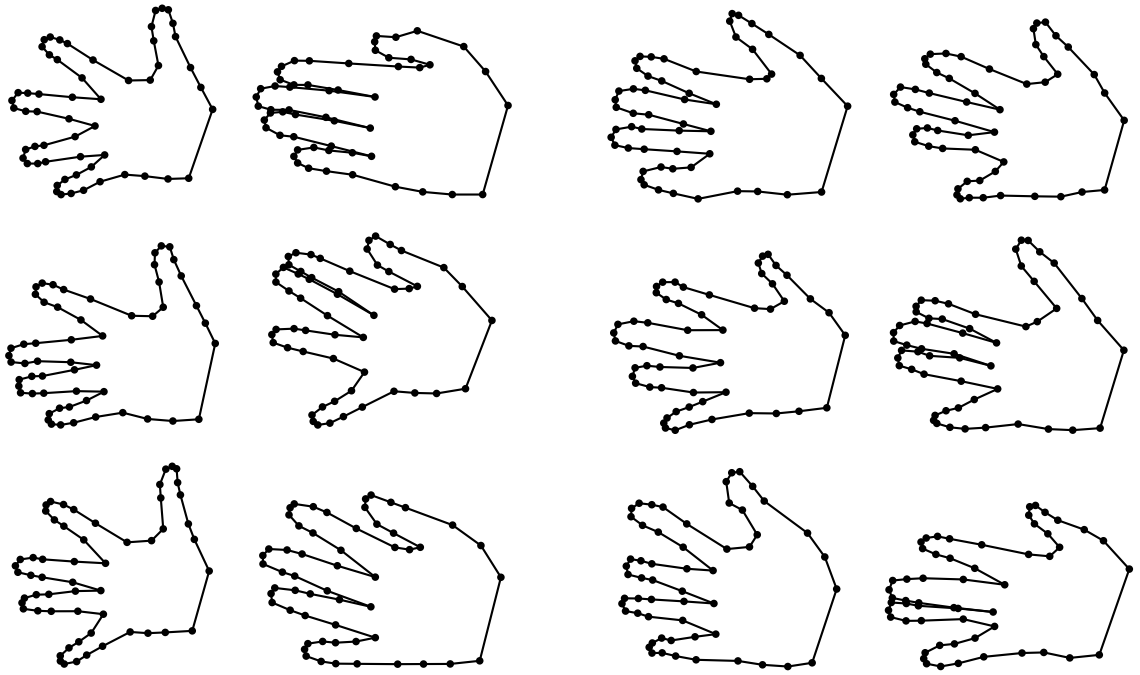
(a) examples (b) samples

Figure 17: A few of a total of 34 example leaves are shown in (a). Random samples from the learned model are shown in (b).

the typical shape variability for hands reasonably well, but the prior can not enforce that the relative lengths and thicknesses of the fingers be perfect because of the assumption that the shape of each triangle in the model is independent.

5 Conclusion

In this paper we described how triangulated polygons can be used to represent two dimensional shapes. This representation is closely related to the medial axis transform and provides a natural decomposition of 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 detect non-rigid objects in images. Our detection algorithm efficiently computes a global optimal solution to the deformable template matching problem. 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



(a) examples (b) samples

Figure 18: A few of a total of 40 example hands are shown in (a). Random samples from the learned model are shown in (b).

by the detection algorithm.

Acknowledgment

This work was done while the author was a graduate student at the Massachusetts Institute of Technology. The author would like to thank his advisor W. Eric L. Grimson for many helpful discussions.

References

[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] F. L. Bookstein. Size and shape spaces for landmark data in two dimensions. *Statistical Science*, 1(2), May 1986.

[8] 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.

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

[10] 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.

[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] 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.

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

[15] D. Geiger, A. Gupta, L.A. Costa, and J. Vlontzos. Dynamic-programming for detecting, tracking, and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):294–302, 1995.

[16] D. Geiger, T.L. Liu, and R.V. Kohn. Representation and self-similarity of shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):86–99, 2003.

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

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

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

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

[21] F. Harary and E.M. Palmer. On acyclic simplicial complexes. *Mathematika*, 15(1):115–122, 1968.

[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] 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.

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

[26] 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.

[27] 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.

[28] D. J. Rose. On simple characterizations of k-trees. *Discrete Mathematics*, 7(3-4):317–322, 1974.

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

[30] 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.

[31] 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.

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

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

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

[35] M. B. Stegmann and D. D. Gomez. A brief introduction to statistical shape analysys. Technical report, Informatiocs and Mathematical Modelling, Technical University of Denmark, DTU, March 2002.

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

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

[38] 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.