The Role of Saliency and Error Propagation in Visual Object Recognition

by

Tao Daniel Alter

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

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1995

Author .... [illegible] ....
Department of Electrical Engineering and Computer Science
May 26, 1995

Certified by .... [illegible] ....
W. Eric L. Grimson
Professor
Thesis Supervisor

Accepted by .... [illegible] ....
Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

The Role of Saliency and Error Propagation in Visual Object Recognition

by

Tao Daniel Alter

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 1995, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Man-made environments typically contain many objects with well-defined boundary edges, such as staplers, tables, and computers. These edges may determine the shapes of the objects and are invaluable in their identification. This thesis is about using such edge information for recognizing three-dimensional objects in images. In particular, we consider an approach that first identifies image features that are likely to come from a single object, and later uses these features to project similar model features into the image for verification. In light of this approach, we focus on two critical problems: identifying salient curves in images and the propagation of error from image features to projected model features.

In our study of saliency, we analyze one of the notable approaches to the problem, Shashua and Ullman's Saliency Network, and propose a new method based on shortest-paths algorithms. The new method is designed to overcome some of the most critical problems in the network. Using shortest-paths methods, we find the optimal curves in the image according to a measure that prefers contours that are long, smooth, and connected.

For the verification stage, we consider how to make the verification robust to locational errors in the image features. Error in the image features leads to uncertainty in the projected model features, which deters effective recognition. We show how error propagates when poses are based on three pairs of model and image points, for both Gaussian and bounded error in the detection of image points, and for both scaled-orthographic and perspective projection models. This result applies to objects that are fully three-dimensional, where past results considered only two-dimensional objects. In addition, we show how we can utilize *linear programming* to compute the propagated error region for any number of initial matches. Finally, we use these results to extend, from two-dimensional to three-dimensional objects, robust implementations of *alignment*, *interpretation-tree search*, and *transformation clustering.*

Thesis Supervisor: W. Eric L. Grimson
Title: Professor

Acknowledgments

I wish to thank the people who have enriched my life the most over my years at MIT, in particular, my deeply supportive and loving parents Ronald and Arlene Alter, my wonderful brothers Robin and Roy, my collaborators Ronen Basri and David Jacobs, my long-time officemate, Kah Kay Sung, and not least, my encouraging advisor, Eric Grimson. I have been privileged over my years at the Artificial Intelligence Lab to have had the nicest professor in the Lab as an advisor and the nicest graduate student in the Lab as an officemate.

I would also like to mention my friends whom I know from the AI Lab, and with whom I have greatly enjoyed interacting while working my Ph. D., namely, Ronen Basri, David Beymer, Todd Cass, John Harris, Scott Hofmeister, Charles Isbell, David Jacobs, Tina Kapur, Lily Lee, Stephen Lines, Pam Lipson, Liana Lorigo, Sundar Narasimhan, Jose Robles, Pawan Sinha, Kah Kay Sung, Paul Viola, Sandy Wells, and Steve White. Furthermore, I cherish the time I have spent with my good friends, Ronen Basri, David Jacobs, Scott Hofmeister, and Amy Yacus. Finally, for serving on my thesis commitee, I am grateful to Eric Grimson, Ronen Basri, Berthold Horn, David Jacobs, and Tomás Lozano-Pérez.

Contents

[1]This section is joint work between the author and David Jacobs.
[2]This chapter is joint work between the author and Ronen Basri.

[3]This chapter is joint work between the author and Ronen Basri.
[4]This chapter is joint work between the author and David Jacobs.

Chapter 1

Introduction

Enabling machines to recognize objects visually is a fundamental goal of the field of Artificial Intelligence. The foremost justification for this goal is that visual recognition is needed for robots to find and manipulate objects in their environment. Another, equally important, reason is that machines which recognize objects would be of great value in a wide variety of human tasks. For instance, a computerized camera that can recognize people could aid a concierge in security. In medical domains, it is often desired to automate the demarcation of anatomical structures in images. As an example, a system that could automatically select out the contour of the brain in a sequence of MRI images could be used to build a 3D model of the brain, which could be used to render any view to a surgeon planning an operation. In automobiles, computer vision systems could give drivers the option to be warned of oncoming cars or objects entering their path. Another potentially valuable application is inspecting manufactured parts for defects.

There are many factors that make the problem of building computers that recognize objects difficult. First, pictures of three-dimensional objects can appear very different when the objects are seen from different viewpoints or under different lighting conditions (e.g., a picture of a room can look different when illuminated by ceiling lights or by sunlight through a window). Also, other objects in the scene can partially occlude an object of interest, or can cast shadows onto it, or can reflect light onto it. Finally, it can be especially difficult to recognize instances of classes of objects, such as identifying any instance of a telephone (consider describing how to recognize a telephone when a phone can be anything from a standard push-button office phone to an old-style rotary phone in which the microphone is part of the base instead of the handset). In recognizing objects in the face of all these factors, humans are able to incorporate many sources of information, such as an object's shape, color, texture, and even function.

1.1 Framework

In this work we focus on using information about an object's *shape*, and we do so in the context of recognizing particular objects, not classes. Even though we are limited to particular objects, we may be given a large library of such objects and the task, then, is to rapidly identify any instance of an object in the library. The form of input we assume is two-dimensional grey-scale images.

1.1.1 Geometric models

Representation is necessarily a key issue in any recognition scheme. We need to represent how objects can appear from any viewpoint. The approach we adapt is known as *model-based.* In model-based schemes, we consider three-dimensional objects that are rigid and contain well-defined edges, such as staplers, tables, and computers. The edges are then represented by geometric primitives, such as line segments and points. Line segments may represent straight edges, and points may correspond to inflection points on curved edges or to corners (e.g., Fig. 1-1).

The model-based representation is popular and has become so primarily for three reasons. First, the features used frequently are *local,* which means they are insensitive to occlusion. Non-local, region-based features such as moments [27, 74], skeletons [85, 86], and geons [11] are difficult to obtain if regions can be partially occluded. Even though model-based representations may include such global features, in this work we are interested in features that can be recovered locally. For a line segment, its orientation is robust to occlusion, and for a point, positional information is always

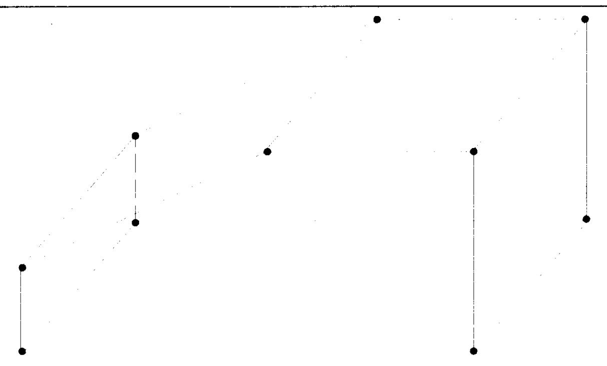


Figure 1-1: Model of a telephone.

available.

Another reason the model-based approach is popular is that the features are *invariant* to projection. For instance, a straight edge always projects to a straight edge, an inflection point to an inflection point, and a corner to a corner. This representation allows recognition schemes to focus on matching invariant features between a model and an image, instead of matching every location.

A third motivation for the model-based approach is that geometric constraints can be brought to bear. The constraints arise from the rigidity of an object and a projection model, which is a mathematical model of how a three-dimensional object appears when projected into an image. Rigidity and projection determine the set of images that an object can produce as the viewpoint changes, and any match between a set of two-dimensional image features and three-dimensional model features must be consistent with a viewpoint.

1.1.2 Model-based recognition and error propagation

In model-based recognition, the basic problem is to find correspondences between model and image features. Specifically, we suppose we are given a 3D model composed of geometric features, and an image of a scene which may contain the modeled object, along with occlusion and clutter. We run the image through a feature detector to get a set of 2D features that are similar to the 3D features in the model. The problem is to find which image features, if any, correspond to model features and to exactly which model features they correspond.

Model-based systems commonly perform recognition by searching the space of all possible correspondences between model and image features. This is illustrated in Fig. 1-2, which shows a tree where every path from the root is a sequence of mutually-consistent pairings of model and data (image) features. By "consistent," we mean that there is a view of the model from which the model features in the pairings will project to their corresponding image features, up to error in the data features. One approach to recognition, called *constrained search*, performs a depth-first search of this tree, backtracking whenever an inconsistent set of pairings is found (e.g., [13, 32, 71, 39, 40, 46]).

Another recognition approach, which is particularly pertinent to this thesis, only goes down to a short depth in the tree and then uses the current set of correspondences to *align* the model with the image for immediate verification (e.g., [77, 22, 31, 64, 5, 52]). This method has become known as *alignment* [52], and could potentially work much faster than constrained search, because a relatively small portion of the search space is explored. The approach is justified by two main ideas. First, a small number of correspondences generally determines the pose (viewpoint) of the model up to the error in the data features. For example, three corresponding 2D image and 3D model points are needed to compute the viewpoint from which the 3D points were seen [31, 52]. (The fact that only three point correspondences are necessary illustrates the value of geometric constraints.) Once the viewpoint is computed, alignment is performed by using the viewpoint to project the model into the image.

After projection, the model features in general will not lie exactly on their corresponding image features, due to errors in feature detection and to inaccuracies in modeling the imaging process. The verification step consists of looking near each projected model feature for a confirming image feature, and then deciding whether the model is present based on some measure of similarity between the predicted and confirming features. Note that this form of verification does not guarantee that the newly discovered matches between model and image features will be consistent with a single view of the model. In fact, this inconsistency brings up the second main idea

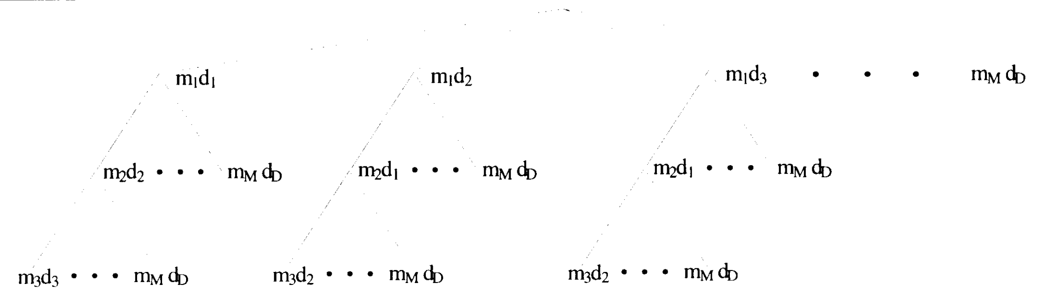


Figure 1-2: Searching correspondence space. Each node of the tree defines a consistent set of correspondences between model features m_i, $i = 1, \ldots, M$, and data features d_i, $i = 1, \ldots, D$. Each node's correspondences are listed on the path from the node to the root. Nodes that would correspond to inconsistent sets are first pruned. For point features, all possible nodes would be consistent up to depth three. The tree shows the case where every model and data feature is matched at most once in a set, which is appropriate for point features. If desired, for line features the tree could be generated to allow more than one data feature to match a single model feature.

of alignment, namely, that there is enough redundancy in the partial constraints on the projected model features that the recognition will still be accurate.

To date, many questions remain concerning the performance of model-based recognition systems. For example, although we know the minimal number of features needed to generate a model pose, we do not know how accurate the pose must be to allow us to identify the object. In addition, some authors stress the importance of using a minimal set of features [31, 52], while others contend that this will not produce a sufficiently accurate pose for recognition [64]. It is in general difficult to characterize the conditions under which these systems will succeed or fail, or to evaluate the relative effectiveness of the different strategies for recognition, or to understand the extent to which each approach makes the best possible use of the information available. A careful understanding of the effects of sensing error is a prerequisite to doing all of these.

1.1.3 Grouping and saliency

Another basic issue concerning the performance of model-based systems is efficiency. In order to find a correct set of correspondences, systems that rely solely on geometric constraints are faced with a matching problem that is inherently expensive. When error is carefully controlled, the fastest algorithms require time that is high polynomial in the numbers of model and image features. Specifically, the required time is $O(m^8n^8)$, where m is the number of model features and n is the number of image features [17, 21]. Even if error is not carefully controlled, a system would still need to consider all minimally-sized hypotheses (sets of correspondences) between model and image features. For point features, there are $3!\binom{m}{3}\binom{n}{3}$ such minimal sets. Typically, m could be 50 and n could be 500, which would give about 10^{12} hypotheses to try.

Not surprisingly, many authors have argued that for recognition of this form to be feasible, *grouping* is necessary as a precursor to the matching process (e.g., Lowe [64], Grimson [33], Jacobs [55]). The purpose of grouping is to identify subsets of image features that are likely to come from a single object, without reference to a model database. The idea is that images of modeled objects tend to have certain properties in common. For instance, the edges on man-made objects tend to be smooth (or piecewise-smooth) and connected. The regions formed by the edges tend to be convex and closed. And straight edges often are parallel or co-terminate. These properties of edges carry through to their projections.

The simplest form of grouping is "local grouping," which collects edge features that are nearby in the image. Examples of local groups include points from the same curve segment [51] and straight edges that form corners [88] or three-line junctions [46]. A difficulty with such methods is that local groups tend to arise accidentally, and so false-positive groups of this kind can be ubiquitous in an image.

More global grouping criteria are needed to reduce the number of false-positive groups. Such criteria include convexity, parallelism, symmetry, and global smoothness. The basic difficulty with more global approaches is that there are an exponential number of ways to form groups from the image edges. One approach is to perform a backtracking, constrained search through all of the edges [47, 26, 55]. The danger, however, is that either the search time will be prohibitive or else the constraints will be so strong that the discovered curves may not be near optimal.

On the other hand, there may be certain curves which appear to stand out from

Figure 1-3: The circle and blob are apparent although their constituent elements may not be locally salient. The blob image is from [65].

the rest of the image, and these *salient* curves may correspond to objects of interest. In Fig. 1-3-left, a circle stands out amidst a background of random dots. If the circle boundary is observed locally, it is often unclear exactly which points lie on the circle. Nevertheless the global structure is apparent. Similarly, in Fig. 1-3-right a large blob stands out although the blob's boundary is not salient locally. In general, the problem of saliency is to identify structures that "pop out." This includes the global saliency in Fig. 1-3, and it also includes "odd-man out" problems, such as spotting a blue dot in a field of red dots or a circle amidst a background of squares.

In this work, we concern ourselves with global saliency, because we believe it to be particularly useful in our domain. That is, we are interested in identifying man-made objects in images that contain substantial amounts of noise and scene clutter. In these images, the clutter commonly is due to events like surface markings, textures, and specularities. The edges these events introduce are often relatively short and somewhat random in distribution. A curve through these edges easily can contain many gaps and inflections. In contrast, the objects of interest tend to be relatively large, and, as mentioned above, their edge contours tend to be smooth and connected. A process that can identify such curves rapidly could inform a grouping system where to focus its attention.

1.1.4 Overall approach

The work in this thesis follows a certain view for building a system that could recognize three-dimensional objects from two-dimensional images. In this view, first grouping is used to identify features that are likely to come from a single object.

Then these features are matched to similar features in a model. When the groups of image features are larger than minimal sets, then some form of indexing (table look-up) would be used to quickly access those model groups that could produce the image groups. For each image group, the indexing step provides a set of candidate hypotheses, each of which is a set of correspondences between image and model features. After the indexing step, each hypothesis is aligned with (i.e., projected into) the image for verification. The verification stage proceeds in two phases, one that uses the remaining invariant model features, and another that uses a more detailed 3D representation of the object (say, a point-by-point listing of all of the object's edge contours). The first phase would serve as a quick filter by checking consistency with the invariant features. In the second phase, an extensive verification would be performed by comparing the object's contours against the image.

1.2 Contributions

This thesis concentrates on advancing two areas which are critical to this grouping-plus-alignment approach: identifying salient curves in images and robust verification by controlling propagated uncertainty. As was indicated above, these are both significant pieces of the problem which are not well understood. These pieces should be further developed in order to build in the future a complete recognition system that successfully demonstrates the approach.

1.2.1 Robust verification

For robust verification, we analyze how sensing error propagates from image features through the computations performed by model-based systems. In model-based systems, matches between model features and sensed image features typically are used to compute a model pose and then project the unmatched model features into the image. The error in the image features results in uncertainty in the predicted image locations of the remaining model features. We call the sets of possible image locations the *uncertainty regions* of the model features. Using bounded and Gaussian error models, we derive either bounds on these regions or probability distributions on them, depending on our model of error.

Knowing the propagated regions of uncertainty for bounded error allows one to build a system which can guarantee that it will find the model if it is there, as long as the input data falls within the error bounds. This option of "no false negatives" is important for reliability. In particular, such a system would accept as verified any model feature for which at least one image feature falls into its uncertainty region.

On the other hand, such a system would produce many false positives (i.e., incorrect identifications of the model). In general we would prefer a method which could pick out those hypotheses that are more likely. For instance, if an image feature arises inside a larger uncertainty region, then the feature is more likely to have come from some other object. As another example, another source of information for judging the likelihood of a hypothesis is the total number of image features. The more spurious features there are in the image, the more likely it is that some lie inside the uncertainty regions.

Yet another piece of information which we may have available is the distribution of error in the image features. Bounded error is a "no knowledge" assumption, and could be used if one suspects the true distributions may be significantly skewed from Gaussian. But if the error is believed to be Gaussian, then each candidate image feature could be ranked according to how likely the model feature was of producing it. In fact, Sarachik and Grimson [80] gave a method which aggregates such rankings into a single measure. Using the measure, their method decides whether to accept or reject the hypothesis based on a pre-determined, formally chosen threshold.

This thesis proposes an alternative method for deciding whether to accept or reject a hypothesis that is formally grounded in the error model. The method is much simpler mathematically than Sarachik and Grimson's, and it uses no pre-determined thresholds. To avoid pre-set thresholds, the method decides *on-line* whether to accept or reject a hypothesis, and does so using the likelihood-ratio test. The decision criterion incorporates the available information about the sizes of the uncertainty regions, the total numbers of model and data features, and the numbers of data features that arise in the individual regions. In addition, the method applies to both bounded and Gaussian error models.

1.2.2 Saliency

In the area of saliency, we consider two methods for identifying image curves that are globally salient, an existing method and a new method proposed here. For evaluating the methods, there are four fundamental criteria we wish a saliency mechanism to satisfy: (1) It should be able to find curves that are salient globally in complex images (images with noise, clutter in the background, and multiple objects), (2) the time required to identify a salient curve generally should be on the order of the length of curve, or at least not much more than the length (if the curve is barely salient, then more time is permissible), (3) the salient curves identified should be long, smooth, connected (contain few gaps), and closed, and (4) curve selection should be invariant to similarity transformations (rotation, translation, and scale). The last criterion is for general consistency with human perception. For example, if the mechanism chooses one curve over another as more salient, and then the two curves are uniformly scaled in size, the mechanism's choice should not change.

Our notion of global saliency is borrowed from Shashua and Ullman [82], who first demonstrated how saliency could be computed for structures like the circle and the blob in Fig. 1-3. They defined a network of locally connected elements on top of an image. Using dynamic programming, they optimized a chosen measure of saliency at every element in the network. For each element the optimization was performed over all possible curves passing through the element.

Our study of saliency begins with an analysis of Shashua and Ullman's "Saliency Network." This analysis is the first of its kind and is of particular interest because Shashua and Ullman's network has attracted much attention. Although Shashua and Ullman's method is elegant and inspirational, we show it has a number of drawbacks. Among these, the measure-of-saliency's preferences of curves are not scale invariant (criterion 4). In addition, the network may traverse the boundary of a closed curve many times before it converges (criterion 2); specifically the time to process a closed

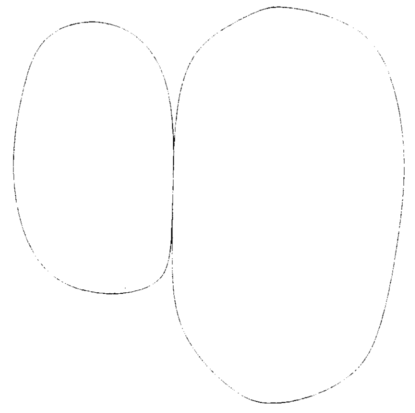


Figure 1-4: In Shashua and Ullman's method each element of a curve chooses one neighboring element with which to combine. Consequently the shared portion must choose between the two shapes.

curve can be on the order of the square of its length, even if the curve is alone in the image.

Another potential drawback Shashua and Ullman's approach regards identifying salient curves when there are junctions in the image (criterion 1). Shashua and Ullman's mechanism computes the saliencies of the optimal curves according to their saliency measure. At every time step, each element makes a local choice as to which element came immediately before it on the best curve. Then at convergence, the optimal curve from any element can be traced by following the local choices. To see the problem with junctions, consider the abutting objects in Fig. 1-4. Since an element at the shared edge must choose one of the object contours with which to combine, only one of the two curves can be recovered.

It is interesting to note that Shashua and Ullman suggest running their network for small numbers of iterations, stopping the process long before the network converges. We will show that stopping early can greatly affect the saliency values of the optimal curves, and, moreover, prevents the network from recovering the optimal curves. It is possible, however, to retain the optimal curves by storing a history of images, or more exactly, storing the network in all of its previous states. Nevertheless, such an approach is somewhat unattractive because, as we will show, the length of the history would be on the order of the total number of elements squared.

We believe that recovering the optimal curves in the presence of junctions is essential for making the link from saliency to grouping. To overcome this and other problems with Shashua and Ullman's method, we have designed a new approach for computing saliency that is based on finding shortest paths in graphs. In shortest-paths algorithms, a chosen measure (cost function) is optimized over all possible paths, and the optimal curves are always obtained. As we will show, the class of measures that can be optimized by shortest-paths algorithms is strictly larger than that which can be optimized by Shashua and Ullman's network.

This feature is particularly advantageous for dealing with complex images which can contain junctions (criterion 1). In our method, each optimal curve makes an individual set of choices at junctions, and does so based on a cost function that is defined with respect to the start of the curve. We will show that the Saliency Network cannot optimize such a function. Another advantage of our method is that an optimal path is traversed at most once, though some extra computation may be used in deciding which paths are optimal (criterion 2). Furthermore, our saliency measure can be simpler than Shashua and Ullman's while still having the same basic properties. Shashua and Ullman's measure was complicated by a need to form a decreasing geometric series when evaluated over a closed curve, which was necessary for convergence. Partly due to this simplicity, we are able to choose a measure that both selects long, smooth, connected contours and is invariant to similarity transformations (criteria 3 and 4).

1.3 Background

Model-based object recognition and grouping were first addressed by Roberts in 1965 [77]. In his seminal work, Roberts searched the image for convex groups of line segments, and matched the line segments against a geometric model. Over time model-based recognition and grouping have grown into subfields of their own. We discuss past work for each separately.

1.3.1 Model-based recognition and error propagation[1]

Due to the value of top-down knowledge in model-based vision, it is common to generate hypotheses about an object's pose based on a small amount of information, and then to look for evidence to confirm or reject the hypotheses. In the alignment approach, a small number of image features are matched to model features to determine the object's pose. This pose is used to project additional model features into the image, which are matched to nearby image features for verification (e.g., Roberts [77], Clark et al. [22], Fischler and Bolles [31], Lowe [64], Ayache and Faugeras [5], Huttenlocher and Ullman [52]). In constrained search, additional matches between model and image features are then used to look for more matches, backtracking if enough valid matches cannot be found (e.g., Bolles and Cain [13], Goad [32], Oshima and Shirai [71], Grimson and Lozano-Pérez [39, 40], Horaud [46]). To obtain the object's pose, some approaches use minimal sets of matches between model and image features (e.g., Clark et al. [22], Fischler and Bolles [31], Ayache and Faugeras [5], Horaud [46], Huttenlocher and Ullman [52]). Other approaches use indexing to match more than the minimal number before looking for confirming features (e.g., Rothwell et al. [78], Thompson and Mundy [88], Lamdan et al. [60], Jacobs [55]).

Most recognition systems take an ad-hoc approach to the problem of accounting for the effects of sensing error on the projected positions of unmatched model features. Some systems match projected model features to image features if they are separated by a distance that is less than some threshold (e.g., Clark et al. [22], Fischler and Bolles [31], Brooks [18], Bolles and Cain [13], Huttenlocher and Ullman [52]). Other systems rank the unmatched image features using heuristics involving distance and orientation, and then pick the feature with highest rank (e.g., Ayache and Faugeras [5], Lowe [64]).

Recently, there has been considerable effort aimed at better understanding the effects of error on the matching process. Some of this work attempts to design algorithms that are guaranteed to perform well in the presence of error (e.g., Baird [7],

[1]This section is joint work between the author and David Jacobs.

Cass [21], Breuel [17]), but most relevant to this thesis is work that also examines the propagation of error in recognition systems.

Two-dimensional objects

Huttenlocher [51] examined the effects of bounded error on the alignment approach to recognition. This analysis considered planar objects viewed from arbitrary 3D positions, assuming scaled-orthographic projection. Poses were determined by matching three model and image points, and the poses were used to predict the image locations of the unmatched model points. Assuming circular error in the three image points, Huttenlocher placed upper bounds on the uncertainty regions, by independently bounding the error in the x and y coordinates of the predicted model points. The x and y error bounds are guaranteed to hold only if the matched model points are far enough apart (approximately five pixels).

Subsequently, Jacobs [54] showed that the true uncertainty regions are discs, and gave analytic expressions for their centers and radii. These regions are circular because in this case the projection model is linear in such a way that error in any of the three matched image points causes error in a projected model point that is identical but scaled by a constant factor. This constant factor depends on the model structure, but not on the viewpoint. Consequently, the sizes of the uncertainty regions are independent of how far apart in the image are the three matched points, which means the uncertainty is independent of the pose of the model. Jacobs' result was used by Grimson et al. [38] to analyze the false-positive sensitivity of planar alignment.

A number of researchers have also considered the effect of Gaussian error on alignment methods. As mentioned above, for planar objects, each predicted model point can be written as a linear combination of the matched image points. Therefore, Gaussian error in the image points leads to Gaussian uncertainty in every predicted point (e.g., [87]). Sarachik and Grimson [80] used this observation to propose a new method of performing and evaluating alignment approaches to recognition. Beveridge et al. [10] use a robust method to evaluate particular model poses.

Error propagation has also been studied in the context of Geometric Hashing approaches to recognition [59, 60, 61]. Costa et al. [25] considered the distribution of uncertainty regions in terms of the affine invariant parameters that describe the image points. Rigoutsos and Hummel [75, 76] also considered this issue for Gaussian and uniform error. Both Costa et al. and Rigoutsos and Hummel then considered the implications of these results for recognition schemes. Lamdan and Wolfson [62] considered the related problem of determining when three image points provide an unstable basis for Geometric Hashing. Grimson and Huttenlocher [35] considered the effects of bounded error on Geometric Hashing, and provided loose bounds on this effect. Jacobs [54] determined exactly how bounded error effects Geometric Hashing indices. Grimson et al. [38] then further developed this result and used it to analyze the performance of Geometric Hashing algorithms. Sarachik and Grimson's [80] results also apply to Geometric Hashing.

Three-dimensional objects

Error propagation is more complex in recognition systems that deal with fully three-dimensional objects. Bolles et al. [14] studied how error propagates from the parameters of a model-to-image transformation to the predicted model points. Bolles et al. assumed that the errors in the parameters were independent and normally-distributed and that estimates of the distributions would be available. Unlike other previous work, Bolles et al. dealt with perspective projection, which made the relationship between the error vectors in the transformation parameters and the predicted points non-linear. In fact, their analysis is the most similar to our own, because they took a (first-order) approximation that linearizes the error-vector relationship. As a result they obtained Gaussian uncertainty distributions. The main difference with our work, in addition to our treatment of bounded error, is that we will let the error be in the matched image points, instead of assuming we know the distributions for all of the transformation parameters. Furthermore, we will derive direct expressions for the predicted points in terms of the matched points, so that we do not explicitly go through a rigid transformation.

Recently, Grimson et al. [37] presented a formal analysis of error propagation starting from the matched image points, for three-dimensional objects. They considered scaled-orthographic projection and bounded, circular error. Starting from three matched points, they provided a numerical method of bounding the uncertainty in the transformation parameters. Then they used the bounds on the parameters to obtain complicated, loose bounds on the uncertainty regions of the predicted points. Via these bounds, they analyzed the false-positive sensitivity of 3D-from-2D alignment and transformation clustering, in the domain of point features. The numerical technique is less practical, however, for use at run-time in a recognition system.

Using the same projection and error models as Grimson et al. [37], Alter and Grimson [4] presented experiments that show that the true uncertainty regions tend to be circular to a good approximation, and presented a numeric method for more accurately bounding the uncertainty regions. This technique was used to study again the false-positive sensitivity of 3D-from-2D alignment, except also using line features for verification. Alter and Grimson demonstrated that using points for generating hypotheses and lines for verification could lead to robust recognition. As before, the numerical error-propagation technique is less practical for a real-time system. Furthermore, the two weak-perspective solutions lead to two distinct uncertainty regions, which is not true when the model is planar. Alter and Grimson's technique sometimes performed poorly when the two regions overlapped, because it had difficulty distinguishing them.

Also for 3D objects, Weinshall and Basri [94] provided analytic bounds on the amount of error in a least-squares solution that is used to match four model and image points. This is useful because, currently, the least-squares solution itself can be found only through iterative methods.

For both 3D and 2D objects, Wells [96, 97] used a Bayesian approach and Gaussian error assumptions to derive an evaluation function that measures the likelihood of any given pose. Wells then used heuristic search and gradient descent methods to find the most probable pose.

Lastly, there has been a great deal of work on finding a pose that minimizes error, when enough image and model features have been matched to over-determine the pose. Some of this work analyzes the effect that errors in image features have on the accuracy of the resulting pose, including Kumar and Hanson [58] and Hel-Or and Werman [45]. The work of Hel-Or and Werman is particularly relevant to us, because they also consider how error propagates through the pose to the projections of unmatched feature points. Assuming Gaussian error, they use an extended Kalman filter to find the minimal error pose resulting from a match between any number of image and model points. The Kalman filter then allows them to compute a Mahalanobis distance that indicates the likelihood that error can account for the apparent deviation between a projected model point and a potentially matching image point.

In summary, there are simple analytic solutions for how error propagates from three matched image points, when the objects are two-dimensional and undergo scaled-orthographic projection. This is true both when the image-point error is bounded by circles and when it is normally distributed. In the case of circular error, every propagated uncertainty region is a circle, whose size is independent of the camera viewpoint.

For three-dimensional objects, it appears empirically that circular error again propagates to circular uncertainty regions. Nevertheless, there is no analytic solution, which would be preferred for building an efficient system. As well, current numerical solutions either significantly overestimate the uncertainty regions or can break down when the two regions that arise from the two weak-perspective solutions overlap. Further, it is not known whether the uncertainty regions are exactly or approximately circles, or whether the sizes of the regions depend on the viewpoint. If the regions are circles only approximately, one would like to know which configurations of the model and image points cause the regions to deviate from circularity. Although much progress has been made in understanding the effects of propagated error, there are significant problems that are not yet understood.

Finally, we mentioned a number of sensitivity analyses that determine the susceptibility of recognition systems to false-positive errors. Most of these analyses are restricted to two-dimensional objects, because this is where error propagation is most readily understood. Nonetheless, there do exist sensitivity analyses for three-dimensional objects, which use numerical techniques to get a handle on the propagated error.

1.3.2 Grouping and saliency

The task of grouping is to identify groups of features in the image that are likely to come from a single object. Often this is done by searching the image for groups that optimize a specific set of criteria. This search requires choosing from an exponential number of possible edge groupings. To find desirable groups, some approaches potentially search the entire exponential space, but then may reduce the search in practice by using heuristics which can lead to sub-optimal solutions [47, 26, 49, 55]. Other methods use greedy algorithms which need not be optimal [73, 53, 83, 72], or else incorporate substantial domain-specific information to constrain the possible groupings [67, 98, 68].

On the problem of saliency, Shashua and Ullman [82] proposed a measure for saliency that encourages long and straight curves. Using dynamic programming, they implemented a relaxation process on a network of locally connected elements to optimize the measure. The network could identify the most salient element in the image, which could lie on either an open or closed curve, and could simultaneously smoothly fill-in gaps and tolerate noise. Subirana and Sung [85, 86] extended Shashua and Ullman's method to look for skeletons of regions.

Shashua and Ullman later used the output of their saliency network to implement a grouping system [83]. The grouping technique does not adhere to the local decisions of the network's elements for their optimal neighbors. As a result, the grouping system could make choices at junctions that were not optimal according to the saliency measure.

Guy and Medioni [43] also produced a saliency map from an edge image. In their scheme, each point in the image receives a saliency value equal to a weighted sum of contributions from the individual edge elements. The weight assigned to an element is based on a circular-arc completion between it and the image point; the weight decreases with the total curvature of the arc, preferring straighter and shorter completions. Unlike Shashua and Ullman, there is no notion of optimality in the saliency values, that is, there is no attempt to optimize a measure of saliency over the set of image curves.

The use of dynamic programming to find optimal curves in images was first introduced in 1971 by Montanari [69]. Montanari's method needs to know the length of the curve for which it is searching, since otherwise the best curve could be infinitely long. Shashua and Ullman's dynamic-programming method is similar to Montanari's. Shashua and Ullman handled the problem of infinitely long curves by designing their measure-of-fit to decrease according to a geometric series when evaluated around any closed-curve. This allowed them to run their process to convergence, without knowing the length of the optimal curve.

Amini et al. [6] applied a variant of Montanari's method to implement "active contours" [56]. Active contours are a top-down approach for pinpointing a desired contour in an image, given an approximate location of the contour as a starting point. Assuming a starting contour, Amini et al. used dynamic programming to search over a window about each sample point on the contour.

In addition to dynamic programming, shortest-paths algorithms also have been used before to find curves in images. To our knowledge there are only two such works. The first was by Martelli in 1976 [66]. Martelli's formulation involves searching for best paths through a sequence of tables whose sizes are on the order of the size of the image. The number of tables is potentially the size of the image as well. To make the method feasible, Martelli used a heuristic, A* search algorithm to prune the number of table entries that actually had to be generated. In our approach, a fixed graph is constructed on top of the image and all of the work is done in place.

More recently, Hu et al. [50] used a shortest-paths approach to build a user-assisted system for finding roads in aerial images. Like us, Hu et al. searched a graph that is constructed on top of the image. However their graph is not locally connected and starts from higher-level, ribbon-based primitives. A locally connected graph is one of the defining characteristics of our saliency approach—we are looking to provide a means to obtain the higher-level primitives in the first place.

Other work on grouping addresses different aspects of the problem. For instance, [44, 99, 42, 30] identify occluded and subjective contours, and [90, 79, 15, 16, 48, 93, 19] determine the shapes of gap completions.

1.4 Overview of the Thesis

This thesis contains four pieces of work, divided as such into the next four chapters. Saliency is discussed in the second and third chapters, and robust verification in the fourth and fifth. Each pair of chapters is closely linked in terms of methodology. Nonetheless the chapters are individually motivated and can be read separately. This is natural because the individual problems are of interest beyond the grouping-plus-alignment framework.

In Chapter 2 we analyze the well-known saliency technique of Shashua and Ullman. The analysis motivates our own saliency approach, given in Chapter 3. The analysis indicates that Shashua and Ullman's Saliency Network can change its preferences as the curves are uniformly scaled. Also, we consider the case of a circle among random background elements, and we study the effect of gaps. Finally, we analyze the convergence properties of the network and discuss problems due to coarse sampling of the range of possible orientations.

As mentioned, grouping is our main motivation for studying saliency. We believe that mechanisms for saliency and grouping should be closely tied. To this end, we believe that the optimal curves evaluated during the saliency stage should be made available to do grouping and should be robust to junctions. Chapter 3 details a new approach to saliency that is based on finding shortest paths in graphs. In this approach, a curve is found through each element in the image that is optimal over all possible curves through that element. Each optimal curve makes an independent set of choices at junctions.

Chapter 4 considers the following fundamental problem. Given a set of correspondences between 2D and 3D points, in which the 3D points are rigidly connected, how does uncertainty in the positions of the 2D points propagate to uncertainty in the predicted image position of any additional 3D point? This problem is important for alignment and backtracking search (constrained search) approaches, as well as for approaches that use indexing or a Hough transform as a preprocessing step for later verification. Under the assumptions of bounded and Gaussian error in the image points, we show how to compute the region of uncertainty for any additional projected point. We rigorously derive an analytical result for the case of a minimal set of three point correspondences using a weak-perspective (scaled-orthographic) projection model. Our solution is an approximation which we experimentally demonstrate is accurate for amounts of error that are of interest in most recognition applications. Additionally, we show that the true shapes of the uncertainty regions can be considerably more complex. Furthermore, we extend the solution to perspective projection and to the case of any number of initial point correspondences. Our solution for more than three correspondences utilizes linear programming.

The basic contribution of Chapter 4 is that it solves the error-propagation problem for three-dimensional objects, where previous results applied only to two-dimensional objects. As we will show, this work can be used to extend existing methods for performing robust recognition to 3D objects. In particular, we extend the following robust methods: Alter and Grimson's alignment using line features, Sarachik and Grimson's alignment for Gaussian error, Cass' transformation-space search, and Baird's and Breuel's constrained-search techniques.

Chapter 5 explains how to build a robust alignment-style system for recognizing three-dimensional objects in images. To handle objects that are three-dimensional, the method relies on the results of Chapter 4, particularly that we can efficiently compute the propagated uncertainty regions. The method avoids pre-set thresholds by deciding on-line whether to accept or reject a set of correspondences. Finally, the method applies equally well to bounded error, Gaussian error, or any other error distribution for which the propagated distribution of uncertainty can be computed.

The new method is offered without experimentation to demonstrate the ideas. The chapter is intended as a proposal for how one might use the results of Chapter 4 to build a more robust, formally grounded alignment system for recognizing 3D objects from images. The method of error control is expected to be fast and could serve as a filter to an extensive verification stage that utilizes the entire object contour.

Chapter 6 considers again the overall grouping-plus-alignment framework which subsumes the thesis. The chapter discusses the domain where the framework can be applied most readily. Then within this domain the chapter describes how to build a three-dimensional object recognition system which, in order to be efficient and robust, takes advantage of our new results.

Chapter 2

Analysis of Shashua and Ullman's Saliency Network[1]

In line drawings, certain shapes attract our attention more than others. For example, these shapes may be the ones that are smooth, convex, and closed. (See for example Fig. 2-1.) A mechanism that can rapidly locate such salient shapes could significantly reduce the computational burden of a grouping system, where for grouping the task is to identify sets of image features that come from the same object. For a saliency operator, we define its goal as identifying salient locations in an image, in which we consider a location to be salient if a salient curve passes through it, according to some measure of curve saliency. Notice that this definition does not require the saliency operator to collect the salient locations into curves. Nonetheless, a saliency method which also provides the salient curves would be especially useful to a grouping system. In general it is more difficult to also compute the curves, because computing a curve that passes through a series of edge fragments and gaps requires making many choices, particularly at junctions. Most existing methods for saliency do not consider the problem of providing the actual curves (e.g., [43, 42, 44, 99]).

Shashua and Ullman [82] proposed a method for marking salient locations in an image that also provides the salient curves. The saliency of a location was determined

[1]This chapter is joint work between the author and Ronen Basri.

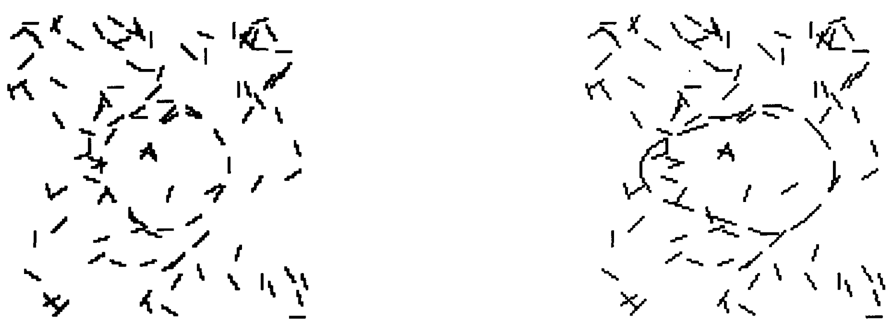


Figure 2-1: A fragmented circle and a fragmented convex shape in the middle of noise. In both cases the global shapes are apparent.

by optimizing a measure of fitness over all curves through that location. The measure has the following properties. First, when all other parameters are held constant, it monotonically increases with the length of the evaluated curve. In addition, it decreases monotonically with the total squared curvature of the curve. Thirdly, the function evaluates fragmented curves, in which case it penalizes according to the amount of fragmentation. Finally, the function is a sum of contributions from different sections of the curve, where these contributions decay with the sections' accumulated squared curvature and gap length from the beginning of the curve. Using this saliency function, Shashua and Ullman defined the "saliency map" of an image to be an image in which the intensity value of each pixel is proportional to the score of the most salient curve emanating from that pixel.

A network of locally connected elements (the Saliency Network) was proposed for computing the saliency map. The Saliency Network's computation involves local interactions between image locations, and its size is proportional to the size of the image. The network implements a relaxation process that optimizes the saliency measure. As a consequence of the optimization, the network can identify the most salient location in the image, which could lie on either an open or closed curve. Additionally, the method attempts to fill in gaps smoothly while simultaneously overcoming noise.

The Saliency Network is an efficient and elegant method, well suited to locating salient structures in images. In this chapter we provide an analysis of Shashua and Ullman's method. Our analysis can be divided along two basic issues, namely the particular function for the saliency of a curve (Section 2.2), and the computation that optimizes the measure (Sections 2.3, 2.4, 2.5). The first issue is important for determining which curves will be given to a grouping system, and second issue relates to the Network's ability to find those curves. These issues are intimately connected, for the design of the Saliency Network restricts the set of saliency measures that can be used.

Our analysis begins by introducing a continuous formulation of Shashua and Ullman's discrete saliency measure (Section 2.1). Then in Section 2.2 we demonstrate that in many cases the network indeed locates the long and smooth curves in the image. We also find cases, however, in which the network returns unexpected results. In particular, there is some conflict in the way the network ranks open and closed curves. We show that short line segments next to a salient closed curve may be judged more salient than the closed curve. In this situation, the most salient locations in the network will represent the saliency of the closed curve, but will not lie on the closed curve.

Another conflict in the network is in how it decides between a pair of curves that have the same length. The network can change its preference as the length of the curves changes. For instance, consider a straight line and a circle of the same length. For lengths less than a certain value, the line is preferred over the circle, whereas for larger lengths this preference reverses. Shashua and Ullman's rankings of curves, therefore, are not invariant to uniform scaling of the image. Recall that in Chapter 1 we listed invariance to scale as one of our four criteria for a saliency mechanism.

In Section 2.3, we analyze the convergence properties of the network and show that the complexity of the network in serial implementations is quadratic in the number of elements. On a closed curve that is alone in the image, the time to process the curve can be on the order of the square of its length; this behavior does not meet our time criterion for a saliency mechanism (criterion 2 from Section 1.2.2). Section 2.4 considers problems due to coarse sampling of the range of possible orientations. We show that with proper sampling the complexity of the network becomes cubic in the size of the image. Lastly, we discuss the class of measures that the network can optimize, and the issues involved with using the output of the network for grouping

(Section 2.5).

In terms of our interest in grouping, the main conclusion of the chapter is that the Saliency Network has some fundamental drawbacks, largely captured by our four saliency criteria. In the next chapter, this will lead us to design a new mechanism for saliency. Nevertheless, the Saliency Network can often be an effective means for identifying salient curves in images, and in many situations may be the method of choice.

2.1 Definitions

Shashua and Ullman defined their saliency measure as follows. For every pixel in the image, there is a fixed set of "orientation elements" connecting the pixel to neighboring pixels (Fig. 2-2-left). Each orientation element is called "actual" or "real" if it lies on an edge in the underlying image, and otherwise it is called "virtual" or "gap." Given a curve Γ composed of the $N+1$ orientation elements $p_i, p_{i+1}, ..., p_{i+N}$ (Fig. 2-2-right), the saliency of Γ is defined by

$$\Phi(\Gamma) = \sum_{j=i}^{i+N} \sigma_j \rho_{ij} C_{ij}, \tag{2.1}$$

with

$$\sigma_j = \begin{cases} 1, & \text{if } p_j \text{ is actual} \\ 0, & \text{if } p_j \text{ is virtual} \end{cases} \qquad \text{and} \qquad \rho_{ij} = \prod_{k=i}^{j} \rho_k = \rho^{g_{ij}},$$

where $\rho_{ii} = 1$ and where ρ is some constant in the range $[0,1)$.[2] (Shashua and Ullman set ρ to 0.7.) σ_j ensures that only actual elements will contribute to the saliency measure. g_{ij} is the number of gap elements between p_i and p_j, and ρ_{ij} reduces the contribution of an element according to the total length of the gaps up to that element. Further,

$$C_{ij} = e^{-k_{ij}},$$

[2]The formula for ρ_{ij} appeared in [82] as $\rho_{ij} = \prod_{k=i+1}^{j} \rho_k$, but the computation actually performed by the network (which is given by Eq. 2.5) converges to the modified formula given here.

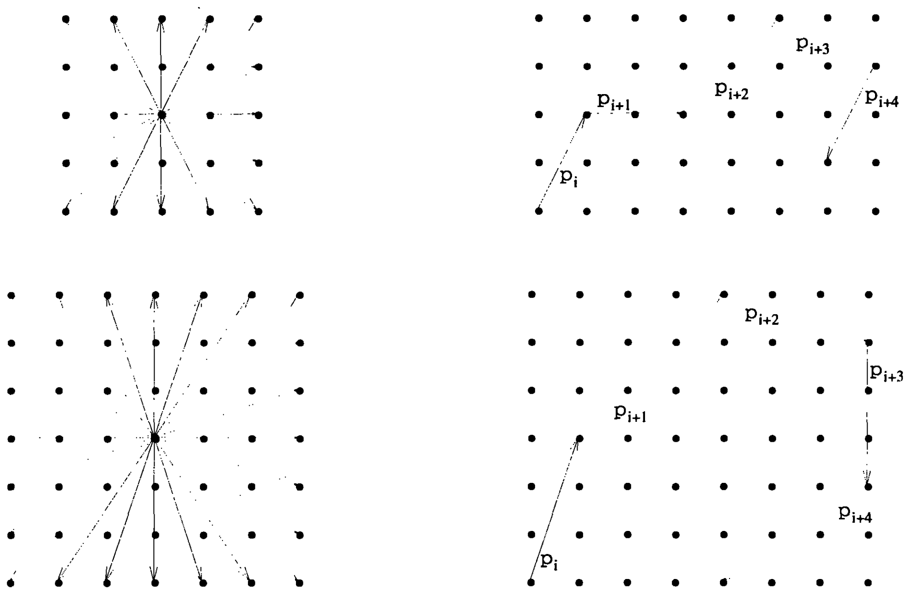


Figure 2-2: Example of the connectivity of Shashua and Ullman's Saliency Network, for the cases of sixteen and twenty-four orientation elements per pixel. In the left pictures, the neighbors of a pixel (x, y) are $\{(x+\Delta x, y+\Delta y) | \max(|\Delta x|, |\Delta y|) = \Delta e\}$, where $\Delta e = 2$ for 16 elements per pixel and $\Delta e = 3$ for 24 elements per pixel. Given the pixel neighborhoods in the left pictures, the right pictures show examples of five-element curves.

with

$$k_{ij} = \int_{p_i}^{p_j} \kappa^2(s) ds,$$

where $\kappa(s)$ is the curvature at position s. k_{ij} reduces the contribution of elements according to the accumulated squared curvature from the beginning of the curve.

The saliency of an element p_i is defined to be the maximum saliency over all curves emanating from p_i:

$$\Phi(i) = \max_{\Gamma \in \mathcal{C}(i)} \Phi(\Gamma), \tag{2.2}$$

where $\mathcal{C}(i)$ denotes the set of curves emanating from p_i. Shashua and Ullman showed how to compute $\Phi(i)$ on a network of locally connected elements. Denote by $\Phi_N(i)$ the saliency of the most salient curve of length $N + 1$ or less emanating from p_i. The measure $\Phi_N(i)$ satisfies

$$\Phi_N(i) = \max_{p_j \in \mathcal{N}(i)} F(i, j, \Phi_{N-1}(j)), \tag{2.3}$$

where $\mathcal{N}(i)$ is the set of all neighboring elements of p_i, and where $F()$ is a function of $\Phi_{N-1}()$ and constants stored at elements p_i and p_j. Shashua and Ullman referred to this type of measure as "extensible."[3] In the Saliency Network,

$$F(i, j, \Phi_{N-1}(j)) = \sigma_i + \rho_i C_{ij} \Phi_{N-1}(j), \tag{2.4}$$

which gives

$$\Phi_N(i) = \sigma_i + \rho_i \max_{p_j \in \mathcal{N}(i)} C_{ij} \Phi_{N-1}(j). \tag{2.5}$$

To analyze the network we define a continuous version of the saliency measure. Given a curve $\Gamma(s)$ of length l ($0 \leq s \leq l$, s denotes arc length), we define Φ by

$$\Phi(\Gamma) = \int_0^l \sigma(s)\rho(0, s)C(0, s)ds, \tag{2.6}$$

where

$$\begin{aligned} \sigma(s) &= \begin{cases} 1, & \text{if } \Gamma(s) \text{ is actual} \\ 0, & \text{if } \Gamma(s) \text{ is virtual} \end{cases} \\ \rho(s_1, s_2) &= \rho^{g(s_1, s_2)} \\ C(s_1, s_2) &= e^{-k(s_1, s_2)}, \end{aligned}$$

where $g(s_1, s_2)$ is the total gap length of Γ between s_1 and s_2 and $k(s_1, s_2)$ is the total squared curvature between s_1 and s_2, defined by

$$g(s_1, s_2) = \int_{s_1}^{s_2} (1 - \sigma(t))dt, \qquad k(s_1, s_2) = \int_{s_1}^{s_2} \kappa^2(t)dt. \tag{2.7}$$

A useful tool in computing saliencies is the following rule. Given a curve Γ which

[3]Note that this definition of extensibility is different than that used by Brady et al. [15].

is composed of two smoothly concatenated sections, Γ_1 and Γ_2, the saliency of Γ is given by

$$\Phi(\Gamma) = \Phi(\Gamma_1) + \rho^{g(\Gamma_1)} e^{-k(\Gamma_1)} \Phi(\Gamma_2), \tag{2.8}$$

where $g(\Gamma_1)$ is the total gap length and $k(\Gamma_1)$ is the total squared curvature of Γ_1.

Over the remainder of the chapter, we will present examples of the Saliency Network on simulated and real images. Our implementation replicates Shashua and Ullman's original implementation, except that we generally increased the number of orientation elements per pixel to obtain greater accuracy. Shashua and Ullman used sixteen orientation elements per pixel. Unless otherwise specified, we use twenty-four elements per pixel, and we set $\rho = .7$ (as in the original implementation).

2.2 Properties of the Saliency Measure

We begin our analysis by examining the saliency measure proposed by Shashua and Ullman. Section 2.2.1 below discusses the treatment of cycles. Section 2.2.2 analyzes the behavior of the measure when applied to simple curves. Lastly, Section 2.2.3 analyzes the behavior of the measure when applied to curves that include gaps.

2.2.1 Cycles

The measure of saliency proposed by Shashua and Ullman is a positive function that increases monotonically with the lengths of the curves in the image. Closed curves (cycles) are considered to have infinite length, even though they form finite structures in the image. Shashua and Ullman showed that their network is guaranteed to converge when applied to closed curves. The reason it converges is that the contribution to the saliency from remote elements attenuates geometrically with the curvature accumulated from the beginning of the curve. In cycles this generates a geometric series that converges to a finite value.

Formally, given a closed curve Γ, denote by Φ the saliency of an element of Γ that is obtained by starting from that element and then proceeding once around the curve. Denote by k the total squared curvature of the cycle and by g the number of gap elements in Γ. Then by repeatedly applying Eq. 2.8 we obtain

$$\Phi(\Gamma) = \Phi + \rho^g e^{-k}\Phi + \rho^{2g} e^{-2k}\Phi + \ldots = \frac{\Phi}{1 - \rho^g e^{-k}}. \tag{2.9}$$

When the network is applied to an open curve, after going once along the curve the network may choose to take a 180° turn and then walk back along the curve. This way an open curve could be considered as closed. As we shall see next, the attenuation due to the 180° turn is so high that the additional score is negligible. Let Γ be an open curve. Let Φ_f and Φ_b be the saliencies of Γ measured by going once along the curve in the forward and backward directions, respectively. Denote by k the total squared curvature of Γ. Then, the saliency of Γ is given by

$$\Phi(\Gamma) = \Phi_f + e^{-k-\pi^2}\Phi_b + e^{-2k-2\pi^2}\Phi_f + e^{-3k-3\pi^2}\Phi_b + \ldots = \frac{\Phi_f + e^{-k-\pi^2}\Phi_b}{1 - e^{-2k-2\pi^2}}. \tag{2.10}$$

If Γ is symmetric then $\Phi_f = \Phi_b$ and we obtain

$$\Phi(\Gamma) = \frac{\Phi_f}{1 - e^{-k-\pi^2}} \approx \frac{\Phi_f}{1 - .000051723\, e^{-k}}. \tag{2.11}$$

The largest increase in saliency is obtained for a straight line ($k = 0$), where the saliency becomes

$$\Phi(\Gamma) = \frac{\Phi_f}{1 - e^{-\pi^2}} \approx 1.000051725\, \Phi_f. \tag{2.12}$$

One can see that the additional saliency obtained by wrapping around an open curve is very small and practically can be ignored. As a consequence, the network is likely to prefer connecting the curve through gaps to other curves, or even around to itself, since such connections often will result in higher saliencies.

2.2.2 Straight lines and circles

In this section we compute the saliencies of a few simple curves. In general, we will only be interested in the measure of saliency obtained for the most salient element of the curve. Throughout this section we shall use the continuous definition of the saliency measure (Eq. 2.6). We consider only curves with no gaps (we will analyze curves with gaps in Section 2.2.3); hence $\sigma(s) = 1$ and $\rho(0, s) = 1$ for all s. Eq. 2.6 therefore becomes

$$\Phi(\Gamma) = \int_0^l C(0, s) ds, \tag{2.13}$$

where

$$C(0, s) = e^{-\int_0^s \kappa^2(t) dt}.$$

The examples below demonstrate some of the problems with Shashua and Ullman's saliency measure. In particular, we compare the saliency of a line segment of length l to that of a circle of perimeter l. We show that for small values of l, the straight line is preferred over the circle, and that this preference reverses for large values of l. The saliency function, therefore, ranks curves differently when these curves are scaled uniformly. In another example, we analyze the results of applying the saliency measure to a picture containing a circle and short line segments connected to it. We see that a short line segment increases its saliency value by connecting to the circle. As a result of this increase, it is not unusual for a short segment to become more salient than a circle. The saliency of the short line segment in this case represents the saliency of the circle, but the most salient element is in fact not part of the circle.

We begin by deriving explicit formulas for the saliency of straight lines and curves. For straight lines $C(0, s) = 1$ for all s. Therefore, ignoring the possibility that a line can wrap around itself (see Section 2.2.1), a straight line of length l will obtain the score $\Phi(\Gamma) = l$. The saliency of a straight line, therefore, grows linearly with the length of the line.

For a circle of radius r, the curvature is constant, $\kappa = 1/r$, and so for a circular arc of length s,

$$C(0, s) = e^{-\int_0^s \frac{1}{r^2} dt} = e^{-\frac{s}{r^2}}. \tag{2.14}$$

The saliency attributed for the circular arc is

$$\Phi(0, s) = \int_0^s C(0, t) dt = \int_0^s e^{-\frac{t}{r^2}} dt = r^2 \left(1 - e^{-\frac{s}{r^2}}\right). \tag{2.15}$$

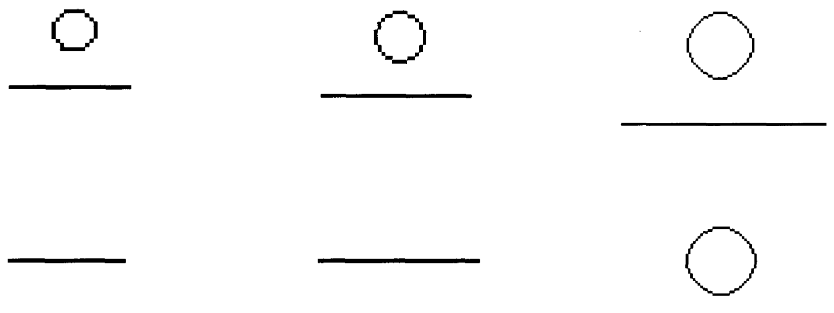


Figure 2-3: Lack of scale invariance in the Saliency Network. Top figures: three images that contain a straight line and a circle of roughly the same length. Bottom figures: the most salient curves that were found in these images. Lengths are 27 (left), 39 (middle), and 84 (right). The saliency values obtained for the circles are 15.39 (left), 33.60 (middle), and 132.05 (right), and for the lines are 27.06 (left), 39.00 (middle), and 84.00 (right). (These examples were run on a 24-orientation network.)

Thus, at convergence ($s = \infty$), the saliency of the circle is given by

$$\Phi(\Gamma) = \lim_{s \to \infty} r^2 \left(1 - e^{-\frac{s}{r^2}}\right) = r^2. \tag{2.16}$$

The score of a circle, therefore, grows quadratically with the radius (and thus also with the perimeter) of the circle.

The fact that the saliency of a straight line grows linearly with its length, whereas the saliency of a circle grows quadratically with its perimeter, suggests that the network may treat the two differently when they are scaled. Consider a straight line of length l and a circle of perimeter $l = 2\pi r$. These two entities will have exactly the same saliency when $l = 0$ and when $l = 4\pi^2 \approx 39.48$. (The saliencies in the two cases are 0 and $4\pi^2$, respectively.) When $0 \leq l \leq 4\pi^2$ the line will be more salient than the circle, whereas when $l > 4\pi^2$ the circle will be more salient. Fig. 2-3 shows an example of three images, each of which contains a straight line and a circle of the same length. Consistent with our analysis, the Saliency Network found the straight line to be more salient than the circle at shorter lengths, and found the circle to be more salient at longer lengths.

An interesting case is a circle connected to short line segments. Given the picture in Fig. 2-4-left, which is the most salient element? Counterintuitively, the most salient element computed by the network may be on one of the line segments. The reason is that a neighboring line segment may increase its saliency by connecting to the circle, without affecting the saliency of the circle. Consider, for example, a circular arc of length 1 and curvature κ connected smoothly to a circle of radius r (which corresponds to a single element connected smoothly to the circle via curvature κ). Using Eq. 2.8 we obtain that the saliency of the first element on the arc is

$$\Phi_e = \Phi(\Gamma) + e^{-\kappa^2}\Phi_c, \tag{2.17}$$

where Γ represents the circular arc and Φ_c is the saliency of the circle. Now, using Eq. 2.14,

$$\Phi(\Gamma) = \int_0^1 C(0,s)ds = \frac{1}{\kappa^2}(1 - e^{-\kappa^2}). \tag{2.18}$$

Combining Eqs. 2.16-2.18, we obtain that

$$\Phi_e = (1 - e^{-\kappa^2})/\kappa^2 + e^{-\kappa^2}r^2.$$

If we now compare the saliency of the element, Φ_e, to that of the circle $\Phi_c = r^2$ (Eq. 2.16), we obtain that $\Phi_e > \Phi_c$ when

$$\frac{1}{\kappa^2}(1 - e^{-\kappa^2}) + e^{-\kappa^2}r^2 > r^2 \qquad \text{or} \qquad |\kappa| < \kappa_c, \tag{2.19}$$

where $\kappa_c = 1/r$. That is, the element will be more salient than the circle if and only if it connects to the circle at a curvature that is less than the curvature of the circle. This is consistent with the network's preference for straight curves. Notice that if the element is a line tangential to the curve ($\kappa = 0$) the element will be more salient than the circle regardless of the circle's radius.

This phenomenon, that curves connecting to a circle may increase their saliencies due to these connections and actually beat the circle, is more likely to occur for longer curves. Suppose a curve Γ connects to a circle C such that the total squared curvature of Γ, including the connection point, is k. Then the saliency of the element on Γ that is most distant from the circle is given by Eq. 2.17, where κ^2 is replaced by k, namely,

$$\Phi_e = \Phi(\Gamma) + e^{-k}\Phi_c. \tag{2.20}$$

The longer Γ is, the more likely it is to become more salient than the circle. Suppose for example that Γ is a straight line of length l that connects to the circle via curvature κ. We have that $\Phi(\Gamma) = l$ and $k = \kappa^2$, which implies

$$\Phi_e = l + e^{-\kappa^2}\Phi_c. \tag{2.21}$$

Now $\Phi_e > \Phi_c$ when

$$l + e^{-\kappa^2}r^2 > r^2. \tag{2.22}$$

Substituting $r = 1/\kappa_c$, this implies that

$$\kappa_c^2 > \frac{1 - e^{-\kappa^2}}{l} \approx \frac{\kappa^2}{l}, \tag{2.23}$$

when κ^2 is small, or

$$\kappa_c > \frac{|\kappa|}{\sqrt{l}}. \tag{2.24}$$

Clearly, the longer the line is, the more likely it is to become more salient than the circle.

Fig. 2-4-left shows a picture of a circle with a few short curves connected to it. When the Saliency Network is applied to this picture, the most salient element does not lie on the circle, although most of its saliency is due to the circle. Indeed, if we disconnect these short curves from the circle, then the circle becomes the most salient structure in the image.

Fig. 2-4-middle shows an image of a dashed circle on a background of random line segments, and Fig. 2-4-right displays the most salient curve that results from applying the Saliency Network to the image. The network identifies the circle, though the most salient image location is a point outside the circle.

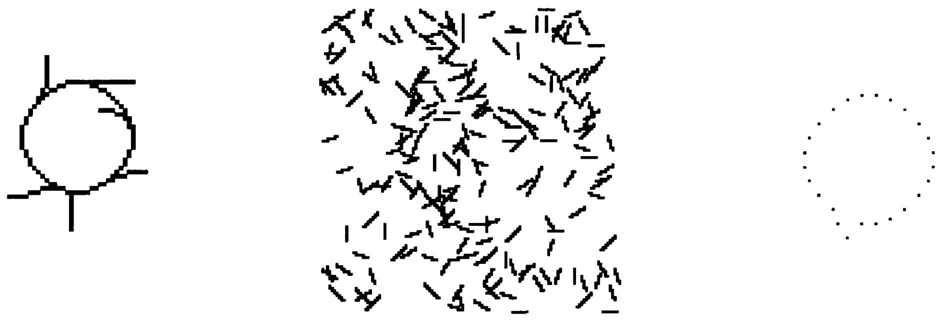


Figure 2-4: Left: An image of a circle with a few short curves connecting to it. The most salient element (for which $\Phi = 142.64$) was not on the circle, although its saliency came mostly from the circle (the saliency of the circle is 130.74). When we add short gaps between the curves and the circle, the circle becomes the most salient curve in the image. Middle: A fragmented circle on a background of 200 random elements. Right: The best curve found by the Saliency Network for the center picture (the dots mark the tails of the elements). The most salient element (for which $\Phi = 17.66$) is outside the circle. Due to the gaps in the circle in the center picture, the circle's saliency is greatly reduced as compared to the leftmost picture.

2.2.3 Curves with gaps

One of the most important properties of Shashua and Ullman's saliency network is its ability to fill in gaps while computing the saliencies. The network handles gaps by using virtual elements, which compute the saliencies of curves emanating from their locations and transfer these saliencies to their neighboring elements. Via these transfers, actual elements evaluate the saliencies of curves that emanate from their locations and contain any number of gaps. The network avoids curves with large gaps by attenuating the scores of curves exponentially with gap size.

In this section we analyze the performance of the saliency network in the presence of gaps. Due to the saliency measure attenuating exponentially with gap size, the network is capable of overcoming small gaps, but is unlikely to overcome large ones. As an example, consider the problem mentioned in Section 2.2.2, that a short line segment in the neighborhood of a circle may increase its saliency by connecting to the circle. One consequence of the fast attenuation is that this problem almost disappears when the segment is not physically connected to the circle. On the other hand, we show below that, due to the exponential decay, very long structures (straight lines and circles) obtain very low scores even when only a small fraction of the curves are gaps.

Finally, we explore the question of whether the network prefers fragmented curves (dashed lines) over curves with single gaps of the same total size. At first glance Shashua and Ullman's saliency measure appears indifferent to this property, because the total size of gaps is taken into account, irrespective of the fragmentation. In fact, for open curves there is no clear preference between a curve having many small gaps or a few long gaps. For closed curves, however, we show that a curve with a single large gap is preferred over the same curve with several small gaps of the same total size; this preference is inconsistent with the psychophysical experiments of Elder and Zucker [28].

In computing the saliency of a fragmented curve, gaps affect the total score in two ways (see Eq. 2.1). First, gap elements themselves do not contribute at all to the total score (since $\sigma_j = 0$ for virtual elements). Secondly, the actual elements of the curve that lie on the other side of a gap are attenuated by a factor ρ^g, where g is the total gap length accumulated from the beginning of the curve. Consider for example a curve Γ with one gap of length g. Denote the first part of the curve (before the gap) by Γ_1 and the second part of the curve (after the gap) by Γ_2, and denote by $k(0,m)$ the total squared curvature of Γ_1 plus the gap. The saliency of Γ is given by

$$\Phi(\Gamma) = \Phi(\Gamma_1) + \rho^g e^{-k(0,m)} \Phi(\Gamma_2) \tag{2.25}$$

Thus, Γ_1 contributes to the saliency of Γ as if there is no gap, the gap elements contribute nothing, and the contribution of Γ_2 is attenuated by ρ^g. Clearly, the longer the portion of Γ before the gap (Γ_1), the less the saliency of Γ will be attenuated. If the gap appears near the end of the curve the saliency of Γ is hardly attenuated. If the gap appears at the beginning, the entire saliency of Γ is attenuated by the factor ρ^g. Notice that since the network evaluates open curves starting from both endpoints, if a curve contains a relatively smooth section on one of its sides and a relatively wiggly section on its other side, then the highest score will be obtained when the gaps are distributed along the wiggly side.

To analyze the effect of gaps, consider a straight line Γ with gaps distributed uniformly along the line. Let p $(0 \leq p \leq 1)$ be the fraction of the line containing the actual elements, and let $q = 1 - p$ be the fraction of the line which is virtual. We can thus set $\sigma(s) = p$. The gap length g of a line segment of length l is given by ql. Since we are dealing with a straight line, $C(0, s) = 1$ for all s. Consequently, the expected saliency of a straight line of length l with fraction q in uniform gaps is given by

$$\Phi(\Gamma) = p \int_0^l \rho^{qs}\, ds = \frac{p}{q \ln \rho}(\rho^{ql} - 1)\,. \tag{2.26}$$

This score converges as l approaches infinity to

$$\Phi_\infty = -\frac{p}{q \ln \rho}\,. \tag{2.27}$$

Thus, the saliency of an infinitely long straight line with uniformly distributed gaps is always finite and, in fact, proportional to p/q. Note that since the saliency measure monotonically increases with the length of a curve the score of an infinitely long straight line with uniform gaps provides an upper bound on the score of any finitely long line segment with the same distribution of gaps.

Examples for the values assumed by Φ_∞ as a function of p and ρ are given in Table 2.1. Consider $\rho = 0.7$: When 95% of the line includes actual elements (5% gaps), the score is only 53.27, and when 90% of the line includes actual elements (10% gaps), the score drops to 25.23. This means that a straight line of length 54 will be better than any line that contains 5% gaps. Similarly, a straight line of length 26 will always be better than an infinite line with 10% gaps.

A similar analysis can be performed for a circle with uniformly distributed gaps. Note that we are talking about a circle of finite size. Given a circle with radius r and fraction p actual elements and $q = 1 - p$ virtual elements, we set $\rho(s) = p$ for all s, $g(0, s) = qs$ and, using Eq. 2.14, $C(0, s) = e^{-s/r^2}$. Thus, the saliency of Γ is given by

$$\Phi(\Gamma) = p \int_0^\infty \rho^{qs} e^{-\frac{s}{r^2}}\, ds = p \int_0^\infty e^{(q \ln \rho - \frac{1}{r^2})s}\, ds, \tag{2.28}$$

$p \setminus \rho$	0.1	0.3	0.5	0.7	0.9
0.5	0.43	0.83	1.44	2.80	9.49
0.7	1.01	1.94	3.37	6.54	22.15
0.9	3.91	7.48	12.98	25.23	85.42
0.93	5.77	11.03	19.17	37.25	126.10
0.95	8.25	15.78	27.41	53.27	180.33
0.97	14.04	26.86	46.65	90.65	306.88
0.99	43.00	82.23	142.83	277.56	939.63
1	∞	∞	∞	∞	∞

Table 2.1: Φ_∞ for a straight infinite line with uniform gaps as a function of p and ρ. Note that the score for infinite lines gives an upper bound on the score of finite ones.

which, since $q \ln p < 0$, simplifies to

$$\Phi(\Gamma) = \frac{p}{\frac{1}{r^2} - q \ln \rho}. \tag{2.29}$$

Examples for the values assumed by $\Phi(\Gamma)$, for $\rho = 0.7$, are given in Table 2.2. Similar to the case of straight lines, the saliency of circles attenuates very fast with gap size. For example, the saliency of a circle of radius 16 that contains no gaps is 256. With 5% gaps its saliency reduces to 43.70. This saliency (43.70) is identical to the saliency of a gap-free circle of radius 6.61. Similarly, with 10% gaps the saliency of the same circle reduces to 22.74, which corresponds to the saliency of a gap-free circle of radius 4.77.

Next, we analyze the case of a short curve, Γ, that lies near a circle such that the two are not touching. Again, we shall ask whether such a curve may become more salient than the circle by using the saliency of the circle. Let $\Phi(\Gamma)$ denote the saliency of Γ, let g be the gap length between Γ and the circle, and let k be the total squared curvature of Γ plus the gap to the circle. The saliency Φ_e of the first element on Γ is given by

$$\Phi_e = \Phi(\Gamma) + \rho^g e^{-k} \Phi_c. \tag{2.30}$$

$p \setminus r$	1	2	4	8	16
0.5	0.42	1.17	2.07	2.58	2.74
0.7	0.63	1.96	4.13	5.71	6.31
0.9	0.87	3.15	9.17	17.55	22.74
0.93	0.91	3.38	10.63	22.91	32.21
0.95	0.93	3.55	11.82	28.39	43.70
0.97	0.96	3.72	13.25	36.85	66.41
0.99	0.99	3.90	14.98	51.58	132.48
1	1	4	16	64	256

Table 2.2: The saliency values of circles with uniform gaps as a function of p and r (for $\rho = 0.7$).

We obtain that $\Phi_e > \Phi_c$ (recall that $\Phi_c = r^2$) when

$$\Phi(\Gamma) + \rho^g e^{-k} r^2 > r^2, \tag{2.31}$$

which implies that

$$\frac{1}{r^2}\Phi(\Gamma) > 1 - \rho^g e^{-k}. \tag{2.32}$$

Note that since $\rho < 1$ the right-hand side grows larger as the gap size increases. Consequently, the chance of an element becoming more salient than a circle by taking its saliency from the circle decreases with the gap size. Suppose finally that Γ is a straight line of length l such that its continuation is tangential to the circle, in which case $\Phi(\Gamma) = l$, $k = 0$, and the condition (Eq. 2.32) becomes

$$\frac{l}{r^2} > 1 - \rho^g. \tag{2.33}$$

For $l = 1$ and $\rho = .7$ we obtain that Γ is almost never more salient than the circle:

$$\frac{1}{r^2} > 1 - .7^g \tag{2.34}$$

or

$$r < \frac{1}{\sqrt{1 - .7^g}}. \tag{2.35}$$

From this equation, r must be extremely small to allow an element to win with gaps: For $g = 1$, we have $r < 1.826$, and for $g = 2$, we have $r < 1.400$. As l increases the likelihood of Γ becoming more salient increases.

The final issue we discuss is the saliency measure's preference for how gaps are distributed along a curve. Elder and Zucker [28] conducted experiments which demonstrate that, when a fraction of the boundary of an object is missing, humans' recognition ability is hindered more when the missing fraction is contained all in one gap than when spread over several gaps. For any curve, the saliency measure encourages gaps to be as far as possible from the starting point. For an open curve with a fixed total gap length, the best and worst cases are when the curve has one large gap at the start (worst) or end (best). Since the network evaluates the saliency of curves from all possible starting points it prefers that gaps are pushed as far as possible from the smooth sections of the curve.

While for open curves there is no clear preference for a single long gap versus a few short gaps, for closed curves such a preference does exist. Consider a circle Γ with one large gap. Let Γ_1 be the open curve corresponding to the part of the circle that is actual, and let Γ_2 be the gap. The most salient element on the circle will be the first element of Γ_1, since the saliency measure prefers gaps to be as far as possible from the start of the curve. So the most salient curve will go first through Γ_1, then through Γ_2, and then loop back to Γ_1. Let αr denote the length of gap Γ_2. Since only the actual elements contribute to the saliency of a curve, the saliency obtained by going once around the circle is simply $\Phi(\Gamma_1)$. Using Eq. 2.9 the saliency of the circle becomes

$$\Phi(\Gamma) = \frac{\Phi(\Gamma_1)}{1 - \rho^{\alpha r} e^{\frac{2\pi}{r}}}. \tag{2.36}$$

If the circle now contains, say, two gap sections of the same total length αr, then the saliency obtained by going once around the circle will be reduced. This is because a gap will be closer to the start of the curve. As a consequence, the numerator in Eq. 2.36 will become smaller. The denominator, however, will remain unchanged since the total gap length and curvature did not change. This analysis clearly applies

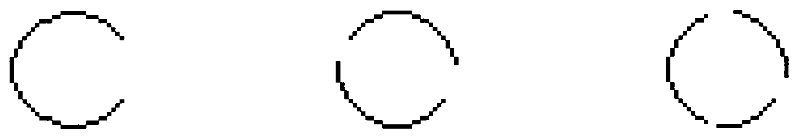


Figure 2-5: Three circles of the same radius with the same total gap size. Using Shashua and Ullman's network the saliency values are 46.85 (left), 27.93 (middle), and 23.27 (right).

when the circle is fragmented by more than two gaps. Consequently, the saliency of the circle will become smaller as a result of fragmentation. An example is given in Fig. 2-5. The figures shows three circles of the same radius and with the same total gap size. The network prefers the one that contains one long gap over the ones in which the gaps are fragmented. This behavior disagrees with Elder and Zucker's results.

2.3 Complexity and Convergence Analysis

In this section we analyze the complexity of Shashua and Ullman's saliency network. Denote the total number of pixels in the image by p and the number of discrete orientation elements at every pixel by b. The network has pb elements. At each iteration each element has to evaluate all the saliencies obtained from elements connected to it. The complexity of every iteration therefore is pb^2. The question is then how many iterations are required before the network converges. Clearly, if we did not allow cycles the longest curve may be of length p, and so the total complexity of the computation would be p^2b^2. But when cycles are considered, we show below that the network converges in a linear number of iterations, and so the total complexity is indeed $O(p^2b^2)$.

Given a cycle Γ, denote by Φ_n the score obtained after going n times around the cycle, by k the total squared curvature of Γ, and by g the total gap size. Then from Eq. 2.9 the saliency of Γ is $\Phi = \Phi_1/(1 - \rho^g e^{-k})$. After going n times around the cycle the accumulated score becomes (this is the finite sum of the geometric series in Eq. 2.9)

$$\Phi_n = \frac{1 - \rho^{ng} e^{-nk}}{1 - \rho^g e^{-k}} \Phi_1. \tag{2.37}$$

Define the relative error by

$$E = \frac{\Phi - \Phi_n}{\Phi} = \rho^{ng} e^{-nk}. \tag{2.38}$$

We can now compute the number of cycles, n, needed to achieve an $E = \epsilon$ error:

$$\ln \epsilon = n(g \ln \rho - k) \qquad \Longrightarrow \qquad n = \frac{\ln \epsilon}{g \ln \rho - k}. \tag{2.39}$$

Assume Γ is a circle of radius r with no gaps. Then $k = \frac{2\pi}{r}$ and

$$n = -\frac{r \ln \epsilon}{2\pi}. \tag{2.40}$$

The number of cycles around the circle is $O(r)$. Assuming one iteration covers one unit of arc length, the number of iterations for each cycle is $2\pi r$. Thus, from Eq. 2.40 the total number of iterations needed to achieve an ϵ error is

$$N = 2\pi r n = -r^2 \ln \epsilon. \tag{2.41}$$

Consequently, the total number of iterations required is $O(r^2) = O(p)$, where p is the size of the image. As an example, the number of cycles required to achieve 1% error ($\epsilon = 0.01$, $\ln \epsilon \approx -4.605$) is $n \approx 2.303\, r/\pi \approx 0.733r$, and therefore the number of iterations is $N \approx 4.605r^2$.

Fig. 2-6 shows an image of a gap-free and a fragmented circle on a noisy background. As expected, the Saliency Network chooses the gap-free circle as the most salient curve. Using Eq. 2.41, we could predict the number of iterations for the network to converge on the gap-free circle: The radius of the circle is $r \approx 22.79$, and one iteration covers an arc length $\Delta s \approx 5.966$ (r and Δs are discussed in the next

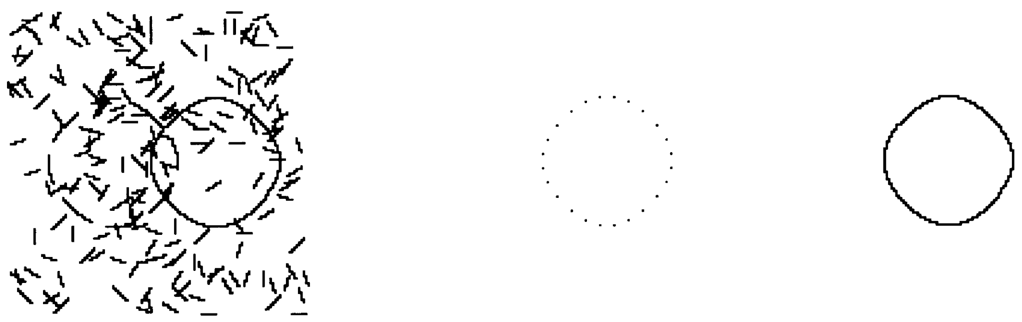


Figure 2-6: Running the Saliency Network on an image with gap-free and fragmented circles and a background of 200 random line segments (at the left). The saliency map and most salient curve image look the same, and are shown in the center picture. The dots mark the tails of the elements on the gap-free circle. After 400 iterations, the maximum saliency was 520.0. On the right: the same circle with no background elements. The time to convergence and the maximum saliency are the same for the images at the left and the right.

section). So to obtain 1% error, Eq. 2.41 gives

$$N = \left(\frac{2\pi r}{\Delta s}\right) n \approx \frac{4.605 r^2}{\Delta s} \approx 400.9. \tag{2.42}$$

We ran the Network for 400 iterations on the left image in Fig. 2-6, and the maximum saliency converged to 520.0. This generally agrees with $r^2 = 519.2$, as predicted by Eq. 2.16, and with the exact saliency of the circle under discretization, 525.3 (see next section). The image dimensions are 128×128. The 400 iterations took 1 hour and 25 minutes using C code on a Sun SPARCstation 5 with 32M of memory. Note that the time for convergence is independent of the background. So the execution time would be the same if the gap-free circle were alone in the image, as in Fig. 2-6-right.

By taking the maximal possible circle in the image, we account for the worst case complexity of the network. This is because any larger closed curve must accumulate comparable total squared curvature in order not to exceed the boundaries of the image. We can therefore conclude that the worst case complexity of the network is $O(p^2 b^2)$, which is the squared number of elements in the network.

2.4 Discrete Implementations

Our analysis of Shashua and Ullman's method has concentrated on the theoretical, continuous version of their saliency measure. Shashua and Ullman proposed to compute this measure using a network of finitely many, locally connected elements. In this section we analyze the effect of computing the saliency measure on discrete networks. We show in particular that the network is extremely sensitive to the number of discrete orientation elements allocated for the network.

Shashua and Ullman's network has the following structure. Let p be the number of pixels in the image, and let b be the number of orientation elements at each pixel. (Shashua and Ullman set $b = 16$.) The network contains $p \times b$ processors, a processor for every orientation element at every pixel in the image. A continuous arc is assigned between every two elements that meet at the same pixel in the underlying image. The value of $k = \int \kappa^2 ds$ corresponding to such an arc is approximated using the formula

$$k = \frac{2\alpha \tan \frac{\alpha}{2}}{\Delta e}, \tag{2.43}$$

where α denotes the angle between the neighboring elements and Δe denotes the length of an orientation element. This formula represents the curvature of a circular arc that joins the midpoints of two elements of the same length. As an example, the gap-free circle of Fig. 2-6 was generated using a 24-sided regular polygon with one element per side and with $\Delta e = 6$. Then $\alpha = \pi/12$, and Eq. 2.43 gives $k \approx .04388$. Therefore the radius of the circular-arc approximation is $r = 1/k \approx 22.78$ and the arc length covered by one iteration is $\Delta s = \alpha r \approx 5.966$.

Shashua and Ullman set Δe to be constant, and hence ignore the different sizes of elements of different orientations. As a result, a horizontal (or vertical) line of length l obtains the same saliency as a diagonal line of length $l\sqrt{2}$. Shashua and Ullman's implementation therefore encourages curves that are aligned with the main axes of the image.

A more critical issue is the number of orientation elements in the network. Consider for example a horizontal straight line segment cut in the middle so that one part

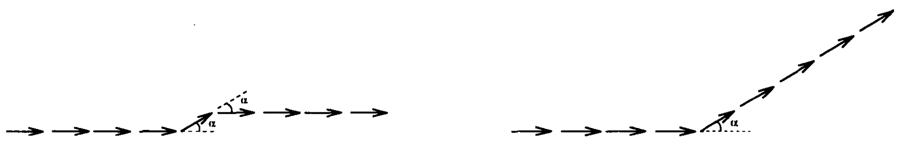


Figure 2-7: Discretization effect on a straight line. Left figure: the discretization of a straight line. Right figure: a corner. The saliency value obtained for a perfectly horizontal line of length 20 is 20.00, the saliency value for a straight line of the same length is 18.41, and the saliency value of a corner is 19.10.

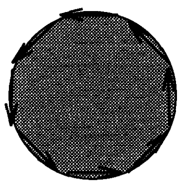


Figure 2-8: Discretizing a circle with a regular polygon.

of the line is raised up by one pixel (see Fig. 2-7). Let $2l$ be the length of the line. The saliency of the first element along the line is given by

$$\Phi_e = l + e^{-\kappa^2} + (l-1)e^{-2\kappa^2}, \tag{2.44}$$

where κ is the curvature formed by the change in orientation α corresponding to raising the line up by one pixel (which is also the curvature formed when the line returns to horizontal).

Consider now a pair of lines of length l meeting at a corner such that they form the same orientation change α. Since a corner forms only one turn the obtained saliency will be

$$\Phi_c = l + le^{-\kappa^2}. \tag{2.45}$$

Consequently, we obtain the paradoxical result that the corner is more salient than the nearly straight continuation. Hence straight lines oriented such that they deviate slightly from horizontal will often be less salient than corners.

The discretization problem is carried over to other, more complicated, examples. Consider a circle of radius r. When r is sufficiently small, the circle can be approximated by a regular polygon where each side includes a single orientation element (Fig. 2-8). The discrete saliency of such a regular polygon is given by

$$\Phi = 1 + e^{-\kappa^2} + e^{-2\kappa^2} + \ldots = \frac{1}{1 - e^{-\kappa^2}} \approx \frac{1}{\kappa^2} \approx r^2, \tag{2.46}$$

when κ^2 is small. (κ is the curvature corresponding to a turn $\alpha = 2\pi/n$ where n is the number of sides of the polygon.) The approximation in Equation 2.46 improves as κ decreases; this happens when the number of sides in the polygonal approximation increases and as a result fits a circle more closely. When r is big so that a good approximation by a regular polygon would require finer orientation changes (less than $\frac{2\pi}{b}$), a faithful discretization of the circle would involve many inflections (that is, clockwise turns balanced by counter-clockwise turns). These inflections would be penalized unduly by the network. We could improve the saliency of the discretization if we instead represent the circle by a regular polygon of b sides; each side now contains more than one element. This, however, will still not result in a reasonable approximation to the continuous saliency of the circle. This can be seen by the following observation. The saliency of a regular polygon with b sides each of a unit length is, according to Eq. 2.46,

$$\Phi_1 = \frac{1}{1 - e^{-\kappa^2}}, \tag{2.47}$$

where κ is the curvature assigned for a turn of $\frac{2\pi}{b}$. The saliency of a similar regular polygon in which every side is of length l is given by

$$\Phi_l = \frac{l}{1 - e^{-\kappa^2}} = l\Phi_1. \tag{2.48}$$

The saliency of a regular b-sided polygon, therefore, increases linearly with the length of each side, l. Since l is directly related to the radius of the circumscribed circle, the saliency of the polygon increases linearly also with the radius of that circle. Since the continuous saliency of a circle grows quadratically with the radius of the circle (Eq. 2.16), we obtain that, as r grows, the saliency of the polygon will considerably underestimate the saliency of the circle.

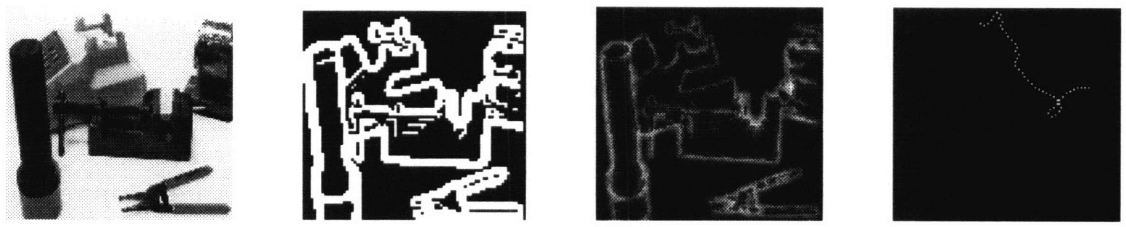


Figure 2-9: Shashua and Ullman's saliency map for a cluttered scene using 16 orientations per pixel. From left to right, the first picture is the scene image (width 124, height 117), and the second is an edge image obtained from the scene image by thresholding the gradient magnitude. The edge image was the input to the network. The third picture displays Shashua and Ullman's saliency map, and the fourth shows the first 50 elements of the curve emanating from the most salient element, for which $\Phi = 14.0$. The saliencies of the straight edges of the flashlight and the clamp are attenuated because they are not aligned with the horizontal or vertical axis of the image.

The results shown in this section establish that the Saliency Network would face serious difficulties due to the discretization of orientation elements. A faithful implementation of the continuous saliency measure requires a very fine discretization of the orientation elements. The number of orientation elements needed to completely avoid the problems mentioned in this section is of the order of $\sqrt{p}$, where p is the total number of pixels in the image. With this number of orientation elements the overall time complexity of the network (see Section 2.3) becomes $O(p^2 b^2) = O(p^3)$.

Fig. 2-9 shows an example of applying Shashua and Ullman's network to a real image. The image was smoothed with a Gaussian of standard deviation 1 and then thresholded to get a binary image. The saliency map obtained from the binary image highlights the edges of the objects, and the method found a long winding curve that has saliency 14.0. This is somewhat surprising, since in the input image, which has width 124 and height 117, the flashlight and clamp have edges that are straight and are much longer than fourteen. From examining the active elements along the straight edges of the flashlight and clamp, we observed that a sequence of these elements could not form a perfectly straight line for either edge due to the discretization. Instead, such a sequence was forced to make turns of the kind in Fig. 2-7-left, causing the saliencies of the boundaries to be attenuated.

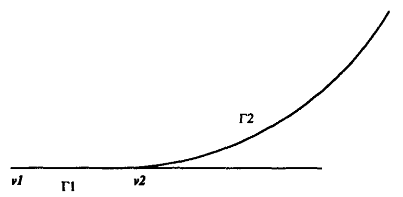


Figure 2-10: Characterization of extensible functions. If the most salient curve from v_1 goes through v_2 then, at convergence, it should coincide with the most salient curve from v_2.

2.5 Discussion

In this section we draw the reader's attention to a few additional issues concerning the Saliency Network. First we discuss the particular class of measures that can be optimized by the Network. Then we consider the issue of running the Network for small numbers of iterations, instead of waiting for convergence. Lastly we discuss the issue of using the output of the Network for grouping.

The saliency measure exhibits two important properties which enable its computation on the kind of network introduced by Shashua and Ullman. The two properties are (1) extensibility and (2) the measure induces a geometric sum for cycles. Due to the extensibility (defined in Eq. 2.3) saliencies can be computed using a procedure of recursive optimization (dynamic programming). Extensibility implies that an optimal curve must be composed of sub-curves which are themselves optimal. In particular, under Shashua and Ullman's definition at convergence if the most salient curve emerging from an element v_1 passes through another element v_2 then beginning at v_2 the most salient curve from the two nodes must coincide (see Fig. 2-10). The geometric sum is essential for convergence. Because of the geometric sum, for example, the ranking of a curve when it is traced from the beginning to the end will in general differ from its ranking when it is traced from the end to the beginning. These two properties thus considerably constrain the possible functions that can be used as measures of saliency. It remains to be seen whether variations of the measure that maintain these two properties can be defined to remedy some of the weaknesses of the current measure.

One of the benefits of extensibility is that although the Saliency Network finds the element from which the best curve emanates rather than extracting the best curve itself, the best curve can be extracted in a simple way. As has been proposed by Shashua and Ullman, to extract the best curve one has to store during the computation for each element a single pointer which points to the second element on the best curve emanating from the element. At the end of the computation the best curve is retrieved by tracing these pointers starting from the most salient element. This procedure will produce the most salient curve at convergence.

On the other hand, this procedure is not guaranteed to extract the best curve at any finite time. Consider for instance the picture in Figure 2-10. After $n-1$ iterations (where n is equal to the length of the straight line Γ_1) the best curve emerging from v_1 is the straight line of length n, Γ_1. Since the length of Γ_1 counted from v_2 is shorter than that of Γ_2 the best curve emerging from v_2 is Γ_2. If we now trace the pointers starting from v_1 we will mistakenly think that the best curve of length n contains the segment of Γ_1 from v_1 to v_2 and then a part of Γ_2. This problem can be avoided if one stores the entire history of the computation, but that of course will increase the storage space required by the method considerably.

Our analysis concentrated on the asymptotic behavior of the Saliency Network. In their experiments, Shashua and Ullman demonstrated that good results could be obtained already after a few dozen iterations (between 10 and 160). In this they relied on the property that after the n'th iteration the score attributed by the network to every element represents the saliency of the best curve of length $n+1$ emanating from the element. There is a drawback to this approach, however. Whereas after running the network for a small number of iterations the saliency values obtained for short curves already approximate their asymptotic saliencies, long curves still are undervalued significantly. This is particularly problematic when closed curves are considered, because their asymptotic scores benefit from being considered infinitely long. Thus, when the network is run for a relatively small number of iterations, closed curves are evaluated as if they were short, open curves, and as a result closure is not encouraged by the network.

The Saliency Network is viewed by many people not only as a mechanism for shifting attention to salient structures, but also as a method for the initial grouping of curves. The problems of identifying salient structures and the grouping of curves are not identical. Saliency can be viewed as the problem of identifying the "odd man out," whereas grouping is the problem of identifying image structures that are likely to belong to a single object. The criteria of length and straightness can separate a smooth object from a background of short, broken curves (e.g., a disc on a background of grass), but they may be inappropriate for segmenting equally-smooth objects in cluttered scenes. For example, one cannot expect to extract the boundary curve of a man-made object if the method used associates significant penalties to corners. The binary input image in Figure 2-9 has many long, smooth curves, some of which go around corners and some of which run across different objects. The Saliency Network in such cases often will prefer to select a curve that traverses a few objects than to select curves that include sharp turns.

Computing saliencies with the Network involves a series of local optimizations. At every step of the computation each element must choose between a few candidate curves and select the one that optimizes the measure of saliency. Consequently, when two curves share a common section (as in Figure 2-11) the elements on the common section must decide between the two curves. As a result, the most salient curves emerging from the beginning of the two curves (v_1 and v_2) will coincide as soon as they reach the common section and will never separate again. Due to this property, non-salient curves next to a salient one will often include the salient curve as part of them. We have already seen an example in which, due to this property, a non-salient curve becomes salient unduly (Section 2.2.2). This phenomenon is particularly problematic if we attempt to use the Saliency Network for grouping, since it implies that all curves in the neighborhood of a relatively salient objects will be considered part of that object. A post-processing procedure for separating the curves may be required so that reasonable groups can be obtained.

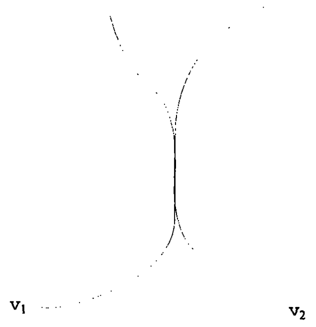


Figure 2-11: Each element of a curve chooses one neighboring element with which to combine. Consequently the shared portion must choose between the two shapes, and so the best curves emerging from v_1 and v_2 will coincide once they reach the common section, and will never split again.

2.6 Summary

The Saliency Network is a mechanism for identifying salient curves in images based on length and straightness. The method is attractive for several reasons. First, the measure of saliency generally prefers long and smooth curves over short or wiggly ones. In addition, the network is guaranteed to find the most salient structure according to the measure. While doing so, the network fills in gaps with smooth completions and tolerates noise. The network itself is locally connected and its size is proportional to the size of the image. The locality is further emphasized since the contribution of remote elements to the score of a given element attenuates with the curvature and gap length separating them from that element.

Our analysis, however, revealed certain weaknesses with the network. Some of these weakness related specifically to the measure of saliency. In particular, we showed that the network encounters problems with the ranking of open and closed curves. We showed that a short segment next to a salient closed curve may be judged more salient than the closed curve although its saliency represents mostly the saliency of the closed curve. Also, the network sometimes changes its preferences when the image is uniformly scaled.

The weaknesses with the saliency measure may in part be due to restrictions imposed by the network's computation. To guarantee convergence, the network imposes that the saliency measure decrease in a geometric series when evaluated along a cycle. Furthermore, we analyzed the network's time complexity, implied by the geometric series. In particular, the number of iterations required by the method is linear in the size of the image, and that as a consequence the overall complexity of the network in serial implementations is quadratic in the size of the network. In addition, we discussed problems due to coarse sampling of the range of possible orientations. We showed that with proper sampling the complexity of the network becomes cubic in the size of the image. Finally, our discussion revealed that the method has difficulties at curve junctions. This leaves unclear how to make the transition to grouping in complex images.

Chapter 3

Extracting Salient Contours Using Shortest Paths[1]

In this chapter we introduce a new approach to the problem of identifying salient curves in images. Our method is based on finding shortest paths in graphs. The method is designed to locate salient areas in complex images, while simultaneously making available the curves that make those areas salient. We are specifically interested in a mechanism for global saliency, which was illustrated in Figs. 1-3 and 2-1.

One such method is Shashua and Ullman's Saliency Network. The Saliency Network guarantees finding globally salient curves according to a saliency measure based on length and straightness. The last chapter showed, however, that Shashua and Ullman's method can have difficulties at junctions and their saliency measure can change its preferences when the image curves are uniformly scaled. We carefully consider these issues in the design of our method. As with the Saliency Network, our method will place significant restrictions on the possible saliency functions. Given these restrictions, we construct a measure that as closely as possible exhibits the desired behavior.

Before describing our method for saliency, then, it may be beneficial to discuss what we would like in an ideal measure of curve saliency. In particular, several

[1]This chapter is joint work between the author and Ronen Basri.

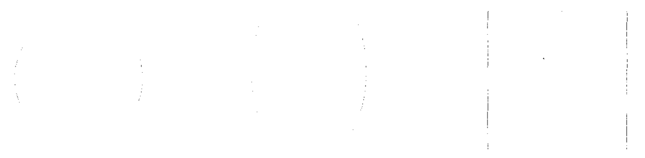


Figure 3-1: The above objects have constant curvature, changing curvature, and sharp changes in orientation. Is it clear which is the most salient?

potentially desirable properties were suggested over the course of Chapters 1 and 2. First, we may wish our measure to take advantage of closure. If the contours in the models surround closed regions, then closure imposes a strong constraint on saliency.

A second cue for a saliency measure is size. Large objects seem to attract humans' attention more than small objects, and, moreover, it is common in images for the objects of interest to be relatively large. Of course, this criterion will not help to find a person standing in a forest, but there are many circumstances in which it may be useful, such as in avoiding an oncoming truck.

Another potentially worthwhile property is a preference for smoothness, where smooth here refers to "non-wiggly." Note that this relates to the number of inflections in the curve and differs from the mathematical notion of smoothness defined by discontinuity in orientation. One such measure of smoothness is convexity. Normalizing by size, this measure would say that all convex shapes of the same size are equally salient (see Fig. 3-1). The basic idea is that winding curves with many points of inflection should be less salient than curves that are purely convex. Our justification for this criterion is that the man-made objects in which we are interested tend to have smooth contours, whereas, a curve through a series of random background segments would tend to be more wiggly.

In addition to smoothness, a saliency measure should prefer that curves be connected, that is, to contain as few gaps as possible. Within this preference for few gaps, there is a significant issue as to where on a curve the gaps should occur. Specifically, on gaps we also want the measure to prefer straightness, because it has been established that humans are better at filling-in gaps when they occur on straighter sections of curves (Biederman [11]). One possible explanation is that it is less am-

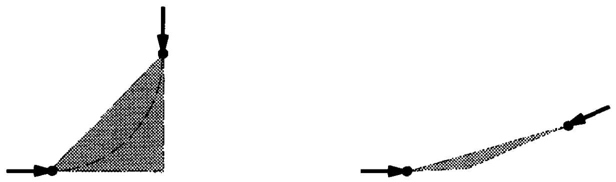


Figure 3-2: The area of no-inflection curves that could explain a pair of points tangents becomes smaller as the tangents become more collinear.

biguous how to fill in a straight gap, making them easier to complete (see Fig. 3-2). This explanation supposes that humans prefer smooth completions and that, to fill-in a gap, they only consider the position and tangent information the gap's endpoints.

Biederman on the other hand explained humans' apparent preference for straighter completions by noting that collinearity is a special property when viewing line segments. The explanation incorporates the notion of *non-accidental properties* (see Lowe [64] for an introduction and also Jacobs [55] for an interesting analysis). In particular, collinearity is invariant to projection, and so two collinear 3D line segments always produce collinear image segments. In contrast, two 3D line segments in general position are very unlikely to produce collinear 2D line segments when projected from an arbitrary viewpoint. The fact that collinearity is special in this sense may explain why people are better at filling gaps that are straighter.

A second-order issue with gaps concerns how the total gap length is distributed when the curve is equally straight (same curvature) everywhere. In particular, there is evidence that humans are better at filling-in several small gaps than one large gap of the same total length (Elder and Zucker [28]). On the overall issue of gaps, we would like a saliency measure to prefer gaps that are shorter and straighter.

Lastly, it is preferable that the measure ranks curves in a way that is invariant to similarity transformations (rigid transformations and scalings). Note that in this property we are not requesting absolute invariance, but only relative invariance in the rankings of curves, since this is all that is important for grouping. In fact, to satisfy the length requirement, the saliency measure should *not* be invariant to scale.

Notice that straight line segments of any length are related by scale, as are all circles of any size. If the measure is scale invariant it cannot possibly distinguish between line segments of different lengths or circles of different sizes. Thus we must require scale invariance in a weaker sense. Let $S(\Gamma)$ in general be the saliency of a curve Γ. Then we want

$$S(\Gamma_1) < S(\Gamma_2) \implies S(z\Gamma_1) < S(z\Gamma_2), \tag{3.1}$$

where $z\Gamma_1$ and $z\Gamma_2$ are respectively the scaled versions Γ_1 and Γ_2 ($z > 0$). Below we choose a measure for gap-free curves that obeys this *rank scale invariance.*

Although it may be possible to describe a function that has all of these properties, such a function may be difficult to optimize efficiently on complex images. By "complex" we mean that image curves can be fragmented, noisy, and can intersect, all of which make junctions difficult to handle. In Chapter 1 we listed the following two principles. Foremost we want our saliency mechanism to be able to find curves that are salient globally in complex images. In addition, the time required by the saliency mechanism to identify a salient curve generally should be on the order of the length of curve, or at least not much more than the length. This is particularly true if the curve is alone in the image. When there is a significant clutter in the image, causing the curve to be less salient, then more time is permissible.

Our basic problem is to develop a method which offers an effective tradeoff between the desired properties of the saliency measure and the saliency mechanism. Shashua and Ullman's method offered a tradeoff which we believe is too restrictive in the saliency measures it can optimize and in its ability to cope with junctions. We have developed a new approach which we think offers a more attractive tradeoff. In the first section of this chapter, we present our method, which is based on shortest-paths algorithms. In the following section, we compare the method to Shashua and Ullman's to elucidate the differences. Then in Section 3.3 we discuss the class of saliency functions the shortest-paths method can implement, and in Section 3.4 we develop a function that is a member of this class and that satisfies most of the desired properties for a saliency measure. Section 3.6 describes experiments that test the method in various situations. The experiments demonstrate that the method is capable of finding salient curves in complex images.

3.1 Our Method

Our method proceeds by applying shortest-paths algorithms to a graph constructed from the image. In a shortest-paths problem, we are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is a set of edges, and we are given a weight (or cost) function $w : \mathcal{E} \rightarrow \mathbf{R}$ mapping edges to real values. The cost of a path is the sum of the weights of the edges along the path, and a shortest path between two vertices is any path of minimal cost between those vertices. We let the graph $\mathcal{G}$ have a vertex for each orientation element, and a pair of directed edges between every two vertices that correspond to neighboring elements. The cost function is given in Section 3.4. Appendix A presents the well-known Bellman-Ford algorithm for finding the shortest paths from one vertex (source) to every other. By repeating this "single source" shortest-paths computation from any chosen set of vertices, we can find the optimal paths from those vertices to every other.

To date, we have focused on computing saliency maps for closed curves (though in Section 3.7, we discuss how we might produce a saliency map for open curves as well). For each vertex, then, there is one optimal closed curve, which is the shortest path that leads from the vertex back to itself (excluding the trivial path of length 0). For the saliency of a vertex, we use the inverted cost of the optimal closed curve through that vertex.

Like Shashua and Ullman, we define the "saliency map" of an image to be an image in which the intensity value of each pixel is proportional to the score of the most salient curve emanating from that pixel. We could produce this map by computing the shortest paths for every possible vertex (element) and then setting the saliency of a pixel to be the largest saliency among the elements connected to that pixel.

In actuality, we first detect a subset of edge points and estimate the tangent vectors at those points. Then we use these oriented points as the sources. Although the resulting saliency maps will be much more sparse, there is little actual loss of information, since every salient contour should have at least one non-gap element on it. In practice, we run the single-source shortest-paths computation at most once from each of the p image pixels, and typically much less.

3.2 Comparison to Shashua and Ullman's Method

Like Shashua and Ullman's network, our method extracts the long and smooth curves in the image while performing gap completion. In addition, our method overcomes some of the difficulties encountered by the Saliency Network. Specifically, our saliency measure avoids high rankings of short open curves when these connect to closed curves, and its rankings of curves are invariant to scale changes. Furthermore, the method can deal with junctions, and it converges without having to follow a geometric series. Additionally this is performed in a run-time complexity that is comparable to Shashua and Ullman's.

Fundamentally, however, there are basically two properties of our approach that differentiate it from Shashua and Ullman's. The first is that we find the optimal curve from each element separately, and the second is that we minimize instead of maximize. Finding the optimal curve starting from each element separately allows our method to choose between curves according to how well they complete a path from that starting element. In the Saliency Network, there is no notion of a starting element, and this forces the Network to choose between equally good curves at junctions. To see this, it is useful to observe that the Saliency Network basically implements a shortest-paths algorithm, specifically, the Bellman-Ford single-source shortest-paths algorithm (given in Appendix A). For every iteration, the Saliency Network performs the operation in Eq. 2.4 at every element. For the Bellman-Ford algorithm, we equivalently would perform the following operation at every edge (v_j, v_i):

$$\begin{aligned} &\textbf{If} \quad \widehat{\Phi}(i) < F(i,j,\widehat{\Phi}(j)), \\ &\textbf{then} \quad \widehat{\Phi}(i) \leftarrow F(i,j,\widehat{\Phi}(j)), \end{aligned} \tag{3.2}$$

where $F()$ is given in Eq. 2.5 and would be computed using state variables stored at vertices v_i and v_j and at edge (v_j, v_i). In Eq. 3.2 we used "<" instead of ">" to be consistent with taking the maximum in Eq. 2.3, though to compute *least*-cost paths, we alternatively could have negated the values of the state variables. The difference in the results of the Bellman-Ford and Shashua and Ullman's algorithms arises from how $\Phi(i)$ is initialized. In the Bellman-Ford algorithm, initially $\widehat{\Phi}(i) = 0$ and $\pi_0 = v_0$, and for all other vertices we set $\widehat{\Phi}(i) = \infty$. In the Saliency Network, initially $\Phi(i) = \sigma_i$ for all i. This shows that, by simply changing how the edge weights are initialized, a shortest-paths algorithm can optimize any extensible measure (extensibility was discussed in Chapter 2).

The Saliency Network's difficulty with junctions is captured by the requirement of extensibility. Recall Fig. 2-10. If v_2 is an element on the most salient curve emerging from v_1 (denoted by Γ_1) then, at convergence, the subcurve of Γ_1 from v_2 must be the most salient curve emerging from v_2. This restriction does not apply for our measures. It is possible in our method that the most salient curve emerging from v_2 will not be a subcurve of Γ_1. The framework proposed in this chapter, therefore, strictly extends the set of possible functions that can be used to express the saliency of curves.

Nonetheless, this broader class of measures does not come without a cost. Finding the shortest paths in an arbitrary graph can take asymptotically longer than the convergence time for the Saliency Network. For example, to use the Bellman-Ford algorithm as described above, we would have to run the Saliency Network many times, once for every starting vertex. To remedy this problem, we restrict the edge weights in the graph to be positive, which allows for using much faster shortest-paths algorithms. Our method then obtains a worst-case running time equal to that of the Saliency Network (see Section 3.5).

Although both our method and Shashua and Ullman's use positive weights on the edges, we minimize a function over these weights whereas Shashua and Ullman maximize. Consequently, our method converges after going around a closed curve once, since it is always more costly to go around more times. Additionally, there is no need for our function to form a geometric series, since there is no such requirement for convergence. Nevertheless, the overriding advantage of our approach is that we can handle junctions, by allowing each optimal curve to make an independent set of choices.

3.3 Optimal Substructure

Shortest-paths problems generally exhibit the property of *optimal substructure* [24].[2] For every pair of vertices, v_1 and v_2, the optimal path between this pair of vertices (denoted by Γ^*) must contain the optimal path between any pair of vertices on Γ^*. There exist functions that do not obey this restriction. For example, the measure proposed in [95, 19], $\Psi(\Gamma) = lk$, where l denotes the length of Γ and k denotes its total squared curvature, does not have optimal substructure. To see this, consider the two curves on Fig. 3-3. The curve Γ_1 is composed of a circular arc connecting the elements v_1 and v, followed by a straight line connecting the elements v and v_2. The curve Γ_2 is composed of the curve of least energy connecting the elements v_1 and v, followed by the same straight line between v and v_2. Denote by l_c and by k_c the length and total squared curvature of the circular arc in Γ_1, and by l_e and by k_e those of the curve of least energy in Γ_2. Denote by d the length of the straight line between v and v_2. First, notice that $k_e < k_c$ (because k_e is the energy of the least-energy curve between v_1 and v). It is shown in [95, 19] that a circular arc always minimizes the measure lk; therefore $l_c k_c < l_e k_e$. Depending on the value of d we can make $\Psi(\Gamma_1)$ be either less or greater than $\Psi(\Gamma_2)$. When $d = 0$,

$$\Psi(\Gamma_1) = l_c k_c < l_e k_e = \Psi(\Gamma_2).$$

When $d > 0$ we obtain that $\Psi(\Gamma_1) = (l_c + d)k_c$ and $\Psi(\Gamma_2) = (l_e + d)k_e$, from which we obtain that $\Psi(\Gamma_2) < \Psi(\Gamma_1)$ when

$$d > \frac{l_c k_c - l_e k_e}{k_c - k_e} \tag{3.3}$$

[2]Brady et al. used the term "extensibility" for this property [15].

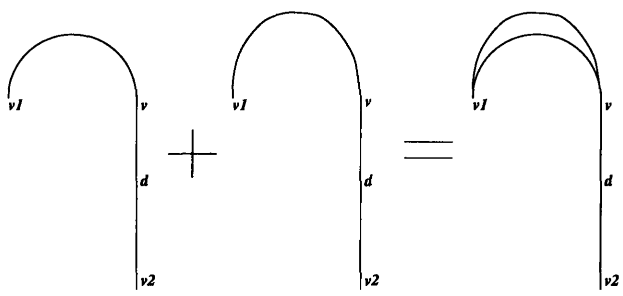


Figure 3-3: Left figure: $\Gamma_1 + \Gamma_2$. Middle: Γ_1. Right: Γ_2. Γ_1 is composed of a circular arc (between v_1 and v) and a straight line (between v and v_2). Γ_2 is composed of the curve of least energy between v_1 and v and a straight line (between v and v_2).

Therefore, when the straight line between v and v_2 is sufficiently long, the optimal curve between v_1 and v_2 will be Γ_2. Nevertheless, the optimal curve between v_1 and v will still be the circular-arc section of Γ_1 which is not part of Γ_2. Consequently, the measure $\Psi(\Gamma) = lk$ does not exhibit optimal substructure. It should be noted however that by increasing significantly the number of vertices in the graph it is possible to compute discrete approximations of this measure. This issue, however, extends beyond the scope of this work.

3.4 Saliency Measure for Closed Curves

For each vertex there is one optimal closed curve, which is the shortest path that leads from the vertex back to itself (excluding the trivial path of length 0). For the saliency of a vertex, we simply use the inverted cost of the optimal closed curve through that vertex, that is, $S(\Gamma) = \frac{1}{\Psi(\Gamma)}$. For the remainder of the section, we concentrate on developing the cost function $\Psi(\Gamma)$.

3.4.1 Gap-free curves

According to our criteria, we wish the rankings of curves by the saliency measure to be invariant to similarity transformations. Furthermore, we wish to prefer curves that are large and smooth. For a gap-free curve Γ, one measure which partly meets these conditions and can be optimized by shortest paths is simply the total squared curvature k:

$$\Psi(\Gamma) = k \tag{3.4}$$

This measure obviously is indifferent to rigid transformations.

As for scale, the total squared curvature is inversely dependent on scale; that is, if we scale a figure by z, the total squared curvature becomes k/z. To see this, consider a curve $\Gamma(t)$ of length l and curvature $\kappa(t)$, where $0 \leq t \leq l$. The total squared curvature is $k = \int_0^l \kappa^2(t)dt$. Suppose now that we scale the picture by z. The scaled curve has length zl, and the curvature at position $\tau = zt$ becomes

$$\kappa_z(zt) = \frac{\kappa(t)}{z}. \tag{3.5}$$

Thus, the total squared curvature becomes

$$k_z(0, zl) = \int_0^{zl} \kappa_z^2(\tau)d\tau = \int_0^l \frac{\kappa^2(t)}{z^2}(z\,dt) = \frac{k(0,l)}{z}. \tag{3.6}$$

Consider for example a circle of radius r, so that its total squared curvature is $k = 2\pi/r$. If we now scale the circle by z, its radius becomes zr, and its total squared curvature becomes $k_z = 2\pi/(zr) = k/z$.

The inverse dependence of the measure on scale implies that large closed curves would in general be judged more salient than short closed curves. For example, the saliency of a circle of radius r is

$$S(\text{circle}) = \frac{1}{\Psi(\text{circle})} = \frac{r}{2\pi} \tag{3.7}$$

and so the saliency grows with the circle's perimeter. Thus the measure k leads to preferring larger circles, and it is rank scale invariant. Of course, since it is possible to increase the length of a curve faster than its curvature (e.g., Fig. 3-4), the measure does not strictly prefer larger shapes.

The total squared curvature is often used as a smoothness measure (e.g., [12, 56, 82, 43]). In actuality, however, k is a measure of straightness, since it penalizes any deviation from straightness and only straight lines receive zero cost. For example, a

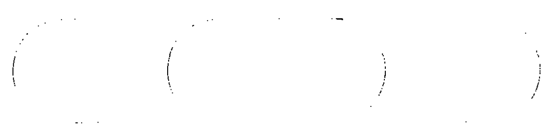


Figure 3-4: Although the total squared curvature tends to prefer larger shapes because it decreases with scale, it does not distinguish between these two shapes.

perfectly smooth circle is penalized more than a line. For our purposes, this creates a preference for larger circles, which agrees with the size criterion. On the other hand, the area of a region would be a better indicator of its size. A narrow, elongated shape (as in Fig. 3-4) can have significant size but large cost due to high-curvature turns. Shashua and Ullman also proposed using the total squared change in curvature as a smoothness measure [82], which has zero cost on any circle. Nevertheless, this measure penalizes perfectly smooth ellipses over circles.

As discussed above, for smoothness we may want to measure the convexity of the curve. One such measure is the total absolute curvature, which also has been used before for smoothness (e.g., [69, 73]). For closed curves, the total absolute curvature reaches its minimum value of 2π on all convex shapes, including circles, ellipses, and squares. In terms of our saliency measure, one unfortunate consequence is that the total absolute curvature has no preference for larger shapes. In general, it appears very difficult to construct a measure that simultaneously prefers convex and large shapes, is rank scale invariant, and can be optimized locally. Leaving this as an open problem, for now we shall use k as our measure of size and smoothness.

As a result, the method can have problems on curves with corners (e.g., the square in Fig. 3-1), since across discontinuities in orientation the total squared curvature is infinite. Corners are similarly a problem for the Saliency Network. One possible way in which we could circumvent this problem is to detect corners bottom-up, as we do for edge and gap elements, and then use a different penalty term at the corner points. As yet, we have not explored this possibility, and so our method is intended for smooth curves in which orientation changes slowly.

3.4.2 Curves with gaps

When there are gaps in a curve, we want the gaps to be as straight and short as possible. One possibility is to use the total squared curvature again. Consider a gap that can be completed by a circular arc of radius r and length ℓ. Then over this gap $k = \frac{\ell}{r^2}$, which increases with length and decreases with straightness, as desired. Unfortunately, on perfectly straight gaps this measure is 0, regardless of the gap's length. To alleviate this problem, we add a term for the length of the gap; specifically, on gaps we use $k + \frac{g}{r_m^2}$, where r_m is a constant. This term penalizes longer straight gaps, but we have lost rank scale invariance. We can, nonetheless, obtain a similar behavior on circles by choosing r_m large enough, as follows. On an all-gap circle of radius r,

$$\Psi(r) = k + \frac{g}{r_m^2} = 2\pi\left(\frac{1}{r} + \frac{r}{r_m^2}\right). \tag{3.8}$$

This function decreases from infinity at $r = 0$ and grows back towards infinity as r becomes large. Taking the derivative,

$$\frac{d\Psi(r)}{dr} = 2\pi\left(\frac{-1}{r^2} + \frac{1}{r_m^2}\right). \tag{3.9}$$

Solving $d\Psi(r)/dr = 0$ gives $r = r_m$. So $\Psi(r)$ strictly decreases to $\Psi(r_m) = 4\pi$ at $r = r_m$ and then strictly increases. Consequently if we set r_m to be at least the radius of largest possible circle in the image, then the function will always decrease as r grows.

In terms of the criteria for gaps mentioned above, our gap measure prefers fewer gaps overall and it prefers gaps to lie on the straighter sections of a curve. On the issue of preferring several small gaps versus a few large gaps of the same total gap length, the measure exhibits no preference. So on gaps our saliency measure meets the first two of the three criteria we suggested.

Putting the two terms together gives

$$\Psi(\Gamma) = k + \lambda\left(k_g + \frac{g}{r_m^2}\right), \tag{3.10}$$

where k_g is total squared curvature over the gap part of Γ, g is the total gap length, and λ is a straightness constant greater than one. The larger λ is, the straighter the gaps must be. Using this measure, an upper bound on the cost of the best cycle through any element is the cost of the largest all-gap circle through that element that fits inside the image.

Representing the measure in Eq. 3.10 a little differently, we can notice a resemblance to the measure used by Shashua and Ullman. Define

$$\tilde{\Phi}(\Gamma) = e^{-\Psi(\Gamma)} = e^{-\left(k+\lambda\left(k_g+\frac{g}{r_m^2}\right)\right)} = e^{-\left(k+\frac{\lambda}{r_m^2}\left(r_m^2 k_g+g\right)\right)} \tag{3.11}$$

Taking the exponent did alter the ranking order of curves, and negating the cost function turned the minimization into maximization. Letting $\rho = e^{-\frac{\lambda}{r_m^2}}$, we obtain

$$\tilde{\Phi}(\Gamma) = \rho^{r_m^2 k_g+g} e^{-k} \tag{3.12}$$

Now if we remove the repeated summing for the geometric series in Eq. 2.6, we get

$$\Phi(\Gamma) \;=\; \rho(0,l)C(0,l) = \rho^g e^{-k}, \tag{3.13}$$

which differs only in missing the k_g term for gaps. In our scheme the k_g term is essential in order to prefer straighter completions for gaps. Without this term, an upper bound on the cost of a best cycle through any element would be the cost of the shortest all-gap cycle; this shortest cycle is trivially two elements long (starting from the element and then returning by making a 180^o turn).

Given two arbitrary orientation elements, we can use our shortest-paths framework to find the curve Γ^* that connects the two elements and minimizes the measure $\Psi(\Gamma)$. To do this we first construct a graph $\mathcal{G}$ in which the sum of edge weights amounts to $\Psi(\Gamma)$. We let $\lambda_{ij} = 0$ if edge (v_i, v_j) is actual, and $\lambda_{ij} = \lambda$ if edge (v_i, v_j) is virtual. And for every edge (v_i, v_j) we assign the weight

$$w(v_i, v_j) = k_{ij} + \lambda_{ij}\left(k_{ij} + \frac{s_{ij}}{r_m^2}\right), \tag{3.14}$$

where k_{ij} denotes the squared curvature produced by the two orientation elements v_i and v_j, and where s_{ij} denotes the arc length. Then the cost $\Psi(\Gamma)$ of a curve Γ is the sum of the weights along the path corresponding to Γ. A shortest-paths algorithm will then minimize this sum over all curves traveling between the two given elements. Of course, the cost of a path obtained by such an implementation provides only a discrete approximation to the true cost $\Psi(\Gamma)$.

3.5 Complexity Analysis

An efficient algorithm for finding the single-source shortest paths in a graph with no negative-weight edges is Dijkstra's. Dijkstra's algorithm can be implemented straightforwardly using a binary heap, which is what we used in our experiments. For a graph with V vertices and E edges, the running time is $O(E \log V)$ [24]. Using the same notation as Chapter 2, the graph $\mathcal{G}$ has pb vertices, one per orientation element. Also, there is an edge for every pair of elements that meet at the same pixel in the image. Each element that is not on the boundary of the image, then, has b edges emerging from it. In total, this gives bV edges minus the missing boundary edges, which for an image of dimensions $w \times h$ is precisely $E = b(pb - (b-2)(w+h-4) - (3b-4))$. Since w and h are $O(\sqrt{p})$, E is $O(pb^2)$. To produce the saliency map, we run the single-source computation separately for each pixel whose gradient magnitude is large enough. In the worst case there would be one active element for all p pixels in the image. Substituting for V and E, this gives a worst-case running time for our method of $O(p^2b^2(\log p + \log b))$, using Dijkstra's algorithm with a binary heap. If we instead use a Fibonacci heap, the running time would be $O(p(V \log V + E)) = O(p^2b(\log p + b))$. Moreover, it is possible to do even better because our graph is locally connected, using the recent, linear-time algorithm of Klein et al. [57]. As a consequence the time for our method is bounded asymptotically by $O(pE) = O(p^2b^2)$, which is the same as the time for Shashua and Ullman's method.

It is worth noting that for most images, we obtain a substantial speedup by running the algorithm only from the pixels in an edge map. If there are a edge pixels, then the running time becomes $O(apb)$. The space required to store the graph $\mathcal{G}$ is $O(V+E) = O(pb^2)$, which is comparable to the space required to store the saliency network, $O(pb)$. To additionally store the top a optimal paths, we would require space for about $O(a\sqrt{p})$ nodes. This estimate of $\sqrt{p}$ for the path length comes from the fact that the optimal closed curves are simple (no loops) and tend to have slowly changing orientation.

It should be noted that in our method, as in Shashua and Ullman's method, the issue of discretization is critical for obtaining reasonable performance of the system (see the discussion in Section 2.4). A faithful implementation requires $b = O(\sqrt{p})$ orientation elements at every pixel. The overall time complexity becomes $O(p^3)$, which reduces to $O(ap^2)$ for binary images. The space requirements become $O(p^2)$ to store the graph and $O(p^3)$ to store the optimal paths between all pairs of elements. Note that, like Shashua and Ullman's network, our method can be implemented on parallel architectures to obtain a significant reduction in time complexity.

For the case of a single, gap-free curve in the image, our method is very efficient. In particular, Dijkstra's algorithm uses a best-first search to find the optimal paths. For any starting node, we end the search as soon as the optimal cycle back to the start is found. So in the case of a lone curve, only the nodes on the curve are examined. For an isolated curve of length ℓ, then, the running time for Dijkstra's single-source algorithm reduces to $O(\ell b \lg \ell)$. If fragmentation is added to the isolated curve, the time for the method should degrade slowly, for Dijkstra's algorithm will have to explore a small area around the curve as well. When the complexity of the image increases, the time for the method goes up considerably.

Lastly, we mention some possibilities for speeding-up the method in practice. First, once a salient closed curve is found, that curve need not be recomputed later, because a path between any two nodes on the closed curve that does not include the start node will always be optimal. When there is a salient closed curve in the image, we expect that the optimal cycles for elements that are on or near the salient closed curve will pass through that salient closed curve. By re-using the already-computed optimal paths, we might obtain substantial savings. For the special case of a single curve in the image, we would only have to run one single-source computation.

Another way we could speed-up the computation is to prune high-cost searches through background clutter. The idea is that clutter in the image will lead to long closed curves that pass through a large number of gaps, because the term $k_g + \frac{g}{r_m^2}$ prefers long, straight curves. As mentioned above, a bound on the worst optimal cycle through any element is the largest all-gap circle through that element that fits inside the image. Then if we have an estimate of the worst cost associated with the salient cycles in the image, we can prune searches when the cost of the best path becomes significantly greater. One way to estimate the cost may be to look for a salient cycle by choosing among the starting elements randomly.

3.6 Experiments

We first tested our method on simulated images to illustrate the points made above concerning the saliency measure and the mechanism. Then we tested the method on real images. We implemented our saliency method using Dijkstra's single source algorithm to optimize the saliency measure at every source element. For an implementation of Dijkstra's algorithm, we made use of a library of shortest-paths routines (called SPLIB) by Cherkassky, Goldberg, and Radzik. All of the reported execution times are for C code on a SPARCstation 5 with 32M of memory. Unless otherwise stated, we used twenty-four orientation elements per pixel, which gave the radius of the largest discrete circle to be $r = 22.78$ (see Section 2.4). To satisfy the condition that r_m is at least the radius of the largest circle, we set $r_m = 31.62$ (so we multiplied the gap length by $1/r_m^2 = .001$). In all cases we used $\lambda = 10$. The real images were smoothed with a Gaussian of standard deviation 1 before processing.

The first example demonstrates the operation of the method at junctions. The leftmost picture in Fig. 3-5 contains two circles, a small circle with no gaps, and a larger circle with a gap at every other element. After running the shortest-paths method, the most salient element (for which $S = 1.782$) was on the smaller circle. The lower-left picture shows the most salient curve, which is the smaller circle. Even

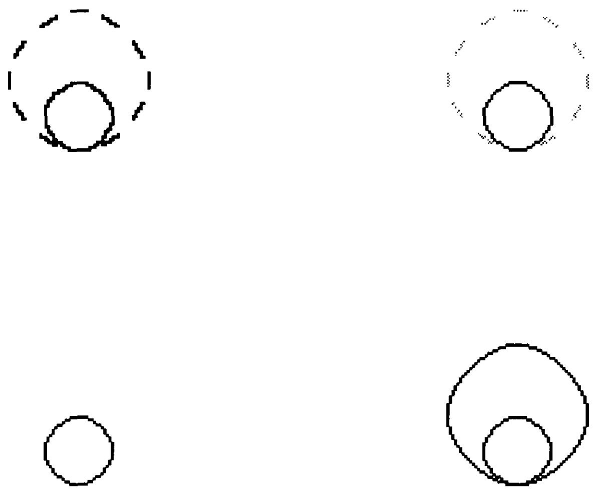


Figure 3-5: Shortest-paths method at image junctions. Upper left: the input graph showing two circles meeting at a junction. Upper right: the shortest-paths saliency map. Lower left: the most salient curve. Lower right: the top two most salient curves. The most salient curve has saliency 1.782.

though our saliency measure prefers larger circles, the smaller circle is more salient in this case because of the gaps in the larger circle. To obtain the second most salient curve, we first chose the next most salient element in the image that was not any of the elements on the most salient curve. Then we used the optimal cycle through this chosen element, which was the larger circle. The lower-right picture displays both optimal cycles.

For comparison to Shashua and Ullman's method, Fig. 3-6 shows the results of the Saliency Network on the same input. To get the optimal curves, we first traced the curve from the most salient element (for which $\Phi = 131.3$), which gave the smaller circle. Then to get the next most salient curve, we repeated what we did for the shortest-paths case. That is, we picked the next most salient element after excluding all the elements on the optimal curve, and then we traced the curve from this element (third picture from the left in the figure). This next most salient element

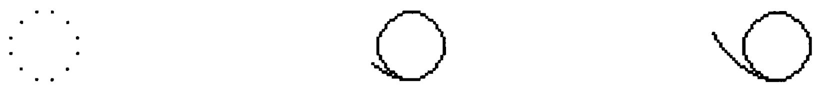


Figure 3-6: Shashua and Ullman's method at image junctions. Left to right: The most salient curve (only the tail points of the elements are shown), the second most salient curve (which starts from the endpoint of the open curve attached to the circle and then goes around the circle), and the third most salient curve (which again starts from the endpoint of the open curve and proceeds around the circle). The saliencies of the top three curves are 131.3, 21.10, and 8.426.

was an active element on the larger circle and was separated from the junction by a single gap element. Tracing the curve that followed the chosen element gave the smaller circle again. Repeating the process to get the third most salient curve gave the second most salient curve again, except that it was two elements longer (right picture the figure). From the analysis in the last chapter, elements near the most salient cycle tend to merge with the cycle and then draw their saliencies from it. The saliencies of these elements attenuate as they become further away. To obtain the larger circle, we would have to trace the curve from the least salient active element.

On the two-circle example in Fig. 3-5-top-left, in which there is no clutter, one may expect a saliency mechanism to run very fast. On this example, the running times until convergence were 52 seconds for the shortest-paths method and 47 minutes for the Saliency Network. In fact, the shortest-paths method might have run almost instantaneously if it had memorized the optimal cycles instead of recomputing them from all sources (see discussion in Section 3.5).

In more complex images, in which there is significant clutter, our shortest-paths implementation will be slowed down considerably, since it searches for optimal paths through every background element individually. To better compare the required times for the two methods, we also ran the shortest-paths method on the two-circle image from Chapter 2 that has a cluttered background (Fig. 3-7). There are 200 background

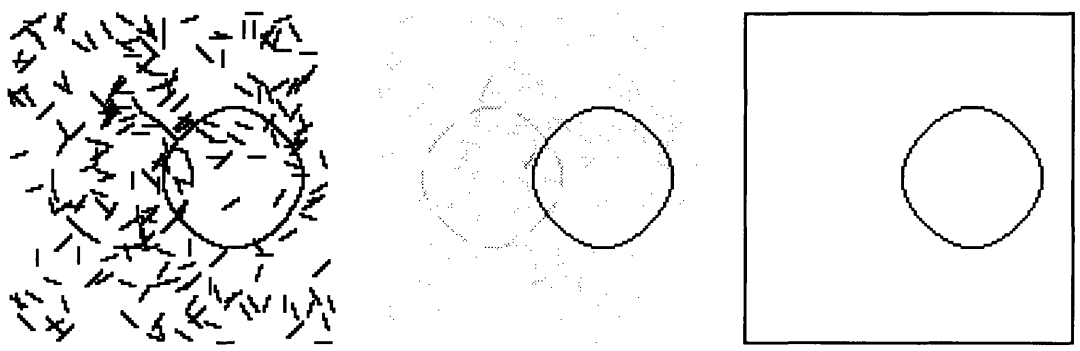


Figure 3-7: Left: two salient circles on a cluttered background of random line segments. The middle and right pictures show the shortest-paths saliency map and most salient cycle (for which $S = 3.627$), respectively.

elements in this image, and the shortest-paths method took 35 minutes. For this image it is unclear that storing the optimal paths would significantly reduce the search, since most of the time is spent searching for optimal paths through the background elements (though there is still the possibility of cutting short such searches). This example is also worse for the Saliency Network because the most salient cycle is now larger (Fig. 2-6-center), and we know from the last chapter that the convergence time grows with the radius squared. As the last chapter reported, on this example the Network requires 1 hour and 25 minutes.

The next example is a synthetic image of a convex shape among random background elements (Fig. 3-8). After running the shortest-paths method, the convex shape is highlighted and the best curve through the shape can be traced. There were 122 active elements, and the method took 19 minutes to complete.

For our first example of a non-synthetic image, Fig. 3-9-left shows a large blob on a field of smaller blobs that are connected together. To detect active elements, we thresholded the image to remove the black background and then placed active elements between all of the surviving pixels. This gave 12954 active elements in the graph. For a starting element, we randomly selected one of the active elements at each surviving pixel. The large blob in the picture appears salient partly because of size and partly because of smoothness. The rest of the image contains less salient cycles which either surround the small blobs or trace out less smooth curves, via the connections between the small blobs. In this situation, there are no gaps which must

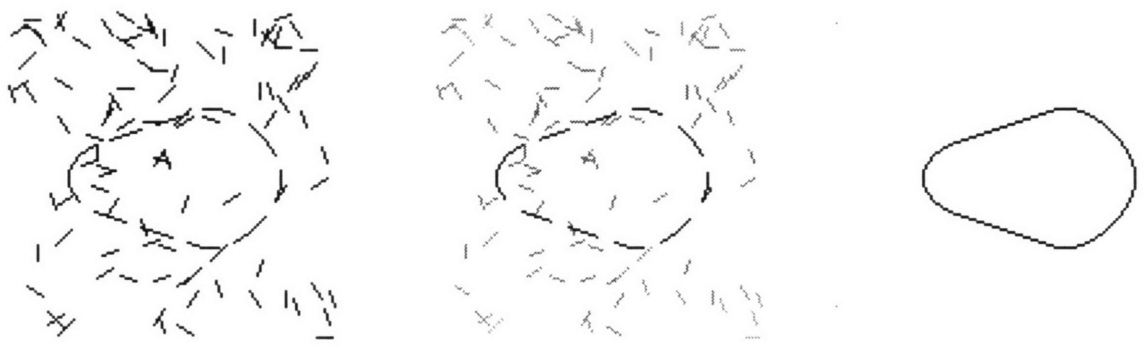


Figure 3-8: Left: smooth, fragmented shaped on a noisy background. Middle: shortest-paths saliency map. Right: the most salient curve ($S = 0.5445$).

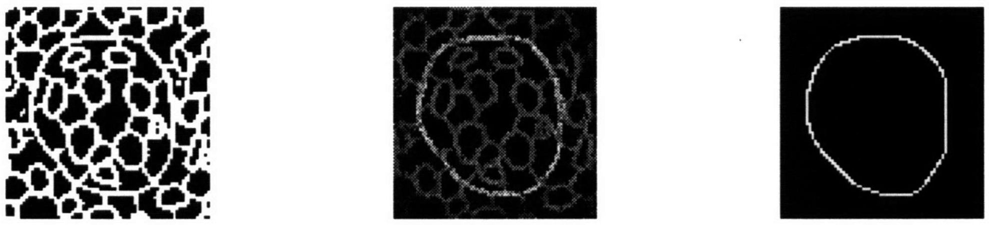


Figure 3-9: Smoothness and size can be important for saliency. On the left, the input image contains a globally salient blob (the image is from [65]). The middle and right images show the shortest-paths saliency map and best closed curve (for which $S = 0.1272$), respectively.

be bridged to find the salient curve, and so the relevant term in the saliency measure is just the total squared curvature k. Using this measure, the method successfully found the salient blob. The running time, however, was 5 hours and 42 minutes. We saw that the network spent a lot of time recomputing paths around the large blob. Storing the optimal paths may speed-up the method substantially.

Finally we try the method on two real images, shown in Figs. 3-10 and 3-11. To compute saliency maps for real images, we first process the intensity image to produce two binary input images, a Canny edge map and a thresholded gradient-magnitude image. The thresholded image is used to generate the graph, and the Canny edges are used to select a reduced number of sources. More specifically, the graph is created by making an element active if the magnitude of the gradient in the

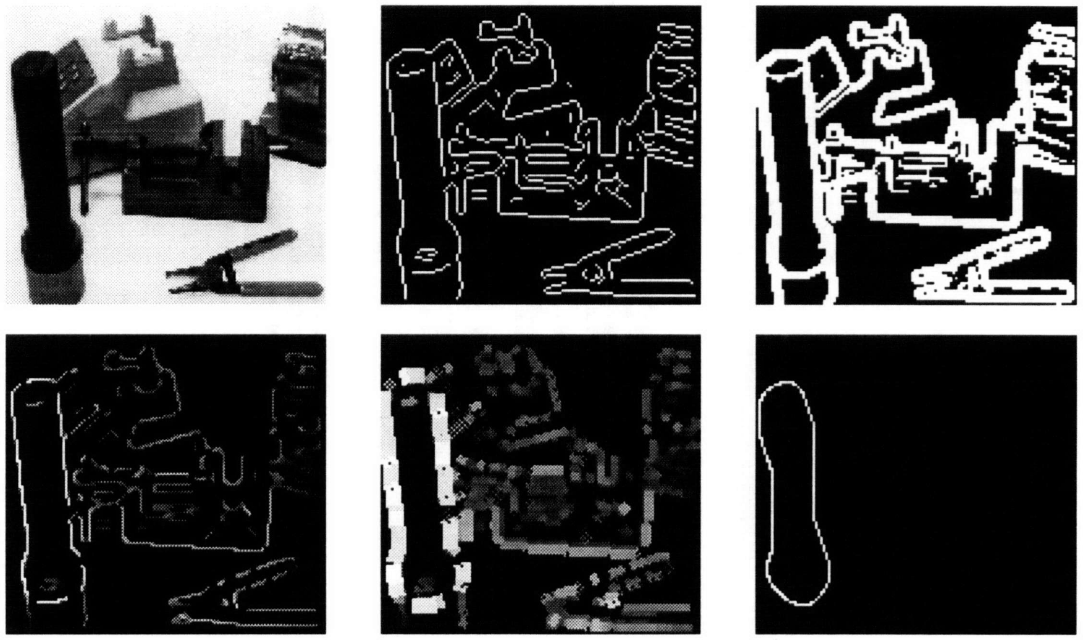


Figure 3-10: Shortest-paths saliency maps for a cluttered scene. The top center image is a Canny edge map, and the top right image was obtained from thresholding with the average gradient magnitude. The bottom row shows two saliency maps and the most salient curve. In the center saliency map, each element is displayed as bar whose width is proportional to its saliency.

direction perpendicular to the element (the directional derivative) is larger than the average gradient magnitude. Then using the Canny edges, source elements are placed between all neighboring edge pixels. Since the Canny edges resulted in thin saliency maps, in which it is difficult to see which elements are brighter, we also provide a thickened saliency map in which each element is represented by a bar whose width and brightness are proportional to its saliency, up to a maximum width of 4 pixels (this bar form of display was used by Shashua and Ullman).

For both real images, the thresholded gradient-magnitude maps contain many junctions and curves to choose from, although they have few gaps. In Fig. 3-10, the method found the flashlight as the most salient curve (saliency 1.137). In the original and the bar saliency maps, the flashlight is brightest. In addition, the saliency map indicates that most of the boundary of the clamp, and parts of the boundaries of the pliers and the telephone keypad are the next most salient areas. From the saliency

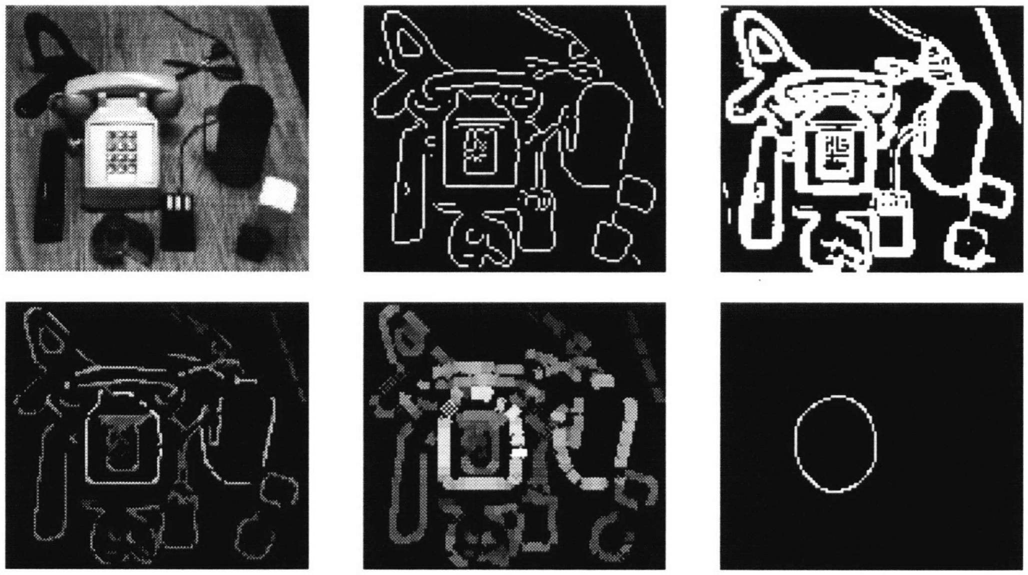


Figure 3-11: Shortest-paths saliency maps for a cluttered scene. The top center image is a Canny edge map, and the right image was obtained from thresholding with the average gradient magnitude. The bottom row shows two saliency maps and the most salient curve. In the center saliency map, each element is displayed as bar whose width is proportional to its saliency.

map, these saliencies of edges are about 0.75 for the clamp, 0.85 for the pliers and 0.9 for telephone keypad. The other edges have saliencies around 0.3 to 0.6.

In Fig. 3-11, the most salient curve lies on the boundary of a region surrounding the telephone keypad. This cycle has saliency 1.813. The cycle is rounded because the method finds the smoothest curve through the edge-thickened map in the top-right image, starting from a edge point in the top-middle image. In addition to the region surrounding the telephone keypad, the telephone handset and the mug are clearly salient as well. From the saliency map, their saliencies are both around 1.45. These results seem promising considering how little information was used (the total squared curvature).

In terms of execution time, however, the results are somewhat less attractive. For Fig. 3-10, the Canny edges gave 1587 source elements and the thresholded gradient-magnitude image gave 22848 active elements in the graph (the image dimensions are 124×117). The program took 13 hours and 18 minutes to run. For Fig. 3-11, there were 1416 sources and 40654 active elements (the image dimensions are 118×104). In this case, the execution time was 8 hours and 33 minutes. As discussed earlier, one possibility for speeding-up the method is to prevent recomputation of the salient paths and to cut off high-cost searches. Another possibility is to run the method on a parallel machine. Parallelism is possible both for running many single-source shortest-paths computations at the same time, and for the implementation of the shortest-paths algorithm itself.

An additional possibility to explore is scale-space techniques. If we find the optimal curves in a small, down-sampled image, the results could be used to narrow the necessary search area at finer scales. With this in mind, we re-ran the shortest-paths method on the same two real images, but down-sampled to size 83×78 for Fig. 3-10 and 80×70 for Fig. 3-11. The resulting saliency maps and optimal curves were similar to before. The running times, however, reduced to 5 hours and 15 minutes for Fig. 3-10 and 2 hours and 17 minutes for Fig. 3-11. The reductions are significant, but still are far from sufficient for a saliency mechanism.

Lastly, we shall demonstrate how an operator can use the shortest-paths output to obtain the most salient curves from or through any point in an image, or between any two points in an image. In Fig. 3-12 we picked an element on the boundary of the stapler, namely pixel $(13, 35)$ at orientation $0°$. This pixel is on the bottom edge of the stapler, close to the leftmost point on the stapler boundary. The program then found the most salient closed curves through the element, shown in the bottom row. The first two pictures in the bottom row compares the closed curves obtained using 16 and 24 orientations per pixel. As expected, the 24-orientation curve is smoother.

In the rightmost picture in Fig. 3-12, we tried an experiment where we incorporated the gradient magnitude at the actual elements into the cost function. Specifically, we let $c_j \in [0, 1]$ be the normalized directional derivative across element v_j. Then for every actual edge (v_i, v_j) in the input graph, we changed its weight from $w(v_i, v_j) = k_{ij}$ to $w(v_i, v_j) = k_{ij} + \lambda'(1 - c_j^2)$, with λ' arbitrarily set to 0.3. The resulting cost function resembles the energy functions used in [12, 56, 66, 69, 73]. As shown in the picture, incorporating intensity compelled the curve to adhere to the

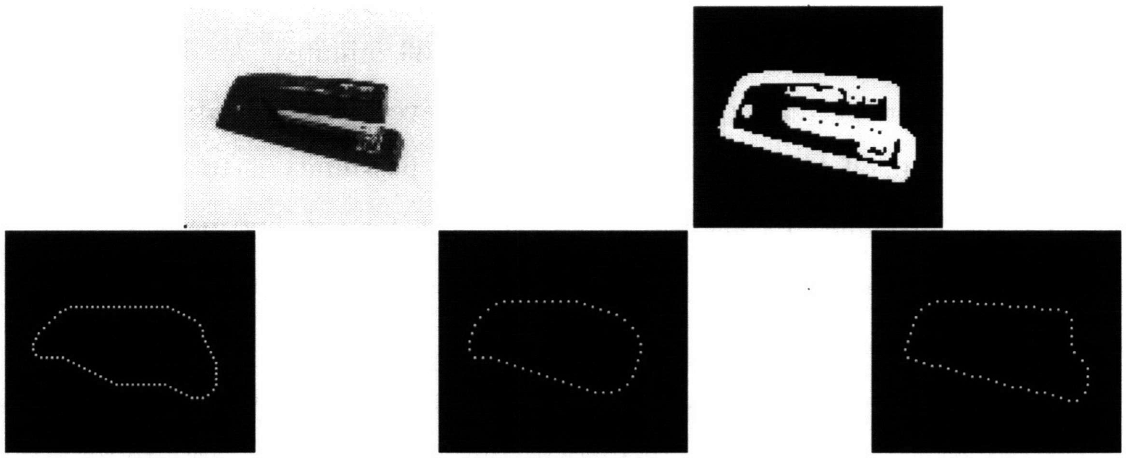


Figure 3-12: User-assisted curve selection, demonstrating the effect of discretization and the use of intensity information. The user picks a point in the image, and then the algorithm finds the most-salient closed curve passing through that point. The upper-left image shows a stapler on a white background. The upper-right image is the input graph obtained by thresholding the directional derivative across each element. For the threshold we used the average gradient magnitude. The bottom row of images show the most salient curves emanating from element (13,35,0°). The left and center pictures show the most-salient cycle using 16 and 24 orientations, respectively; the 24-orientation curve is notably smoother. The final image shows the most-salient cycle using 24 orientations that is obtained directly from the intensity image, suggesting that incorporating intensity may lead to better curves.

actual shape of the object. This application is similar to *active contours* [56], except that we require as input one point on the desired shape and active contours typically require an entire contour near the desired shape.

In Fig. 3-13, the two center pictures show three salient closed curves from the scene image in Fig. 3-10, which were obtained by selecting one point on each of their boundaries. More specific control over the optimal curves can be obtained by supplying two points, which we did in the rightmost picture to obtain a cycle, composed of two open curves, that demarcates the unoccluded part of the telephone keypad. Two points also can be used to obtain open curves that cross over gaps, as described in the figure. In terms of efficiency, after the shortest paths have been computed for all sources, the most salient curves are located in real time.

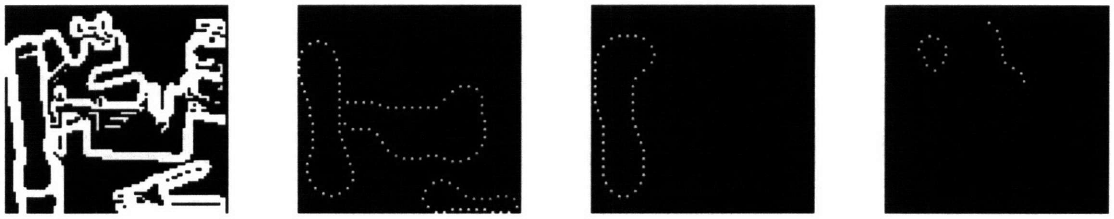


Figure 3-13: User-assisted curve selection. The second picture shows three optimal closed curves obtained from the binary image (see Fig. 3-10 for the original intensity image). Each cycle was obtained by selecting one point on the boundary of an object, specifically, the flashlight (for which the cycle contains 43 elements), the clamp (45 elements), and the pliers (26 elements). The third picture shows the optimal cycle that was found when we selected a point on the boundary of the occluded telephone keypad (46 elements). Observe that this curve includes edges from both the flashlight and the keypad. By specifying two points on the occluded keypad's boundary, we obtained two curves that surround only the keypad, which are displayed in the rightmost picture (6 elements apiece). The final picture also shows an open curve between two user-selected points on the boundary of the phone (9 elements). Notice that this curve fills a gap in the telephone's silhouette. Specifying two points gives the user more control over the discovered curves.

3.7 Proposed Saliency Measure for Open Curves

In this section we describe a possible approach for evaluating the saliencies of open curves. For open curves the situation is more complicated than for closed. It is no longer sufficient to compute the cost of one optimal path from each element, since a shortest path trivially is one element long. Instead, we would compute the shortest paths between all pairs of elements. Then for each element, we choose among all the least-cost paths emerging from it, using a measure that prefers longer paths. More specifically, our open curve saliency map will be a mapping $S : \mathcal{V} \rightarrow \mathbf{R}$ which assigns for every vertex $v_i \in \mathcal{V}$ a value that reflects the saliency of the most-salient open curve emerging from that vertex. Formally, let $\mathcal{C}_i$ denote the set of curves emerging from v_i. The saliency $S(v_i)$ is defined by

$$S(v_i) = \max_{\Gamma \in \mathcal{C}_i} S(\Gamma), \tag{3.15}$$

where $S(\Gamma)$ represents the saliency associated with a particular curve Γ. Our view is that the optimal curve has already emerged from the shortest-paths computation, and we shall define the saliency measure $S(\Gamma)$ in such a way to ensure that optimal curves will be considered more salient than other curves which travel the same distance. Thus, we will be able to compute $S(v_i)$ by considering only the optimal curves. Let $\Gamma^*(v_i, v_j)$ denote the optimal curve from v_i to v_j. Then, $S(v_i)$ is given by

$$S(v_i) = \max_{v_j \in \mathcal{V} - \{v_i\}} S(\Gamma^*(v_i, v_j)). \tag{3.16}$$

For the saliencies $S(\Gamma)$ we choose a measure that decreases with the cost $\Psi(\Gamma)$ of that curve, because the cost function decreases with the smoothness and non-gap length of the curve. To encourage long curves as well, we select a measure based on a notion of "efficiency." Namely, we wish to find those curves that travel the longest distance while maintaining low total squared curvature and low gap length. By using the distance between endpoints instead of the actual curve length, we maintain that the optimal curve (according to Ψ) will always be more salient than any other curve which travels between the same endpoints. For an open curve Γ let d be the Euclidean distance between the endpoints, and as before let k denote the total squared curvature and g denote the total gap length along Γ. Assume Γ contains no gaps ($g = 0$). One measure that encourages longer distances and lower total squared curvature is

$$S(\Gamma) = \frac{d^p}{k}, \tag{3.17}$$

where p is a constant ($p > 0$). This measure is also rank scale invariant. However, the measure becomes unbound when Γ is a straight line, in which case $k = 0$. We shall see that achieving rank scale invariance for gap-free open curves is non-trivial.

To develop a rank scale-invariant measure, notice that the measure dk is scale invariant (in the strong sense). It is zero for all straight lines and 2π for all half circles. This is related to the measure proposed in [95, 19] for finding the best gap completion between two points by optimizing length times total squared curvature. Since the measure dk is scale invariant in the strong sense, it cannot be used as a measure of saliency, but we will use it as a basis for constructing our measure.

Consider the function $S(\Gamma) = 1/d + k$. Clearly this measure decreases with scale. Furthermore, it is scale invariant in the weak sense, since both $1/d$ and k are inversely dependent on z. Since we want a measure that grows with scale (so that a long straight line segment will have a higher score than a short segment), let us invert it:

$$S(\Gamma) = \frac{1}{\frac{1}{d} + k} = \frac{d}{1 + dk}. \tag{3.18}$$

Notice what happens here. The denominator $1 + dk$ is scale invariant in the strong sense. So the only term that changes with scale is the numerator d, and d grows linearly with scale. Consequently $S(z\Gamma) = zS(\Gamma)$. In this way we can define other measures that are rank scale invariant, such as

$$S(\Gamma) = \frac{d^p}{1 + a(dk)^q}, \qquad \text{or} \qquad S(\Gamma) = \frac{e^d}{1 + a(dk)^q}, \tag{3.19}$$

where a, p, and q are positive constants. These measures can be computed directly from the output of the shortest-paths algorithm, since

$$S(\Gamma) = \frac{d}{1 + a\,(d\Psi(\Gamma))^q} \qquad \text{and} \qquad S(\Gamma) = \frac{e^d}{1 + a\,(d\Psi(\Gamma))^q}. \tag{3.20}$$

To conclude, we will compute the values of $S(\Gamma) = \frac{d}{1+dk}$ for a few simple curves. The score of a straight line ($k = 0$) is simply $S = d$, the length of the line. The score of half a circle of radius r ($d = 2r$, $k = \pi/r$) is $S = 2r/(1 + 2\pi)$. So half a circle is always a multiplicative constant below the straight line that connects the same endpoints.

Next we determine the best circular arc on a circle. Let αr be the length of a circular arc, then the distance between the endpoints of the arc is given by $d = 2r\sin(\alpha/2)$ and $k = \alpha/r$. The saliency of a circular arc, therefore, is

$$S(\alpha) = \frac{2r\sin(\frac{\alpha}{2})}{1 + 2\alpha\sin(\frac{\alpha}{2})}. \tag{3.21}$$

Taking the derivative with respect to α and equating to zero, we obtain

$$\cos\left(\frac{\alpha}{2}\right) - 4\sin^2\left(\frac{\alpha}{2}\right) = 0. \tag{3.22}$$

As is expected, since $S(\Gamma)$ is scale invariant, the angular span of the best circular arc is independent of r. Replacing $\sin^2(\alpha/2)$ by $1 - \cos^2(\alpha/2)$, we obtain a quadratic equation in $\cos(\alpha/2)$. Letting $x = \cos(\alpha/2)$, we have $4x^2 + x - 4 = 0$. And its solutions are given by

$$x = \frac{-1 \pm \sqrt{65}}{8}, \tag{3.23}$$

which provides two solutions, namely $x_1 = 0.883$, which corresponds to about $\alpha = 56^o$, and $x_2 = -1.13$, which is not feasible. The saliency of a circle as an open curve, as is reflected by the saliency of the most salient element on the circle, equals the saliency of a circular arc that spans 56^o.

3.8 Conclusion

In this chapter, we have cast saliency as a problem of finding shortest paths in graphs. We defined a graph in which the least-cost paths in the graph amount to salient curves in the image. The complexity of our method is comparable to that of Shashua and Ullman's. Like their method, our method can be implemented efficiently on parallel architectures, though on serial machines the running times of the two methods can be prohibitive. Even so, we mentioned constraints which may significantly decrease the required time. Additionally, our method can be an efficient tool for user-assisted curve selection. Finally, a problem with our method which was carried over from Shashua and Ullman's is the need for relatively massive discretization of orientation in order to achieve good approximations of the non-discretized curves. We intend to address this problem in the future in an attempt to reduce this limitation.

Chapter 4

Uncertainty Propagation in Model-Based Recognition[1]

Given a correspondence between a set of image features and model features, a basic problem in model-based recognition is to evaluate the correspondence and extend it if necessary. Important computations involved in evaluating and extending the correspondence include (1) deciding whether the correspondence provides an accurate alignment of the model the to the image, (2) determining which image features could correspond to each unmatched model feature, and (3) choosing a new match to extend the correspondence. These computations are intertwined with the issue of error propagation, that is, the issue of how uncertainty in a set of matched image features propagates to uncertainty in the predicted image locations of the remaining model features.

We consider an approach where points are used for generating correspondences and where the correspondences are verified using line features as well as points. Given a set of correspondences, errors in the locations of the matched image points propagate to uncertainty in the predicted position of any unmatched model feature. The error in each image point generally can be limited with high probability to some small region. This observation is independent of the distribution of the error within the region,

[1]This chapter is joint work between the author and David Jacobs.

and, when looking for additional matches, can be used to limit the necessary search area in the image. Specifically, we will assume that errors in the sensed or *nominal* locations of the image points are bounded by circles of radius ϵ. Then, as the matched image points move independently around their ϵ-circles, every predicted model feature traces out a region of possible image locations. We call the sets of possible image locations the *uncertainty regions* of the model features, and in this chapter we derive either bounds on these regions or probability distributions on them, depending on our model of error.

To illustrate how the uncertainty regions could be used in a recognition system, we outline a robust version of Huttenlocher and Ullman's 3D-from-2D alignment algorithm [52]. In this algorithm, the 3D models contain both points and line segments. The points are used for generating hypotheses, and the line segments are used for verification. We show below that the uncertainty regions for points are circular and we give a formula for computing these "uncertainty circles." We can use this result to bound the uncertainty regions of line segments by finding the uncertainty circles of their endpoints (as was done by Alter and Grimson [4]). Given a model and a cluttered image of a scene containing the model, repeat the following steps until the model is identified:

1. Hypothesize a pairing of three model and image points.

2. Using the hypothesis, compute the model-to-image transformation and project all of the model line segments into the image.

3. Use Equation 4.26 to compute the uncertainty circles for the two endpoints of every projected line segment. Then construct a tight overestimate of every line segment's uncertainty region from the two uncertainty circles (see Fig. 4-1).

4. Accept or reject the hypothesis, based on the number of line segments for which there exist candidate image segments, and on the sizes of the uncertainty regions (as in [4]). If accepted, return the identified model and pose.

Alter and Grimson demonstrated that this algorithm is expected to be insensitive to

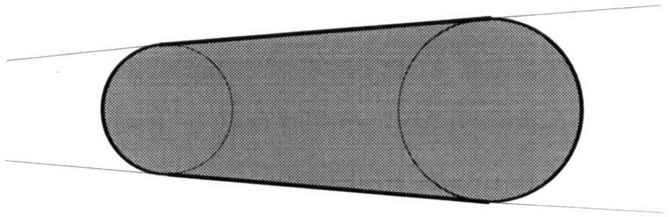


Figure 4-1: Region to search for candidate line segments.

false positives in cluttered scenes.

There are several reasons why it is useful to carefully understand the propagation of uncertainty, as opposed to assuming some small, simple uncertainty region and using it in all cases. First, as we will show, uncertainty regions can vary considerably in size, and may be quite large for the predicted model features, resulting in many candidate image features for each prediction. In particular, grouping techniques commonly find image features that are close together on an object (e.g., [18, 13, 46, 59, 52, 55]), and we will see that this easily can lead to large uncertainty regions. Even when the matched features are far apart in the image, the uncertainty regions of the unmatched points may still be large, due to the depth of the 3D model. Second, both when the image features are nearby and when they are far apart, there are situations in which the pose of the model is unstable, and the uncertainty regions assume surprising shapes. By understanding the propagation of uncertainty, then, we can determine exactly where to look for features, and we can evaluate the stability of the pose produced by the initial correspondence.

4.1 Summary of Results

Given a set of matched image and model points, we determine an unmatched model point's uncertainty region. We handle both scaled-orthographic and perspective projection models. We also consider two different models of error. As mentioned above, we consider a bounded error model, in which we suppose that the error distributions are unknown. In addition, we consider image points detected with errors that have known, independent Gaussian distributions. Given no other information, Gaussians may be the preferred error distribution, since image features are displaced by a sum of error vectors, incurred over a series of processes such as digitization, smoothing, and edge detection. A bounded error model may be useful, however, when errors contain a consistent bias that results in distributions that are significantly skewed from Gaussian. In the first case, we show how bounded error in image points propagates to a bounded uncertainty region describing the possible location of an additional model point. In the second case, we show how Gaussian error in matched image points propagates to an uncertainty region with a Gaussian distribution for an unmatched point.

First we compute the uncertainty regions for unmatched points based on sets of three matched points. We derive a simple linear expression that approximates the relationship between the matched and unmatched points. This relationship allows us to show that, for bounded error, the uncertainty region for a fourth point is circular, and to derive analytic expressions for the center and radius of the circle. For Gaussian error, this relationship implies that the propagated distribution of uncertainty is also Gaussian, and provides analytic expressions for the center and standard deviation. We perform experiments to verify that these expressions are accurate for the amount of error that is of interest in most recognition applications.

We also take advantage of the linear relationship by introducing a new algorithm that allows us to determine the uncertainty region for any number of matched points. To do this we approximate our bounded error regions with convex polygons, and then show that we can use linear programming to derive a convex polygon that describes the uncertainty region of the unmatched model point. We experiment with both synthetic images and a real image to observe the accuracy of the uncertainty regions that we compute, and to determine the extent to which they shrink as we match more points.

Finally, we show how to extend previous work for linear projection models to the cases of scaled-orthographic and perspective projections. Using the linear approximation, we show that we can use Baird's [7] algorithm to tell whether a set of matches between image and model points is geometrically consistent, and that we can apply Cass' [21] and Breuel's [17] algorithms to find, in polynomial time, the model pose that aligns the maximum number of model and image features to within error bounds. We also extend Jacobs' [54] and Sarachik and Grimson's [80] planar alignment algorithms to 3D objects.

4.2 Projection Models

For reference, we review the models of projection to which we refer in this chapter. For perspective projection, we can write the corresponding image position (x, y) of a 3D model point $(\overline{x}, \overline{y}, \overline{z})$ in terms of a 3D, rigid rotation matrix $\mathbf{R}$, a 3D translation vector $\vec{u}$, and a camera focal length f. Letting r_{ij} be the elements of $\mathbf{R}$, we have

$$x = f\frac{r_{11}\overline{x} + r_{12}\overline{y} + r_{13}\overline{z} + u_x}{r_{31}\overline{x} + r_{32}\overline{y} + r_{33}\overline{z} + u_z}, \qquad y = f\frac{r_{21}\overline{x} + r_{22}\overline{y} + r_{23}\overline{z} + u_y}{r_{31}\overline{x} + r_{32}\overline{y} + r_{33}\overline{z} + u_z}, \tag{4.1}$$

where the rows of $\mathbf{R}$ are orthonormal, and where we assume the origin is at the center of projection. When the focal length f is known, there are six degrees of freedom, and consequently three corresponding model and image points are "minimal" to determine the transformation. Given three corresponding points, there exist up to four solutions for the model pose [31].

This work extensively considers scaled-orthographic (also known as weak-perspective) projection, in which a 3D object is scaled down and projected orthographically into the image. This projection model is appropriate when the camera is far from the objects being viewed with respect to their sizes. In this case, the image position of $(\overline{x}, \overline{y}, \overline{z})$ can be written in terms of the first two rows of a scaled, 3D rotation matrix, $\mathbf{S} = s\mathbf{R}$, and of a scaled, 3D translation vector, $\vec{b}$. Letting s_{ij} be the elements of $\mathbf{S}$, we have

$$x = s_{11}\overline{x} + s_{12}\overline{y} + s_{13}\overline{z} + b_x, \qquad y = s_{21}\overline{x} + s_{22}\overline{y} + s_{23}\overline{z} + b_y, \tag{4.2}$$

where $\| (s_{11}, s_{12}, s_{13}) \| = \| (s_{21}, s_{22}, s_{23}) \|$ and $(s_{11}, s_{12}, s_{13}) \cdot (s_{21}, s_{22}, s_{23}) = 0$. There are six degrees of freedom in the scaled-orthographic model-to-image transformation, and consequently three corresponding points are minimal to determine the transformation. Given three corresponding points, the transformation always exists if the model points are not collinear and it generally has two solutions [52, 2]; in particular, the scale factor and translation are always unique, and the rigid rotation matrix is unique up to a reflection of the rotated model about a plane parallel to the image.

For 3D linear projection, we remove the two non-linear constraints on the rotation parameters in the scaled-orthographic projection model. This transformation is equivalent to applying a scaled-orthographic transformation to the model, and then applying a scaled-orthographic transformation to the resulting image; in total, this is like taking a picture of a photograph [55]. There are eight degrees of freedom in linear projection, and four corresponding points are minimal to determine the transformation. Given a minimal set of matches, this is the only transformation of the three (perspective, scaled-orthographic, 3D linear) in which the unmatched model points can be written *linearly* in terms of the matched image points. In particular, let the four image and model points be (x_i, y_i) and $(\overline{x}_i, \overline{y}_i, \overline{z}_i)$, respectively, for $i = 1, 2, 3, 4$. Then we can obtain the first row of the transformation by solving

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \overline{x}_1 & \overline{y}_1 & \overline{z}_1 & 1 \\ \overline{x}_2 & \overline{y}_2 & \overline{z}_2 & 1 \\ \overline{x}_3 & \overline{y}_3 & \overline{z}_3 & 1 \\ \overline{x}_4 & \overline{y}_4 & \overline{z}_4 & 1 \end{bmatrix} \begin{bmatrix} s_{11} \\ s_{12} \\ s_{13} \\ b_x \end{bmatrix}. \qquad (4.3)$$

A similar equation holds for the second row of the transformation. These equations give linear expressions for the transformation parameters in terms of the image point coordinates. Since multiplying a matrix by a vector is a linear operation, applying the computed transformation to any unmatched model point gives a linear expression for the model point's image position in terms of the matched image points.

4.3 Fourth-Point Uncertainty Region

In this section, we assume we are given exactly three point correspondences, and that the correspondences are used to calculate a model pose and then project a fourth model point into the image, obtaining its predicted image position. We specifically address the following problem: Given exactly three matching point pairs, $(\vec{i}_0, \vec{m}_0)$, $(\vec{i}_1, \vec{m}_1)$, and $(\vec{i}_2, \vec{m}_2)$, where the locations of $\vec{i}_0$, $\vec{i}_1$, and $\vec{i}_2$ contain small amounts of error, what is the error in the predicted image position of a fourth model point, $\vec{m}_3$? This section presents an analytic solution to this problem, which is based on a first-order approximation, and results in a linear relationship between the errors in the matched (basis) image points and the error in an unmatched, projected model point. Here we consider weak-perspective projection, and in a later section we extend the results to perspective.

For weak-perspective projection, we show that the linear relationship takes a simple form that can be used to predict the uncertainty region for an unmatched model point when there is bounded error or Gaussian error in the image points. When we allow the three basis image points to be perturbed within bounded error regions, the resulting uncertainty region is also bounded. When we allow for Gaussian error in the image points, the uncertainty region is a probability distribution. Previously these uncertainty regions were known analytically only for planar objects [54, 80]. Our results have no such restriction, and they reduce to the known solutions when the model is planar. Furthermore, when the error in the image points is bounded by circles, the region takes the form of a circle centered at the "nominal point," which is the point that $\vec{m}_3$ projects onto when there is no error in the basis points. This result agrees with the experimental observations of Alter and Grimson [4].

4.3.1 The Basic Geometry

We begin by examining the propagated uncertainty when there is error in exactly one of three matched image points, $\vec{i}_2$. To do this, we introduce a particular representation for any third model point that allows us to see how a change in the location of the

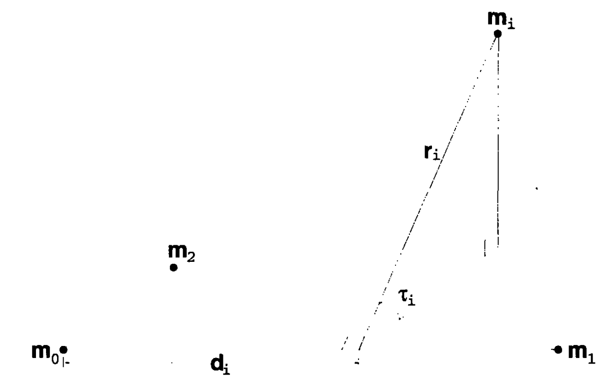


Figure 4-2: Every model point has unique (d_i, r_i, τ_i) coordinates, unless it is on the line through $\vec{m}_{01}$, where τ_i is unrestricted. Note that $\tau_i = 0$ for points in the basis plane.

third image point affects the projected location of any unmatched model point. In this representation, we let the origin of the image coordinate system be at $\vec{\imath}_0$, the z direction be orthogonal to the image plane, and the x axis point in the same direction as $\vec{\imath}_{01}$, where $\vec{\imath}_{01} = \vec{\imath}_1 - \vec{\imath}_0$.

Furthermore, we use the following representation of 3D model points in terms of the three basis model points (see Fig. 4-2): Originally, the model points lie in some model coordinate system. For any model point $\vec{m}_i$, $i \geq 0$, let r_i be the length of the perpendicular from $\vec{m}_i$ to the infinite line containing $\vec{m}_0$ and $\vec{m}_1$, let d_i be the distance from $\vec{m}_0$ to the intersection of the perpendicular and the line, and let τ_i be the rotation of the perpendicular off the basis plane (the plane containing the three basis points).

A view of a 3D model is determined by choosing the six parameters of a weak-perspective transformation that will be applied to the model. (There are two parameters for in-plane translation, three for rotation, and one for scale.) In this section we have fixed $\vec{\imath}_0$ and $\vec{\imath}_1$. By fixing the locations in the image where two of the model points project, we have determined four of the transformation's parameters. In particular, we initially can rigidly transform and scale the model so that $\vec{m}_0 = \vec{\imath}_0$, $\vec{m}_1 = \vec{\imath}_1$, and

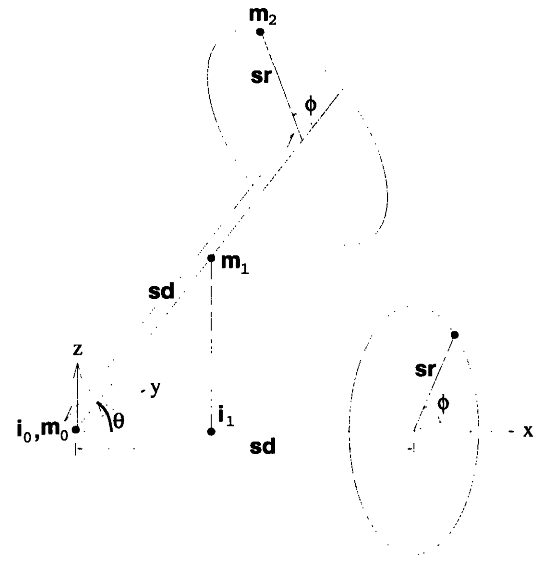


Figure 4-3: The out-of-plane rotations: a rotation about the y axis and a rotation about the vector $\vec{m}_{01}$.

$\vec{m}_2$ is in the $z = 0$ plane—in so doing, the model is scaled by

$$s_0 = \frac{\| \vec{i}_{01} \|}{\| \vec{m}_{01} \|} . \tag{4.4}$$

In order to keep $\vec{m}_0$ and $\vec{m}_1$ projecting onto $\vec{i}_0$ and $\vec{i}_1$, respectively, there can be no further in-plane translation nor in-plane rotation. As shown in Fig. 4-3, we still are free to rotate about the y axis as long as $\vec{m}_1$ continues to project onto $\vec{i}_1$, which means that any such rotation about the y axis determines the scale factor. After rotating about the y axis and rescaling, the only remaining degree of freedom is a rotation about the vector $\vec{m}_{01}$.

Next we derive an expression for the image position of $\vec{m}_2$ as a function of the two free parameters. As illustrated in Fig. 4-3, the model is scaled by s, then rotated by ϕ about the x axis, and then rotated by θ about the y axis (denoted by $\mathbf{R}_{\{\theta,y\}}$). This aligns the projections of the three model points with their corresponding image points. As in Fig. 4-2, we let $\vec{m}_2$'s coordinates relative to the basis model points be $(d_2, r_2, \tau_2) = (d, r, 0)$, where we have dropped the subscripts. Notice that the last element is 0 since $\vec{m}_2$ is in the basis plane. So $\vec{m}_2 = \mathbf{R}_{\{\theta,y\}}(sd, sr\cos\phi, sr\sin\phi)$, where

$\phi \in [0, 2\pi)$, which gives

$$\vec{m}_2 = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} sd \\ sr\cos\phi \\ sr\sin\phi \end{bmatrix} = s \begin{bmatrix} d\cos\theta - r\sin\theta\sin\phi \\ r\cos\phi \\ d\sin\theta + r\cos\theta\sin\phi \end{bmatrix} \tag{4.5}$$

By our choice of image coordinate system, $\theta \in [0, \pi/2)$. Project orthographically:

$$\vec{m}_2^{\pi} = (x, y) = (sd\cos\theta, 0) + sr(-\sin\theta\sin\phi, \cos\phi).$$

As $\vec{m}_1$ rotates around the y axis, the scale factor is constrained to keep $\vec{m}_1$ projecting onto $\vec{\imath}_1$. From Fig. 4-3, this constraint is

$$s \parallel \vec{m}_{01} \parallel \cos\theta = \parallel \vec{\imath}_{01} \parallel \quad \Longrightarrow \quad s = \frac{\parallel \vec{\imath}_{01} \parallel}{\parallel \vec{m}_{01} \parallel \cos\theta} = \frac{s_0}{\cos\theta} \tag{4.6}$$

Consequently,

$$(x, y) = (s_0 d, 0) + s_0 r \left(-\tan\theta\sin\phi, \frac{\cos\phi}{\cos\theta} \right), \tag{4.7}$$

which gives the image position of $\vec{m}_2$ as a function of the "out-of-plane" rotation angles θ and ϕ. Fig. 4-4 shows graphically the successive rotations of $\vec{m}_2$ by θ and ϕ, followed by the orthographic projection.

By setting (x, y) in Equation 4.7 to the nominal location of $\vec{\imath}_2$, we could solve for the nominal values of θ and ϕ. If done, this would provide a solution to the problem of recovering 3D pose from three corresponding points. There are several solutions to this problem already, however, and instead we could apply one of them and then use the solution to compute θ and ϕ (see [3] for a review of the solutions).

4.3.2 First-Order Approximation

Next we allow for error in one of the basis points, $\vec{\imath}_2$. The problem is to determine how $\vec{m}_3^{\pi}$ changes as a function of $\vec{\imath}_2$, with $\vec{\imath}_0$ and $\vec{\imath}_1$ remaining fixed. In Section 4.3.4, we explain how to extend the result to the case where all three points can move. The out-of-plane rotations, θ and ϕ, and the scale will change as $\vec{\imath}_2$ moves in the plane.

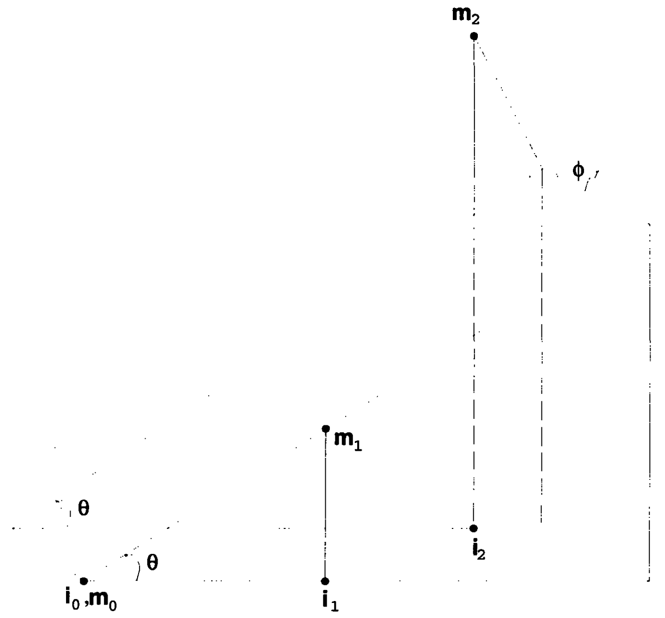


Figure 4-4: The image position of $\vec{m}_2$ is a function of the out-of-plane rotations θ and ϕ.

If the changes in θ and ϕ are sufficiently small, then the changes in x and y as a function of θ and ϕ will be given by their first derivatives. From Equation 4.7,

$$\frac{\partial x}{\partial \theta} = -s_0 r \frac{1}{\cos^2\theta}\sin\phi, \qquad \frac{\partial y}{\partial \theta} = s_0 r \frac{\sin\theta}{\cos^2\theta}\cos\phi \tag{4.8}$$

$$\frac{\partial x}{\partial \phi} = -s_0 r \frac{\sin\theta}{\cos\theta}\cos\phi, \qquad \frac{\partial y}{\partial \phi} = -s_0 r \frac{1}{\cos\theta}\sin\phi \tag{4.9}$$

$\overrightarrow{t_\theta} = \left(\frac{\partial x}{\partial \theta}, \frac{\partial y}{\partial \theta}\right)$ and $\overrightarrow{t_\phi} = \left(\frac{\partial x}{\partial \phi}, \frac{\partial y}{\partial \phi}\right)$ are the tangent vectors at the point (x, y) to the image curves that are traced out by changing θ and ϕ.

$$\| \overrightarrow{t_\theta} \| = \frac{s_0 r}{\cos^2\theta}\sqrt{\sin^2\phi + \sin^2\theta\cos^2\phi} \tag{4.10}$$

$$\| \overrightarrow{t_\phi} \| = \frac{s_0 r}{\cos\theta}\sqrt{\sin^2\phi + \sin^2\theta\cos^2\phi} \tag{4.11}$$

For a small change in θ, let α_θ represent the direction in which the image point $\vec{i}_2$ moves, measured counter-clockwise from the x axis, and similarly for ϕ and α_ϕ. Then since $\theta \in [0, \pi/2)$

$$\alpha_\theta = \tan^{-1}\left(\frac{\partial y}{\partial \theta}, \frac{\partial x}{\partial \theta}\right) = \tan^{-1}(\sin\theta\cos\phi, -\sin\phi) \tag{4.12}$$

$$\alpha_\phi = \tan^{-1}\left(\frac{\partial y}{\partial \phi}, \frac{\partial x}{\partial \phi}\right) = \tan^{-1}(-\sin\phi, -\sin\theta\cos\phi) \tag{4.13}$$

The dot product of the arguments to the inverse tangent is 0, and so the tangent vectors $\overrightarrow{t_\theta}$ and $\overrightarrow{t_\phi}$ are perpendicular. We can use the normalized cross product of the arguments to see whether the angle between $\overrightarrow{t_\theta}$ and $\overrightarrow{t_\phi}$ is $\pm 90°$:

$$\sin(\alpha_\phi - \alpha_\theta) = \frac{(-\sin\phi)(-\sin\phi) - (\sin\theta\cos\phi)(-\sin\theta\cos\phi)}{\|\,(-\sin\phi, \sin\theta\cos\phi)\,\|\|\,(-\sin\theta\cos\phi, -\sin\phi)\,\|} = 1$$

Thus $\alpha_\phi = \alpha_\theta + 90°$.

4.3.3 Relative Error between the Third and Fourth Points

We still are considering the case where only $\vec{\imath}_2$ moves. This section shows that if the projected third model point moves by a small amount, then any projected fourth model point moves by a constant factor times that small amount, and in an analogous direction. We also show that the two weak-perspective solutions lead to different constant factors.

Equations 4.10 and 4.11 give the magnitude of the changes in $\vec{m}_2^\pi$ for small changes in θ and ϕ. The only differences for $\vec{m}_3^\pi$ are in the relative coordinates of the model points (Fig. 4-2). For $\vec{m}_2$, the coordinates are $(r_2, d_2, \tau_2) = (r, d, 0)$. For $\vec{m}_3$, let the coordinates be $(r_3, d_3, \tau_3) = (r', d', \tau')$. Consequently, in the expressions, we change $r \to r'$ and $d \to d'$. Further, θ and s_0 do not change since they are measured with respect to $\vec{m}_0$ and $\vec{m}_1$, and do not depend on whether we are considering $\vec{m}_2$ or $\vec{m}_3$. However, ϕ is the amount of rotation of $\vec{m}_2$ away from the plane of the wedge in Fig. 4-4. For any other model point, $\vec{m}_3$, the amount of rotation away from the wedge is given by $\phi + \tau'$, since, according to Fig. 4-2, τ' is the amount of rotation of

$\vec{m}_3$ away from $\vec{m}_2$. Hence for $\vec{m}_3$, we change $\phi \to \phi + \tau'$. All together, we get

$$\| \overrightarrow{t'_\theta} \| = \frac{s_0 r'}{\cos^2 \theta} \sqrt{\sin^2(\phi + \tau') + \sin^2 \theta \cos^2(\phi + \tau')} \tag{4.14}$$

$$\| \overrightarrow{t'_\phi} \| = \frac{s_0 r'}{\cos \theta} \sqrt{\sin^2(\phi + \tau') + \sin^2 \theta \cos^2(\phi + \tau')} \tag{4.15}$$

Note that $\| \overrightarrow{t'_\theta} \| / \| \overrightarrow{t_\theta} \| = \| \overrightarrow{t'_\phi} \| / \| \overrightarrow{t_\phi} \|$. Therefore, when either ϕ or θ changes, the ratio of the size of the change in $\vec{m}_3^\pi$ to the size of change in $\vec{m}_2^\pi$ equals

$$S = \frac{r'}{r} \sqrt{\frac{\sin^2(\phi + \tau') + \sin^2 \theta \cos^2(\phi + \tau')}{\sin^2 \phi + \sin^2 \theta \cos^2 \phi}} \tag{4.16}$$

Note that due to θ and ϕ, this expression depends on the viewpoint from which the model is observed, which is not true in the planar case [54]. In the planar case $\tau' = 0$, because all model points are in the basis plane (Fig. 4-2), and Equation 4.16 reduces to $\frac{r'}{r}$.

Recall that there are two reflective solutions to the weak-perspective geometry [2]. The two solutions correspond to a reflection of the basis model points about the image plane. From Fig. 4-4, this reflection corresponds to negating θ and ϕ. Plugging into Equation 4.16, the constant scaling factors for the two solutions are

$$\begin{aligned} & \frac{r'}{r} \sqrt{\frac{\sin^2(\sigma\phi + \tau') + \sin^2(\sigma\theta) \cos^2(\sigma\phi + \tau')}{\sin^2(\sigma\phi) + \sin^2(\sigma\theta) \cos^2(\sigma\phi)}}, \quad \text{where } \sigma = \pm 1 \\ = & \frac{r'}{r} \sqrt{\frac{\sin^2(\sigma\phi + \tau') + \sin^2 \theta \cos^2(\sigma\phi + \tau')}{\sin^2 \phi + \sin^2 \theta \cos^2 \phi}} \end{aligned} \tag{4.17}$$

Thus the two weak-perspective solutions give different scaling constants. Again, this differs from the planar case, in which the scaling constant in both cases is $\frac{r'}{r}$. More generally, in the planar case the two solutions collapse to one when projected onto the image, and so the existence of two solutions makes no difference.

From Equations 4.12 and 4.13,

$$\alpha'_\theta = \tan^{-1}(-\sin(\phi+\tau'), \sin\theta\cos(\phi+\tau')) \tag{4.18}$$

$$\alpha'_\phi = \tan^{-1}(-\sin\theta\cos(\phi+\tau'), -\sin(\phi+\tau')) \tag{4.19}$$

Through the same calculation that showed $\sin(\alpha_\phi - \alpha_\theta) = 1$, we can calculate that $\sin(\alpha'_\phi - \alpha'_\theta) = 1$, so that

$$\alpha'_\phi - \alpha'_\theta = \alpha_\phi - \alpha_\theta = 90^\circ. \tag{4.20}$$

Thus the angles between the tangent vectors and their relative sizes are the same for $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$. As a note, this implies that the mapping between curves traced out by changing θ and ϕ for $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$ is *conformal* [1].

Since we are making a first-order approximation, any movement of $\vec{m}_2^\pi$ in the image plane can be viewed as the sum of the effects of changes in θ and ϕ. From $\| \overrightarrow{t'_\theta} \| / \| \overrightarrow{t_\theta} \| = \| \overrightarrow{t'_\phi} \| / \| \overrightarrow{t_\phi} \|$, we see that for any change in $\vec{m}_2^\pi$ by some small amount, there is a change in $\vec{m}_3^\pi$ by that amount times a constant, given by Equation 4.16. Furthermore, as θ changes, $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$ each moves in some direction, by some amount. Then as ϕ changes, Equation 4.20 implies that the two points move at right angles to their previous directions. Hence any change in $\vec{m}_2^\pi$ produces a change in $\vec{m}_3^\pi$ that is scaled and rotated by fixed amounts. Consequently, any error region about the nominal position of $\vec{m}_2^\pi$ results in a mathematically similar error region about the nominal position of $\vec{m}_3^\pi$, which means they are related by an image plane translation, rotation, and scaling.

We can explicitly write the relationship between the errors in $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$ using a 2×2 scaled rotation matrix $\mathbf{A}$. ($\mathbf{A}$ is a similarity transform with zero translation. In the sequel we will refer to $\mathbf{A}$ interchangeably as a scaled rotation matrix or a similarity transform.) $\mathbf{A}$ must satisfy

$$\overrightarrow{t'_\theta} = \mathbf{A}\overrightarrow{t_\theta} \quad \text{and} \quad \overrightarrow{t'_\phi} = \mathbf{A}\overrightarrow{t_\phi}, \tag{4.21}$$

which gives four equations in four unknowns. Actually only two of the equations are needed: In general, let a_{11}, a_{12}, a_{21}, and a_{22} be the elements of a 2×2 matrix $\mathbf{A}$. Then for a similarity transform, $a_{21} = -a_{12}$ and $a_{22} = a_{11}$. Solving the equations leads to (Appendix B)

$$\begin{aligned} a_{11} &= k(\cos\tau' - \cos^2\theta\cos\phi\cos(\phi+\tau')) && (4.22) \\ a_{12} &= -k\sin\tau'\sin\theta, && (4.23) \\ \text{where} \quad k &= \left(\frac{r'}{r}\right)\frac{1}{1-\cos^2\theta\cos^2\phi}\,. && (4.24) \end{aligned}$$

Note that the constant scale S from Equation 4.16 must equal $\sqrt{a_{11}^2 + a_{12}^2}$.

4.3.4 General Formula for the Fourth Point Error

We can use Equations 4.22-4.24 to obtain a formula for the error in $\vec{m}_3^\pi$ as a function of the error in $\vec{i}_0$ or $\vec{i}_1$ in the same way. This gives three scaled rotation matrices relating the individual errors in the basis points to the error in $\vec{m}_3^\pi$. Under a first-order approximation, these errors affect the error in $\vec{m}_3^\pi$ independently and the total error in $\vec{m}_3^\pi$ is the sum of the individual errors. Let $\mathbf{A}$ be the scaled rotation matrix between $\vec{i}_0$ and $\vec{m}_3^\pi$, $\mathbf{B}$ be the scaled rotation matrix between $\vec{i}_1$ and $\vec{m}_3^\pi$, and $\mathbf{C}$ be the scaled rotation matrix between $\vec{i}_2$ and $\vec{m}_3^\pi$. Also let $\vec{e}_0$, $\vec{e}_1$, and $\vec{e}_2$ be the errors in the basis points, and, for $i \geq 3$, let $\vec{e}_i$ be the error vector from the nominal position of $\vec{m}_i^\pi$ to its true position (see Fig. 4-5). Then the error in $\vec{m}_3^\pi$ is given by the following linear relationship:

$$\vec{e}_3 = \mathbf{A}\vec{e}_0 + \mathbf{B}\vec{e}_1 + \mathbf{C}\vec{e}_2 \qquad (4.25)$$

The two weak-perspective pose solutions, $\pm(\theta, \phi)$, lead to two possibilities for each of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, and for the error in the fourth point $\vec{e}_3$. When we combine the errors in Equation 4.25, we must be sure to use the same weak-perspective solution for the three matrices.

Suppose now that the error in each image point is bounded by some amount ϵ. This results in some bounded uncertainty region about $\vec{m}_3^\pi$. From Section 4.3.3, for

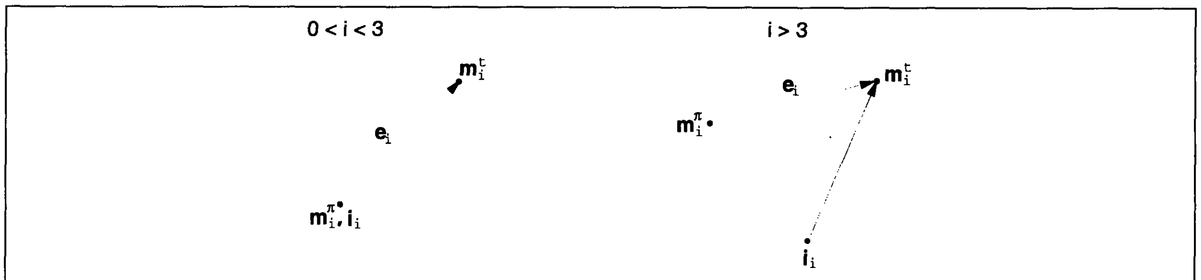


Figure 4-5: For each model point $\vec{m}_i$, there are three points of interest in the image: (1) its nominal (detected) position $\vec{i}_i$, which is determined by the feature detector, (2) its no-error (true) position $\vec{m}_i^t$, which would equal $\vec{i}_i$ if there were no error, and (3) its predicted position $\vec{m}_i^\pi$, which is computed using the first three point pairs to compute the pose and project $\vec{m}_i$ into the image. For $i \leq 3$, $\vec{m}_i^\pi = \vec{i}_i$. For any i, we define $\vec{e}_i$ to be the correction vector from $\vec{m}_i^\pi$ to $\vec{m}_i^t$.

each image point separately its ϵ-circle propagates to a circle around $\vec{m}_3^\pi$ that is scaled by S (Equation 4.16), which gives a radius of $S\epsilon$. The error in each image point affects the fourth point independently, and so the uncertainty region around the fourth point is a circle centered at $\vec{m}_3^\pi$, and the three radii simply sum together (where we must be careful to use the radii from the same weak-perspective solution). In Equation 4.25, let $S_0 = \sqrt{a_{11}^2 + a_{12}^2}$, $S_1 = \sqrt{b_{11}^2 + b_{12}^2}$, and $S_2 = \sqrt{c_{11}^2 + c_{12}^2}$. Then the radius of the uncertainty circle for $\vec{m}_3^\pi$ is

$$R = S_0\epsilon_0 + S_1\epsilon_1 + S_2\epsilon_2 + \epsilon_3, \tag{4.26}$$

where ϵ_3 is added to account for error in sensing $\vec{i}_3$ (an image point that corresponds to $\vec{m}_3^\pi$).

When the error in the image points is normally distributed, the linear relationship allows us to determine the propagated uncertainty distribution about any fourth point. If the Gaussian error in the ith image point has standard deviation σ_i, then the uncertainty distribution about $\vec{m_3}^\pi$ is normally distributed with standard deviation (Appendix D)

$$\sigma = \sqrt{S_0^2\sigma_0^2 + S_1^2\sigma_1^2 + S_2^2\sigma_2^2 + \sigma_3^2}. \tag{4.27}$$

In summary, we have explained why Alter and Grimson found that the uncertainty regions for three matched points are circular. Moreover, we have provided a simple, analytic expression for those uncertainty regions.

4.4 A Study of Uncertainty from One Basis Point

We showed in Section 4.3 that for any shape traced out by the third model point, the fourth model point traces out a mathematically similar shape, up to an approximation. This section provides a study of the true shape of the uncertainty region. We begin by considering exactly how larger changes in θ and ϕ effect the appearance of each additional model point. Using Equation 4.7, it is straightforward to see that, as ϕ changes with θ held constant, (x, y) traces out an ellipse, with center at $(s_0 d, 0)$ and with the major axis parallel to the y axis. We rewrite this equation as

$$(x, y) = (s_0 d, 0) + (-a \sin\phi, b \cos\phi), \tag{4.28}$$

with minor axis $a = s_0 r \tan\theta$, major axis $b = s_0 r \sec\theta$, and eccentricity $e = \sqrt{b^2 - a^2} = s_0 r$. For $\theta = 0$, the ellipse equation becomes $(x, y) = (s_0 d, 0) + (0, s_0 r \cos\phi)$, which forms a line segment between the points $(s_0 d, s_0 r)$ and $(s_0 d, -s_0 r)$. As θ increases from 0, the center of the ellipse is unchanged; Fig. 4-6 shows the 3D interpretation. In addition, as θ increases both axes a and b grow, and the ratio $a/b = \sin\theta$ approaches 1. Consequently increasing θ sweeps out a growing family of concentric ellipses that become increasingly circular.

As θ changes, (x, y) traces out a hyperbola, which we can see by eliminating θ from Equation 4.7. From the x and y coordinates, we have respectively:

$$\left(\frac{x - s_0 d}{s_0 r \sin\phi}\right)^2 = \tan^2\theta, \qquad \text{and} \qquad \left(\frac{y}{s_0 r \cos\phi}\right)^2 = \sec^2\theta.$$

Using $1 + \tan^2\theta = \sec^2\theta$,

$$-\left(\frac{x - s_0 d}{s_0 r \sin\phi}\right)^2 + \left(\frac{y}{s_0 r \cos\phi}\right)^2 = 1. \tag{4.29}$$

This equation is a hyperbola centered at $(s_0 d, 0)$, with foci $(s_0 d, \pm s_0 r)$, vertices

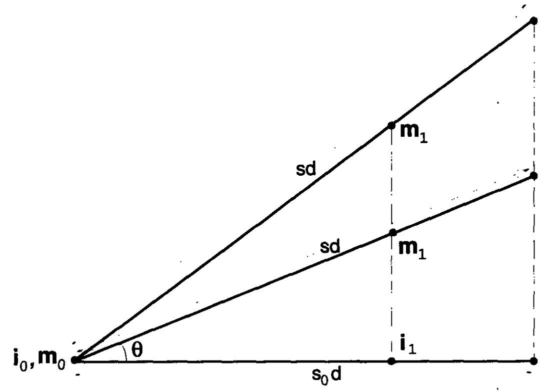


Figure 4-6: As θ changes, $\vec{m}_{01}$ is scaled so that $\vec{m}_1$ always projects onto $\vec{i}_1$. The figure shows that any point on the line through $\vec{m}_{01}$ always projects onto the same location in the image. When ϕ changes, $\vec{m}_2$ rotates about a point on this line, and that point projects to the ellipse center; therefore the ellipse center does not change.

$(s_0 d, \pm s_0 r \cos\phi)$, and asymptotes $x = \pm y \tan\phi + s_0 d$. For $\phi = 0, \pi$, the hyperbola becomes $(x, y) = (s_0 d, 0) \pm (0, \frac{s_0 r}{\cos\theta})$, which gives two vertically infinite half-lines starting at the foci. As ϕ increases, the center and foci are unchanged, and the asymptotes rotate about $(s_0 d, 0)$, becoming increasingly parallel to the x axis. Consequently, changing ϕ from 0 or π sweeps out a concentric family of hyperbolas that reach the x axis at $\phi = \frac{\pi}{2}, \frac{3\pi}{2}$.

To capture the dependence of (x, y) on the out-of-plane rotations θ and ϕ, we will switch to a set of normalized coordinates (X, Y) that depend only on θ and ϕ. These coordinates shall allow us to plot graphs which make it easier to view the dependence of the error in the predicted fourth model point on the error in the third image point. By a simple translation and scale of (x, y), we get the equation

$$(X, Y) = \frac{1}{s_0 r}\left((x, y) - (s_0, d)\right) = \left(-\tan\theta\sin\phi, \frac{\cos\phi}{\cos\theta}\right). \qquad (4.30)$$

Fig. 4-7 plots families of curves for changing ϕ and θ for this equation. The elliptical curves are functions of ϕ, and the hyperbolic curves are functions of θ. Observe that, although the plots are for (X, Y), the corresponding plots for (x, y) would look the same, differing only in the labels on the axes. This is because the effect of the normalizing translation is to center the plots at $(0, 0)$ instead of (s_0, d_0), and the effect of the normalizing scale cannot be seen since the plots are rescaled to fit the page.

The plots in Fig. 4-7 are for $\vec{m}_2^\pi$ translated and scaled. For $\vec{m}_3^\pi$, we again shall switch to normalized coordinates (using a different normalizing translation and scale):

$$(X',Y') = \frac{1}{s_0 r'}\left((x',y') - (s_0, d')\right) = \left(-\tan\theta \sin(\phi+\tau'),\ \frac{\cos(\phi+\tau')}{\cos\theta}\right). \qquad (4.31)$$

In the resulting equation for (X',Y'), the only difference from (X,Y) is in τ'. That is, varying ϕ causes (x,y) and (x',y') to traverse ellipses that may differ in translation and scale, but that do not differ in shape. In Fig. 4-7, we show the image regions that the projected points $\vec{m}_2^\pi$ and $\vec{m}_3^\pi$ occupy as a result of varying θ and ϕ. Since we normalized the plots, as the third model varies along an ellipse, the fourth model point projects onto the same ellipse. If (X,Y) moves along the curves in Fig. 4-7 and traces out a closed region, then so will (X',Y'). The two filled-in areas in each plot illustrate a different pair of bounded error regions that could be traced out simultaneously by some (X,Y) and (X',Y'); (X,Y) or (X',Y') could be either error region. In this case, as one point varies within one of the bounded error regions, the other point will vary within the other error region, illustrating how uncertainty propagates.

In the figure, there are two distinguished points where the grid collapses. These points are the hyperbola foci $(0,\pm 1)$, which were $(s_0 d, \pm s_0 r)$ in the original (x,y) curve. These foci correspond to $(\theta,\phi) = (0,0)$ and $(0,\pi)$, which occur when the plane containing the basis model points is parallel to the image. From the areas around $(0,1)$ in Figs. 4-7 and 4-8, it is clear that an (X',Y') uncertainty region may not be simply a scaled and rotated version of the (X,Y) region. Therefore, circular error in the third point in general will not produce circular uncertainty, exactly, in the fourth point. In fact, we can see from Fig. 4-8-left that when θ and ϕ are small, uncertainty in the third image point can lead to strange shapes for the propagated uncertainty region: Suppose that (X,Y) traces out the large region on the right (region A). If $\tau' = 128\,\text{deg}$, then (X',Y') traces out the similar-looking, large region on the left (region B). But if $\tau' = 72\,\text{deg}$, then (X',Y') traces out the non-convex, curved region on top (region C). It should be kept in mind that the shapes not the sizes of these

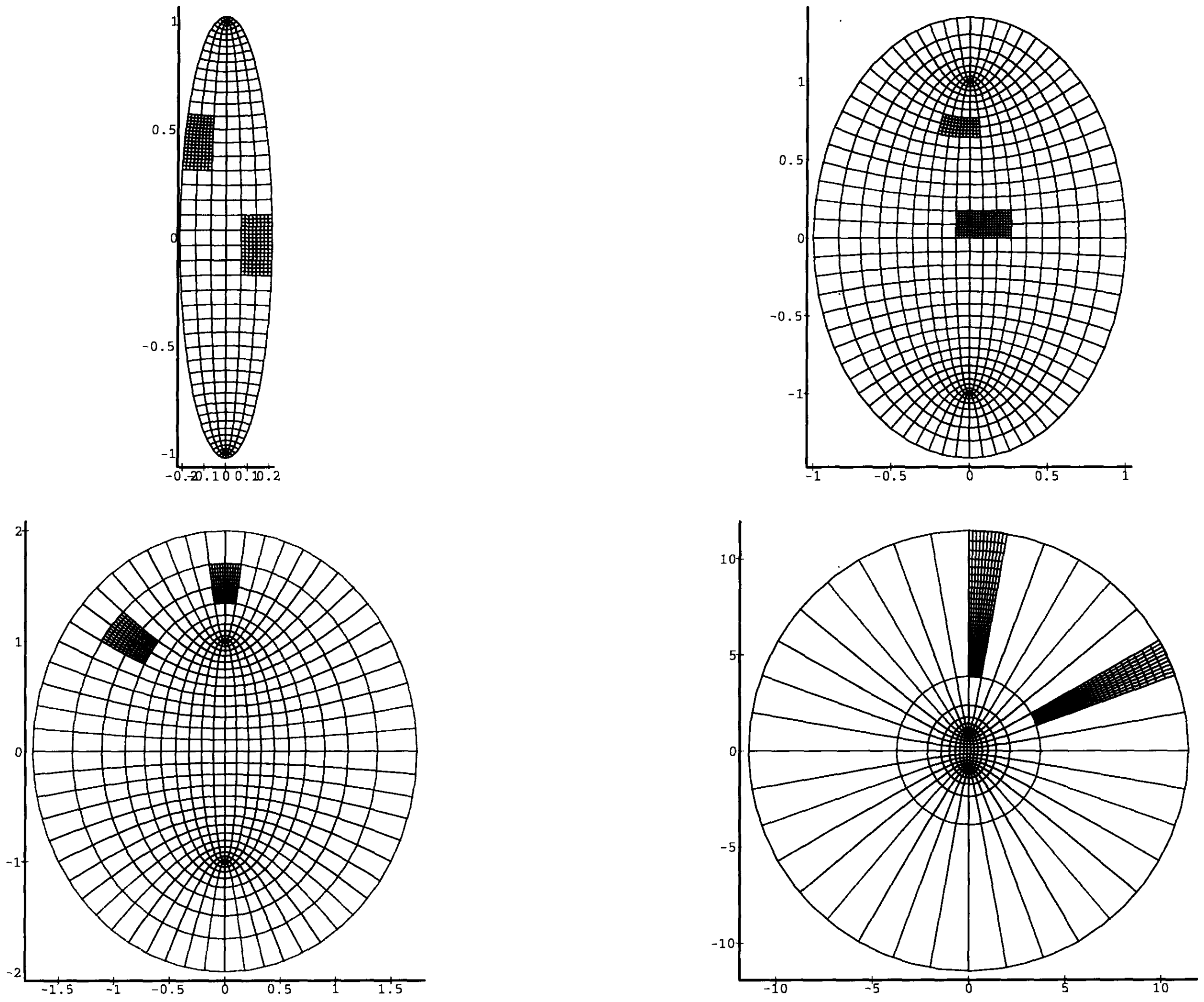
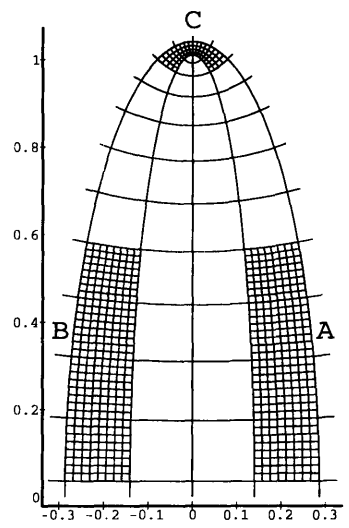


Figure 4-7: Families of curves traced out by (X, Y) as ϕ and θ change. The elliptical curves are functions of ϕ, and the hyperbolic curves are functions of θ. The four plots show the same figure for different ranges of θ, plotted at different scales. In left-to-right, top-to-bottom order, each plot is a close-up of the center area in the next plot. Upper left: $\theta \in [0, 12]$ deg. Upper right: $\theta \in [0, 45]$ deg. Lower left: $\theta \in [0, 60]$ deg. Lower right: $\theta \in [5, 85]$ deg. The curves are separated by changes in angle of 4 deg, 5 deg, 6 deg, and 10 deg (in left-to-right, top-to-bottom order).



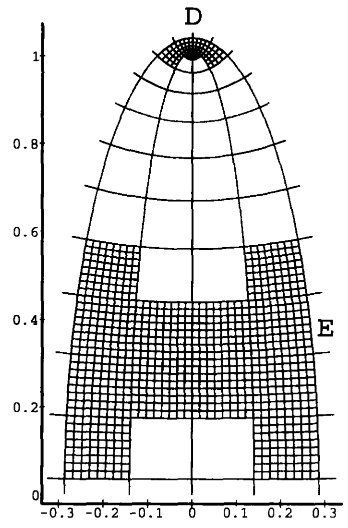


Figure 4-8: Close-up where $\phi \in [0, 90]$ deg. In both plots, $\theta \in [0, 16]$ deg and the curves are separated by changes in θ and ϕ of 8 deg.

regions are what matters here, since the plot is normalized: After being scaled by $\frac{r'}{r}$, the region on top may be significantly larger than then the region on the right, but its unusual shape would be unchanged. As a result, odd-shaped uncertainty regions can occur even when there is little error in the basis points.

When the third point's error region contains one of the foci, then the two uncertainty regions for the fourth point's two weak-perspective pose solutions are one and the same; otherwise the regions will be distinct, although they may overlap. As an example, in Fig. 4-8-left suppose now that region C is traced out by (X, Y). If $\tau' = 72$ deg, then the two large regions correspond to the two weak-perspective solutions for the (X', Y') region, obtained by alternating the sign of (θ, ϕ). If $\tau' = 8$ deg, then the two uncertainty regions for the two solutions would overlap, but they still would be distinct. On the other hand, suppose that the region traced by (X, Y) additionally includes the point $(0, 1)$, as in Fig. 4-8-right (region D). Then when $\tau' = 72$ deg the two large regions merge into the single, "H"-shaped uncertainty region shown in the figure (region E). So we see that the similarity transform can be a poor approximation for poses with small values of θ and ϕ even with small amounts of error.

For large θ (bottom, right picture in Fig. 4-7), the ellipses become concentric circles, and the hyperbolas become straight lines. In this circumstance, any (X', Y') region is the same as the corresponding (X, Y) region, except for a rotation about the origin. In this case, a similarity transform will exactly relate the errors, and will be independent of the model pose (as it was for planar objects).

Another conclusion we can draw applies when the third point has a bounded error region that is very large. Suppose there is circular error of radius ϵ in the third point. As ϵ grows, the error circle will include the point where the two pose solutions merge, and the boundaries of the error circle and the fourth point's propagated uncertainty region will reach the range of θ where they are related by a similarity transformation. For large enough ϵ, then, the error in the third point will result in a single, circular uncertainty region for the fourth point, regardless of where the image and model points are nominally located. This is surprising because one might expect that the error incurred by using the similarity-transform approximation grows as ϵ grows. Even though this may be true when ϵ is small, for large enough ϵ the error will decrease.

Finally, the above discussion shows that the similarity-transform approximation may hold further than the two first-order approximations that we used to compute it. To obtain an analytic expression for the propagated uncertainty, Section 4.3 took first-order approximations to the errors in the third and fourth points in terms of the 3D pose parameters, θ and ϕ. Then these approximations were combined to get a similarity transform that directly relates the errors in the points. This similarity transform may hold further than the two first-order approximations. For instance, for high values of θ, where the similarity transform holds exactly, the first-order approximations do not (see Fig. 4-9). This suggests that higher-order error terms may be cancelling when we combine the two first-order approximations to get a similarity transform. Section 4.5.3 empirically shows that the similarity transform can hold further than the first-order approximation. This indicates that it may be wise to propagate errors directly in image space rather than from transformation space, as was done for example in [37].

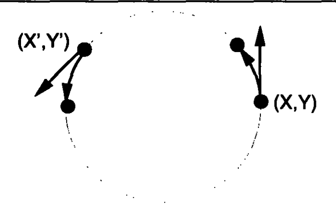


Figure 4-9: For large θ, the ellipses traced out by changing ϕ are circles. In this case, changing ϕ causes movements in (X, Y) and (X', Y') that are exactly related by a similarity transform. First-order approximations to the movements, however, assume that the two points move along the tangent lines at (X, Y) and (X', Y'), which is correct only if the movements are small.

4.5 Experiments

We have run three experiments to test the results in Sections 4.3 and 4.4. The first experiment compares our results to those of Alter and Grimson [4], who studied the case of bounded, ϵ error. In particular, we test our formula for the radius of the fourth point error circle. The second experiment looks more generally at the accuracy of the similarity transforms in Equation 4.25 for predicting where the fourth point moves when there is error in the basis points. This experiment is repeated for uniform and Gaussian error. The third experiment compares the accuracy of the similarity transform with the first order approximation used to derive it.

4.5.1 Comparison to past results

Alter and Grimson assumed bounded error in the image points using circles of radius ϵ. They showed experimentally that the uncertainty region for a fourth model point is closely approximated by a circle centered at the nominal point, which is the uncertainty region we derived analytically in Section 4.3.4. Following Alter and Grimson, we refer to these fourth-point error circles as *uncertainty circles.* Alter and Grimson computed the radius of an uncertainty circle by densely sampling the fourth-point uncertainty region, and then took the radius to be the maximum distance from the nominal point to a sampled point. As a result, the computed uncertainty circle was an upper bound on the uncertainty region, up to the fineness of the sampling.

To see how our uncertainty circles compare to Alter and Grimson's, we run a series of trials like those in [4]: At each trial, we randomly generate three pairs of model and image points and a set of seven unmatched model points. We let $\epsilon = 5$ pixels. For each unmatched model point, we use Equation 4.26 to compute our predicted radius (R_f). To compute the "maximum radius" (R_M) as in [4], we take 25 samples along the boundary of each ϵ-circle, and then take all triples between the samples to get 15,625 triples. For each triple, we solve for the model pose and use the pose to project each unmatched model point into the image. This gives 15,625 projected model points per uncertainty region. For each uncertainty region, we compute the maximum distance from the nominal point to any of its projected model points. For an error measure, we use the relative error from our radius to the maximum radius, that is, $\frac{R_M - R_f}{R_f}$.

We tested 1,163 uncertainty circles, over which the average relative error was 1.45%. Table 4.1 shows the percent of uncertainty circles for which relative error was less than some threshold. The results show that our circles reasonably approximate the maximum-radius circles from Alter and Grimson, who showed that the maximum-radius circles reasonably approximate the true regions. The results also show that the maximum-radius circles can at times overestimate our circles by a significant amount. For instance, 1.2% of the time our circles will be 10% smaller than the maximum-radius circles.

Table 4.1 could be used to determine a correction factor for the circle radius: Suppose that a recognition system has matched three model and image points and is using the analytic solution for the uncertainty circles to decide what region in the image to search for additional matches. Suppose further that we want the system to conservatively estimate the effects of error, so that it can give some guarantee of no false negatives (i.e., that no valid image points will be missed) some high percentage of the time, while at the same time increasing the chance of false positives as little as possible. Then, for a chosen percent of the uncertainty circles, the percent relative error in the table tells us by how much to increase the radii of the circles so that all points in the true uncertainty region are included.

Percent relative error	2	4	6	8	10	12
Percent of uncertainty circles	86.7	94.2	97.1	98.2	98.8	99.0

Table 4.1: The percent of uncertainty circles for which the relative error between the circles predicted by our formula and by Alter and Grimson's method is less than a given percent.

4.5.2 Accuracy of the similarity transform

Section 4.3.4 showed how we can approximate the effects of error in the basis points using three similarity transforms. Equation 4.25 gives the error in the fourth point as a function of the errors in the basis points. The next experiment estimates how well this approximation works when the error in the image points is distributed uniformly, or according to a Gaussian. We run a series of trials like those in the previous experiment, where at each trial we generate a random model and a random image triple. Assuming uniform error for now, we then uniformly perturb each image point within a circle of radius ϵ. Using the perturbed image points, we compute the distances between where the similarity transforms predict the fourth model point will appear and its actual projected location. For each unmatched model point, there are two such distances corresponding to the two weak-perspective solutions. For each trial, there are seven unmatched model points, giving fourteen distances per trial. Over 10,000 trials, 140,000 unmatched model points were tested.

For uniform error, the average distance between the similarity point (the point predicted by the similarity transforms) and the true point was .06 pixels. Table 4.2 gives the percent of unmatched points for which the distance is less than various amounts of tolerated error. The results show that the similarity transforms very accurately predict the movement of the fourth point, with 98% probability of being in error by less than one pixel.

For Gaussian error, we run the same experiment but instead sample a 2D Gaussian distribution, using a standard deviation of 2.5 pixels as was done by Sarachik and Grimson [80]. We still use a bound of $\epsilon = 5$ pixels on the allowed error in the image points to guarantee that Gaussian error will lead to better results than uniform error. In fact, we might expect the results to be significantly better, since the sampled

Allowed distance error (px)	1	2	3	4	5
Uniform error (percent)	98.25	99.60	99.79	99.87	99.90
Gaussian error (percent)	98.53	99.64	99.82	99.88	99.91

Table 4.2: For uniform and Gaussian error, the percent of unmatched model points for which the distance between the similarity point and the true point is less than some tolerated number of pixels.

points will tend to contain less error. It turns out, however, that the results are only slightly better than for uniform (see Table 4.2), with the same average error of .06 pixels. It appears that for Gaussian error to improve over simple uniform error, the Gaussian distributions would have to be significantly more peaked around their nominal positions.

In conclusion, the two experiments on random model and image points indicate that the linear approximation is reasonably accurate for up to five pixels of error. Fig. 4-10 shows circular uncertainty regions that we computed using a real model and image of a telephone. By hand, we measured corners of the telephone to obtain a model and selected corresponding corners in the image. The three smaller circles are the error bounds for the matched image points. The remaining circles are the correct search regions for finding additional matches.

4.5.3 Similarity transform vs. a first-order approximation

We have used two first-order approximations. One relates changes in $\vec{m}_2^\pi$ to changes in pose. The second relates changes in pose to changes in $\vec{m}_3^\pi$. When we use both approximations, we obtain a similarity transform. We will now compare this to the results of using an exact determination of the effect of changes in $\vec{m}_2^\pi$ on pose, along with a first-order approximation to the subsequent effect of pose changes on $\vec{m}_3^\pi$. If the effects of these two first-order approximations are uncorrelated, we would expect to obtain more accurate results when we replace one with an exact value. On the other hand, if error in the first-order approximations tends to cancel, the similarity transform that follows from both approximations will tend to be more accurate. Section 4.4 showed analytically that this can happen in some circumstances.

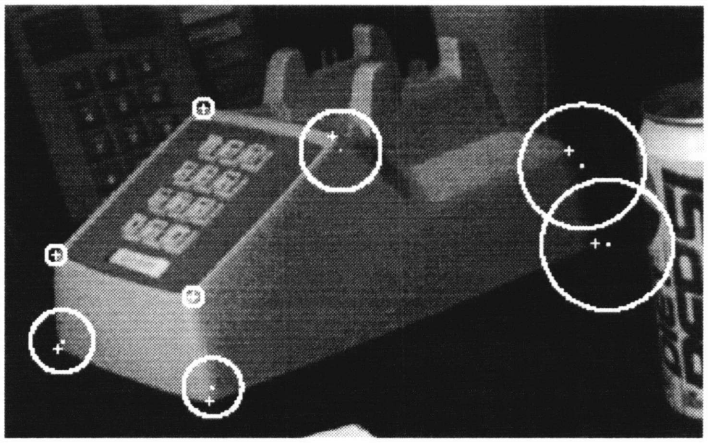


Figure 4-10: Circular uncertainty regions computed for points at the corners of a telephone. The three smaller circles show error regions of five pixels about the three points used to generate the model pose. The remaining, larger circles show uncertainty circles. Small crosses show the actual locations of the image points, which were selected by hand but are still a bit noisy. Small dots show the projected locations of the model points in the determined pose.

We now explore this possibility experimentally.

In the same way as in the previous experiment, we generate trials of random models and image triples, and project the random unmatched model points into the image. As in Section 4.4, we put error in only one of the basis points, say $\vec{\imath}_2$. To better compare the first-order and similarity approximations, at each trial we generate an error basis point as follows: We change either ϕ or θ from its value at the nominal pose until $\vec{\imath}_2$ moves 5 pixels from its nominal position. This gives us an error vector in the third image point. For each of the projected, unmatched model points we compute the scaled rotation matrix **A** as in Equation 4.21, and apply it to this error vector. This gives the error point predicted by the similarity transform.

To compute the error point predicted by a first-order approximation from pose space, we move the projected point along one of the tangent lines $\overrightarrow{t'_\phi}$ or $\overrightarrow{t'_\theta}$, defined in Section 4.3.3. The amount we move is determined by which pose parameter we changed to compute the error in $\vec{\imath}_2$. In particular, for changing ϕ the first-order error vector is $\Delta\phi\overrightarrow{t'_\phi}$ and for changing θ it is $\Delta\theta\overrightarrow{t'_\theta}$, where we know $\Delta\phi$ and $\Delta\theta$ exactly from our generation of the error basis point.

To measure propagated error, we first use the error basis point to calculate the

Absolute error (px)	99%: Changing	ϕ	θ	90%: Changing	ϕ	θ
Similarity trans.		0.57	0.51		0.09	0.09
1st-order approx.		1.02	1.67		0.26	0.45

Table 4.3: The amount of error allowed in the fourth point that includes 99% or 90% of the tested points. The error is the distance from the true location of the fourth point to the location predicted by the similarity transform (which is the approximation we suggest) or the location predicted by the first-order approximation. In this experiment there was error in one basis point, which was moved 5 pixels by changing either ϕ or θ from its nominal value.

two possible locations where the fourth point actually goes. To measure the error in the similarity transform, we calculate the distance from the point predicted by the similarity transform to the closer of the two actual points, and similarly for the first-order approximation. Table 4.3 shows the results from histogramming the error data over a series of 10,000 trials with seven projected points per trial. In the table, the similarity transform is about two to five times more accurate than the first-order approximation. This suggests that better results may be obtained by propagating error directly from the basis image points to the predicted locations of the fourth points. By first estimating the errors in pose space and then propagating these errors back to image space, some accuracy may be lost, unless special care is taken to make sure the appropriate high-order error terms cancel.

4.6 Perspective Projection

In this section we apply the same technique to perspective projection to obtain again a linear relationship between the errors, but in this case the form of the linear relationship is much more complicated. As before, we begin by looking at the effects of error in exactly one of the basis points. We assume we know the camera focal length f and the center of projection $\vec{c}$.

We introduce a similar representation for any third model point to the one we used in the scaled-orthographic case. Since $\vec{\imath}_0$ and $\vec{\imath}_1$ are fixed, the line segment between $\vec{m}_0$ and $\vec{m}_1$ is free to rotate and translate in the plane through $(\vec{c}, \vec{\imath}_0, \vec{\imath}_1)$, as long as $\vec{m}_0$ and $\vec{m}_1$ remain on the lines through $(\vec{c}, \vec{\imath}_0)$ and $(\vec{c}, \vec{\imath}_1)$, respectively (see Fig. 4-11).

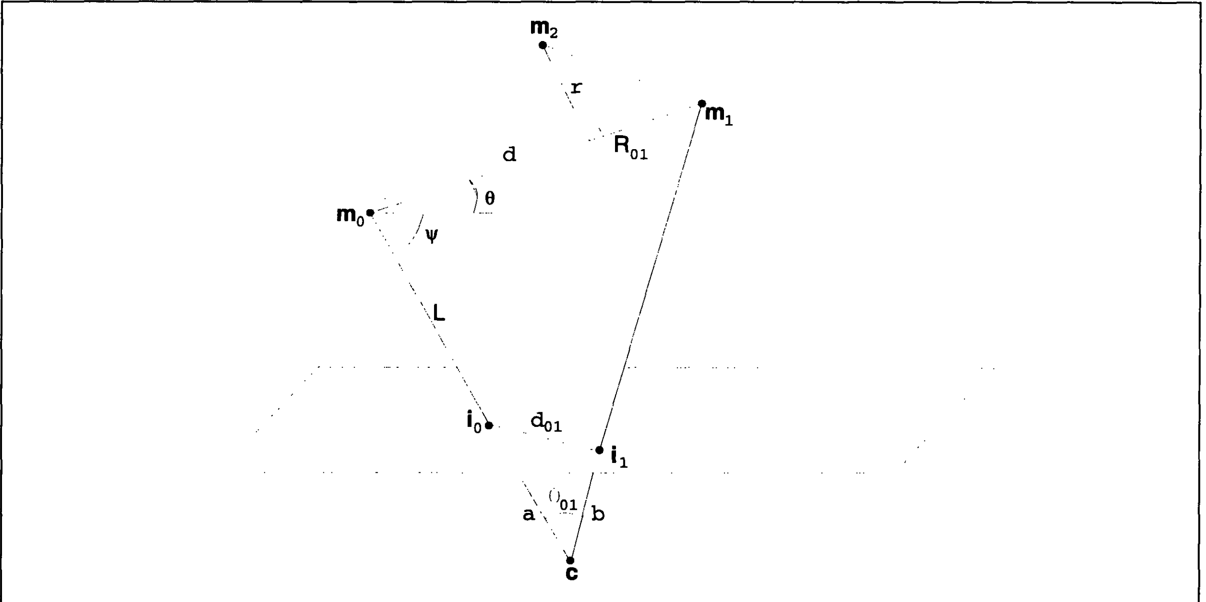


Figure 4-11: a, b, and L are the distances from $\vec{c}$ to $\vec{i}_0$, $\vec{i}_1$, and $\vec{m}_0$, respectively.

After this rotation and translation, the only remaining degree of freedom is a rotation about the line through $(\vec{m}_0, \vec{m}_1)$.

Initially, we rigidly transform the model so that $\vec{m}_0 = \vec{i}_0$, $\vec{m}_{01}$ points in the same direction as $\vec{i}_{01}$, and $\vec{m}_2$ is in the $z = 0$ plane. For the image coordinate system, we let the origin be at $\vec{i}_0$, the z direction be orthogonal to the image plane, and the positive x axis be along $\vec{i}_{01}$. Also, we let $\hat{n}$ be the unit vector that is normal to the plane through $(\vec{c}, \vec{i}_0, \vec{i}_1)$. Next the model is rotated by ϕ about the x axis, then rotated by θ about $\hat{n}$ (denoted by $\mathbf{R}_{\{\theta,\hat{n}\}}$), and then translated by $\vec{u}$. Let $\vec{p}$ equal the second model point after the ϕ rotation. Then using the relative coordinates for the model points from Fig. 4-2, $\vec{p} = (d, r\cos\phi, r\sin\phi)$, so that

$$\vec{m}_2 = \vec{u} + \mathbf{R}_{\{\theta,\hat{n}\}}\vec{p}. \tag{4.32}$$

For the translation $\vec{u}$, let L be the distance from $\vec{c}$ to $\vec{m}_0$, and let $\hat{v}$ be the unit vector pointing from $\vec{c}$ to $\vec{i}_0 = (0,0,0)$, which implies $\hat{v} = -\vec{c}/\parallel \vec{c} \parallel$. Then $\vec{u} = \vec{c} + L\hat{v}$, where L must still be determined. In Fig. 4-11, let ψ be the angle between $\vec{c} - \vec{i}_0$ and $\vec{i}_1 - \vec{i}_0$, and let θ_{01} be the angle between $\vec{i}_0 - \vec{c}$ and $\vec{i}_1 - \vec{c}$. Also let R_{01} be the distance between $\vec{m}_0$ and $\vec{m}_1$. Through some trigonometry, Appendix C computes that

$$L = \sin(\theta + \psi + \theta_{01}) \left(\frac{R_{01}}{\sin \theta_{01}} \right). \tag{4.33}$$

Let $\vec{m}_2 = (\overline{x}, \overline{y}, \overline{z})$. Since the image coordinate system is based at $\vec{i}_0$ with the camera center point at $\vec{c} = (c_x, c_y, -f)$, projecting $\vec{m}_2$ into the image gives

$$\vec{m}_2^\pi = (x, y) = \left(f \frac{\overline{x} - c_x}{\overline{z} + f} + c_x,\ f \frac{\overline{y} - c_y}{\overline{z} + f} + c_y \right). \tag{4.34}$$

From this equation, we can compute the partial derivatives of x and y with respect to θ and ϕ, which give $\overrightarrow{t_\theta}$ and $\overrightarrow{t_\phi}$. By substituting r' for r, d' for d, and $\phi + \tau'$ for ϕ, we can analogously compute $\overrightarrow{t'_\theta}$ and $\overrightarrow{t'_\phi}$. Then we can solve Equation 4.21 for $\mathbf{A}$, and we will get a linear transform relating the error in $\vec{i}_2$ to the error in $\vec{m}_3^\pi$ (see Appendix C): Let $x_\theta = \frac{\partial x}{\partial \theta}$, $y_\theta = \frac{\partial y}{\partial \theta}$, and similarly for x_ϕ, y_ϕ, x'_θ, y'_θ, x'_ϕ, and y'_ϕ. Then

$$\mathbf{A} = \frac{1}{x_\theta y_\phi - x_\phi y_\theta} \begin{bmatrix} y_\phi x'_\theta - y_\theta x'_\phi & -x_\phi x'_\theta + x_\theta x'_\phi \\ y_\phi y'_\theta - y_\theta y'_\phi & -x_\phi y'_\theta + x_\theta y'_\phi \end{bmatrix}. \tag{4.35}$$

If there is bounded, circular error in $\vec{i}_2$, the linear transform will produce an elliptical uncertainty region for $\vec{m}_3^\pi$, whose parameters could be determined analytically. This differs from the scaled-orthographic case, where the uncertainty region is a disc. To handle circular error in all three points for scaled-orthographic projection, we had to convolve together three circles. For perspective, we would have to convolve three ellipses. To handle bounded error in three basis points using perspective projection, we can apply the algorithm that will be proposed in Section 4.7.

For Gaussian error, on the other hand, the method in Appendix D applies equally well to linear transforms as to similarity transforms. As before, we get an analytic solution for the uncertainty region of a projected model point, and that uncertainty region is normally distributed. The only difference is that the normal distribution need not be circularly symmetric.

4.7 nth-Point Uncertainty Region

It has been shown [37] that recognition algorithms that use a small number of randomly-matched points to determine pose are sensitive to false positive identifications of objects, because these poses are not sufficiently stable and lead to large uncertainty regions. One solution to this difficulty is to use poses based on more information to derive smaller uncertainty regions. Assuming a bounded error model, this section shows how to compute the uncertainty region of an $n+1$'st model point given n matched model and image points, for any value of n. Our linearized models of projection allow us to determine linear constraints on the set of feasible error vectors consistent with a match between model and image points. Having expressed our knowledge about pose using linear constraints, we apply linear programming to optimize a set of objective functions whose solution tightly bounds the uncertainty region. In general, this technique applies to any linear projection model, including affine models and including the linearized perspective and weak-perspective models that we derived in this chapter.

To demonstrate the idea, we suppose that the error in each image point is bounded by a square of width 2ϵ. We emphasize, however, that the same reasoning will apply to any convex, polygonal error bound, so that we may approximate a circle, or any other convex error bound, as closely as we wish. With square error bounds, a match between image and model points can be consistent only if there exists a pose that brings the x- and y-coordinates of all matched points to within ϵ pixels of each other. Let $\vec{m}_i^{\pi} = (x_i^{\pi}, y_i^{\pi})$ be the projection of the i'th matched model point, in some nominal pose. Let $\vec{i}_i = (x_i, y_i)$ be the location of the corresponding image point. Also, let $\vec{e}_i = (x_i^e, y_i^e)$ be a vector representing the deviation between a model point's projected position in the nominal pose and its true position (see Fig. 4-5). Since we choose a pose by aligning the first three model points with image points, for $0 \leq i \leq 2$ this deviation is also the actual error that occurred in sensing the image points. For $i \geq 3$, $\vec{e}_i$ does not depend on where we have sensed the i'th image point, but rather is a function of the sensing error in the first three image points. We then model error by assuming a model point's true position, $\vec{m}_i^{\pi} + \vec{e}_i$, is within ϵ of its corresponding image point, $\vec{i}_i$. That is,

$$x_i - \epsilon \leq x_i^{\pi} + x_i^e \leq x_i + \epsilon \quad \text{and} \quad y_i - \epsilon \leq y_i^{\pi} + y_i^e \leq y_i + \epsilon \tag{4.36}$$

Previously, we used the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ to represent linear transformations between error in the first three points and the fourth point. We now let $\mathbf{A_i}, \mathbf{B_i}, \mathbf{C_i}$ be the corresponding matrices for the $i + 1$'st point (e.g., $\mathbf{A_3} = \mathbf{A}$). We let $\overrightarrow{a_i^1}$ and $\overrightarrow{a_i^2}$ be the first and second rows of the matrix $\mathbf{A_i}$, and define $\overrightarrow{b_i^1}, \overrightarrow{b_i^2}, \overrightarrow{c_i^1}, \overrightarrow{c_i^2}$ similarly. In both the cases of scaled-orthographic and perspective projections, we know that we can write

$$\vec{e}_i = (\overrightarrow{a_i^1} \cdot \vec{e}_0, \overrightarrow{a_i^2} \cdot \vec{e}_0) + (\overrightarrow{b_i^1} \cdot \vec{e}_1, \overrightarrow{b_i^2} \cdot \vec{e}_1) + (\overrightarrow{c_i^1} \cdot \vec{e}_2, \overrightarrow{c_i^2} \cdot \vec{e}_2), \tag{4.37}$$

for $3 \leq i \leq n - 1$. Consequently, for matched points 3 through $n - 1$, we may substitute a linear combination of the first three error vectors for $\vec{e}_i$, giving us additional constraints that the first three points must meet in order to lead to a consistent solution. In all, we get the following constraints:

$$\text{For } i \in [0, 2], \qquad -\epsilon \leq x_i^e \leq \epsilon, \qquad\qquad -\epsilon \leq y_i^e \leq \epsilon, \tag{4.38}$$

$$\text{For } i \in [3, n-1], \left\{ \begin{array}{l} x_i - \epsilon \leq x_i^{\pi} + \overrightarrow{a_i^1} \cdot \vec{e}_0 + \overrightarrow{b_i^1} \cdot \vec{e}_1 + \overrightarrow{c_i^1} \cdot \vec{e}_2 \leq x_i + \epsilon, \\ y_i - \epsilon \leq y_i^{\pi} + \overrightarrow{a_i^2} \cdot \vec{e}_0 + \overrightarrow{b_i^2} \cdot \vec{e}_1 + \overrightarrow{c_i^2} \cdot \vec{e}_2 \leq y_i + \epsilon. \end{array} \right. \tag{4.39}$$

This set of constraints can be satisfied if and only if there is a set of error vectors for the first three points that will bring all projected model points to within the error bounds of their matching image points.

We have formulated our knowledge of model pose, based on a match between n image and model points, in terms of linear constraints on the components of the three error vectors $\vec{e}_0$, $\vec{e}_1$, and $\vec{e}_2$. We may now use linear programming to maximize any linear objective function, subject to these constraints. In particular, we can formulate linear objective functions that express, for several directions, the errors in an additional model point's predicted position, and then extremize these functions. For example, if we maximize the linear objective function

$$x_n^e = \overrightarrow{a_n^1} \cdot \vec{e}_0 + \overrightarrow{b_n^1} \cdot \vec{e}_1 + \overrightarrow{c_n^1} \cdot \vec{e}_2, \tag{4.40}$$

we will find the maximum x displacement that the projection of the $n+1$'st model point can have from its nominal position. By maximizing the negation of this expression, and similar expressions for the y values, we may put a rectangle about the possible locations of the $n+1$'st point.

Using a similar method, we can in general place a convex polygon of any shape about the possible locations of the $n+1$'st point. Suppose we wish to bound the location of the $n+1$'st point in some direction other than along the x or y axis. Let $(\Delta x, \Delta y)$ be a vector in that direction. Then we may achieve this by maximizing the objective function formed by taking the dot product of $(\Delta x, \Delta y)$ and $\vec{e}_n$. By substituting our expression for $\vec{e}_n$ as a linear combination of the first three error vectors, we get a linear expression in these values. By finding the extreme values of the feasible positions of the $n+1$'st model point, we may put a convex polygon about these positions which will be more accurate than a square.

We should note that linear programming is very efficient. It is known to be polynomial time in the worst case. In practice, for problems with l variables and m constraints, the most common algorithm, simplex, is found to usually take time proportional to lm^2 (Strang [84]), and many highly optimized commercial implementations of simplex exist. Our problem has 6 variables and $4n$ constraints when n points are matched. When the number of variables in a problem is fixed and only the number of constraints grows, as in our case, there are algorithms that take linear expected time (see Seidel [81], for example).

When the errors in the image points are Gaussian distributed and there are more than three matched points, the Kalman filter can be used to recursively compute Gaussian distributions for the error vectors $\vec{e}_0$, $\vec{e}_1$, and $\vec{e}_2$, similar to Hel-Or and

Werman [45]. Given Gaussian distributions for $\vec{e}_0$, $\vec{e}_1$, and $\vec{e}_2$, Appendix D shows how to obtain a Gaussian distribution for the possible locations of any $n+1$'st point.

These methods could be quite useful to the indexing or alignment approaches to recognition that we previously described. To illustrate, a recognition system that uses the linear programming method might work as follows:

1. Match k image and model points, using a search or indexing method. Assume that the error in each image point is bounded by some m-sided polygon.

2. Use the matches to generate km linear constraints on the possible errors in the first three matched points.

3. For each unmatched model point, run l linear programs to compute an l-gon bounding the point's possible image locations. Look there for a matching image point.

Recognition systems commonly use more complex features such as line segments for verification, instead of points. As discussed in Section 4.3.4, it would be straightforward to use our results to bound the uncertainty regions of line segments by finding the uncertainty regions of their endpoints. Additionally, our results allow us to measure experimentally the extent to which additional point matches decrease our uncertainty about the location of unmatched points. We discuss this in the next section.

4.8 Experiments

To demonstrate the value of narrowing the feasible region in which model points may appear by using additional matches, we have implemented a test system for the case of weak-perspective projection. In this system, we match some image and model points. We then examine the regions in which additional points might appear. We can see how much smaller these feasible regions become as we derive additional constraints from more matches.

We first describe the results of this system on synthetic data. This allows us to systematically explore two key issues. First, since our linear transformation is only an approximation, how often might it cause us to make errors? Second, how much can the addition of further matches reduce the space in which we must search for even more matches?

We use the following experimental conditions. First, we generate random sets of seven model points inside a cube. We form an image by projecting these points orthographically, scaling so that the cube projects to a 1000×1000 square image. We then add error so that each sensed point shows up inside a circle of radius five pixels centered at the projected position of the point. For error, a uniform random distribution is used.

We then match the first three noisy image points and model points, and use this match to generate a noisy pose of the model. This pose is then used to compute the linear transformations describing the location of each additional model point as a function of the error in the first three image points. Of the two possible model poses that can be derived using a match of three points, we automatically select the one that is closer to the correct pose. In a real system, of course, we would have to explore both possibilities, and see which led to more confirming evidence.

Given these linear transformations, we allow for a square error bound of width 10 pixels around each error circle. As described above, we then compute a rectangular bound on the image location of each additional model point using linear programming. Next we check to see which image points actually appear within the predicted rectangular boundary. Since we have perturbed the image points within their error circles, any time we fail to find a model's image point in the predicted rectangle, this mistake must be due to limitations in our linear approximation. When we do find an image point, we record the size of the rectangle in which we looked.

We then augment our hypothesis by matching the fourth image and model points, and, using the additional constraints, we further narrow down the location of the remaining model points in the image. Again we keep track of how often we fail to find a model point in a predicted rectangle, and we record the areas of the successful rectangles. We continue this process with additional matched points. We repeat this experiment $2,500$ times, continuing to perturb each image point by up to five pixels, but allowing for varying levels of error in our predictions. We can use these results to see how many mistakes of the system could be eliminated by overestimating the expected error, and how much of a price we would pay for this by producing larger rectangles.

Table 4.4 lists the results. There are several conclusions we may draw. First, we see that few overall mistakes are made. The predictions are generally between 95% and 99% accurate. The significance of this will depend on exactly how we incorporate these error regions into a recognition algorithm. But typically, recognition systems search through many hypothetical matches between image and model points, and it is understood that a system may have to consider more than one correct hypothesis before recognizing an object. This is because even a correct set of matches may lead to an inaccurate pose. We can quantitatively see that our method of computing error regions leads to few such unstable poses.

Second, we can see that additional matches do provide considerable extra constraint in determining the locations of unmatched points. The most dramatic effect occurs when one matches a fourth point. This can reduce the size of possible error regions by a factor of fifteen or more. But even after the fourth point, there is a continuing significant benefit in using additional matches to constrain the error regions. These results also help us to assess in general the stability of poses generated by a small number of feature matches. We can see that if we use three points to compute a pose, small changes in these points can result in large changes in the locations of additional points. Poses computed from more points would be much more stable.

The error regions we compute are in general quite large. Several factors, however, may have exaggerated the sizes of the error regions. First, if we truly wish to consider error as bounded within a disc we should use polygonal error regions that more closely approximate a circle. Rectangles were used in this example only for the sake of simplicity. Second, a uniform error distribution bounded by five pixels may be pessimistic. In real systems, sensed image points probably tend to cluster around the point's true, error-free position, and the error may well be less than five pixels. Therefore a system that allowed for less error may produce much smaller error regions,

Error Allowed for	Num. Points Matched	Rectangle Average Area	Percentage Correct
5.0	3	16,700	98.9
	4	1,460	97.6
	5	823	96.2
	6	599	94.6
5.25	3	18,300	99.0
	4	1,600	97.8
	5	898	96.7
	6	655	95.5
5.5	3	10,800	98.9
	4	1,750	98.0
	5	969	97.0
	6	730	96.4
6.0	3	12,700	99.2
	4	2,200	98.8
	5	1,370	98.4
	6	948	97.9
6.5	3	12,400	99.2
	4	2,380	98.7
	5	1,410	98.4
	6	1,080	98.0
7.0	3	16,600	99.4
	4	2,950	99.0
	5	1,740	98.9
	6	1,350	98.8

Table 4.4: This shows the size of error regions computed as more points are matched, and the frequency with which noisy model points fail to appear in these error regions. Image points are always perturbed by a uniform error bounded by five pixels. The first column gives half the width of the square error bound that we allow for in each image point. The second column gives the number of matches used in computing the error region for an additional point. The third column gives the average size of this error region, and the fourth column gives the percentage of times that a model point's image shows up in this predicted error region.

without making many mistakes. Of course, allowing for less error should only make our linear approximation more accurate.

It is also interesting to note that the accuracy of the error regions drops a bit as we add more point matches. It seems that errors in the linear approximation accumulate as we compute the feasible set of error vectors. One way to compensate for this effect would be to allow for slightly more error as we use more matched points. For example, if we allow for 5 pixels of error when we match three points, we might allow for 6 pixels when matching four points. This would allow us to significantly reduce the size of the error regions, while keeping the error rate essentially constant.

As before, we have run this system on a real model and image to further illustrate its performance. Fig. 4-12 shows the resulting rectangular error regions for $\epsilon = 5.25$. The figure demonstrates how the uncertainty regions shrink as we match more points, while still containing the true image points.

In summary, we have used linear programming to compute the propagated uncertainty regions in simulation and in a real image for matches with more than three model and image points. The experiments demonstrate that additional matched points can significantly reduce the uncertainty regions with little loss in accuracy.

4.9 Applications

We have shown how to approximate the effect of changes in model pose using a linear relationship between the error vectors. For predicting the locations of unmatched points, we have demonstrated that this approximation is quite good within the range of error usually considered by object recognition systems. This suggests that for many recognition applications we may model this relationship linearly.

In past research, the use of linear projection models has led to algorithmic simplicity. Projection of a 3D object may be approximated as a 3D-to-2D affine transformation to gain the advantages of linearity, at the loss of fully capturing the rigidity of objects. Also, scaled-orthographic projection of a planar object is equivalent to a 2D affine transformation of the object, which is linear. Many algorithms have taken advantage of this linearity, either to find matches that are consistent with a bounded error model (e.g., [7, 21, 17, 54]) or to find likely sets of matches assuming Gaussian error (e.g., [76, 80, 96, 10]). We can now extend some of these algorithms to full 3D-from-2D recognition while maintaining object rigidity.

In Section 4.7, we outlined an algorithm which could be useful to most alignment and indexing approaches to recognition. Some alignment approaches use grouping methods to generate an initial match of more than three points (such as Lowe's [64], Roberts' [77], Jacobs' [55], and Wayner's [92]), and some alignment approaches create an initial alignment using only three points [77, 31, 64, 52, 91]. In the latter case, a recognition system might attempt to add matches, and use these additional matches to narrow the area in which it must search for still more consistent matches. Additionally, the algorithm from Section 4.7 may be useful in methods that match image to model features by indexing, and then verify these matches [60, 23, 55, 88, 78, 92]. In these approaches, some model features are matched to image features to determine a model pose, and then this pose is used to find matches for additional model features. Our results show exactly where to search for these matches when we have matched three image and model points.

As mentioned above, other approaches to recognition have derived linear constraints on model poses using linear projection models. The linear constraints were used to robustly match models and images in the presence of bounded uncertainty. This line of work originated with Baird [7], who considered models of 2D points undergoing 2D rotation, translation, and scaling. Baird used convex polygons to bound the errors in the image points. He then showed that, when we match an image point to a model point, each side of the polygon places a linear constraint on the set of feasible model poses, if the transformation from matched image points to projected, unmatched model points is linear.

Baird used these constraints as part of an interpretation-tree approach to recognition. His system searched a tree that represented all possible ways of matching image and model points. At each node of the tree, linear programming was used to decide whether the proposed matches were consistent with the polygonal error bounds. In

Section 4.7, we went beyond Baird, not only in handling the scaled-orthographic projection of 3D objects, but also in showing how to use linear programming to find the uncertainty regions of unmatched points.

Breuel [17] used a modification of Baird's approach to produce a tree-search algorithm that in the worst case runs in polynomial time. Cass [21] used linear constraints to show that finding the pose of the model that aligns the most image and model features to within error bounds is inherently a polynomial-time problem. Jacobs [54] showed how to perform a Hough transform in error space, instead of model pose space, by discretely computing the feasible region in error space (the space formed by the cross product of the error vectors in the first three image points).

Jacobs makes use of a linear relationship between error vectors that exists for the case of affine transformations of 2D objects. Our linearized perspective and weak-perspective models give us a linear relationship for 3D objects as well. As a consequence, Jacobs' method readily can be extended to 3D objects, and without increasing the dimensionality of the problem.

In addition to Jacobs' method, our linear relationship can be used to extend any of the above methods. To illustrate, we extend Cass' approach to the case of scaled-orthographic projection. Suppose we match three image to three model points and wish to know which pose will match the most additional model and image points. We know that these three matches give us simple linear constraints on the first three error vectors, just from the image points' error bounds. Now match each additional model point to each additional image point. These give us more linear constraints. Each linear constraint describes a 5D hyperplane in a 6D space of the possible error values: $e_{\{0,x\}}, e_{\{0,y\}}, e_{\{1,x\}}, e_{\{1,y\}}, e_{\{2,x\}}, e_{\{2,y\}}$. If we take each set of six linear constraints, the constraints in general will intersect at a point. This point corresponds to a set of error vectors for the first three points, and hence to a possible pose of the object. As Cass showed, if we now consider all of these poses, we will be guaranteed to find one that matches the most model points to image points. In fact we will find all poses that match the model to different collections of image points.

Cass has developed efficient heuristic algorithms for exploring the space of poses

Uncertainty Region	3 matched points	>3 matched points
Scaled ortho., Gaussian	Circularly-symmetric Gaussian	Gaussian
Scaled ortho., Bounded	Circle	Linear Programming
Perspective, Gaussian	Gaussian	Gaussian
Perspective, Bounded	Linear Programming	Linear Programming

Table 4.5: Propagated uncertainty regions for circularly-symmetric Gaussian and bounded errors in the image points. Our solution for the uncertainty region either is analytic, in which case a description of the analytic solution is given, or is numerical, in which case the solution can be found by Linear Programming.

in the case of a rigid 2D rotation and translation. For 3D recognition, the algorithm becomes costly, however. In our case, if we have m model points and n image points, for each of the $O(m^3n^3)$ initial matches we must consider $O(m^6n^6)$ poses. Hopefully, however, a polynomial-time formulation of matching for scaled-orthographic projection may lead to more efficient heuristic solutions, such as the ones that Cass has found in other domains. Alternately, Jacobs' approach of performing a Hough transform in error space may be more effective since error space is more compact.

4.10 Conclusion

This work will allow recognition systems to accurately take account of the effects of sensing error, during a process that finds supporting evidence to confirm a hypothetical set of matches. We showed that a linear approximation to scaled-orthographic projection is accurate when reasonable amounts of sensing error have occurred. In addition, we showed how to compute the propagated uncertainty regions for the rigid projection of a 3D model into a 2D image. The uncertainty regions for three matched points are described by a simple analytic expression, for the projection and error model cases of (scaled-orthographic, Gaussian), (scaled-orthographic, bounded), and (perspective, Gaussian). When more points are matched, there are simple and efficient algorithms for computing the uncertainty regions. Table 4.5 summarizes the results.

Both analytic results and experiments have demonstrated the value of accurately computing the uncertainty regions. The uncertainty region of a point can vary greatly depending on both the model geometry and pose. Therefore, any naive approach that uses an hypothesized pose to match additional model and image points is likely either to match many of the model points to image points they could not have produced, or to miss many image points they could have produced.

We found that uncertainty regions based on randomly matching only three points tend to be quite large, supporting past work [37, 4] that they will lead to many false matches. We also observed, however, that uncertainty regions shrink dramatically when we match even one more point, and still further when we match more. This demonstrates that matching larger sets of points, while being careful about error, can produce much more accurate recognition systems.

Finally, we extended several existing approaches to handling error in recognition systems, which were previously restricted to domains with linear projection models.

Availability

To facilitate the use of the results in this chapter, we have made available our C code for computing the 3D pose solution implied by 3 point correspondences under weak perspective, the three scaled rotation matrices (Equation 4.25), and the uncertainty circles (Equation 4.26). To retrieve the code, ftp to "ftp.ai.mit.edu," then log in as "anonymous," then cd to "pub/users/tda/," and then get and uncompress "alignment-code.tar.Z."

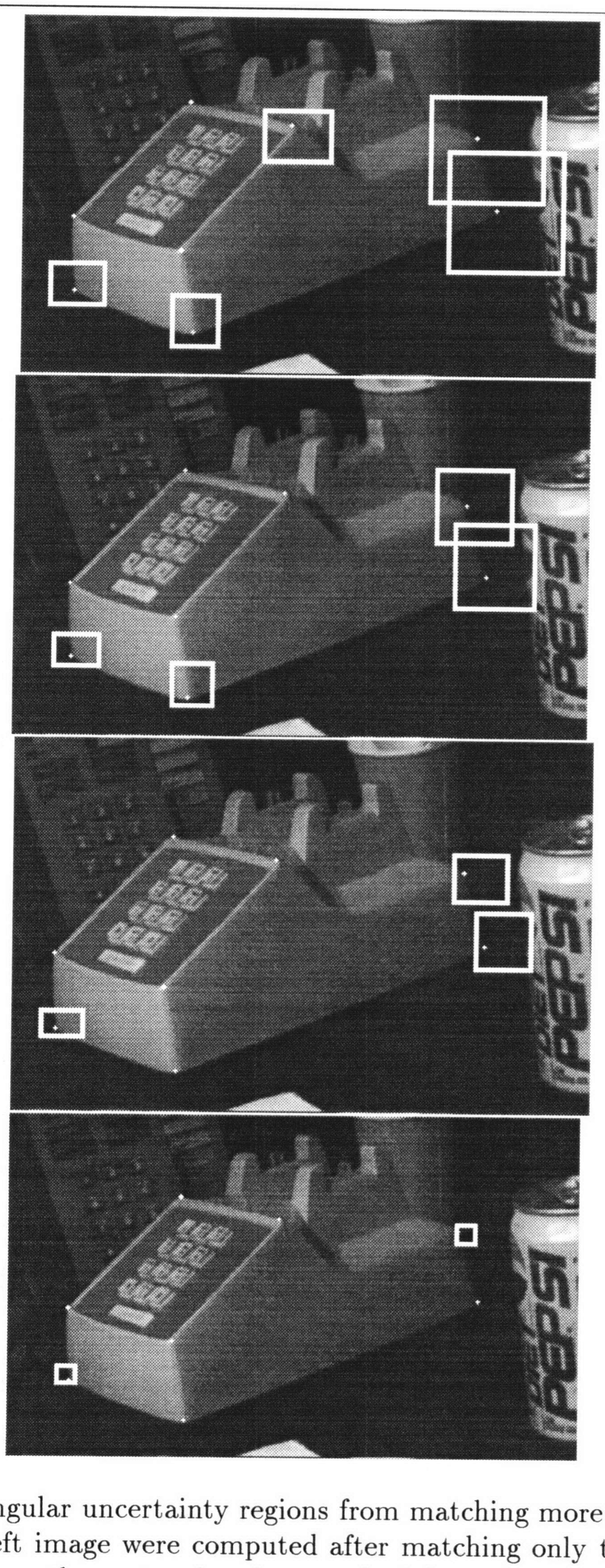


Figure 4-12: Rectangular uncertainty regions from matching more points. The rectangles in the top left image were computed after matching only three points. The following images show the rectangles that resulted from successively matching one more point. When an additional point was matched, we stopped computing rectangles for that point.

Chapter 5

On-Line Evaluation of Hypotheses in Alignment

This chapter proposes a new method for implementing alignment robustly. To motivate the approach, we review the basic model-based recognition problem discussed in Chapter 1. Briefly, we are given a 3D model composed of point and line features, and an image of a scene which may contain the modeled object, along with any amount of clutter. We run the image through a feature detector to get an image of points and lines similar to the ones in the model, except these point and lines contain errors due to the feature detection process. The problem is to find which image features, if any, correspond to model features and to exactly which model features they correspond.

Approaches to model-based recognition can be divided into three main classes. *Constrained search* searches the space of all possible correspondences between model and image features (e.g., [13, 32, 71, 39, 40, 46]). It has been shown that most implementations take exponential time in the numbers of model and image features [33], though it can be done in high-polynomial time [17]. *Transformation clustering* searches the space of all possible transformations between model and image features (e.g., [8, 89, 88, 63]). Although most implementations take low polynomial time, they are sensitive to false positives, that is, they are likely to incorrectly identify the model [34]. There are implementations of this technique that are robust to false positives, but they require high-polynomial time [21]. The third technique is *alignment*, which is low polynomial, and one of the goals of this chapter is to show how to make alignment robust as well.

5.1 Our Approach

We consider an alignment method in which poses are based on minimal sets of features. As in the last chapter, we use points for generating sets of correspondences (hypotheses), and we verify the hypotheses using extended features as well as points. Hypotheses, then, contain three point pairs, since three is the minimal number needed to compute a pose. We model the way in which a 3D object is brought into the image as a rigid transformation followed by weak-perspective projection (orthographic projection plus a scale).

Chapter 1 discussed how alignment can be viewed in the context of a tree search (see Fig. 1-2). Given a minimal set of features, the alignment method computes the model pose and then uses it to project each unmatched model feature into the image to locate a corresponding data feature. In so doing, the method considers all additional pairings of model and image features that are consistent with the first three pairings up to error. In terms of the search tree, this corresponds to proceeding down one level beyond the minimal set, and so global consistency is not guaranteed. Given a hypothesis, the key steps of alignment are to

1. decide which image features could correspond to each unmatched model feature, and
2. use the candidate image features to accept or reject the hypothesis.

Huttenlocher and Ullman [52, 51] found candidate image features by looking in a region of fixed size and shape about each unmatched model feature, and then accepted or rejected the hypothesis using a threshold on the number of model features for which candidate image features were found. Alter and Grimson [4] showed that the correct search (uncertainty) region, however, can vary considerably in size, using only simple, circular bounds to model error in the image points. For computing the regions, Chapter 4 gave a solution that is efficient and easy to use.

Having computed the correct search regions, the next step is to use the candidate matches to evaluate the hypothesis. To do this, we make use of the *likelihood ratio test* [87]. Given a hypothesized pairing of three model and image features, we must decide whether to accept or reject this hypothesis, where accepting implies we have found an instance of the model. Let H be the event that the given three-point match is correct. The decision is whether H or $\overline{H}$ is true given the image, which we call event I. So we choose H if $Pr[H|I] > Pr[\overline{H}|I]$, and otherwise we choose $\overline{H}$. Using Bayes' rule, this decision can be rewritten as

$$\frac{Pr[I|H]}{Pr[I|\overline{H}]} \underset{\text{pick } \overline{H}}{\overset{\text{pick } H}{\gtrless}} \frac{Pr[\overline{H}]}{Pr[H]},$$

where this notation says to choose H if $>$ is true and otherwise choose $\overline{H}$. This rule is the likelihood ratio test. The test minimizes the probability of making an error, and consequently treats false positives (choosing H when $\overline{H}$ is true) and false negatives (choosing $\overline{H}$ when H is true) equally. To penalize one of these more than the other, we can associate costs with correct and incorrect decisions: Let C_{XY} be the cost of choosing X when Y is true. Minimizing the expected cost generalizes the above rule to

$$\frac{Pr[I|H]}{Pr[I|\overline{H}]} \underset{\text{pick } \overline{H}}{\overset{\text{pick } H}{\gtrless}} \frac{Pr[\overline{H}]}{Pr[H]} \left(\frac{C_{H\overline{H}} - C_{\overline{H}\overline{H}}}{C_{\overline{H}H} - C_{HH}} \right).$$

This generalized rule reduces to the previous one if we set the costs of deciding correctly, C_{HH} and $C_{\overline{H}\overline{H}}$, to 0, and if we make equal the cost of a false positive, $C_{H\overline{H}}$, and the cost of a false negative, $C_{\overline{H}H}$. In general, the user of the system will pick these costs. Therefore if we can compute the ratio $Pr[I|H]/Pr[I|\overline{H}]$ and the a priori probability $Pr[H] = 1 - Pr[\overline{H}]$, then the above rule tells how to judge the hypothesis.

5.2 Our Domain

We are searching for an instance of a model in the image, and all image features that do not come from the model are spurious. To evaluate $Pr[I|H]/Pr[I|\overline{H}]$ and $Pr[H]$, we need to define more precisely what we mean by events I and H, which we will do in Secs. 5.4-5.6 for different types of features and error distributions. In general we will suppose that there are two processes by which image features are produced: If the hypothesis is correct, then we know that a "model process" produced an image feature for each unmatched model feature. By this process, image features are produced by first projecting 3D model features into the image and then corrupting them according to a noise distribution. The noise distribution allows for errors in feature detection, which results in uncertainty in feature position and, for line features, in orientation and length.

All image features that do not come from the model process are produced by a "random process." This process produces image features by dropping spurious features uniformly over the image. In truth, when the hypothesis is correct, spurious features arise from events such as object edges that are not represented in the model, other objects in the scene, shadows, and specularities. When the hypothesis is incorrect, spurious features also arise from edges of the model. So we are assuming that the features these events introduce effectively occur at random. This assumption has been made before for analyzing the verification stage of 2D recognition, and has yielded accurate results [36].

It is important to understand that the event I is dependent on the recognition algorithm. For instance, for Huttenlocher and Ullman's alignment algorithm, I is the event that a particular number of search regions contain one or more features, since this is the only information that Huttenlocher and Ullman use to evaluate the hypothesis. Given the number of non-empty search regions, the hypothesis is accepted if and only if the number is greater than a pre-chosen threshold. The alignment algorithms examined in [36], [38], and [4] evaluate a hypothesis using a threshold on the probability of a false positive, that is, of a random feature arising in each of the search regions. In these algorithms, I is the event that a false positive occurs.

Using the likelihood ratio test to evaluate the hypothesis would improve over using a fixed threshold, but we could improve even further by taking advantage of more information. To use all possible information, we would define I to be the event that the image features arose at their precise locations. Computing the probability of this event, however, requires enforcing all the constraints between model features, which is what the Backtracking Search algorithm does. As we discussed in Sections 1.1.2 and 5.1, the idea of alignment is that we can ignore the constraints between the unmatched model features. This assumes that their projected image locations can be treated independently. A choice of I that is consistent with this alignment philosophy, and uses more information than [52], [36], [38], and [4], is the event that the actual numbers of image features arose in the uncertainty regions. We investigate this choice in Sec. 5.4 for bounded error.

If the noise distributions for unmatched model points are known, for example, to be Gaussian, we could set I to be the event that the sum of the Gaussian-weighted contributions from the image points is a certain value (as in [80]). Even though this measure is somewhat intuitive, it is an arbitrary choice. In Sec. 5.5.2, we look in detail at the question of the appropriate measure for Gaussian or any distributed error.

We note that $Pr[I|H]/Pr[I|\overline{H}]$ measures how much relevant information is contained in our choice of I. Specifically, it measures how effective is the choice of I at distinguishing the images where the model is present from ones where it is not.

5.3 Uncertainty Regions and Selectivity

Chapter 4 showed that the uncertainty region for a model point generally is a disc centered at the nominal location, and gave a formula for its radius. Chapter 4 also mentioned that we can use this result to bound the uncertainty in predicted line segments. In particular, we assume that the orientation of an image line segment is constrained such that its endpoints lie within ϵ circles. Then, for each model line segment we calculate the uncertainty circles for its endpoints. Next, we find candidates for each model segment by gathering all image line segments that lie entirely within the uncertainty region formed by the uncertainty circles and their common outer tangents (Fig. 4-1) and whose extensions intersect both of the uncertainty circles.

We define the *selectivity* of an uncertainty region to be the probability that a feature distributed randomly over an image falls into an uncertainty region [38]. This quantity will be used in the next section to compute $Pr[I|H]/Pr[I|\overline{H}]$. For point features, the selectivity is the area of the region divided by the image area, $\frac{A}{A_I}$, where the area of the region takes into account the uncertainty in the unmatched image points by expanding the propagated region outwards by ϵ.

For line features, the selectivity is the chance that a spurious line segment randomly falls into a line uncertainty region. For an image line segment of any given length, [4, 3] derived a formula for the line selectivity (repeated for convenience in Appendix E). The formula computes the selectivity to be the range of rotations and translations that both keep the line segment within the line uncertainty region and the line segment's extensions intersecting the two uncertainty circles, divided by the range of rotations and translations over the entire image.

5.4 Image Probabilities for Bounded Error

This section computes the image probabilities $Pr[I|H]$ and $Pr[I|\overline{H}]$ for the basic alignment method discussed in Sec. 5.1. Since this alignment method does not care where in an uncertainty region that an image feature arises, we define the image I to be the event that the particular numbers of image features were found in the regions, ignoring their locations. As in Sec. 5.1, H is the event that the three-point match is correct. We assume for simplicity that the search regions do not intersect. Let k be the number of unmatched model features whose search regions are non-empty, and let r equal the number of unmatched image features. Further, let μ_i denote the selectivity of region R_i, and r_i be the number of image features found in R_i, for

$i = 1, 2, \ldots, k$. Also, let

$$r_{k+1} = r - \sum_{i=1}^{k} r_i \tag{5.1}$$

$$\mu_{k+1} = 1 - \sum_{i=1}^{k} \mu_i \tag{5.2}$$

μ_{k+1} is the selectivity of the background. In the domain of Sec. 5.2, if the hypothesis is incorrect, then all features in the uncertainty regions arose at random. For uniformly distributed features, the chance of r_1 features landing in R_1, r_2 landing in R_2, ..., r_k landing in R_k, and r_{k+1} landing in the background is $\mu_1^{r_1}\mu_2^{r_2}\cdots\mu_{k+1}^{r_{k+1}}$. The number of ways to choose $r_1, r_2, \ldots, r_k$ features from r is given by the multinomial coefficient,

$$\binom{r}{r_1, r_2, \ldots, r_{k+1}} = \frac{r!}{r_1! r_2! \cdots r_{k+1}!},$$

so that

$$Pr[I|\overline{H}] = \binom{r}{r_1, r_2, \ldots, r_{k+1}} \mu_1^{r_1}\mu_2^{r_2}\cdots\mu_{k+1}^{r_{k+1}} \tag{5.3}$$

Next, assume that if the hypothesis is correct, then the model features the system found matches for were not actually occluded. Then we get $Pr[I|H]$ by just subtracting one feature from every propagated region:

$$Pr[I|H] = \binom{r-k}{r_1 - 1, r_2 - 1, \ldots, r_k - 1, r_{k+1}} \mu_1^{r_1-1}\mu_2^{r_2-1}\cdots\mu_k^{r_k-1}\mu_{k+1}^{r_{k+1}} \tag{5.4}$$

Dividing Equation 5.4 by 5.3,

$$\frac{P[I|H]}{P[I|\overline{H}]} = \frac{(r-k)! r_1 r_2 \cdots r_k}{r!} \left(\frac{1}{\mu_1\mu_2\cdots\mu_k}\right) \tag{5.5}$$

Note that the number of multiplications is about equal to three times the number of regions k, which is relatively small.

5.5 Image Probabilities for Distributed Error

The alignment algorithm we have considered so far looks only at the numbers of image features within the uncertainty regions as evidence for the model. This treats all of the image features in a region as being equally likely to come from the model. However, we might expect image features that are closer to the nominal location of the predicted model feature to be more likely. If so, taking this likelihood into account could increase the effectiveness of alignment.

For distributed error, we must take into account the actual positions of the image features within the uncertainty regions. And we will do so in a way that is consistent with the alignment approach, so that we do not have to enforce global consistency. In particular, we re-define I to be the event that the image features arose at their particular locations and that the model process produced features in the regions independently of one another. As before, if the hypothesis is incorrect, then all the image features arose randomly. Let ρ be the probability that a feature dropped randomly into the image lands at a particular location: ρ equals $\frac{1}{A_I}$ for points and $\frac{1}{V_I}$ for lines. Then the probability of the r image features landing at their particular locations is ρ^r. Since there are $r!$ possible orderings of the image features, the probability of r spurious features occurring at the r locations is

$$Pr[I|\overline{H}] = r!\rho^r \tag{5.6}$$

Now suppose that the hypothesis is correct. Then for each search region in which there is at least one image feature, one image feature came from the model process and the rest occurred randomly. As before, let R_i be the search regions in which at least one image feature occurs, for $i = 1, 2, \ldots, k$, and let r_i be the number of features found in the ith search region. We number the features in region R_i from $t_i = 1$ to $t_i = r_i$, for $i = 1 \ldots k$. Also, let M_i represent the ith model feature, and let $D_i(t_i)$ be the propagated distribution for the uncertainty region R_i, which we assume is known: $D_i(t_i) = Pr[M_i = t_i]$. We consider the probability of a particular assignment to the model features: $\{M_i = t_i, \text{ for } i = 1 \ldots k\}$. For the remaining features, there are always $(r-k)!$ ways for them to have landed in the image at random. Then the probability that the model process produced the particular assignment, $\{M_i = t_i,\ \text{for } i = 1 \ldots k\}$, and that the random process produced the rest is

$$Pr[M_1 = t_1 \wedge M_2 = t_2 \wedge \cdots \wedge M_k = t_k](r-k)!\rho^{r-k} =$$
$$D_1(t_1)D_2(t_2)\cdots D_k(t_k)(r-k)!\rho^{r-k},$$

where independence comes from our choice of I. For each region, the image feature that was produced by the model is free to be any of the r_i candidates, where we assume again that the regions are disjoint. Summing over all of the possibilities for the correct image features,

$$\begin{aligned} Pr[I|H] &= \sum_{t_1=1}^{r_1} \cdots \sum_{t_k=1}^{r_k} D_1(t_1)\cdots D_k(t_k)(r-k)!\rho^{r-k} \\ &= (r-k)!\rho^{r-k}\left(\sum_{t_1=1}^{r_1} D_1(t_1)\right)\cdots\left(\sum_{t_k=1}^{r_k} D_k(t_k)\right) \qquad (5.7) \end{aligned}$$

Then

$$\frac{Pr[I|H]}{Pr[I|\overline{H}]} = \frac{(r-k)!}{r!}\frac{1}{\rho^k}\left(\sum_{t_1=1}^{r_1} D_1(t_1)\right)\cdots\left(\sum_{t_k=1}^{r_k} D_k(t_k)\right) \qquad (5.8)$$

5.5.1 Uniform Uncertainty

If the model features are uniformly distributed as well as being independent, then the actual locations of the image features in the uncertainty regions are irrelevant, as in the case of bounded error. Note that having uniformly distributed model features is not the same as the errors in the image points being uniformly distributed, a case which we do not consider. If $\overline{H}$ is true, then Eq. 5.6 gives $Pr[I|\overline{H}]$. If H is true, then, in each region, one point occurred due to the model and the rest arose randomly. Let ρ_i be the probability that a feature dropped randomly into region R_i lands at a particular location, for $i = 1 \ldots k$. For points, $\rho_i = \frac{1}{A_i}$, and for lines, $\rho_i = \frac{1}{V_i}$. Since the uncertainty is uniform, Eq. 5.7 becomes

$$\begin{aligned} Pr[I|H] &= (r-k)!\rho^{r-k}(r_1\rho_1)(r_2\rho_2)\cdots(r_k\rho_k) \\ &= r_1 r_2 \cdots r_k (r-k)!\rho_1\rho_2\cdots\rho_k\rho^{r-k}, \end{aligned} \tag{5.9}$$

so that

$$\frac{Pr[I|H]}{Pr[I|\overline{H}]} = \frac{(r-k)!r_1 r_2 \cdots r_k}{r!}\left(\frac{\rho_1\rho_2\cdots\rho_k}{\rho^k}\right), \tag{5.10}$$

which equals Eq. 5.5 for bounded error, since $\mu_i = \frac{\rho}{\rho_i}$.

5.5.2 Gaussian Uncertainty

Gaussian uncertainty distributions are of particular interest, since when there is Gaussian error in the image points, the uncertainty distributions for point features generally are Gaussian as well, and Chapter 4 gave a closed-form solution for the distributions: In Eq. 5.8, let

$$D_i = G_i = \frac{1}{2\pi\sigma_i^2}e^{-\frac{d^2}{2\sigma_i^2}},$$

where d is the distance of the image point to the the nominal image location of the unmatched model point, and where σ is given in Chapter 4. Then

$$\frac{Pr[I|H]}{Pr[I|\overline{H}]} = \frac{(r-k)!}{r!}A_I^k\left(\sum_{t_1=1}^{r_1} G_1(t_1)\right)\cdots\left(\sum_{t_k=1}^{r_k} G_k(t_k)\right) \tag{5.11}$$

For line features, we do not have an analytic solution for how Gaussian error in the image points propagates. For analysis, we could estimate the distribution of line features numerically by sampling. For use in a system, we could then try to fit an analytic function to the distribution.

5.6 Hypothesis Probability

$Pr[H]$ is the a priori probability that the three-point hypothesis is correct. The hypothesis is correct if the three model points are visible and if they happen to project within the error bounds of their corresponding image points. Let V be the event that the model points are visible. Then

$$Pr[H] = Pr[H|V]Pr[V]$$

Assuming that the model is equally likely to be seen from any point on a viewing sphere, we could estimate $Pr[V]$ for a given model by computing the fraction of points on the sphere from which the triple is visible. Then we could estimate $Pr[H|V]$ by the fraction of visible viewpoints from which the three projected model points can be scaled, rotated in 2D, and translated in 2D to lie within the error circles of the three image points. These estimates could be obtained by sampling the viewing sphere. For each sample, a non-trivial problem is how to compute whether three 2D points can be rigidly-transformed and scaled to lie within the error bounds of three other 2D points. It is possible to do this with linear programming.

It will be useful to see how important is $Pr[H|V]$ at distinguishing correct hypotheses. Its importance will depend on the effectiveness of the "probabilistic peaking effect," studied by Ben-arie [9] and applied to alignment by Olson [70]. The point of the probabilistic peaking effect is that, although all configurations of image triples can be produced by the model triple, some of these configurations may be much more likely.

5.7 Discussion

It would be informative to check whether the formulas we derived exhibit the intuitive behavior described in Section 1.2.1. In particular, we expect a hypothesis to be less likely to be correct if the uncertainty regions are larger or if there are more random features in the image. In either case, the features in the uncertainty regions could more easily have arisen at random. In addition, we would expect the likelihood of a hypothesis being correct to increase with size of the image, since as the image grows it becomes less likely that the the features fell into the same-sized uncertainty regions.

Consider the case of point features and bounded error, in which Equation 5.5 becomes

$$\frac{P[I|H]}{P[I|\overline{H}]} = \frac{r_1 r_2 \cdots r_k}{r(r-1)\cdots(r-k+1)} \left(\frac{A^k}{A_1 A_2 \cdots A_k} \right).$$

This quantity decreases with A_i and r and increases with A, as desired.

Chapter 6

Applications

In Chapter 1 we discussed an approach for recognizing three-dimensional objects from two-dimensional images. We pointed out that many of the techniques assumed by the approach were already in place, including techniques for 3D pose solution, alignment, and indexing. In addition, we argued that, for the approach to be effective, there were two fundamental problems which required further study, namely saliency with a close link to grouping and error propagation. Over the next four chapters, we investigated and developed new solutions to these two problems. As a result, we are now in a position to put these solutions together with the existing techniques and demonstrate a complete recognition system. Nevertheless, we have not built such a system. In this chapter, we describe (without building) how we envision the solutions in this thesis fitting into a specific recognition system.

First we summarize the procedure that we propose for recognizing three-dimensional objects from grey-scale images, and next we will detail the steps for a specific example. The input to the imagined system is an intensity image and library of models composed of points and line segments. As illustrated in Fig. 6-1, the image is first preprocessed to obtain a binary edge map. Then a grouping system is used to extract curves which are likely to come from the same object. From each curve, a set of feature points is extracted, which could be corner points at the intersections of line segments or invariant points on curves (e.g., zeros of curvature, which are always invariant, or curvature extrema, which are invariant over limited ranges). Each group of points is then used to index into the library of models to extract those sets of models points which produce the group of image points under projection (as in [55]). Since some of the points in any group of image points may be spurious (i.e., not correspond to feature points in the model), indexing shall be also be performed using subsets of the points in each group, if larger groups do not lead to recognition. For each set of correspondences between model and image points obtained from the indexing step, a robust alignment is performed to verify the correspondence.

Next, we describe in detail the steps of the imaginary recognition algorithm. Although at times we will provide pictures of example input and output of the various steps, these pictures were created artificially by the author for the sake of illustration, and in particular are not the results of applying any existing code. For concreteness in our description, we choose a particular object to be recognized from a particular scene image. For the object, we use the clamp which appeared in a cluttered real image in Chapter 3. This image, shown again on the left in Fig. 6-2, will serve as the input to the recognition process. Our basic recognition approach consists of grouping followed by 3D alignment, as follows.

1. Grouping stage

   (a) Saliency. The shortest-paths saliency method converts the input image into a saliency map that emphasizes edge points which lie on salient closed curves (center image in Fig. 6-2). In addition, the saliency method delivers, for each edge point, the most salient closed curve through that point.

   (b) Focus of attention. A simple method for narrowing the set of curves to be considered is to threshold the saliency map (Fig. 6-2-right).

   (c) Curve selection. We begin with the set of points in Fig. 6-2-right along with the optimal cycle through each such point. For selecting out a few salient image curves, we propose the following voting scheme: Each point casts a vote for its own optimal curve. After collecting each total number of votes, we divide by the curve length (to prevent long curves from being overly weighted). The curves are then considered in decreasing order of average number of votes. The idea behind this scheme is that a salient curve generally will have a large percentage of its contour elements choosing that curve as the most salient curve. Although we have not implemented this grouping scheme, we imagine its output to look like the leftmost picture in Fig. 6-3.

(d) Feature detection. The task of feature detection is to select feature points from the set of image curves. These points will be used as a basis for pose computation and alignment. Since the clamp's edges are long and straight, this property can be used as a constraint. In particular, for each curve point we would apply a dedicated line-fitting process that looks back in the intensity image at the area around the curve to find long, straight line segments (see for example Burns et al. [20]). The endpoints of these segments become our basis points. We suppose the output from this step looks like that shown in Fig. 6-3-center.

2. Alignment stage. The alignment stage expects as input several groups of feature points, where the points in each group are mostly from a single object. It is likely, however, that some points in a group shall not be from the object, due to occlusion or poor feature detection. For each group, we consider all triples of points from the group and match them to all triples of model points in turn. If the groups contain more than three points, then we would first try indexing with larger subsets of the points. In any case, for the remainder of this discussion, we just use minimal triples. In Fig. 6-3-center, there are three groups of points, two of size four, and one of size six. This gives $2\binom{4}{3} + \binom{6}{3} = 128$ image triples to try. For each unique pairing (hypothesis) of image and model triples, the alignment stage uses the pairing to bring the model into the image for verification. The verification utilizes the results of Chapters 4 and 5 to achieve robustness. Specifically, the system repeats the following steps for each hypothesis:

   (a) Pose computation and model projection. Using the hypothesis, compute the model-to-image transformation and project all of the model line segments into the image. This is illustrated on the right in Fig. 6-3. Note that the modeled line segments do not need to cover the entire boundary of the object. The picture shows a case in which the pose is correct.

(b) Uncertainty regions. Use Equation 4.26 from Chapter 4 to compute the uncertainty circles for the two endpoints of every projected line segment. Then construct a tight overestimate of every line segment's uncertainty region from the two uncertainty circles (see Fig. 4-1).

(c) Selectivity computation. For each projected model line segment, compute the selectivity of its line uncertainty region using the formula in Appendix E.

(d) Evaluation of the hypothesis. Gather all image line segments that lie entirely within the uncertainty region formed by the uncertainty circles and their common outer tangents. Then use Equation 5.5 and the likelihood ratio test to accept or reject the hypothesis, as described in Chapter 5.

The accepted hypotheses are returned as the recognized instances of the model.

The framework underlying this recognition system applies to a large but nonetheless specific class of images and objects. In particular, the objects should be rigid and contain a number of long, sharp edges, so that these edges can be modeled accurately with sets of rigidly-connected geometric features. Man-made objects like those in Figs. 3-10 and 3-11 satisfy this condition. Furthermore, since ultimately identification of an object is based on its edge contours, these edges must be rich enough to distinguish the objects of interest in the image. This places some constraint on both the types of objects and the amount of clutter in the images. Another limit on the set of images arises from our saliency mechanism. In particular, appropriate images are ones where the modeled objects produce the most salient shapes in the image, in terms of the size, smoothness, and connectedness of its bounding contours. These requirements steer the immediate domain of applicability to tasks where the environment can be controlled. Even so, the task itself may include the problems of occlusion, noise, and scene clutter. Tasks of this form arise most naturally in industrial settings, including the inspection of parts for metric accuracy, the identification of defects in circuit boards, and the recognition and localization of objects on a conveyer belt (perhaps for subsequent manipulation by a robotic arm).

As a specific example, Grimson et al. [41] recently reported a model-based system that accurately inspects three-dimensional, bent cylindrical sections, which are used in aircraft. The system performed alignment-based recognition and incorporated a bounded error model. For tube lengths averaging 6" and diameters in the range $\left[\frac{3}{16}'', 3''\right]$, the tube axis was consistently estimated within $\pm .008''$. The recognition method relied on range data obtained from a laser light scanner. The results in this thesis would enable one to build a similar system that could work from two-dimensional images obtained from an off-the-shelf CCD camera.

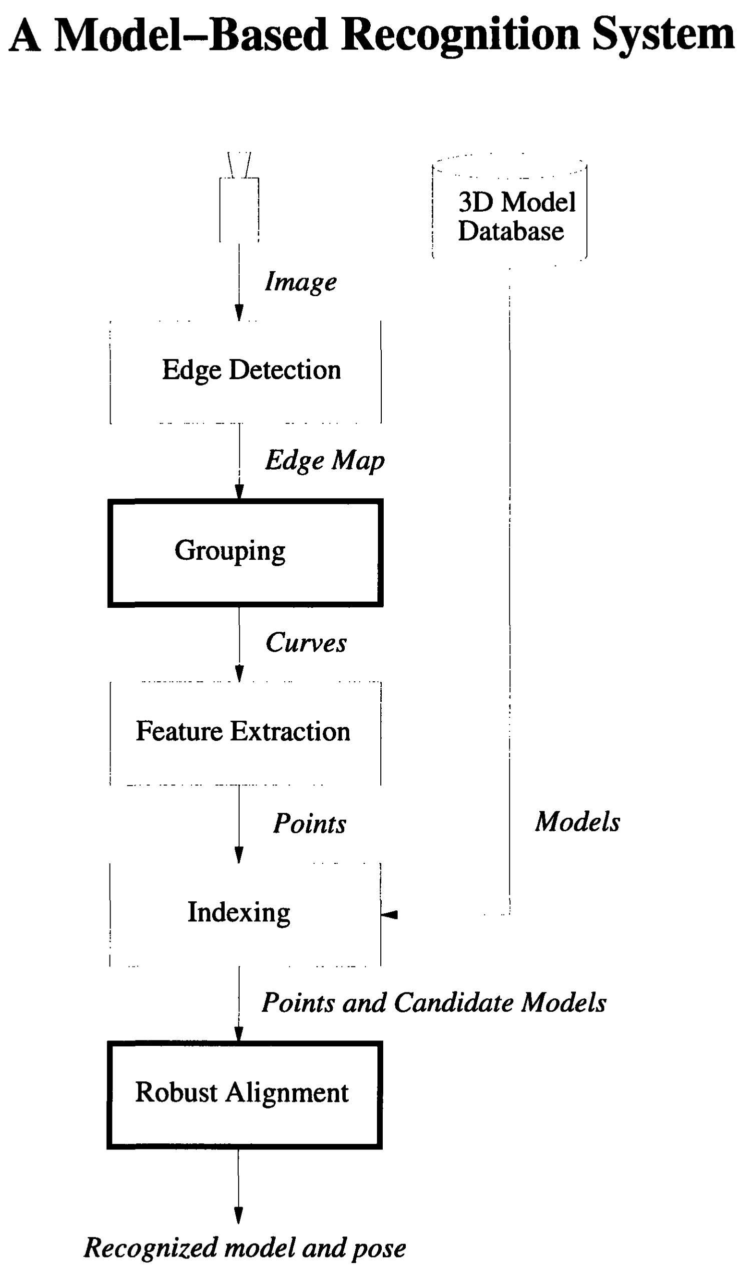


Figure 6-1: Flowchart for a model-based recognition system. The input to the system is a grey-scale image and a database of three-dimensional models. The system's goal is to return, for any instance of a model in the image, the model itself and its pose. The thicker boxes show the steps to which this thesis contributes. In particular, the grouping box is comprised of a saliency mechanism for focusing attention on edges that are likely to correspond to modeled objects, and a curve-selection process for choosing salient curves through those edges. After grouping determines a set of salient curves, the feature detector picks invariant points from those curves. Each group of points is used to index into the database to find the models which could produce those points. Given a set of matched model and image points, the robust-alignment step uses the error-propagation solution in this thesis to verify the presence of the model in the image while controlling propagated error.

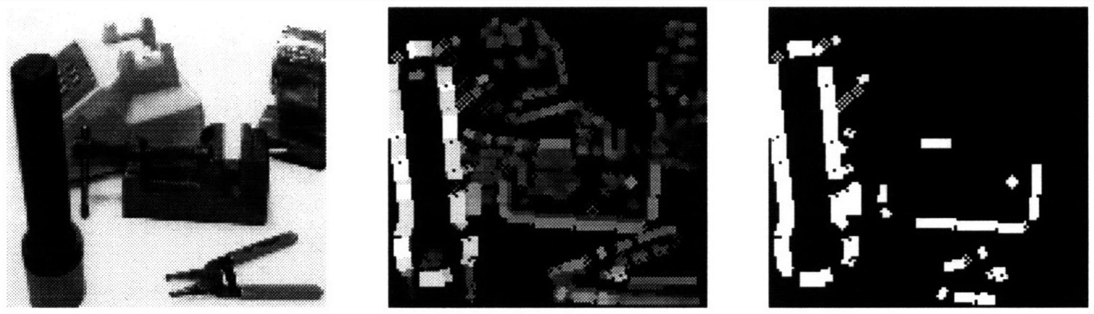


Figure 6-2: Left: Input is a real image of a scene containing occlusion and multiple objects. Center: Saliency map obtained by the shortest-paths method (max saliency 1.137, average positive saliency 0.5675). Right: Binary image for informing curve selection where to focus its attention, which was obtained by thresholding the saliency map (threshold 0.7).

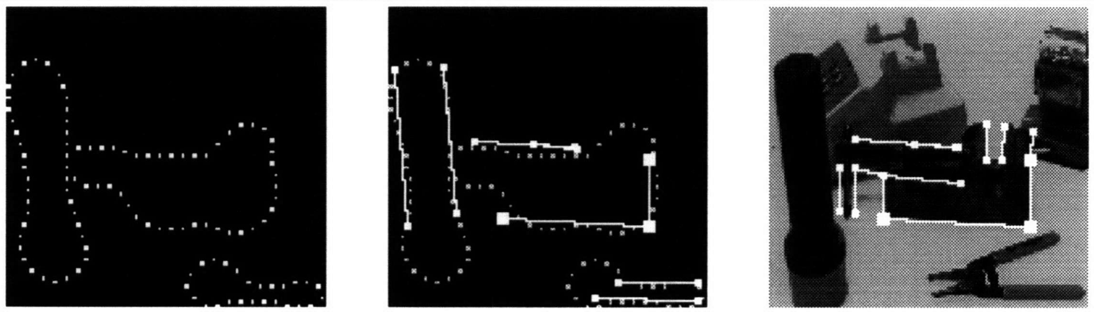


Figure 6-3: Left: Three salient image curves which we suppose are the top three curves obtain by the curve selection step. Middle: Points and line segments which we suppose are detected on the curves. The three larger squares show one triple of alignment points, which correctly contains three point features from the clamp. Right: When the model line segments are projected using this triple of basis points, they will be correctly aligned with the image, since in this example the basis points do not contain error.

Chapter 7

Conclusion

Man-made environments typically contain many objects with well-defined boundary edges. These edges often determine the shapes of the objects and are invaluable in their identification. This thesis was about using such edge information for recognizing three-dimensional objects in images. We represented our objects by metrically accurate geometric models, within a representational approach called "model based."

For model-based recognition systems to be computationally feasible, bottom-up grouping would be useful as a precursor to recognition. To address the problem of grouping in complex images, we have described a method for using smoothness, connectedness, and size to identify objects that are globally salient. This is a specific kind of saliency, nonetheless, and it is not appropriate for all saliency tasks, such as finding a circle in a field of squares. The basic motivation is that a curve through a number of randomly placed, short edge segments is usually not smooth. The method is intended for detecting man-made objects that produce long, smooth edges in images with surface markings, textures, and specularities that produce short, fragmented edges.

In our study of saliency, we first analyzed an existing, noteworthy approach to the problem, Shashua and Ullman's Saliency Network. We showed that the Network has problems with junctions. We also showed that the Network converges according to a geometric series and that this increases the complexity of the saliency measure. Partly for this reason, the saliency measure's rankings of curves are not invariant to scale, and the measure may prefer short open curves next to salient closed curves.

To overcome some of the drawbacks of the Saliency Network, we proposed a new method for saliency which is based on algorithms for finding shortest paths in graphs. Using shortest-paths algorithms, our method identifies the optimal curves in the image according to a saliency measure that prefers contours that are long, smooth, and connected. The experiments demonstrated that, using this saliency measure, the shortest-paths method is capable of finding salient curves in complex images.

For the verification stage of model-based recognition, we considered how to make the verification robust to locational errors in the image features. Error in the image features leads to uncertainty in the projected model features. We showed how error propagates when poses are based on three pairs of model and image points, for both Gaussian and bounded error in the detection of image points, and for both scaled-orthographic and perspective projection models. This result applies to objects that are fully three-dimensional, where past results considered only two-dimensional objects. In addition, we showed how we can utilize *linear programming* to compute the propagated error region for any number of initial matches. Furthermore, we used these results to extend, from two-dimensional to three-dimensional objects, robust implementations of *alignment*, *interpretation-tree search*, and *transformation clustering.* Finally, we gave a new scheme for rapidly evaluating hypotheses in robust alignment that is formally grounded in a model of error.

Appendix A

Bellman-Ford Single-Source Shortest-Paths Algorithm

This appendix describes the Bellman-Ford algorithm, drawing from [15]. We assume we are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a weight (or cost) function $w : \mathcal{E} \to \mathbf{R}$. In the Bellman-Ford algorithm, we first pick a vertex called the "source" and compute the shortest paths (not just the costs of the paths) from the source to every other vertex. Specifically, denote the source by v_0. For every vertex v_i, we store an upper bound on the weight of a shortest path from v_0 to v_i and a pointer to v_i's predecessor vertex along that path. We denote the upper bound by $\widehat{\Psi}(i)$ and the pointer to the predecessor vertex by π_i. Initially, for the source vertex v_0 we set $\widehat{\Psi}(0) = 0$ and $\pi_0 = v_0$, whereas for all other vertices we set $\widehat{\Psi}(i) = \infty$ and $\pi_i = \text{NIL}$. For a graph with V vertices, the Bellman-Ford algorithm performs $V - 1$ iterations, in which each iteration passes over every edge (v_j, v_i) of the graph and performs the operation

$$\begin{aligned} &\textbf{If} \quad \widehat{\Psi}(i) > \widehat{\Psi}(j) + w(v_j, v_i), \\ &\textbf{then} \quad \widehat{\Psi}(i) \leftarrow \widehat{\Psi}(j) + w(v_j, v_i) \quad \textbf{and} \quad \pi_i \leftarrow v_j. \end{aligned} \tag{A.1}$$

Note that the predecessor pointer π_i is updated each time the upper bound $\widehat{\Psi}(i)$ is reduced. At the end of the computation, the π_i values induce a subgraph $\mathcal{G}_\pi$ that is a tree of shortest paths back to the source [24]. Formally, $G_\pi = (\mathcal{V}_\pi, \mathcal{E}_\pi)$, where

$$\mathcal{V}_\pi = \{v_i : \pi_i \neq \text{NIL}\} \bigcup \{v_0\} \text{ and } \mathcal{E}_\pi = \{(\pi_i, v_i) : v_i \in \mathcal{V}_\pi \text{ and } \pi_i \neq \text{NIL}\}.$$

Appendix B

Scaled-Orthographic Similarity Transform

This appendix solves the following equations for $\mathbf{A}$.

$$\overrightarrow{t'_\theta} = \mathbf{A}\,\overrightarrow{t_\theta} \quad \text{and} \quad \overrightarrow{t'_\phi} = \mathbf{A}\,\overrightarrow{t_\phi}\,,$$

Let $x_\theta = \frac{\partial x}{\partial \theta}$, $y_\theta = \frac{\partial y}{\partial \theta}$, and similarly for x_ϕ, y_ϕ, x'_θ, y'_θ, x'_ϕ, and y'_ϕ. Then let $\overrightarrow{t_\theta} = (x_\theta, y_\theta)$, $\overrightarrow{t_\phi} = (x_\phi, y_\phi)$, $\overrightarrow{t'_\theta} = (x'_\theta, y'_\theta)$, $\overrightarrow{t'_\phi} = (x'_\phi, y'_\phi)$. Expanding the first row of each equation gives $x'_\theta = a_{11}x_\theta + a_{12}x_\theta$ and $x'_\phi = a_{11}x_\phi + a_{12}y_\phi$, which implies $[a_{11} \quad a_{12}]^T = \mathbf{T}^{-1}[x'_\theta \quad x'_\phi]^T$, where $\mathbf{T} = \begin{bmatrix} x_\theta & y_\theta \\ x_\phi & y_\phi \end{bmatrix}$. Then from Equations 4.8 and 4.9,

$$\begin{aligned}
\mathbf{T}^{-1} &= \left(\frac{s_0 r}{\cos^2\theta} \begin{bmatrix} -\sin\phi & \sin\theta\cos\phi \\ -\cos\theta\sin\theta\cos\phi & -\cos\theta\sin\phi \end{bmatrix} \right)^{-1} \\
&= \left(\frac{\cos^2\theta}{s_0 r} \right) \frac{1}{\cos\theta(\sin^2\phi + \sin^2\theta\cos^2\phi)} \begin{bmatrix} -\cos\theta\sin\phi & -\sin\theta\cos\phi \\ \cos\theta\sin\theta\cos\phi & -\sin\phi \end{bmatrix} \\
&= \frac{\cos\theta}{s_0 r(1 - \cos^2\theta\cos^2\phi)} \begin{bmatrix} -\cos\theta\sin\phi & -\sin\theta\cos\phi \\ \cos\theta\sin\theta\cos\phi & -\sin\phi \end{bmatrix}.
\end{aligned}$$

Lastly we multiply $\mathbf{T}^{-1}$ by

$$\begin{bmatrix} x'_\theta \\ x'_\phi \end{bmatrix} = \frac{s_0 r'}{\cos^2\theta} \begin{bmatrix} -\sin(\phi+\tau') \\ -\cos\theta\sin\theta\cos(\phi+\tau') \end{bmatrix},$$

to get

$$\begin{aligned}
\begin{bmatrix} a_{11} \\ a_{12} \end{bmatrix} &= \left(\frac{s_0 r'}{\cos^2\theta}\right)\left(\frac{\cos\theta}{s_0 r(1-\cos^2\theta\cos^2\phi)}\right) \\
&\quad \begin{bmatrix} -\cos\theta\sin\phi & -\sin\theta\cos\phi \\ \cos\theta\sin\theta\cos\phi & -\sin\phi \end{bmatrix} \begin{bmatrix} -\sin(\phi+\tau') \\ -\cos\theta\sin\theta\cos(\phi+\tau') \end{bmatrix} \\
&= \frac{r'}{r}\left(\frac{1}{1-\cos^2\theta\cos^2\phi}\right) \begin{bmatrix} \sin\phi\sin(\phi+\tau') + \sin^2\theta\cos\phi\cos(\phi+\tau') \\ -\sin\theta(\cos\phi\sin(\phi+\tau') - \sin\phi\cos(\phi+\tau')) \end{bmatrix} \\
&= \frac{r'}{r}\left(\frac{1}{1-\cos^2\theta\cos^2\phi}\right) \begin{bmatrix} \cos\tau' - \cos^2\theta\cos\phi\cos(\phi+\tau') \\ -\sin\tau'\sin\theta \end{bmatrix} \qquad \text{(B.1)}
\end{aligned}$$

This equation gives $\mathbf{A}$, since $a_{21} = -a_{12}$ and $a_{22} = a_{11}$.

Appendix C

Derivation of the Perspective Linear Transform

Using perspective projection, this appendix derives a linear transform that relates the errors in third and fourth points. First we compute the 3D position of $\vec{m}_2$ in camera coordinates, as given in Equation 4.32. In Fig. C-1 (a copy of Fig. 4-11), let the normal to the plane through $(\vec{c}, \vec{i}_0, \vec{i}_1)$ be

$$\hat{n} = \frac{\vec{c} \times \vec{i}_1}{\| \vec{c} \times \vec{i}_1 \|} = (0, n_y, n_z) \tag{C.1}$$

By Rodriguez' formula,

$$\begin{aligned}
\mathbf{R}_{\{\theta,\hat{n}\}}\vec{p} &= (\cos\theta)\vec{p} + (1-\cos\theta)(\hat{n}\cdot\vec{p})\hat{n} + \sin\theta(\hat{n}\times\vec{p}) \\
&= (d\cos\theta + r\sin\theta(n_y\sin\phi - n_z\cos\phi), \\
&\quad\ r\cos\phi\cos\theta + r(1-\cos\theta)(n_y\cos\phi + n_z\sin\phi)n_y + n_z d\sin\theta, \\
&\quad\ r\sin\phi\cos\theta + r(1-\cos\theta)(n_y\cos\phi + n_z\sin\phi)n_z - n_y d\sin\theta)
\end{aligned} \tag{C.2}$$

To compute the translation $\vec{u}$, let a and b be the (known) distances from $\vec{c}$ to $\vec{i}_0$ and from $\vec{c}$ to $\vec{i}_1$, respectively. From the Law of Cosines,

$$\theta_{01} = \cos^{-1}\left(\frac{d_{01}^2 - a^2 - b^2}{2ab}\right), \qquad \psi = \cos^{-1}\left(\frac{b^2 - a^2 - d_{01}^2}{2ad_{01}}\right), \tag{C.3}$$

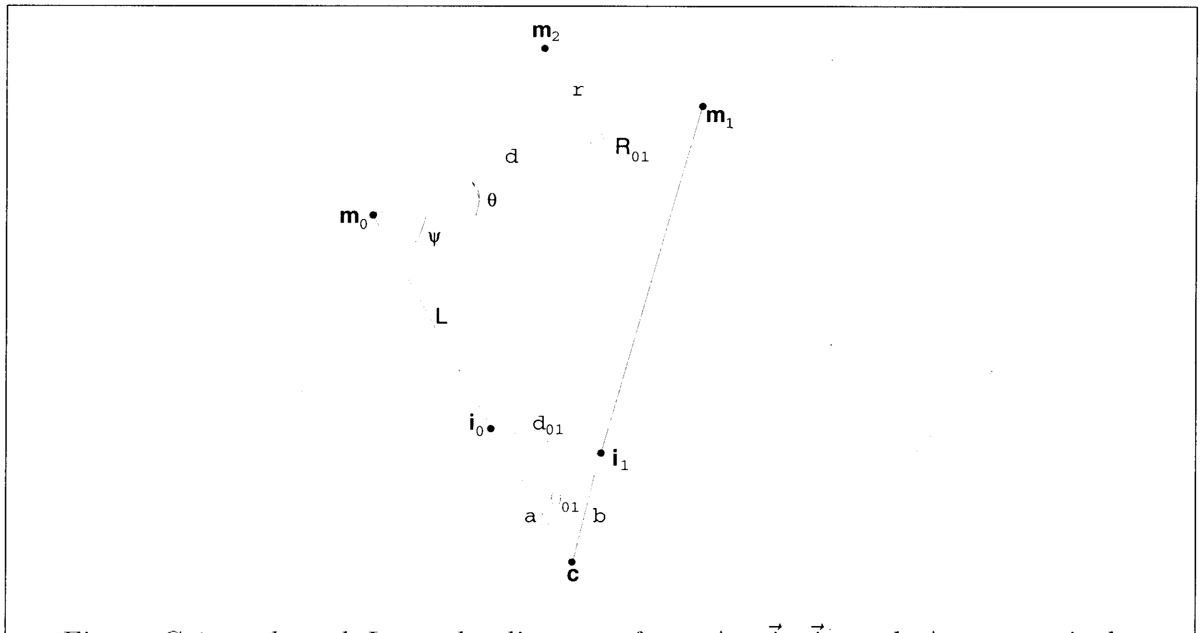


Figure C-1: a, b, and L are the distances from $\vec{c}$ to $\vec{i}_0$, $\vec{i}_1$, and $\vec{m}_0$, respectively.

where $\theta_{01}, \psi \in (0, \pi)$. From the Law of Sines,

$$L \;=\; \sin\left(\pi - (\theta + \psi) - \theta_{01}\right)\left(\frac{R_{01}}{\sin\theta_{01}}\right) \;=\; \sin(\theta + \psi + \theta_{01})\left(\frac{R_{01}}{\sin\theta_{01}}\right)$$

In total, we have

$$(\overline{x}, \overline{y}, \overline{z}) \;=\; \vec{m}_2 \;=\; \vec{c} + L\widehat{v} + \mathbf{R}_{\{\theta,\widehat{n}\}}(d, r\cos\phi, r\sin\phi). \tag{C.4}$$

Substituting $\overline{x}$, $\overline{y}$, and $\overline{z}$ into Equation 4.34 gives $\vec{m}_2^{\pi}$.

Next we take a first approximation to x and y in Equation 4.34 with respect to θ and ϕ. Let $x_\theta = \frac{\partial x}{\partial \theta}$, $y_\theta = \frac{\partial y}{\partial \theta}$, and similarly for x_ϕ, y_ϕ, $\overline{x}_\theta$, $\overline{y}_\theta$, $\overline{y}_\phi$, $\overline{y}_\phi$, and L_θ. From Equation 4.34,

$$x_\theta \;=\; f\left(\frac{\overline{x}_\theta}{\overline{z} + f} - \frac{\overline{z}_\theta(\overline{x} - c_x)}{(\overline{z} + f)^2}\right), \qquad y_\theta \;=\; f\left(\frac{\overline{y}_\theta}{\overline{z} + f} - \frac{\overline{z}_\theta(\overline{y} - c_y)}{(\overline{z} + f)^2}\right). \tag{C.5}$$

For x_ϕ and y_ϕ, we substitute ϕ for θ in these equations. Using Equations C.2-C.4,

$$\overline{x}_\theta \;=\; -d\sin\theta + r\cos\theta(n_y\sin\phi - n_z\cos\phi) + L_\theta v_x \tag{C.6}$$

$$\overline{y}_\theta \;=\; -r\cos\phi\sin\theta + r\sin\theta(n_y\cos\phi + n_z\sin\phi)n_y + n_z d\cos\theta + L_\theta v_y \tag{C.7}$$

$$\overline{z}_\theta = -r\sin\phi\sin\theta + r\sin\theta(n_y\cos\phi + n_z\sin\phi)n_z - n_y d\cos\theta + L_\theta v_z \quad (C.8)$$

$$L_\theta = \cos(\theta + \psi + \theta_{01})\left(\frac{R_{01}}{\sin\theta_{01}}\right) \quad (C.9)$$

$$\overline{x}_\phi = r\sin\theta(n_y\cos\phi + n_z\sin\phi) \quad (C.10)$$

$$\overline{y}_\phi = -r\sin\phi\cos\theta + r(1-\cos\theta)(-n_y\sin\phi + n_z\cos\phi)n_y \quad (C.11)$$

$$\overline{z}_\phi = r\cos\phi\cos\theta + r(1-\cos\theta)(-n_y\sin\phi + n_z\cos\phi)n_z \quad (C.12)$$

The above equations give $\overrightarrow{t_\theta}$ and $\overrightarrow{t_\phi}$. By substituting r' for r, d' for d, and $\phi + \tau'$ for ϕ, we get $\overrightarrow{t'_\theta}$ and $\overrightarrow{t'_\phi}$. Solving Equation 4.21 leads to

$$\mathbf{A} = \frac{1}{x_\theta y_\phi - x_\phi y_\theta}\begin{bmatrix} y_\phi x'_\theta - y_\theta x'_\phi & -x_\phi x'_\theta + x_\theta x'_\phi \\ y_\phi y'_\theta - y_\theta y'_\phi & -x_\phi y'_\theta + x_\theta y'_\phi \end{bmatrix} \quad (C.13)$$

Appendix D

Gaussian Error Propagation

For this appendix, we adopt Therrian's notation [87]. In general, let $\vec{x}$ be a Gaussian random vector and let $\vec{y} = \mathbf{M}\vec{x} + \vec{b}$, where $\mathbf{M}$ is $n \times m$. For random vectors $\vec{x}$ and $\vec{y}$, respectively, denote their expected values by $\vec{m}_x$ and $\vec{m}_y$ and their covariance matrices by $\mathbf{K_x}$ and $\mathbf{K_y}$. Then

$$p_{\vec{y}}\left(\widehat{\vec{y}}\right) = \frac{1}{(2\pi)^{\frac{n}{2}} \left|\mathbf{K_y}\right|^{\frac{1}{2}}} e^{\left(-\frac{1}{2}(\widehat{\vec{y}}-\vec{m}_y)^{\mathrm{T}}\mathbf{K_y}^{-1}(\widehat{\vec{y}}-\vec{m}_y)\right)},$$

where $\vec{m}_y = \mathbf{M}\vec{m}_x + \vec{b}$ and $\mathbf{K_y} = \mathbf{M}\mathbf{K_x}\mathbf{M}^{\mathrm{T}}$ [87]. In our case, we have four two-dimensional Gaussian distributions, corresponding to the errors in three matched image points and one unmatched image point. This gives eight uncorrelated Gaussian random variables, of which we are taking a linear combination. When $\vec{\imath}_0$, $\vec{\imath}_1$, $\vec{\imath}_2$, and $\vec{\imath}_3$ are normally distributed with standard deviations σ_0, σ_1, σ_2, and σ_3, respectively, $\mathbf{K_x}$ is a diagonal matrix with on-diagonal elements $(\sigma_0^2, \sigma_0^2, \sigma_1^2, \sigma_1^2, \sigma_2^2, \sigma_2^2, \sigma_3^2, \sigma_3^2)$. Further, the linear combination in Equation 4.25 is given by

$$\mathbf{M} = \begin{bmatrix} a_{11} & a_{12} & b_{11} & b_{12} & c_{11} & c_{12} & 1 & 0 \\ a_{21} & a_{22} & b_{21} & b_{22} & c_{21} & c_{22} & 0 & 1 \end{bmatrix}, \tag{D.1}$$

where $n = 2$ and $m = 8$. Expanding $\mathbf{M}\mathbf{K}_{\mathbf{x}}\mathbf{M}^{\mathrm{T}}$ leads to

$$\begin{aligned}\mathbf{K}_{\mathbf{y}} &= \sigma_0^2 \begin{bmatrix} a_{11}^2 + a_{12}^2 & a_{11}a_{21} + a_{12}a_{22} \\ a_{11}a_{21} + a_{12}a_{22} & a_{21}^2 + a_{22}^2 \end{bmatrix} + \sigma_1^2 \begin{bmatrix} b_{11}^2 + b_{12}^2 & b_{11}b_{21} + b_{12}b_{22} \\ b_{11}b_{21} + b_{12}b_{22} & b_{21}^2 + b_{22}^2 \end{bmatrix} \\ &+\sigma_2^2 \begin{bmatrix} c_{11}^2 + c_{12}^2 & c_{11}c_{21} + c_{12}c_{22} \\ c_{11}c_{21} + c_{12}c_{22} & c_{21}^2 + c_{22}^2 \end{bmatrix} + \sigma_3^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (D.2)\end{aligned}$$

Under weak-perspective projection, $a_{21} = -a_{12}$, $a_{22} = a_{11}$, $b_{21} = -b_{12}$, $b_{22} = b_{11}$, $c_{21} = -c_{12}$, and $c_{22} = c_{11}$. Letting $S_0 = \sqrt{a_{11}^2 + a_{12}^2}$, $S_1 = \sqrt{b_{11}^2 + b_{12}^2}$, and $S_2 = \sqrt{c_{11}^2 + c_{12}^2}$, the expression for $\mathbf{K}_{\mathbf{y}}$ simplifies to

$$\mathbf{K}_{\mathbf{y}} = \left(S_0^2\sigma_0^2 + S_1^2\sigma_1^2 + S_2^2\sigma_2^2 + \sigma_3^2\right) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (D.3)$$

Then

$$p_{\vec{y}}\left(\widehat{\vec{y}}\right) = \frac{1}{2\pi\sigma} e^{-\frac{\|\widehat{\vec{y}} - \vec{m}_y\|^2}{2\sigma}},$$

where

$$\sigma = |\mathbf{K}_{\mathbf{y}}|^{\frac{1}{2}} = \sqrt{S_0^2\sigma_0^2 + S_1^2\sigma_1^2 + S_2^2\sigma_2^2 + \sigma_3^2} \qquad (D.4)$$

Appendix E

Line Selectivity Formula

Given a model line segment, [3] derives the following formula for its selectivity, μ. Let r and R be the radii of the two uncertainty circles for the endpoints of the line segment, such that $r \leq R$ (Fig. E-1). r and R can be computed using the technique given in Chapter 4. Next, let L be the distance between the centers of the two circles, and let ℓ be the expected length of a random line segment in the image. Define

$$\begin{aligned}
v_1(\omega_1,\omega_2) &= (R+r-\ell)2r(\omega_2-\omega_1) + 2rL(\sin\omega_2 - \sin\omega_1) \\
v_2(\omega_1,\omega_2) &= (R+r-\ell)(R+r)(\omega_2-\omega_1) + (R+r-\ell)L(\cos\omega_2 - \cos\omega_1) \\
&\quad +(R+r)L(\sin\omega_2 - \sin\omega_1) - \frac{1}{2}L^2(\sin^2\omega_2 - \sin^2\omega_1) \\
v_3(\omega_1,\omega_2) &= (2R-\ell)2r(\omega_2-\omega_1) \\
v_4(\omega_1,\omega_2) &= (2R-\ell)(R+r)(\omega_2-\omega_1) + (2R-\ell)L(\cos\omega_2 - \cos\omega_1)
\end{aligned}$$

If $R+r \leq L$, let

$$\theta_1 = \sin^{-1}\frac{R-r}{L}, \quad \theta_2 = \sin^{-1}\frac{R+r}{L}, \quad \phi = \cos^{-1}\left(\frac{\ell-(R+r)}{L}\right).$$

Otherwise, if $R-r \leq L \leq R-r$, let

$$\theta_1 = \sin^{-1}\frac{R-r}{L}, \quad \theta_1' = \cos^{-1}\frac{R-r}{L}, \quad \phi = \begin{cases} \cos^{-1}\left(\frac{\ell-(R+r)}{L}\right), & \text{if } \ell \geq R+r-L, \\ \pi/2 & \text{otherwise.} \end{cases}$$

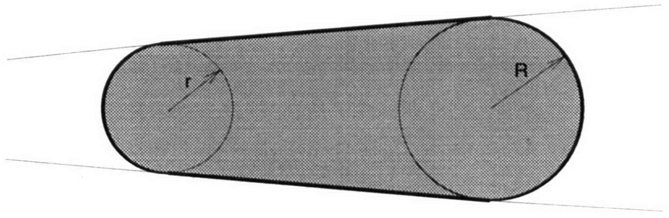


Figure E-1: The uncertainty region for a line segment is constructed from the uncertainty regions of the segment's endpoints.

Next, if $R + r \leq L$, let

$$V = 2 \begin{cases} v_1(0,\phi) & \text{if } \phi \leq \theta_1, & l \leq R+r+L, \\ v_1(0,\theta_1) + v_2(\theta_1,\phi) & \text{if } \theta_1 \leq \phi \leq \theta_2, & l \leq R+r+L, \\ v_1(0,\theta_1) + v_2(\theta_1,\theta_2) & \text{if } \theta_2 \leq \phi, & l \leq R+r+L, \\ 0 & \text{otherwise.} \end{cases}$$

Otherwise if $R - r \leq L \leq R + r$, let

$$V = 2 \begin{cases} v_1(0,\phi) & \text{if } \phi \leq \theta_1, \theta_1', & l \leq R+r+L, \\ v_1(0,\theta_1) + v_2(\theta_1,\phi) & \text{if } \theta_1 \leq \phi \leq \theta_1', & l \leq R+r+L, \\ v_1(0,\theta_1) + v_2(\theta_1,\theta_1') + v_4(\theta_1',\pi/2) & \text{if } \theta_1 \leq \theta_1' \leq \phi, & l \leq R+r+L, \\ v_1(0,\theta_1') + v_3(\theta_1',\pi/2) & \text{if } \theta_1' \leq \phi \leq \theta_1, & l \leq R+r+L, \\ v_1(0,\theta_1') + v_3(\theta_1',\theta_1) + v_4(\theta_1,\pi/2) & \text{if } \theta_1' \leq \theta_1 \leq \phi, & l \leq R+r+L, \\ 0 & \text{otherwise.} \end{cases}$$

Otherwise if $L \leq R - r$, let

$$V = \begin{cases} 2\pi r(2R - \ell) & \text{if } \ell \leq 2R, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, let w and h be the width and height of the image, and let

$$V_I = \pi wh - 2\ell(w+h) + \ell^2.$$

The line selectivity is

$$\mu = \frac{V}{V_I}.$$

Bibliography

[1] Ahlfors, L. V., *Complex Analysis*, 2nd ed., New York: McGraw-Hill, Inc., 1966.

[2] Alter, T. D., "3-D Pose from 3 Points Using Weak-Perspective," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **16**(8):802-808, 1994.

[3] Alter, T. D., "Robust and Efficient 3D Recognition by Alignment," MIT Artif. Intell. Lab. Tech. Report 1410, 1992.

[4] Alter, T. D., and W. E. L. Grimson, "Fast and Robust 3D Recognition by Alignment," *Proc. Fourth Inter. Conf. on Computer Vision*, Berlin, Germany, May 1993.

[5] Ayache, N., and O. Faugeras, "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **8**(1):44-54, 1986.

[6] Amini, A. A., T. E. Weymouth, and R. C. Jain, "Using Dynamic Programming for Solving Variational Problems in Vision," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **12**(9):855-867, 1990.

[7] Baird, H., *Model-Based Image Matching Using Location*, MIT Press, Cambridge, 1985.

[8] Ballard, D. H., "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.

[9] Ben-arie, J., "The Probabilistic Peaking Effect of Viewed Angles and Distances with Application to 3-D Object Recognition," *IEEE Trans. Pat. Anal. Machine Intell.*, vol. 12, no. 8, August 1990.

[10] Beveridge, R., R. Weiss, and E. Riseman, "Combinatorial Optimization Applied to Variable Scale 2D Model Matching," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 18-23, 1990.

[11] Biederman, I., "Human Image Understanding: Recent Research and a Theory," *Computer Vision, Graphics, and Image Proc.*, vol. 32, pp. 29-73, 1985.

[12] Blake, A., and A. Zisserman, *Visual Reconstruction*, Cambridge: The MIT Press, 1987.

[13] Bolles, R. C., and R. A. Cain, "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method," *Inter. Journal of Robotics Research*, **1**(3):57-82, 1982.

[14] Bolles, R. C., L. H. Quam, M. A. Fischler, and H. C. Wolf, "The SRI Road Expert: Image-to-Database Correspondence," *Proc. DARPA IU Workshop*, pp. 163-174, 1978.

[15] Brady, M., W. E. L. Grimson, and D. J. Langridge, "Shape Encoding and Subjective Contours," *Proc. First Annual Conf. Artif. Int.*, 1980.

[16] Brady, M., and W. E. L. Grimson, "The Perception of Subjective Surfaces," *MIT A. I. Memo No. 666*, November 1981.

[17] Breuel, T., "Model Based Recognition using Pruned Correspondence Search," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 257-268, 1991.

[18] Brooks, R. A., "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intell.*, **17**:285-348, 1981.

[19] Bruckstein, A. M., and A. N. Netravali, "On Minimal Energy Trajectories," *Computer Vision, Graphics, and Image Processing*, 49, pp. 283-296, 1990.

[20] Burns, J. B., A. R. Hanson, and E. M. Riseman, "Extracting Straight Lines," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 4, pp. 424-455, July 1986.

[21] Cass, T., "Polynomial Time Object Recognition in the Presence of Clutter, Occlusion and Uncertainty," *Second European Conf. on Computer Vision*, pp. 834-842, 1992.

[22] Clark, C. S., W. O. Eckhardt, C. A. McNary, R. Nevatia, K. E. Olin, and E. M. VanOrden, "High-accuracy Model Matching for Scenes Containing Man-Made Structures," *Proc. Symp. Digital Processing of Aerial Images*, SPIE, vol. 186, pp. 54-62, 1979.

[23] Clemens, D., and D. W. Jacobs, "Space and Time Bounds on Model Indexing," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **13**(10):1007-1018, 1991.

[24] Cormen, C. H., C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, 1990.

[25] Costa, M., R. M. Haralick, and L. G. Shapiro, "Optimal Affine-Invariant Point Matching," *Proc. Sixth Israeli Conf. on Artif. Intell.*, pp. 35-61, 1990.

[26] Cox, I. J., J. M. Rehg, and S. Hingorani, "A Bayesian Multiple-Hypothesis Approach to Edge Grouping and Contour Segmentation," *Inter. Journal of Computer Vision*, **11**(1):6-24, 1993.

[27] Cyganski, D., and J. Orr, "Applications of Tensor Theory to Object Recognition and Orientation Determination," *IEEE Trans. Pat. Anal. Machine Intell.*, vol. 7, no. 6, November 1985.

[28] Elder, J., and S. Zucker, "The Integration of Figure Fragments into Representations of Planar Shape," McGill Research Centre for Intelligent Machines TR-93-2, February 1993.

[29] Ellis, R. E., "Uncertainty Estimates for Polyhedral Object Recognition," *Proc. IEEE Conf. Rob. Aut.*, pp. 348-353, 1989.

[30] Finkel, L. H., and P. Sajda, "Object Discrimination Based on Depth-from-Occlusion," *Neural Computation*, 4, pp. 901-921, 1992.

[31] Fischler, M. A., and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Analysis and Automated Cartography," *Communications of the Association of Computing Machinery*, **24**(6):381-395, 1981.

[32] Goad, C. A., "Special Purpose Automatic Programming for 3D Model-Based Vision," *Proc. DARPA Image Understanding Workshop*, pp. 94-104, 1983.

[33] Grimson, W. E. L., "The Combinatorics of Heuristic Search Termination for Object Recognition in Cluttered Environments," *IEEE Trans. Pat. Anal. Machine Intell.*, vol. 13, no. 9, Sept. 1991.

[34] Grimson, W. E. L., and D. P. Huttenlocher, "On the Sensitivity of the Hough Transform for Object Recognition," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **12**(3), March 1990.

[35] Grimson, W. E. L., and D. P. Huttenlocher, "On the Sensitivity of Geometric Hashing," *Proc. Third Inter. Conf. Computer Vision*, pp. 334-338, 1990.

[36] Grimson, W. E. L., and D. P. Huttenlocher, "On the Verification of Hypothesized Matches in Model-Based Recognition," *IEEE Trans. Pat. Anal. Machine Intell.*, vol. 13, no. 12, December 1991.

[37] Grimson, W. E. L., D. P. Huttenlocher, and T. D. Alter, "Recognizing 3D Objects from 2D Images: An Error Analysis," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1992.

[38] Grimson, W. E. L., D. P. Huttenlocher, and D. W. Jacobs, "A Study of Affine Matching with Bounded Sensor Error," *Proc. Second European Conf. on Computer Vision*, pp. 291-306, 1992.

[39] Grimson, W. E. L., and T. Lozano-Pérez, "Model-Based Recognition and Localization from Sparse Range or Tactile Data," *Int. J. Robotics Res.*, **3**(3):3-35, 1984.

[40] Grimson, W. E. L., and T. Lozano-Pérez, "Localizing Overlapping Parts by Searching the Interpretation Tree," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **9**(4):469-482, 1987.

[41] Grimson, W. E. L., T. Lozano-Pérez, N. Noble, and S. J. White, "An Automatic Tube Inspection System That Finds Cylinders in Range Data," in *Proc. IEEE Conf. Computer Vision Pat. Rec.*, 1993.

[42] Grossberg, S., and M. Mingolla, "Neural Dynamics of Surface Perception: Boundary Webs, Illuminants, and Shape-from-Shading," *Computer Vision, Graphics, and Image Processing*, 37, pp. 116-165, 1987.

[43] Guy, G., and G. Medioni, "Inferring Global Perceptual Contours from Local Features," USC report.

[44] Heitger, F., and Rüdiger von der Heydt, "A Computational Model of Neural Contour Processing: Figure-Ground Segregation and Illusory Contours," *Proc. of ICCV*, pp. 32-40, 1993.

[45] Hel-Or, Y., and M. Werman, "Absolute Orientation from Uncertain Point Data: A Unified Approach," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 77-82, 1991.

[46] Horaud, R., "New Methods for Matching 3-D Objects with Single Perspective Views," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **9**(3):401-412, May 1987.

[47] Horaud, R., F. Veillon, and T. Skordas, "Finding Geometric and Relational Structures in an Image," *Proc. First European Conf. Computer Vision*, pp. 374-384, April 1990.

[48] Horn, B. K. P., "The Curve of Least Energy," *ACM Trans. on Math. Soft.*, vol. 9, no. 4, pp. 441-460, December 1983.

[49] Herault, L., and R. Horaud, "Figure-Ground Discrimination: A Combinatorial Optimization Approach," *IEEE Trans. on Pattern Analysis and Artificial Intelligence*, 15(9), 1993.

[50] Hu, J., B. Sakoda, and T. Pavlidis, "Interactive Road Finding for Aerial Images," *IEEE Workshop on Applications on Computer Vision*, Palm Springs, CA, pp. 56-63, 1992.

[51] Huttenlocher, D., "Three-Dimensional Recognition of Solid Objects from a Two-Dimensional Image," MIT Artif. Intell. Lab. Tech. Report 1045, 1988.

[52] Huttenlocher, D., and S. Ullman, "Recognizing Solid Objects by Alignment with an Image," *Inter. Journal of Computer Vision*, **5**(2):195-212, 1990.

[53] Huttenlocher, D. P., and P. Wayner, "Finding Convex Edge Groupings in an Image," *Inter. J. Computer Vision*, vol. 5, no. 2, pp. 195-212, 1990.

[54] Jacobs, D. W., "Optimal Matching of Planar Models in 3D Scenes," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 269-274, 1991.

[55] Jacobs, D. W., "Recognizing 3-D Objects Using 2-D Images," MIT Artif. Intell. Lab. Tech. Report 1416, 1993.

[56] Kass, M., A. Witkin, and D. Terzopolous, "Snakes: Active Contour Models," *Inter. J. Computer Vision*, vol. 1, no. 4, 1988.

[57] Klein, P., S. Rao, M. Rauch, and S. Subramanian, "Faster shortest-path algorithms for planar graphs," *Proc. Symp. the Theory of Computing*, 1994, and personal communication.

[58] Kumar, R., and A. Hanson, "Robust Estimation of Camera Location and Orientation from Noisy Data having Outliers," *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pp. 52-60, 1989.

[59] Lamdan, Y., J. T. Schwartz, and H. J. Wolfson, "Object Recognition by Affine Invariant Matching" *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 335-344, 1988.

[60] Lamdan, Y., J. T. Schwartz, and H. J. Wolfson, "Affine Invariant Model-Based Object Recognition," *IEEE Transactions Robotics and Automation*, **6**:578-589, 1990.

[61] Lamdan, Y., and H. J. Wolfson, "Geometric Hashing: A General and Efficient Model-Based Recognition Scheme," *Proc. Second Inter. Conf. on Computer Vision*, Tampa, Florida, pp. 238-249, December 1988.

[62] Lamdan, Y., and H. J. Wolfson, "On the Error Analysis of 'Geometric Hashing'," *IEEE Conf. Computer Vision and Pattern Recognition,* pp. 22-27, 1991.

[63] Linnainmaa, S., D. Harwood, and L. S. Davis, "Pose Determination of a Three-Dimensional Object Using Triangle Pairs," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 5, pp. 634-647, Sept. 1988.

[64] Lowe, D., *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, The Netherlands, 1985.

[65] Mahoney, J. V., "Image Chunking: Defining Spatial Building Blocks for Scene Analysis," MIT Artif. Intell. Lab. Tech. Report 980, 1987.

[66] Martelli, A., "An Application of Heuristic Search Methods to Edge and Contour Detection," *Comm. ACM*, 19(2), February 1976.

[67] Mohan, R., and R. Nevatia, "Perceptual Grouping for the Detection and Description of Structures in Aerial Images," *Proc. DARPA Image Understanding Workshop*, pp. 512-526, 1988.

[68] Mohan, R., and R. Nevatia, "Perceptual Grouping for Scene Description and Segmentation," *Trans. on Pattern Analysis and Machine Intelligence*, 14(6), 1992.

[69] Montanari, U., "On the Optimal Detection of Curves in Noisy Pictures," *Comm. ACM*, 14(5), May 1971.

[70] Olson, C. F., "Fast Alignment Using Probabilistic Indexing," in *Proc. IEEE Conf. Computer Vision Pat. Rec.*, pp. 387-392, June 1993.

[71] Oshima, M., and Y. Shirai, "Object Recognition Using Three-Dimensional Information," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 5, no. 4, pp. 353-361, July 1983.

[72] Parent, P., and S. W. Zucker, "Trace Inference, Curvature Consistency, and Curve Detection," *Trans. on Pattern Analysis and Machine Intelligence*, 11(8), 1989.

[73] Pavlidis, T., and Y.-T. Liow, "Integrating Region Growing and Edge Detection," *Trans. on Pattern Analysis and Machine Intelligence*, 12(3), pp. 225-233, March 1990.

[74] Reeves, A. P., and R. W. Taylor, "Identification of Three-Dimensional Objects Using Range Information," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, no. 4, pp. 403-410, April 1989.

[75] Rigoutsos, I., "Massively Parallel Bayesian Object Recognition," Ph. D. Thesis, New York University Department of Computer Science, 1992.

[76] Rigoutsos, I., and R. Hummel, "Robust Similarity Invariant Matching in the Presence of Noise," *Eighth Israeli Conf. on Artif. Intell. Computer Vision*, Tel Aviv, 1991.

[77] Roberts, L., "Machine Perception of Three-Dimensional Solid Objects," *Optical and Electro-optical Information Processing*, edited by J. Tippett, MIT Press, Cambridge, 1966.

[78] Rothwell C., A. Zisserman, J. Mundy, and D. Forsyth, "Efficient Model Library Access by Projectively Invariant Indexing Functions," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 109-114, 1992.

[79] Rutkowski, W. S., "Shape Completion," *Computer Vision, Graphics, and Image Proc.*, vol. 9, pp. 89-101, 1979.

[80] Sarachik, K. B., and W. E. L. Grimson, "Gaussian Error Models for Object Recognition," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 400-406, June 1993.

[81] Seidel, R., "Linear Programming and Convex Hulls Made Easy," *Proc. of the Sixth Annual Symp. on Computational Geometry*, pp. 211-215, 1990.

[82] Sha'ashua, A., and S. Ullman, "Structural Saliency: The Detection of Globally Salient Structures Using a Locally Connected Network," in *2nd ICCV*, pp. 321-327, December 1988, also Weizmann Institute of Science Report CS88-18, October 1988.

[83] Shashua, A., and S. Ullman, "Grouping Contours by Iterated Pairing Network," *NIPS 3*, December 1990.

[84] Strang, G., *Linear Algebra and its Applications*, Harcourt, Brace, Jovanavich, Publishers, 1988.

[85] Subirana-Vilanova, J., "Curved Inertia Frames: Visual Attention and Perceptual Organization Using Convexity and Symmetry," *MIT A. I. Memo No. 1137*, Dec. 1991.

[86] Subirana-Vilanova, J., and K. K. Sung, "Multi-Scale Vector-Ridge-Detection for Perceptual Organization Without Edges," *MIT A. I. Memo No. 1318*, Dec. 1992.

[87] Therrien, C. H., *Decision, Estimation, and Classification*, Wiley & Sons, 1989.

[88] Thompson, D., and J. L. Mundy, "Three-Dimensional Model Matching From an Unconstrained Viewpoint", *Proc. IEEE Conf. Rob. Aut.*, pp. 208-220, 1987.

[89] Turney, J. L., T. N. Mudge, and R. A. Voltz, "Recognizing Partially Occluded Parts," *IEEE Trans. Pat. Anal. Machine Intell.*, vol. 7, no. 4, July 1985.

[90] Ullman, S., "Filling-In the Gaps: The Shape of Subjective Contours and a Model for Their Generation," *Biological Cybernetics*, 25, pp. 1-6.

[91] Ullman, S., and R. Basri, "Recognition by Linear Combinations of Models," *IEEE Transactions on Pattern Anal. and Machine Intell.*, **13**(10):992-1007, 1991.

[92] Wayner, P. C., "Efficiently Using Invariant Theory for Model-based Matching," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 473-478, 1991.

[93] Webb, J. A., and E. Pervin, "The Shape of Subjective Contours," *Proc. National Conf. on Artificial Intelligence*, pp. 340–343, Aug. 1984.

[94] Weinshall, D., and R. Basri, "Distance Metric Between 3-D Models and 2-D Images for Recognition and Classification," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 220-225, 1993.

[95] Weiss, I., "3D Shape Representation by Contours," *CVGIP*, 41, pp. 80-100, 1988.

[96] Wells, W., "MAP Model Matching," *IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 486-492, 1991.

[97] Wells, W., "Statistical Object Recognition," MIT Artif. Intell. Lab. Tech. Report 1398, 1993.

[98] Williams, L. R., "Perceptual Organization of Occluding Contours," *Proc. Image Understanding Workshop*, pp. 639-649, Sept. 1990.

[99] Williams, L. R., and D. W. Jacobs, "Stochastic Completion Fields: A Neural Model of Illusory Contour Shape and Salience," to appear in *Proc. Fifth Inter. Conf. on Computer Vision*, June 1995.