A Model-Based 3-D Object Recognition System using Geometric Hashing with Attributed Features

Jyh-Jong Liu

A Dissertation Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Department of Computer Science

New York University

January, 1996

Approved: _____

Professor Robert Hummel

Research Advisor

To My Family

Acknowledgments

I would like to express my deepest gratitude to my advisor Professor Robert Hummel for his guidance, support, and encouragement over these years. He taught me the spirit of attacking problems. Most of my Ph.D. research could not have been fulfilled without his ideas and inspiration.

My special thanks are to Professor Richard Wallace and Professor Stéphane Mallat for being in my Oral Committee. They have helped me throughout my graduate school years. I would like to express my gratitude to Dr. Isidore Rigoutsos. This thesis work is built upon and benefited by his work. I also thank Dr. Davi Geiger for generously discussing topics with me. Many thanks go to Dr. Xiaonan Tan and Professor Jiawei Hong. Without them, I would not be able to go through the tough time during my study. I am also grateful to all of the friends I met in NYU. Their companionship made the stay in New York City a pleasant experience. In particular, my thanks are due to Dr. Chee-Da Tsai, Dr. Jen-Lung Chiu and Mr. Yaw-Tai Lee for their helpful discussions. We shared memorable times together.

Finally, I would like to express my deeply gratitude to my family. I am indebted to my sister and brother for their constant encouragement, as well as my parents for their love and persistent support. They always have confidence in me, and always accompany me to go through tough moment even though they are on the other side of the earth.

Abstract

We build an object recognition system that is able to recognize 3-D objects such as vehicles embedded in highly complicated backgrounds. We use the *geometric hashing* method, augmenting the approach through the use of *attributed features*, k-d trees for access to features, and the use of bounds in order to limit the search.

We make use of expressive features to improve the performance of a geometric hashing object recognition system. Various kinds of attributed features, such as the midpoint of a line segment with its orientation, the endpoints of a line segment with its orientation, and the center and the circle features are extracted and used in our system.

The number of features as well as the type of features in each model can vary. We make use of weighted voting, which has a Bayesian interpretation. The distribution of the invariants for various features as well as the bounds of the weighted voting formula are analyzed. In order to improve the performance of the system, we use a *k-d tree* to search entries in high-dimensional hash tables. The method is generalized in order to treat variables taking on values from a non-interval domain, such as data measuring angles. To make use of available computer resources, we distribute the computation, assigning evidence accumulation for a single hypothesis to one processor in a multiple processor and multiple workstation environment. The implementation reduces the communication overhead to minimum. The system is implemented using the *Khoros* software development system.

The results of target recognition are reported in numerous experiments. The experiments show that the use of more expressive features improves the performance of the recognition system.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the major goals for computer vision researchers is to build a vision system that can recognize 3-D objects and interpret a 3-D scene from 2-D images. The system should be robust enough so that it can work for real data. On the other hand, the method should be efficient so that even though the problem is ill-formulated, we are still able to get a reasonable answer within a reasonable time.

The 3-D recognition problem can be stated as follows: Given prior knowledge of 3-D objects, we want to determine the occurrence of objects from the given imagery data. In addition, the position of the objects in the image should be determined.

This research is an attempt to build a 3-D object recognition system that is able to recognize 3-D objects such as vehicles embedded in a highly complicated background. The method that we use is called *geometric hashing*. We give a brief review of the geometric hashing method in the next section. A complete description of the modern form for geometric hashing method can be found in the paper written by Lamdan and Wolfson [64]. Our contributions to the development of geometric hashing centers on *attributed features*. Further, to improve the performance of the system, we use a *k-d tree* to search entries in high-dimensional hash tables. To make use of available computer resources, we distribute the computation into pieces. We will address these issues in Chapter 4 and Chapter 5 respectively.

1.1 A Brief Review of Geometric Hashing

The idea of geometric hashing is to use invariants to index from an extracted scene into a prestored hash table in order to discover the possible candidate matches. The method is an efficient technique that uses spatial arrangements of features to locate instances of models. Because this method does not match models one by one, it is capable of recognizing objects effectively with very large databases. The invariants are the result of normalization which

usually depend upon the choice of a basis set. Thus the basis information as well as the model index are recorded in the hash space in the preprocessing stage. A voting process is involved to recover the transformation between the object in the scene and the object in the model database during the recognition stage.

The original idea of geometric hashing comes from the research work of matching boundary curves [54]. The research done by Schwartz and Sharir [99], Wolfson [119], Hong and Wolfson [49], all rely on the technique of geometric hashing. They develop the technique of finding invariants for boundary curves that are called *footprints*. Hong and Wolfson [49] mention the idea of weighted footprints in order to improve the robustness of the system. Guéziec and Ayache [42, 43] use differential invariants and the geometric hashing scheme for 3-D curve matching to register 3-D medical images.

Lamdan and Wolfson give a description of the geometric hashing method [64]. Early prototype systems for recognizing flat industrial parts and synthesized 3-D objects are reported by Lamdan *et al.* [62, 61, 60, 63]. The features are called "interest points." That is, the geometric hashing method performs point pattern matching in these experiments. They also analyze the error of geometric hashing by computing the probability of false matches. Their conclusion is that geometric hashing should be viewed as a "filtering" procedure which can eliminate a lot of candidate false solutions before direct verification is applied [65]. Gavrila and Groen use a geometric hashing system to recognize 3-D CAD models [37].

A parallel implementation of geometric hashing on the Connection Machine is reported by Rigoutsos and Hummel [87, 90], and also one by Khokhar and Prasanna [56]. Rigoutsos and Hummel also report a distributed version of geometric hashing for object recognition [91].

Rigoutsos and Hummel [88, 89] assume the appearance of Gaussian noise for the position of point pattern and derive analytic solutions for the features in hash space. A precise weighted voting formula with a Bayesian interpretation for geometric hashing is given. Tsai [108] analyzes the affine invariants for line features, where line features are represented as a point in (θ, r) space.

Grimson and Huttenlocher [39] analyze the performance of geometric hashing by assuming that the noise model of the feature points is an ϵ-disc. Lamdan and Wolfson derive the false alarm rate empirically and analytically. Their analysis is performed on (r, θ) space with a bounded error model. Sarachik and Grimson [98, 97] investigate the performance of geometric hashing with the assumption of a Gaussian noise model. With a Gaussianly-distributed noise model, they obtain predictions of operating characteristics of simple recognition systems, which show acceptable performance under low-noise conditions.

Califano and Mohan [14, 15] use higher-order features to improve the performance as

well as the fault tolerance of the recognition system. Liu and Hummel [69] also adopt the strategy of using higher order features, i.e., features are attributed with extra information. The discrimination power of attributed features is such that a 3-D object embedded in a complicated background can still be recognized.

This thesis is based on the previous work done by Rigoutsos and Hummel [86]. Several ideas are introduced in order to make this object recognition system more robust, more efficient, more general, and more adequate for the task of automatic target recognition. Specifically, the concept of *attributed features*, i.e., features endowed with additional information that can be used to filter the matches, is introduced so that an object embedded in a highly complicated background can be recognized. The generalization to the case when each model has different number of features makes the object recognition system more realistic. The use of a k-d tree data structure makes the search in a high dimensional hash table efficient. We also propose a method of partitioning the hash table so that the computation can be performed in a distributed computer system. Our implementation is based upon the Khoros [57] software development environment, which is suitable for information processing and visualization.

1.2 Outline of Thesis

The rest of this thesis is organized as follows. We review the recent development of model-based object recognition systems in Chapter 2. Chapter 3 discusses the formulation of current geometric hashing. Chapter 4 describes our use of attributed features. Implementation issues for efficient and distributed computation are addressed in Chapter 5. We present our results in Chapter 6. Finally, Chapter 7 gives concluding remarks and describes the future research which is related to the scope of this work.

Chapter 2

Related Work in Object Recognition

2.1 Imaging Models

Object recognition is one of the most important task of the field of computer vision. Given a set of input images, we are interested in knowing if the specific object appears in the scene, and if so, where? There are several issues related to this task, namely, the representation of the object, the detection of the object, and the location of the object.

In this section, we consider the kinds of models that have been applied in the past to the imaging process. There are several kinds of assumptions about the objects, for example, rigid or nonrigid, two dimensional or three dimensional, curvilinear or polyhedral, occluded or not occluded. The pixel size, shape and connectivity might even be nonuniform over the whole image [111]. These assumptions drastically affect the kinds of images that must be analyzed.

There are various kinds of images that are obtained by different sensors. Basically, they can be categorized into two classes, intensity images and range images. A pixel value in an intensity image represents the intensity of a specific spectral distribution, for example, the gray images for visible light and FLIR image for infrared. A pixel value in a range image represents the distance metric at that position, as in a LADAR image (an image captured by a laser radar device) or a raw HRR image (an image captured by a high resolution ranging radar [81]).

There are different assumptions for the image model, namely, an orthographic projection, weak perspective projection, or perspective projection. The choice of the projection model affects the features that we can use. The orthographic model is based on the assumption that the depth of the objects in the scene and their variance are small compared to the focal length of the camera. The projection collapses the scene onto a plane by an orthogonal mapping. Thus the z-coordinate is discarded, and only x and y-coordinates are kept.

The weak perspective imaging model is approximated by an orthographic projection plus a scale factor [51]. The weak perspective imaging model is more accurate than the orthographic projection, but still allows simple computation. The underlying assumption is that the depth of the centroids of the objects in the scene is large compared to the focal length of the camera, and that the depth variation of the objects are small compared to the depth of their centroids. This model uses a single scale factor to capture the size of the change that occurs with increasing distance. The approximation becomes poor when an object is deep with respect to its distance from the viewer. This approximation can be modeled by a similarity transformation, i.e., rotation, translation, and scale.

Some researchers also use affine transformations to approximate the perspective projection [62, 61, 64, 2]. This approximation is more general than the similarity transformation since it can handle shear effects. Under the same assumption as the weak perspective model, two different images of the same flat object can also be related by an affine transformation.

In some application areas, the affine approximation is still inappropriate. For example, oblique views of roadways, landing strips, and buildings are not modeled well by affine transformations. In general, when the depth variation of the objects are comparable to the focal length of the camera, we need a more accurate model—the pinhole camera model. The pinhole camera model is a first-order approximation to the model of imaging devices. The projection is modeled by mapping scene points onto the image plane by a line through a single point which is the lens center. This projection process is also called perspective projection. The physical layout of pinhole camera model is that the lens center lies between the image plane and scene points. Usually it is more convenient to use a revised camera model [4, 100, 28]. The point of projection lies between the lens center and the corresponding scene point. The mathematics is the same, but the revised camera model is more intuitive and easier to compute.

Perspective approximations are also studied for application to object recognition. For example, two other perspective approximations, namely, paraperspective and orthoperspective transformations are surveyed by Aloimonos [1]. He gives formulas relating the image to the world for the properties of length and area. Both approximations are formed by two steps: orthographically projecting to a particular plane passing through the center of mass of the region, and then perspectively projecting to the image plane. The particular plane is parallel to the image plane for paraperspective projection and perpendicular to the line connecting the focal point of the camera for orthoperspective projection.

There are assumptions for the number of images to be used (monocular image, binocular image, trinocular image) or image sequence. There are also two categories of the approaches, passive vision and active vision, depending upon whether the algorithm has the direct control of camera parameters, including view angle and focusing distance

[58, 111, 32].

There are different characteristics for the recognition methods, for example, shape from texture, shape from contour, shape from shading, etc. Depending on the properties of the images and the objects, we can categorize the various recognition problems into the following classes: 2-D from 2-D, 3-D from 2-D, 3-D from 3-D. We can also have the factor of *time* as another dimension; then the problem becomes very complicated.

The difficulty is that many problems are ill-posed and there is no benchmark to evaluate the performance of various approaches. Even though facing these difficulties, researchers in this area focus on various subproblems: model building [118, 46, 16, 104], model localization [26, 40], object recognition [2, 13, 14, 15, 22, 29, 33, 37, 62, 51], 2-D or 3-D curve matching [119, 42, 43, 44, 54, 99, 49], motion parameter estimation [17, 30, 31, 84, 92, 71, 68, 73, 109], motion tracking [77], camera calibration [66, 72, 48, 85, 114, 115, 20, 41], correspondence problem [23, 38, 78, 113], invariant theory [6, 11, 34, 80, 7, 102, 116], etc. A comprehensive survey of previous work related to the field of 3-D object recognition can be found in Besl and Jain [9], Suetens *et al.* [105], Chin and Dyer [22], and Dohond and Aggarwal [27]. We do not intend to review all the problems in the whole field, instead, we only survey the work related to our approach involving geometric hashing. However, we are concerned with matching patterns to patterns, and so we include treatments of the correspondence problem, stereo vision, as well as model-based object recognition. We discuss approaches for different problems and their relations to the task of object recognition in the following sections.

2.2 Object Recognition

We first categorize the approaches to 3-D object recognition into two classes, i.e., the methods that use a monocular image and the methods that use multiple images. For monocular image methods, only one snapshot of the scene is given. The recognition task is performed using the information from a single image. Usually prior knowledge about the object is available. Methods that use prior knowledge of the objects are also called "model-based". To our knowledge, most successful recognition systems are model-based [86]. The matching is performed in the 2-D image space either by searching the parameters that map the 3-D objects to 2-D image [75, 51], or using view invariant features to extract the correct model [62, 61, 64, 91, 37]. Matching 3-D objects in 3-D space using a single 2-D image is difficult, since there are infinitely many object shapes in 3-D space that all yield the same image after projection [55].

For multiple image methods, 3-D depth information is reconstructed using multiple images before performing matching in 3-D space. There are several ways to obtain 3-D information from multiple images. Current methods include binocular or stereo vision

systems [27, 118, 16], trinocular vision systems [3, 32], and depth from image sequences [109, 71, 103]. Most of the approaches emphasize the extraction of parameters, i.e., the parameters related to these multiple images and the parameters related to the world coordinate and camera coordinate systems. There are several uses for the motion parameters. For example, motion parameters can be used to estimate of the movement of the object, or to reconstruct the depth information of the objects. There are other methods to obtain the depth information using special sensors, for example, LADAR images or other radar modalities. Pixel values in a range image are then related to the depth information. For example, Flynn and Jain [33] use range images and a hashing scheme to recognize 3-D objects.

Kanatani [55] classifies matching methods into two classes: 3-D Euclidean approaches and 2-D non-Euclidean approaches. In the first class, 3-D shapes are backprojected into 3-D space given the 2-D intensity image. Some constraints on the object are applied to select a particular solution. The disadvantage of this approach is that the resulting 3-D recovery equation may be very complicated; analytical solutions are difficult to obtain. Numerical methods always suffer from the difficulty that the iterations may not converge, multiple solutions may exist, and the computation time may be long. For the second class, the matching is performed in the image space using image characteristics. In this approach, various invariant properties over groups of transformations are used. The difficulty is that it is not easy to find good invariant properties that are rich enough to describe a broad range of objects and robust enough to noise. The noise comes from the sensor as well as the feature detectors. Sometimes the distinction between these two approaches is not clear-cut. Hybrid methods that mix these two classes also exist. Our classification of the approaches is similar to Kanatani's. The difference is that the methods that use multiple images belong to a 3-D Euclidean approach in his classification scheme, while the methods that use monocular images belong to 2-D non-Euclidean [62] or hybrid approaches [51].

Usually a model-based object recognition system can be divided into four modules: feature extraction, model building, matching, and the verification and refinement module. The feature extraction module feeds the extracted features to the model building module and the matching module. The reason for the use of feature extraction is not only for the reduction of matching complexity, but also for robustness and flexibility of the recognition system. The goal is to recognize the object from the test image which is different from the images that are prestored in the database. Matching the objects in the test image from the prestored image database directly is unrealistic. The features can be classified as region based or edge based. For a region based approach, several methods have been proposed in order to segment stable regions. For example, there are energy minimization methods [93], and multiresolution schemes [77]. For edge based approaches, high curvatures, line

segments, circles, etc., are extracted based on an edge map and are used as the matching primitives.

In next two sections, we review related work based on our classification scheme. Their relationship to the task of object recognition will be discussed. In the last section, we review several results of invariant theory, which is related to the use an indexing approach to the task of object recognition.

2.3 Monocular Image Methods

Applications that use monocular images are surveyed in this section. The subsequent section is devoted to methods that use multiple images.

2.3.1 Matched filter

Novak and Netishen [81] use four spatial matched filter classifiers for the problem of automatic target recognition (ATR) using synthetic aperture radar (SAR). The concept of a matched filter has been applied to the area of signal processing for years [82]. The candidate regions of interest are extracted by a CFAR (constant false alarm rate) detector. Then the position and orientation of the detected object are determined by a template matching process. This step is equivalent to applying a 2-D matched filter to the candidate objects. Three textural features, the standard deviation, the fractal dimension, and the ranked fill ratio, are also extracted at this stage. Finally, a four-class classifier is used to reject false alarms and to classify the detected objects. The approach is like the classical template matching method but performed in a more efficient way. The method can not handle targets at variable scales and orientations other than those of the predetermined models.

2.3.2 Generalized Hough transform

The original idea of the Hough transform is that instead of performing the search by template matching, we can transform the problem to searching for a peak in parameter space. The Hough transform provides a robust scheme for the extraction of various features even when occlusion occurs [28]. A comprehensive survey of variant Hough transforms is given by Illingworth and Kittler [52].

Linnainmaa *at al.* [68] use the technique of generalized Hough transforms to solve the pose determination problem. By matching triples of points on the object to possibly corresponding triples in the image, all six parameters of the object position are estimated. Artificial polyhedral objects are used in their experiments. The result can be used to build the model or to reconstruct the model from the image so that we can perform matching in 3-D space.

Both the Hough transform and geometric hashing involve a voting process. The main difference between the generalized Hough transform and geometric hashing is the representation of the index space for the recognition stage. In the generalized Hough transform, the range of parameters is a continuous unbounded space (unless geometric constraints are used to limit the space), whereas in geometric hashing, the voting process uses a discrete predictable collection of object model transformations.

2.3.3 Iteration method

Lowe proposes the SCERPO computer vision system to recognize 3-D objects from unknown viewpoints in a single gray-scale image [75]. The objects are represented as polyhedral solids. By using viewpoint invariants as trigger features to initiate a search process, the best viewpoint parameters are found by means of a least-squares-error process in image space. Newton's method is used to guide the optimization.

One of the major contributions of his approach is the proposed *perceptual organization* process [74, 75]. Perceptual organization can be defined as the ability of the human visual system to draw relevant groupings and data structures from an image without any prior knowledge of its contents. He uses groupings of image elements to assist in establishing the spatial correspondence between the model features and their image counterparts. Collinearity, proximity, and parallelism, are viewpoint invariant, and are used as the basis for grouping line features. However, parallel lines are not entirely perspective invariant, but they are invariant under a weak perspective imaging model. This grouping process can reduce the search space such that only a few initial matches are required.

The least-squares minimization process has been extended by Lowe to treat parameterized 3-D objects [76]. That is, the 3-D object is not necessarily rigid and is modeled by parameters. One of the innovations of his minimization process is to adjust the parameters simultaneously, which is different from a more typical approach to solving the parameters by separating the rotation part and translation part [72]. The author argues that although the projection from 3-D to 2-D is a nonlinear operation, it is a smooth and well-behaved transformation. Thus, it is a promising candidate for the application of Newton's method. The experimental results suggest that the method can apply to models with curved surfaces and a large number of internal parameters.

Chen and Stockman [19, 18] also use an iteration method to match curved 3-D objects using sets of 2-D image edge maps. Their approach includes the use of internal edges of the object in order to compute more accurate computation of pose [18]. Newton's method and least-squares minimization is used to compute the parameters of the transformation of 3-D objects to 2-D image space. A process analogous to template-matching is used to find the correspondences for the pixels on the curves. A quite accurate starting point is required in order for the solution to converge to the global minimum.

The iteration method can give only one solution when convergence is obtained. The solution may be wrong and is dependent on the initial value. The iteration method provides a way to refine the final result after a good initial matching is derived.

2.3.4 Interpretation tree

Grimson and Lozano-Pérez propose a method called the *interpretation tree* to organize the search for consistent matches between sensed data and object surfaces [40]. The method can be categorized as a *hypothesis-and-test* paradigm. The generation and exploration of the hypotheses for possible matching model and image features is performed by constructing a tree in depth-first fashion. Certain geometric constraints are used to prune the search tree. However, the tree can potentially grow exponentially and the search process can become exhaustive search.

2.3.5 Alignment

Huttenlocher and Ullman use the term *alignment* to refer to the transformation from model to image coordinate frames [51]. They also define the alignment system to find analytic solutions for the parameters from corresponding model features and image features. They propose a method for computing a transformation from three non-collinear points under a weak perspective assumption. The system is operated in a prediction-and-verification fashion. After a possible alignment from a pair of triplets of points is determined, complete edge contours are then used to verify the hypothesized match. For m model points and n image points, there are $\begin{pmatrix} m \\ 3 \end{pmatrix} \begin{pmatrix} n \\ 3 \end{pmatrix} 3!$ possible alignments, which are explored in an exhaustive search. In their implementation, each model point is associated with an orientation attribute. The intersection of two lines which are defined by two points and their orientations are used to induce the third point. Thus each pairing of two model and two image points forms an alignment. Using this technique, they reduce the number of alignments to $\begin{pmatrix} m \\ 2 \end{pmatrix} \begin{pmatrix} n \\ 2 \end{pmatrix} 2!$. Each possible alignment must be verified by matching the transformed model with the image. They organize the verification process in a hierarchical fashion: segment endpoints are used for initial verification first, and only those alignments that pass the initial verification use the entire contour to perform detailed verification. Actually, the solution they find is unique up to a reflection ambiguity. Since the alignment of features is local and is obtained by identifying corners and inflections in edge contours, the features are insensitive to partial occlusion. On the other hand, this method can be viewed as the estimation of the transformation parameters, which requires a least squares fit to improve the final position estimate as in Lowe's system.

Compared with Huttenlocher and Ullman's approach, Dhome *et al.* use triplets of

lines to compute the alignment under a full perspective transformation assumption [26]. The origin of the image coordinate system and the image lines form an *interpretation plane*. Using the condition that the line corresponding to the image line must lie on the interpretation plane, they obtain an equation related to rotation parameters only. Instead of solving the system of equations directly, they define another model coordinate system and viewer coordinate system and derive a polynomial equation of degree eight in one unknown. Special line configurations such as coplanar lines simplify the equation to fourth degree. By solving the equation iteratively, they derive several solutions that need to be pruned by the geometric constraints.

Horaud *et al.* provide an analytic solution using four non-coplanar points [50]. The non-coplanar points are equivalent to a pencil of three non-coplanar lines that share one of the four points. Instead of solving the transformation between the object coordinate system and the camera coordinate system directly, they also introduce another image coordinate system that is rigidly attached to the projections of the object features. The solution can be derived by solving a biquadratic polynomial equation in one unknown.

Research in this direction involves finding a way to derive a system of equations that can be solved analytically. Usually a special configuration for the model is assumed. But analytic solutions are usually not available. These systems all derive sets of equations whose solutions provide the transformations from model to image coordinate frames. The transformations form hypotheses that need further verification.

2.3.6 Indexing method

Based on affine invariant features, Lamdan *et al.* [62, 61, 64] develop an efficient technique, *geometric hashing*, for the model-based recognition task. A brief review of geometric hashing is given in Chapter 1. The basic idea of this method will be further addressed in Chapter 3.

Compared with the alignment method which performs an exhaustive enumeration over all possible pairs of minimal sets of model and image features, geometric hashing can be viewed as an indexing scheme that trades space for computation complexity in an average case setting. One of the distinct features of the geometric hashing method is that the matching among the models is performed simultaneously. That is, the information in each model is not matched to the scene one after another.

Clemens and Jacobs [25] discuss the fundamental limitations on the minimum amount of information that must be stored and on the speed up of the recognition process when using this kind of hashing.

2.4 Multiple Images Methods

In this section, we discuss the methods using multiple images. We review several results for motion parameter estimation. Based on the primitives used in the recovery equation, we classify the methods into two categories, namely, the methods based on point correspondence and the methods based on line correspondence. We can use the estimated motion parameters to build the models or to reconstruct the depth information in order to match the object in 3-D space. Finally, we review the methods that fuse the information provided by multiple sensors.

To perform 3-D motion analysis one can assumes that either the objects are moving or the observer is moving. Generally speaking, the problem for both cases are essentially the same. If all of the objects in the scene moving in the same direction, the situation is the same as if the observer were moving in the opposite direction. The most complicated case is that there are several objects in the scene and each object moves with different motion parameters. If the motion is large, the techniques such as optical flow analysis are not applicable. Then the problem of estimating the three-dimensional motion parameters can be divided into three steps: establishing the feature correspondences for all pairs of consecutive images in the sequence; estimating motion parameters from these correspondences; and then using the computed motion parameters to understand the local motion or to recognize objects based on the pre-established object models. The first step is usually the most difficult one, since an image may have more than one object and thus many features to choose from.

There are many researchers focusing on the correspondence problem. The early work done by Barnard and Thompson [5] find point correspondences by relaxation to perform disparity analysis between two images. Wang *et al.* [113] also use a relaxation method based on patterns of local features to find the correspondences. They use corner features between pairs of images. Grimson [38] develops a modified algorithm based on Marr and Poggio's work. In Grimson's paper, experiments using natural images are also provided. Mohan *et al.* [78] use the *figural continuity constraint* on the disparity data and develop an algorithm for error detection and correction of disparity. The figural continuity constraint on disparities is defined as the condition that the disparity along an edge contour changes smoothly, i.e., there should be no disparity discontinuities along a contour. The algorithm may be used for *edgel* based stereo matching, where an edgel is an edge having an extent of one pixel. Experiments using natural stereo images are also described in Mohan's paper. Hoff and Ahuja [46] integrate the processes of feature matching, contour detection, and surface interpolation to extract surfaces from stereo. Planar and quadratic patches are used as local models of the surface of objects, i.e., the approach uses surface information as the matching primitives instead of using feature points. Dohond and Aggarwal [27] review

recent developments in establishing stereo correspondence for the purpose of finding the 3-D structure of a scene. Chen and Huang [17] perform matching of 3-D line segments for correspondence analysis. They divide the matching step into two stages: first dealing with the orientation using tree-search and then dealing with the position of the line segments using a Hough clustering technique. They argue that some of the distance metrics that previous researchers used were not very suitable. The method developed by Yeshurun and Schwartz [120] is an interesting method among all these approaches. Their algorithm is motivated by the observed pattern of ocular dominance columns of a one-eyed macaque monkey. By arranging the image data in a columnar fashion, the algorithm can efficiently detect the disparity of the two images by using a "cepstral filter". The algorithm is suitable for implementing a real-time binocular vision system.

In general, there is a trade-off in computing the solution to the correspondence problem and accuracy of the information extracted from the images. That is, if the images are quite different, then more information can be extracted and the data will be more accurate. However, difficulty arises when we try to establish correspondences between the features. The difficulty for the correspondence problem arising from the 3-D objects themselves will be the distortion by perspective projection and occlusion of parts of objects by the other objects in the scene.

2.4.1 Point correspondences

The pioneer work done by Roach and Aggarwal [92] depends on numerical methods to derive the motion parameters by solving a system of nonlinear equations. They note that two views of six points or three views of four points are needed to provide an overdetermined set of equations when the images are noisy. Their paper does not provide experimental results.

There is no developed systematic theory for solving systems of nonlinear equations. Consequently, the equations have to be solved by numerical methods. The difficulty with numerical methods is that they sometimes fail because either they fail to converge or they converge to the wrong answer. The numerical methods also require proper initial guesses for the unknowns. Thus it is generally preferable to find closed-form solutions.

The work done by Tsai and Huang [109] is the first to solve this problem analytically. They use techniques in linear algebra to derive a closed-form solution to the problem. They show that seven point correspondences are sufficient to uniquely determine the motion parameters from two perspective views. They introduce a set of "essential parameters" that can uniquely determine the motion parameters up to a scale factor and can be derived from eight image points. That is, given eight point correspondences in general position, the essential matrix E can be determined uniquely by solving eight linear equations. The approach is of uncertain noise immunity. According to their simulation

results, the error grows rapidly as the perturbation of image points grows. As an example, when the noise in the location of image points is one percent of the disparities, the error for translation and rotation parameters can be up to 54% and 15% respectively. If twenty point correspondences are given, the error can be reduced to around 10% for translation parameters and 1% for rotation parameters, which is still quite large. It is quite easy to produce one percent error positional information during image processing.

Although the work done by Fang and Huang [30, 31] still uses point correspondences and numerical methods. They show experiments using real world images. In the first paper, they compare different numerical methods based on results from Tsai and Huang's work [109]. A more detailed error analysis is also provided. They show that if nine points are not on a second-order surface passing through the viewing point, then the solution of the motion parameters can be uniquely determined. According to the experiments reported by Fang and Huang, a modified Newton method and a modified Levenberg-Marquardt method behave well for these motion equations, yielding an average success rate of 70%. The failures are due to the procedure converging to a local minimum which is not the global minimum, or the procedure not converging. Their experiment is based on 54 sets of solutions with rotation angles varying from 0.01 to 0.1 radians and the distance of the object from the camera, z, ranging between 5 and 50 focal lengths. One important result is that the distance z has a large impact on the accuracy of the solution to the equations: the larger the distance, the more difficult the task is, which Fang and Huang conjecture is due to the effect of round-off errors.

In a subsequent paper, Fang and Huang [31] give a method to find coordinates of corner features to subpixel accuracy. They use real images to find the motion parameters. The main conclusion is that the estimation of the general 3-D motion parameters is sensitive to errors in the raw data. This situation becomes more serious when they use real-world images from a low resolution camera. A small rotation (2-4°) is helpful for obtaining good results. A larger rotation makes it difficult to obtain a good initial guess for solving the nonlinear equations. The amount of translation in the z-direction, i.e., the relative scale between the two images, has a significant effect on the accuracy of the results. The larger the translation in the z-direction, the more accurate[1] the results. But a larger translation in the z-direction causes a larger scale difference in the two images, which makes matching the corresponding points more difficult. There are also interactions between the rotation and the translation. The results for motion parameter estimation using a TV camera are not very accurate and the angle of rotation can only be in the range of 1-5°.

Weng *et al.* [117] treat the problem of motion parameter estimation from point correspondences beginning from the work of Tsai and Huang [109]. Their contribution is to

[1]Note that this result is opposite to the one reported in the first paper [30], even though the algorithm is the same.

provide further error analysis and error estimation involving least-squares optimization or pseudoinverse.

2.4.2 Line correspondence

All the works surveyed in the previous section use point correspondences, whereas the work done by Liu and Huang [71] uses line correspondences. The significance of using line correspondences is that precise point correspondences are not required, and that the less informative matching of pairs of lines is likely to be more robust. Since it can be shown that it is impossible to solve the rigid body motion problem with line correspondences over only two frames, they use three frames to solve the problem. More specifically, they assumed that a 3-D object moves and three image views are taken by a stationary camera at three time instants. Correspondences of a line over three image frames are obtained. Then, there are two sets of motion parameters: R_{12} and T_{12} for the object moving from position 1 to position 2, and R_{23} and T_{23} for the object moving from position 2 to position 3. Each set of parameters contains rotation parameters R and translation parameters T.

Using the constraint that the normals of the three corresponding interpretation planes are coplanar, an equation with six unknowns results. Their approach uses an iterative method to solve this nonlinear equation based on six or more line correspondences over three frames, yielding the rotation part. After the rotation parameters are found, the remaining unknown is the translation. Again, they considered the relations between the three parallel lines, and derived a linear equation from each line correspondence pair. They provide experimental result by simulation. Reasonable results are obtained when the noise is less than 1%.

The main advantages of their approach is that (1) they use line correspondences only; (2) they separate the rotation part and the translation part. However, since the input data for determining the translation involves the results of the rotation estimation, the computed results of the translation estimation are sensitive to the errors of the solution of the rotation part. What is worse, iterative methods that are used in solving the rotation part often converge to a local minimum instead of the global minimum. These methods must start with a good initial guess to obtain good results. Liu and Huang's experiments shows that convergence to the correct solution requires that the rotation lie in a quite small range ($\pm 3^\circ$ to $\pm 5^\circ$). Liu *et al.* [72] use line or point correspondences to determine the camera location, following up on their previous work [71], and making use of the work of Liu and Huang.

Spetsakis and Aloimonos [103] also compute the 3-D motion and structure from dynamic imagery using line correspondences only. As compared with Liu and Huang's approach which uses an iterative method to solve the nonlinear equations for the rotation parameters, they develop a calculus to obtain a closed-form solution for this problem.

Instead of separating the rotation part and translation part, they adopt the concept developed by Tsai and Huang [109] for computing the essential matrix with 26 variables (essential parameters).[2] Two independent linear equations[3] can be obtained from one line correspondence. Thus they require at least thirteen lines to solve for the 26 parameters.

Two methods are suggested to derive the solutions when more than thirteen corresponding lines are available: the least-squares technique and a *Singular Value Decomposition* (SVD) method. The main result is that they derive a closed-form solution for the motion parameters from the essential parameters. The uniqueness of the solution is also discussed. Experimental results from simulations show that their algorithm is unstable for a small number of lines but becomes stable as the number of lines increases. For 1% error[4] in input locations and for thirty line correspondences, they obtain about 1.8% error in translation estimation. As compared with Tsai and Huang's work [109], this shows that the line-based algorithm is more stable than an algorithm based on point correspondences.

2.4.3 Multisensor fusion

Since there are several kinds of sensors and several kinds of feature detectors are available nowadays, the fusion of various information becomes an appealing approach. The cooperative-method paradigm has been applied to various applications. For example, Schufelt and McKeown, Jr. [101] fuse hypotheses to detect man-made structures visible in aerial images. The information provided by four kinds of building extractors, BASE, SHADE, SHAVE, and GROUPER, are fused together by a simple strategy. The BASE (builtup area building extraction) building extractor uses lines and corners to extract buildings. The SHADE (shadow detection) building extractor is based on a shadow analysis method. The SHAVE (shadow verification) is a hypothesis verification system using shadow analysis. The GROUPER is a system for grouping fragmented building hypotheses using the edge information of buildings and shadows. The segmentations of the final result are generated by applying connected component region extraction to the accumulator image of the building extractors followed by a thresholding. The threshold value one is used in their experiments which in fact the resulting regions are the geometric unions of the regions detected by these building detectors. Their experiments show the performance improvement over individual building extraction methods.

Chu and Aggarwal [24] combine region and edge information extracted from different types of sensors and image processing methods. A cost function is defined and a maxi-

[2] These unknowns are dependent; only five of them are independent.

[3] The equations are nonlinear in terms of the motion parameters but are linear in terms of the essential parameters.

[4] I.e., the length of the error vector is $f/100$, where f is the focal length.

mum likelihood estimation strategy is used to optimize the criteria of fidelity, smoothness and connectivity. Combining different sources of information, they obtain a better segmentation results.

2.5 Invariant Theory

In this section, we review the work that addresses invariant theory. As described in the previous section, to facilitate the recognition process, projective invariant features may be needed, using either affine invariance or perspective invariance. The purpose of the study of the invariant theory is to design suitable and stable invariant features that can be used in 3-D object recognition systems.

2.5.1 Non-existence of general-case view invariants

It is desirable to define a general-case feature that is viewpoint invariant. But Burns *et al.* [11, 12] provide an argument that proves that there is no *general-case* viewpoint invariant feature defined for any number of points under orthographic, weak perspective, or perspective models. Only in *special-case*, that is, only for special configurations of 3-D points, do invariant features exist.

Their argument shows that general case non-trivial view invariant does not exist for the orthographic, the weak perspective, and the perspective projection models. Only for special configurations such as coplanar points will invariants exist.

In next section, we list several results for various invariants.

2.5.2 Invariant features

We use invariant features to recognize an object by its intrinsic properties, independent of the particular view. Invariant features are defined in terms of a specific projection model. Some features are invariant with respect to affine transformation, whereas some are invariant to perspective transformation. The invariant should be formulated in terms that they are easy to compute from a particular image of the object and robust to the existence of noise.

The well known cross-ratio [28, 94] relates collinear points in the plane to the projections through a point onto a line. It is a concept relating distances between collinear points. The cross-ratio depends only on relative distances along the line and is independent of the coordinate system.

The cross ratio is an invariant quantity under linear fractional transformations which can be formulated by two general methods [6]: the ratio-of-determinants method and a homogeneous equations method. Linear fractional transformations are mappings that take

each complex point z into a complex point w according to the equation

$$w = T(z) = \frac{az + b}{cz + d},$$

where $ad - bc \neq 0$.

The spirit of the ratio-of-determinants method is to judiciously select combinations of points so that the determinants involving the parameters (a, b, c, d) cancel in the numerator and the denominator of a ratio; leaving the desired invariant as a residue. The idea of the homogeneous equation method is to use the constants in the linear fractional transformation as place-holders that define a homogeneous relationship among the "object-points" $z_1, \ldots, z_4$ and "image-points" $w_1, \ldots, w_4$. This method is useful in making generalization.

Generalizing from the idea of the cross-ratio, Barrett *et al.* [6] develop the following invariants:

- Cross-ratios in two-dimensions (noncollinear object points) for binocular images. The generalization to multiple nonparallel images by using the homogeneous equations method involves nine images with six image points.
- Cross ratio between the coplanar object points and their corresponding projected image. The generalization to multiple nonparallel images by using the homogeneous equations method involves nine images with six image points.
- Cross ratio between the coplanar object points and their corresponding projected image.
- Cross ratio in three-dimensions (noncoplanar object points) for binocular images. The generalization to multiple nonparallel images involve 34 images with eight object points which are distributed in 3-D space. The ratio of functions of image points is equal to the ratio of functions of volumes of tetrahedron defined by the object points.

Arbter *et al.* [2] develop a method to produce a set of normalized coefficients for *Fourier descriptors* which are invariant under affine transformations. Fourier descriptors are a method of describing the shape of a closed figure, and has been used in aircraft identification for years [112, 59, 21]. Note that the contour descriptors suffer difficulty when objects are occluded.

Forsyth *et al.* [34] construct shape descriptors for planar objects that are unaffected by the transformation between the object and the image plane. In addition to the invariants in the previous paragraphs, some other invariant examples that may be interesting can be found in their review. For example, they define:

- Differential invariants, which are invariant functions of the position and derivatives of a curve such as curvature and torsion at a single point.

- Projective invariants for pairs of plane conics. By writing planar conic as $\vec{x}^t C \vec{x} = 0$ for $\vec{x}^t = (x, y, 1)$ and a symmetric matrix C, a pair of coplanar conics has two scalar invariants $I_{C_1C_2}$ and $I_{C_2C_1}$. Here $I_{C_1C_2} = \text{trace}(C_1^{-1}C_2)$.

- Projectively invariant measurements. If there is a distinguished conic curve in the plane (say, C), then for two points $\vec{x_1}$ and $\vec{x_2}$ that do not lie on the conic, the function
$$\frac{(\vec{x_1}^t C \vec{x_2})^2}{(\vec{x_1}^t C \vec{x_1})(\vec{x_2}^t C \vec{x_2})}$$
is independent of the frame in which the points and the conic are measured.

- Permutation invariants. By using symmetric functions, they can define new invariants based on the original invariants that are unaffected by permutations of their argument.

Forsyth *et al.* also discuss two techniques for constructing the invariants [34], the infinitesimal method and the symbolic method. The infinitesimal method gives invariants as the solutions to a system of differential equations which is applicable to all Lie groups. Symbolic methods have been used for finding invariants of forms under the action of a range of groups by a number of mathematicians in the nineteenth century.

The affine invariant of planar point sets used by Lamdan *et al.* [62, 61, 64] is based on the concept that when an affine basis is specified by a triplet of non-collinear points, then the coordinates of all the other points that are given in the coordinate system of the triplet are affine invariant. That is, suppose three non-collinear points e_{00}, e_{10}, and e_{01} are used as an affine basis. Then any point v in the plane can be represented as
$$v = \xi(e_{10} - e_{00}) + \eta(e_{01} - e_{00}) + e_{00}.$$
That is, v has coordinate (ξ, η) in the coordinate system formed by e_{00}, e_{01}, and e_{10}. The coordinate (ξ, η) for point v is an invariant quantity if the entire plane is transformed by an affine transformation.

Similarly, Lamdan and Wolfson further extend the above concept to the case of affine transformation from 3-D space to a 2-D image [64]. The singular affine transformation is to approximate the perspective transformation. A set of four non-coplanar points $e'_0, \ldots, e'_3$ in 3-D space is chosen as the 3-D affine basis. Any 3-D point e' is defined by its coordinate triplet in this coordinate system. Since in the 2-D image space, only two linearly independent vectors are required to span the space, the invariant coordinate of a point in 3-D space is no longer one-to-one corresponding to a single point in 2-D

Reference	Name & Features	Special configuration	Comment
[28] [94]	Cross ratio (four points)	Collinear	Distance of points and their projection on the line
[6]	Angle cross ratio & area cross ratio (five points)	Noncollinear in a plane	Between binocular images
	Volume cross ratio (six points)	Noncollinear in a plane	Between nine nonparallel image planes
	Area cross ratio (five points)	Coplanar object point	Object points and their projected image points
	Volume cross ratio (six points)	Noncoplanar object points	Binocular images, volume of object points
	Volume cross ratio (eight points)	Noncoplanar object points	Between 34 nonparallel image planes, their volume of object points
[2]	Relative invariant (contour descriptor)	Coplanar	Invariant under affine transformation
[34]	Conic cross ratio (two conics)	Coplanar	Similarity transformation, projective transformation
	Distance invariant (two points & one conic)	Coplanar but points do not lie on the conic	Projective transformation
[62] [61]	Point invariant (three points as basis)	Noncollinear	Affine transformation from 2-D to 2-D
[64]	Point invariant (four points as basis)	Noncoplanar	Affine transformation from 3-D to 2-D

Table 2.1: A summary of invariants given by different authors, organized by reference. For each invariant, preconditions and comments are given.

space. Instead, it corresponds to a line in 3-D hash space. The derivation is based on the linearity of the affine transformation for vectors.

Various invariants and their special configurations are summerized in Table 2.1. Other invariants can also be found [116, 80, 102]. We may find that some of the special configurations are too restricted. The goal is to find projective invariant features that are robust to noise, easy to compute, and functionally independent. In addition, the special configurations should be easily found in most objects that we want to recognize.

Chapter 3

Bayesian Updating

The basic idea of geometric hashing can be described as follows. In the model building phase, we normalize the features by using the notion of a basis set. The normalized features are tagged with the basis information together with the information of the model that possess these features. This process is done for every model and is repeated for every possible basis. The result is stored in a hash table. During the recognition phase, we randomly choose a basis in the test image and normalize the features. Those entries in the hash table that are close to the normalized features in the test image contribute votes. The votes with the same tag information are accumulated. Since we build the hash table redundantly, once we select a correct basis, we expect that the corresponding model with the correct basis will accumulate a large vote. The geometric hashing scheme for the case of similarity transformation is shown in Figure 3.1.

More precisely, a model consists of a prototype collection of features $\{(x_i, y_i)\}_{i=1}^{n}$. We will have many models, $m_1, \ldots, m_n$, so that $m_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{n_k}$. Different models may have different number of features n_k.

If T is a similarity transformation, then T can be described as a translation by some vector (a, b), followed by a rotation angle ϕ, followed by a scaling by s:

$$T = \mathcal{S}_s \circ \mathcal{R}_\phi \circ \mathcal{T}_{(a,b)},$$

$$\mathcal{T}_{(a,b)}(x, y) = (x + a, y + b),$$

$$\mathcal{R}_\phi(x, y) = (x \cos\phi - y \sin\phi, x \sin\phi + y \cos\phi),$$

$$\mathcal{S}_s(x, y) = (sx, sy).$$

In general, T could be a translation transformation, an affine transformation, or a projective transformation, and the dimension of the features can also vary. Specifically, for the case of similarity transformation, consider model m_k and basis points $(x_{k,\mu}, y_{k,\mu})$,

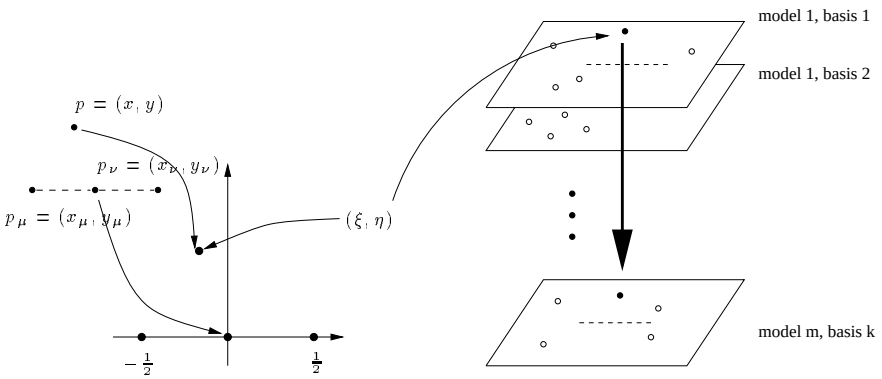

Figure 3.1: The geometric hashing scheme for the case of similarity transformation. Point pair (p_μ, p_ν) is used as a basis which defines a coordinate system. The normalized coordinate (ξ, η) for feature point p is used as the key to index into the hash table.

$(x_{k,\nu}, y_{k,\nu})$ from model k, which we will denote by p_μ and p_ν respectively. For every other point $p = (x_{k,i}, y_{k,i})$ in m_k, we define the hash function

$$h(p_\mu, p_\nu, p) = (\xi, \eta), \tag{3.1}$$

where (ξ, η) gives the feature value of $T(p)$, where T is defined to be the similarity transformation such that

$$\begin{aligned} T(x_{k,\mu}, y_{k,\mu}) &= (-1/2, 0), \\ T(x_{k,\nu}, y_{k,\nu}) &= (1/2, 0). \end{aligned}$$

Unlike the typical computer science notion of a hash function, h is continuous in all variables.

The object recognition problem is then stated as follows. Given a collection of models $m_1, \ldots m_n$, with each model formed by a set of features $m_k = \{(x_{k,i}, y_{k,i})\}_{i=1}^{n_k}$ and given a scene $S = \{(\hat{x}_j, \hat{y}_j)\}_{j=1}^{s}$, find *instances* of models in the scene. An instance is defined to be a model number k, a transformation T, and a subset S' of S such that for each feature $(\hat{x}_j, \hat{y}_j) \in S'$ in the subset of scene features, some transformed model feature approximates it: $T(x_{k,i}, y_{k,i}) \approx (\hat{x}_j, \hat{y}_j)$. In order to be a valid instance, the number of features in the subset S' must be some large fraction of the number of features n_k in model m_k.

During the preprocessing stage of geometric hashing, hash entries are formed by computing the normalized features given by Equation 3.1. Each hash entry is *tagged* with information (m_k, p_μ, p_ν). The process is repeated for all reasonable bases for all models.

Conceptually, a filter is formed by all the normalized features corresponding to the same (model, basis) information. The hash table consists of a set of filters. The recognition problem becomes the problem of finding the filter with the maximum response given the normalized scene features. In reality, all the filters are collapsed together and form the entire hash space.

In the recognition stage, features are extracted from a scene, yielding a set of $S = \{\hat{p}_j = (\hat{x}_j, \hat{y}_j)\}_{j=1}^{s}$. A pair of points are chosen as a *trial basis*, say p_α and p_β. Using the trial basis, we perform a set of *probes*, which together constitute a single *trial*. A probe consists of the computation of a hash value $h(p_\alpha, p_\beta, p) = (\xi, \eta)$, based on a point $p \in S$ and then a determination of all entries that lie close to (ξ, η) in hash space. For each such entry, we can compute a distance from its location to (ξ, η), and we then increment an accumulator associated with its tag (a model number k and a pair of point indices in m_k, say (μ, ν)), by an amount related to that distance. This is done for every entry near (ξ, η) and is repeated for all possible probes as p varies over S. The exact meaning of "near" in the hash space, the distance metric, and the accumulator increment function, are components of the geometric hashing algorithm that are specified by the Bayesian reasoning interpretation. We next turn to this interpretation.

Geometric hashing has a Bayesian interpretation [86, 91]. Our interpretation will extend this formulation, but the resulting formula will remain essentially the same. We describe our Bayesian interpretation in the next section. The Bayesian interpretation provides a basis for assigning the proper weight for the vote of a normalized scene feature voting to a nearby entry in hash table. The generalization for the case that various number of features for each model and the case that each feature carries its own individual non-obscuration ratio are also discussed. The definition of non-obscuration will be addressed later. We conclude this chapter with the derivation of distribution for invariants. The formula will be used in the implementation.

3.1 Bayesian Formulation of Geometric Hashing

The Bayesian framework for geometric hashing is described in this section. Our treatment reflects the work of Rigoutsos and Hummel [86, 91], although some refinements in the formulation are incorporated. We form the hypothesis that model m_k is present in the scene with basis B in m_k matching the basis B' in the scene. We denote the hypothesis as the event $E(m_k, B, B')$. We are interested in knowing which hypothesis has higher probability $\Pr(E(m_k, B, B')|S)$, where S represents the normalized scene features,

$$S = \{\vec{u_j} | 1 \leq j \leq s\},$$

and s is number of features in the scene. From Bayes' theorem [82, 28], we can compute the *a posteriori* probability $\Pr(E(m_k, B, B')|S)$ from *a priori* probability $\Pr(E(m_k, B, B'))$:

$$\Pr(E(m_k, B, B')|S) = \frac{f(S|E(m_k, B, B')) \cdot \Pr(E(m_k, B, B'))}{f(S)},$$

where f denotes the probability density function. If we assume that all prior probabilities of the events $E(m_k, B, B')$ are uniform, then we are asked to compute and maximize

$$\frac{f(S|E(m_k, B, B'))}{f(S)}. \tag{3.2}$$

Actually, if model m_k is present in the scene, there may be many pairs of (B, B') for which the event $E(m_k, B, B')$ is true. However, we need to find one such pair if the model appears in the scene. It is equivalent to maximize the logarithm of Equation 3.2. That is, we wish to compute and maximize

$$\log\left(\frac{f(S|E(m_k, B, B'))}{f(S)}\right) \tag{3.3}$$

for each hypothesis $E(m_k, B, B')$.

If the features are conditionally independent, Equation 3.3 can be decomposed further. That is, we assume that

$$f(S|E(m_k, B, B')) = \prod_{j=1}^{s} f(\vec{u_j}|E(m_k, B, B')),$$

then Equation 3.3 becomes:

$$\log K + \sum_{j=1}^{s} \log\left(\frac{f(\vec{u_j}|E(m_k, B, B'))}{f(\vec{u_j})}\right),$$

where

$$K = \frac{\prod_{j=1}^{s} f(\vec{u_j})}{f(S)}.$$

Since K is independent of $E(m_k, B, B')$, the $\log(K)$ term can be ignored. Thus, we need to compute the following equation:

$$\sum_{j=1}^{s} \log\left(\frac{f(\vec{u_j}|E(m_k, B, B'))}{f(\vec{u_j})}\right). \tag{3.4}$$

The denominator $f(\vec{u_j})$ is the prior probability that a normalized feature appears at location $\vec{u_j}$. The features are chosen randomly from a random scene. We will assume in

many cases that the features are uniformly distributed in the entire image domain. For example, suppose that the features represent normalized position information, where the normalization induces an affine transformation

$$\vec{u} = A\vec{x} + \vec{b},$$

the density distribution is modeled by

$$f(\vec{u}) = \frac{1}{\det(A^{-1}) \cdot |R|}, \tag{3.5}$$

where $|R|$ is the area of the image domain, $\det(A^{-1})$ is the *Jacobian* of the transformation from the image domain to the normalized feature domain. In other cases, the distribution is nonuniform, such as the case described by Rigoutsos and Hummel [91, 86].

The numerator of Equation 3.4, $f(\vec{u_j}|E(m_k, B, B'))$, will be derived in the next section. We will also discard the assumption that every model has a uniform number of features. Instead, we will assume that model m_k has n_k features.

3.2 Variable number of features for each model

Prior formulations of implementations of Bayesian-based geometric hashing have assumed that each model m_k has an equal number of features. Our treatment below relaxes this restriction.

The probability density function $f(\vec{u_j}|E(m_k, B, B'))$ is computed under the condition that event $E(m_k, B, B')$ is true. That is, we know that the normalized features $\{\vec{x_i}\}$, for $i = 1, \cdots n_k$, are embedded in the scene. We assume that each occurrence of model feature is represented by a Gaussian spike in the distribution function at this moment. We will derive the distribution for a normalized model feature in Section 3.3. A residual background distribution is also included. The conditional density function is thus:

$$f(\vec{u}|E(m_k, B, B')) = \tilde{\rho}(\vec{u}) = \frac{s - \beta n_k}{s} \cdot f(\vec{u}) + \frac{\beta}{s} \cdot \sum_{i=1}^{n_k} G_{C_i}(\vec{u} - \vec{x_i}), \tag{3.6}$$

where x_i's represent the expected features and each C_i represents the covariance matrix for its variation. Here, s is the number of features in the scene, and n_k is the number of features in model m_k. Further, G_{C_i} represents multi-dimensional Gaussian function with C_i as its covariance matrix. We also assume that a certain amount of non-obscuration, having a non-obscured percentage of the expected features, which we denote as β. The generalization for the case that each individual feature possesses its own non-obscuration rate will be discussed in Section 3.2.1. Plugging Equation 3.6 into Bayesian formula

(Equation 3.4), we have

$$\sum_{j=1}^{s} \log \left(\frac{s - \beta n_k}{s} + \frac{\beta}{s\rho(\vec{u_j})} \sum_{i=1}^{n_k} G_{C_i}(\vec{u_j} - \vec{x_i}) \right),$$

which is equivalent to

$$-s \log \left(\frac{s}{s - \beta n_k} \right) + \sum_{j=1}^{s} \log \left(1 + \frac{\beta}{(s - \beta n_k) \cdot \rho(\vec{u_j})} \sum_{i=1}^{n_k} G_{C_i}(\vec{u_j} - \vec{x_i}) \right).$$

The function ρ is an instance of the background distribution f. We can stop here. However, for any scene feature $\vec{u_j}$, we would like to associate it with at most one of the model features $\vec{x_i}$ that is the nearest. The reason is that a scene feature can not be matched to several model features when a basis is given. Note that a scene feature still can contribute to several hash entries if these entries come from a different model or if they are normalized by different bases. Suppose the closest entry is indexed by i_j. The summation of Gaussian can be approximated by a single term $G_{C_{i_j}}(\vec{u_j} - \vec{x_{i_j}})$. Thus, we obtain the following weighted voting formula:

$$-s \log \left(\frac{s}{s - \beta n_k} \right) + \sum_{j=1}^{s} \log \left(1 + \frac{\beta}{(s - \beta n_k) \cdot \rho(\vec{u_j})} G_{C_{i_j}}(\vec{u_j} - \vec{x_{i_j}}) \right). \tag{3.7}$$

The Gaussian term $G_{C_{i_j}}(\vec{u_j} - \vec{x_{i_j}})$ will contribute a vote to the corresponding bucket if the hash entry $\vec{x_{i_j}}$ lies within a certain range of the normalized scene feature $\vec{u_j}$. It will contribute zero if the closest entry is too far away from $\vec{u_j}$, or if the hash entry $\vec{x_{i_j}}$ has been "occupied" by another closer scene feature. In that case, the normalized scene feature $\vec{u_j}$ will be considered as a background feature which will contribute a vote with the amount of $\log \left(\frac{s - \beta n_k}{s} \right)$.

Due to the way we have formulated things, we observe that each scene feature $\vec{u_j}$ increments s, and hence has an effect on the bias term, and may (or may not) contribute to the second term. Disregarding the derivation of the formula, we could consider two plausible approaches for contributions by scene features:

(1) When a scene feature is associated with a model feature, the scene feature can contribute both to the log bias term and the Gaussian term. The reason is that the summation of these two terms is the the value of the conditional density function for that realization. The other Gaussian terms are neglected. In this case, we need the assumption that two model features are far enough that the contributions to all but one of the model features is negligible. If a scene feature can not be associated with any model feature due to the closest model feature being occupied, or, if it is too far from any model feature, it will contribute a vote to the amount of the log bias term only.

(2) When a scene feature is associated with a model feature, the scene feature contributes the Gaussian term only. Scene features without an association contribute to the log bias term. Again, we can use the first order approximation. Thus, there are at most n_k Gaussian terms and at least $s - n_k$ log terms summing together.

In either case, our formula given by the above derivation works if the number of scene features s is large compared to $\beta \cdot n_k$. When s is close to $\beta \cdot n_k$, or even smaller than $\beta \cdot n_k$, the analysis does not work. We need a more complicated analysis of the association of scene features to model features. For example, a model can accumulate the same vote regardless the number of background scene features in some cases. But our current formulation does not cover this case.

In the following sections, we will assume that s is large compared to $\beta \cdot n_k$ and use Equation 3.7 as the weighted voting formula.

3.2.1 Individual non-obscuration ratio of features

The above formula (Equation 3.7) assumes that the non-obscuration ratio for features is a constant. Suppose each feature has its own non-obscuration ratio. The revised formula gives important features more weight, where important features are ones that are more likely to be detected in the image of the model.

We recall that the expected density distribution for features under the condition that event $E(m_k, B, B')$ is true is $\tilde{\rho}$:

$$\tilde{\rho}(\vec{x}) = \frac{s - \beta n_k}{s} \cdot \rho(\vec{x}) + \frac{\beta}{s} \cdot \sum_{i=1}^{n_k} G_{C_i}(\vec{x} - \vec{x_i}).$$

Now we assume that each feature x_i has its own non-obscuration ratio β_i; the conditional density function becomes:

$$\tilde{\rho}(\vec{x}) = \frac{s - \sum_{i=1}^{n_k} \beta_i}{s} \cdot \rho(\vec{x}) + \frac{1}{s} \cdot \sum_{i=1}^{n_k} \beta_i G_{C_i}(\vec{x} - \vec{x_i}).$$

When each β_i is a constant β, the revised formula is the same as the old formula.

Following the same derivation as before, the revised weighted voting formula for normalized scene feature becomes:

$$-s \log(\frac{s}{s - \beta_t}) + \sum_{j=1}^{s} \log\left(1 + \frac{\beta_{i_j} G_{C_{i_j}}(\vec{u_j} - \vec{x_{i_j}})}{(s - \beta_t)\rho(\vec{u_j})}\right), \tag{3.8}$$

where

$$\beta_t = \sum_{i=1}^{n_k} \beta_i,$$

and index i_j refers to the nearest normalized model feature. In reality, we can decompose each non-obscuration ratio β_i into two parts, β_G and $\tilde{\beta}_i$, where β_i equals to $\beta_G \cdot \tilde{\beta}_i$. We call β_G the global non-obscuration ratio, which describes the visibility of the whole object. This constant can be assigned for each test image during the recognition stage. The local non-obscuration ratio $\tilde{\beta}_i$ describes the individual visibility of each feature and is determined in the model building phase.

The covariance matrix C_i captures the dispersion of each feature while the non-obscuration ratio β_i describes the importance and detectability of each feature.

3.3 The Distribution of Invariants

We next derive the formula for the distribution of normalized features (see Equation 3.7). Our derivation is systematic and is a standard procedure to compute the distributions when a Gaussian noise model is adopted. We compute only to a first order approximation, so that the transformation of a Gaussianly-distributed random variable will still be a Gaussian according to our formulas. As in [106], we make use of the following results, quoted from [47]:

Theorem 3.1 *Let $X_1, X_2, ..., X_n$ have a multivariate* Gaussian *distribution with vector $\mathbf{u}$ of means and positive definite covariance matrix $\mathbf{\Sigma}$. Then the* moment-generating function *of the multivariate* Gaussian *p.d.f. is given by $exp[\mathbf{t}^T\mathbf{u} + \frac{\mathbf{t}^T\mathbf{\Sigma}\mathbf{t}}{2}]$, for all real vectors of $\mathbf{t}$.*

Corollary 3.2 *Let $\mathbf{Y}^T = (\mathbf{Y}_1, \ldots, \mathbf{Y}_m)$ such that $\mathbf{Y} = W\mathbf{X}$, where $\mathbf{X}^T = (X_1, X_2, ..., X_n)$ and $W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ \vdots & & & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{pmatrix}$, a real $m \times n$ matrix. Then the random variable $\mathbf{Y}$ is $\mathcal{N}(W\mathbf{u}, W\mathbf{\Sigma}W^T)$.*

In the corollary, $\mathcal{N}(W\mathbf{u}, W\mathbf{\Sigma}W^T)$ means that $\mathbf{Y}$ is Gaussian with mean $W\mathbf{u}$ and covariance $W\mathbf{\Sigma}W^T$, where $\mathbf{Y} = W\mathbf{X}$ is a linear transformation of $\mathbf{X}$, which in turn is a multivariate Gaussian distribution with vector $\mathbf{u}$ of means and covariance matrix $\mathbf{\Sigma}$. This result allows us to compute the statistics of a linear transformation of Gaussian random variables.

Sometimes the transformation is not linear. We use a first order approximation to represent the transformed random variable as a linear combination of original random variables. Thus, for a transformation $\mathbf{Y} = f(\mathbf{X})$, we approximate

$$\begin{aligned} f(\mathbf{X} + \Delta\mathbf{X}) &= \mathbf{Y} + \Delta\mathbf{Y}, \\ \Delta\mathbf{Y} &\approx \nabla f(\mathbf{X}) \cdot \Delta\mathbf{X}, \end{aligned} \tag{3.9}$$

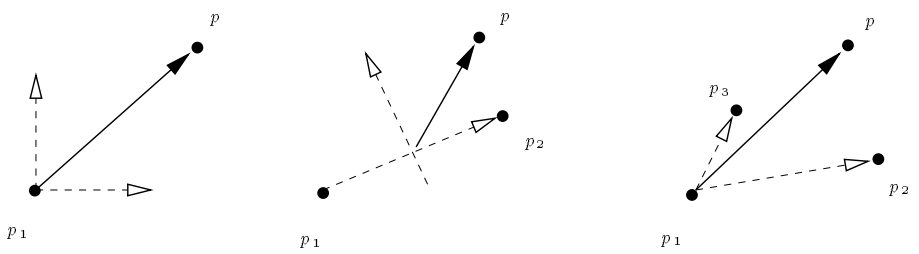

Figure 3.2: The configurations for (1) translation, (2) similarity, and (3) affine transformation invariant. Point p_i's is used as the basis to define a coordinate system. The coordinate of a point p in the defined coordinate system is used as the invariant.

where $\Delta\mathbf{X}$ and $\mathbf{X}$ are generalized to $n \times 1$ vectors. Vector $\nabla\mathbf{X}$ represents the perturbation of vector $\mathbf{X}$. The first derivative of function f should exist and be continuous. The ∇ operator is defined as a mapping from $\mathbb{R}$ to $\mathbb{R}^n$, i.e.,

$$\nabla f(x_1, x_2, \ldots, x_n) = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}].$$

We generalize the function f in Equation (3.9) to a $m \times 1$ vector of functions so that every element of the vector represents a function of $\mathbf{x}$. Thus, the ∇ operator can be generalized as a mapping from $\mathbb{R}^m$ to $\mathbb{R}^{m \times n}$. That is,

$$\begin{aligned} \nabla\mathbf{f}(\mathbf{x}) &= \begin{bmatrix} \nabla f_1(\mathbf{x}) \\ \nabla f_2(\mathbf{x}) \\ \vdots \\ \nabla f_m(\mathbf{x}) \end{bmatrix} \\ &= \{f_{i,j}\}_{m \times n} = [W_1^T, W_2^T, \ldots, W_m^T]^T, \end{aligned}$$

where

$$\{f_{i,j}\}_{m \times n} = \frac{\partial f_i}{\partial x_j}.$$

Using Equation (3.9) and the moment generating function of the multivariate Gaussian probability density function (Corollary 3.2), we are able to compute the covariance matrix of the distribution of the transformed values throughout this thesis. In Figure 3.2, we display the transformations used for the translation, similarity, and affine cases. Points p_i's with coordinate (x_i, y_i) are used as the basis to define a coordinate system. The coordinate of a point p in the new defined coordinate system is used as the invariant. The

general form of the invariant for 2-D point matching is:

$$h_q(\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_q, \mathbf{p}) = \begin{bmatrix} u \\ v \end{bmatrix} = A_q^{-1} b_q,$$

where A_q is 2×2 a matrix, b_q is a 2×1 vector, and $q = 1$ for translation, $q = 2$ for similarity, $q = 3$ for affine transformation. That is, for translation invariance, we have:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x - x_1 \\ y - y_1 \end{bmatrix},$$

for similarity invariance, we have:

$$\begin{bmatrix} x_2 - x_1 & -y_2 + y_1 \\ y_2 - y_1 & x_2 - x_1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x - (x_1 + x_2)/2 \\ y - (y_1 + y_2)/2 \end{bmatrix},$$

and finally, for an affine transformation, we have:

$$\begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x - (x_1 + x_2 + x_3)/3 \\ y - (y_1 + y_2 + y_3)/3 \end{bmatrix}.$$

Matrices A_q and b_q are given as follows:

$$\begin{aligned} A_1 &= I, (\text{the identity matrix}), \\ A_2 &= \begin{bmatrix} x_2 - x_1 & -y_2 + y_1 \\ y_2 - y_1 & x_2 - x_1 \end{bmatrix}, \\ A_3 &= \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}, \end{aligned}$$

and

$$b_1 = \begin{bmatrix} x - x_1 \\ y - y_1 \end{bmatrix}, \quad b_2 = \begin{bmatrix} x - (x_1 + x_2)/2 \\ y - (y_1 + y_2)/2 \end{bmatrix}, \quad b_3 = \begin{bmatrix} x - (x_1 + x_2 + x_3)/3 \\ y - (y_1 + y_2 + y_3)/3 \end{bmatrix}.$$

Depending on whether the basis points are perturbed or not, we use the distributions for the *exact matching* hypothesis and the *approximate matching* hypothesis as derived in the next two sections. We will use the following notation:

$$\xi_{ij} = x_i - x_j, \tag{3.10}$$

$$\eta_{ij} = y_i - y_j. \tag{3.11}$$

3.3.1 Exact matching hypothesis

Recall that we need to compute the density function under the condition that event (m_k, B, B') is true (Equation 3.6). Suppose basis B in model m_k matches basis B' in the scene *exactly*, i.e., there is no perturbation for basis points, then the invariant is a linear combination of the x and y-coordinates of p. Assume that point p is Gaussian distributed with covariance σ^2 in both the x-coordinate and y-coordinate. We also assume that the deviation in x-coordinate and y-coordinates are uncorrelated. By Corollary 3.2, matrix W is equal to A_q^{-1}, and we conclude that the invariant $h_q(p_1, \cdots, p_q, \mathbf{p})$ is Gaussian distributed with covariance C_q, where

$$C_q = \sigma^2 \cdot [A_q^T A_q]^{-1}.$$

Thus, the covariance matrix for each case is given by

$$\begin{aligned} C_1 &= \sigma^2 \cdot I, \\ C_2 &= \frac{\sigma^2}{\| p_2 - p_1 \|^2} \cdot I, \end{aligned} \tag{3.12}$$

and

$$C_3 = \frac{\sigma^2}{| \delta_{21} \times \delta_{31} |^2} \cdot \begin{bmatrix} \xi_{31}^2 + \eta_{31}^2 & -\xi_{21}\xi_{31} - \eta_{21}\eta_{31} \\ -\xi_{21}\xi_{31} - \eta_{21}\eta_{31} & \xi_{21}^2 + \eta_{21}^2 \end{bmatrix}, \tag{3.13}$$

where $(\delta_{21} \times \delta_{31})$ is defined as $(\xi_{21}\eta_{31} - \xi_{31}\eta_{21})$.

3.3.2 Approximate matching hypothesis

If the hypothesis for event (m_k, B, B') is that basis B in model m_k matches basis B' in the scene *approximately*, i.e., there are perturbations applied on coordinate of p as well as all the basis points p_i's. The covariance matrix for invariant h_1 becomes Σ_1,

$$\Sigma_1 = 2\sigma^2 \cdot I = r_1 C_1, \tag{3.14}$$

where $r_1 = 2$, and C_1 is defined by (3.12). The invariants h_q's are no longer linear combinations of coordinates of p_i's for $q \geq 2$. We use Equation 3.9 to derive the first order approximation and express the invariant as the linear combination of p_i's. By plugging the variables for A_q and b_q and arranging the random variables as $\mathbf{X_i}, \mathbf{Y_i}, (i = 1 \ldots q), \mathbf{X}, \mathbf{Y}$, we have the following relations for the coefficients and covariance matrices.

For the similarity transformation $(q = 2)$, we have

$$W_2 = \frac{1}{\| p_2 - p_1 \|^2}.$$

$$\begin{bmatrix} \xi_{21}(u-\frac{1}{2})+\eta_{21}v & -\xi_{21}v+\eta_{21}(u-\frac{1}{2}) & -\xi_{21}(u+\frac{1}{2})-\eta_{21}v \\ -\eta_{21}(u-\frac{1}{2})+\xi_{21}v & \eta_{21}v+\xi_{21}(u-\frac{1}{2}) & \eta_{21}(u+\frac{1}{2})-\xi_{21}v \end{bmatrix}$$

$$\begin{bmatrix} \xi_{21}v-\eta_{21}(u+\frac{1}{2}) & \xi_{21} & \eta_{21} \\ -\eta_{21}v-\xi_{21}(u+\frac{1}{2}) & -\eta_{21} & \xi_{21} \end{bmatrix},$$

Applying Corollary 3.2, we have

$$\begin{aligned} \Sigma_2 &= \frac{(4 \parallel (u,v) \parallel^2 +3)\sigma^2}{2 \parallel p_2 - p_1 \parallel^2} \cdot I \\ &= r_2 C_2, \end{aligned} \tag{3.15}$$

where

$$r_2 = \frac{(4 \parallel (u,v) \parallel^2 +3)}{2}. \tag{3.16}$$

Again, C_2 is the covariance matrix from the exact matching hypothesis case, in Equation 3.12. Note that $r_2 > 1$ always.

For affine transformation ($q = 3$), we have

$$W_3 = \frac{1}{(\delta_{21} \times \delta_{31})} \cdot$$

$$\begin{bmatrix} -\eta_{31}(\frac{1}{3}-u-v) & \xi_{31}(\frac{1}{3}-u-v) & -\eta_{31}(\frac{1}{3}+u) & \xi_{31}(\frac{1}{3}+u) \\ \eta_{21}(\frac{1}{3}-u-v) & -\xi_{21}(\frac{1}{3}-u-v) & \eta_{21}(\frac{1}{3}+u) & -\xi_{21}(\frac{1}{3}+u) \end{bmatrix}$$

$$\begin{bmatrix} -\eta_{31}(\frac{1}{3}+v) & \xi_{31}(\frac{1}{3}+v) & -\eta_{31} & -\xi_{31} \\ \eta_{21}(\frac{1}{3}+v) & -\xi_{21}(\frac{1}{3}+v) & -\eta_{21} & \xi_{21} \end{bmatrix},$$

Applying Corollary 3.2 again, we have

$$\Sigma_3 = \frac{(2(u^2+v^2+uv)+4/3)\sigma^2}{\mid \delta_{21} \times \delta_{31} \mid^2} \cdot \begin{bmatrix} \xi_{31}^2+\eta_{31}^2 & -\xi_{21}\xi_{31}-\eta_{21}\eta_{31} \\ -\xi_{21}\xi_{31}-\eta_{21}\eta_{31} & \xi_{21}^2+\eta_{21}^2 \end{bmatrix}. \tag{3.17}$$

By comparing Equation 3.13 and Equation 3.17, we find that C_3 and Σ_3 again are related only by a constant factor. We have

$$\Sigma_3 = r_3 C_3,$$

where

$$\begin{aligned} r_3 &= 2(u^2+v^2+uv)+\frac{4}{3} \\ &= 2(u+\frac{v}{2})^2+\frac{3v^2}{2}+\frac{4}{3}. \end{aligned} \tag{3.18}$$

Once again, note that $r_3 > 1$ always.

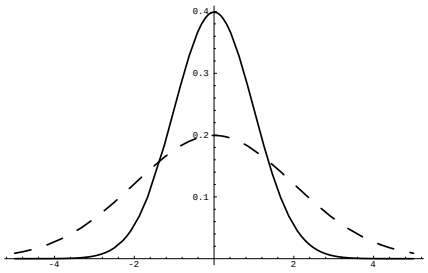

Figure 3.3: The plot of two Gaussian functions: $G(x)$ and $G(\frac{x}{2})$. Depending on x, $G(x)$ may be larger than $G(\frac{x}{2})$ or vice versa.

From Equation 3.14, Equation 3.16, and Equation 3.18, we observe that the probability density functions for the invariant values under the exact matching hypothesis and the approximate matching hypothesis have similar form to first order when the noise model is Gaussian. The covariance matrices for both cases are different by a constant factor which is dependent upon the coordinate of the normalized feature. The exact matching hypothesis possesses a smaller covariance than under the approximate matching hypothesis. As shown in Figure 3.3, depending on the distance to the mean, one hypothesis may accumulate a larger total vote than the other.

3.4 The Bounds for Weighted Voting

Given the weighted voting formula (Equation 3.7), we can consider the following problems:

1. What is the maximum vote that a single model/basis candidate can accumulate?
2. If the total vote is below a preset threshold, can we bound the number of votes and the proximity of those votes to model parameters?
3. Given a total vote, what is the range for the number of entries that lie within a given bound?

We analyze the above questions in the following subsections. We focus on the similarity case only. Since the covariance matrix is independent of the hash location for the exact matching hypothesis,[1] we discuss the case of the exact matching hypothesis with a uniform

[1] We assume that the variance of x and y-coordinate for each feature is uniform here.

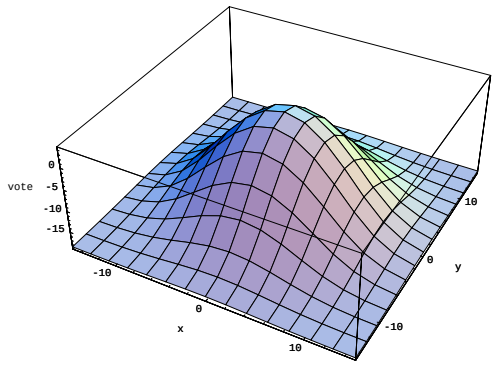

Figure 3.4: The visualization of weighted voting formula for exact matching hypothesis with uniform background density. The x and y deviation is in original image space.

background density. Plugging Equation 3.12 and Equation 3.5 into Equation 3.7, we have the weighted voting formula:

$$-s \log \left(\frac{s}{s - \beta n_k} \right) + \sum_{j=1}^{s} \log \left(1 + \frac{\beta \cdot |R|}{2\pi\sigma^2 \cdot (s - \beta n_k)} e^{\frac{-||p_2 - p_1||^2 \cdot ||\vec{u_j} - \vec{x_{i_j}}||^2}{2\sigma^2}} \right), \qquad (3.19)$$

where p_1 and p_2 are the selected basis. Equation 3.19 will be used in the following sections. A visualization of Equation 3.19 is shown in Figure 3.4. When numbers are required, we will assume that the original image is of the size 512×512 pixels, and that the standard deviation for both the x and y coordinates are 5 pixels. The number of features in the test image s is assumed to be 300, while the number of features in the model n_k is 30. The non-obscuration ratio β is assumed to be 0.6. In Figure 3.4, we also assume that there are typically fifteen features that contribute to the vote (i.e., $\beta = 0.5$). The x and y deviation is in the original image space.

3.4.1 Maximum vote

We are interested in knowing the maximum vote that we can get during the recognition phase. Let us call a feature *perfectly matched* if it coincides with the corresponding normalized feature in a model; i.e., there is no deviation for a perfectly matched feature. The exponential term becomes one for perfectly matched features. There are at most n_k features that can matched perfectly for model k. Thus the formula for the maximum vote

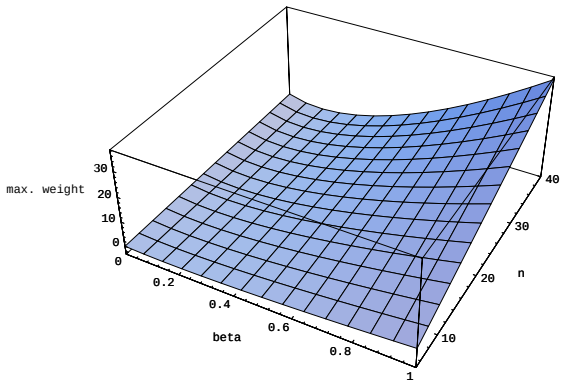

Figure 3.5: The function of the maximum vote as a function of β and n, where the number of scene features $s = 300$, and the number of perfectly matched features equals to $\beta \cdot n_k$. The non-obscuration ratio β varies from 0 to 1. The number of features n_k in model k varies from 5 to 40.

becomes:

$$-s \log \left(\frac{s}{s - \beta n_k} \right) + n_k \log \left(1 + \frac{\beta \cdot |R|}{2\pi\sigma^2 \cdot (s - \beta n_k)} \right).$$

When d is the number of features in the scene that exactly match model features, then the vote will be d times the maximum biased by a constant, thus:

$$- s \log \left(\frac{s}{s - \beta n_k} \right) + d \log \left(1 + \frac{\beta \cdot |R|}{2\pi\sigma^2 \cdot (s - \beta n_k)} \right). \tag{3.20}$$

The value of Equation 3.20 as a function of various subsets of (n_k, β, s, d) is shown in Figures 3.5, 3.6, 3.7, and 3.8, fixing certain variables and varying two other variables.

In Figure 3.5, the number of perfectly matched features d is assumed to be equal to $\beta \cdot n_k$, which assumes that we can estimate the non-obscuration ratio β accurately. The maximum vote that we may obtain becomes larger as the number of features in a model increases.

Figure 3.6 shows the dependence on the estimated non-obscuration ratio and the number of detected perfectly matched features. We assume that the number of model features is 30. We can obtain a larger total vote if we estimate the non-obscuration ratio β correctly and detect a larger number of perfectly matched features. The maximum attainable vote decreases when the number of features detected does not achieve the estimated non-obscuration ratio. The bias term $-s \log(\frac{s}{s-\beta n_k})$ will dominate Equation 3.20 if the non-obscuration ratio is low. On the other hand, it is possible that we can obtain a

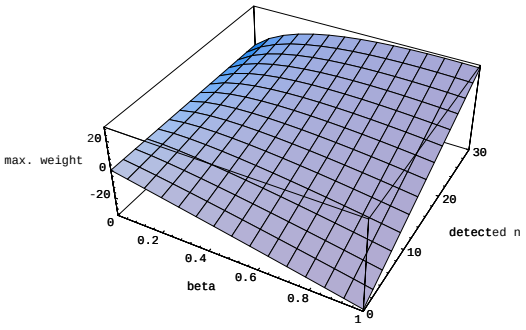

Figure 3.6: The function of the maximum vote as a function of β and the number of perfectly matched features, where the number of scene features $s = 300$, and the number of features n_k in the model k is 30. The non-obscuration ratio β varies from 0 to 1.

larger vote with a lower non-obscuration ratio even though the number of detected features is smaller. For example, suppose $d_1 = 6$ and $\beta_1 = 0.3$; the maximum vote is then $z_1 = -3.13292$. On the other hand, when $d_2 = 8$ and $\beta_2 = 0.8$, we have $z_2 = -10.9004$. Even though d_1 is smaller than d_2, we have z_1 larger than z_2 because we overestimate the non-obscuration ratio in the latter case. Correct estimation of the non-obscuration ratio thus certainly will affect recognition results.

The dependence on the non-obscuration ratio β and the number of scene features s is shown in Figure 3.7, assuming that we estimate β accurately, i.e., the number of detected features perfectly matches the predicted number of features and is equal to $\beta \cdot n_k$. A large total vote can be obtained when the number of scene features approaches the number of detected perfectly matched features, which means that only object features appear in the scene, and no background features are detected. However, when the signal to noise ratio is lower, a good estimation of β is less important.

The relation between the maximum vote and the number of features in the model n_k and the number of detected perfectly matched features d is shown in Figure 3.8. The non-obscuration ratio β is fixed as 0.8. For fixed n_k, the maximum vote is linearly proportional to the number of detected perfectly matched features d. For fixed d, the maximum vote decreases if the number of features in the model increases. If we change the non-obscuration ratio β, the slope of the relation will be changed accordingly.

To sum up, we may conclude that

1. Accurate estimation of the non-obscuration ratio β is important, which may affect

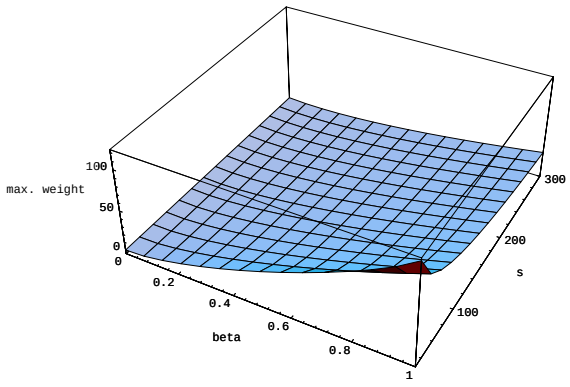

Figure 3.7: The function of the maximum vote as a function of β and the number of scene features, where the number of the number of features in the model $n = 30$. The non-obscuration ratio β varies from 0 to 1. The number of scene features s varies from 30 to 300.

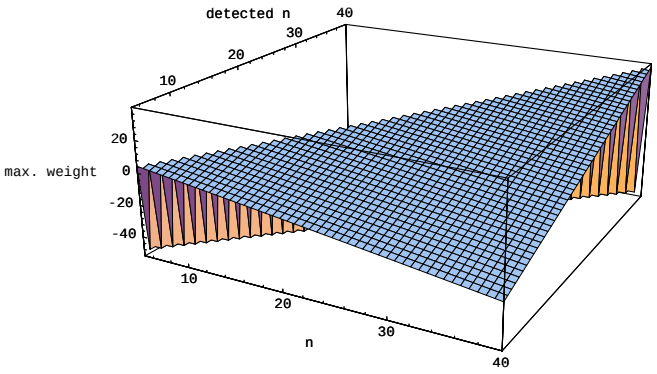

Figure 3.8: The function of the maximum vote we can obtain as a function of the number of detected perfectly matched features d and the number of features in the model n, where the number of scene features $s = 300$, the non-obscuration ratio β is fixed as 0.8. The number of features n_k in the model k varies from 5 to 40.

the recognition result;

2. We expect a larger total vote when the number of detected perfectly matched features is larger for a fixed non-obscuration ratio β;

3. When the number of scene features is only slightly greater than the number of detected perfectly matched features, we can obtain a very large vote;

4. The threshold value for recognition should be dependent upon the number of features in the scene s, the non-obscuration ratio β, and the number of features in each model n_k.

3.4.2 Deviations

Suppose that a scene feature contributes less than z to a particular model feature in the voting process. How close can it be to the model feature?

Let the actual vote be z_j,

$$\begin{aligned} z_j &= \log\left(1 + \frac{\beta \cdot |R|}{2\pi\sigma^2 \cdot (s - \beta n_k)} e^{\frac{-||p_2 - p_1||^2 \cdot ||\vec{u_j} - \vec{x_{i_j}}||^2}{2\sigma^2}}\right) \\ &= \log(1 + b \cdot e^{-\frac{x^2}{2a^2}}). \end{aligned}$$

Here, $a = \frac{\sigma}{||p_2 - p_1||}$, which is the standard deviation in hash space; $x = ||\vec{u_j} - \vec{x_{i_j}}||$, which is the distance of a scene feature a model feature in hash space; and $b = \frac{\beta \cdot |R|}{2\pi\sigma^2 \cdot (s - \beta n_k)}$, which is the coefficient of the exponential term. We have that

$$\log(1 + b \cdot e^{-\frac{x^2}{2a^2}}) \leq z.$$

That is,

$$\begin{aligned} 1 + b \cdot e^{-\frac{x^2}{2a^2}} &\leq e^z, \\ -\frac{x^2}{2a^2} &\leq \log\left(\frac{e^z - 1}{b}\right), \\ |x| &\geq \left[2a^2 \cdot \log\left(\frac{b}{e^z - 1}\right)\right]^{1/2} = c(z) \cdot a, \end{aligned}$$

where

$$c(z) = \sqrt{2 \log\left(\frac{b}{e^z - 1}\right)}.$$

Thus, we see that for the case of the similarity invariant formula, if the entry contributes vote less than z, then the deviation will be at least $c(z) \cdot a$, where $c(z)$ is given above and a is the standard deviation in hash (i.e., feature) space.

3.4.3 Number of features within a region

Suppose that we accumulate a total vote of z (see Equation 3.19); we are interested in knowing how many features lie within a region σ^* of the predicted locations. We will claim that there are n^* features with deviations less than σ^*, where $n^* = n^*(\sigma^*, z)$. In this case, we know that at most $n_k - n^*$ features lie outside the σ^* region. The case that leads to the largest vote is that there are n^* features matched perfectly and $n_k - n^*$ features lie on the boundary of the region. We have the following inequality for the accumulated vote:

$$n^* \cdot \log(1+b) + (n_k - n^*) \log(1 + b \cdot e^{-\frac{\sigma^{*2}}{2a^2}}) \geq z',$$

where

$$z' = z + s \log\left(\frac{s}{s - \beta n_k}\right).$$

That is, at least n_1 features lie within the range σ^*,

$$n^* \cdot \left[\log\left(\frac{1+b}{1 + b \cdot e^{-\frac{\sigma^{*2}}{2a^2}}}\right)\right] \geq z' - n_k \cdot \log(1 + b \cdot e^{-\frac{\sigma^{*2}}{2a^2}}).$$

We have

$$n^* \geq n_1,$$

where

$$n_1 = \left\lfloor \frac{z' - n_k \cdot \log(1 + b \cdot e^{-\frac{\sigma^{*2}}{2a^2}})}{\log\left(\frac{1+b}{1+b \cdot e^{-\frac{\sigma^{*2}}{2a^2}}}\right)} \right\rfloor. \tag{3.21}$$

In other words, we have the following result for estimating the lower bound of the number of features within the specified region:

Result 3.1 Given the parameters n_k, z, and σ^*, there are at least $n_1 = n_1(n_k, z, \sigma^*)$ features that land within a region σ^* of the predicated locations, where n_1 is given in Equation 3.21. □

The function of the lower bound of the number of the features within the specified region as a function of n_k is shown in Figure 3.9.

On the other hand, if we ignore the vote contributed by the entries that has deviation larger than σ^* and assume that n^* features land on the boundary of σ^* region, we have another inequality for the accumulated vote:

$$n^* \cdot \log(1 + b \cdot e^{-\frac{\sigma^{*2}}{2a^2}}) \leq z'.$$

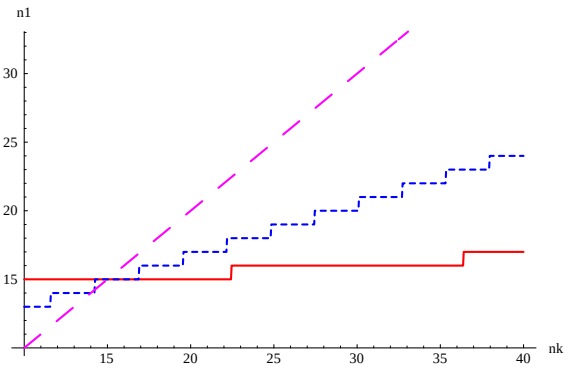

Figure 3.9: The function of the lower bound of the number of the features within the specified region as a function of n_k. Here, $a = 1/10$, the number of scene features $s = 300$, the non-obscuration ratio β is fixed as 0.8. The number of features n_k in the model k varies from 10 to 40. The dotted line indicates the lower bound for the region $\sigma^* = 1$, while the solid line indicates the lower bound for the region $\sigma^* = 0.2$. The standard deviation in the original image space σ is 4 pixels. We assume that the accumulated vote z is 20. The dashed line is the function $n_1 = n_k$

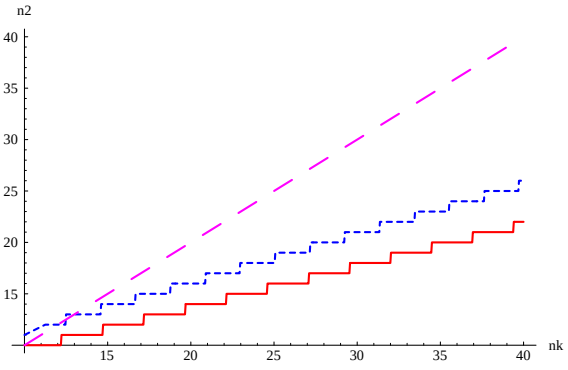

Figure 3.10: The function of the upper bound of the number of the features within the specified region as a function of n_k. Again, $a = 1/10$, the number of scene features $s = 300$, the non-obscuration ratio β is fixed as 0.8. The number of features n_k in the model k varies from 10 to 40. The dotted line indicates the lower bound for the region $\sigma^* = 0.1$, while the solid line indicates the lower bound for the region $\sigma^* = 0.05$. The standard deviation in the original image space σ is 4 pixels. We assume that the accumulated vote z is 10. The dashed line is the function $n_2 = n_k$.

Thus we have

$$n^* \leq n_2,$$

where

$$n_2 = \left\lceil \frac{z'}{\log(1 + b \cdot e^{-\frac{\sigma^{*2}}{2a^2}})} \right\rceil . \tag{3.22}$$

In other words, we have the following result for estimating the upper bound of the number of features within the specified region:

Result 3.2 Given the parameters z, and σ^*, there are no more than $n_2 = n_2(z, \sigma^*)$ features within the range σ^* of predicted locations, where n_2 is given in Equation 3.22. □

The function of the upper bound of the number of the features within the specified region as a function of n_k is shown in Figure 3.10.

Thus, given n_k, z, and σ^*, we can use Equation 3.21 and 3.22 to find integer solutions for n^* to estimate the number of features within the region σ^*.

3.4.4 Estimation of accumulated vote

Let z_t be the accumulated total vote without the bias term $-s\log\left(\frac{s}{s-\beta n_k}\right)$ (see Equation 3.19).

Result 3.3 Suppose there are at least n_1 features that lie within a region of size σ^*. The total vote must be greater than the vote when n_1 features lie on the σ^* boundary. If we ignore all other votes, we see that

$$z_t \geq n_1 \log(1 + b \cdot e^{-\frac{\sigma^{*2}}{2a^2}}). \tag{3.23}$$

□

Result 3.4 Suppose that there are no more than n_2 features lie within the range σ^* of the predicted locations, then the total accumulated vote is no more than z_2, where z_2 is defined as:

$$z_2 = n_2 \cdot \log(1 + b) + (n_k - n_2)\log(1 + b \cdot e^{-\frac{\sigma^{*2}}{2a^2}}). \tag{3.24}$$

□

Given the range of the number of features within the region σ^*, we can use Equations 3.23 and 3.24 to estimate the range of total accumulated vote.

3.5 Larger Vote Under Rearrangement

We described the algorithm for geometric hashing at the beginning of this chapter. Although we did not express it explicitly, we indeed use the following rules in the implementation:

1. For a feature in the scene, at most one entry in a filter may accumulate a vote, namely the closest one.

2. A feature in the scene may contribute to several hash entries if they are in different filters.

3. For each hash entry, at most one feature in the scene may contribute a vote to that entry.

As shown in Figure 3.1, a filter is constructed by the normalized features for a specified basis from a model. We do not allow that a scene feature contributes more than one entry if the entries are from the same filter, otherwise, a scene feature is associated with many entries based on the same hypothesis, which can not be true in the real case.

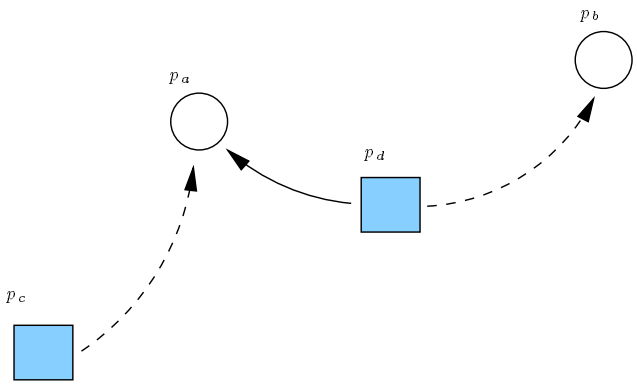

Figure 3.11: Rearrangement of assignment may produce a better result. The features p_a and p_b are two hash entries, and p_c and p_d are two normalized features. Distance relations are $||\overline{p_a p_d}|| < ||\overline{p_b p_d}||$ and $||\overline{p_a p_d}|| < ||\overline{p_a p_c}||$. According to our rule, p_d contributes to p_a, even though rearrangement of assignment such that p_c is associated with p_a and p_d is associated with p_b can contribute more vote to this hypothesis.

In Rule 1, we choose the closest entry in order to accumulate the highest vote. This is our heuristic which is quite natural when we are dealing with the features in the scene one by one. We do not intend to find all combinations of assignment in order to find the best assignment, due to computational consideration.

However, sometimes this heuristic will not work. Let us consider the following scenario. As shown in Figure 3.11, suppose p_a and p_b are two entries in a filter, p_c and p_d represent two normalized features in the scene. Distance $||\overline{p_a p_d}||$ is smaller than distance $||\overline{p_a p_c}||$, and distance $||\overline{p_a p_d}||$ is smaller than distance $||\overline{p_b p_d}||$. According to our rule, p_d will be associated with p_a as shown in the arrow sign in the figure. The normalized feature p_c will not contribute a vote in this occasion since p_a is already occupied. Entry p_b will not receive a vote since when we compute distance at p_c and p_d, the closest entry is entry p_a. If we rearrange the assignment such that p_c is assigned to p_a and p_d is assigned to p_b as shown in the dash arrow in Figure 3.11, then both normalized features p_c and p_d can contribute vote to this hypothesis. It is possible that this new arrangement can accumulate more vote for this hypothesis. This new arrangement could be quite reasonable since the position of basis may be inaccurate so that the new arrangement should be the correct assignment. Even though we face this kind of difficulty, we argue that this kind of situation does not occur very often.

Chapter 4

Attributed Features

In Chapter 3, we discussed the fundamental theory for geometric hashing. The formulation given there is based upon matching of 2-D point patterns. When the object is embedded in a complicated background, the recognition system can easily fail due to false alarms. The reason is that when the background is complicated, feature points behave like random dots, and it is easy to get a satisfiable match. As shown in Figure 4.1, even for this simple example, the recognition result is incorrect. Figure 4.2 shows another example of incorrect matching for real image. The problem occurs owing to insufficient discrimination power of the feature points with positional information only. A close examination of the examples shown in Figure 4.1 and Figure 4.2 suggests that we may attach more information to the point features in order to increase the discrimination power. A possible choice is the orientation of the line segments. Thus, the features are *attributed*. We posit that by using this kind of higher-order features, we may reduce the likelihood of accidental matches. Orientation information is simply one realization of attributed features; we may extend the theory to other kinds of attributes if the attributes are invariant. *Invariant* means that the value is preserved under transformation from a specified class of transformations.

In the following sections, we will describe a particular formulation of a geometric hashing system using point features and orientation attributes. This system will be used with realistic images to find objects that contain many line segment features. The density function for the weighted voting formula of our orientation-attributed features is addressed in this chapter. We conclude this chapter by discussing abstraction of the attributed features.

4.1 Formulation for Orientation-Attributed Features

Suppose a single feature consists of a point location (x, y) and an orientation θ. We name the features to be *attributed features* since positional information (x, y), which is

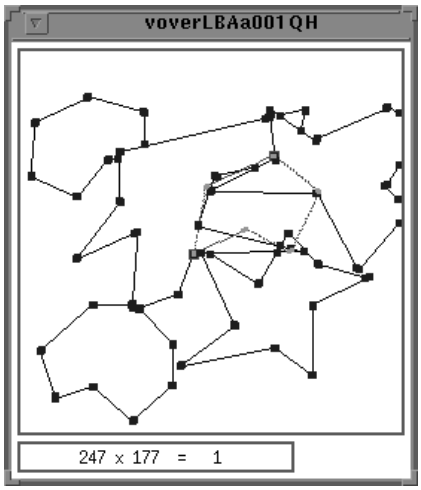

Figure 4.1: A false alarm for the recognition of polygon. The positions of endpoints for the extracted line segments are used as the features. Without the orientation information, an incorrect model can match to incorrect position easily.

Figure 4.2: A false alarm for the recognition of Buick LeSabre. The positions of midpoints for the extracted line segments are used as the features. Eighteen midpoints out of 27 model features (from the model of Honda Prelude) match the scene features in this example.

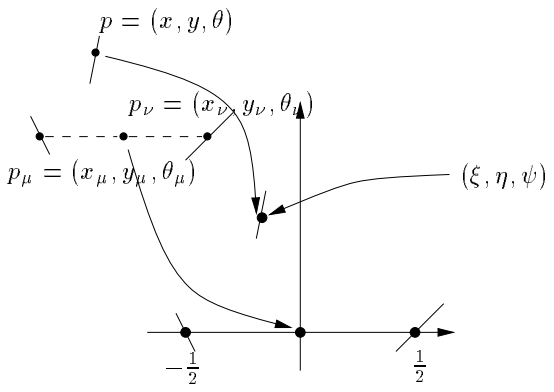

Figure 4.3: The construction of the hash function, $h(p_\mu, p_\nu, p)$.

sufficient to define the transformation as described in Chapter 3, has been supplemented. The orientation information θ is used as the attribute that provides more information about the features. Other kinds of attributes are possible; we will discuss this issue later on. Orientations are not directed, so $0 \leq \theta < \pi$. In practice, θ is stored as a cosine/sine pair to avoid trigonometric evaluations.

Following the same notation as in Chapter 3, we assume that T is a similarity transformation. Although T is a transformation of point features, it operates on attributed features (x, y, θ) by transforming (x, y) to $T(x, y)$, and rotating θ to $\theta + \phi$, where ϕ is the rotation induced by the similarity transformation T. This is because a line through the point (x, y) having orientation θ will be transformed under T to a line through $T(x, y)$ having orientation $\theta + \phi$. The new orientation should be regarded as being modulo π, so that

$$T(x, y, \theta) = (T(x, y), \theta + \phi \bmod \pi).$$

Figure (4.3) displays graphically the definition of the hash function h. Note that the hash function $h(p_\mu, p_\nu, p)$ ignores the information in θ_μ and θ_ν.

Now, for a model to be matched, not only will all locations of the features match, the orientation information should also match in order for the instance to be a valid embedding. In fact, even the orientation information of the transformed basis points should match.

Compared to the formulation described in Chapter 3, an attribute is used as an additional constraint for feature points to be matched. We can expect that the discrimination power of attributed features will be increased. Under this formulation, this realization of attributed features can fit into our geometric hashing paradigm naturally. If we assume

that attributed features are also Gaussian distributed, we can use the weighted voting formula described in Chapter 3 directly (see Equation 3.7). In the presence of noise, there will be variability in the positions of the normalized features in the hash space. We will assume that individual features in the scene domain are subject to Gaussian perturbations. Thus a position of feature (x, y) belonging to a model in the scene can be perturbed by a distance in the Euclidean plane with standard deviation σ, and the orientation θ can be rotated by an angle whose standard deviation is τ radians. In practice, we measure the angle deviation using the sine of the angle difference, rather than the actual angle difference. For small angular perturbations, the two are the same.

To use the Bayesian framework, we need the assumption of independence of the perturbations. The validity of this assumption will depend on the models and choice of the features. Independence means the following: Under the assumption that a particular model m_k is embedded in the scene, and under the assumption that basis pair p_μ and p_ν in m_k match a particular basis of scene points, say p_1 and p_2, then the joint density distribution function in multiple (x, y, θ)-space for any collection of t features, $t \leq s$, is simply the product of t density distribution functions in (x, y, θ)-space for single features. On the other hand, if the orientation information is also independent of the positional information for the attributed features, the Gaussian density function is separable which will simplify the computation for the weighted voting formula. We discuss the validity of the independence assumption for various feature representations (using orientation-attributed features) in the next section.

Using the above assumptions and using Equation 3.7 for uniform background, we have the following weighted voting formula for orientation-attributed features:

$$-s \log\left(\frac{s}{s - \beta n_k}\right) + \tag{4.1}$$
$$\sum_{j=1}^{s} \log\left(1 + \frac{\beta \cdot |R|}{(2\pi)^{\frac{3}{2}}\sigma^2\tau \cdot (s - \beta n_k)} \exp\left(-\frac{||p_2 - p_1||^2 \cdot ||\vec{u_j} - \vec{x_{i_j}}||^2}{2\sigma^2} - \frac{\sin^2(\phi - \theta)}{2\tau^2}\right)\right),$$

where p_1 and p_2 are the selected basis from the scene, $\vec{x_{i_j}}$ is the nearby hash entry with angle θ associated as its attribute, $\vec{u_j}$ is a normalized feature of a scene configuration with angle ϕ attached as its attribute.

Tsai [108, 107] uses a different approach for the recognition problem of 2-D flat objects with line features. The line features are encoded as (r, θ) pairs. He derives the close-form formula for the density function for the affine invariants when the (r, θ) parameters are perturbed by Gaussian noise. Several experiments with flat 2-D objects cluttered with a noisy background are reported. However, by representing lines by two parameters, line equations are represented as lines, and two line segments will match if they lie on the same line.

4.2 Orientation-Attributed Features

We discuss three cases for orientation-attributed features that are used in our experiments. We assume that the orientation information is Gaussianly perturbed and is independent of the positional information for attributed features in Equation 4.2. That is, the conditional density function is assumed to be separable. The covariance matrix without this independence assumption for each case will be discussed in this section also.

We have three ways to extract the orientation-attributed features:

1. Endpoints of the line segments as the attributed point features, with a separate entry for each oriented segment emanating from the point;

2. Midpoints of the line segments as the attributed point features, with a single entry using the line orientation as the attribute;

3. Bisectors of corners formed by pairs of line segments as the attributed point features, with a single entry located at the intersection of two lines and using the bisector of the angle formed by the two lines as the attribute.

In the case of method (1), we do not have the property of independence of the features. We ignore the lack of independence at this moment. In the case of method (2), attributed features are independent of each other intuitively, since given the information of one midpoint, we cannot predict the appearance of the other midpoints. Furthermore, we can show that the orientation information is independent of the positional information, which means that the probability density function is indeed separable. The proof is given in Section 4.4 and Appendix A. Another advantage of midpoint features over endpoints is that the number of features is reduced by one-half, since each line segment only has one hash entry instead of two hash entries. This means that the size of the hash table is reduced to one-eighth of the size using endpoints [69]. The computation time is also affected accordingly. Intuitively, in the case of method (3), attributed features are independent of each other. Again, one appearance of a bisector cannot predict the other bisectors. We do not use the opening of the corner at this moment. We can use this information as another attribute, however. Figure 4.4 shows these three methods of obtaining orientation-attributed features.

When using attributed point features, there is extra information attached to the basis points. To form a hypothesis, we only make use of the positional information of the selected basis. We may use the orientation information of the selected basis to filter out impossible hypotheses. In any case, when the basis points participate the evidence accumulations, orientation mismatches will also penalize the total accumulation.

We next turn to the derivations of covariance matrices for each case by assuming first order approximation and a Gaussian model.

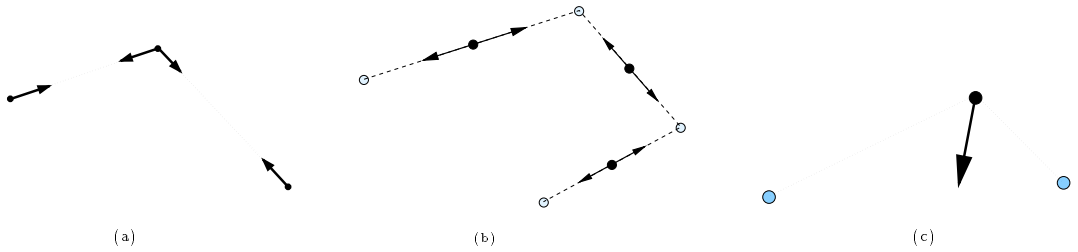

Figure 4.4: The three methods of obtaining orientation-attributed features. (a) Endpoints of line segments as the attributed features. (b) Midpoints of line segments as the attributed features. (c) Bisectors of the corners formed by pairs of line segments as the attributed features.

4.3 Derivations of the Covariance Matrices

We want to compute the conditional density function for attributed features. A Gaussian noise model for the positional information is assumed. We also assume that the orientation information of the attributed features is derived from the three methods that we described in the previous sections. Similar to the derivations we discussed in Section 3.3, the basic tool we use to derive the covariance matrix is Corollary 3.2 and the first order approximation (see Equation 3.9).

Furthermore, to avoid trigonometric computations, we use $\sin(\Delta\theta)$ instead of $\Delta\theta$ to compute the deviation of the orientation invariant θ. For small angular perturbations, the two are the same. Suppose the orientation invariant θ is a function of vector $\mathbf{x}$, where vector $\mathbf{x}$ is the coordinate of the attributed features. In order to use Corollary 3.2 to compute the covariance matrices, we are required to compute $\nabla\theta(\mathbf{x})$ and express it as the linear combination of x_i. We have

$$\nabla\theta(\mathbf{x}) = \frac{1}{\cos\theta} \cdot \nabla \sin(\theta(\mathbf{x})).$$

Suppose points p_1 and p_2 are the selected basis. Points p_3 and p_4 form a line segment. The orientation of $p_1\vec{p}_2$ and $p_3\vec{p}_4$ is θ_b and θ_3 respectively (see Figure 4.5). The sine of the orientation invariant is:

$$\sin(\theta_3 - \theta_b) = s_{3b} = s_3 c_b - c_3 s_b,$$

where

$$\begin{bmatrix} c_b \\ s_b \end{bmatrix} = \begin{bmatrix} \xi_{21}/N_{21} \\ \eta_{21}/N_{21} \end{bmatrix},$$

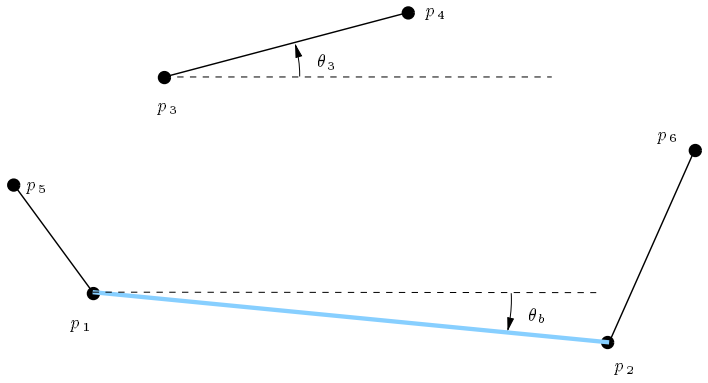

Figure 4.5: The configuration of basis pair (p_1, p_2) and line segment $\overline{p_3 p_4}$ for endpoint feature.

and

$$\begin{bmatrix} c_3 \\ s_3 \end{bmatrix} = \begin{bmatrix} \xi_{43}/N_{43} \\ \eta_{43}/N_{43} \end{bmatrix}.$$

Here, we denote $\sin(\theta_k)$ as s_k and $\cos(\theta_k)$ as c_k for convinence. We also denote $\theta_i - \theta_b$ as θ_{ib}, where $i = 1, 2, \cdots$. Recall that for ξ_{ij} and η_{ij} are defined in Equation 3.11 as

$$\begin{aligned} \xi_{ij} &= x_i - x_j, \\ \eta_{ij} &= y_i - y_j. \end{aligned}$$

We also define N_{ij} as

$$N_{ij} = \text{dist}(p_i, p_j) = ||(x_i, y_i) - (x_j, y_j)|| = \sqrt{\xi_{ij}^2 + \eta_{ij}^2}.$$

In the following sections, we discuss the case for similarity transformation only. Recall that the order of the random variables are organized as $[x_1, y_1, x_2, y_2, \ldots, x_n, y_n]$, where (x_i, y_i) is the coordinate for point p_i. We also assume that the variances for each random variable x_i and y_i are independent of each other with standard deviation σ.

4.3.1 Endpoints under the approximate matching hypothesis

Let us ignore the orientation information of basis pair p_1 and p_2 for the moment. For the case of endpoint analysis, suppose (u, v) is the positional invariant of p_3, and s_{3b} is the

orientation invariant of p_3, then the coefficient matrix for first order approximation the under similarity transformation is

$$\begin{aligned}
W_1 &= \frac{1}{N_{21}^2}[\xi_{21}(u-\frac{1}{2})+\eta_{21}v, -\xi_{21}v+\eta_{21}(u-\frac{1}{2}),\\
&\quad -\xi_{21}(u+\frac{1}{2})-\eta_{21}v, \xi_{21}v-\eta_{21}(u+\frac{1}{2}),\\
&\quad \xi_{21},\ \ \eta_{21},\ \ 0,\ \ 0],\\
W_2 &= \frac{1}{N_{21}^2}[-\eta_{21}(u-\frac{1}{2})+\xi_{21}v, \eta_{21}v+\xi_{21}(u-\frac{1}{2}),\\
&\quad \eta_{21}(u+\frac{1}{2})-\xi_{21}v, -\eta_{21}v-\xi_{21}(u+\frac{1}{2}),\\
&\quad -\eta_{21},\ \ \xi_{21},\ \ 0,\ \ 0],\\
W_3 &= \frac{1}{N_{21}N_{43}}[-N_{43}s_b,\ \ N_{43}c_b,\ \ N_{43}s_b,\ \ -N_{43}c_b,\\
&\quad N_{21}s_3,\ \ -N_{21}c_3,\ \ -N_{21}s_3,\ \ N_{21}c_3],
\end{aligned} \tag{4.2}$$

where row vector W_i is i-th row vector of matrix W defined in Corollary 3.2. The covariance matrix for random vector $[u, v, \theta_{3b}]$ is $\hat{\Sigma}$:

$$\begin{aligned}
\hat{\Sigma} &= \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} \Sigma\ [W_1^T, W_2^T, W_3^T]\\
&= \frac{\sigma^2}{N_{21}^2}\begin{bmatrix} \frac{4(u^2+v^2)+3}{2} & 0 & -2v+\frac{N_{21}}{N_{43}}s_{3b} \\ 0 & \frac{4(u^2+v^2)+3}{2} & 2u-\frac{N_{21}}{N_{43}}c_{3b} \\ -2v+\frac{N_{21}}{N_{43}}s_{3b} & 2u-\frac{N_{21}}{N_{43}}c_{3b} & 2+\frac{2N_{21}}{N_{43}^2} \end{bmatrix},
\end{aligned} \tag{4.3}$$

where Σ is the covariance matrix of random vector $[x_1, y_1, \ldots, x_n, y_n]$, i.e., $\sigma^2 \cdot I_{8\times 8}$.

Now let us consider the statistics of the orientation attributes associated with the basis points, i.e., we compute the covariance matrix for random vector $[u, v, \theta_{3b}, \theta_{1b}, \theta_{2b}]$. The dimension of the covariance matrix becomes 5 by 5. We introduce other feature points p_5 and p_6 as shown in Figure 4.5; vector $p_1\vec{p}_5$ and vector $p_2\vec{p}_6$ give the orientation information for the basis p_1 and p_2 respectively. That is, two extra row vectors W_4 and W_5 are:

$$\begin{aligned}
W_4 &= \frac{1}{c_{1b}}[c_b\cdot\nabla s_1 - s_b\cdot\nabla c_1 + s_1\cdot\nabla c_b - c_1\cdot\nabla s_b]\\
&= [\frac{s_1}{N_{51}}-\frac{s_b}{N_{21}},\ \ -\frac{c_1}{N_{51}}+\frac{c_b}{N_{21}},\ \ \frac{s_b}{N_{21}},\ \ -\frac{c_b}{N_{21}},\ \ 0,\ \ 0,\ \ 0,\ \ 0,\ \ -\frac{s_1}{N_{51}},\ \ \frac{c_1}{N_{51}},\ \ 0,\ \ 0]\\
W_5 &= \frac{1}{c_{2b}}[c_b\cdot\nabla s_2 - s_b\cdot\nabla c_2 + s_2\cdot\nabla c_b - c_2\cdot\nabla s_b]\\
&= [-\frac{s_b}{N_{21}},\ \ \frac{c_b}{N_{21}},\ \ \frac{s_2}{N_{62}}+\frac{s_b}{N_{21}},\ \ -\frac{c_2}{N_{62}}-\frac{c_b}{N_{21}},\ \ 0,\ \ 0,
\end{aligned}$$

$$0,\ \ 0,\ \ 0,\ \ 0,\ \ -\frac{s_2}{N_{62}},\ \ \frac{c_2}{N_{62}}]. \tag{4.4}$$

The vectors W_1, W_2, and W_3 become 1 by 10 row matrices with 0's appended at the end. We have the following result:

$$\begin{aligned}
\hat{\Sigma}_{14} &= \hat{\Sigma}_{41} = \frac{\sigma^2}{N_{21}^2}\left(-2v + \frac{N_{21}(s_{1b}(u-\frac{1}{2}) + c_{1b}v)}{N_{51}}\right), \\
\hat{\Sigma}_{15} &= \hat{\Sigma}_{51} = \frac{\sigma^2}{N_{21}^2}\left(-2v - \frac{N_{21}(s_{2b}(u+\frac{1}{2}) + c_{2b}v)}{N_{62}}\right), \\
\hat{\Sigma}_{24} &= \hat{\Sigma}_{42} = \frac{\sigma^2}{N_{21}^2}\left(2u - \frac{N_{21}(c_{1b}(u-\frac{1}{2}) - s_{1b}v)}{N_{51}}\right), \\
\hat{\Sigma}_{25} &= \hat{\Sigma}_{52} = \frac{\sigma^2}{N_{21}^2}\left(2u + \frac{N_{21}(c_{2b}(u+\frac{1}{2}) - s_{2b}v)}{N_{62}}\right), \\
\hat{\Sigma}_{34} &= \hat{\Sigma}_{43} = \frac{\sigma^2}{N_{21}^2}\left(2 - \frac{N_{21}c_{1b}}{N_{51}}\right), \\
\hat{\Sigma}_{35} &= \hat{\Sigma}_{53} = \frac{\sigma^2}{N_{21}^2}\left(2 + \frac{N_{21}c_{2b}}{N_{62}}\right), \\
\hat{\Sigma}_{44} &= \frac{2\sigma^2}{N_{21}^2}\left(1 + \frac{N_{21}^2}{N_{51}^2} - \frac{N_{21}c_{1b}}{N_{51}}\right), \\
\hat{\Sigma}_{45} &= \hat{\Sigma}_{54} = \frac{\sigma^2}{N_{21}^2}\left(2 - \frac{N_{21}c_{1b}}{N_{51}} + \frac{N_{21}c_{2b}}{N_{62}}\right), \\
\hat{\Sigma}_{55} &= \frac{2\sigma^2}{N_{21}^2}\left(1 + \frac{N_{21}^2}{N_{62}^2} + \frac{N_{21}c_{2b}}{N_{62}}\right),
\end{aligned}$$

where $\hat{\Sigma}_{ij}$ is the (i,j) element of the covariance matrix $\hat{\Sigma}$. The other elements of this 5×5 covariance matrix are given in Equation 4.3 as its 3×3 submatrix.

4.3.2 Endpoints under the exact matching hypothesis

We derive 5×5 and 3×3 covariance matrices for endpoint features under the exact matching hypothesis. Exact matching hypothesis means the basis pair (p_1, p_2) can be matched exactly, i.e., there are no deviations for the position of basis pair (p_1, p_2).

We use the Equation 4.2 and 4.4 and discard the elements that correspond to random variables x_1, y_1, x_2, and y_2. The 5×5 covariance matrix for exact matching hypothesis

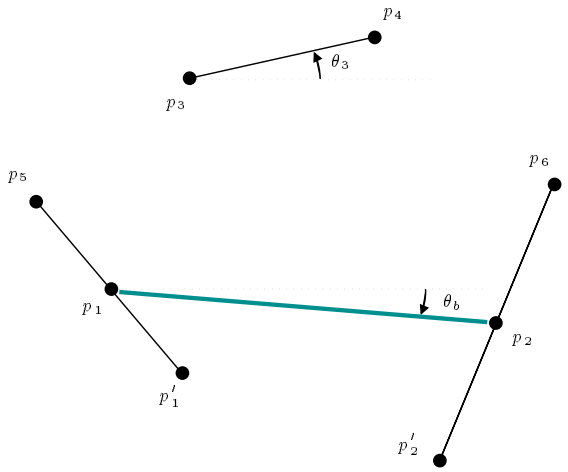

Figure 4.6: The configuration of basis pair (p_1, p_2) and line segment $\overline{p_3p_4}$ for midpoint feature.

is:

$$\hat{\Sigma} = \sigma^2 \begin{bmatrix} \frac{1}{N_{21}^2} & 0 & \frac{s_{3b}}{N_{21}N_{43}} & 0 & 0 \\ 0 & \frac{1}{N_{21}^2} & -\frac{c_{3b}}{N_{21}N_{43}} & 0 & 0 \\ \frac{s_{3b}}{N_{21}N_{43}} & -\frac{c_{3b}}{N_{21}N_{43}} & \frac{2}{N_{43}^2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{N_{51}^2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{N_{62}^2}. \end{bmatrix}. \tag{4.5}$$

The 3×3 covariance matrix for this case is simply the upper-left three by three submatrix in Equation 4.5.

4.3.3 Midpoints under the approximate matching hypothesis

For the case of midpoint analysis, suppose (u, v) is the positional invariant of the midpoint for line segment $\overline{p_3p_4}$, and the basis points (p_1, p_2) are the midpoints of the other two line segments $\overline{p_1'p_5}$ and $\overline{p_2'p_6}$ respectively. We organize the random variables before transformation as $[x_1', y_1', x_2', y_2', x_3, y_3, \ldots, x_6, y_6]$. The configuration is shown in Figure 4.6. We have W_1, W_2, and W_3 as

$$W_1 = \frac{1}{2N_{21}^2}[\xi_{21}(u - \frac{1}{2}) + \eta_{21}v, \quad -\xi_{21}v + \eta_{21}(u - \frac{1}{2}),$$

$$
\begin{aligned}
& -\xi_{21}(u+\frac{1}{2})-\eta_{21}v, \quad \xi_{21}v-\eta_{21}(u+\frac{1}{2}), \\
& \xi_{21}, \quad \eta_{21}, \quad \xi_{21}, \quad \eta_{21}, \\
& \xi_{21}(u-\frac{1}{2})+\eta_{21}v, \quad -\xi_{21}v+\eta_{21}(u-\frac{1}{2}), \\
& -\xi_{21}(u+\frac{1}{2})-\eta_{21}v, \quad \xi_{21}v-\eta_{21}(u+\frac{1}{2})], \\
W_2 \;=\; & \frac{1}{2N_{21}^2}[-\eta_{21}(u-\frac{1}{2})+\xi_{21}v, \quad \eta_{21}v+\xi_{21}(u-\frac{1}{2}), \\
& \eta_{21}(u+\frac{1}{2})-\xi_{21}v, \quad -\eta_{21}v-\xi_{21}(u+\frac{1}{2}), \\
& -\eta_{21}, \quad \xi_{21}, \quad -\eta_{21}, \quad \xi_{21}, \\
& -\eta_{21}(u-\frac{1}{2})+\xi_{21}v, \quad \eta_{21}v+\xi_{21}(u-\frac{1}{2}), \\
& \eta_{21}(u+\frac{1}{2})-\xi_{21}v, \quad -\eta_{21}v-\xi_{21}(u+\frac{1}{2})], \\
W_3 \;=\; & \frac{1}{N_{21}N_{43}}[-\frac{N_{43}s_b}{2}, \quad \frac{N_{43}c_b}{2}, \quad \frac{N_{43}s_b}{2}, \quad -\frac{N_{43}c_b}{2}, \quad N_{21}s_3, \quad -N_{21}c_3, \\
& -N_{21}s_3, \quad N_{21}c_3, \quad -\frac{N_{43}s_b}{2}, \quad \frac{N_{43}c_b}{2}, \quad \frac{N_{43}s_b}{2}, \quad -\frac{N_{43}c_b}{2}]. \qquad (4.6)
\end{aligned}
$$

Note that the variances of x and y component for points p_1 and p_2 become $\sigma^2/2$, which are different from the variations of the endpoints of a line segment in the previous section. The computed covariance matrix for $[u, v, \theta_{3b}]$ is:

$$
\begin{aligned}
\hat{\Sigma} \;&=\; \begin{bmatrix} W_1 \\ W_2 \\ W_3 \end{bmatrix} \Sigma \; [W_1^T, W_2^T, W_3^T] \\
&=\; \frac{\sigma^2}{N_{21}^2} \begin{bmatrix} \frac{4(u^2+v^2)+3}{4} & 0 & -v \\ 0 & \frac{4(u^2+v^2)+3}{4} & u \\ -v & u & 1+\frac{2N_{21}^2}{N_{43}^2} \end{bmatrix}. \qquad (4.7)
\end{aligned}
$$

Again, if we consider the orientation attributes for the basis pair, we need to compute W_4 and W_5. They are given as

$$
\begin{aligned}
W_4 \;&=\; \frac{1}{c_{1b}}[c_b \cdot \frac{ds_1}{dx} - s_b \cdot \frac{dc_1}{dx} + s_1 \cdot \frac{dc_b}{dx} - c_1 \cdot \frac{ds_b}{dx}] \\
&=\; \frac{1}{2}[\frac{s_1}{N_{51}} - \frac{s_b}{N_{21}}, \quad -\frac{c_1}{N_{51}} + \frac{c_b}{N_{21}}, \quad \frac{s_b}{N_{21}}, \quad -\frac{c_b}{N_{21}}, \quad 0, \quad 0, \quad 0, \quad 0, \\
& \quad -\frac{s_1}{N_{51}} - \frac{s_b}{N_{21}}, \quad \frac{c_1}{N_{51}} + \frac{c_b}{N_{21}}, \quad \frac{s_b}{N_{21}}, \quad -\frac{c_b}{N_{21}}] \\
W_5 \;&=\; \frac{1}{c_{2b}}[c_b \cdot \frac{ds_2}{dx} - s_b \cdot \frac{dc_2}{dx} + s_2 \cdot \frac{dc_b}{dx} - c_2 \cdot \frac{ds_b}{dx}]
\end{aligned}
$$

$$= \frac{1}{2}[-\frac{s_b}{N_{21}},\ \frac{c_b}{N_{21}},\ \frac{s_2}{N_{62}}+\frac{s_b}{N_{21}},\ -\frac{c_2}{N_{62}}-\frac{c_b}{N_{21}},\ 0,\ 0,\ 0,\ 0,$$
$$-\frac{s_b}{N_{21}},\ \frac{c_b}{N_{21}},\ -\frac{s_2}{N_{62}}+\frac{s_b}{N_{21}},\ \frac{c_2}{N_{62}}-\frac{c_b}{N_{21}}]. \quad (4.8)$$

The 5×5 covariance matrix for random vector $[u,\ v,\ \theta_{3b},\ \theta_{1b},\ \theta_{2b}]$ is:

$$\hat{\Sigma} = \frac{\sigma^2}{N_{21}^2}\begin{bmatrix} \frac{4(u^2+v^2)+3}{4} & 0 & -v & -v & -v \\ 0 & \frac{4(u^2+v^2)+3}{4} & u & u & u \\ -v & u & 1+\frac{2N_{21}^2}{N_{43}^2} & 1 & 1 \\ -v & u & 1 & 1+\frac{N_{21}^2}{2N_{51}^2} & 1 \\ -v & u & 1 & 1 & 1+\frac{N_{21}^2}{2N_{62}^2} \end{bmatrix}.$$

Unlike the case of endpoints, the basis pair (p_1, p_2) is dependent upon p_5 and p_6 for midpoint case. Because of the symmetry of the midpoint representation, the covariance matrices are simpler than that of the endpoints case.

4.3.4 Midpoints under the exact matching hypothesis

The case for midpoint features under exact matching hypothesis is different from the case of endpoint features as described in Section 4.3.2. The reason is that under exact matching hypothesis, the basis points p_1 and p_2 are matched to the selected basis points in the test image. Since p_1 and p_2 are the midpoints of line segments $\overline{p_1' p_5}$ and $\overline{p_2' p_6}$ respectively (see Figure 4.6), the endpoints of these two line segments are not independent. However, when the midpoints p_1 and p_2 are fixed, we can determine the location of p_1' and p_2' if the Gaussianly perturbed points p_5 and p_6 are given. We can again discard the terms corresponding to random variables x_1', y_1', x_2', and y_2' in $W_1, \ldots, W_5$ which are described in Equation 4.6 and 4.8. Furthermore, since the basis pair (p_1, p_2) is fixed now, the deviation of points p_5 and p_6 will affect the orientation attributes of the basis, i.e, θ_{1b} and θ_{2b} only. The resulting W vectors for random vector $[x_3, y_3, \ldots x_6, y_6]$ become

$$W_1 = \frac{1}{2N_{21}^2}[\xi_{21},\ \eta_{21},\ \xi_{21},\ \eta_{21},\ 0,\ 0,\ 0,\ 0],$$
$$W_2 = \frac{1}{2N_{21}^2}[-\eta_{21},\ \xi_{21},\ -\eta_{21},\ \xi_{21},\ 0,\ 0,\ 0,\ 0],$$
$$W_3 = \frac{1}{N_{43}}[s_3,\ -c_3,\ -s_3,\ c_3,\ 0,\ 0,\ 0,\ 0],$$
$$W_4 = \frac{1}{N_{51}}[0,\ 0,\ 0,\ 0,\ -s_1,\ c_1,\ 0,\ 0],$$
$$W_5 = \frac{1}{N_{62}}[0,\ 0,\ 0,\ 0,\ 0,\ 0,\ -s_2,\ c_2].$$

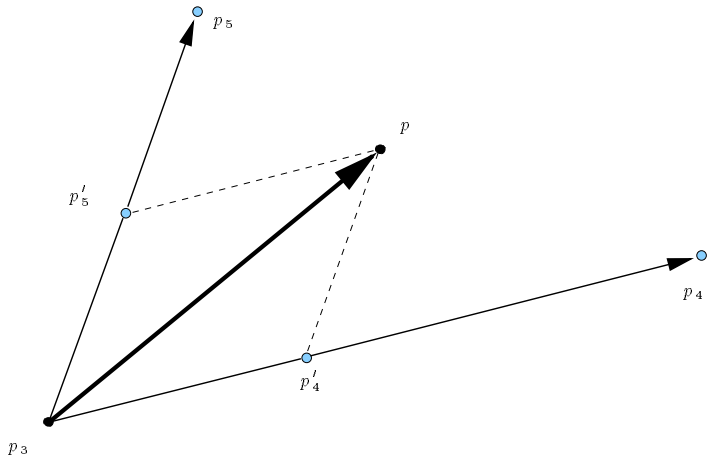

Figure 4.7: The configuration of a corner formed by p_3, p_4, and p_5.

The resulting 5×5 covariance matrix becomes:

$$\hat{\Sigma} = \sigma^2 \begin{bmatrix} \frac{1}{2N_{21}^2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2N_{21}^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{N_{43}^2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{N_{51}^2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{N_{62}^2} \end{bmatrix}. \tag{4.9}$$

Again, the 3×3 covariance matrix is simply the three by three submatrix of Equation 4.9. The covariance matrix shows that the covariance is smaller if the line segment is longer, which means the information is more stable. Note that the covariance for the positional information is dependent upon the length of the basis, which is independent of the hash location. While the covariance for the orientation information is dependent upon each hash location.

4.3.5 Bisectors under the approximate matching hypothesis

We next derive the covariance matrix for bisector case by introducing another two points p_4 and p_5. The configuration of the corner is shown in Figure (4.7). We normalize vectors $p_3\vec{p}_4$ and $p_3\vec{p}_5$ to unit vectors $p_3\vec{p}'_4$ and $p_3\vec{p}'_5$ respectively. Then vector $p_3\vec{p} = p_3\vec{p}'_4 + p_3\vec{p}'_5$ bisects the angle $\angle p_4 p_3 p_5$. Let the angle formed by vector $p_3\vec{p}$ be θ_3, then we have

$$[c_3, s_3] = [\frac{N_{53}\xi_{43} + N_{43}\xi_{53}}{N_3}, \frac{N_{53}\eta_{43} + N_{43}\eta_{53}}{N_3}],$$

where

$$N_3 = \sqrt{(N_{53}\xi_{43} + N_{43}\xi_{53})^2 + (N_{53}\eta_{43} + N_{43}\eta_{53})^2}$$

Again, we use the sine of the angle perturbation to approximate the angle perturbation. Following the same procedure as before, we have

$$\begin{aligned}
W_1 &= \frac{1}{N_{21}^2}[\xi_{21}(u-\frac{1}{2}) + \eta_{21}v, \quad -\xi_{21}v + \eta_{21}(u-\frac{1}{2}), \\
&\quad -\xi_{21}(u+\frac{1}{2}) - \eta_{21}v, \quad \xi_{21}v - \eta_{21}(u+\frac{1}{2}), \\
&\quad \xi_{21}, \quad \eta_{21}, \quad 0, \quad 0, \quad 0, \quad 0], \\
W_2 &= \frac{1}{N_{21}^2}[-\eta_{21}(u-\frac{1}{2}) + \xi_{21}v, \quad \eta_{21}v + \xi_{21}(u-\frac{1}{2}), \\
&\quad \eta_{21}(u+\frac{1}{2}) - \xi_{21}v, \quad -\eta_{21}v - \xi_{21}(u+\frac{1}{2}), \\
&\quad -\eta_{21}, \quad \xi_{21}, \quad 0, \quad 0, \quad 0, \quad 0],
\end{aligned}$$

and

$$\begin{aligned}
W_3 &= \Bigg[\frac{-\eta_{21}}{N_{21}^2}, \quad \frac{\xi_{21}}{N_{21}^2}, \quad \frac{\eta_{21}}{N_{21}^2}, \quad \frac{-\xi_{21}}{N_{21}^2}, \\
&\quad \frac{1}{N_3^2}\left((\xi_{53}\eta_{43} - \xi_{43}\eta_{53})\left(\frac{\xi_{43}N_{53}}{N_{43}} - \frac{\xi_{53}N_{43}}{N_{53}}\right) + (N_{43}+N_{53})(\eta_{53}N_{43} + \eta_{43}N_{53})\right), \\
&\quad \frac{1}{N_3^2}\left((\xi_{53}\eta_{43} - \xi_{43}\eta_{53})\left(\frac{\eta_{43}N_{53}}{N_{43}} - \frac{\eta_{53}N_{43}}{N_{53}}\right) - (N_{43}+N_{53})(\xi_{53}N_{43} + \xi_{43}N_{53})\right), \\
&\quad \frac{-1}{N_3^2}\left((\xi_{53}\eta_{43} - \xi_{43}\eta_{53})\frac{\xi_{43}N_{53}}{N_{43}} + N_{53}(\eta_{53}N_{43} + \eta_{43}N_{53})\right), \\
&\quad \frac{-1}{N_3^2}\left((\xi_{53}\eta_{43} - \xi_{43}\eta_{53})\frac{\eta_{43}N_{53}}{N_{43}} - N_{53}(\xi_{53}N_{43} + \xi_{43}N_{53})\right), \\
&\quad \frac{1}{N_3^2}\left((\xi_{53}\eta_{43} - \xi_{43}\eta_{53})\frac{\xi_{53}N_{43}}{N_{53}} - N_{43}(\eta_{53}N_{43} + \eta_{43}N_{53})\right), \\
&\quad \frac{1}{N_3^2}\left((\xi_{53}\eta_{43} - \xi_{43}\eta_{53})\frac{\eta_{53}N_{43}}{N_{53}} + N_{43}(\xi_{53}N_{43} + \xi_{43}N_{53})\right)\Bigg].
\end{aligned}$$

The final result for every element of $\hat{\Sigma}$ is shown as follows:

$$\begin{aligned}
\hat{\Sigma}_{11} &= \hat{\Sigma}_{22} = \frac{\sigma^2(4(u^2+v^2)+3)}{2N_{21}^2}, \\
\hat{\Sigma}_{12} &= \hat{\Sigma}_{21} = 0, \\
\hat{\Sigma}_{13} &= \hat{\Sigma}_{31} = \frac{\sigma^2}{N_{21}^2}\{-2v + \frac{N_{21}(N_{43}+N_{53})s_{3b}}{N_3} +
\end{aligned}$$

$$\frac{(\xi_{53}\eta_{43}-\xi_{43}\eta_{53})}{N_3^2}[\xi_{21}(\frac{\xi_{43}N_{53}}{N_{43}}-\frac{\xi_{53}N_{43}}{N_{53}})+\eta_{21}(\frac{\eta_{43}N_{53}}{N_{43}}-\frac{\eta_{53}N_{43}}{N_{53}})]\}$$

$$= \frac{\sigma^2}{N_{21}^2}\{-2v+\frac{N_{21}(N_{43}+N_{53})s_{3b}}{N_3}+$$
$$\frac{(dx_{53}\eta_{43}-\xi_{43}\eta_{53})N_{21}}{N_3^2}[\frac{N_3(N_{53}-N_{43})c_{3b}}{N_{43}N_{53}}+c_b(\xi_{43}-\xi_{53})+s_b(\eta_{43}-\eta_{53})]\},$$

$$\hat{\Sigma}_{23} = \hat{\Sigma}_{32}=\frac{\sigma^2}{N_{21}^2}\{2u-\frac{N_{21}(N_{43}+N_{53})c_{3b}}{N_3}+$$
$$\frac{(\xi_{53}\eta_{43}-\xi_{43}\eta_{53})}{N_3^2}[\xi_{21}(\frac{\eta_{43}N_{53}}{N_{43}}-\frac{\eta_{53}N_{43}}{N_{53}})-\eta_{21}(\frac{\xi_{43}N_{53}}{N_{43}}-\frac{\xi_{53}N_{43}}{N_{53}})]\}$$

$$= \frac{\sigma^2}{N_{21}^2}\{2u-\frac{N_{21}(N_{43}+N_{53})c_{3b}}{N_3}+$$
$$\frac{(\xi_{53}\eta_{43}-\xi_{43}\eta_{53})N_{21}}{N_3^2}[\frac{N_3(N_{53}-N_{43})s_{3b}}{N_{43}N_{53}}-s_b(\xi_{43}-\xi_{53})+c_b(\eta_{43}-\eta_{53})]\},$$

$$\hat{\Sigma}_{33} = 2\sigma^2\{\frac{1}{N_{21}^2}+\frac{1}{N_3^4}[(N_{43}^2+N_{43}N_{53}+N_{53}^2)N_3^2-$$
$$(\xi_{53}\eta_{43}-\xi_{43}\eta_{53})^2(\xi_{43}\xi_{53}+\eta_{43}\eta_{53}+(N_{43}+N_{53})^2)]\}.$$

We can also derive the covariance matrix for random vector $[u, v, \theta_{3b}, \theta_{1b}, \theta_{2b}]$ by introducing $p_5, \ldots, p_8$ for bisector case. The angles $\angle p_6p_1p_7$ and $\angle p_8p_2p_9$ provide the directional information for the basis p_1 and p_2. Again, we use sine of the angle variations to approximate the angle variations. The result is listed in the following equations:

$$\hat{\Sigma}_{14} = \hat{\Sigma}_{41}=\frac{\sigma^2}{N_{21}^2}\{-2v+\frac{N_{21}(N_{61}+N_{71})(s_{1b}(u-\frac{1}{2})+c_{1b}v)}{N_1}+$$
$$\frac{N_{21}(\xi_{71}\eta_{61}-\xi_{61}\eta_{71})}{N_1}[\frac{(N_{71}-N_{61})(c_{1b}(u-\frac{1}{2})-s_{1b}v)}{N_{61}N_{71}}+$$
$$\frac{1}{N_1}((\xi_{61}-\xi_{71})(c_b(u-\frac{1}{2})+s_bv)+(\eta_{61}-\eta_{71})(s_b(u-\frac{1}{2})-c_bv))]\},$$

$$\hat{\Sigma}_{15} = \hat{\Sigma}_{51}=\frac{\sigma^2}{N_{21}^2}\{-2v-\frac{N_{21}(N_{82}+N_{92})(s_{2b}(u+\frac{1}{2})+c_{2b}v)}{N_2}-$$
$$\frac{N_{21}(\xi_{92}\eta_{82}-\xi_{82}\eta_{92})}{N_2}[\frac{(N_{92}-N_{82})(c_{2b}(u+\frac{1}{2})-s_{2b}v)}{N_{82}N_{92}}+$$
$$\frac{1}{N_2}((\xi_{82}-\xi_{92})(c_b(u+\frac{1}{2})+s_bv)+(\eta_{82}-\eta_{92})(s_b(u+\frac{1}{2})-c_bv))]\},$$

$$\hat{\Sigma}_{24} = \hat{\Sigma}_{42}=\frac{\sigma^2}{N_{21}^2}\{2u-\frac{N_{21}(N_{61}+N_{71})(c_{1b}(u-\frac{1}{2})-s_{1b}v)}{N_1}+$$
$$\frac{N_{21}(\xi_{71}\eta_{61}-\xi_{61}\eta_{71})}{N_1}[\frac{(N_{71}-N_{61})(s_{1b}(u-\frac{1}{2})+c_{1b}v)}{N_{61}N_{71}}-$$

$$\frac{1}{N_1}((\xi_{61}-\xi_{71})(s_b(u-\frac{1}{2})-c_bv)-(\eta_{61}-\eta_{71})(c_b(u-\frac{1}{2})+s_bv))]\},$$

$$\hat{\Sigma}_{25} = \hat{\Sigma}_{52} = \frac{\sigma^2}{N_{21}^2}\{2u+\frac{N_{21}(N_{82}+N_{92})(c_{2b}(u+\frac{1}{2})-s_{2b}v)}{N_2}-$$
$$\frac{N_{21}(\xi_{92}\eta_{82}-\xi_{82}\eta_{92})}{N_2}[\frac{(N_{92}-N_{82})(s_{2b}(u+\frac{1}{2})+c_{2b}v)}{N_{82}N_{92}}-$$
$$\frac{1}{N_2}((\xi_{82}-\xi_{92})(s_b(u+\frac{1}{2})-c_bv)-(\eta_{82}-\eta_{92})(c_b(u+\frac{1}{2})+s_bv))]\},$$

$$\hat{\Sigma}_{34} = \hat{\Sigma}_{43} = \frac{\sigma^2}{N_{21}^2}\{2-\frac{N_{21}(N_{61}+N_{71})c_{1b}}{N_1}+$$
$$\frac{N_{21}(\xi_{71}\eta_{61}-\xi_{61}\eta_{71})}{N_1}[\frac{(N_{71}-N_{61})s_{1b}}{N_{61}N_{71}}-\frac{(\xi_{61}-\xi_{71})s_b-(\eta_{61}-\eta_{71})c_b}{N_1}]\},$$

$$\hat{\Sigma}_{35} = \hat{\Sigma}_{53} = \frac{\sigma^2}{N_{21}^2}\{2+\frac{N_{21}(N_{82}+N_{92})c_{2b}}{N_2}-$$
$$\frac{N_{21}(\xi_{92}\eta_{82}-\xi_{82}\eta_{92})}{N_2}[\frac{(N_{92}-N_{82})s_{2b}}{N_{82}N_{92}}-\frac{(\xi_{82}-\xi_{92})s_b-(\eta_{82}-\eta_{92})c_b}{N_2}]\},$$

$$\hat{\Sigma}_{44} = \frac{2\sigma^2}{N_{21}^2}+\frac{2\sigma^2}{N_{21}N_1^2}[(\xi_{71}\eta_{61}-\xi_{61}\eta_{71})(\frac{N_1(N_{71}-N_{61})s_{1b}}{N_{61}N_{71}}-$$
$$(\xi_{61}-\xi_{71})s_b+(\eta_{61}-\eta_{71})c_b)-N_1(N_{61}+N_{71})c_{1b}]+$$
$$\frac{2\sigma^2}{N_1^4}[N_1^2(N_{61}^2+N_{61}N_{71}+N_{71}^2)+2N_1(\xi_{71}\eta_{61}-\xi_{61}\eta_{71})((\xi_{61}N_{61}-\xi_{71}N_{71})s_1-$$
$$(\eta_{61}N_{61}-\eta_{71}N_{71})c_1)-(\xi_{71}\eta_{61}-\xi_{61}\eta_{71})^2(\xi_{61}\xi_{71}+\eta_{61}\eta_{71}+2N_{61}N_{71})],$$

$$\hat{\Sigma}_{45} = \hat{\Sigma}_{54} = \frac{2\sigma^2}{N_{21}^2}+\frac{\sigma^2}{N_{21}}\{\frac{(\xi_{71}\eta_{61}-\xi_{61}\eta_{71})}{N_1^2}[\frac{N_1(N_{71}-N_{61})s_{1b}}{N_{61}N_{71}}-$$
$$(\xi_{61}-\xi_{71})s_b+(\eta_{61}-\eta_{71})c_b]-\frac{(\xi_{92}\eta_{82}-\xi_{82}\eta_{92})}{N_2^2}[\frac{N_2(N_{92}-N_{82})s_{2b}}{N_{82}N_{92}}-$$
$$(\xi_{82}-\xi_{92})s_b+(\eta_{82}-\eta_{92})c_b]-\frac{(N_{61}+N_{71})c_{1b}}{N_1}+\frac{(N_{82}+N_{92})c_{2b}}{N_2}\},$$

$$\hat{\Sigma}_{55} = \frac{2\sigma^2}{N_{21}^2}-\frac{2\sigma^2}{N_{21}N_2^2}[(\xi_{92}\eta_{82}-\xi_{82}\eta_{92})(\frac{N_2(N_{92}-N_{82})s_{2b}}{N_{82}N_{92}}-$$
$$(\xi_{82}-\xi_{92})s_b+(\eta_{82}-\eta_{92})c_b)-N_2(N_{82}+N_{92})c_{2b}]+$$
$$\frac{2\sigma^2}{N_2^4}[N_2^2(N_{82}^2+N_{82}N_{92}+N_{92}^2)+2N_2(\xi_{92}\eta_{82}-\xi_{82}\eta_{92})((\xi_{82}N_{82}-\xi_{92}N_{92})s_2-$$
$$(\eta_{82}N_{82}-\eta_{92}N_{92})c_2)-(\xi_{92}\eta_{82}-\xi_{82}\eta_{92})^2(\xi_{82}\xi_{92}+\eta_{82}\eta_{92}+2N_{82}N_{92})].$$

We assume that the basis pair and the encoded features are of the same type in the derivation described in previous sections. For the case of heterogeneous feature types and nonuniform variances, the closed form formula for the covariance matrices are compli-

cated. We may resort to empirical solutions.

4.4 Separability of the Density Function for Midpoints

In our formulation, we need several assumptions of independence. For example, we require independence among the features and independence of the positional information and the orientation attribute. Usually, it is difficult to justify these assumptions. The independence property of the positional attribute and orientation attribute for an attributed feature can simplify the computation since the conditional density function becomes separable. The midpoint of a line segment possesses such an independence property under certain assumptions. We provide both an indirect proof and a direct proof. In the proofs, we do not use first order approximations as we did in previous sections. Here, we show an indirect proof of the independence of the positional information and orientation information for midpoint features. The direct proof is given in an Appendix.

We show that the positional attribute and orientation attribute are independent for the midpoint of a line segment. We need the following lemmas, taken from [82], first.

Lemma 4.1 *If two random variables are uncorrelated they are not necessarily independent. However, for normal random variables uncorelatedness is equivalent to independence.*

Lemma 4.2 *If random variables X and Y are independent, then random variables $g(X)$ and $h(Y)$ are also independent.*

Lemma 4.3 *Consider two random variables X and Y such that $E\{X^2\} = E\{Y^2\}$, then $E\{(X+Y)(X-Y)\} = E\{(X^2 - Y^2)\} = 0$. That is, $X+Y$ and $X-Y$ are uncorrelated.*

From Lemma 4.1 and Lemma 4.3, if X and Y are normal random variables with the same variance, then random variables $X + Y$ and $X - Y$ are independent. The random variables $g(X + Y)$ and $h(X - Y)$ are also independent.

The above lemmas can be extended to random vectors. We have the following theorem:

Theorem 4.4 *The positional attribute and orientation attribute for the midpoint of a line segment are independent.*

Proof: Suppose the coordinates for the endpoints of a line segment are $P_1 = (X_1, Y_1)$ and $P_2 = (X_2, Y_2)$ respectively. The X_i and Y_i are Gaussian random variables with the same variance. The positional attribute of the midpoint for that line segment is $(\frac{X_1+X_2}{2}, \frac{Y_1+Y_2}{2})$,

which is a function of $P_1 + P_2$. The orientation attribute of the midpoint for that line segment is $\tan^{-1} \frac{(Y_2 - Y_1)}{(X_2 - X_1)}$, which is a function of $P_1 - P_2$. From Lemmas 4.1, 4.2, and 4.3, the positional attribute and the orientation attribute are independent.

The above argument uses the assumption that the variances are the same for every random variable.

4.5 Abstract Attributed Features

We discussed three realizations of attribute features in Section 4.2. We generalize this idea to heterogeneous feature types. With heterogeneous features, the information accumulation is based upon the features that are not of the same type. In this way, multisensor fusion is possible, since the image could be formed by different kinds of sensors, and the features could be extracted by different kinds of feature detectors. We could combine various sources of information after we find candidates for the target object. We discuss an alternative here, i.e., we mix various kinds of features together during the voting process. Various types of normalized features carry their own attributes in the same hash space. The resulting performance is dependent upon the implementation method for pattern matching. Recall that a *pattern* is defined as a collection of normalized attributed features based upon a normalization basis of a model. We will discuss the implementation issues in Chapter 5.

An abstract attributed feature is defined as $\vec{v}$, where $\vec{v} = (v_1, \ldots, v_t)$, and t is dependent upon the type of feature. Conceptually, two attributed features of different types could have some common fields and the values in common fields can be compared. On the other hand, two fields in the same coordinate location from different features may not form a common field. This can happen even if the features have the same dimensionality. For example, the position of the midpoint of a line can not be matched to a bisector located at the same position. Even though their positional information can match perfectly, we can not accumulate votes since these two features are not compatible. On the other hand, different types of features could match together in certain situations. For example, a bisector for a corner could match to a multi-angle corner, since a single corner may be detected at a multi-angle corner due to noise. Even though the dimensionality of the two features are different, they could match to each other.

Conceptually, we map every kind of feature to a universal attributed feature. The features are unified by the universal attributed feature type. Those fields from different feature types that point to the same fields in a universal attributed feature type can be compared and matched together. Figure 4.8 shows this concept. Heterogeneous features $f_0 \ldots f_3$ are mapped to the universal attributed feature which contains the union of the fields for all feature types. From now on, we discuss the heterogeneous attributed features

using this universal attributed feature platform.

Basically, a universal feature is composed of *essential components* and *annotative components*. Essential components are essential to define the transformation. For example, since the (x, y) components of a basis in our orientation-attributed features are used to define the coordinate system for a similarity transformation, both x and y components are essential. The angle θ in an orientation-attributed feature is simply an attribute that provides an additional description of the feature; it is an annotative component. To fit into our geometric hashing method, a feature must contain essential components. As shown in Figure 4.8, recall that feature type f_1 and f_2 are midpoint and endpoint features respectively; since they are not compatible to each other, the x, y, and θ fields are mapped to different essential components of universal feature. Feature type f_0 is the corner feature which is found by the intersection of two line segments, while feature type f_3 is the bisector of the corner feature. Since feature type f_0 and f_3 are compatible to each other, the x and y fields are mapped to the same essential components e_{10} and e_{11}. The importance of each feature can be defined by β_i as described in Section 3.2.1. In general, some fields will be missing in a universal feature space when a feature type is mapped to universal feature. We may associate an *always_TRUE* or *always_FALSE* function to these missing fields during the matching process. The interpretation of these fields is that the distance is 0 (or ∞) when that field is missing. Note that the basis that defines the transformation of the coordinate system is also attributed, and the attributed information can be mapped to annotative components of universal feature when a feature is normalized. The basis set can also be heterogeneous.

In this fashion, heterogeneous features are unified by the universal feature type so that the fusion of various information is possible.

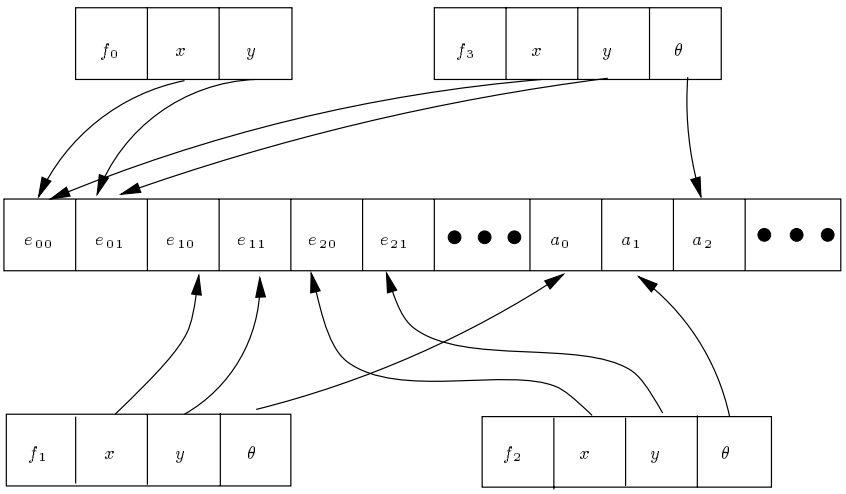

Figure 4.8: Heterogeneous types of features $f_0 \ldots f_3$ are mapped to universal feature type for similarity transformation. The (x, y) components of feature type f_0 and feature type f_3 are mapped to the same fields e_{00} and e_{01} respectively since f_0 and f_3 are compatible.

Chapter 5

Efficient and Distributed Implementation

Having the formulation for our geometric hashing method described in Chapter 3 and Chapter 4, we now consider efficient implementation methods. The key capability is to find efficiently the closest hash entries for a given normalized attributed feature. We discuss alternatives for accessing the hash table, i.e., linear access, a binning idea, and a binary tree search.

In order to make use of a distributed computing environment, we can divide the hash table into several pieces. Our method of distributed computation makes this object recognition system an ideal application example that can fit into many distributed computation models.

5.1 Linear Access and Binning

We first discuss three methods of accessing the hash table, namely, linear access, binning, and binary tree search. However, our focus is on the last method, more specifically, k-d tree search, which is the generalization of binary tree search in k-dimensional space.

Recall that in Section 3.5, we need to find the nearby entries for the normalized scene features in order to accumulate votes for various hypotheses. The naive way is to compute the distance from the given hash location to each hash entry and to maintain minimums for all filters. In essence, this kind of linear access method has no power of "hashing". A better performance can be achieved if we make use of the attribute information of the basis to quickly discard improper filters. Recall that a filter is defined as all the normalized features corresponding to the same (model, basis) hypothesis (i.e., a model pattern). When the hash table is organized as one filter after another, we can use the attribute information of the basis to skip filters that have no chance of matching. A

particular implementation shows the performance improvement is about four times faster than the linear approach, when the number of features per filter averages thirty. An even better performance improvement can be achieved if we store the filters in the table indexed by the attributed information of the basis. Only the filters with proper attributed basis information then participate in the computation.

If the hash entries are approximately uniformly distributed in the whole hash space, then the binning idea can be used. That is, we may partition the hash space into *bins*. Bins can be uniform or nonuniform, i.e., the size of each bin can vary if the hash space density is nonuniform. The entries located in the bins that are close to the normalized scene features are the candidates for vote accumulations. The entries in a bin can be stored as a linear list, or a binary search tree. Ideally, if we have a large number of partitions, there is at most one entry in each bin, thus achieving the "hashing" power in $O(1)$ time. In reality, this is not practical if the domain which we partition represents only feature positional information. Figure 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8 shows the histogram analysis of a hash table for synthesized polygons and fourteen car models. From top to bottom, the first five figures show the histogram in u-v space, then the histogram for orientation information of the hash entries. The last two figures show the density distribution of the angular attributes of the bases. Recall that (u, v) is the coordinate of a feature in the coordinate system defined by the selected basis. The statistics of the corresponding histograms are shown in Table 5.1 and 5.2. The figures show that an extremely high peak occurs near the origin for the histogram in u-v space. For the case of fifteen-polygon models, the bounding box for including all the hash entries in u-v space is $(-8.5419, -12.6836)$ and $(8.5419, 12.6836)$. We have a quasi-uniform distribution for each bin in the range of $(-0.4188, -0.6220)$ and $(0.4188, 0.6220)$ which is about $10\sigma \times 10\sigma$ in area. The distribution for hash entries and basis attributes are more likely to be uniform, which shows the feasibility of using basis attributes as the indexing key as just described. For the case of fourteen car models, the bounding box for including all hash entries extends to $(-32.7800, -62.9728)$ and $(32.7800, 62.9728)$. An almost uniform distribution of the entries occurs when in the range of $(-0.3128, -0.1564)$ and $(0.3128, 0.1564)$ which is around $10\sigma \times 5\sigma$ in area. Note that there are a large number of entries which have a horizontal orientation attribute. The reason is that there are many horizontal line segments in the car models. We will describe the data set for the experiments in Chapter 6. It is possible to use a rehashing function to redistribute the hash entries [86]. However, the disadvantage of rehashing is that the neighborhood relation among the hash entries might not be preserved.

The binning idea can be extended to the ellipse selection method as described in the following. Recall the covariance matrices that we derived in Chapter 4 define ellipsoids. In the hash table, instead of finding nearby entries from the nearby bins during the

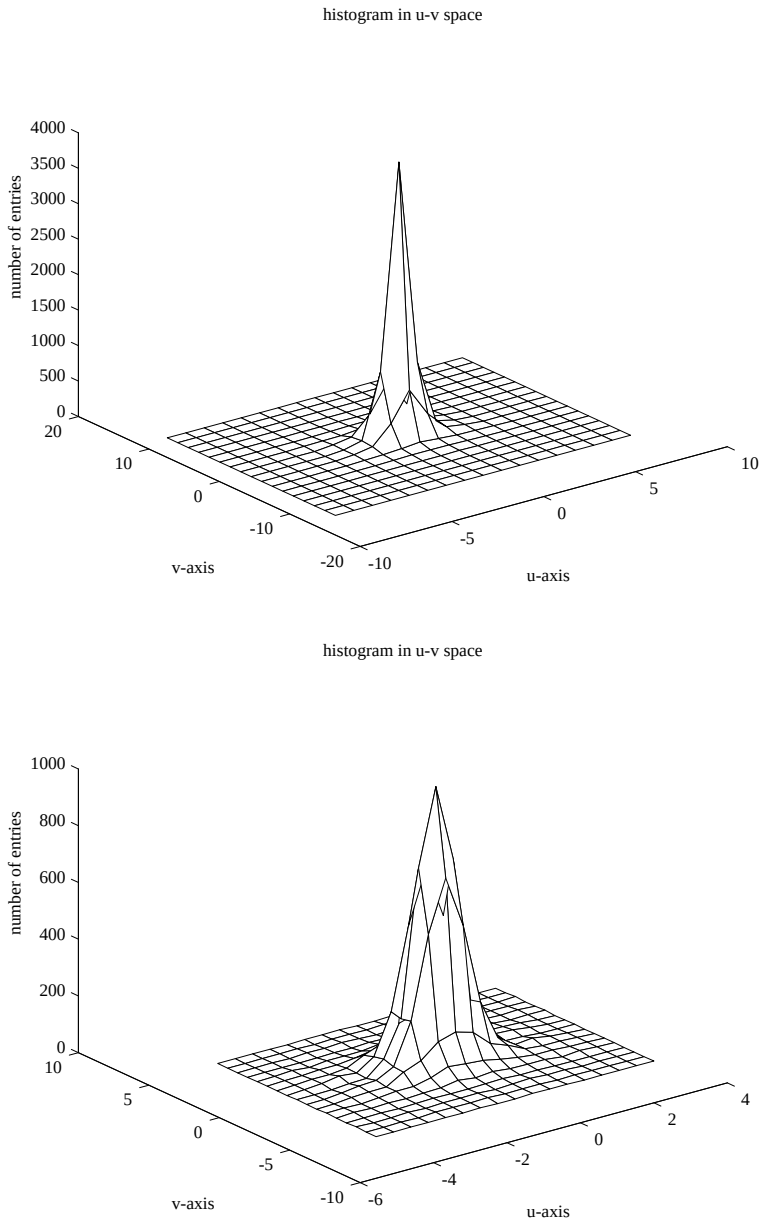

Figure 5.1: The histogram analysis of u-v space for the hash table of fifteen polygon models.

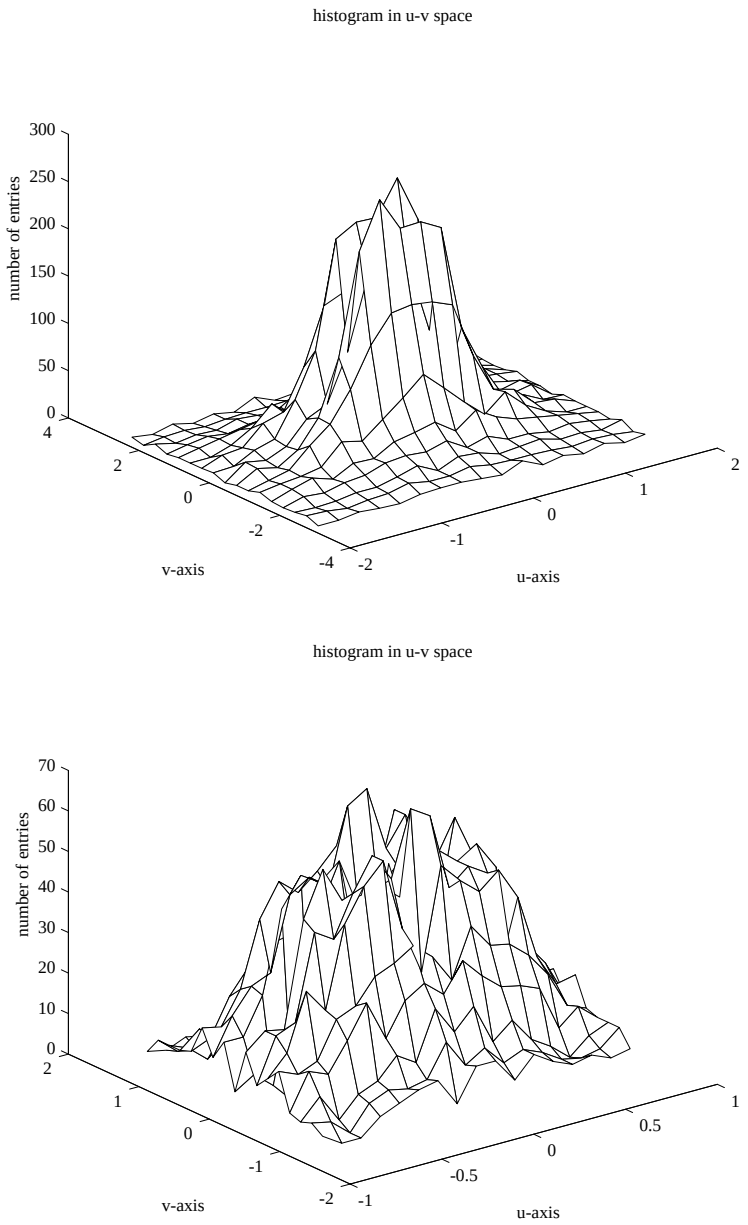

Figure 5.2: The histogram analysis of u-v space for the hash table of fifteen polygon models. The figure focus on the center of the histogram in Figure 5.1.

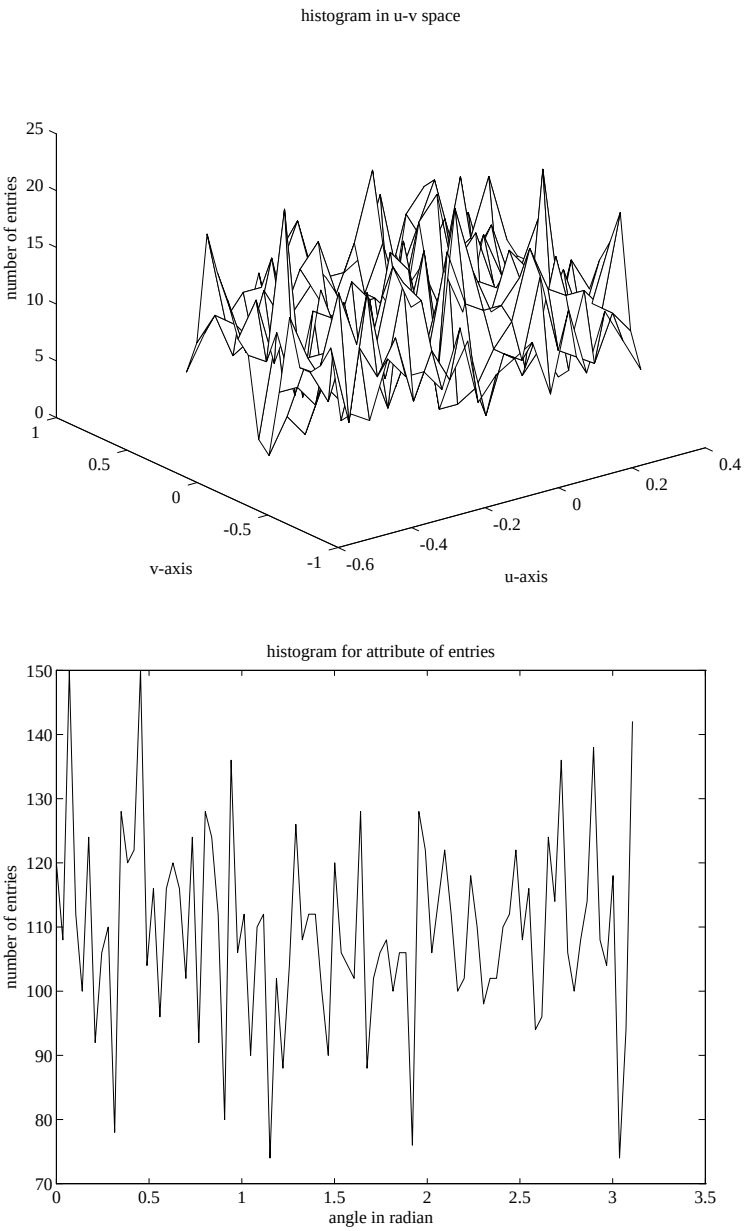

Figure 5.3: The histogram analysis for the hash table of fifteen polygon models. Top: The center of the histogram of Figure 5.2. Bottom: The histogram for the angular attribute of the entries.

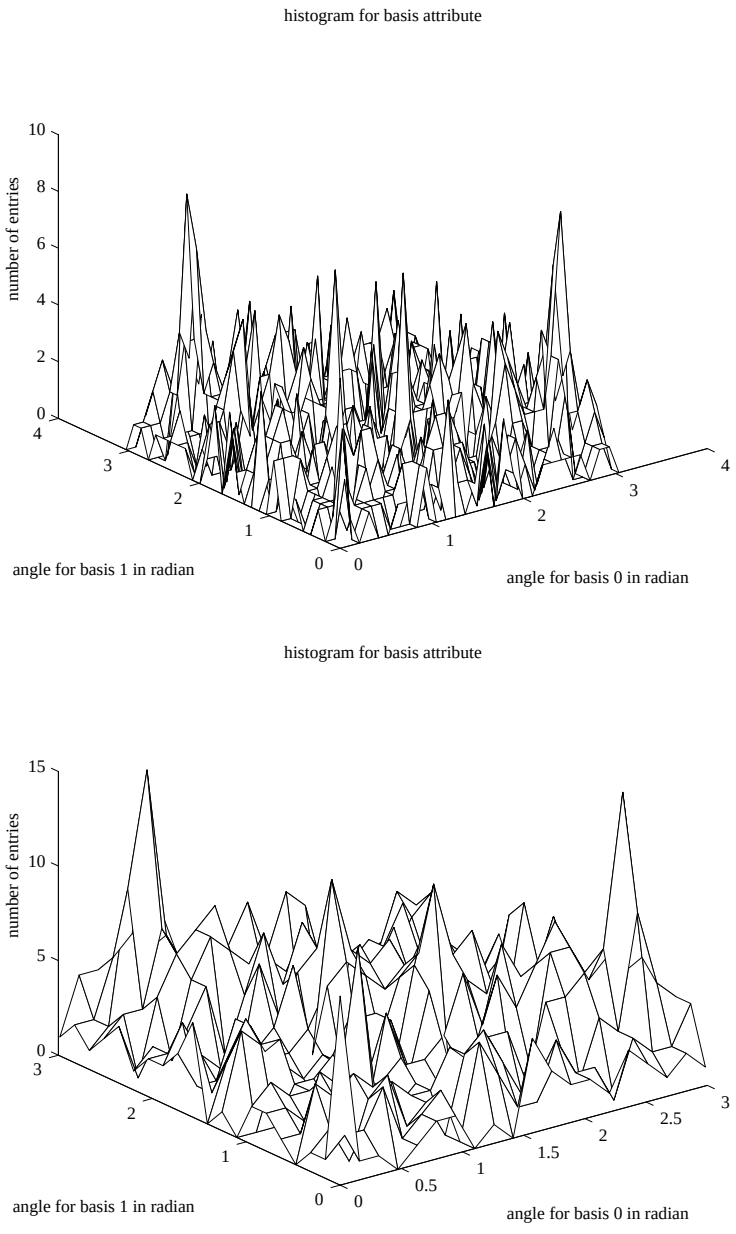

Figure 5.4: The histogram analysis for the hash table of fifteen polygon models. The figure shows the histogram of the angular attribute of the bases for two different quantization steps.

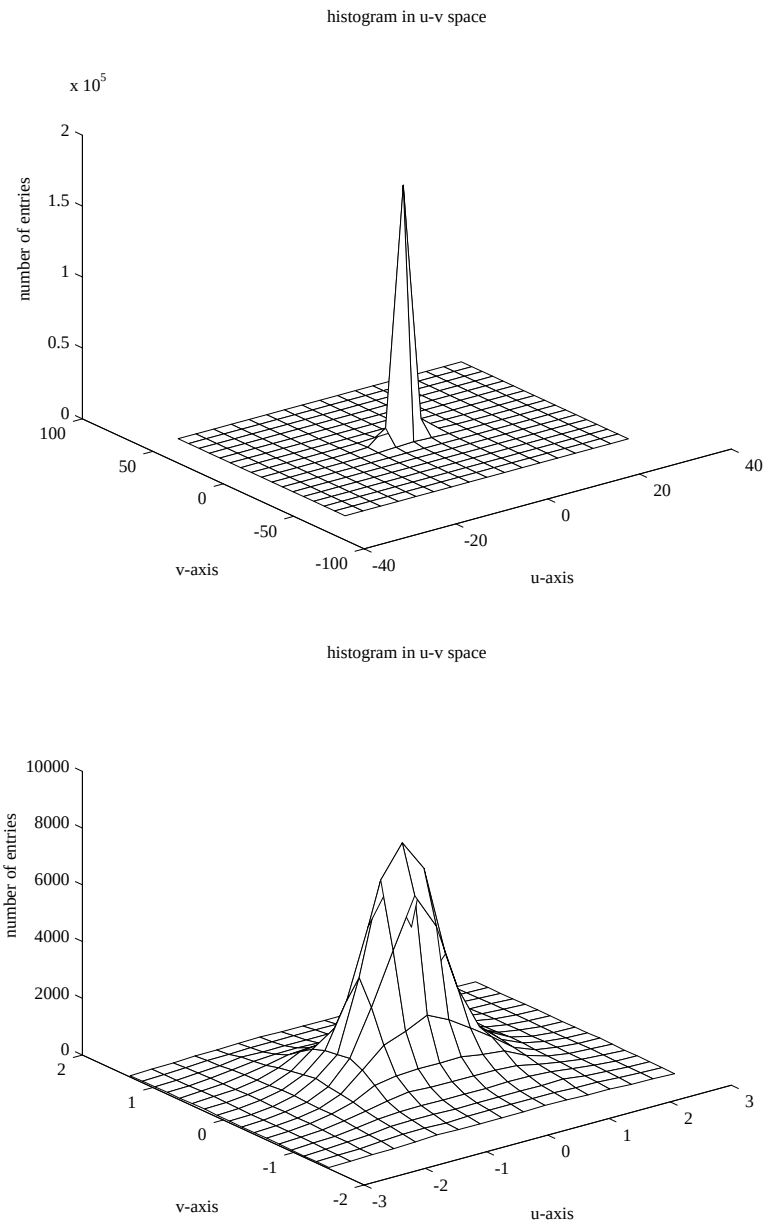

Figure 5.5: The histogram analysis for the hash table of fourteen car models.

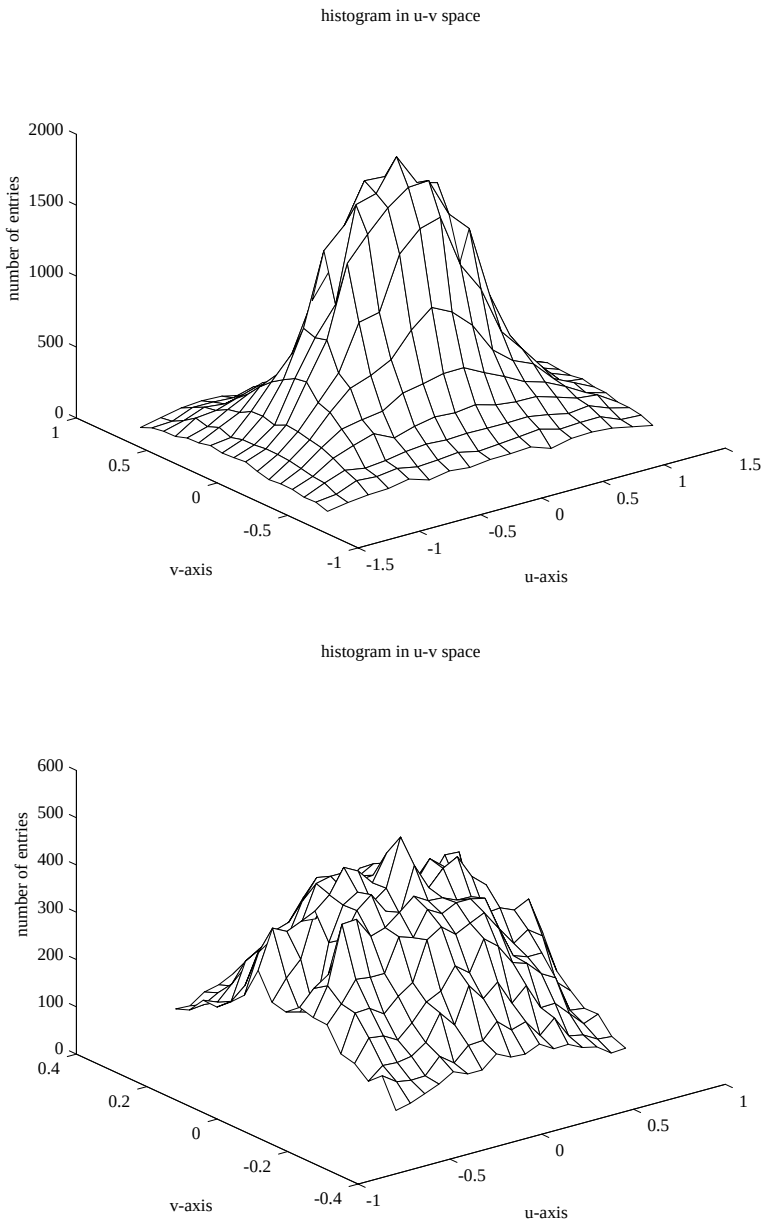

Figure 5.6: The histogram analysis for the hash table of fourteen car models. The figure focus on the center of the histogram in Figure 5.5.

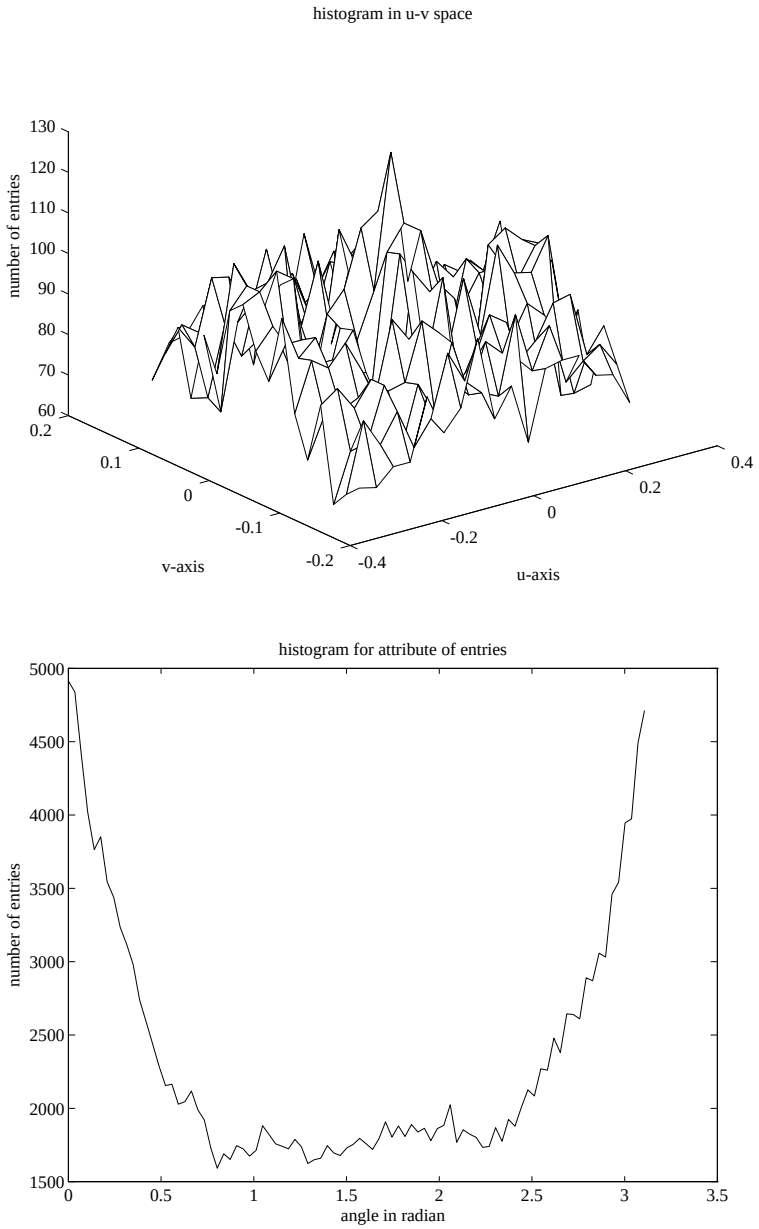

Figure 5.7: The histogram analysis for the hash table of fourteen car models. Top: The center of the histogram of Figure 5.6. Bottom: The histogram for the angular attribute of the entries.

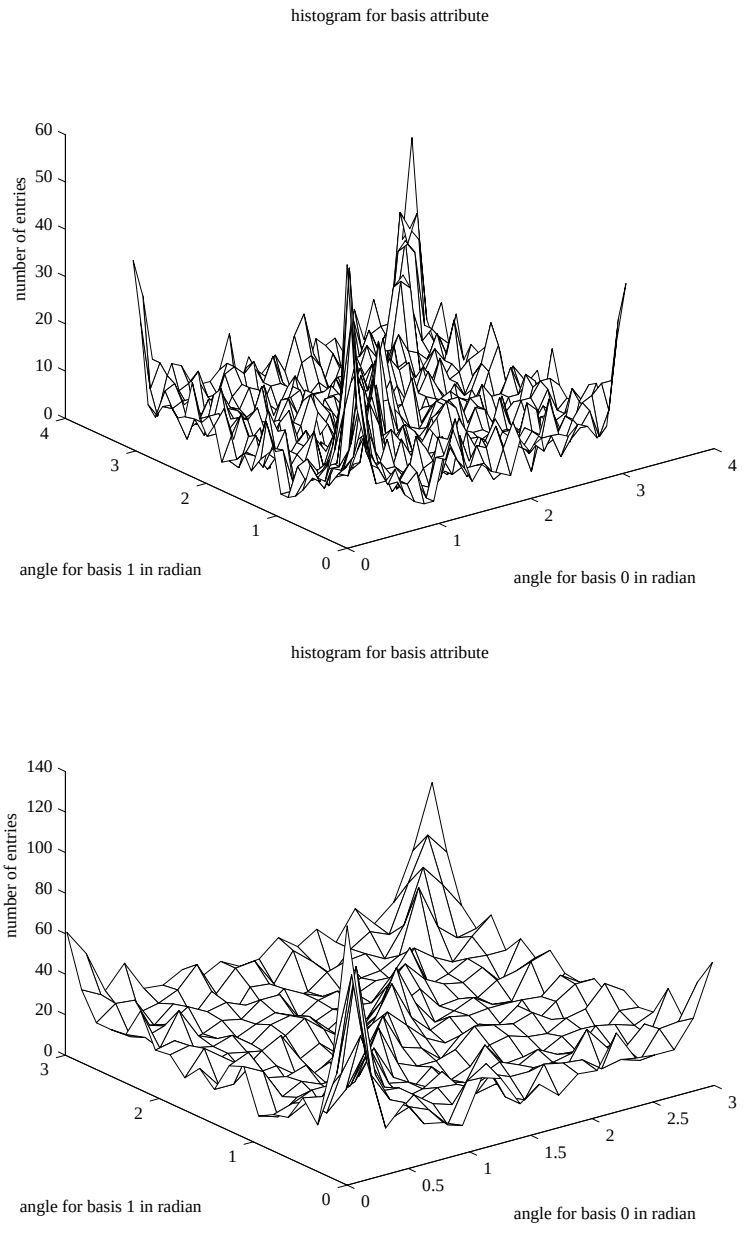

Figure 5.8: The histogram analysis for the hash table of fourteen car models. The figure shows the histogram of the angular attribute of the bases for two different quantization steps.

histogram name	u-v-0	u-v-1	u-v-2
number of entries	9858	9858	9858
bounding box	$(-8.5419, -12.6836)$ $(8.5419, 12.6836)$	$(-4.0196, -5.9688)$ $(4.0196, 5.9688)$	$(-1.8916, -2.8088)$ $(1.8916, 2.8088)$
number of grids	17×17	17×17	17×17
entries discarded	0	108	858
step size	(1.0049, 1.4922)	(0.4729, 0.7022)	(0.2225, 0.3304)
bounding box for peak cell	$(-0.5025, -0.7461)$ $(0.5025, 0.7461)$	$(-0.2364, -0.3511)$ $(0.2364, 0.3511)$	$(-0.3338, -0.4957)$ $(-0.1113, -0.1652)$
number of peaks	1	1	2
maximum	3868	972	264
minimum	0	0	0
mean	34.1107	33.7370	31.1419
standard deviation	250.0370	114.9270	51.5005

histogram name	u-v-3	u-v-4	
number of entries	9858	9858	
bounding box	$(-0.8900, -1.3216)$ $(0.8900, 1.3216)$	$(-0.4188, -0.6220)$ $(0.4188, 0.6220)$	
number of grids	17×17	17×17	
entries discarded	3180	6916	
step size	(0.1047, 0.1555)	(0.0493, 0.0732)	
bounding box for peak cell	$(-0.0524, -0.3887)$ $(0.0524, -0.2332)$	$(-0.0246, -0.3293)$ $(0.0246, -0.2561)$	
number of peaks	2	4	
maximum	68	21	
minimum	0	1	
mean	23.1073	10.1799	
standard deviation	16.4853	3.9375	

histogram name	hash entry attribute	basis attribute-0	basis-attribute-1
number of entries	9858	1176	1176
bounding box	$(0, \pi)$	$(0, \pi)(0, \pi)$	$(0, \pi)(0, \pi)$
number of grids	90	30×30	20×20
entries discarded	0	0	0
step size	0.0349	(0.1047, 0.1047)	(0.1571, 0.1571)
bounding box for peak cell	(0.0698, 0.1047)	(2.7727, 0.4189) (2.8274, 0.5236)	(2.6704, 0.4712) (2.8274, 0.6283)
number of peaks	2	2	2
maximum	150	9	15
minimum	74	0	0
mean	109.5330	1.3067	2.9400
standard deviation	15.0497	1.3455	2.2730

Table 5.1: The statistics of the histogram for the hash table of fifteen-polygon models.

histogram name	u-v-0	u-v-1	u-v-2
number of entries	211362	211362	211362
bounding box	$(-32.7800, -62.9768)$ $(32.7800, 62.9728)$	$(-3.0000, -1.5000)$ $(3.0000, 1.5000)$	$(-1.4116, -0.7060)$ $(1.4116, 0.7060)$
number of grids	17×17	17×17	17×17
entries discarded	0	34988	78218
step size	$(3.8565, 7.4086)$	$(0.3529, 0.1765)$	$(0.1661, 0.0831)$
bounding box for peak cell	$(-1.9282, -3.7043)$ $(1.9282, 3.7043)$	$(-0.1765, -0.0882)$ $(0.1765, 0.0882)$	$(-0.0830, -0.0415)$ $(0.0830, 0.0415)$
number of peaks	1	1	1
maximum	178820	8214	1986
minimum	0	40	85
mean	731.3560	610.2910	460.7060
standard deviation	10533.6000	1189.5000	433.5660

histogram name	u-v-3	u-v-4	
number of entries	211362	211362	
bounding box	$(-0.6644, -0.3324)$ $(0.6644, 0.3324)$	$(-0.3128, -0.1564)$ $(0.3128, 0.1564)$	
number of grids	17×17	17×17	
entries discarded	138144	186502	
step size	$(0.0782, 0.0391)$	$(0.0368, 0.0184)$	
bounding box for peak cell	$(-0.0391, -0.0196)$ $(0.0391, 0.0196)$	$(-0.0184, -0.0092)$ $(0.0184, 0.0092)$	
number of peaks	1	1	
maximum	502	130	
minimum	95	61	
mean	253.3490	86.0208	
standard deviation	99.5292	10.9079	

histogram name	hash entry attribute	basis attribute-0	basis-attribute-1
number of entries	211362	8826	8826
bounding box	$(0, \pi)$	$(0, \pi)(0, \pi)$	$(0, \pi)(0, \pi)$
number of grids	90	30×30	20×20
entries discarded	0	0	0
step size	0.0349	$(0.1047, 0.1047)$	$(0.1571, 0.1571)$
bounding box for peak cell	$(0.0000, 0.0349)$	$(0.0000, 0.0000)$ $(0.1047, 0.1047)$	$(0.0000, 0.0000)$ $(0.1571, 0.1571)$
number of peaks	1	1	1
maximum	4914	60	128
minimum	1592	0	6
mean	2348.4700	9.8067	22.0650
standard deviation	852.6690	6.2543	12.7703

Table 5.2: The statistics of the histogram for the hash table of fourteen-car models.

recognition stage, we can precompile the hash table so that each bin stores all the hash entries that we need to examine. Once a normalized scene feature is computed, it hashes to one of the bins and retrieves all the entries stored in that bin. Since a hash entry is now replicated to occur in all the bins enveloped by the ellipsoid, this method requires larger memory space than the method described before.

5.2 K-d Tree and Its Generalizations

In this section, we discuss another approach that can be used to access the nearby entries in $O(\log n)$ time on average, namely, k-d tree search. The k-d tree implementation will be used in our experiments. We will describe the experiments in Chapter 6.

A k-d tree is a generalization of the data structure of a binary search tree [96], where k denotes the dimensionality of the spatial data being represented. In essence, the k-d tree is a binary search tree. The difference between an ordinary binary search tree and a k-d tree is that the discriminator for a k-d tree is chosen from one of the k-fields. The discriminator for a node n means the key used to perform the comparison, which determines whether the search proceeds to the left, right, or down both sub-branches of n. We will denote the discriminator for the node n as **disc**(n). The choice of discriminator can lead to various k-d trees, such as an optimized k-d tree [8] or an adaptive k-d tree [36]. For simplicity, we discuss the original k-d tree, i.e., the discriminator of successive levels is simply the next field of the discriminator in the current level. In our application, our main concern is with the construction of a k-d tree and the search process in a k-d tree, since we never delete a node from the tree.

The use of a k-d tree permits us to perform the range query of spatial data quite efficiently. In this section, we discuss the generalization of the discriminator values in k-d tree search so that the data can be non-linear; specifically, cyclic data, such as angles. Further generalizations for searching multi-angle data such as tri-corners are also discussed.

There are three issues related to the search in a k-d tree, namely:

(1) The definition of the comparison function, **kdt_compare**;

(2) The method for searching; and

(3) The method for pruning the branches.

Once we have defined the comparison function, issues (2) and (3) can be solved accordingly. The returned value of the comparison function is either **left_son** or **right_son**. The idea of the k-d tree for ordinary one-dimensional data defined on an interval is as follows. Suppose we want to query all the nodes that represent data lying within

the range of $(k_{\min}, k_{\max})$. The comparison is ordinary linear comparison. That is, if $n.\mathbf{disc}(n_i) \leq n_i.\mathbf{disc}(n_i)$, we know that node n is located at the left branch of node n_i (include the node n_i if equality occurs). Similarly, we should visit the right branch if $n.\mathbf{disc}(n_i) > n_i.\mathbf{disc}(n_i)$. Here, we use the notation that $n.j$ represents the j-th field of the data stored in node n. Tree pruning occurs if

- **kdt_compare**$(k_{\min}, n_i)$ is a right son, then we cut off the left branch including node n_i,
- **kdt_compare**$(k_{\max}, n_i)$ is a left son, then we cut off the right branch including node n_i.

For the case that neither of the above two conditions are true, we have to visit both left and right branches. The principle behind the search is that when we visit a node of the tree, we divide the space into two parts $\mathbb{R}_+$ and $\mathbb{R}_-$ according to the chosen discriminator, so that we need only to search one of the two branches in most cases when tree pruning occurs. Furthermore, we should make sure that comparison function and space partition method preserve the transitive property. That is, if a node n_B is in the left branch of node n_A and a node n_C is in the left branch of the node n_B, we can conclude that the node n_C is in the left branch of the node n_A.

5.2.1 K-d tree for angular data

The generalization of the k-d tree for cyclic data such as angular data is straitforward. For example, suppose the midpoint feature (u_i, v_i, θ_i) is stored at the node n_i. If the discriminator for n is θ_n, the notation for ordinary comparison is not useful since the angle data is cyclic. Suppose that angle θ and angle $\theta + \pi$ are treated as a perfect match. Let us assume the range query is (θ_1, θ_2), where $\theta_1 = \theta - \delta_\theta$, and $\theta_2 = \theta + \delta_\theta$. We also assume that $\delta_\theta < \frac{\pi}{4}$. The function **kdt_compare** can be defined as

$$\mathbf{kdt\_compare}(\theta, \theta_i) = \begin{cases} \text{left\_son} & \text{if } |\theta - \theta_i| \leq \frac{\pi}{4} \\ \text{right\_son} & \text{otherwise.} \end{cases}$$

The tree is organized such that the left branch stores all the nodes with an angle difference from θ_i less than $\frac{\pi}{4}$, while the right branch store the rest of the nodes. During the search, when we visit a node n_i of the tree, the tree pruning can be performed as follows:

- if the result of **kdt_compare**(θ_1, θ_i) and **kdt_compare**(θ_2, θ_i) are both **left_son**, then we prune the right branch,[1] and

[1] If $(\theta_1 - \theta_i) \cdot (\theta_2 - \theta_i) > 0$, then node n_i can be excluded also.

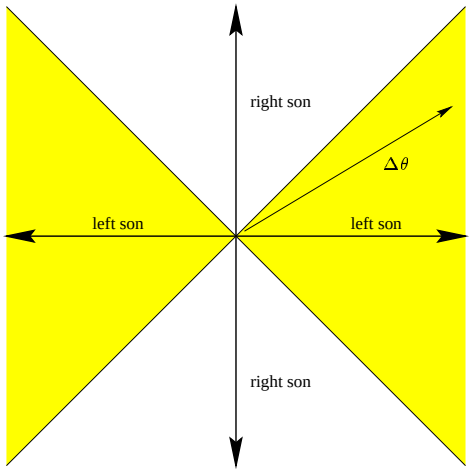

Figure 5.9: The space partition for $\Delta\theta$ when θ and $\theta + \pi$ are considered as the same, where $\Delta\theta = \theta - \theta_i$. The comparison function **kdt_compare** returns either **left_son** or **right_son** according to the range of $\Delta\theta$.

- if the results of **kdt_compare**(θ_1, θ_i) and **kdt_compare**(θ_2, θ_i) are both **right_son**, then we prune the left branch including node θ_i.

The space partition for the difference of the angle $\Delta\theta = \theta - \theta_i$ is shown as Figure 5.9.

If we consider angle θ and angle $\theta + \pi$ as a mismatch, we can modify the method of space partition as shown in Figure 5.10 and use the similar method as described above to search and prune tree.

5.2.2 K-d tree for multi-angle features

The generalization to multi-angle features such as tri-corners can be achieved in the following. We make use of the k-d tree in a slightly different way than described in the previous sections.

A multi-angle corner C is represented as $(u, v, \theta_1, \theta_2, \ldots)$, which is a corner located at the position (u, v) with several distinct angular components $(\theta_1, \theta_2, \ldots)$. We will require a notion of when two multi-angle features match. Let $C_1 = (u, v, \theta_1, \theta_2, \ldots \theta_m)$ and $C_2 = (u', v', \theta'_1, \theta'_2, \ldots \theta'_n)$. We have the following three matching definitions for multi-angle corners:

Strong matching: We say that C_1 strongly matches C_2 if $(u, v) = (u', v')$, $m = n$, and $\{\theta_1, \theta_2, \ldots \theta_m\} = \{\theta'_1, \theta'_2, \ldots \theta'_m\}$.

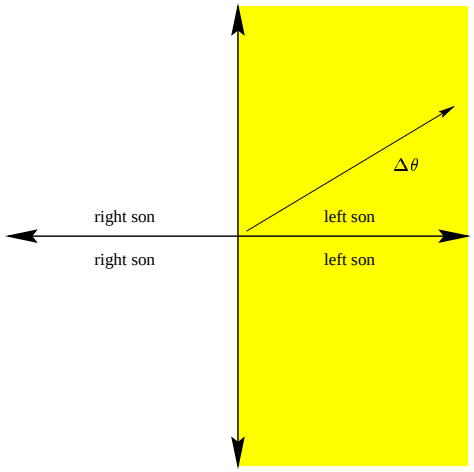

Figure 5.10: The space partition for $\Delta\theta$ when θ and $\theta + \pi$ are considered as different.

Weak matching: We say that C_1 weakly matches C_2 if $(u, v) = (u', v')$, and $\{\theta_1, \theta_2, \ldots \theta_m\} \subseteq \{\theta'_1, \theta'_2, \ldots \theta'_n\}$ or $\{\theta_1, \theta_2, \ldots \theta_m\} \supseteq \{\theta'_1, \theta'_2, \ldots \theta'_n\}$.

Partial matching: We say that C_1 partially matches C_2 if $(u, v) = (u', v')$, and $\{\theta_1, \theta_2, \ldots \theta_m\} \cap \{\theta'_1, \theta'_2, \ldots \theta'_n\} \neq \Phi$.

Each notion of matching extends naturally to a notion of approximate matching.

When testing scene multi-angle features to see if they match model multi-angle features, a k-d tree approach can be used to speed the associations. Depending on the level of matching, the pruning strategy of the k-d tree will vary. We discuss pruning strategies below. The position of the multi-angle corner typically will be considered at the other levels of the k-d tree. We only consider the multi-angle fields here. Further, we consider only strong matching.

Although we could order the partition elements $\theta_1 < \theta_2 < \ldots < \theta_m$, and make use of this ordering to assist in organizing the search in the k-d tree, it is simpler to make use of the fact that for any given location (u, v), there are likely to be only a small number of multi-corners stored at the given location (each will belong to a different model). We can define the following distance measure $\mathbf{D}$ for strong matching:

$$\mathbf{D}(\{\theta_1, \ldots \theta_m\}, \{\theta'_1, \ldots \theta'_m\}) = \min_{\mathbf{P}} \mathbf{D}(\mathbf{P}(\theta_1, \ldots \theta_m), (\theta'_1, \ldots \theta'_m)),$$

where $\mathbf{P}$ runs over the set of permutations, and $\mathbf{D}$ measures the difference between vectors

of ordered angular components as defined by:

$$\mathbf{D}((\theta_1, \ldots \theta_m), (\theta'_1, \ldots \theta'_m)) = \sum_{i=1}^{m} (|\theta_i - \theta'_i| \pmod{\pi}).$$

Even though the cost of finding the distance for a multi-angle corner is relatively high, they are not likely to occur often. The structure of the k-d tree is organized so as to place matched and neighboring corners in the left branch and the mismatched corners in the right branch. The tree is unbalance at this level, which can be balanced back to some degree at the other levels using other attributes as discriminators. Accordingly, we construct a tree such that each node contains a set of angles $\{\theta_1, \ldots \theta_n\}$, constituting the union of angles at the children nodes, and each angle is marked as being derived from the left, right, or both children. Locating an entry then becomes a simple matter of testing successive angles for set membership, and walking down the tree. Leaf nodes represent stored multicorners. For strong matching, we must check that the number of angles match. For weak matching, all descendent leaf nodes, after exhausting the angles in the test multicorner, constitute weak matches.

There is no obvious way to speed weak matching and partial matching other than by brute force. The problem is that it is difficult to define a distance measure with transitivity property using these two matching definition. In essence, if a node a matches node b and b matches node c, we are not sure if node a can match node c.

5.2.3 K-d tree for heterogeneous data

We have assumed that all the nodes inserted into the k-d tree are of the same type. That is, the dimensionality and the corresponding fields for all nodes are homogeneous. Suppose that some of the nodes have some fields missing and the nodes inserted into the k-d tree are heterogeneous. Consider the situation that we are currently visiting an internal node n of the k-d tree. We need to decide how to insert an new hash entry n' to the tree, even though n' does not have the field corresponding to the current discriminator of node n. If we treat that missing field as a perfect match (distance 0) in all cases, we can insert n' to the left branch of the current node n. During the search process, we need to find all the entries that are close to n'. Again, suppose that a feature n' does not have the field corresponding to the current discriminator of the internal node n. We can not prune branches of the tree from node n. That is, we should search both branches. The reason is that both branches may contain nodes with that missing field filled. All of them are considered as perfect matches to node n'.

On the other hand, if we treat that missing field as a totally mismatch (distance ∞), we can insert the new node to the right branch. In the search process, if the visited internal node n uses a field as the discriminator which does not exist for the feature n',

the left branch should include node n and can be pruned during the search. We still need to continue the search process to the right branch of node n. The reason is that the right branch may contain the nodes with that field missing also, and they might match the feature n'. Recall that in Section 4.5, heterogeneous features can be realized by defining mappings to a universal feature. Different search and pruning procedures for a k-d tree are associated to various fields of the universal feature.

5.3 Distributing the Hash Table

To make use of the power of the current computer network to speed up the computation, we distribute the computation among processors. Our platform is based upon loosely coupled workstations communicating over a network. We discuss various distribution methods in this section. Depending upon the access method for the hash table, one distribution method could be more feasible than others. There are several possibilities:

(1) Every processor handles a portion of the hash space;

(2) Every processor handles a portion of the total trials;

(3) Every processor handles one or several models, which is the method adopted by Rigoutsos and Hummel [91];

(4) Every processor handles a collection of (model, basis) hypotheses, which is the method we use in our current implementation (with k-d tree search).

Method (1) has the following disadvantages:

1. Since the distribution of the hash entries in hash space is not uniform (see Figure 5.1 and 5.5), the partition should be nonuniform in order to balance the work load for each processor. For higher-dimensional features, the partition can become very complicated to compute.

2. For each normalized feature in the test image, it is possible that more than one partition is involved in the voting computation, which means that there must be interprocessor communication involved in the total vote computation for each normalized feature.

3. Again, there is interprocessor communication involved in vote accumulation for each (model, basis) hypothesis.

The problem is that more than one processor must be involved in the evidence accumulation for a hypothesis. The communication overhead makes this method impractical.

If each processor has enough memory to store the whole hash table, each processor can generate a set of trial basis and perform the vote accumulations. Ideally, we should be able to decide an a priori threshold for the stopping criterion so that once a processor finds a trial basis which support a hypothesis with sufficient confidence, the processor then broadcasts this message to the rest of processors. This distribution method is shown in Figure 5.11, which is an example of *loosely synchronous* complex system as described by Fox *et al* [35]. The disadvantage of this method is that the whole hash table must be replicated for each processor. However, this drawback can be solved if we combine this idea with method (3) or method (4). We will discuss this hybrid issue again later.

Method (4) is similar to method (3). However, method (4) partitions the computation more finely which makes it more practical if the number of features for a model varies. Finer granularity means we can divide the computation load more evenly. On the other hand, we maintain the property that only one processor is involved in the evidence accumulation for a single hypothesis. The main problem with method (4), like method (3), is that the hash space must be replicated for each processor. If we make use of the binning idea as described in Section 5.1, although each processor stores a portion of the hash table, each processor still needs a large amount of memory for the quantization array.

Figure 5.12 shows the partition of the hash table where one k-d tree sits on top of each partition. During the initialization stage, each processor reads in the corresponding part of the hash table and its own k-d tree. Note that the k-d tree can be built in a preprocessing stage once the partition has been determined. There is no communication required during the evidence accumulation stage until a trial with sufficient evidence is found or a certain number of trials is finished. The computation flow for this method is shown as Figure 5.13. Ideally, the size of each partition can depend upon the computation power of each processor, which is determined statically beforehand. This partitioning method is easily invoked using the parallel execution mode and remote execution mode of *cantata* in the Khoros software development environment [57]. A depiction of the *cantata* implementation for method (4) is shown as Figure 5.14. The glyph labeled *vbuild* is a procedure for building the hash table and the corresponding k-d tree. Its expansion is shown at left-bottom corner, containing *vsim* which builds the hash table for similarity-invariant recognition. Three instances of *vkdt* build three distinct k-d trees which depend on (command-line) parameters. The glyph labeled *vrecog* is a procedure for the recognition stage whose expansion is shown at right-bottom corner. The glyph *vpresim* initializes and generates the set of trial basis. Three glyphs follow, labeled *voting*, which computes the evidence for the hypotheses. Note that two of the *voting* glyphs are executed remotely by the machine *wilma* and *fred,* and the third is executed locally by the host *simulation.* Finally, a glyph labeled *vlogf* combines the results produced by *voting*'s. The implementation is the realization of the computational model described in Figure 5.13.

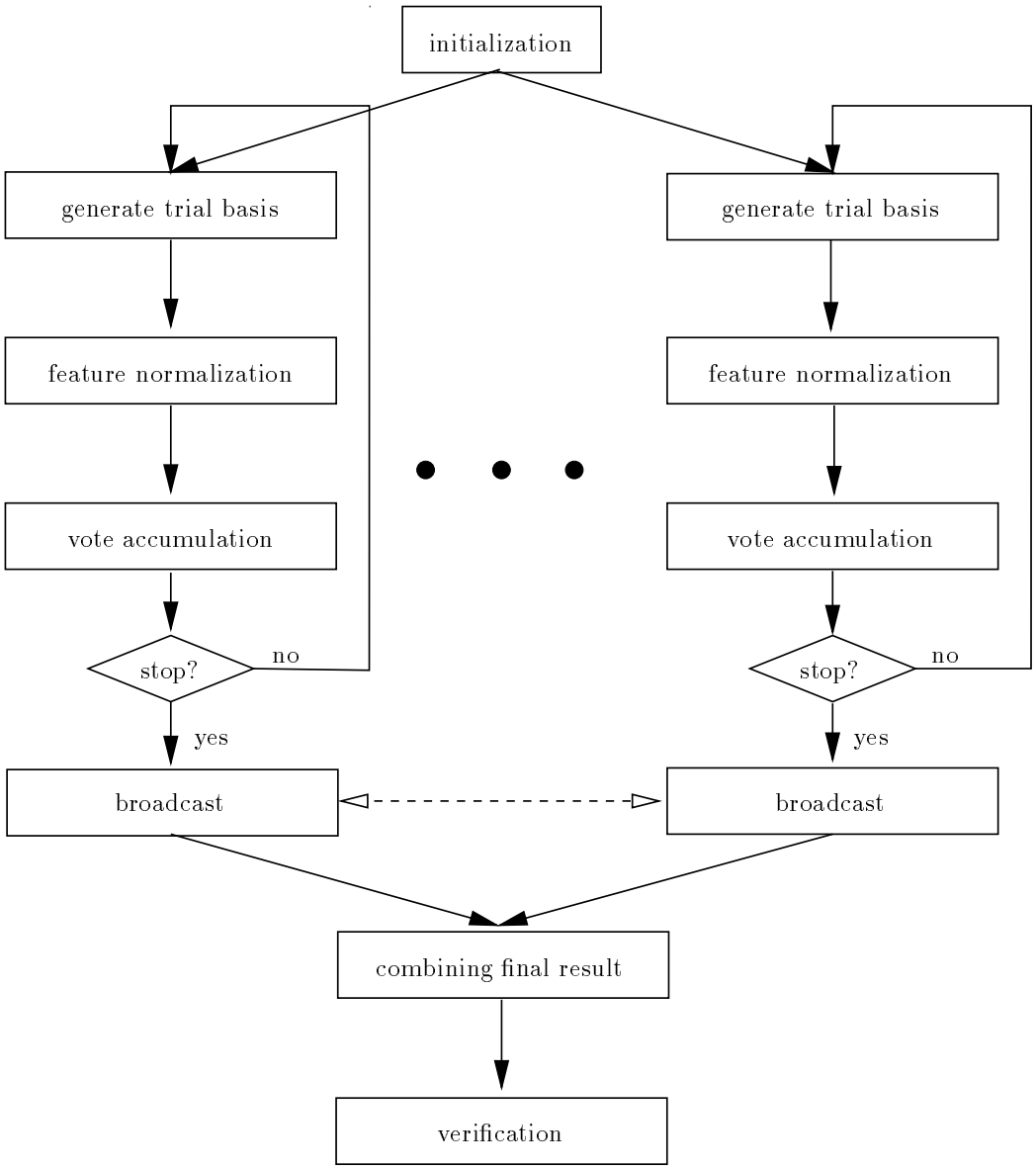

Figure 5.11: Distribution of trial basis.

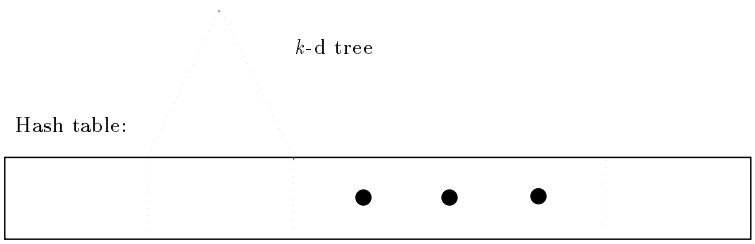

Figure 5.12: The partition of the hash table with one k-d tree sits on top of each partition.

Each partition together with the computation for a trial can be viewed as a task to be performed. Then the entire computation is composed of a list of tasks to be finished. From this point of view, the "work farm" model of parallel computation and the language PLinda[53] would be ideal for its implementation. A snapshot of the revised implementation is shown as Figure 5.15. This implementation uses the concept of abstract attributed features (Section 4.5) to handle the case of heterogeneous feature types (Section 5.2.3). It also contains supporting modules to display the recognition results. Detailed information for each module is described in Appendix B.

A hybrid parallel approach using methods (2) and (4) is also reasonable. Each processor computes the evidence for its own portion of the set of hypotheses based on the trial bases it generates. The set of trial bases for every processor is different, which means that the ranking of the evidence for a trial basis may be incorrect. However, as long as we can define a threshold value for the evidence, the solution we find is still reasonable. We may distribute the hypotheses of a model among the processors, so that each processor has the chance to fire the correct matching if that model indeed appears in the test image. We introduce this randomization to reduce the probability of unbalanced work load.

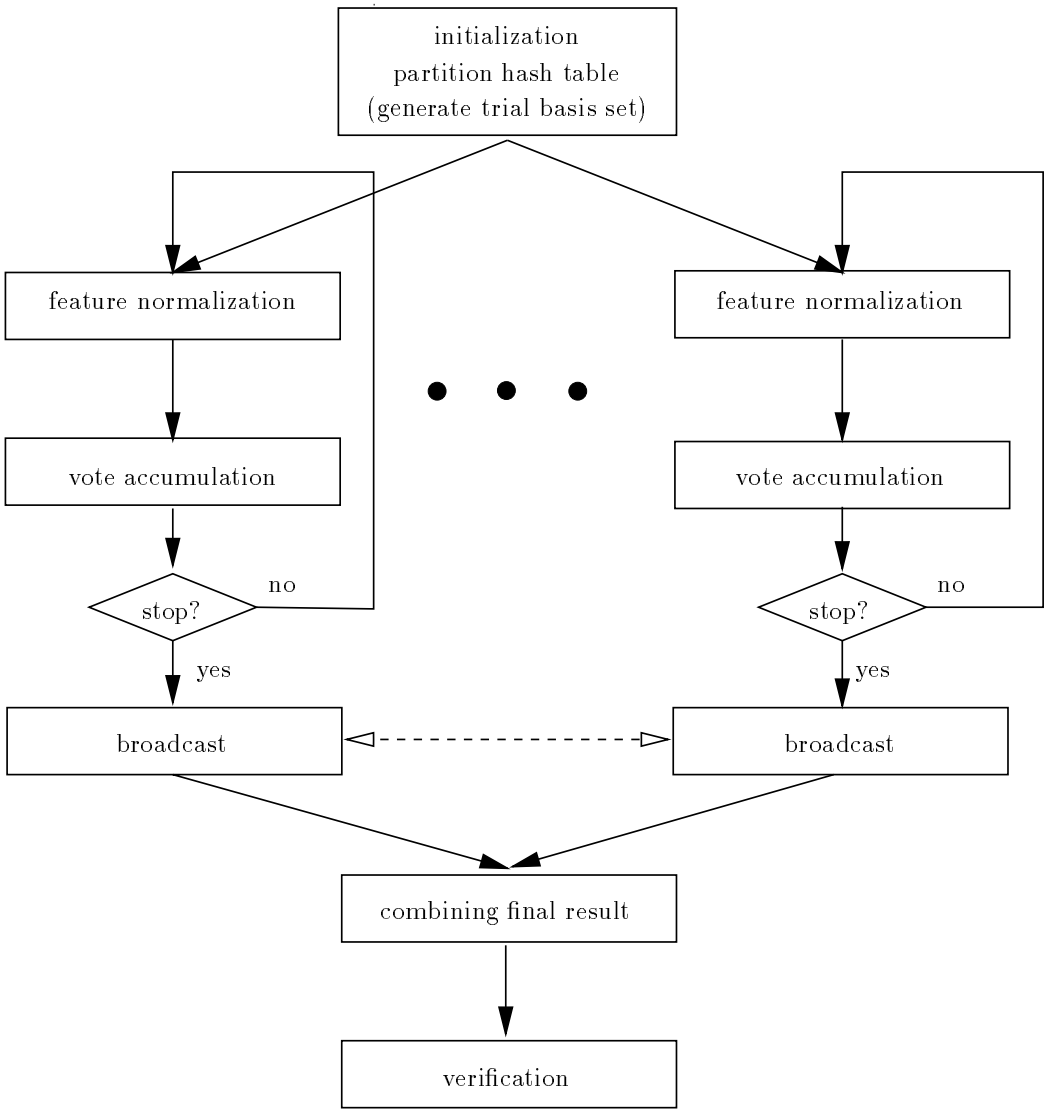

Figure 5.13: Distribution of hash table.

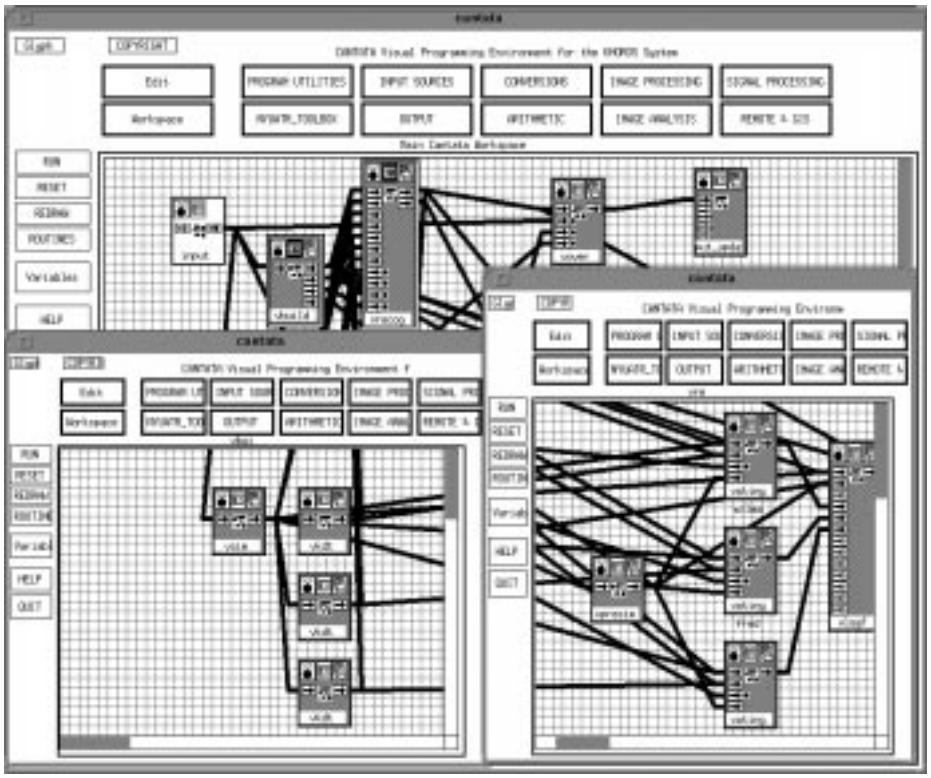

Figure 5.14: The snapshot shows the implementation for method (4) as Khoros modules.

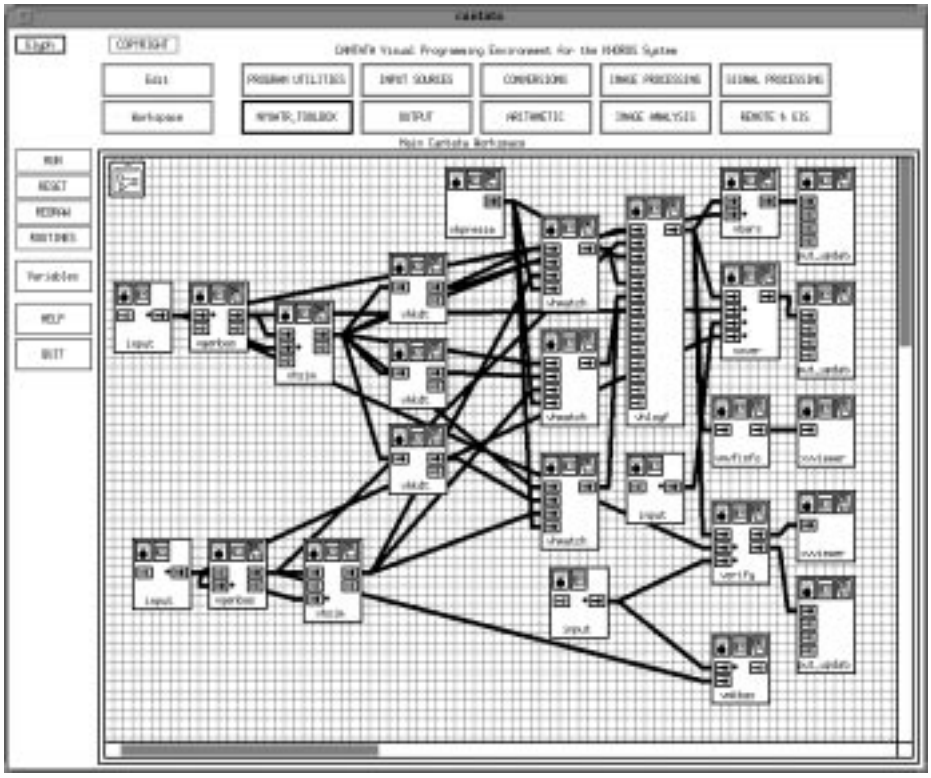

Figure 5.15: Here is a different snapshot of the revised implementation. This implementation handles the case of heterogeneous feature types.

Chapter 6

Experimental Results

We present results from many experiments. We first describe the model database used in our experiments. The feature extractors for various cases are described next. The recognition results are given in the last section.

6.1 Model Database

We first describe the models used in our experiment. Five sets of experiments are performed during our research. The model databases that we use for each set of experiments are respectively:

(1) Fifteen polygonal object models,

(2) Nineteen industrial part models,

(3) Fourteen views of car models,

(4) Three views of military land vehicles, from gray scale imagery.

(5) Thirty two views military land vehicles created from CAD-CAM models.

In subsequent experiments, the database in set (5) was increased to 88 views. The first set consists of a collection of fifteen synthesized polygons which are used to test the validity of the various programs. The models are shown in Figure 6.1.

The second set consists of a collection of nineteen industrial parts. Some of the objects have multiple stable states. An image of each stable state is built as a model. The images for models and test images are taken using a Sony CCD camera [69]. The gray-scale images of the models are shown in Figure 6.2.

The third set consists of seven brands of commercial vehicles. The pictures are taken from the street. For model images, the pictures are scanned using a flatbed scanner. Tires

name	size of hash table (bytes)	number of entries	feature type	number of models	number of objects	data source
poly15	221196	9858	midpoints	15	15	synthesized
cad19	1059188	49742	bisectors	19	9	Sony CCD camera
car14	4404268	211362	midpoints	14	7	Sony CCD camera,
						Silver scanner
tank3	993580	47760	endpoints	3	2	CCD camera
brl32	15994036	NA	hybrid (midpoints, circles)	32	8	CAD model

Table 6.1: The information for the model database used in our experiments.

and background are removed. For test images, the pictures are obtained using the CCD camera [69, 70]. The extracted edge maps for the models are shown in Figure 6.3.

The forth set of models consists of two kinds of military vehicles, the M60 tank and M113 vehicle. The images are obtained from the black-and-white portion of a color image, taken from a range of about 500 meters by a color CCD camera. The data was obtained from the Demo B imagery of the Unmanned Ground Vehicle program, made available to the RSTA (Reconnaissance, Surveillance, and Target Acquisition) research community [70]. For the M60 tank, we flip the image of the M60 to get the mirror reflective version of the model. The test image is from the same data set, but a different image than the model. The edge maps of the models are shown in Figure 6.4.

The fifth set of models consists of four kinds of military vehicles. Since the M35 truck has different appearance when a canvas is attached, we treat it as another object. The M35 truck with canvas attached and M60 tank have snapshots from two different tilt angles. Also, each object has four snapshots from four different pan angles. The data is generated from the imagery that was generated using BRL-CAD models. The SAIL software (Synthetic Assembly Image Layout) is used to generate simulated laser radar images (i.e., range images) from BRL-CAD models. The test image is from a dataset provided by Wright Laboratory, having been taken using a Cincinnati Electronics Indium-Antimonide (InSb)mid-wave infrared (MWIR) camera. The information for the model database used in our experiments is listed in Table 6.1.

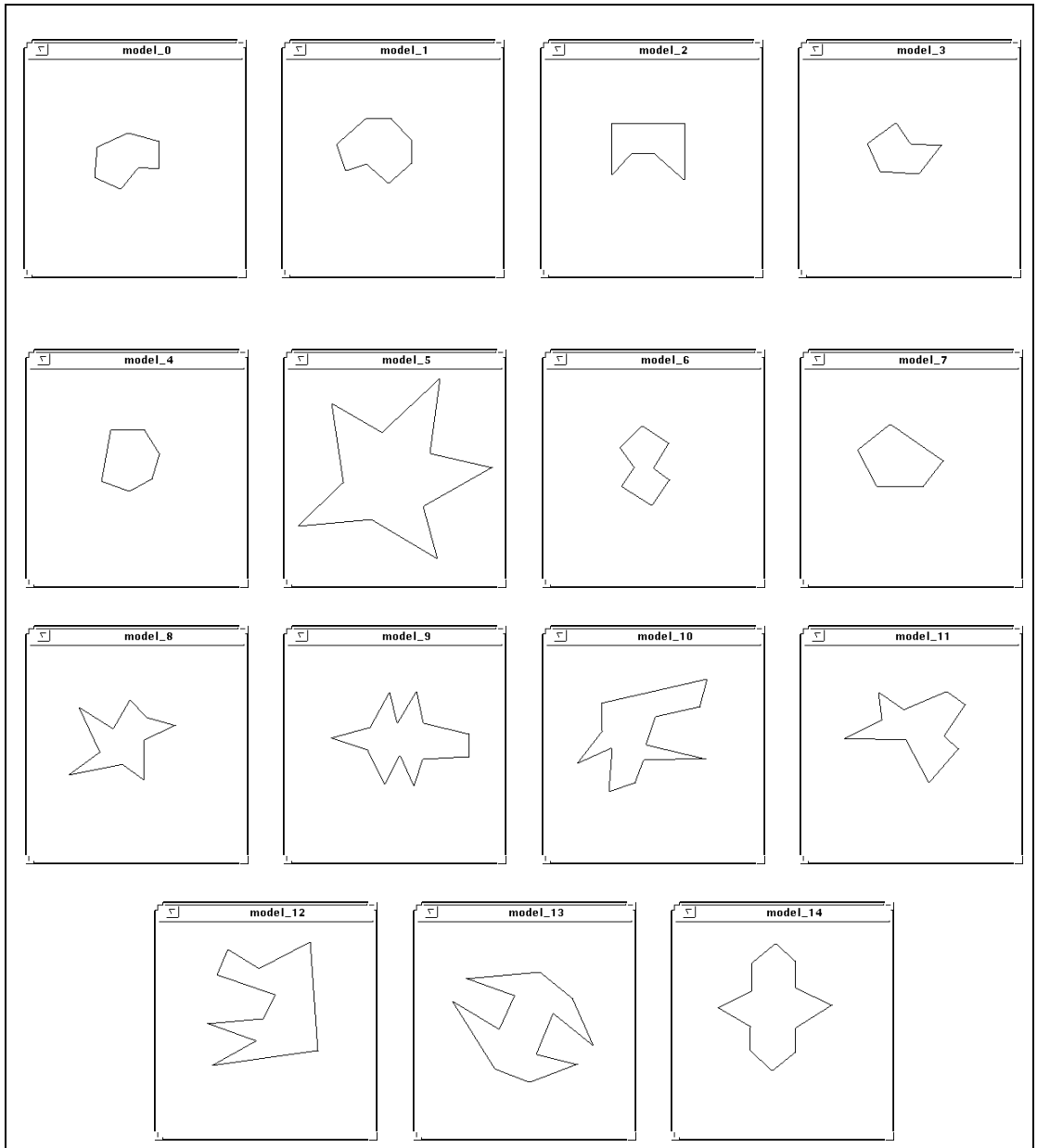

Figure 6.1: The collection of fifteen polygon models.

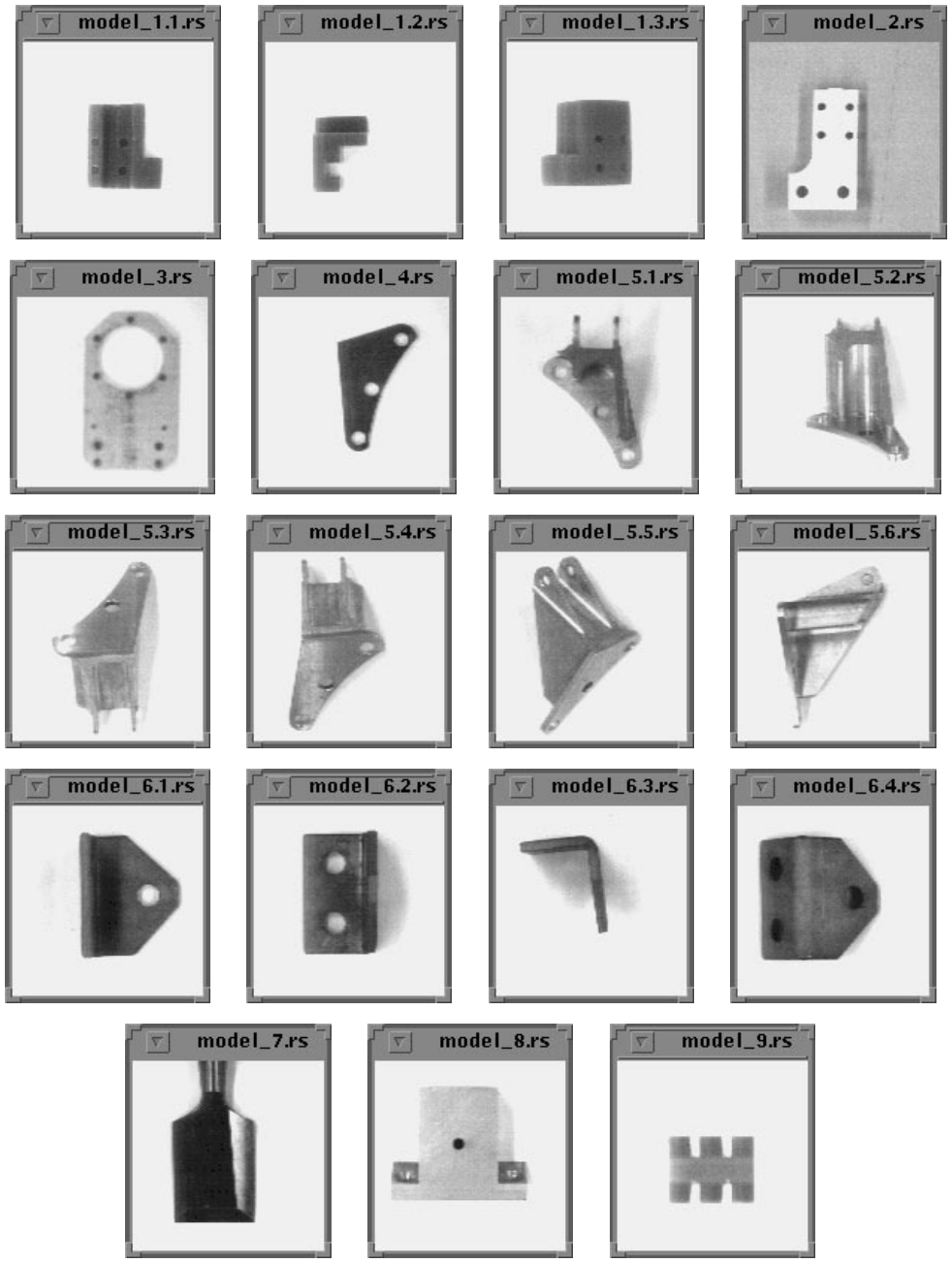

Figure 6.2: The collection of nineteen CAD models.

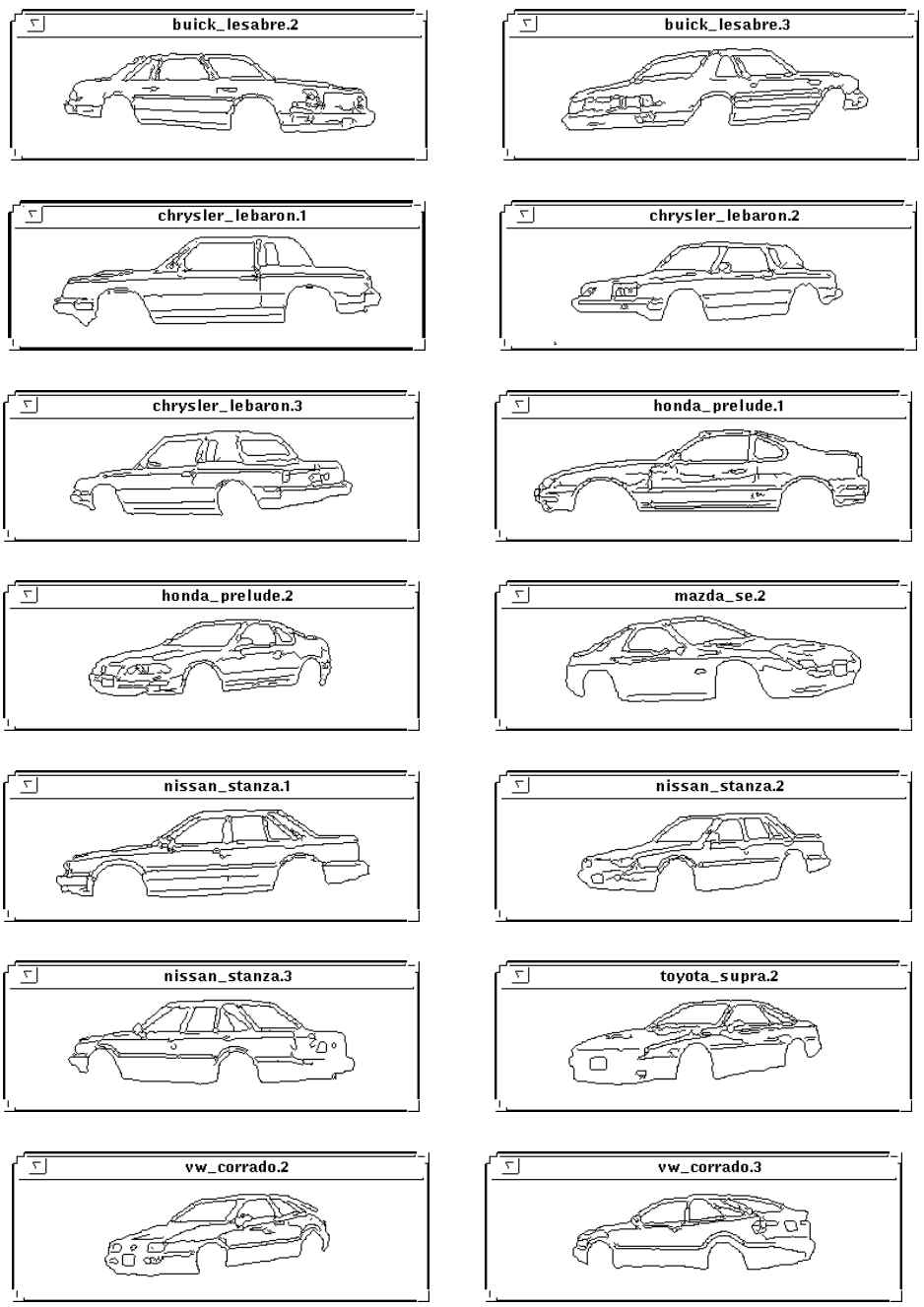

Figure 6.3: The collection of fourteen car models.

Figure 6.4: The collection of three tank models.

6.2 Feature Extraction

Currently our experiments rely on the extraction of line segments which is grouped from edges. To extract edges, we use a version of Cox-Boie edge extractor [10], which we have ported to Khoros. The Cox-Boie edge detector uses a matched filter approach to detect and locate the edges in an image. The input image is convolved with a Gaussian mask and uses four differential edge operators aligned at intervals of $\pi/4$ radians and the maximum response among these four directions is determined. The pixel with the response larger than a threshold signifies that a candidate edge pixel is detected. The location of the edge is found by computing a zero-crossing along the detected edge profile. An improved result can be achieved using a hysteresis analysis that follows. Those pixels above a low threshold that are within a connected segment of contour already marked as edge pixels are also marked as edges. This strategy can achieve the goal of retaining high sensitivity as well as maintaining small probability of marking noise edges. Finally, the last module in the Cox-Boie edge detector is a non-maximum suppression thinning algorithm to ensure that only one response to a single edge is reported. The snapshot of the Cantata screen of an implementation in Khoros is shown in Figure 6.6. The glyph labeled *vfilter* generates the 1-D Gaussian mask. The glyphs *vconv* and *vtranspos* represent convolution and transposition of the 1-D mask respectively. The four glyphs *vdet* represent differential edge operators on four directions. The glyph *vedge* performs zero-crossing edge localization and hysteresis analysis. The last glyph *vthin* represents the non-maximum suppression thinning.

After edge elements are extracted, they are grouped to form line features. The group process uses the following steps:

1. Trace the edges via their 8-connected neighbors. This is implemented by means of the region tracing algorithm given by Pavlidis [83].

2. Line approximation by iteratively splitting the traced segments as long as the max-

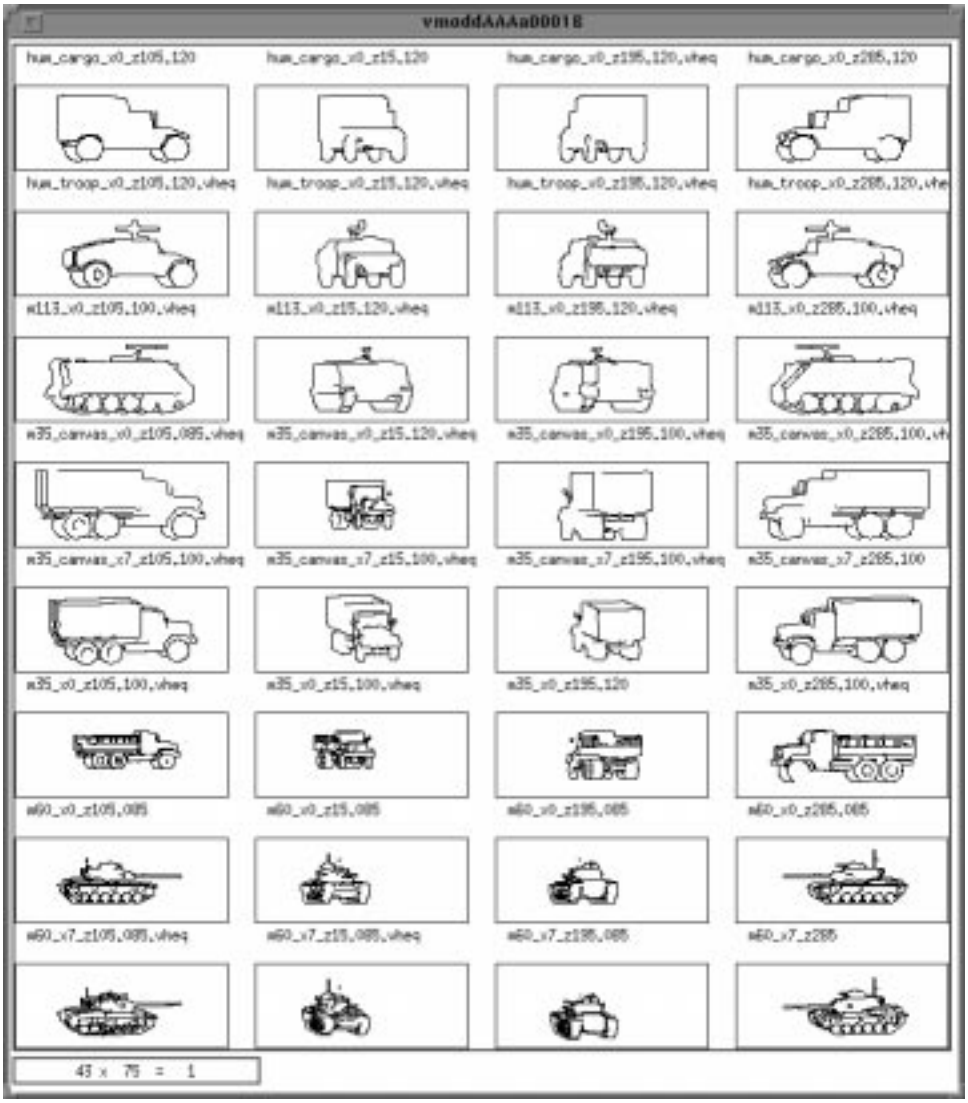

Figure 6.5: The collection of thirty two military vehicle models. The edges are obtained by applying extraction software to ray-traced depth images. Ideally, the depth images would be converted to simulated EO (Electronic Optics) images, but this has not been done for these experiments, and was not necessary to obtain rich edge maps. In some cases, internal edges have been lost.

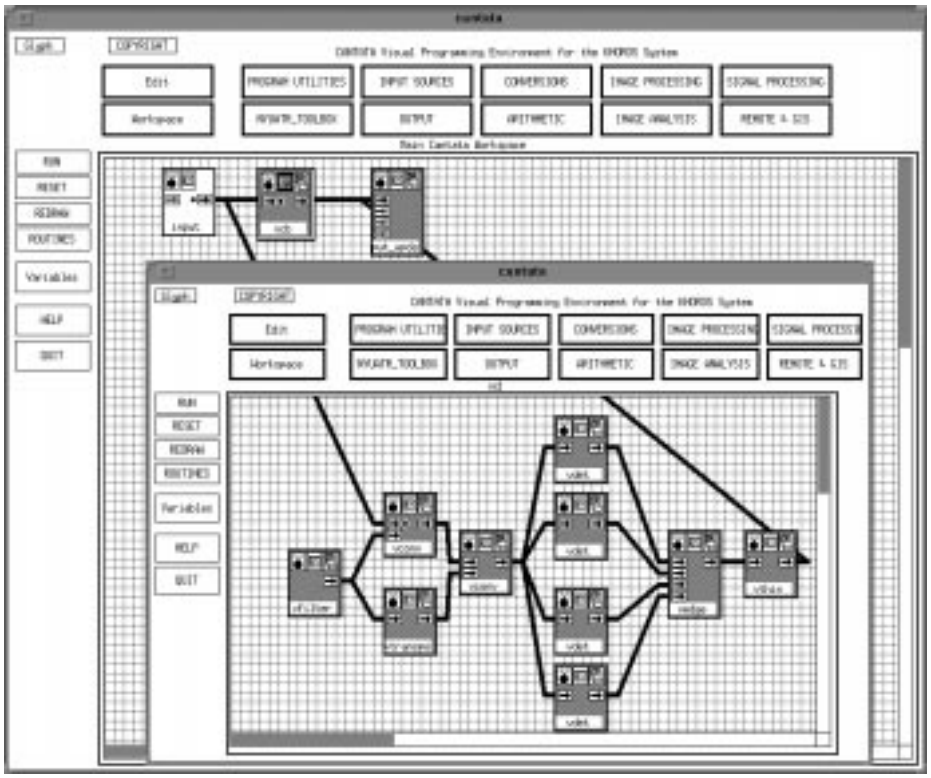

Figure 6.6: The Cox-Boie edge detector implemented as glyphs in Khoros environment.

imum distance deviation is larger than a certain threshold. The distance deviation is defined by the distance of an edge point to the base line which is defined by connecting the endpoints of the segment. This is the conventional iterative "endpoint fits" approach [28, 79].

3. Compute the orientation of each line in a least-squares sense [28, 4] for the segments found in the previous step. Nearby line segments with the same orientations are merged together. Other merging criterion for collinear lines can be found in Venkateswar and Chellappa [110]. Conceptually, the merge process finds the connected components of a graph G, where $G = (E, V)$. Each line segment is represented by a vertex in V. An edge (v_i, v_j) exists if the lines labeled by v_i and v_j satisfies the merging criterion.

4. Compute the least-squares line approximation for each of the merged segments. The endpoints of each line segment are orthogonally projected onto the corresponding fitted line segment. The distance between these two projected points is the extent of the line segment.

The bisector features are obtained by extracting the intersected line features first, then we compute the intersections and the bisectors. The intersected line features are determined using an additional intersection criterion in step 3 above. We build another graph $G' = (E', V)$ for the intersected lines. Recall that a line segment is represented by a vertex in V. If an edge v_i lies within the neighborhood of one of the endpoint of an edge v_j and the slopes of v_i differs than the slope of v_j for a certain amount, then we say that the line segment v_i and the line segment v_j satisfy the intersection criterion so that the edge (v_i, v_j) belongs to E'. The intersections for all the line segments within a connected component of G' are computed.

We also extract circular features. The idea is that instead of fitting the small segments into lines, we fit the segments into circles in a least-squares sense. The merging process combines arcs that are centered at the same position with the same radii. Haralick and Shapiro [45] provide a closed-form formula for fitting circles in the least-squares sense. Essentially, we use the same split-and-merge process 1-4 above, but replace step 3 with a least-squares circle fitting process. The circular features extracted from a snapshot of industrial parts is shown in Figure 6.7. Figure 6.8 shows another example of the extractions of circular features; there is only one wheel that is missed for the target tank M60. The image is a mid-wave infrared image of an M60 tank which is obtained using a Cincinnati Electronics Indium InSb mid-wave IR sensor. It is resampled to increase the sampling rate by a factor of two in both dimensions using bilinear interpolation.

Other methods to extract features can be found in literature. For example, Leung and Huang [67] also study the problem of wheel detection for military vehicle. Stereo

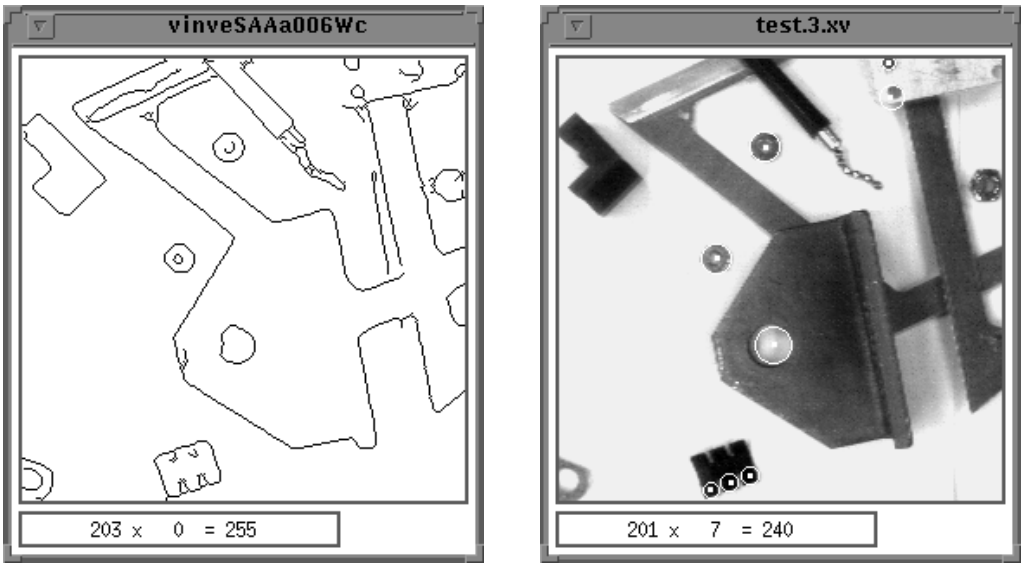

Figure 6.7: The extraction of circular features for industrial parts. The edge map is shown to the left. The detected circles are overlaid on top of the original image which is shown to the right.

images and circle extraction by template matching and Hough transform are used in their experiments. Roth and Levine [95] extract geometric primitives such as lines, circles and ellipses using robust statics. A number of minimal subsets are chosen from the geometric data randomly, followed by the evaluation of each subset based on the cost function. The primitive with the minimum cost is then extracted from the geometric data.

6.3 Target Recognition and Verification

This section contains four sets of experiments. The first set of experiments is based on images of industrial parts. Bisector features are used in the experiment. The left hand side of Figure 6.9 shows the test image with features overlaid on top. The white dots indicate the position of extracted bisectors, associated with line segments to indicate the orientations of the bisectors. The right hand side of Figure 6.9 shows the correct recognition of the industrial part. The approximate matching hypothesis is used, and eighty trials are performed in the experiment. Weighted accumulation according to formula 3.15 and 3.16 is used. Efficient access to hash table entries is performed using the k-d tree construction of section 5.2. The recognized object (i.e., the model-basis with the greatest

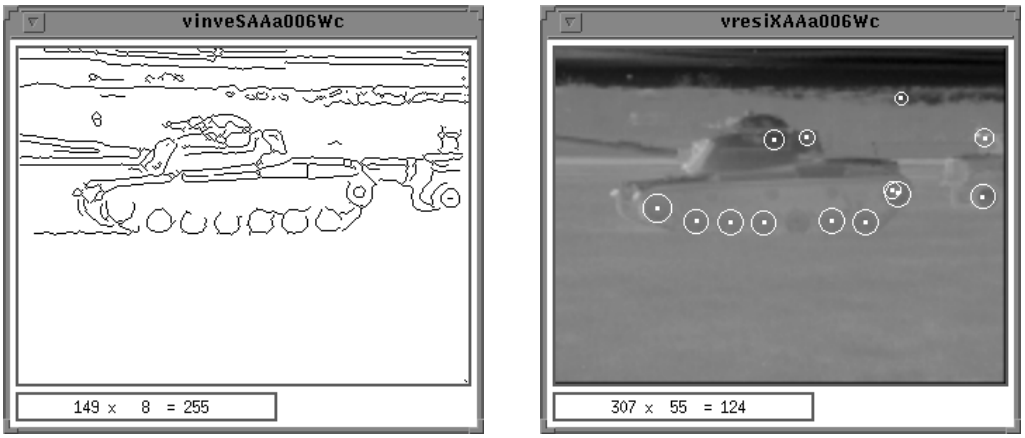

Figure 6.8: The extraction of circular features from mid-wave infrared image of an M60 tank. Again, the edge map is shown to the left and the detected circles are overlaid on top of the original image which is shown to the left.

vote) is indicated by an overlay of the model which is rotated and scaled according to the matched basis. Figure 6.10 shows an array of model/basis pairs that received substantial weighted vote corresponding to this trial. The panels show the sketches of the models with the bars above them to display the magnitude of the vote. The length of the bar is normalized linearly by the following formula:

$$\text{bar\_length} = 0.5p \cdot (1 + \frac{v_i}{\max(v_{\min}, v_{\max})}),$$

where p is the width of the panel, v_i is the vote accumulated for the model, $v_{\min}$ and $v_{\max}$ are the minimum and the maximum of the vote accumulated among the top (model, basis) hypotheses.

Figure 6.11 displays a false alarm. Although it looks like a random match, however, from the given bisector features, the false alarm matches quite well. The accumulated vote, and other statistic information is shown in Table 6.2. The "touched number" refers to the number of the features of the scene that lie within a certain range of the feature for that model. This example demonstrates the power of recognition of the attributed features combined with a verification stage. Even with a small number of features (seven features in both cases), we are able to filter out a number of candidate matches based on a verification step. We use the following strategy to verify the validity of the recognition results. Since we can transform the model to overlap on top of the test image based on the basis information, we can verify the recognition result using the edge information.

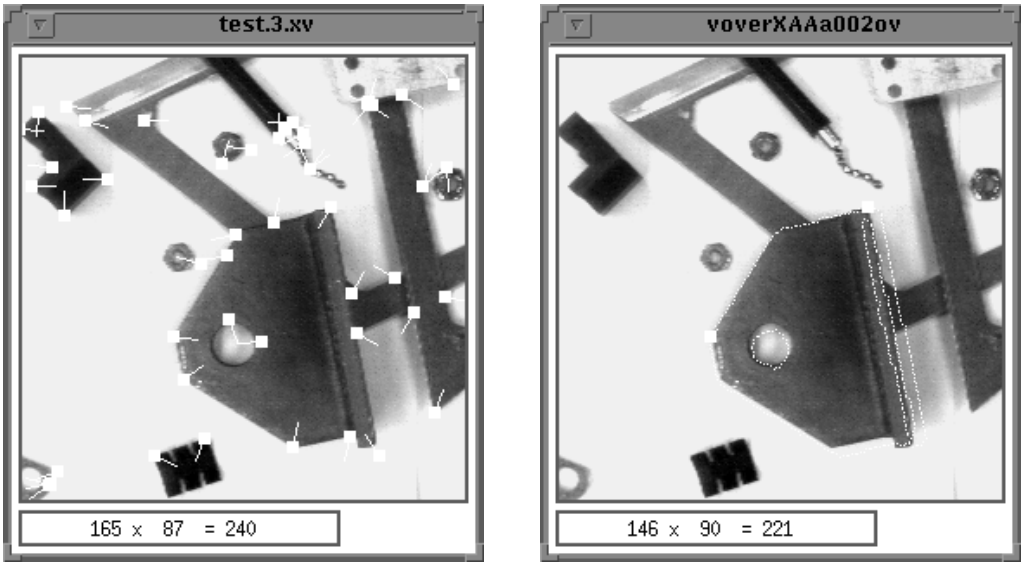

Figure 6.9: Left: the test image with the extracted bisector features overlaid on top. Right: the correct recognition of the industrial part, the edge map of the model which accumulates the largest vote is rotated and scaled according to the matched basis.

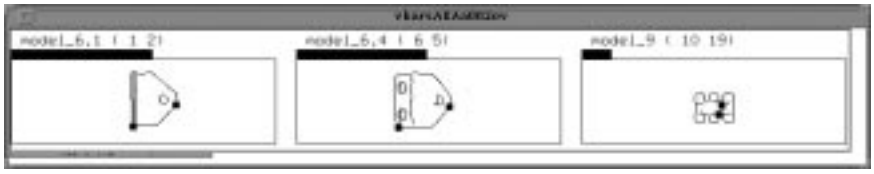

Figure 6.10: The top three model/basis vote-getters for the recognition result. There are only three models which can accumulate vote by using this basis.

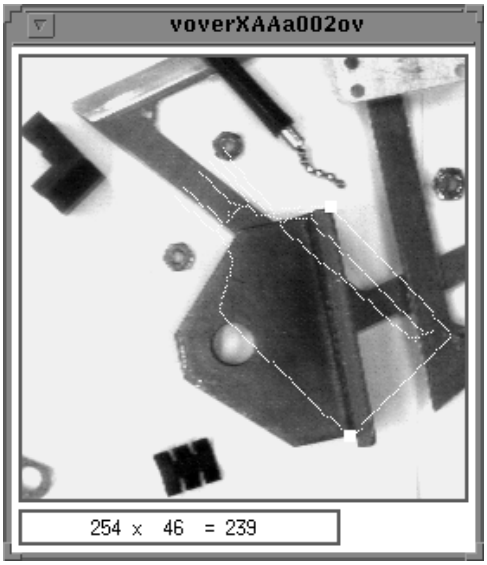

Figure 6.11: A false alarm for the recognition of an industrial part.

remark	vote	touched number	model name	number of features	number of edges	matched edges	matched portion	distance variance
$\surd$	1.0322	2	model_6.1	7	420	363	86.43%	17.50
$\times$	0.5921	2	model_7	7	534	341	63.86%	27.77

Table 6.2: The statistics for the recognition result of industrial part. The search radius is 6 pixels and the penalty factor is 1.5. The mark "$\surd$" indicates a correct recognition, while the mark "$\times$" indicates a false alarm.

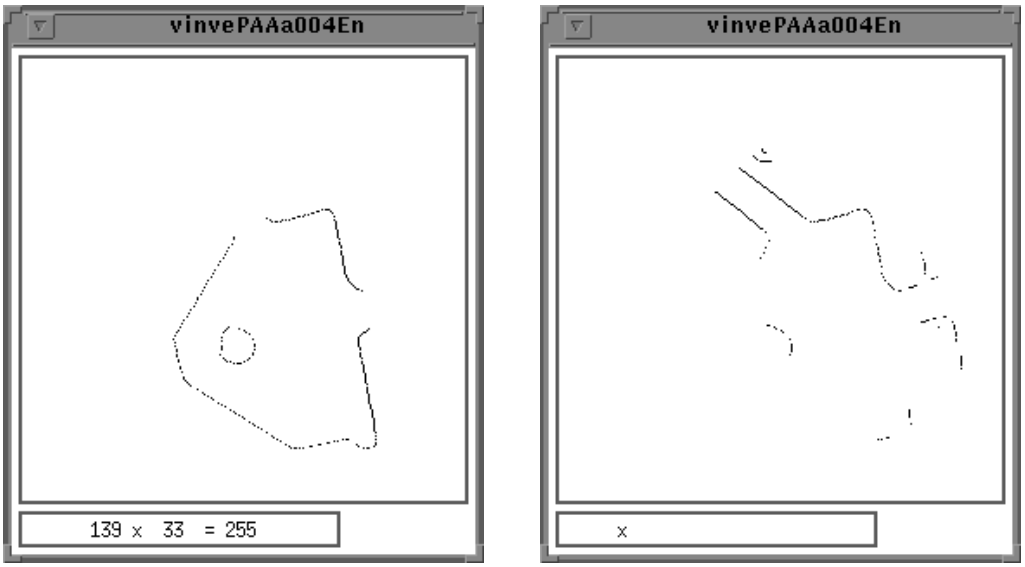

Figure 6.12: Verifications for the recognition of the industrial parts.

For each transformed edge pixel of the model, we search the edge pixels of the test image within a certain bound. The minimum distance between the matched edge pixel from the scene and from the model is computed. The accumulated squared distances together with the matched portion are used as an indicator for the quality of the recognition. An edge pixel m in the model belongs to the *matched portion* if there is an edge pixel of the test image within a predefined search range centered at the edge pixel m. For the distance variance of the unmatched portion, we define a penalty factor which is multiplied by the square of the search radius. Figure 6.12 shows the edge pixels that are matched to the edges in the model with minimum distances for Figure 6.9 and 6.11. Note that the results shown in Figure 6.9 and 6.11 are corresponding to two different trial bases.

Figure 6.13 shows another test image for industrial parts with extracted bisector features. The recognition result for the top vote-getter and the third highest vote-getter corresponding to the same scene basis are shown in Figure 6.15. The length-encoded weight for the top nine model/basis corresponding to the trial with the largest vote is shown in Figure 6.14. In this case, the third and the fourth vote-getters are the correct matches. Because the number of bisector features for each model is usually smaller than other types of features, the descriptive power of the pattern of bisector features alone is quite limited. However, with the verification stage, we can easily reject the false alarms. Corresponding to Figure 6.14, Table 6.3 shows the statistics of the verification stage

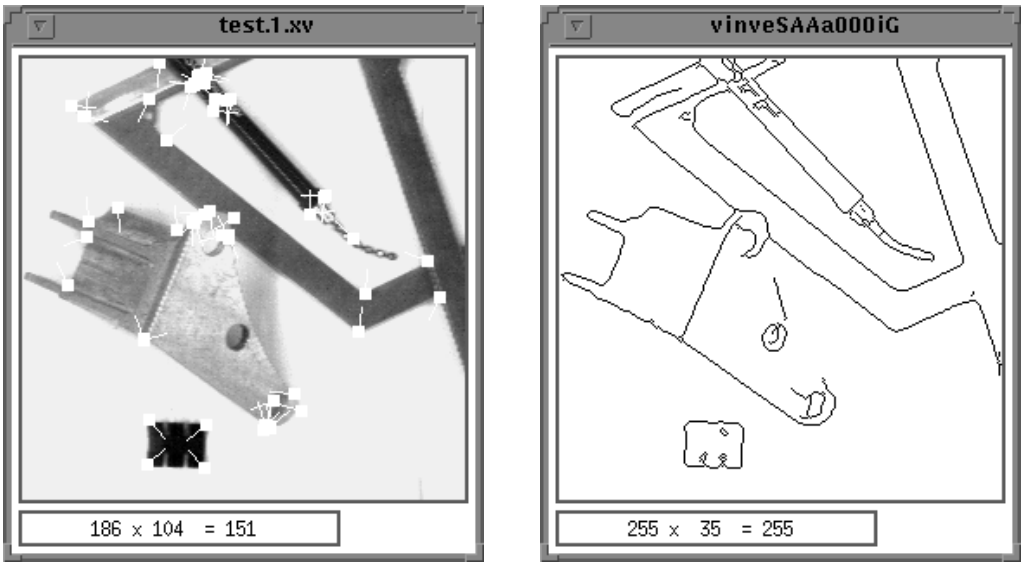

Figure 6.13: Another test image for industrial parts. Left: the extracted bisector features overlaid on top of the original image. Right: the edge map.

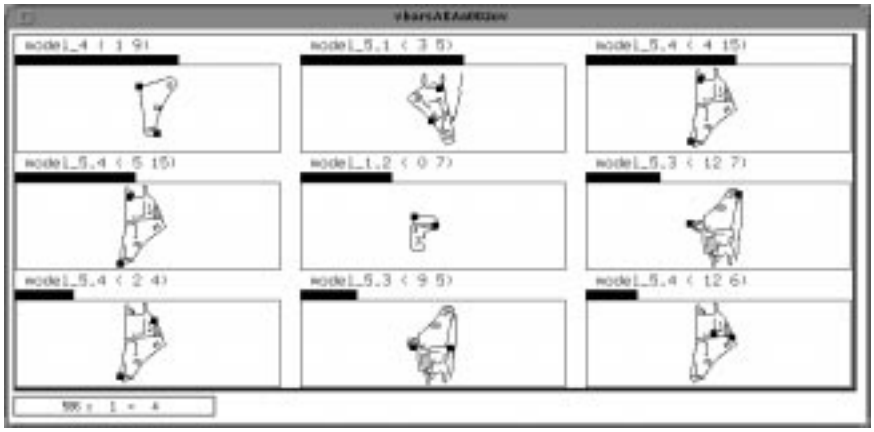

Figure 6.14: The top nine model/basis vote-getters for the recognition result of industrial parts. The top vote-getter is a false alarm. The third vote-getter is the correct recognition.

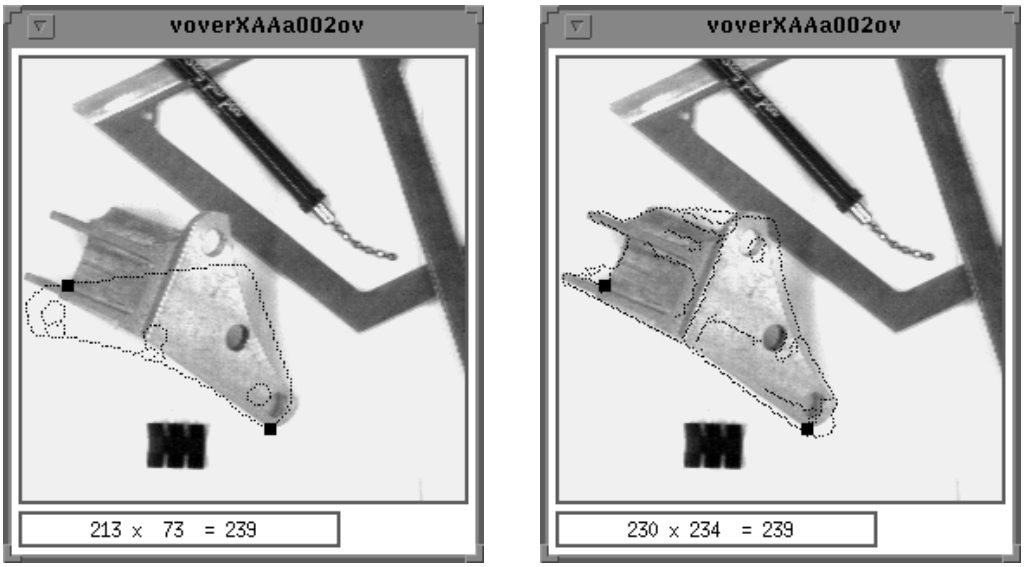

Figure 6.15: The recognition result for the first and the third vote-getters. The third vote-getter is the correct match.

for top nine candidates corresponding to this trial basis. Note that the percentage of the matched portion for the correct matches are 81.07% and 82.96% respectively. The distance variances for the correct matches are 16.77 and 15.87 pixels. This information can be used to reject the incorrect matches easily.

Figure 6.16 shows a test image with midpoint features overlaid on top of the original image for the recognition of commercial vehicles. The extracted features are shown as square dots, with a line segment passing through them in order to indicate the orientation information. The edge extracted from the image is shown to the right of Figure 6.16. Figures 6.17 and 6.18 show the recognition result.

To demonstrate the scale invariant capabilities, we shrink the image down to sixty percent on each side and embed the image of one car in the image of another car, as shown by Figure 6.19. The number of pixels on target is reduced from 43000 pixels to 15000 pixels, roughly. The correct car model, a Buick LeSabre, is still recognized at the correct location, which is shown in Figure 6.20 and 6.21. Further reduction in size often results in failed recognition. The limitation on recognition ability comes from the power of feature extraction. When reliable features are undetectable due to the small size of the object in the image, the recognition system may fail even though we use features with descriptive attributes. Note that the signal to noise ratio (the number of features in Buick

remark	vote	touched number	model name	number of features	number of edges	matched edges	matched portion	distance variance
×	2.1464	3	model_4	10	283	137	48.41%	32.67
×	2.1464	3	model_5.1	10	604	144	23.84%	44.04
√	1.2343	8	model_5.4	20	581	471	81.07%	16.77
√	-0.8259	7	model_5.4	20	581	482	82.96%	15.87
×	-2.8092	1	model_1.2	10	232	114	49.14%	34.22
×	-3.9826	1	model_5.3	16	737	441	59.84%	29.09
×	-5.1182	1	model_5.4	20	581	247	42.51%	36.72
×	-5.1995	1	model_5.3	16	737	230	31.21%	40.91
×	-5.5628	1	model_5.4	20	581	68	11.70%	49.39

Table 6.3: The statistics for another recognition result of industrial part. Again, the search radius is 6 pixels and the penalty factor is 1.5. The data shown is for the top nine candidates corresponding to the same trial basis.

Figure 6.16: A test image, Buick LeSabre. Left: the test image with extracted midpoint features overlaid on top. Right: the extracted edge map.

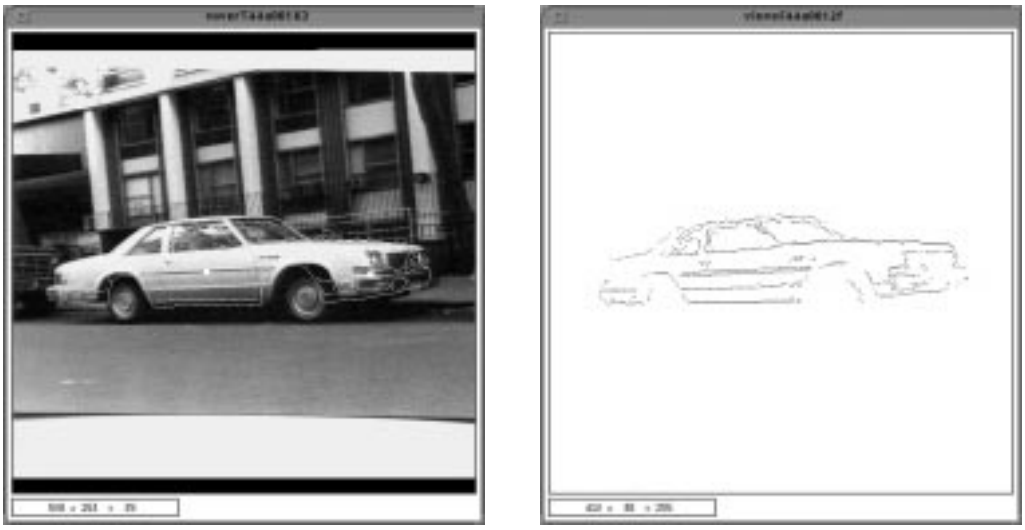

Figure 6.17: The recognition result of the top vote-getters for Buick LeSabre. Left: the test image with recognized model overlaid on top. Right: the result of verification.

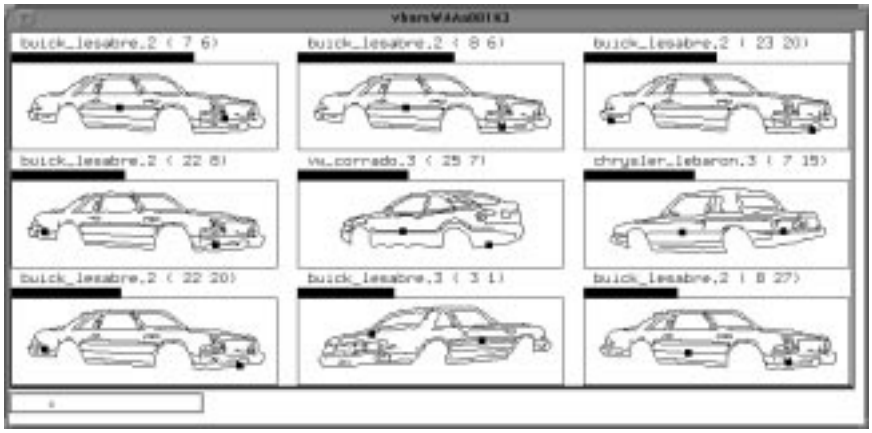

Figure 6.18: The top nine model/basis vote-getters for the recognition result of Buick LeSabre.

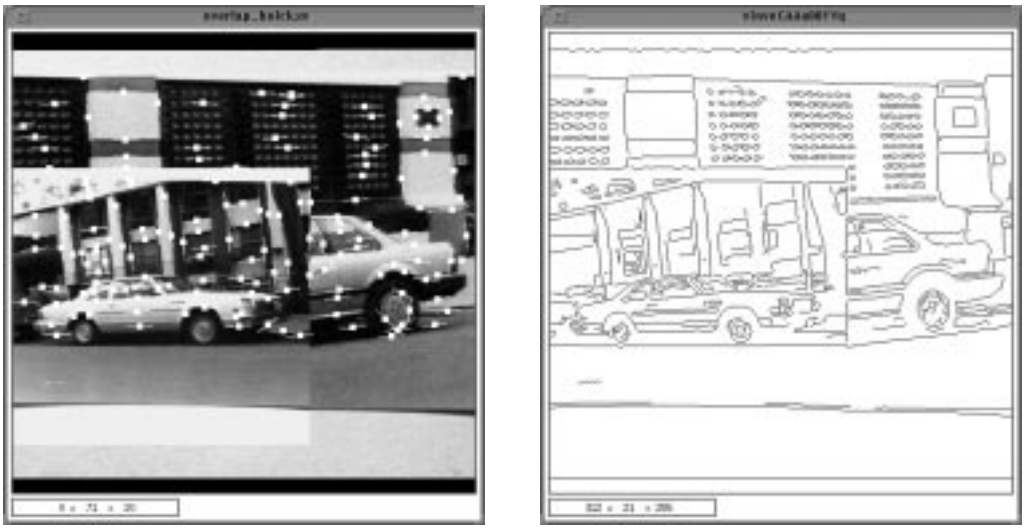

Figure 6.19: Another test image, the picture is shrunk and embedded into another image. Left: the test image with extracted midpoint features overlaid on top. Right: the extracted edge map.

LeSabre divided by the number of features in the background) in Figure 6.20 is quite low (28 / 166 = 0.18767). The success of the recognition shows that if we can extract stable features from the images, our recognition scheme is able to recognize objects even when the background is quite complicated.

Next we discuss results for recognizing military vehicles. An original image is shown in Figure 6.22. For the target tank, an M60, there are roughly 2064 pixels on target (86 by 24) in the original image. In order to extract stable features without modification of the existing feature extractors, we extract a portion of the original image and enlarge the image by three times using bilinear interpolation on each side. The resulting image with the extracted endpoint features overlaid on top is shown in Figure 6.23. With more sophisticated multiresolution feature extractors, this result could be obtained without rescaling the image. The recognition result is shown in Figure 6.24. Figure 6.25 shows the top vote-getters among the various (model/basis) hypotheses.

Figure 6.26 shows a false alarm. In this case, a tank has been recognized in the upside-down configuration which can be rejected by further filtering. The verification process for the correct recognition of the target and the incorrect recognition are shown in Figure 6.27. Both recognitions pass this particular verification process. The matching statistics for these two cases are listed in Table 6.4, showing that both candidates

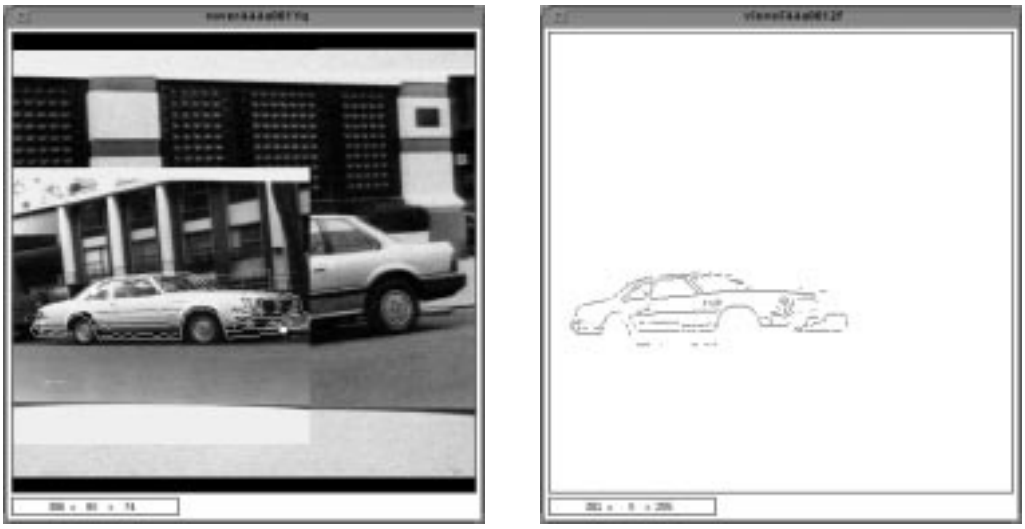

Figure 6.20: The recognition result of the top vote-getters for the embedded Buick LeSabre. Left: the test image with recognized model overlaid on top. Right: the result of verification.

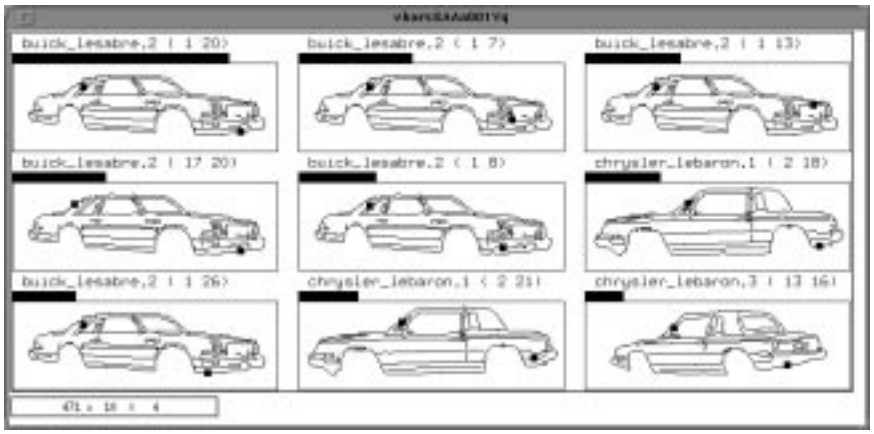

Figure 6.21: The top nine candidates for the recognition result of the embedded Buick LeSabre.

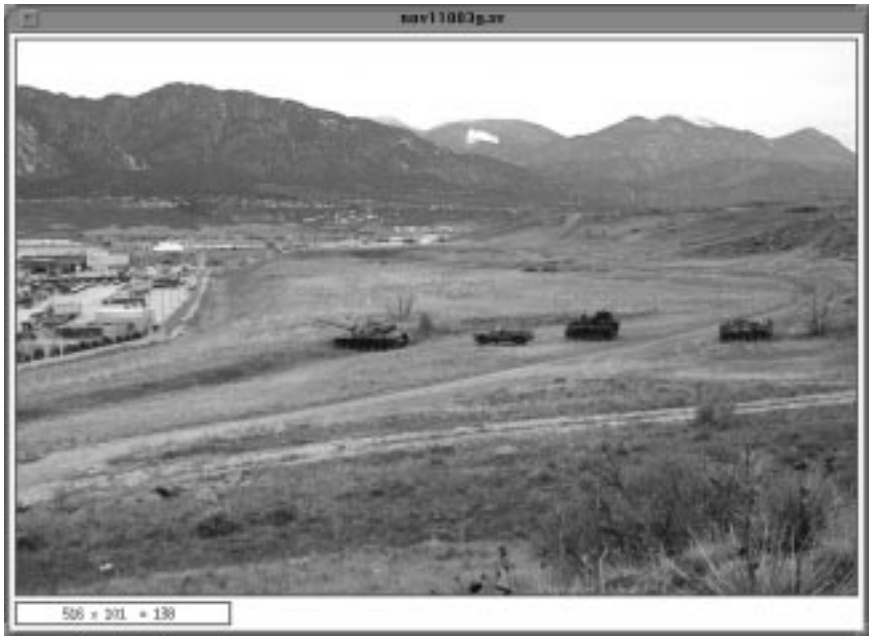

Figure 6.22: The original test image for military vehicles.

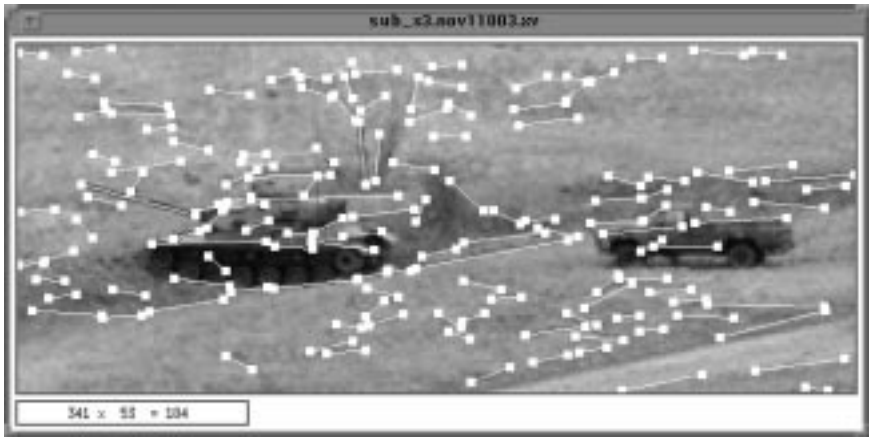

Figure 6.23: The features extracted from the original image resized by a factor of three on each side.

Figure 6.24: The recognition result for an M60 tank.

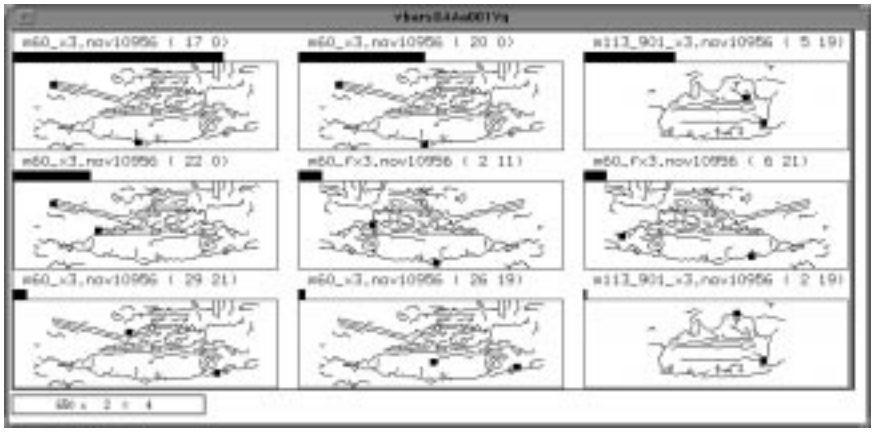

Figure 6.25: The top nine candidates for the recognition result of the military vehicle.

Figure 6.26: A false alarm for the recognition of an M60 tank.

(including the incorrect one) that passed these statistical filters.

Finally, the last experiment demonstrates the power of using hybrid features. The feature types that we currently use are midpoints and circles. The test image and the images for model database are from different sources, as discussed in Section 6.1. Therefore, the pixel values in the images have different meanings. For laser radar image, the pixel values represent the distance from the observer, while in the infrared image, each pixel value represents the temperature. We simply make use of the discontinuity information of these two different physical properties regardless of their original physical meaning. The size of the original test image is 160 by 120 pixels with roughly 4000 pixels on the target

Figure 6.27: The verification for the recognition of military vehicle. Left: a correct recognition. Right: a false alarm.

remark	vote	touched number	model name	number of features	number of edges	matched edges	matched portion	distance variance
$\surd$	4.4420	17	M60	32	2648	2615	98.75%	4.58
$\surd$	-0.3663	16	M60	32	2648	2648	99.02%	7.12
$\times$	-2.3446	7	M113	22	1170	610	52.14%	30.75
$\times$	-3.1363	11	M60	32	2648	1810	68.35%	23.87
$\times$	-6.2055	7	M60 (flip)	22	2689	1378	51.25%	30.98
$\times$	-6.2602	5	M60 (flip)	22	2689	1859	69.13%	23.72
$\times$	-6.8075	12	M60	32	2648	1979	74.74%	22.07
$\times$	-7.2112	7	M60	32	2648	1247	47.09%	33.27
$\times$	-7.4540	3	M113	32	1170	819	70.00%	23.11
$\times$	2.9743	19	M60	32	2648	1911	72.17%	22.24

Table 6.4: The statistics for the recognition result of military vehicle. The search radius is 6 pixels and the penalty factor is 1.5. The data shown for the first nine rows is for the top nine candidates corresponding to the same trial basis. The last row is the statistics for the false alarm corresponding to the wrong trial basis.

(100 pixels by 40 pixels). In order to extract stable features using our feature extractors, we enlarge the original image by two times on each side using bilinear interpolation. The resulting image with the extracted hybrid features overlaid on top is shown in Figure 6.28. The recognition result and the result of verification are shown in Figure 6.29. Figure 6.30 shows the top vote-getters among the model/basis hypotheses. The matching statistics are listed in Table 6.5. In this experiment, we use the exact matching hypothesis, and the non-obscuration ratio equals 0.5. Note that even for the correct matching, the number of features that contribute to this recognition is 9 out of 28 features. If we expect a very high non-obscuration ratio, say, 0.9, but in reality we have less than half of the features matched, then the recognition result may be incorrect. This means that we are looking for the model with 90% features matched, but a much lower percentage of features actually match. In this case, the bias term can dominate the vote. The impact of different parameter settings on the bounds of weighted voting has been discussed in Section 3.4. Table 6.6 lists the matching statistics with the non-obscuration ratio set to 0.9. Giving the same trial basis pair, the correct model appears in the third place. Furthermore, there is another trial basis that produces an incorrect recognition with a higher total vote than the correct one. However, there is only one feature that contributes to the total vote in this process, and we conclude that the bias term favors models with fewer features.

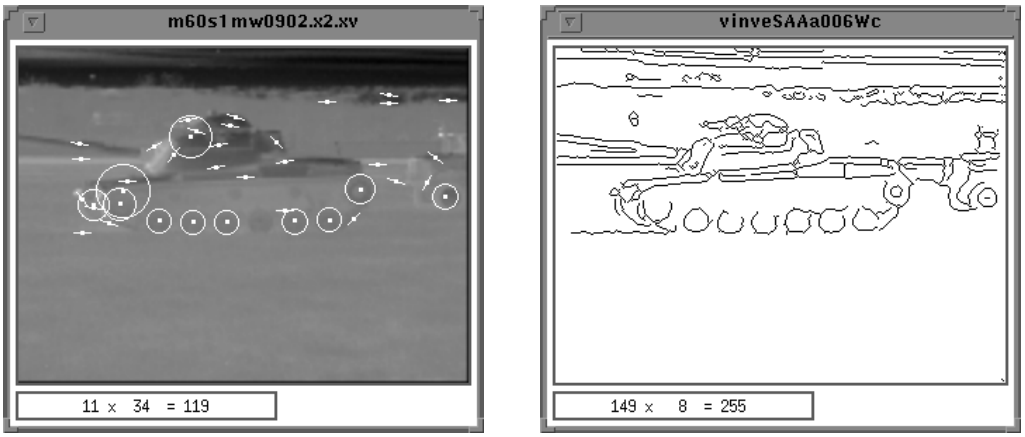

Figure 6.28: The test image for military vehicles. Left: the test image with extracted midpoint features and circles overlaid on top. Right: the extracted edge map.

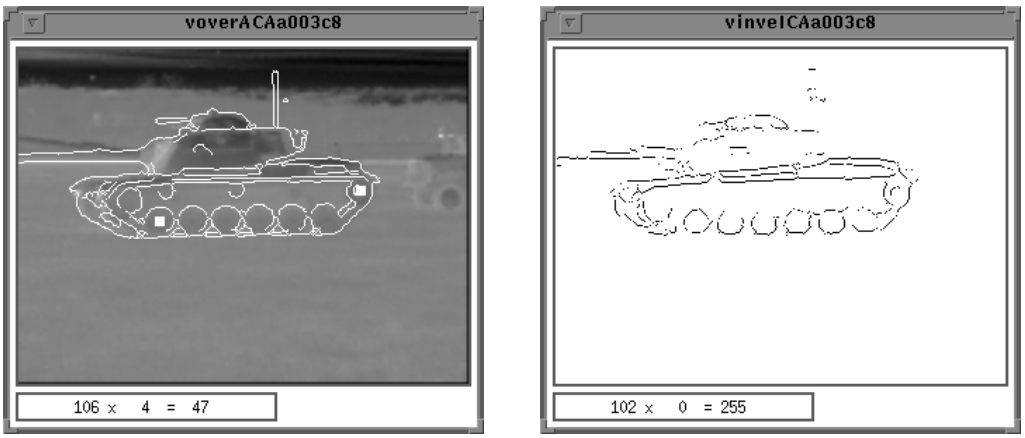

Figure 6.29: The recognition result of an M60 tank. Left: the test image with recognized model overlaid on top. Right: the result of verification.

remark	vote	touched number	model name	number of features	number of edges	matched edges	matched portion	distance variance
$\surd$	-0.9702	9	M60 (0, 285)	28	2602	2435	93.58%	8.97
$\times$	-6.5728	1	M113 (0, 195)	16	983	272	27.67%	41.67
$\times$	-7.7181	1	M113 (0, 195)	16	983	274	27.87%	41.35
$\times$	-8.0811	2	Hum (T) (0, 15)	20	1031	513	49.76%	32.07
$\times$	-8.2363	2	M35 (C) (0, 285)	23	1301	827	63.57%	25.05
$\times$	-9.1707	2	M113 (0, 105)	24	1326	1026	77.38%	18.74
$\times$	-9.7398	3	M113 (0, 105)	24	1326	580	43.74%	34.12
$\times$	-10.0345	1	M113 (0, 105)	24	1326	589	44.42%	33.31
$\times$	-10.3548	1	M113 (0, 105)	24	1326	979	73.87%	21.55

Table 6.5: The statistics for the recognition result of military vehicles. Again, the search radius is 6 pixels and the penalty factor is 1.5. The data shown for the first nine rows is for the top nine candidates corresponding to a single trial basis. The model M35 (C) represents an M35 truck with canvas attached. Hum (T) represents Humvee troop vehicle. The number pair below each model name represents the (tilt, pan) angle pair. The vote is computed by using the exact matching hypothesis with non-obscuration ratio β equal to 0.5.

remark	vote	touched number	model name	number of features	number of edges	matched edges	matched portion	distance variance
×	-11.5198	1	Hum (C) (0, 195)	13	762	391	51.31%	14.84
×	-11.6601	1	M113 (0, 195)	16	983	453	46.08%	15.07
×	-13.6062	1	M113 (0, 195)	16	983	233	23.70%	19.72
√	-13.9617	9	M60 (0, 285)	28	2602	2297	88.28%	6.66
×	-15.3147	1	M113 (0, 195)	16	983	242	24.62%	19.49
×	-18.2008	2	Hum (T) (0, 15)	20	1031	422	40.93%	16.33
×	-20.5822	2	M35 (C) (0, 285)	23	1301	717	55.11%	13.45
×	-22.6215	1	Hum (T) (0, 285)	22	1341	916	68.31%	11.11
×	-23.3255	2	M113 (0, 105)	24	1326	502	37.86%	16.80
×	-23.3775	2	Hum (T) (0, 285)	22	1341	390	29.08%	18.66
×	-23.4678	1	Hum (T) (0, 285)	22	1341	354	26.40%	19.12

Table 6.6: The statistics for the recognition result of military vehicles. Again, the search radius is 6 pixels and the penalty factor is 1.5. The data shown for the last nine rows is for the top nine candidates corresponding to the same trial basis. The model Hum (C) represent Humvee cargo vehicle. The vote is computed by using exact matching hypothesis with non-obscuration ratio β equals 0.9. The correct model appears at the third place.

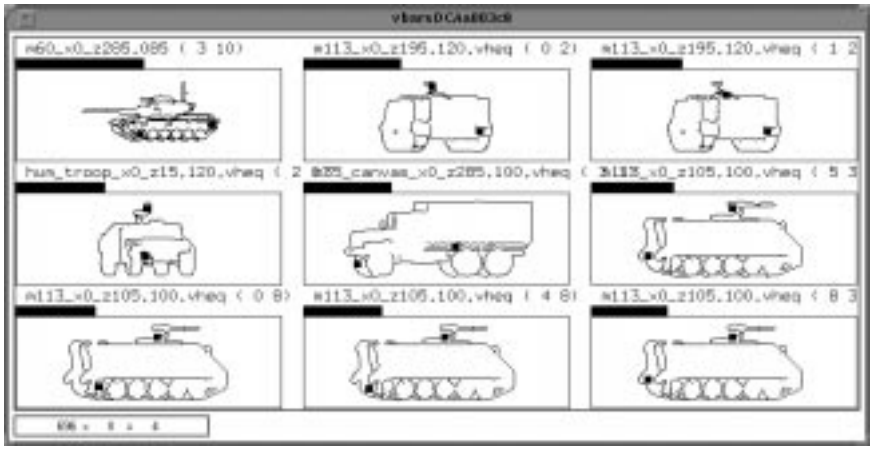

Figure 6.30: The top nine candidates for the recognition result of military vehicle.

Chapter 7

Discussion

We have extended the field of object recognition with geometric hashing in many ways. In Chapter 3 we discussed the generalization to different numbers of features for each model. The use of attributed features is discussed in Chapter 4. Efficient and distributed implementation is addressed in Chapter 5. Our experiments shows that good recognition relies on good detection of features. Future work of target recognition can be explored in the following ways.

7.1 Multiresolution Feature Extraction

In our experiments, we experienced difficulty with target recognition of very small objects embedded in complicated backgrounds. Although the recognition scheme can handle objects at different scales, if the feature extractors can not detect stable features from the targets, the recognition system will fail. In some of our experiments, we had to rescale the entire image in order to obtain stable features. Alternatively, we could have changed parameters in the feature extractors. We could instead design multiresolution feature extractors and use a coarse-to-fine strategy to locate candidate locations for features. Although our feature extractors in this project worked only at a fixed resolution, a multiresolution feature extraction process can tag features with a scale factor. Then, by transforming the scale factor according to the scale of the scene basis, the scale factor can be incorporated into the matching process.

7.2 Design of Attributed Features

The use of attributed features in geometric hashing enhances the discrimination power of this method. The experiments using midpoints of line segments with the orientation as the attribute and high curvature points with bisector directions as the attribute are two realizations of this concept. We consider other realizations so that curved objects can

also be recognized. What criteria are appropriate to judge attributed features that can be applied to our method? We suggest the following design criteria:

- Conciseness—The representation of the feature should be concise so that we can compute variations easily. We want to keep the dimension of the attribute low.
- Robustness—The representation should be robust to noise and occlusion.
- Positional information—To fit into our method, the representation should carry positional information. While non-positional features are possible, they are likely to be less stable and trickier to associate.
- Ease of computation—We need to compute the normalized features efficiently. In our case, we want to compute the similarity transformed version of the attribute efficiently. For example, we need to subtract the angle induced by the similarity transformation for each orientation attribute. For a length attribute, we multiply by the scaling factor. For an area attribute, we can multiply by the square of the scaling factor.
- Description power of the representation—Although we do not require that the representation uniquely identifies the entity that it describes, we do require the representation describes the entity to some extent, providing some degree of filtering of the candidate models and bases.

Applying these criteria, the following ideas for other kinds of attributed features may be explored in the future. The goal is to describe general 3-D objects with curved features.

- Ellipse—We can use the center of the ellipse as the point feature, and the axis lengths of the ellipse are its attribute. The normalization of the attribute is accomplished simply by multiplying the axis lengths with a scaling factor.
- Closed curve—We can use a first order moment as the point feature and use higher order moments, say, second order moments as the attribute. Suppose (x_i, y_i) are the coordinates of points along the curve. The moment of order $(p+q)$ of the curve can be defined as:

  $$m_{pq} = \frac{1}{n} \sum_i x_i^p y_i^q,$$

  where n is the number of points along the curve. (We can assume that the origin $(x, y) = (0, 0)$ is placed at the center of mass, so that $m_{10} = m_{01} = 0$ in all cases.) Assume m'_{pq} is the corresponding $(p + q)$ moment of the curve transformed by a

similarity transformation T. (Again, the origin is moved so that $m'_{10} = m'_{01} = 0$.) We have

$$\begin{aligned}(m'_{10}, m'_{01}) &= T(m_{10}, m_{01}) \\ m'_{20} &= s^2[m_{20}\cos^2\theta + m_{02}\sin^2\theta] \\ m'_{02} &= s^2[m_{20}\sin^2\theta + m_{02}\cos^2\theta],\end{aligned}$$

where θ and s are the rotation angle and scaling factor of T.

- Curve segment—We can use the following information: the maximum distance from the point along the curve to the base line, the length of the base line, the relative location of the projection of the point of maximum distance from the base line onto the base line. The base line is defined as the line connecting two end points of the curve.

7.3 Other Invariants

Our experiments is the coordinates of normalized features as the invariants. It is possible to use other kinds of invariants. The criteria for choosing invariants are the same as described in the previous section, and that the new invariant should be insensitive to noise. The invariant should not involve too many features, so that the probability of choosing combinations of features in the scene matching combinations in the model are reasonable. In our case, suppose there are m features in the target among s scene features. Then the probability of choosing one correct basis pair is $m(m-1)/s(s-1)$. The probability of hitting one of the correct basis pair is proportional to the square of signal-to-noise ratio.

The other consideration of choosing invariants is the ability of describing various objects. The invarint should be rich enough so that it can be applied to many application areas. An invariant with many restrictions are not useful.

7.4 Aspect Graph Considerations

Another direction of improving the recognition system is to automate the model building process. Suppose that we have complete 3-D geometric data of an object. We can simulate various sensors and predict the features that will be detected in various directions for the test image. As successive models are built, they should be tested for recognition in the existing database. The model database is built incrementally by adding those models which can not be recognized in the current model database. In other words, aspects with similar models will be fused into a single model. A feeback loop can be formed by combining the recognition module and model building module.

7.5 Parameter Selection

There are several parameters need to be set during the recognition process. For example, the covarance of the extracted features and the non-obscuration ratio β in Equation 3.7. If these parameters are not set properly, the recognition result may be incorrect. The impact of different parameters setting has been discussed in Section 3.4. Automated selection of these system parameters would enhance the object recognition system, and deserves further study.

Bibliography

[1] John Y Aloimonos. Perspective approximations. *Image and Vision Computing*, 8(3):179–192, Aug. 1990.

[2] Klaus Arbter, Wesley E. Snyder, Hans Burkhardt, and Gerd Hirzinger. Application of affine-invariant fourier descriptors to recognition of 3-D objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):640–647, July 1990.

[3] Nicholas Ayache and Francis Lustman. Trinocular stereo vision for robotics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):73–85, Jan. 1991.

[4] Dana H. Ballard and Christopher M. Brown. *Computer Vision*. Prentice-Hall Inc., 1982.

[5] Stephen T. Barnard and William B. Thompson. Disparity analysis of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(4):333–340, July 1980.

[6] Eamon B. Barrett, Paul M. Payton, Nils N. Naag, and Michael H. Brill. General methods for determining projective invariants in imagery. *CVGIP: Image Understanding*, 53(1):46–65, Jan. 1991.

[7] Eamon B. Barrett, Paul M. Payton, Nils N. Naag, and Michael H. Brill. General methods for determing projective invariants in imagery. *CVGIP: Image Understanding*, 53(1):46–65, Jan. 1991.

[8] Jon Louis Bentley. Multidimensional binery search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, Sep. 1975.

[9] Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *ACM Computing Surveys*, 17(1):75–145, Mar. 1985.

[10] Robert A. Boie, Ingemar J. Cox, and Pavel Rehak. On optimum edge recognition using matched filters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 100–108, June 1986.

[11] J. Brain Burns, Richard Weiss, and Edward M. Riseman. View variation of point set and line segment features. In *Proceedings of the DARPA Image Understanding Workshop*, pages 650–659, 1990.

[12] Edited by Joseph L. Mundy and Andrew Zisserman. *Geometric Invariance in Computer Vision*. MIT press, Cambridge, Massachusetts, 1992.

[13] A. Califano. Feature recognition using correlated information contained in multiple neighborhoods. In *Seventh AAAI 2*, pages 831–836, 1988.

[14] Andrea Califano and Rakesh Mohan. Multidimensional indexing for recognizing visual shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 1991.

[15] Andrea Califano and Rakesh Mohan. Multidimensional indexing for recognizing visual shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):373–392, Apr. 1994.

[16] Homer H. Chen. Determining motion and depth from bincular orthographic views. *CVGIP: Image Understanding*, 54(1):47–55, July 1991.

[17] Homer H. Chen and Thomas S. Huang. Matching 3-D line segments with applications to multiple-object motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):1002–1008, Oct. 1990.

[18] Jin-Long Chen and George C. Stockman. Matching curved 3D object models to 2D images. In *Proceesdings of the Second CAD-Based Vision Workshop*, pages 210–218, Champion, Pennsylvania, Feb. 1994.

[19] Jin-Long Chen, George C. Stockman, and Kashi Rao. Recovering and tracking pose of curved 3D object from 2D images. In *Computer Vision and Pattern Recognition*, pages 393–399. IEEE Computer Society, June 1993.

[20] William Chen and Bernard C. Jiang. 3-D camera calibration using vanishing point concept. *Pattern Recognition*, 24(1):57–67, 1991.

[21] Zen Chen and Shinn-Ying Ho. Computer vision for robust 3D aircraft recognition with fast library search. *Pattern Recognition*, 24(5):375–390, 1991.

[22] Roland T. Chin and Charles R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1):67–108, March 1986.

[23] Sheng-Lin Chou and Wen-Hsiang Tsai. Line segment matching for 3D computer vision using a new iteration scheme. *Machine Vision and Applications*, 6:191–205, 1993.

[24] Chen-Chau Chu and J. K. Aggarwal. The integration of image segmentation maps using region and edge information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1241–1249, Dec. 1993.

[25] David T. Clemens and David W. Jacobs. Space and time bounds on indexing 3-D models from 2-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1007–1017, Oct. 1991.

[26] Michel Dhome, Marc Richetin, Jean-Thierry Lapresté, and Gérard Rives. Determination of the attitude of 3-D objects from a single perspective view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, Dec. 1989.

[27] Umesh R. Dohond and J. K. Aggarwal. Structure from stereo–a review. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1510, Dec. 1989.

[28] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[29] Ting-Jun Fan, Gerard Medioni, and Ramakant Nevatia. Recognizing 3-D objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, Nov. 1989.

[30] J.-Q. Fang and Thomas S. Huang. Solving three-dimensional small-rotation motion equations: Uniqueness, algorithms, and numerical results. *Computer Vision, Graphics, and Image Processing*, 26:183–206, 1984.

[31] Jia-Qi Fang and Thomas S. Huang. Some experiments on estimating the 3-D motion parameters of a rigid body from two consecutive image frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(5):545–554, Sep. 1984.

[32] John C. Fiala, Ronald Lumia, Karen J. Roberts, and Albert J. Wavering. TRICLOPS: A tool for studying active vision. *International Journal of Computer Vision*, 12(2/3):231–250, Apr. 1994.

[33] Patrick J. Flynn and Anil K. Jain. 3D object recognition using invariant feature indexing of interpretation tables. *CVGIP: Image Understanding*, 55(2):119–129, Mar. 1992.

[34] David Forsyth, Joseph L. Mundy, Andrew Zisserman, Chris Coelho, Aaron Heller, and Charles Rothwell. Invariant descriptors for 3-D object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, Oct. 1991.

[35] Geoffrey C. Fox, Roy D. Williams, and Paul C. Messina. *Parallel Computing Works*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1982.

[36] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, Sep. 1977.

[37] D. M. Gavrila and F. C. A. Groen. 3D object recognition from 2D images using geometric hashing. *Pattern Recognition Letters*, 13(4):263–278, Apr. 1992.

[38] W. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):17–34, Jan. 1985.

[39] W. Grimson and D. Huttenlocher. On the sensitivity of geometric hashing. In *International Conference on Computer Vision*, pages 334–338, Osaka, Japan, Dec. 1990.

[40] W. Grimson and Tomás Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, July 1987.

[41] William I. Grosky and Louis A. Tamburino. A unified approach to the linear camera calibration problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):663–671, July 1990.

[42] André Guéziec and Nicholas Ayache. Smoothing and matching of 3-D-space curves. In *Second European Conference on Computer Vision*, Santa Margherita Ligure, Italy, May 1992.

[43] André Guéziec and Nicholas Ayache. New developments on geometric hashing for curve matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, New York, Jun. 1993.

[44] André Guéziec and Nicholas Ayache. Smoothing and matching of 3-D space curves. *International Journal of Computer Vision*, 12(1):79–104, Feb. 1994.

[45] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*, volume 1-2. Addison-Wesley Publishing Company, New York, 1992.

[46] William Hoff and Narendra Ahuja. Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):121–136, Feb. 1989.

[47] Robert V. Hogg and Allen T. Craig. *Introduction to Mathematical Statistics.* Macmillan Publishing Co., Inc., NewYork, fourth edition, 1978.

[48] Robert J. Holt and Arun N. Netravali. Camera calibration problem: Some new results. *CVGIP: Image Understanding*, 54(3):368–383, Nov. 1991.

[49] Jiawei Hong and Haim J. Wolfson. An improved model-based matching method using footprints. In *Proceedings of the 9th International Conference on Pattern Recognition*, pages 72–78, 1988.

[50] Radu Horaud, Bernard Conio, and Olivier Leboulleux. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47:33–44, 1989.

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

[52] J. Illingworth and J. Kittler. A survey of the Hough transform. *Computer Vision, Graphics, and Image Processing*, 44:87–116, 1988.

[53] Karpjoo Jeong and Dennis Shasha. PLinda 2.0: A transactional/checkpointing approach to fault tolerant Linda. In *Proceedings of the 13th Symposium on Reliable Distributed Systems*, 1994.

[54] Alan Kalvin, Edith Schonberg, Jacob T. Schwartz, and Micha Sharir. Two-dimensional, model-based, boundary matching using footprints. *The International Journal of Robotics Research*, 5(4):38–55, 1986.

[55] Kenichi Kanatani. 3D Euclidean versus 2D non-Euclidean: Two approaches to 3D recovery from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):329–332, March 1989.

[56] Ashfaq Khokhar and Viktor K. Prasanna. Scalable data parallel geometric hashing: Experiments on MasPar MP-1 and on Connection Machine CM-5. In *Proceedings of the DARPA Image Understanding Workshop*, pages 851–859, 1993.

[57] Konstantinos Konstantinides and John R. Rasure. The Khoros software development environment for image and signal processing. *IEEE Transactions on Image Processing*, 3(3):243–252, May 1994.

[58] Eric Krotkov and Ruzena Bajcsy. Active vision for reliable ranging: Cooperating focus, stereo, and vergence. *International Journal of Computer Vision*, 11(2):187–203, Oct. 1993.

[59] Frank P. Kuhl, O. Robert Mitchell, Marcus E. Glenn, and Didier J. Charpentier. Global shape recognition of 3-D objects using a differential library storage. *Computer Vision, Graphics, and Image Processing*, 27:97–114, 1984.

[60] Yehezkel Lamdan. *Geometric Hashing.* PhD thesis, New York University, June 1989.

[61] Yehezkel Lamdan, Jacob T. Schwartz, and Haim J. Wolfson. Object recognition by affine invariant matching. In *Proc. Computer Vision and Pattern Recognition*, pages 335–344, 1988.

[62] Yehezkel Lamdan, Jacob T. Schwartz, and Haim J. Wolfson. On recognition of 3-D objects from 2-D images. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1407–1413, 1988.

[63] Yehezkel Lamdan, Jacob T. Schwartz, and Haim J. Wolfson. Affine invariant model-based object recognition. *IEEE Transactions on Robotics and Automation*, 5(6):578–589, Oct. 1990.

[64] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Proc. Second International Conference on Computer Vision*, pages 238–249, 1988.

[65] Yehezkel Lamdan and Haim J. Wolfson. On the error analysis of 'geometric hashing'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 22–27, 1991.

[66] R. K. Lenz and Roger Y. Tsai. Techniques for calibration of the scale factor and image center for high-accuracy 3-D machine metrology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):713–720, Sep. 1988.

[67] Mun K. Leung and Thomas S. Huang. Detecting the wheel pattern of a vehicle using stereo images. *Pattern Recognition*, 24(12):1139–1151, 1991.

[68] Seppo Linnainmaa, David Harwood, and Larry S. Davis. Pose determination of a three-dimensional object using triangle pairs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):634–647, Sep. 1988.

[69] Jyh-Jong Liu and Robert Hummel. Geometric hashing with attributed features. In *Proceesdings of the Second CAD-Based Vision Workshop*, pages 9–16, Champion, Pennsylvania, Feb. 1994.

[70] Jyh-Jong Liu and Robert Hummel. Geometric hashing with attributed features. In *Proceesdings of ATR Systems and Technology Conference*, Monterey, California, Nov. 1994.

[71] Yuncai Liu and Thomas S. Huang. Estimation of rigid body motion using straight line correspondences. *Computer Vision, Graphics, and Image Processing*, 43(1):37–52, May 1988.

[72] Yuncai Liu, Thomas S. Huang, and Olivier D. Faugeras. Determination of camera location from 2-D to 3-D line and point correspondences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):28–37, Jan. 1990.

[73] Yuncai Liu, Thomas S. Huang, and Olivier D. Faugeras. Three-dimensional motion determination from real scene images using straight line correspondences. *Pattern Recognition*, 25(6):617–639, 1992.

[74] David G. Lowe. *Perceptual Organization and Visual Recognition.* Kluwer, Boston, MA, 1985.

[75] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[76] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[77] François G. Meyer and Patric Bouthemy. Region-based tracking using affine motion models in long image sequences. *CVGIP: Image Understanding*, 60(2):119–140, Sep. 1994.

[78] Rakesh Mohan, Gérard Medioni, and Ramakant Nevatia. Stereo error detection, correction, and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(2):113–120, Feb. 1989.

[79] R. Nevatia and K. R. Babu. Linear feature extraction and description. *J. of Computer Graphics and Image Processing*, 13:257–269, 1980.

[80] Lars Nielsen and Gunnar Sparr. Projective area-invariants as an extension of the cross-ratio. *CVGIP: Image Understanding*, 54(1):145–159, July 1991.

[81] Leslie M. Novak, Gregory J. Owirka, and Christine M. Netishen. Radar target identification using spatial matched filters. *Pattern Recognition*, 27(4):607–617, 1994.

[82] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Process.* McGraw-Hill, New York, second edition, 1984.

[83] Theo Pavlidis. *Algorithms for Graphics and Image Processing.* Computer Science Press, Rockville, MD, 1981.

[84] Johan Philip. Estimation of three-dimensional motion of rigid objects from noisy observations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(1):61–66, Jan. 1991.

[85] P. Puget and T. Skordas. Calibrating a mobile camera. *Image and Vision Computing*, 8(4):341–348, Nov. 1990.

[86] Isidore Rigoutsos. *Massively Parallel Bayesian Object Recognition.* PhD thesis, New York University, July 1992.

[87] Isidore Rigoutsos and Robert Hummel. Implementation of geometric hashing on the Connection machine. In *IEEE Workshop on Directions in Automated CAD-Based Vision*, Maui, Hawaii, June 1991.

[88] Isidore Rigoutsos and Robert Hummel. Robust similarity invariant matching in the presence of noise. In *Proceedings of the 8th Israeli Conference on Artificial Intelligence and Computer Vision*, Tel Aviv, Israel, Dec. 1991.

[89] Isidore Rigoutsos and Robert Hummel. Several results on affine invariant geometric hashing. In *Proceedings of the 8th Israeli Conference on Artificial Intelligence and Computer Vision*, Tel Aviv, Israel, Dec. 1991.

[90] Isidore Rigoutsos and Robert Hummel. Massively parallel model matching: Geometric hashing on the connection machine. *IEEE Computer: Special Issue on Parallel Processing for Computer Vision and Image Understanding*, Feb. 1992.

[91] Isidore Rigoutsos and Robert Hummel. Distributed Bayesian object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, New York, Jun. 1993.

[92] John W. Roach and J. K. Aggarwal. Determining the movement of objects from a sequence of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(6):554–562, Nov. 1980.

[93] Rémi Ronfard. Region-based strategies for active contour models. *International Journal of Computer Vision*, 13(2):229–251, Oct. 1994.

[94] Azriel Rosenfeld and Avinash C. Kak. *Digital Picture Processing*, volume 2. Academic Press, New York, second edition, 1982.

[95] Gerhard Roth and Martin D. Levine. Extracting geometric primitives. *CVGIP: Image Understanding*, 58(1):1–22, Jul. 1993.

[96] Hannan Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley Publishing Company, New York, 1989.

[97] K. B. Sarachik and W. E. L. Grimson. Gaussian error models for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, New York City, New York, Jun. 1993.

[98] Karen B. Sarachik. Limitations of geometric hashing in the presence of Gaussian noise. Technical Report 1395, M.I.T. AI Lab, Oct. 1992.

[99] Jacob T. Schwartz and Micha Sharir. Identification of partially obscured objects in two and three dimensions by matching noisy characteristic curves. *The International Journal of Robotics Research*, 6(2):29–44, 1987.

[100] Yoshiaki Shirai. *Three-Dimensional Computer Vision.* Springer-Verlag, New York, 1987.

[101] Jefferey A. Shufelt and Jr. David M. McKeown. Fusion of monocular cues to detect man-made structures in aerial imagery. *CVGIP: Image Understanding*, 57(3):307–330, May 1993.

[102] Gunnar Sparr. Projective invariants for affine shapes of point configurations. Extended manuscript for the "Workshop on Invariants in Vision", Reykjavik, 1991.

[103] Minas E. Spetsakis and John Aloimonos. Structure from motion using line correspondences. *International Journal of Computer Vision*, 4:171–183, 1990.

[104] J. Ross Stenstrom and C. Ian Connolly. Constructing object models from multiple images. *International Journal of Computer Vision*, 9(3):185–212, 1992.

[105] Paul Suetens, Pascal Fua, and Andrew J. Hanson. Computational strategies for object recognition. *ACM Computing Surveys*, 24(1):5–61, March 1992.

[106] Frank Chee-Da Tsai. A statistical approach to affine invariant matching with line features. Technical Report 621, Computer Science Department, Courant Institution of Mathematical Sciences, New York University, Nov. 1992.

[107] Frank Chee-Da Tsai. *A Probabilistic Approach to Geometric Hashing Using Line Features*. PhD thesis, New York University, July 1993.

[108] Frank Chee-Da Tsai. Robust affine invariant matching with application to line features. In *Computer Vision and Pattern Recognition*, pages 393–399. IEEE Computer Society, June 1993.

[109] Roger Y. Tsai and Thomas S. Huang. Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):13–27, Jan. 1984.

[110] V. Venkateswar and Rama Chellappa. Extractions of straight lines in aerial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(11):1111–1114, Nov. 1992.

[111] Richard S. Wallace, Ping-Wen Ong, Benjamin B. Bederson, and Eric L. Schwartz. Space variant image processing. *International Journal of Computer Vision*, 13(1):71–90, Sep. 1994.

[112] Timothy P. Wallace and Paul A. Wintz. An efficient three-dimensional aircraft recognition algorithm using normalized fourier descriptors. *Computer Vision, Graphics, and Image Processing*, 13:99–126, 1980.

[113] Cheng-Ye Wang, Hanfang Sun, Shiro Yada, and Azriel Rosenfeld. Some experiments in relaxation image matching using corner features. *Pattern Recognition*, 16(2):167–182, 1983.

[114] Ling-Ling Wang and Wen-Hsiang Tsai. Computing camera parameters using vanishing-line information from a rectangular parallelepiped. *Machine Vision and Applications*, 3:129–141, 1990.

[115] Ling-Ling Wang and Wen-Hsiang Tsai. Camera calibration by vanishing lines for 3-D computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):370–376, Apr. 1991.

[116] Isaac Weiss. Projective invariants of shapes. In *Proceedings of the DARPA Image Understanding Workshop*, pages 1125–1134, 1988.

[117] Juyang Weng, Thomas S. Huang, and Narendra Ahuja. Motion and structure from two perspective views: Algorithms, error analysis, and error estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(5):451–476, May 1989.

[118] Richard P. Wildes. Direct recovery of three-dimensional scene geometry from binocular stereo disparity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):761–774, Aug. 1991.

[119] Haim Wolfson. On curve matching. In *Proceedings of IEEE Workshop on Computer Vision*, pages 307–310, 1987.

[120] Yehezkel Yeshurun and Eric L. Schwartz. Cepstral filtering on a columnar image architecture: A fast algorithm for binocular stereo segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):759–767, July 1989.

Appendix A

Direct Proof of the Independence Property

This section we show the independence property by computing the joint density function for the positional attribute and orientation attribute directly. An indirect proof was given in Section 4.4.

Assume that the endpoints of a line segment are (X_1, Y_1) and (X_2, Y_2). The position of the midpoint of the line segment is (X, Y), where $X = \frac{X_1+X_2}{2}, Y = \frac{Y_1+Y_2}{2}$. The slope of the line segment is K, where $K = \frac{Y_2-Y_1}{X_2-X_1}$. Random variables X and Y represent the positional attribute of the midpoint, while random variable K represents the orientation attribute of the midpoint. We also assume that the random variables X_i and Y_i are Gaussian distributed with the statistical parameters (μ_i, σ) and (ν_i, σ) respectively. The density functions for the positional attribute for the endpoints are

$$f(x_1, y_1) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{(x_1-\mu_1)^2 + (y_1-\nu_1)^2}{2\sigma^2}\right]$$

and

$$f(x_2, y_2) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{(x_2-\mu_2)^2 + (y_2-\nu_2)^2}{2\sigma^2}\right].$$

We can compute the joint distribution function for random variables X, Y and K.

$$\begin{aligned} F(x,y,k) &= P\{X \le x, Y \le y, K \le k\} \\ &= P\{\frac{X_1+X_2}{2} \le x, \frac{Y_1+Y_2}{2} \le y, \frac{Y_2-Y_1}{X_2-X_1} \le k\} \\ &= \int_R \frac{1}{(2\pi\sigma^2)^2} \exp\left[-\frac{(x_1-\mu_1)^2 + (y_1-\nu_1)^2}{2\sigma^2}\right] \times \\ &\quad \exp\left[-\frac{(x_2-\mu 2)^2 + (y_2-\nu_2)^2}{2\sigma^2}\right] dx_1 dx_2 dy_1 dy_2, \end{aligned}$$

where R is the region that $X_1 + X_2 \leq 2x, Y_1 + Y_2 \leq 2y, \frac{Y_2 - Y_1}{X_2 - X_1} \leq k$.

The goal is to compute the joint density function $f(x, y, k)$, where

$$f(x, y, k) = \frac{\partial^3}{\partial x \partial y \partial k} F(x, y, k).$$

We have

$$\begin{aligned}
& \frac{\partial^2}{\partial x \partial y} F(x, y, k) \Delta x \Delta y \\
\approx\ & F(x + \Delta x, y + \Delta y, k) - F(x, y + \Delta y, k) - F(x + \Delta x, y, k) + F(x, y, k) \\
=\ & \int_{R'} \frac{1}{(2\pi\sigma^2)^2} \exp\left[-\frac{(x_1 - \mu_1)^2 + (y_1 - \nu_1)^2}{2\sigma^2} \right] \times \\
& \exp\left[-\frac{(x_2 - \mu_2)^2 + (y_2 - \nu_2)^2}{2\sigma^2} \right] dx_1 dx_2 dy_1 dy_2 \\
=\ & \int_{R'} \frac{1}{(2\pi\sigma^2)^2} \exp\left[-\frac{(x_1 - \mu_1)^2 + (y_1 - \nu_1)^2}{2\sigma^2} \right] \times \\
& \exp\left[-\frac{(2x - x_1 - \mu_2)^2 + (2y - y_1 - \nu_2)^2}{2\sigma^2} \right] dx_1 dx_2 dy_1 dy_2 \\
=\ & \int\int_{\frac{y - y_1}{x - x_1} \leq k} \left\{ \int_{2x - x_1}^{2(x + \Delta x) - x_1} dx_2 \int_{2y - y_1}^{2(y + \Delta y) - y_1} dy_2 \times \right. \\
& \frac{1}{(2\pi\sigma^2)^2} \exp\left[-\frac{(x_1 - \mu_1)^2 + (y_1 - \nu_1)^2}{2\sigma^2} \right] \times \\
& \left. \exp\left[-\frac{(2x - x_1 - \mu_2)^2 + (2y - y_1 - \nu_2)^2}{2\sigma^2} \right] \right\} dx_1 dy_1,
\end{aligned}$$

where R' specify the region that $2x \leq x_1 + x_2 \leq 2(x + \Delta x), 2y \leq y_1 + y_2 \leq 2(y + \Delta y)$, and $\frac{y_2 - y_1}{x_2 - x_1} \leq k$. Dividing $\Delta x \Delta y$ on both side and let $\Delta x \Delta y \to 0$, we get

$$\begin{aligned}
& \frac{\partial^2 F(x, y, k)}{\partial x \partial y} \\
=\ & 4 \int\int_{\frac{y - y_1}{x - x_1} \leq k} \left\{ \frac{1}{(2\pi\sigma^2)^2} \exp\left[-\frac{(x_1 - \mu_1)^2 + (y_1 - \nu_1)^2}{2\sigma^2} \right] \times \right. \\
& \left. \exp\left[-\frac{(2x - x_1 - \mu_2)^2 + (2y - y_1 - \nu_2)^2}{2\sigma^2} \right] \right\} dx_1 dy_1.
\end{aligned}$$

We change variables by letting $x_1 = x + r\cos\theta, y_1 = y + r\sin\theta$. Then we have

$$\frac{\partial^2 F(x, y, k)}{\partial x \partial y}$$

$$
\begin{aligned}
&= 4\int_0^{\infty}\left\{\left(\int_{-\frac{\pi}{2}}^{\tan^{-1}k}+\int_{\frac{\pi}{2}}^{\tan^{-1}k+\pi}\right)\frac{1}{(2\pi\sigma^2)^2}\times\right.\\
&\quad \exp\left[-\frac{(x+r\cos\theta-\mu_1)^2+(y+r\sin\theta-\nu_1)^2}{2\sigma^2}\right]\times\\
&\quad \left.\exp\left[-\frac{(x-r\cos\theta-\mu_2)^2+(y-r\sin\theta-\nu_2)^2}{2\sigma^2}\right]d\theta\right\}rdr\\
&= 4\int_0^{\infty}\left\{\left(\int_{-\frac{\pi}{2}}^{\tan^{-1}k}+\int_{\frac{\pi}{2}}^{\tan^{-1}k+\pi}\right)\frac{1}{(2\pi\sigma^2)^2}\times\right.\\
&\quad \exp\left[-\frac{2[x^2+y^2-(\mu_1+\mu_2)x-(\nu_1+\nu_2)y+r^2]+\mu_1^2+\mu_2^2+\nu_1^2+\nu_2^2}{2\sigma^2}\right]\times\\
&\quad \left.\exp\left[-\frac{(\mu_2-\mu_1)r\cos\theta+(\nu_2-\nu_1)r\sin\theta}{\sigma^2}\right]d\theta\right\}rdr\\
&= \frac{1}{(\pi\sigma^2)^2}\exp\left[-\frac{2[x^2+y^2-(\mu_1+\mu_2)x-(\nu_1+\nu_2)y]+\mu_1^2+\mu_2^2+\nu_1^2+\nu_2^2}{2\sigma^2}\right]\times\\
&\quad \left\{\int_0^{\infty}\left[\left(\int_{-\frac{\pi}{2}}^{\tan^{-1}k}+\int_{\frac{\pi}{2}}^{\tan^{-1}k+\pi}\right)\times\right.\right.\\
&\quad \left.\left.\exp\left[-\frac{r^2+(\mu_2-\mu_1)r\cos\theta+(\nu_2-\nu_1)r\sin\theta}{\sigma^2}\right]d\theta\right]rdr\right\}.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&f(x,y,k)\\
&= \frac{\partial^3F(x,y,k)}{\partial x\partial y\partial k}\\
&= \frac{1}{(\pi\sigma^2)^2}\exp\left[-\frac{2[x^2+y^2-(\mu_1+\mu_2)x-(\nu_1+\nu_2)y]+\mu_1^2+\mu_2^2+\nu_1^2+\nu_2^2}{2\sigma^2}\right]\times\\
&\quad \int_0^{\infty}\frac{1}{1+k^2}\exp\left[-\frac{r^2}{\sigma^2}\right]\times\left[\exp\left[-\frac{((\mu_2-\mu_1)+(\nu_2-\nu_1)k)r/\sqrt{1+k^2}}{\sigma^2}\right]+\right.\\
&\quad \left.\exp\left[\frac{((\mu_2-\mu_1)+(\nu_2-\nu_1)k)r/\sqrt{1+k^2}}{\sigma^2}\right]\right]rdr\\
&= f_1(x,y)\times f_2(k).
\end{aligned}
$$

The joint density function for random variables (X,Y,K) can be separated into two parts, which proves that the positional attribute and orientation attribute for the midpoint of a line segment are independent. We may also see that the formula is too complicated to be useful in real applications.

Appendix B

Descriptions of the Khoros Modules

The software modules can be categorized into five classes: Cox-Boie edge detector [10], feature extraction, model building, recognition, and MVF (Multiple Vectors File Format) utilities. They will be discussed in the following sections.

B.1 Cox-Boie Edge Detector

There are five modules in this class. The module *vdet, vedge, vthin,* and *vfilter* are based on the programs written by Ingemar J. Cox, Deborah A. Wallach, and W. J. Kropfl (©copyrighted 1988 by Robotics Principles Research Department, AT & T Bell Laboratories). The algorithms are described in [10]. We adapted and ported the code to Khoros modules with permission.

vconv We reimplement the 2-D convolution program in a more efficient way. To save the computation time, the zeros in the kernel will be skipped. The reversed kernel is built first before the convolution computation begins. We provide three modes to handle the boundary conditions: reflect boundary, warp boundary, and pad zeros. For reflect boundary mode, we treat a pixel outside the image the same as its mirror image. Thus the gray value of a pixel located at $(-x, -y)$ is the same as the gray value of the pixel located at (x, y); For warp boundary mode, the gray value at location $(-x, -y)$ is the same as the gray value located at $(W - x, H - y)$, where W and H are width and height of the input image respectively; For pad zero mode, those pixels outside the input image are ignored.

vdet The module computes the central difference of the input image in four directions. The four directions are defined as the direction along x-axis, $along 45°$ direction, along y-axis, and along 135° direction. This module should be used with the modules *vedge*, and *vthin* together.

vedge The module takes the input produced by *vdet* and produces the edge map by finding zero crossings in four directions. The module provides an option to turn on the hysteresis analysis in order to produce a better result.

vthin The module takes the input produced by the module *vedge* and generates the edges with one pixel wide.

vfilter Produces an 1-D Gaussian mask with the specified size.

The Cox-Boie edge detector is composed by the above five modules. We have discussed it in Section 6.2.

B.2 Feature Extraction

There are seven modules in this class. They are used for extracting primitive features.

vbf The module traces the edges and produces the coordinates of the edges. Edge points are linked and grouped via their 8-connected neighbors. The coordinates and the chain codes are stored as VIFF (Khoros Visualization/Image File Format). The default background gray value is 0 which is modifiable by resetting the input parameter. This module also produces an optional line label image. The corresponding edge pixels are converted to their traced line labels.

vlapx The module takes the result of *vbf* and computes the line approximation. It checks the distance deviation of all points in the line segment and break the line into two line segments iteratively if the maximum distance deviation is greater than the specified threshold. The distance deviation is defined by the distance of an edge point to the base line which is defined by connecting the endpoints of the segment. The result is stored in VIFF format.

vline The module merges the nearby split line segments into longer line segments according to the detected slope. Four options are provided in this module to produce four kinds of feature primitives: endpoints of line segments without orientation information, midpoints of line segments with orientation information, endpoints of line segment with orientation information, and bisectors of corners with orientation information. For the bisector features, the corners are extracted by finding the intersections of line features. The resulting features are stored in ASCII file format as well as MVF format. The MVF format will be discussed in Section B.5.

vcircle The module extracts circles in several runs. We first estimate the circles by fitting the segments into circles in a least-squares sense. Then the merging process

combines arcs that are centered at the same position with the same radius. We can optionally turn on the circle verification process to filter out false alarms and produces more stable circles. The resulting features are stored in ASCII file format as well as MVF format.

vmark The module marks the extracted features on top of the input image. Hybrid features can be marked simultaneously. The attributes as well as the feature identification number can be marked optionally.

vmixf The module can mix two kinds of features together and produces hybrid features. The result is stored in MVF format.

vstrf The module stripes off the attribute part of the features. The positional information is kept untouched. Since the semantic meaning of the positional information is different from one feature type to another, we attach different tag on the stripped features depending on the original feature types.

B.3 Model Building

There are six modules in this class. The module *vsim, vkdt, vhist* are based on the assumption that the features are of the same type. The enhanced version *vhsim, vhkdt* are capable of handling the case of heterogeneous feature type. The internal file format used in the enhanced version is MVF format, while some of the internal files in the original version is in VIFF format.

vsim The module build the hash table for the case of similarity transformation. The input parameter specifies the model database which contains the files that participate in this model building process. The feature type is assumed to be homogeneous. For a particular model, all combinations of basis pairs are used to normalize the features if they meet the specified length criteria.

vhsim This is the enhanced version of *vsim*. The features used can be heterogeneous. The basis set is produced by the module *vgenbas* so that it could be different than the features that we want to normalize. Thus it provide one more level of flexibility. This module is used for prepare the hash table as well as the feature normalization for test image.

vkdt, vhkdt The module *vkdt* and *vhkdt* are used for building the k-d tree for the hash table computed by *vsim* and *vhsim* respectively. The input parameter specify the number of pieces of the hash table to be split. Then the module can work on the

specified piece. The distributed computation is performed by building several k-d tree on top of the hash table as described in Section 5.3.

vhist This module computes the histogram of the hash table. The input hash table is assumed to be the one produced by *vsim*.

vgenbas The module generates the basis which can be used in the module *vhsim*. There are two ways to use this module: we can use this module to generate the basis set for model building; we can also use this module to generate the basis set to normalize the other features during the recognition process.

B.4 Recognition

There are eleven modules in this class. The module *voting, vpresim,* and *vlogf* are based on the assumption that all the features are of the same type. Their corresponding enhanced version *vhmatch, vhpresim* and *vhlogf* are capable of handling the case of heterogeneous feature type. The internal file format used in the enhanced version is MVF format, while some of the internal files in the original version is in VIFF format.

voting The module performs the weighted voting computation for similarity case. The system parameters are specified in the module *vpresim*. This module uses one piece of k-d tree information to derive part of the result. The partial results are then combined by the module *vlogf* to produce the final result.

vhmatch This is the enhanced version of the module *voting* which is capable of handling the case of hybrid features.

vpresim Prepare the system parameters, for example, the matching hypothesis, the standard deviation, and the number of candidates that we are interested. Note that the basis set for test image is selected by this module.

vhpresim Prepare the system parameters for the enhanced version of *vmatch*. In particular, we separate the basis set selection into the other module (*vgenbas*) to increase one level of flexibility.

vlogf This module combines the partial results of vote computation (produced by *voting*). At this moment, we assume that we only distribute the computations into at most ten pieces.

vhlogf This is the enhanced version of *vlogf* which is capable of handling the case of hybrid features.

vover This module displays the recognition result. The edgemap of the model is overlaid on top of the input test image. We can optionally specify which trial and which candidate we are interested in.

vmkbas This module can display the selected basis set on the screen on the fly.

vfit The module resize the edgemap to specified size. Both forward transformation and back transformation modes are provided. The module is internally called by the module *vbars*.

vbars The module display the model/basis vote-getters for the recognition result. The model name, the basis in the model, and the length-encoded vote are displayed. We can optionally specify which trial we want to display.

verify The module performs the verification of the recognition result as described in Section 6.3. The number of edge pixels that match to the edge pixels in the model, the distance deviation, and the other statistics are reported. The edgemap of the matched portion is also produced. We can optionally specify which trial and which candidate we are interested in.

vmoddb The module display the edgemaps of the model database. The number of models displayed in a row and the number of the models displayed in a column are configurable. The hybrid features can be overlaid on top of the each model optionally.

B.5 MVF Utilities

The Khoros standard file format VIFF is not general enough for our inter module communications. We define a new file format MVF (Multiple Vectors File Format) which can handle multiple vectors with different vector lengths and data types. The semantic meaning of each use of MVF file is left undefined. We attach a tag for each MVF file so that each module can use the tag to check if the input file is produced by the correct module. The class contains five modules for MVF utilities.

vmvfinfo The module can dump a file with MVF format to ASCII format. This is a general purpose MVF utilities regardless the associated MVF tag.

vmv2asc The module can dump a feature file with MVF format to ASCII format.

vasc2mvf The module converts an ASCII file with features to MVF file.

vmvf2viff The module converts the features which are stored in MVF format to VIFF format. This is no longer used in the enhanced version of modules discussed in the previous sections. The features are assumed to be of the same type.

viff2mvf The module converts the features which are stored in VIFF format to MVF format. This is no longer used in the enhanced version of the modules in the previous sections.