University of Alberta

Efficient and Robust 3-D Object Recognition by Integration of Visual and Tactile Data

by

Michael Boshra ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 1997

0-612-22955-6

University of Alberta

Library Release Form

Name of Author: Michael Boshra

Title of Thesis: Efficient and Robust 3-D Object Recognition by Integration of Visual and Tactile Data

Degree: Doctor of Philosophy

Year this Degree Granted: 1997

[illegible]

Michael Boshra
11105-85 Ave.
Edmonton, Alberta
Canada, T6G 0W7

Date: July 31st, 1997

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Efficient and Robust 3-D Object Recognition by Integration of Visual and Tactile Data** submitted by Michael Boshra in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

[illegible]

Prof. Hong Zhang (Supervisor)

David Lowe

Prof. David Lowe (External)

[illegible]

Prof. Anup Basu (Examiner)

[illegible]

Prof. Alinda Friedman (Examiner)

Xiaobo Li

Prof. Xiaobo Li (Examiner)

Date: July 30th, 1997

Two are better than one: because they have a good reward for their labour.
For if they fall, the one will lift up his fellow:
but woe to him that is alone when he falleth;
for he hath not another to help him up.
Ecclesiastes 4:9-10

To my Edmontonian friends for their support

Abstract

We present a novel system for recognizing a polyhedral object in a robot hand using vision and touch. The system consists of three stages: screening, hypothesis generation and hypothesis verification. The purpose of the screening stage is to quickly eliminate many invalid model objects from further consideration, early in the recognition process. The next stage generates hypotheses about the identity and pose of the scene object, considering only model objects that have passed the screening test. Each generated hypothesis is then passed to the hypothesis-verification stage in order to determine its validity.

Screening is generally performed by applying an approximate test to determine whether model-object features can be consistent with relevant scene ones. In order to integrate visual and tactile features in the context of screening, we formulate the recognition problem as a Constraint-Satisfaction problem (CSP). A novel method is developed for efficiently constructing a CSP corresponding to a combination of visual and tactile features. Model/scene consistency is determined in low-order polynomial time by enforcing local consistency, in particular arc consistency.

In general, hypotheses are generated by matching a small subset of scene features with corresponding model ones. In some situations, visual and tactile features, individually, do not provide sufficient information about the object pose. To overcome this problem, we present a technique for generating hypotheses using a hybrid set of visual and tactile features.

For touch-based verification, we present a highly-discriminative data-driven indexing scheme. Such a scheme can determine the consistency of a hypothesis with a tactile feature in a *constant* time, which is independent of the number of model features. The proposed index is superior to similar ones in that it incorporates almost all the constraints provided by a tactile feature, thus leading to an index with higher

discriminative power.

Finally, we present a pixel-based technique for vision-based verification. In such a technique, a hypothesis is verified by synthesizing the corresponding edge image, and comparing it, pixel wise, with the scene edge image. The main advantage of the proposed technique, over similar pixel-based ones, is its capability of accommodating the uncertainties of both scene data and estimated object pose. This is achieved through dilation of the scene edge image. An analytical method is presented for determining the extent of dilation assuming some bounded-error on the object pose.

Acknowledgements

I would like to thank my supervisor. Prof. Hong Zhang, for his continuous guidance. support. and patience. I am grateful to Profs. Anup Basu. Peter van Beek. and Xiaobo Li. for providing several suggestions that helped in improving this work. Many thanks to members of my examining committee, especially Profs. Alinda Friedman and David Lowe. for the time and effort they have taken to evaluate the work.

Credit for guiding me to the world of graduate studies goes to my dear friends Maged Michael and Hani Gorgi. I do not think that I would have made it without their help and support.

It has been a long journey through all stages of education. I feel compelled to acknowledge some of those who taught me throughout the years. During my B.Sc./M.Sc. studies at the University of Alexandria (Egypt). Profs. Belal. Hamad. Ismail. and Selim greatly influenced my way of thinking. Furthermore. I can not forget the teaching and encouragement of Messrs Attia. Boulos, Halim. Joe. and Sedik during my school years.

Above all. I would like to thank my heavenly Shepherd, The Lord Jesus Christ. for guiding me throughout all the valleys of this life.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There has been a growing interest in the development of flexible-manufacturing environments, in order to reduce production cost, improve product quality, and shorten new product cycle time. Sensor-based robots play a key role in this area. For example, in order for a robot to manipulate objects, it must determine identities and locations of these objects in the workspace. In most of today's manufacturing environments, special devices (e.g., part feeders) are used to position and orient objects before being manipulated by a "blind" robot. Being object-specific, these devices are often replaced upon switching to a new task, involving a different set of objects. Equipping robots with the capability of identifying and localizing 3-D objects, from sensory data, will eliminate or reduce the need for those positioning/orienting devices. Thus, the capability of 3-D object recognition (3DOR) holds the potential for reducing the cost of production, especially for small-volume manufacturing. Furthermore, by reducing the time required to setup a robot workcell for a new task, we can also expect shorter cycle times for new products.

Most existing 3DOR systems rely on a single type of sensory data, which can be of 2-D nature such as visual data, or 3-D nature such as tactile or range data. In some situations, however, both 2-D and 3-D sensory data can be available simultaneously. Examples of such situations are:

- The object of interest is grasped or touched by a robot hand. In this case, tactile sensors mounted on the hand provide tactile data, while a visual sensor monitoring the robot workspace provides visual data.

- The workspace is monitored by an intensity camera, providing 2-D data, along

Aspect	Visual Data	Tactile Data
Scope	global	local
Dimension	2-D	3-D
Occlusion	applicable	n/a
Pertinence	not guaranteed	guaranteed

Table 1.1: A comparison between visual and tactile data.

with a range finder, providing 3-D data.

- The sensor used is a stereo camera. In stereo vision, 3-D features are reconstructed by establishing correspondence between 2-D features extracted from each of the stereo images. However, 3-D data can not be reconstructed if a 2-D feature in one image does not have a unique matching one in the other image. Accordingly, the output of the correspondence process can be viewed as a combination of 2-D and 3-D features, where the 2-D features are those unmatched ones (in both images).

- The object is constrained to lie on a table. In this case, the 2-D data provided by a visual sensor are complemented by an "implicit" piece of 3-D data provided through knowledge of the table's location with respect to the visual sensor.

Using systems that utilize only a single type of data (either 2-D or 3-D) in such situations can be inefficient, or at least "non-optimal" in the sense that not all the available sensory data are utilized in order to improve system performance.

To demonstrate the usefulness of sensor integration, let us consider the case of an object in a robot hand. Data obtained from visual and tactile sensors have different characteristics. Visual data are relatively global; i.e., they capture the visible part of the object as seen from the visual-sensor viewpoint. However, visual data provide only a 2-D projection of the 3-D world. Thus, there is a substantial loss of 3-D information. Tactile data, on the other hand, have dissimilar characteristics. They provide 3-D information about the object, but unfortunately these data are local and in many cases insufficient to completely recognize the object. In addition, visual data are sensitive to occlusion of the object by visual obstacles such as the robot hand. This problem is not applicable to tactile data. In fact, the tactile data can provide information about some visually occluded parts of the object. Finally, it is generally not guaranteed whether a given visual feature (e.g., an edge or a junction) belongs to the object of interest. This is not the case with tactile features, since they come from direct contact with the object. A summary of this comparison between visual and tactile data is shown in Table 1.1. Observing the complementary characteristics of visual and tactile data, we can see that integrating both of them would yield robust and efficient 3DOR, more than what would be achieved using either vision or touch alone.

The main thrust of this dissertation is to develop techniques for 3DOR by integrating 2-D and 3-D data. We will consider the task of recognizing, or just localizing, a polyhedral object in a robot hand. Applicability of the proposed techniques to other recognition tasks involving 2-D and 3-D sensory data is quite straightforward.

The remainder of this chapter is organized as follows. The next section defines the problem of 3DOR. The general architecture of 3DOR systems is presented in Section 1.2. Section 1.3 reviews the few vision/touch-based 3DOR systems reported in the literature. Finally, Section 1.4 provides an overview of the proposed system.

1.1 Problem Definition

Scene-analysis tasks can be classified according to whether each of the scene and the sensor is static or dynamic. Thus, we have four categories:

1. Static Scene/Static Sensor ($SSSS$)
2. Static Scene/Dynamic Sensor ($SSDS$)
3. Dynamic Scene/Static Sensor ($DSSS$)
4. Dynamic Scene/Dynamic Sensor ($DSDS$)

Our problem of 3DOR falls under the $SSSS$ category. Tasks of the $SSDS$ category include autonomous exploration (e.g., [60, 69, 72]). An example of the $DSSS$ tasks is motion tracking (e.g., [6, 71]). Finally, $DSDS$ tasks include autonomous observation (e.g., [68]), and active motion tracking (e.g., [62]).

The problem of 3DOR can be defined as follows:

- *Input:* A set of sensory data, $\mathcal{S}$, corresponding to *an arbitrary* scene, and a model object set, $\mathcal{M} = \{\mathcal{O}\}$.
- *Desired Output:* A set, $\mathcal{L} = \{(\mathcal{O}_i, \mathbf{T}_i)\}$, where $\{\mathcal{O}_i\} \subseteq \mathcal{M}$ is a set of objects in the scene that are captured by $\mathcal{S}$, and $\mathbf{T}_i$ is a description of the pose of $\mathcal{O}_i$ with respect to some global coordinate frame. In 3-D, $\mathbf{T}_i$ can be represented by six transformation parameters, three translational and three rotational.
- In case of ambiguity in the sensory data $\mathcal{S}$, a set $\{\mathcal{L}_i\}$, should be produced, where each $\mathcal{L}_i$ represents a scene interpretation that is consistent with $\mathcal{S}$.
- In some tasks such as bin picking, the output is desired in the form of a directed graph, where nodes represent recognized objects, $\mathcal{L}$, while edges depict occlusion relationships between them (e.g., [12]).

It is important to distinguish the 3DOR problem considered in this work from the following related ones:

- *2-D and 2.5-D Object Recognition:* these problems are concerned with recognizing objects from a *limited* range of viewpoints (e.g., survey [24]).
- *Active 3DOR:* this problem focuses on the development of sensor-control strategies in order to actively gather data that are needed for successful recognition (e.g., [1, 2, 30, 44, 54, 67]).
- *Qualitative 3DOR:* this problem is concerned with identifying 3-D objects *without* localizing them (e.g., [8, 29]).

1.2 Architecture of 3DOR Systems

Fig. 1.1 shows a general architecture of 3DOR systems. As shown in this figure, there are three main modules:

1) Model Descriptor (off-line module): The purpose of this module is to build a description of model objects. Choice of a representation to describe model objects is crucial to the performance of 3DOR systems. In most cases, the object representation is augmented with auxiliary data structures, in the form of indexes or hash tables, in

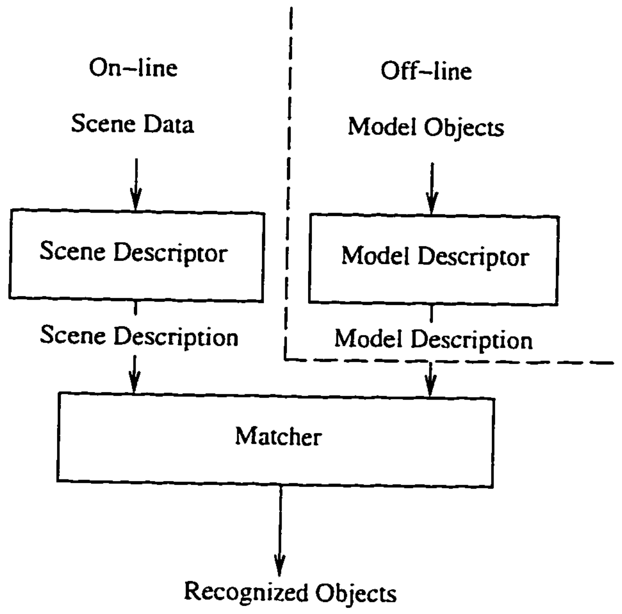


Figure 1.1: General architecture of 3DOR systems.

order to enable fast access to selected model features. Object representation schemes that have been used in 3DOR systems include polyhedral approximation [32, 39, 40], attributed surface graph [34, 51, 73], generalized cones [18], extended Gaussian image [43, 50], multi-view representation (or aspect graphs) [29, 42, 64, 76]. More details about object representation schemes can be found in [4, 10, 11].

2) Scene Descriptor: Since direct interpretation of raw sensory data is difficult, a high-level scene description is important to narrow down the gap between scene data and model-object description. An important criterion in selecting scene and model representation schemes is to make the matching process (described below) as simple and efficient as possible. For this reason, many 3DOR systems choose the same representation scheme for describing both model objects and scene data (e.g., [31, 32, 34, 38, 51, 73]). In some cases, however, it might be difficult or even impossible to build a scene description that is similar to the model one. Examples of these cases are:

1. The sensory data are two-dimensional. In such a case, the scene is usually described by a set of contours (straight or curved). A perceptual organization process is often applied to group the extracted contours into *perceptual structures*, where each structure is likely to belong to the same object. Percep-

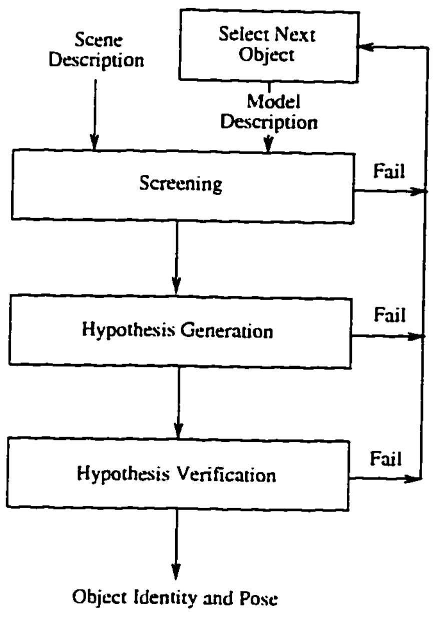


Figure 1.2: Main stages of matching (it is assumed here, for simplicity, that there is only one object to be recognized).

tual structures are constructed based on contour relationships that are invariant over a wide range of viewpoints such as connectivity, collinearity and parallelism (e.g., [49, 56, 74]).

2. The scene is too noisy or cluttered or both. In such a case, only sparse primitive features, such as edges or surfaces, can be extracted from the scene (e.g., [12]).

3) Matcher: The matching process can be viewed as the establishment of a consistent correspondence between scene and model features. That is, given two sets of scene and model features, $\{F_i^s\}$ and $\{F_i^m\}$, respectively, the function of the matcher is to assign to each scene feature, F_i^s, a model one, F_j^m, or *null*, such that the global interpretation of the scene is consistent with the scene data. Notice that the null label is needed to accommodate scene features which belong to alien scene objects. It can be seen that, in principle, the problem is of exponential complexity. However, as will be shown shortly, this complexity can be very much controlled by utilizing the object-rigidity constraint.

The matching process, in the context of vision/touch-based 3DOR, is the focus of this dissertation. Reviewing 3DOR systems in the literature, we can characterize three main stages in the matching process (see Fig. 1.2):

1. Screening: The purpose of this stage is to quickly eliminate many invalid model objects from further consideration. early in the matching process. Screening is usually performed by comparing a model object with a perceptual structure (a group of scene features that are perceived to belong to the same object). Thus. the effectiveness. and sometimes the feasibility, of screening depend on the reliability of extracting those structures.

2. Hypothesis Generation: As mentioned, the strongest constraint that can limit the combinatorial explosion in scene/model matching is the object-rigidity constraint. For this reason, the majority of 3DOR systems first try to establish partial correspondence between scene and model features, in order to generate hypotheses about the pose of the object under consideration. Only very few features are sufficient to generate a hypothesis (e.g., three non-parallel 3-D surfaces, or three 2-D edges).

3. Hypothesis Verification: Each of the generated hypotheses is verified, and possibly refined, by performing extensive comparison between scene and model features. A hypothesis is considered valid, if there is a sufficient number of consistent features.

1.3 Vision/Touch-Based 3DOR

Surprisingly, despite the considerable amount of research done in the areas of vision-based and touch-based 3DOR, we have been able to find only few systems that attempt to integrate both sensing modalities. Luo and Tsai [57] use a pre-compiled decision tree to recognize 3-D objects on a plane. The first level of this tree utilizes moment invariants of a scene-object's silhouette to reduce the uncertainty about the identity of the object. The later levels of the tree identify the scene object using tactile data, which are *actively* acquired from two tactile sensors on the jaws of a parallel-jaw gripper. Allen [1] uses passive stereo vision to guide a tactile sensor, mounted on a robot arm, to explore the scene object and construct a partial 3-D description of it. This description is then matched with model objects to recognize the scene object. Ambiguity in determining the object's identity is resolved by further tactile exploration.

The above two approaches address the problem of active 3DOR using tactile sensors mounted on a robot hand (an *SSDS* task), where the object of interest is *not* grasped by the hand. Obviously, this problem is different from the one considered in this work, that of recognizing an object *in* a robot hand using vision and touch (an *SSSS* task). Furthermore, notice that in those systems, vision sensing is used in the initial stage only, followed by touch sensing in later stages; i.e., visual and tactile data are combined *sequentially*. Accordingly, the techniques developed in those systems are unsuitable in our case, which requires *simultaneous* integration of vision and touch.

The work of Browse and Rodger [19] is among the few efforts that integrate vision and touch simultaneously in the context of 3DOR. They present a system to recognize objects of uniform cross section, which are constrained to lie on a table. Thus, there are only three degrees of freedom to be determined. For each match between a visual or tactile feature and a model one, a set of consistent discretized hypotheses is generated. These sets are then intersected to identify the object and obtain an estimate of its pose. There are several limitations in this system: 1) scope of objects is limited to those with uniform cross section, 2) objects are assumed to have only three degrees of freedom, 3) the estimated pose is only a rough estimate of the true one, depending on the quantization levels of the three transformation parameters, and 4) relationships between scene features are not utilized to improve run-time performance.

1.4 Overview of the Thesis

We propose a novel system that performs 3DOR by integrating visual and tactile data. The model objects are assumed to be polyhedral. We believe that studying this class of objects provides a good starting point for handling more complex object classes. A polyhedral model object, $\mathcal{O}$, can be represented by the triple

$$\mathcal{S}^m = \{S_i^m\}, \quad \mathcal{V}^m = \{V_i^m\}, \quad \mathcal{E}^m = \{E_i^m\}$$

where $\mathcal{S}^m$, $\mathcal{V}^m$ and $\mathcal{E}^m$ are the sets of model surfaces, vertices and edges, respectively. Each element in those sets, S_i^m,V_i^m or E_i^m, consists of its parametric description, in

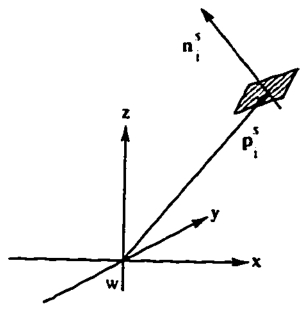


Figure 1.3: A tactile surface patch.

addition to pointers to the other elements that define it. For example, the parametric description of a model surface, S_i^m, is

$$\mathbf{n}_i^m \mathbf{x} = d_i^m \tag{1.1}$$

where $\mathbf{n}_i^m$ is the outward normal of S_i^m, and d_i^m is the distance from the origin to S_i^m in the direction of $\mathbf{n}_i^m$. In addition, S_i^m contains pointers to edges in $\mathcal{E}^m$ and vertices in $\mathcal{V}^m$ that define its boundary. The coordinates of each model vertex, V_i^m, are represented by vector $\mathbf{v}_i^m$. Finally, each edge, E_i^m, is represented by a tuple of vertices (V_{i1}^m, V_{i2}^m).

The scene data are described as follows. Features extracted from the scene are represented by the sets:

$$\mathcal{S}^s = \{S_i^s\}, \quad \mathcal{V}^s = \{V_i^s\}, \quad \mathcal{E}^s = \{E_i^s\}$$

where $\mathcal{S}^s$, $\mathcal{V}^s$ and $\mathcal{E}^s$ are the sets of tactile features, visual vertices and visual edges, respectively. The parametric description of each piece of tactile data depends on its type. For example, if S_i^s is a point-like tactile surface patch, then it can be represented by the following equation:

$$\mathbf{n}_i^s \mathbf{x} = \mathbf{n}_i^s \mathbf{p}_i^s \tag{1.2}$$

where $\mathbf{n}_i^s$ is the inward normal of the patch, and $\mathbf{p}_i^s$ is its center with respect to a world frame (see Fig. 1.3). The coordinates of each visual vertex, V_i^s, are represented by vector $\mathbf{v}_i^s$. Finally, each visual edge, E_i^s, is represented by a tuple of vertices (V_{i1}^s, V_{i2}^s).

The matching process consists of the three stages that have been outlined in Section 1.2: screening, hypothesis generation and hypothesis verification. The techniques developed for each of those stages. in the context of vision/touch-based 3DOR. constitute the contribution of this dissertation. They can be outlined as follows.

In the screening stage, presented in Chapter 2. we formulate the 3DOR problem as a Constraint-Satisfaction problem (CSP), in order to provide a unified framework for integrating different types of sensory data. A novel method is presented for efficiently constructing a CSP corresponding to a match between a scene perceptual structure and a model object. Local consistency, in particular arc consistency. is enforced on the CSP. in order to determine whether to further consider the model object.

For the hypothesis-generation stage. we discuss a method for generating hypotheses using a hybrid set of visual and tactile features. This method, described in Chapter 3. is essential in situations when the visual and tactile features. individually. do not provide sufficient information about the object pose. Furthermore, we will demonstrate its superiority over vision-based methods in the generation of a smaller number of hypotheses and the production of a more accurate object-depth estimate.

For touch-based hypothesis verification, Chapter 4, a highly-discriminative indexing scheme for data-driven verification is presented. Such a scheme can determine whether a tactile feature supports a given hypothesis in a *constant* time, independent of the number of model features.

For vision-based hypothesis verification. Chapter 5. a pixel-based approach is adopted. In such an approach. a hypothesis is verified by synthesizing the corresponding edge image. and comparing it. pixel wise. with the scene edge image. The main advantage of the proposed technique, over similar pixel-based ones, is its capability of accommodating the uncertainties of both scene data and estimated object pose. which is achieved through dilating the scene edge image. An analytical method is presented for determining the extent of dilation assuming some bounded-error on the object pose.

Each of the developed techniques can be regarded as an independent module which can be plugged in any 3DOR system that uses a similar type of data. For example. the touch-based and vision-based verification techniques are applicable to any 3DOR task involving 3-D or 2-D data, respectively. For this reason. we have decided to focus on evaluating the performance of each stage *independently*, rather than evaluating the system as a whole.

Chapter 2

Vision/Touch-Based Screening

2.1 Introduction

In this chapter. we present a technique for screening model objects by integrating visual and tactile data. Despite being critical to the system performance. we have been able to find only few 3DOR systems that address the screening issue.

Flynn and Jain [34] suggest a simple screening scheme which uses intrinsic attributes of scene features (e.g.. radius of a cylinder. angle between surface orientations) to quickly eliminate inconsistent model objects.

In the system developed by Fan *et al.* [31], surface graphs are used to represent both input range data and model objects (each model object is represented by several 2.5-D surface graphs). The scene graph is first segmented into a set a subgraphs, where each subgraph consists of surfaces that are likely to correspond to the same object. Screening is then performed by comparing attribute values of a scene subgraph with those of a model graph. The attributes used in the comparison process include number of nodes. number of planar nodes and visible area of largest node. The model graph is considered for detailed matching with the scene subgraph, only if the difference between each pair of scene and model attribute values is less than some threshold. Thresholds are needed to cope with errors in scene segmentation, as well as occlusion.

The screening scheme presented by Kim and Kak [51] is composed of three stages. called filters. which are arranged in ascending order of their computational complexity. The first filter compares coarse attribute values of a scene graph, which is assumed to correspond to a single object. and each model-object graph. The chosen attributes include total number of nodes. total number of arcs. maximum degree of a node.

and number of surfaces of each type (e.g., planar, cylindrical). A model object is accepted by the first filter, only if each scene attribute value is not more than the corresponding model one. The second filter verifies that each node in the scene graph has a consistent one in the model graph. The third filter applies a polynomial-time bipartite matching algorithm to check for the existence of an injective mapping from scene nodes to consistent model ones. Each model object that is accepted by the above filters is then passed to a *pruning* module. This module applies a discrete-relaxation approach, by utilizing several types of local constraints, in order to reduce the uncertainty in the correspondence between scene and model nodes. This reduction in correspondence uncertainty can considerably decrease the number of scene/model feature comparisons in subsequent matching stages. If a scene node does not have any consistent model one, then the considered model object can not correspond to the scene graph and so it is rejected. Thus, the pruning module can be thought of as a fourth screening filter.

The above approaches use only a single type of sensory data (3-D data), and so they are unsuitable in our case, which involves both 2-D data (visual) and 3-D data (tactile). We present a technique for vision/touch-based screening of model objects. This technique, which is analogous to the pruning module in [51] described above, can be outlined as follows. The 3DOR problem is formulated as a Constraint-Satisfaction problem (CSP) in order to provide a unified framework for integrating different types of sensory data. A novel method is presented for efficiently constructing a CSP corresponding to a combination of 2-D and 3-D scene features. Screening is performed by enforcing local consistency on the CSP. In addition to eliminating many erroneous objects, the step of local-consistency enforcement (LCE) can significantly reduce the uncertainty in the correspondence between scene and model features. Thus, when passing the locally-consistent CSP to the hypothesis generation module, we can expect a significant reduction in the number of scene/model matches, and accordingly the number of generated hypotheses. This will, in turn, reduce the computational load on the hypothesis verification module.

It is interesting to compare the adopted CSP approach, outlined above, with some object recognition systems that apply tree search (e.g., [39, 40]). In those systems, the recognition task is implicitly formulated as a CSP that is solved directly through searching the solution space. a problem of exponential complexity in general. In our work. the exponential complexity is avoided as follows:

1. Local consistency is enforced in low-order polynomial time. This process is typically considered by CSP researchers as a preprocessing stage to tree search. with the purpose of reducing the size of the solution space at a reasonable cost (e.g., [53]).

2. Searching the reduced solution space. of the locally-consistent CSP. is still of exponential complexity. To overcome this problem. we adopt the polynomial-time hypothesis generation/verification approach outlined in Section 1.2.

It should be noted, however, that LCE is not applicable, if the process of extracting perceptual structures is unreliable. This because, in such a case, "null" labels would have to be added in order to accommodate extraneous scene features. Those null labels would prevent LCE from reducing the size of the solution space (this is because null labels would result in the absence of zero rows/columns in constraint matrices: this point will become clear in the next section when presenting the LCE algorithm).

The remainder of the chapter is organized as follows. Section 2.2 defines the problem. and presents the LCE algorithm. Construction of the CSP is discussed in Section 2.3. Section 2.4 presents experimental results. Finally, conclusions are drawn in Section 2.5[1].

2.2 Formulation of the Problem

2.2.1 Definition of a CSP

A CSP is represented by a hypergraph. called a *constraint network* (CN). A CN consists of a set of n nodes. $\mathcal{X} = \{x_1, \ldots, x_n\}$, representing the problem's variables. and a set of m hyperedges. $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$. representing the problem's constraints. Each variable. x_i. can take values from a domain, $\mathcal{Y}_i$. i.e.,

$$x_i \in \mathcal{Y}_i \qquad 1 \leq i \leq n.$$

[1]A preliminary version of this work was published in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*. Pittsburgh. Pennsylvania. 1995 [16].

Each constraint, $\mathcal{C}_i$, defines a set of valid combinations of values for a subset of variables, $\mathcal{X}_i = \{x_{i(1)}, \ldots, x_{i(n(i))}\}$, where $\mathcal{X}_i \subseteq \mathcal{X}$, and $n(i)$ is the *order* of $\mathcal{C}_i$. Thus,

$$\mathcal{C}_i \subseteq \mathcal{Y}_{i(1)} \times \cdots \times \mathcal{Y}_{i(n(i))} \qquad 1 \leq i \leq m.$$

Constraint $\mathcal{C}_i$ can be viewed as an $n(i)$-dimensional binary array, where the value of each entry, e, is 1, if $e \in \mathcal{C}_i$, and 0 otherwise. Thus, a unary constraint (order 1) can be viewed as a binary list, while a binary constraint (order 2) can be viewed as a binary matrix.

An instantiation of the variables in $\mathcal{X}$, $\mathcal{I} = \{x_1 = y_1, \ldots, x_n = y_n\}$ where $y_i \in \mathcal{Y}_i$, is said to be a *solution* of the CSP, if $\mathcal{I}$ satisfies all the constraints in $\mathcal{C}$. Solving a general CSP involves searching the solution space at an exponential time complexity (see, for example, [25, 53]).

2.2.2 Local-Consistency Enforcement

A CSP is said to be locally-consistent of order i, if every solution to any subnetwork of size $i-1$ nodes, N_{i-1}, can be extended to a solution of any subnetwork of size i, N_i, that contains N_{i-1}. Local consistencies of order 1, 2 and 3 are commonly known as node, arc and path consistencies, respectively [58]. Since enforcing local consistency of order i is exponential in i [26], we have chosen to enforce local consistency of order 2 (arc consistency). As will be demonstrated by the experimental results in Section 2.4, we have found that the enforcement of arc consistency is very effective in both eliminating erroneous model objects and reducing the correspondence uncertainty between scene and model features.

The chosen LCE algorithm, which enforces arc consistency, can be written as follows (see [58]):

1. If any constraint, $\mathcal{C}_i$, is null, then stop (a null constraint implies that there is no solution to the CSP).

2. For each constraint $\mathcal{C}_i$, we check each node/label pair, (x_{ij}, y_{ij}), involved in $\mathcal{C}_i$. If there is no element of $\mathcal{C}_i$ that contains y_{ij}, then label y_{ij} can not be part of any solution to the CSP (because constraint $\mathcal{C}_i$ will not be satisfied). Thus, we delete y_{ij} from the domain of x_{ij}.

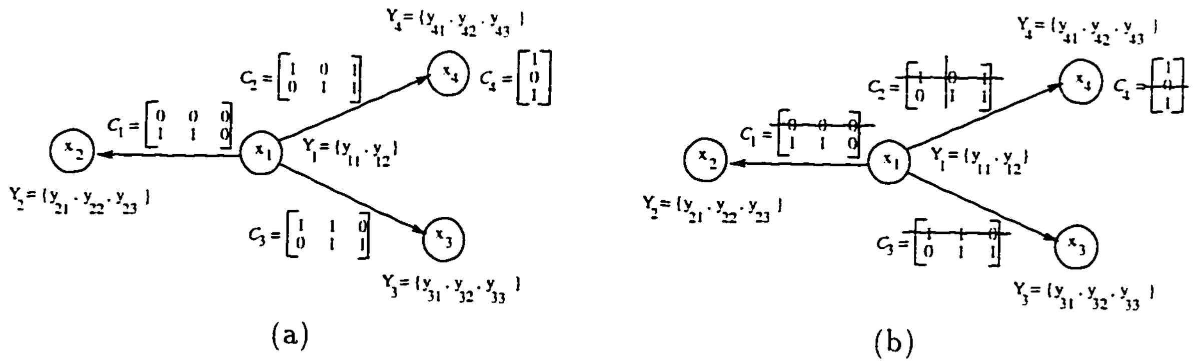


Figure 2.1: An illustration of LCE: (a) a CSP. (b) the CSP after one iteration of LCE.

3. For each deleted node/label pair (x_{ij}, y_{ij}), we *propagate* the deletion to other constraints, C_k, that involve node x_{ij}. This is done by deleting all entries that contain y_{ij} from C_k.

4. If no labels are deleted in step 2, then stop. Otherwise, repeat steps 1 – 3, considering only those constraints that have changed in the current iteration.

As a demonstration of the above steps, Fig. 2.1 shows a CSP before and after the first LCE iteration.

The LCE algorithm is implemented using data structures similar to that presented in [61]. Such an implementation has an optimal time complexity of order $O(cy^2)$, where c is the average of $\mid C_i \mid$, and y is the average of $\mid \mathcal{Y}_i \mid$. It can also be shown that the number of iterations is bounded by cy^2 [61].

The CSP corresponding to our problem is represented by a *scene constraint network* (SCN), which can be defined as follows:

- The set of SCN nodes is composed of tactile and visual features, i.e., $\mathcal{X} = \mathcal{S}^s \cup \mathcal{E}_i^s$, where $\mathcal{E}_i^s \subseteq \mathcal{E}^s$ is a set of visual edges in a perceptual structure (refer to Sections 1.2 and 1.4). We will sometimes refer to the nodes in $\mathcal{S}^s$ as *tactile nodes*, and those in $\mathcal{E}_i^s$ as *visual nodes*. To simplify the presentation, we will consider only one type of tactile features: a point-like surface patch (see Fig. 1.3). Other types can be handled in a similar fashion. Since the scene features are matched with each model object, $\mathcal{O}$, in turn, the domains of the tactile and visual nodes are $\mathcal{S}^m$ and $\mathcal{E}^m$ of $\mathcal{O}$, respectively (refer to Section 1.4).

- The SCN constraints, $\mathcal{C}$, can be classified according to the types of nodes involved in each constraint. The following types of constraints are selected:

  1. Visual constraints (of different orders),
  2. Unary tactile constraints,
  3. Binary tactile constraints, and
  4. Binary visual/tactile constraints.

  Other types (e.g., ternary tactile constraints) can be incorporated as well. However, as will be shown in Section 2.4, we have found that the chosen types of constraints are satisfactory for our purposes.

2.3 Construction of SCN Constraints

In this section, we present the methods used to construct the various types of SCN constraints, outlined at the end of the previous section.

2.3.1 Visual Constraints

Visual constraints are obtained by analyzing the perceptual structure of the sensory data. We have chosen the following types of visual constraints:

1. *Same-Junction Constraint (SJC):* Visual edges of a junction are covered by a *same-junction* hyperedge. Thus, the SJC of a two-edge junction is a binary constraint, while that of a three-edge junction is a ternary one. Vertices of two- and three-edge junctions are likely to correspond to model ones, and so we refer to them as *true* vertices. On the other hand, free vertices are unlikely to correspond to model ones, and so we refer to them as *false* vertices.

2. *Parallel-Edge Constraint (PEC):* Nodes corresponding to edges that are almost parallel and non-collinear are covered by a *parallel-edge* hyperedge.

3. *Collinear-Edge Constraint (CEC):* Nodes corresponding to collinear edges are covered by a *collinear-edge* hyperedge.

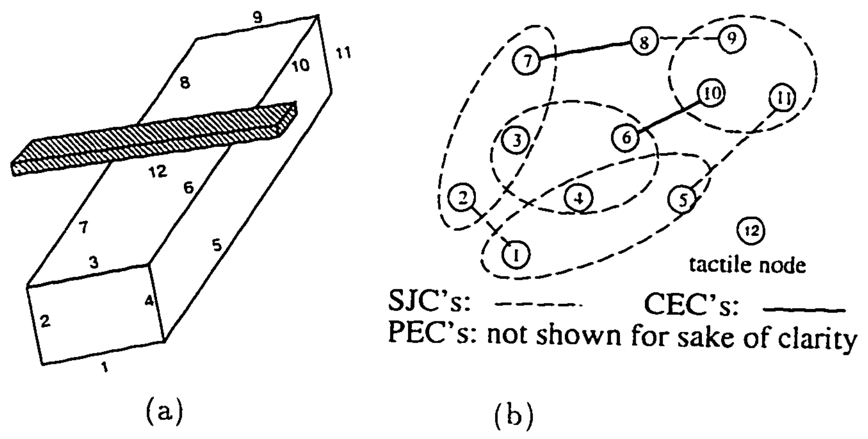


Figure 2.2: An example of an SCN: (a) a scene of a robot finger (the shaded object) in contact with a rectangular polyhedron, (b) corresponding SCN showing some visual constraints.

Other types of visual constraints (e.g., "same-surface" constraints) can also be incorporated into the SCN. As an example, Fig. 2.2(a) shows a scene of a robot finger (the shaded object) in contact with a rectangular polyhedron. In addition, a tactile sensor is assumed to be mounted on the robot finger. The SCN corresponding to this scene, showing some of the visual constraints, is depicted in Fig. 2.2(b).

To simplify the implementation, we replace constraints of order higher than two, by binary constraints between every pair of nodes involved in the original one. For our types of constraints, there is no loss of constraint *tightness* in this conversion. The binary matrix corresponding to each type is obtained through knowledge of the model object under consideration.

2.3.2 Binary Tactile Constraints

We construct a binary constraint between two tactile nodes, (S_i^s, S_j^s), as follows:

- A number of attribute bounds are computed for the scene pair (S_i^s, S_j^s). The chosen attributes are (refer to equation (1.2)):

  1. Angle between patch normals: $\arccos(\mathbf{n}_i^s \cdot \mathbf{n}_j^s)$
  2. Distance between the two patches: $\| \mathbf{p}_i^s - \mathbf{p}_j^s \|$
  3. Projection of $(\mathbf{p}_j^s - \mathbf{p}_i^s)$ on $\mathbf{n}_i^s$: $(\mathbf{p}_j^s - \mathbf{p}_i^s) \cdot \mathbf{n}_i^s$
  4. Projection of $(\mathbf{p}_i^s - \mathbf{p}_j^s)$ on $\mathbf{n}_j^s$: $(\mathbf{p}_i^s - \mathbf{p}_j^s) \cdot \mathbf{n}_j^s$

- For each model pair (S_k^m, S_l^m), we compute bounds on the previous attributes. If scene and model attribute bounds are consistent, then we set entry (k, l) in the constraint matrix to 1; otherwise, we set it to 0.

More details about the above process can be found in [39].

2.3.3 Continuous Visual/Tactile Constraints

In this section, we introduce a number of approximate visual/tactile constraints that are of continuous nature. In the next two sections, we will discuss how to utilize these constraints in order to construct discrete unary tactile and binary visual/tactile constraints for the SCN. The chosen constraints are conditional: each constraint involving a tactile patch, S^s, is obtained by assuming that some model surface, S^m, of an object, $\mathcal{O}$, corresponds to S^s (in this section, we will drop the subscripts associated with scene and model features to simplify the notation). As will be seen later, this assumption is needed to partially compensate for the lack of depth information inherent in the visual data. The chosen constraints are:

1. *Object-Occupancy Constraint (OOC):* An $OOC(S^m)$ of an object, $\mathcal{O}$, provides an approximate bound on the 3-D space occupied by $\mathcal{O}$, given that S^m corresponds to S^s (S^s is dropped from the parameter lists of all continuous constraints to simplify the notation). Constraint $OOC(S^m)$ is constructed as follows. A match between model surface S^m and tactile patch S^s restricts the location of model object $\mathcal{O}$ as follows: 1) Object $\mathcal{O}$ can translate along the plane of S^s such that S^s remains inside S^m, and 2) it can rotate freely about the normal of S^s, $\mathbf{n}^s$. Thus, a possible simple approximation of the space occupied by $\mathcal{O}$, given that S^m corresponds to S^s, is a cylinder whose axis is parallel to $\mathbf{n}^s$ (see Fig. 2.3). Dimensions of such a cylinder, which can be computed off-line, are presented in Appendix A.1.

2. *Vertex-Depth Constraint (VDC):* $VDC(S^m, V^s)$ provides a bound on the depth of a model vertex that projects to V^s. This bound, $[\alpha^{min}, \alpha^{max}]$, is obtained by computing the intersection of $OOC(S^m)$ with the *line of sight* of V^s, $L(V^s)$. The values α^{min} and α^{max} represent the minimum and maximum distances, respectively, between the camera viewpoint and the section of $L(V^s)$ inside $OOC(S^m)$

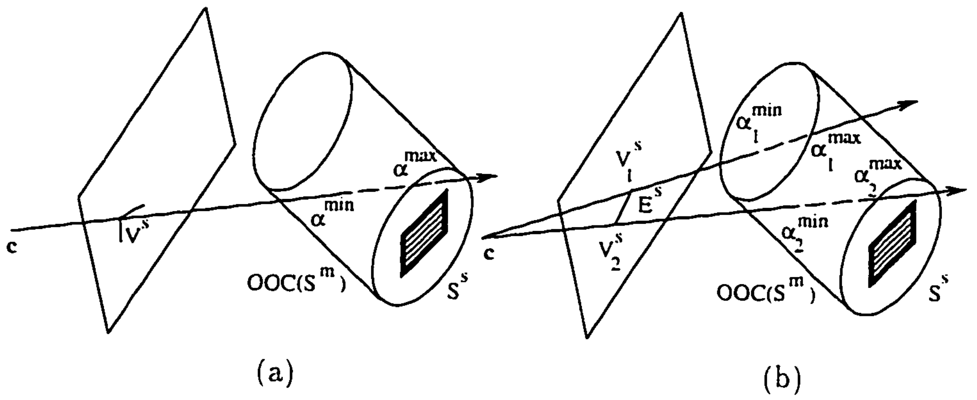


Figure 2.3: (a) Vertex-depth constraint $VDC(S^m, V^s) = [\alpha^{min}, \alpha^{max}]$, (b) Edge-depth constraint $EDC(S^m, E^s) = ([\alpha_1^{min}, \alpha_1^{max}], [\alpha_2^{min}, \alpha_2^{max}])$.

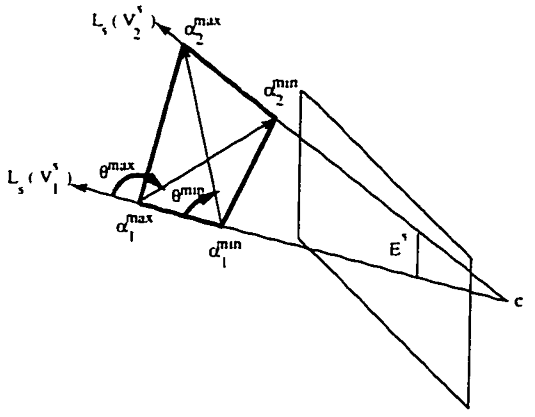


Figure 2.4: An illustration of an edge-angle constraint, $EAC(S^m, E^s) = [\theta^{min}, \theta^{max}]$.

(see Fig. 2.3(a)). If $OOC(S^m)$ does not intersect $L(V^s)$, then $VDC(S^m, V^s)$ is a *null* interval. Notice how $OOC(S^m)$ is utilized to partially compensate for the lack of depth information in the visual data.

3. *Edge-Depth Constraint (EDC):* $EDC(S^m, E^s)$ provides a bound on the depth of a model edge that projects to E^s. This constraint is obtained by computing the intersection of $OOC(S^m)$ with the *plane of sight* of E^s, $P(E^s)$. An example of the intersection process is shown in Fig. 2.3(b). Constraint $EDC(S^m, E^s)$ is simply the vertex-depth constraints corresponding to the two end vertices of E^s, (V_1^s, V_2^s); i.e.,

$$EDC(S^m, E^s) \;=\; (VDC(S^m, V_1^s), VDC(S^m, V_2^s)).$$

4. *Edge-Angle Constraint (EAC):* $EAC(S^m, E^s)$ provides a bound on the angle between a model edge that projects to E^s and line of sight $L(V_1^s)$. An illustration of $EAC(S^m, E^s)$ is shown in Fig. 2.4. The equations used to determine this constraint are presented in Appendix A.2.

5. *Edge-Length Constraint (ELC):* $ELC(S^m, E^s)$ provides a bound on the length of a model edge whose vertices project to those of E^s. The equations used to determine this constraint are presented in Appendix A.3.

2.3.4 Unary Tactile Constraints

The unary tactile constraint associated with a tactile node, S^s, is obtained as follows. For each model surface S^m, we examine $VDC(S^m, V^s)$ for all scene vertices V^s. If $VDC(S^m, V^s)$ is null for *any* one of the scene vertices V^s, then we can conclude that the hypothesis "S^m corresponds to S^s" is *inconsistent* with the assumption "V^s belongs to the model object under consideration". Accordingly, we can eliminate S^m from the domain of S^s.

Notice that although this constraint is defined on a tactile node, it is obtained by utilizing visual data. In addition, notice how *global* data are utilized to reach a *local* decision.

2.3.5 Binary Visual/Tactile Constraints

In this section, we present a method for constructing a binary constraint defined on a pair of tactile and visual nodes, (S^s, E^s). The straightforward construction method examines each model surface/edge pair (S^m, E^m). If (S^m, E^m) is consistent with the scene pair (S^s, E^s), then the corresponding entry in the constraint matrix is set to 1; otherwise, it is set to 0. The model pair (S^m, E^m) is said to be consistent with the scene one (S^s, E^s), if the following conditions can be satisfied simultaneously:

1. The inward normal of the tactile patch S^s is aligned with the outward normal of the model surface S^m.

2. The tactile patch S^s lies inside polygon S^m.

3. The model edge E^m lies inside plane of sight $P(E^s)$, such that its projection on the image plane *covers* E^s.

4. Each vertex of E^m, V_l^m ($l = 1, 2$), lies along line of sight $L(V_l^s)$, if V_l^s is a true vertex (defined in Section 2.3.1).

Checking the validity of the above conditions is computationally expensive, especially when the uncertainty of the scene features is taken into consideration. To handle such a problem, we propose an alternative method which is approximate but much more efficient; only very few comparison operations are needed to determine the value of each entry in the constraint matrix defined on (S^s, E^s). This method, which utilizes the approximate constraints presented in Section 2.3.3, can be outlined as follows:

1. A number of *transformation-invariant* attributes of a model surface/edge pair are selected. Values (or bounds) of these attributes are computed for all possible combinations of model surface/edge pairs, (S^m, E^m). This process is performed off-line.

2. At run-time, we use (S^s, E^s) to derive approximate bounds on the attributes of the model pair (S^m, E^m) that corresponds to (S^s, E^s).

3. Each possible model pair (S^m, E^m) is examined. If the attribute values/bounds of (S^m, E^m) are *consistent* with those of (S^s, E^s), then the entry corresponding to (S^m, E^m) in the constraint matrix is set to 1; otherwise, it is set to 0.

The above stages are described in detail in the following three sections.

1. Model Surface/Edge Attributes

We have chosen the following transformation-invariant attributes of a model surface/edge pair, (S^m, E^m) (see Fig. 2.5):

1. *Edge Length* (L^m)*:* length of edge E^m

2. *Edge Angle* (A^m)*:* angle between edge E^m and the normal of surface S^m, $\mathbf{n}^m$. Note that, in our system, model edges are considered as directed entities. Thus,

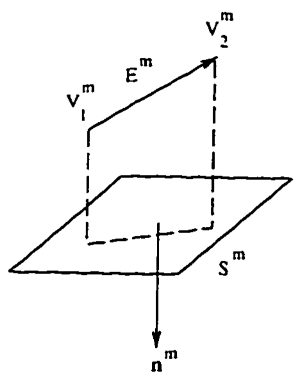


Figure 2.5: A model surface/edge pair (S^m, E^m).

an edge represented by a tuple (V_1^m, V_2^m) is considered different from that represented by (V_2^m, V_1^m).

3. *Vertex/Surface Perpendicular Distance* (P_l^m)*:* the perpendicular distance between vertex V_l^m $(l = 1, 2)$ and surface S^m

4. *Bound on Vertex/Surface Distance* (DB_l^m)*:* the range of distances between vertex V_l^m $(l = 1, 2)$ and surface S^m

Thus, the total number of model attributes of (S^m, E^m) is six. Mathematical definitions of those attributes are presented in Appendix B.1.

We have tried an additional number of attributes (e.g., range of angles between $\mathbf{n}^m$ and the set of vectors from V_l^m to S^m). However, experimentally, we have found that those attributes are marginally useful when used in conjunction with the chosen ones. Accordingly, they have been discarded.

2. Scene Surface/Edge Attribute Bounds

Scene surface/edge attribute bounds, which correspond to the model attributes, are obtained by assuming that a model surface, S^m, corresponds to S^s. As shown in Section 2.3.3, this assumption is needed to partially compensate for the lack of depth information inherent in the visual data. Thus, the scene attribute bounds are reevaluated for *every* possible model surface S^m. The attribute bounds of a scene pair (S^s, E^s) are (refer to Fig. 2.6):

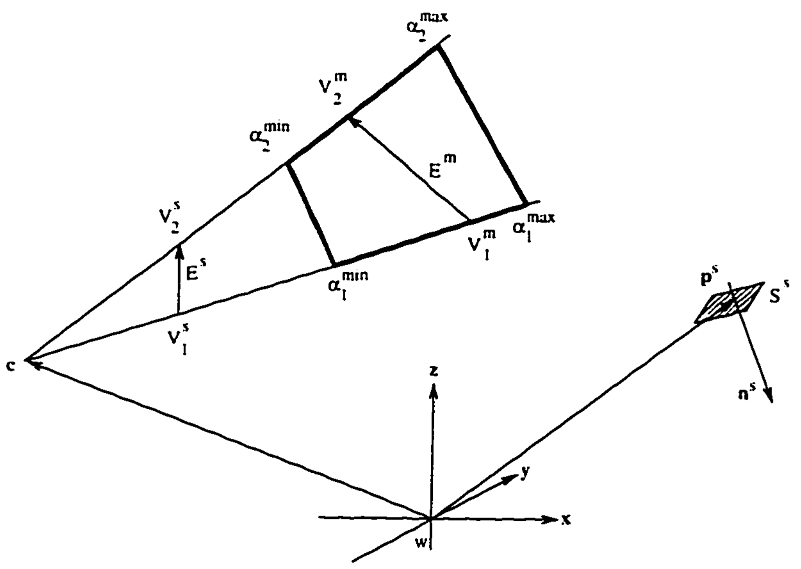


Figure 2.6: A scene surface/edge pair (S^s, E^s).

1. *Bound on Edge Length* $(LB^s(S^m))$*:* provides a bound on the length of E^m, the model edge that projects to E^s. It is easy to see that $LB^s(S^m)$ is the same as edge-length constraint $ELC(S^m, E^s)$.

2. *Bound on Edge Angle* $(AB^s(S^m))$*:* provides a bound on the angle between E^m, and the normal of S^s, $\mathbf{n}^s$. This bound is obtained by utilizing edge-angle constraint $EAC(S^m, E^s)$.

3. *Bound on Vertex/Surface Perpendicular Distance* $(PB_l^s(S^m))$*:* provides a bound on the perpendicular distance between, V_l^m, the model vertex that projects to V_l^s $(l = 1, 2)$ and S^s. This bound is obtained by utilizing vertex-depth constraint $VDC(S^m, V_l^s)$.

4. *Bound on Vertex/Surface Distance* $(DB_l^s(S^m))$*:* provides a bound on the distance between S^s, and V_l^m $(l = 1, 2)$. This bound is also obtained by utilizing the vertex-depth constraint $VDC(S^m, V_l^s)$.

Thus, the total number of scene attribute bounds is six. Mathematical definitions of the above attributes are presented in Appendix B.2.

3. Consistency Checking

Define the following functions:

- inside(p, b) returns *true* if point p is inside interval b; otherwise it returns *false*.
- overlapping(b_1, b_2) returns *true* if bounds b_1 and b_2 are overlapping; otherwise it returns *false*.
- true_vertex(V^s) returns *true* if V^s is a true vertex (defined in Section 2.3.1); otherwise it returns *false*.

A model pair (S^m, E^m) is considered consistent with a scene one (S^s, E^s), only if all the following conditions hold:

1. **if** true_vertex(V_1^s) and true_vertex(V_2^s) **then** inside($L^m, LB^s(S^m)$)
   else $L^m >$ lower bound of $LB^s(S^m)$
2. inside($A^m, AB^s(S^m)$)
3. **for** $l = 1, 2$,
   if true_vertex(V_l^s) **then** inside($P_l^m, PB_l^s(S^m)$)
4. **for** $l = 1, 2$,
   if true_vertex(V_l^s) **then** overlapping($DB_l^m, DB_l^s(S^m)$)

It is easy to see that our approximate method is very efficient; only very few comparison operations are needed to determine the value of each constraint entry. In the next section, we will demonstrate that the proposed set of constraints, although approximate, is still discriminative enough to eliminate many erroneous model objects and achieve significant reduction in scene/model correspondence uncertainty.

2.3.6 Extraction of Perceptual Structures

Perceptual structures are extracted from the visual data as follows:

1. Straight edges are extracted from the image.
2. Edges that belong to the robot hand are discarded, since they can be determined easily through knowledge of the hand's geometric description and the robot kinematics.

3. For each edge vertex, we look for neighboring edge vertices. Visual nodes corresponding to edges with mutually adjacent vertices are covered by a same-junction constraint (SJC).

4. Two visual nodes are assumed to belong to the same perceptual structure, iff there is path of SJC's between the two nodes. This is an equivalence relation, and so it partitions the visual nodes into disjoint subsets.

5. For each edge, we look for parallel and collinear ones in the same perceptual structure. Nodes corresponding to parallel and collinear edges are covered by parallel-edge and collinear-edge constraints (PEC and CEC), respectively.

Since it is generally difficult to determine which perceptual structure belongs to the object of interest, each structure, along with the tactile data, is considered in turn.

It should be noted that the above method does not guarantee that the extracted perceptual structures are valid, due to the possibilities of special object configurations, special viewpoints, and imperfections of feature extraction. Error recovery techniques are needed to handle the possibility of erroneous perceptual structures. This issue, although very important, is not considered in this work.

2.4 Experimental Results

In this section, we present a number of simulated and real experiments to demonstrate the performance of the proposed screening technique. To estimate the reduction in scene/model correspondence uncertainty, achieved by LCE, we define the following performance measure:

$$\rho = \frac{n_b - n_a}{n_b} \qquad (2.1)$$

where n_b and n_a are the sum of all the *one* entries in the SCN constraints, before and after LCE, respectively. Thus, ρ lies in the range $[0, 1]$. Maximum uncertainty reduction occurs when $\rho = 1$, in which case, we can directly conclude that the object is erroneous and so it can be discarded. On the other hand, no reduction occurs when $\rho = 0$. Notice that if $\rho \neq 1$, then we cannot directly reach a conclusion on whether

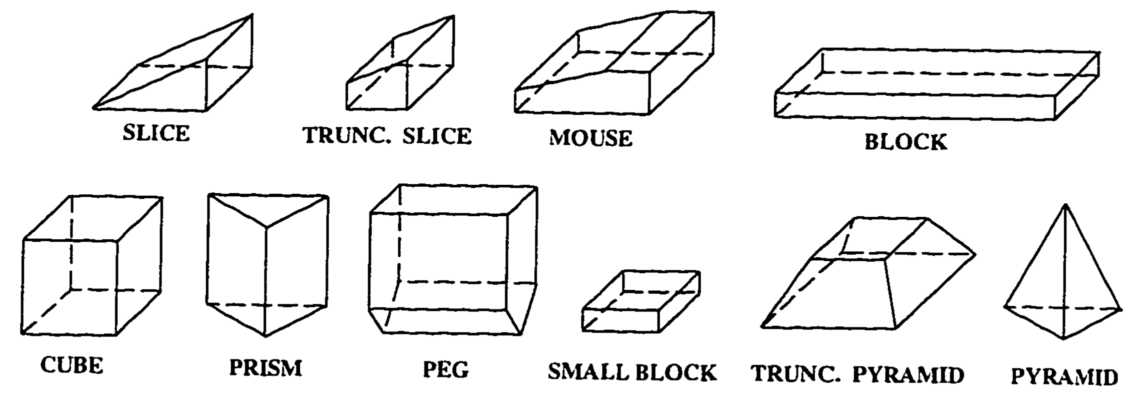


Figure 2.7: Model database.

the object is valid and so further investigation would be required through generation and verification of pose hypotheses.

Finally, we have chosen a model database consisting of ten objects. Those objects are shown in Fig. 2.7.

2.4.1 Uncertainty Handling

Uncertainty of the sensory data has been ignored so far in the discussions, mainly for the sake of presentation clarity. In practice, however, uncertainty has to be dealt with in a formal way. The following bounded-error model is used to model the uncertainty of visual and tactile features:

1. *Tactile Patch:* The actual location of the patch is assumed to be within some distance, Δr^t, of the estimated one. The actual patch normal is assumed to be within some angle, $\Delta\theta^t$, of the estimated one.

2. *Visual Vertex:* The actual visual vertex is assumed to be within some pixel distance, Δr^v, of the estimated one.

Thresholds which are used throughout the system are automatically determined based on the above error model. In addition, the object-occupancy constraints are appropriately dilated to handle the uncertainty of the tactile patches.

2.4.2 A Detailed Experiment

Fig. 2.8(a) shows a synthetic scene of a three-fingered robot hand grasping model object SLICE. In this experiment, as well as the rest of the experiments in this

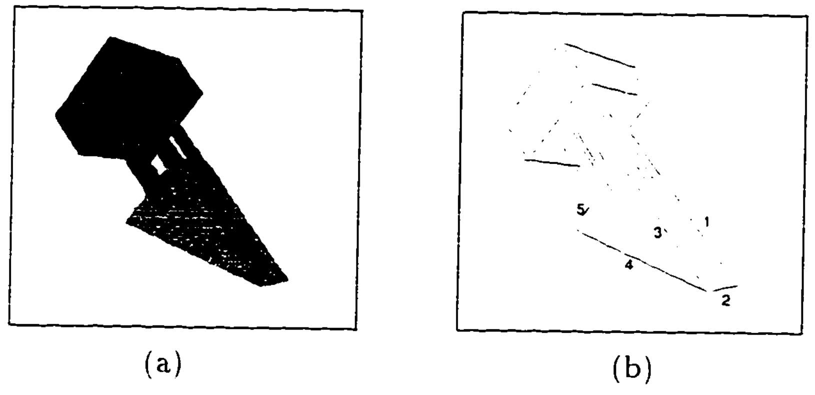


Figure 2.8: A synthetic scene of a robot hand grasping model object SLICE: (a) scene. (b) extracted edges.

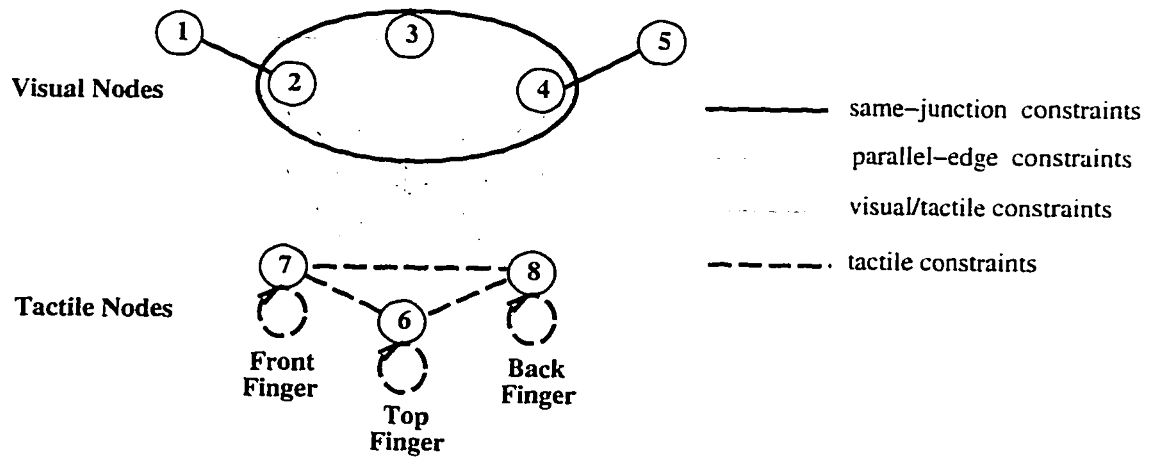


Figure 2.9: The SCN associated with the scene shown in Fig. 2.8.

work, the grasp configuration is chosen empirically. A tactile sensor is assumed to be mounted on each of the robot fingers to provide a point-like surface patch. In this experiment, as well as the simulation ones presented in the next section, uncertainty is introduced in the tactile data by randomly perturbing the position and the normal of each tactile patch, assuming $\Delta r^t = 5$ mm, and $\Delta\theta^t = 5°$. The visual image is processed to extract a number of straight edges which are shown in Fig. 2.8(b). Edges that belong to the robot hand are then eliminated. The remaining edges, the five labelled ones in Fig. 2.8(b), are perceived to belong to the same perceptual structure. The SCN corresponding to such a scene, shown in Fig. 2.9, consists of eight nodes, five of them are visual and three are tactile. In addition, from Fig. 2.9, we can see that the number of SCN constraints is 25 (4 visual, 15 visual/tactile, 3 unary tactile, and 3 binary tactile). Node domains and constraint matrices of the CSP are reconstructed for each model object, as explained in the previous section. For example, let us consider the valid model object, SLICE. From the shape of such an object (see Fig. 2.7), we can see that it consists of five surfaces, and nine edges. Thus, the domains of tactile and visual nodes, $\mathcal{S}^m$ and $\mathcal{E}^m$, are of sizes 5 and 18, respectively (recall that each actual model edge is represented as two directed ones). A sample of five constraint matrices corresponding to object SLICE is shown in Fig. 2.10.

The LCE algorithm, outlined in Section 2.2.2, is applied to each constructed CSP. Fig. 2.10 shows five constraints before and after LCE, when object SLICE is considered. Notice how knowledge of inconsistent constraint labels (those corresponding to *zero* rows or columns) is propagated to neighboring constraints through LCE. For example, in constraint (8, 2), we can see that four model edges (columns 9, 10, 13 and 14) are inconsistent with scene edge 2. This knowledge is propagated to neighboring constraints that involve node 2 by zeroing the corresponding rows or columns (see, for example, the four corresponding columns of constraint (1, 2) before and after LCE).

Results obtained by running our system are shown in Table 2.1. Columns in this table are:

1. *Model Object:* the model object under consideration

2. n_b *(n_a):* number of 1's in the SCN before (after) LCE

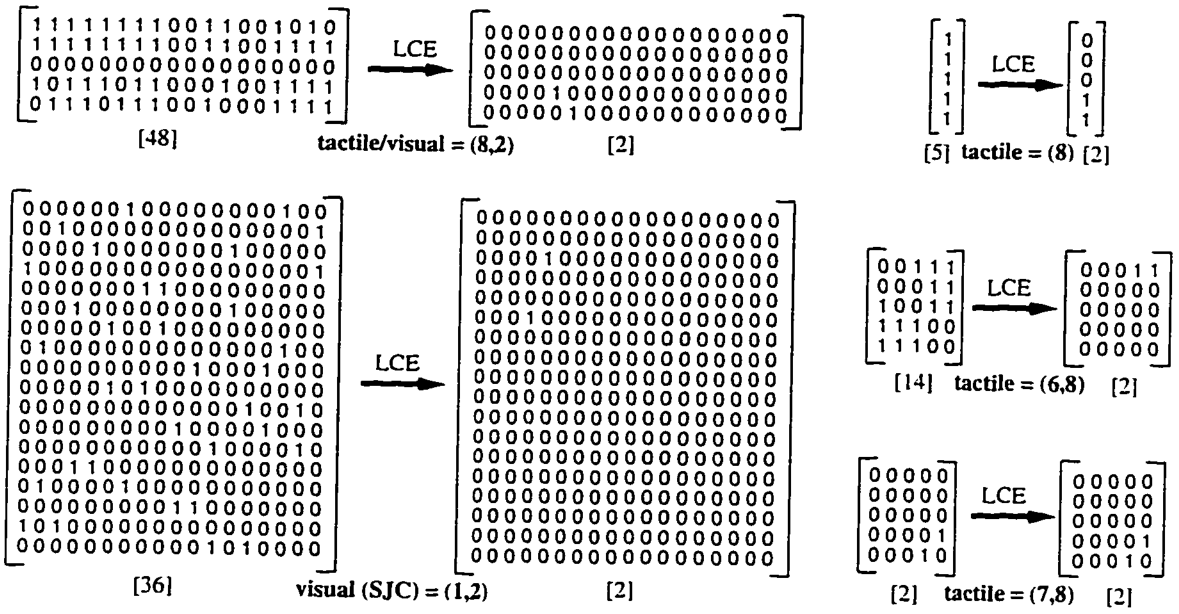


Figure 2.10: Examples of five SCN constraints, before and after LCE, when model object SLICE is considered. Rows or columns of size five correspond to model surfaces, while those of size 18 correspond to model edges. Numbers between square brackets refer to the number of 1's in the associated constraint matrices.

Model Object	n_b	n_a	ρ	I	Touch
SLICE (scene object)	833	69	0.92	3	N/A
TRUNC. SLICE	691	0	1.00	0	No
MOUSE	961	0	1.00	0	No
BLOCK	1064	0	1.00	1	No
CUBE	690	0	1.00	0	No
PRISM	673	0	1.00	0	Yes
PEG	1323	0	1.00	0	Yes
SMALL BLOCK	476	0	1.00	0	No
TRUNC. PYRAMID	1187	0	1.00	0	Yes
PYRAMID	720	0	1.00	0	Yes

Table 2.1: Summary of the experimental results for the scene shown in Fig. 2.8.

3. *I:* number of LCE iterations (see the LCE algorithm in Section 2.2.2)

4. *ρ:* the performance criterion defined in (2.1)

5. *Touch:* an indication of whether tactile data alone can determine the invalidity of the object under consideration

From Table 2.1, we can observe the following:

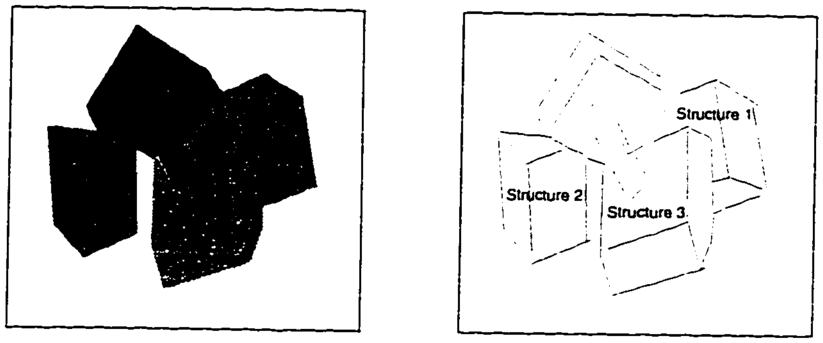
- LCE is robust in eliminating erroneous objects without either generating or verifying pose hypotheses. In this experiment, *all* the erroneous objects are eliminated by LCE. Furthermore, except for model object BLOCK, all erroneous objects are eliminated *without* any LCE iterations. This occurs when one or more of the SCN constraints is a zero matrix.

- LCE is robust in reducing the correspondence uncertainty. In this experiment, we see that the average of ρ is almost 1. Thus, we can expect a drastic reduction in the computational load on subsequent hypothesis generation and verification modules. Furthermore, for the valid model object (SLICE), the value of ρ is about 0.92. Thus, we can also expect significant improvement in performance, even if the task is localizing a known object. As an illustration, notice how LCE reduces the number of 1's in the constraint matrices shown in Fig. 2.10.

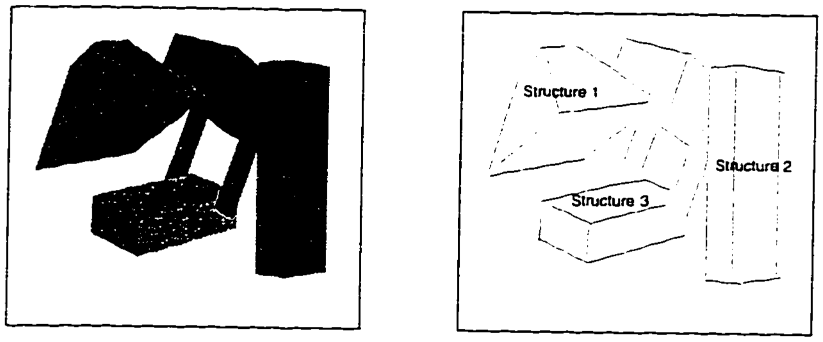
- LCE is computationally very efficient. Only very few LCE iterations are required to enforce consistency (3 for SLICE, 1 for CUBE, and none for the remaining objects). This is despite the fact that the theoretical upper bound is high (see Section 2.2.2).

- Integration of visual and tactile data is useful in screening objects. All erroneous objects (nine) are eliminated using vision and touch, as opposed to only four when using touch alone (see the last column in Table 2.1).

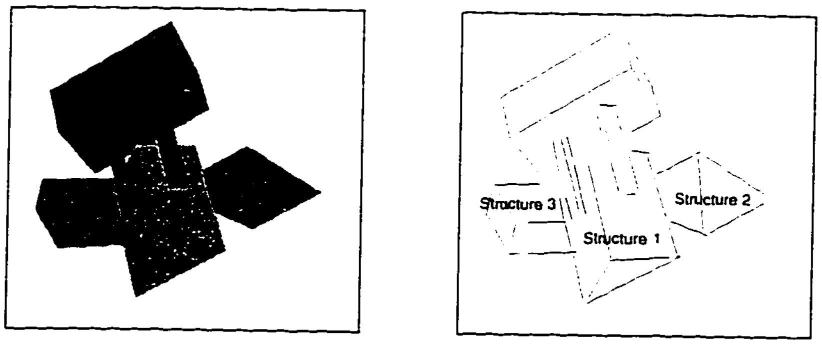
- Although approximate, the CSP visual/tactile constraints are very satisfactory for our purposes. This is clearly demonstrated by the previous observations.



(a)



(b)



(c)

Figure 2.11: Three synthetic scenes along with extracted perceptual structures.

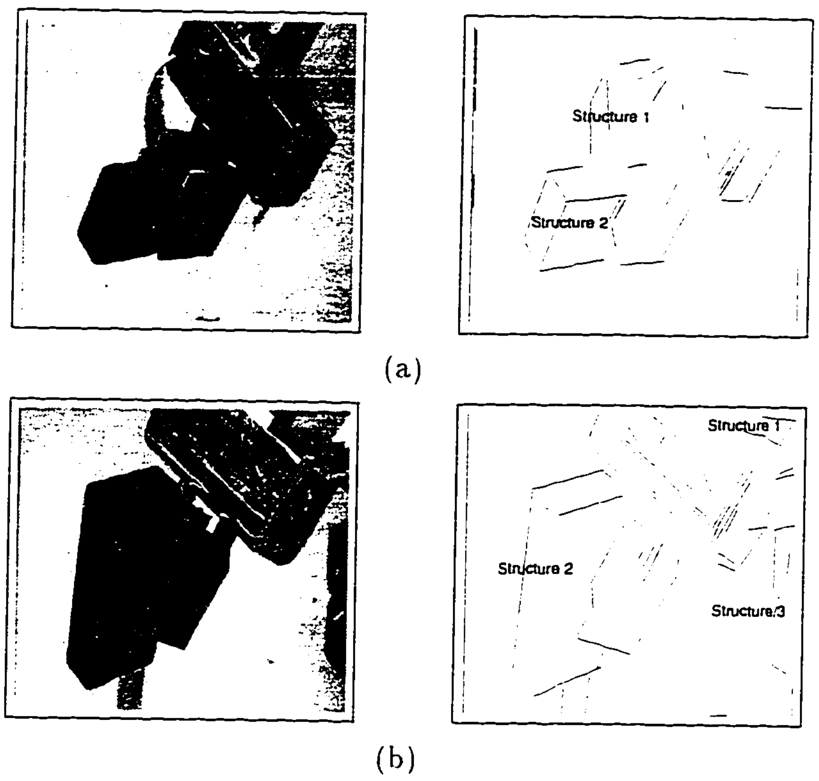


Figure 2.12: Two real scenes along with extracted perceptual structures.

Perceptual Structure	Valid Structure	Valid Object		Erroneous Objects			Average	
		ρ	I	Total	Eliminated	Touch	ρ	I
Fig. 2.11(a)-3	Yes	0.93	4	9	9	7	0.99	0.8
Fig. 2.11(b)-3		0.75	2		8	3	0.96	0.9
Fig. 2.11(c)-1		0.88	6		9	9	0.99	0.6
Fig. 2.12(a)-2		0.43	2		7	7	0.87	1.1
Fig. 2.12(b)-2		0.77	4		8	7	0.97	1.0
Fig. 2.11(a)-1	No	N/A	N/A	10	8	7	0.95	1.2
Fig. 2.11(a)-2					10	7	1.00	0.0
Fig. 2.11(b)-1					10	3	1.00	0.0
Fig. 2.11(b)-2					10	3	1.00	0.0
Fig. 2.11(c)-2					10	9	1.00	0.0
Fig. 2.11(c)-3					10	9	1.00	0.0
Fig. 2.12(a)-1					7	7	0.90	0.8
Fig. 2.12(b)-1					10	7	1.00	0.0
Fig. 2.12(b)-3					10	7	1.00	0.0

Table 2.2: Summary of the experimental results for the scenes shown in Figs. 2.11 and 2.12.

2.4.3 Summarized Experiments

To reinforce the conclusions obtained above. we have performed a number of simulated and real experiments.

Fig. 2.11 shows three synthetic scenes of a robot hand grasping an object. We have added erroneous objects, at random poses, to those scenes in order to test the performance of the technique when applied to erroneous perceptual structures.

The setup of the real experiments can be described as follows. A Schunk parallel-jaw gripper, mounted on a PUMA 260 robot arm, is used for object manipulation. The robot workspace is monitored by an NEC CCD camera, which provides visual images of dimensions 480×512. Tactile patches are provided by tactile sensors mounted on the gripper jaws. We have used an Interlink piezo-resistive tactile sensor, which is a 16×16 planar array of dimensions 25.4 mm $\times$ 25.4 mm. The position of a tactile patch provided by such a sensor is assumed to be the center of the *tactile polygon* formed through contact with the object. Fig. 2.12 shows a couple of real scenes of model object BLOCK being grasped by the parallel-jaw gripper.

Results obtained by feeding the above five scenes to our system are summarized in Table 2.2. Columns in this table are:

1. *Perceptual Structure:* the perceptual structure under consideration
2. *Valid Structure:* an indication of whether the perceptual structure is valid (i.e., belongs to the object that is grasped by the hand)
3. *Valid Object (ρ,I):* ρ and number of LCE iterations for the valid object
4. *Erroneous Objects (Total):* total number of erroneous objects
5. *Erroneous Objects (Eliminated):* number of eliminated objects
6. *Erroneous Objects (Touch):* number of objects that can be eliminated using touch alone
7. *Average (ρ,I):* averages of ρ and LCE iterations for all objects

From Table 2.2, we can derive similar conclusions to those obtained earlier. Furthermore. notice the robustness of the technique when applied to invalid structures.

In seven of the nine invalid structures. all objects are eliminated *without* any LCE iterations. In the remaining two scenes. only very few erroneous objects pass the LCE test (two and three objects for structures Fig. 2.11(a)-1 and Fig. 2.12(a)-1. respectively).

2.5 Conclusions

We have presented a vision/touch-based technique for screening model objects. The 3DOR problem is formulated as a CSP to provide a unified framework for integrating different types of sensory data. A novel method is presented for efficiently constructing a CSP corresponding to a combination of visual and tactile features. Efficiency is achieved by developing a number of approximate visual/tactile constraints. which enable determining the value of each entry in visual/tactile constraints using very few comparison operations. The consistency of the scene data with the model object under consideration is determined by enforcing arc consistency. This step has the advantage of efficiently utilizing all the relevant scene data. early in the recognition process. in order to reduce the correspondence uncertainty between scene and model features. Furthermore. it eliminates many erroneous model objects and perceptual structures efficiently without either generating or verifying any object/pose hypotheses. Thus. by passing the arc-consistent CSP to subsequent recognition modules. we can expect significant reduction in the computational load on them. Performance of the proposed technique has been demonstrated using simulated and real experiments.

Chapter 3

Vision/Touch-Based Hypothesis Generation

3.1 Introduction

There are several approaches reported in the literature for generating object/pose hypotheses. These approaches can be classified into the following three categories.

1) Tree Search: A tree search is used to explore all the correspondence possibilities between a subset of scene features and model ones. This approach has been taken by many researchers (e.g., [31, 32, 34, 39, 40, 73]). The tree is usually constructed in a data-driven fashion: i.e., for a given scene feature, all model features are examined for consistency with the scene one. If the relevant scene data can not be isolated, then a *null* match is needed to handle extraneous scene features (e.g., [34, 40]). The tree is pruned by utilizing unary and binary viewpoint-invariant constraints. Examples of unary constraints are surface type, surface area, edge length and intrinsic feature parameters (e.g., radius of a sphere). Binary constraints include angle between surface orientations, and connectivity information.

Although only few features are sufficient to generate hypotheses, some systems extend the search to include many scene features, without computing pose transformation (e.g., [39, 40, 73]) . This kind of extended search might be needed, if the scene features are too noisy to generate reliable pose estimates.

2) Local Feature-Set Matching: In this approach, a set of local scene features is selected and matched with a corresponding model one to generate one or more pose hypotheses. Scene feature sets that have been used for hypothesis generation include:

1. A three-edge 2-D junction where one of the edges is completely visible [49, 74, 75],

2. A set of three 2-D edges [23, 28],

3. A pair of adjacent 3-D surfaces [27],

4. A set of three 3-D surfaces [46],

5. A 2-D ellipse [65], and

6. A supersegment, which is a set of connected 3-D line segments [70].

An advantage of such an approach over tree search is that it permits fast access to model feature sets that are consistent with a given scene one. This can be achieved by building an index (or a hash table) that is based on viewpoint-invariant attributes of the scene set (e.g., [21, 70]). Furthermore, indexing schemes enable fast identification of erroneous scene feature sets, since they are unlikely to correspond to any model one. It should be noted, however, that tree-search hypothesis generation might still be more suitable if the scene features are too noisy to generate reliable hypotheses using a minimal number of features.

3) Off-line Recognition Strategies: This approach involves analyzing each model object off-line, in order to tailor a strategy, often called a strategy tree, for localizing the object from scene data. The existing off-line recognition strategies can be classified into two categories:

1. Matching is organized as a tree search. However, the search is mainly model-driven; i.e., for a given model feature, scene features are examined for consistency with the model one. The off-line strategy determines the order of model features to be considered for matching in every stage of the search. Model features are ordered based on several criteria such as covisibility with previously matched features (which can be determined by aid of aspect graphs), constraining power (how many degrees of freedom are constrained by the feature), and detectability in the scene data. The recognition strategy might also include information on where to search for a given model feature in the scene data. Such information can be obtained by utilizing partial knowledge of the object

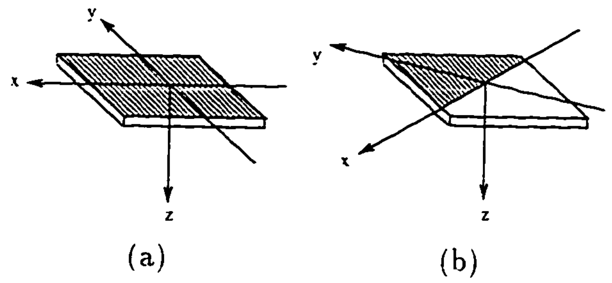


Figure 3.1: The selected tactile features and associated frames: (a) surface patch: 1) origin: center of the patch. 2) z axis: outward normal of the model surface in contact with the patch. 3) x and y axes: placed arbitrarily in the tactile-patch plane. (b) surface-edge patch: 1) origin: mid point of the edge, 2) z axis: outward normal of the model surface in contact with the patch. 3) y axis: normal to the edge towards the interior of the patch. 4) x axis: along the edge such that the right-hand rule is satisfied.

pose, which is gathered through previous scene/model matches. If there is no matching scene feature, then the feature extraction module can be invoked to closely re-examine the search area (e.g., [12]). Thus, an important advantage of this approach is capability of handling imperfections of feature extraction. Most existing off-line strategies belong to this category (e.g., [3, 12, 36, 42]). This approach appears more suitable than data-driven tree search, if the object to be recognized is placed in a cluttered scene. This is because data-driven tree search would generate too many null matches, compared to model-driven search.

2. Model objects are represented as aspect graphs. A decision tree is used to determine the aspect corresponding to the scene object. Once this is done, predetermined 2.5-D matching procedures are used to estimate the object's pose for every possible aspect. An example of this category is the work presented by Gremban and Ikeuchi [38]. A limitation of this approach is incapability of handling occluded objects.

In our work, we have chosen to generate hypotheses by matching a local set of scene features, in order to enable fast access of relevant model ones. This local set consists of a visual junction, of any number of edges, and a tactile feature. For the sake of presentation, we will focus on only two types of tactile features. The first type is a polygonal surface patch, which is formed when the object surface in contact with a polygonal tactile sensor totally covers it (Fig. 3.1(a)). The second one is also a polygonal surface patch, with an edge that is known to correspond to a model one. This type of tactile data is formed when the contact surface partially covers the sensor and only one edge appears in the sensor array (Fig. 3.1(b)). Henceforth, we will refer to surface and surface-edge patches as *S-patch* and *SE-patch*, respectively. Those features correspond to model surface and model surface-edge pair, which we will refer to as *S-surface* and *SE-surface*, respectively.

The proposed pose generation technique can be outlined as follows. First, a subset of the object's degrees of freedom (DOF's) is determined by matching the tactile feature with a corresponding model one. The remaining DOF's, which cannot be determined from the tactile feature, are then obtained by matching the visual one. In addition, we develop a couple of filtering techniques in order to reduce the number of scene/model matches. Those techniques utilize some of the continuous visual/tactile constraints presented in the previous chapter. In particular, one of them utilizes the object-occupancy constraint, while the other uses the edge-angle constraint to derive bounds on model transformation-invariant attributes (refer to Section 2.3.3).

As will be experimentally demonstrated in Section 3.4, the advantages of the proposed vision/touch-based localization technique over vision-based ones are:

1. It is capable of determining the object pose in heavily occluded visual images that are problematic to vision-based methods.

2. The average number of generated hypotheses, per scene feature set, is considerably less than the number of those generated visually. This advantage can greatly reduce the computational load on the subsequent hypothesis verification stage.

3. The accuracy of estimating the object depth (with respect to the visual sensor) can be significantly better when vision is integrated with touch.

The rest of the chapter is organized as follows. In the next section, we describe the pose generation technique in detail. The filtering techniques are presented in Section 3.3. In Section 3.4, we present experimental results using simulated and real data.

and finally, in Section 3.5, we provide conclusions[1].

3.2 Pose Generation

In this section, we present the technique used to generate pose hypotheses by integrating visual and tactile data.

3.2.1 Overview

As mentioned, the scene feature set used to generate pose hypotheses consists of a visual junction and a tactile feature, which is either an S- or SE-patch in our case. The six object DOF's are determined as follows:

1. The tactile S- or SE-patch is matched with a model S- or SE-surface, respectively, to determine a subset of the object's six DOF's. The number of these DOF's is three and five, for S- and SE-patches, respectively.

2. The remaining DOF's are obtained by matching the visual junction with a model one.

A tactile-feature frame is chosen as the reference frame (see Fig. 3.1). As will be seen later, this choice simplifies the computation of the object DOF's. Furthermore, it enables the determination of the touch-based DOF's off-line, thereby reducing the on-line computational requirements.

The matching process can be outlined as follows:

1. A set of *partial* pose hypotheses, $\mathcal{T}$, is generated off-line. Each partial hypothesis, ${}^{i}\mathbf{T} \in \mathcal{T}$, is obtained by matching the tactile patch with the *ith* corresponding model feature. Thus, $\mathcal{T}$ corresponds to the touch-based DOF's.

2. Junctions of the model object, $\mathcal{J}^m$, are transformed off-line by multiplying them with each partial hypothesis ${}^{i}\mathbf{T}$ to form ${}^{i}\mathcal{J}^m$. The set of the partially-transformed junction sets, $\{{}^{i}\mathcal{J}^m\}$, is stored in a *junction database*.

[1]A preliminary version of this work was published in *Proc. IEEE/RSJ/GI Int. Conf. Intelligent Robots and Systems*, Munchen, Germany, 1994 [15].

3. At run-time, the two filtering techniques, presented in Section 3.3, are applied to reduce the number of model junctions, in the junction database, that are used in matching. Each model junction accepted by the filtering process, ${}^{i}J^{m} \in {}^{i}\mathcal{J}^{m}$, is then matched with the visual junction to determine the remaining DOF's that complement the partial hypothesis ${}^{i}\mathbf{T}$.

We have chosen to represent the 3-D pose of an object, $\mathcal{O}$, with respect to a frame, W, by three rotations about the principal axes of frame W, followed by three translations along the same axes. The transformation matrix, ${}^{W}\mathbf{T}_{O}$, corresponding to the 3-D pose of object $\mathcal{O}$ with respect to frame W can be written as follows [63]:

$$\begin{aligned} {}^{W}\mathbf{T}_{O} &= Trans(x,t_x)Trans(y,t_y)Trans(z,t_z)Rot(z,\theta_z)Rot(y,\theta_y)Rot(x,\theta_x) \\ &= \begin{pmatrix} c\theta_z c\theta_y & c\theta_z s\theta_y s\theta_x - s\theta_z c\theta_x & c\theta_z s\theta_y c\theta_x + s\theta_z s\theta_x & t_x \\ s\theta_z c\theta_y & s\theta_z s\theta_y s\theta_x + c\theta_z c\theta_x & s\theta_z s\theta_y c\theta_x - c\theta_z s\theta_x & t_y \\ -s\theta_y & c\theta_y s\theta_x & c\theta_y c\theta_x & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \quad (3.1)$$

where $Rot(\omega,\theta_\omega)$ is the rotation matrix of angle θ_ω about ω axis, $Trans(\omega,t_\omega)$ is the translation matrix of t_ω along ω axis, $c\theta = cos(\theta)$, and $s\theta = sin(\theta)$.

By our choice of the tactile-feature frame (Fig. 3.1), matching a tactile S-patch with a model S-surface determines three DOF's, θ_x, θ_y and t_z, while matching an SE-patch with an SE-surface determines additional two DOF's, θ_z and t_y. Thus, at run-time, the vision-based matcher has to determine θ_z, t_x and t_y for an S-patch, and only t_x for an SE-patch. It is interesting to observe that the original vision/touch-based 3DOR problem is essentially reduced to vision-based 2-D and 1-D recognition problems for S- and SE-patch cases, respectively.

3.2.2 Determination of Touch-Based DOF's

In this section, we outline the procedure which determines the DOF's that can be obtained by matching a tactile feature with a corresponding model one.

The partial hypothesis ${}^{i}\mathbf{T}$, which transforms the ith model feature so as to match the tactile patch, can be written as follows:

$${}^{i}\mathbf{T} = \begin{cases} {}^{i}\mathbf{T}_{s} = Trans(z,t_{iz})Rot(y,\theta_{iy})Rot(x,\theta_{ix}) & \text{for S-surface } S_i^m \\ {}^{i}\mathbf{T}_{se} = Trans(y,t_{iy})Rot(z,\theta_{iz}){}^{j}\mathbf{T}_{s} & \text{for SE-surface } (S_j^m, E_i^m)^2 \end{cases} \quad (3.2)$$

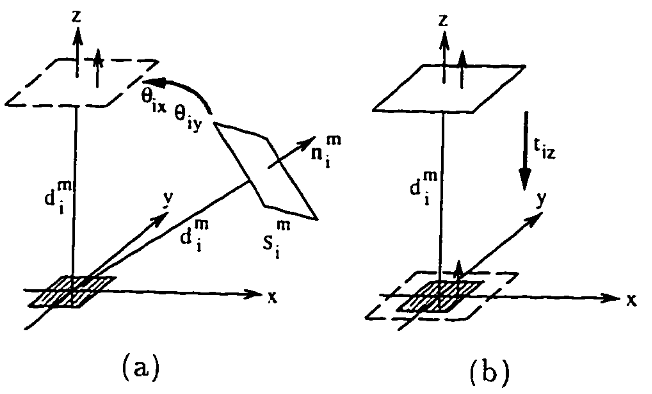


Figure 3.2: Transformation of model S-surface S_i^m to the tactile S-patch: (a) after rotation about the x and y axes. (b) after translation along the z axis.

That is, ${}^i\mathbf{T}_s$ and ${}^i\mathbf{T}_{se}$ are the transformation matrices that transform S-surface S_i^m and SE-surface (S_j^m, E_i^m) from the object frame to the tactile-feature frame, respectively.

Let us first determine the DOF's of ${}^i\mathbf{T}_s$. According to our choice of the tactile-feature frame (Fig. 3.1(a)), the equation of S_i^m after the final transformation is

$$\hat{\mathbf{z}}\mathbf{x} = 0 \tag{3.3}$$

where $\hat{\mathbf{z}}$, the direction vector along the positive direction of the z axis, is the outward normal of the surface after transformation. From (1.1) and (3.3), we observe that the outward normals of S_i^m before and after the final transformation are $\mathbf{n}_i^m$ and $\hat{\mathbf{z}}$, respectively. From (3.1), these normals are related by

$$\hat{\mathbf{z}} = Rot(z, \theta_{iz})Rot(y, \theta_{iy})Rot(x, \theta_{ix})\mathbf{n}_i^m. \tag{3.4}$$

Pre-multiplying both sides of (3.4) by $Rot^{-1}(z, \theta_{iz})$ and considering that $Rot^{-1}(z, \theta_{iz})\hat{\mathbf{z}} = \hat{\mathbf{z}}$, we get

$$\hat{\mathbf{z}} = Rot(y, \theta_{iy})Rot(x, \theta_{ix})\mathbf{n}_i^m. \tag{3.5}$$

Expanding (3.5), we get the following equation:

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c\theta_{iy} & s\theta_{iy}s\theta_{ix} & s\theta_{iy}c\theta_{ix} \\ 0 & c\theta_{ix} & -s\theta_{ix} \\ -s\theta_{iy} & c\theta_{iy}s\theta_{ix} & c\theta_{iy}c\theta_{ix} \end{pmatrix} \begin{pmatrix} n_{ix}^m \\ n_{iy}^m \\ n_{iz}^m \end{pmatrix}. \tag{3.6}$$

[2]To simplify the notation, we assume here that index i uniquely identifies SE-surface features.

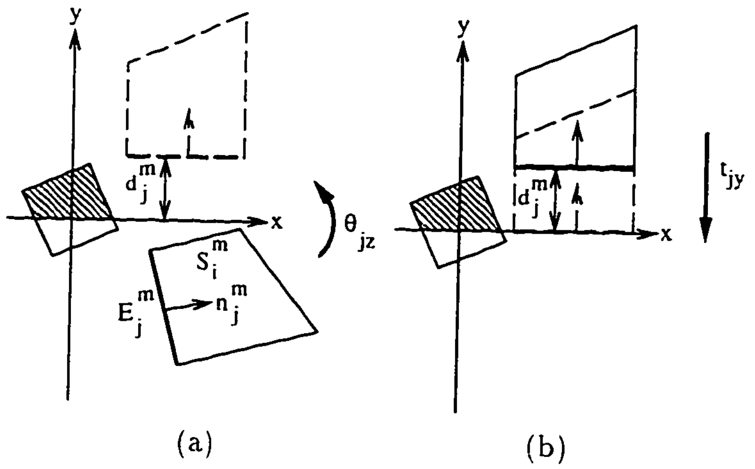


Figure 3.3: Transformation of model SE-surface (S_i^m, E_j^m) to the tactile SE-patch: (a) after rotation about the z axis, (b) after translation along the y axis.

From (3.6), we obtain θ_{ix} and θ_{iy} as follows:

$$\theta_{ix} = \arctan(\frac{n_{iy}^m}{n_{iz}^m}), \tag{3.7}$$

$$\theta_{iy} = -\arctan2(n_{ix}^m, s\theta_{ix}n_{iy}^m + c\theta_{ix}n_{iz}^m). \tag{3.8}$$

If $n_{iy}^m = n_{iz}^m = 0$, when $\mathbf{n}_i^m$ is parallel to the x axis, then there will be an infinite number of solutions for θ_{ix}. In this case, we arbitrarily set θ_{ix} to zero.

After the rotation of S_i^m by θ_{ix} and θ_{iy} about the x and y axes, respectively, the surface is now parallel to the $x-y$ plane, with the outward normal in the direction of $\hat{\mathbf{z}}$ (see Fig. 3.2(a)). Since rotation of a plane does not change its distance from the origin, a translation of $-d_i^m$ along the z axis will translate S_i^m to the $x-y$ plane (see Fig. 3.2(b))[3]. That is

$$t_{iz} = -d_i^m. \tag{3.9}$$

Next, let us determine the two edge-dependent DOF's of ${}^j\mathbf{T}_{se}$, θ_{jz} and t_{jy}. After transformation by the three parameters determined above in equations (3.7)-(3.9), edge E_j^m of surface S_i^m, chosen to match the tactile edge, lies in the $x-y$ plane. Thus, it can be represented by the following 2-D equation:

$$\mathbf{n}_j^m\mathbf{x} = d_j^m \tag{3.10}$$

[3]Notice that we have computed t_{iz} without computing θ_{iz} since the terms $Rot(z, \theta_z)$ and $Trans(z, t_z)$ can be interchanged in (3.1) without affecting the whole transformation.

where $\mathbf{n}_j^m$ is the normal of edge E_j^m that points to the interior of surface S_i^m, and d_j^m is the distance from the origin to the edge in the direction of $\mathbf{n}_j^m$ (see Fig. 3.3(a)). According to our choice of the tactile-feature frame (see Fig. 3.1(b)), the 2-D equation of E_j^m after the final transformation is

$$\hat{\mathbf{y}}\mathbf{x} = 0 \tag{3.11}$$

where $\hat{\mathbf{y}}$, the direction vector along the positive direction of the y axis, is the edge normal that points to the interior of the surface. From (3.10) and (3.11), the normals of edge E_j^m before and after rotation about the z axis are $\mathbf{n}_j^m$ and $\hat{\mathbf{y}}$, respectively. That is

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} c\theta_{jz} & -s\theta_{jz} \\ s\theta_{jz} & c\theta_{jz} \end{pmatrix} \begin{pmatrix} n_{jx}^m \\ n_{jy}^m \end{pmatrix}. \tag{3.12}$$

From (3.12), θ_{jz} is obtained as follows:

$$\theta_{jz} = \text{arctan2}(n_{jx}^m, n_{jy}^m). \tag{3.13}$$

After rotating edge E_j^m by θ_{jz} about the z axis, the edge is now parallel to the x axis at distance d_j^m from it (see Fig. 3.3(b)). Thus, a translation of $-d_j^m$ along the y axis will align edge E_j^m with the tactile edge. That is

$$t_{jy} = -d_j^m. \tag{3.14}$$

3.2.3 Determination of Vision-Based DOF's

Given a partially-transformed model junction from the junction database, ${}^iJ^m$, and a visual junction, J^s, the objective of the vision-based matcher is to compute the undetermined DOF's that transform ${}^iJ^m$ so as to match J^s. These DOF's are θ_z, t_x and t_y for the S-patch case, and t_x for the SE-patch one. Define the following:

1. Functions:

$\text{dir}(\mathbf{u}, \mathbf{v})$	$=$	direction vector from $\mathbf{u}$ to $\mathbf{v}$
$\text{angle}(\mathbf{u}, \mathbf{v})$	$=$	angle between $\mathbf{u}$ and $\mathbf{v}$

2. Camera:

$\mathbf{c}$	$=$	camera viewpoint
f	$=$	camera focal length

3. Line of Sight:

$L(\mathbf{v})$	=	3-D line from $\mathbf{c}$ through image point $\mathbf{v}$
$L(\mathbf{v}, \alpha)$	=	point on $L(\mathbf{v})$ at a distance α from $\mathbf{c}$
$L(v_\omega, \alpha)$	=	ω component of $L(\mathbf{v}, \alpha)$ ($\omega = x$, y or z)

4. Partially-Transformed Model Junction ${}^iJ^m$:

${}^i\mathbf{v}^m$	=	junction vertex
${}^i\mathbf{e}_k^m$	=	direction of the kth junction edge

5. Visual Junction J^s:

$\mathbf{v}^s$	=	junction vertex
$\mathbf{u}_k^s$	=	end vertex of the kth junction edge
$\mathbf{e}_k^s$	=	direction of the kth junction edge ($\mathbf{e}_k^s = \mathrm{dir}(\mathbf{v}^s, \mathbf{u}_k^s)$)
$\mid J^s \mid$	=	number of junction edges
$\mathbf{w}$	=	direction of line of sight $L(\mathbf{v}^s)$ ($\mathbf{w} = \mathrm{dir}(\mathbf{c}, \mathbf{v}^s)$)
P_k	=	half plane formed by $\mathbf{w}$ and $\mathbf{e}_k^s$
$\mathbf{m}_k$	=	normal of half plane P_k ($\mathbf{m}_k = \mathbf{w} \times \mathbf{e}_k^s$)

The equation of half plane P_k is

$$\mathbf{m}_k \mathbf{x} = 0. \qquad (3.15)$$

Line of sight $L(\mathbf{v}^s)$ can be represented as

$$\mathbf{x} = \mathbf{c} + \alpha \mathbf{w}. \qquad (3.16)$$

where α is the distance from $\mathbf{c}$ to $\mathbf{x}$ in the direction of $\mathbf{w}$.

Model junction ${}^iJ^m$ is said to match visual junction J^s, if it can be transformed by the undetermined DOF's, to form ${}^fJ^m$, so that the following necessary conditions are satisfied (refer to Fig. 3.4):

1. The direction of each model edge, ${}^f\mathbf{e}_k^m$, lies inside half plane P_k[4].

2. The model vertex, ${}^f\mathbf{v}^m$, lies on line of sight $L(\mathbf{v}^s)$, and the distance from the camera viewpoint $\mathbf{c}$ to ${}^f\mathbf{v}^m$ is greater than the camera focal length f.

[4]For ${}^f\mathbf{e}_k^m$ to lie inside P_k, $\mathbf{m}_k \cdot {}^f\mathbf{e}_k^m$ should be approximately 0, and $\mathbf{w} \times {}^f\mathbf{e}_k^m$ should be in the same direction of $\mathbf{m}_k$.

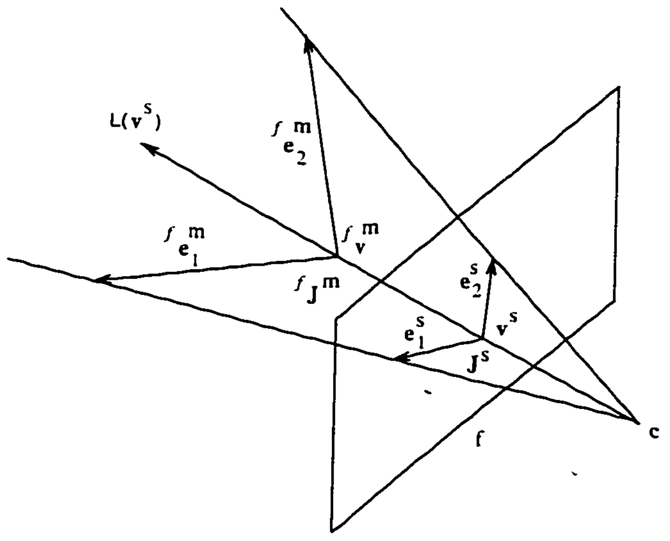


Figure 3.4: A visual junction and a matching model one.

The matching steps for S- and SE-patches are presented in the next two subsections.

A. The S-Patch Case

In this case, the only unknown rotation angle is θ_z. Thus, ${}^f\mathbf{e}_k^m$, the direction of model edge ${}^i\mathbf{e}_k^m$ after the final transformation, is confined to a cone that embeds ${}^i\mathbf{e}_k^m$, and whose axis coincides with the z axis. This cone, C_k, is parametrized by θ_z as follows:

$$\mathbf{x} = \begin{pmatrix} c\theta_z & -s\theta_z & 0 \\ s\theta_z & c\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} {}^i\mathbf{e}_k^m. \tag{3.17}$$

Also, ${}^f\mathbf{e}_k^m$ is confined to lie inside half plane P_k. Thus, ${}^f\mathbf{e}_k^m$ is determined by the intersection of C_k and P_k, as shown in Fig. 3.5(a). Solving (3.15) and (3.17), we obtain an equation of the form:

$$a \sin(\theta_z) + b\cos(\theta_z) = c \tag{3.18}$$

where

$$\begin{aligned} a &= -m_{kx}{}^i e_{ky}^m + m_{ky}{}^i e_{kx}^m \\ b &= m_{kx}{}^i e_{kx}^m + m_{ky}{}^i e_{ky}^m \\ c &= -m_{kz}{}^i e_{kz}^m. \end{aligned}$$

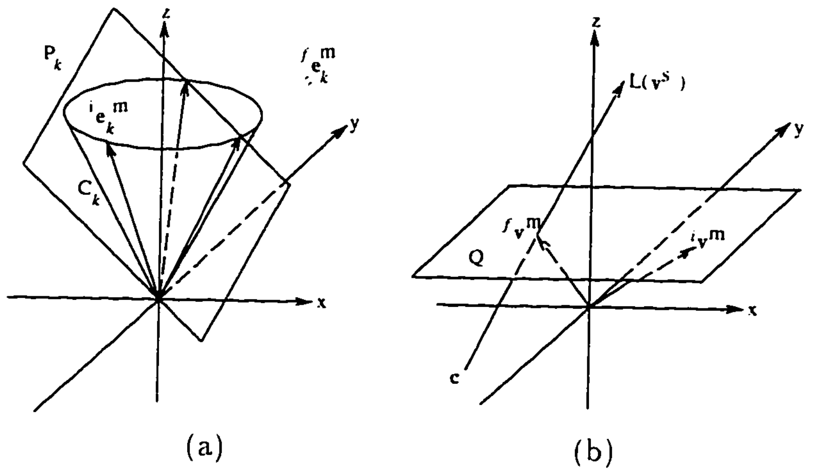


Figure 3.5: Matching in the S-patch case: (a) computing rotation, (b) computing translation.

Solving (3.18), we obtain the following expressions for θ_z:

$$\theta_z = \begin{cases} \arcsin\left(\frac{c}{\sqrt{a^2+b^2}}\right) - \operatorname{arctan2}(b,a) \\ \pi - \arcsin\left(\frac{c}{\sqrt{a^2+b^2}}\right) - \operatorname{arctan2}(b,a) \end{cases} . \tag{3.19}$$

Notice that there is a solution, only if $\mid \frac{c}{\sqrt{a^2+b^2}} \mid \leq 1$. Also, notice that there is a singularity when $a = b = 0$. This occurs when either $\mathbf{m}_k$ or $^i\mathbf{e}^m_k$ is parallel to the z axis. To avoid this case, we select the scene junction edge with the largest $(a^2 + b^2)$ for computing θ_z.

Each generated θ_z is then verified by utilizing the remaining scene edges as follows. The direction of each remaining edge, $^i\mathbf{e}^m_l$, is rotated by θ_z about the the z axis to form $^f\mathbf{e}^m_l$. The first matching condition, discussed above, is then applied to determine whether to accept θ_z.

For an accepted θ_z, matching continues by computing the translation parameters t_x and t_y as follows: The position of model vertex $^i\mathbf{v}^m$ after the final transformation, $^f\mathbf{v}^m$, is confined to a plane, Q, that passes through $^i\mathbf{v}^m$ and is parallel to the $x-y$ plane (see Fig. 3.5(b)). Plane Q is represented by the following equation:

$$\hat{\mathbf{z}}\mathbf{x} = {}^i v^m_z . \tag{3.20}$$

It is easy to see that $^f\mathbf{v}^m$ is determined by the intersection of plane Q with line of sight $L(\mathbf{v}^s)$ (Fig. 3.5(b)). Solving (3.16) and (3.20), we obtain α_L, the distance from $\mathbf{c}$ to $^f\mathbf{v}^m$:

$$\alpha_L = \left(\frac{{}^i v^m_z - c_z}{w_z}\right) . \tag{3.21}$$

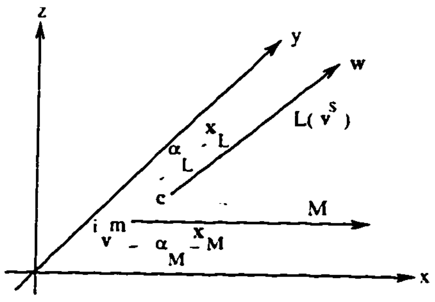


Figure 3.6: Matching in the SE-patch case.

Substituting (3.21) into (3.16), we get ${}^f\mathbf{v}^m$:

$$ {}^f\mathbf{v}^m = \mathbf{c} + \alpha_L \mathbf{w}. \tag{3.22}$$

Notice that ${}^f\mathbf{v}^m$ has to satisfy the second matching condition, for matching to continue. The relationship between ${}^i\mathbf{v}^m$ and ${}^f\mathbf{v}^m$ is

$$ {}^f\mathbf{v}^m = \begin{pmatrix} c\theta_z & -s\theta_z & 0 \\ s\theta_z & c\theta_z & 0 \\ 0 & 0 & 1 \end{pmatrix} {}^i\mathbf{v}^m + \begin{pmatrix} t_x \\ t_y \\ 0 \end{pmatrix}. \tag{3.23}$$

From (3.23), we obtain t_x and t_y as follows:

$$\begin{aligned} t_x &= {}^f v_x^m - (c\theta_z {}^i v_x^m - s\theta_z {}^i v_y^m) & (3.24) \\ t_y &= {}^f v_y^m - (s\theta_z {}^i v_x^m + c\theta_z {}^i v_y^m). & (3.25) \end{aligned}$$

Notice that there is a singularity if $w_z = 0$ (see (3.21)). This occurs when $L(\mathbf{v}^s)$ is parallel to the $x-y$ plane. In this case, matching fails, and another visual junction should be examined.

From (3.1), (3.2), (3.19), (3.24) and (3.25), we obtain ${}^T\mathbf{T}_O$, a complete pose hypothesis of the model object with respect to the tactile-feature frame:

$$ {}^T\mathbf{T}_O = Trans(x, t_x)Trans(y, t_y)Rot(z, \theta_z){}^i\mathbf{T}_s. $$

B. The SE-Patch Case

In this case, the only DOF that needs to be determined on-line is the translation along the x axis, t_x. Since direction vectors do not change with translation, we have ${}^f\mathbf{e}_k^m = {}^i\mathbf{e}_k^m$, for all k. If any ${}^f\mathbf{e}_k^m$ is not accepted by the first matching condition, then matching fails. Otherwise, it proceeds by examining model vertex ${}^i\mathbf{v}^m$. It is easy to see that the position of ${}^i\mathbf{v}^m$ after the final transformation, ${}^f\mathbf{v}^m$, is confined to a line, M, that passes through ${}^i\mathbf{v}^m$ and is parallel to the x axis (see Fig. 3.6). Line M is represented as

$$\mathbf{x} = {}^i\mathbf{v}^m + \alpha\hat{\mathbf{x}} \tag{3.26}$$

where $\hat{\mathbf{x}}$ is the direction vector along the positive direction of the x axis. Thus, ${}^f\mathbf{v}^m$ is determined by the intersection of lines M and $L(\mathbf{v}^s)$, if such intersection exists. To test for intersection in the presence of uncertainty, we compute the distance between lines M and $L(\mathbf{v}^s)$, i.e., length of the common normal. If this distance is less than some threshold, then intersection is assumed to exist; otherwise matching fails. If there is an intersection, then ${}^f\mathbf{v}^m$ is approximated as follows. Let the common normal of $L(\mathbf{v}^s)$ and M intersect them at $\mathbf{x}_L$ and $\mathbf{x}_M$, respectively, and let α_L and α_M be their corresponding α's (see Fig. 3.6). The position of ${}^f\mathbf{v}^m$ can be approximated by substituting α_M into (3.26). That is,

$${}^f\mathbf{v}^m = {}^i\mathbf{v}^m + \alpha_M\hat{\mathbf{x}}. \tag{3.27}$$

The relationship between ${}^i\mathbf{v}^m$ and ${}^f\mathbf{v}^m$ is represented as

$${}^f\mathbf{v}^m = {}^i\mathbf{v}^m + t_x\hat{\mathbf{x}}. \tag{3.28}$$

From (3.27) and (3.28), we directly obtain t_x:

$$t_x = \alpha_M. \tag{3.29}$$

Note that ${}^f\mathbf{v}^m$ has to be accepted by the second matching condition for matching to succeed. It should also be noted that there is a singularity if line $L(\mathbf{v}^s)$ is parallel to the x axis, since the pair $(\mathbf{x}_L, \mathbf{x}_M)$ will not be unique. In this case, another visual junction should be examined.

From (3.1), (3.2) and (3.29), we obtain ${}^T\mathbf{T}_O$, a complete pose hypothesis of the model object with respect to the tactile-feature frame:

$${}^T\mathbf{T}_O = Trans(x, t_x)\,{}^i\mathbf{T}_{se}.$$

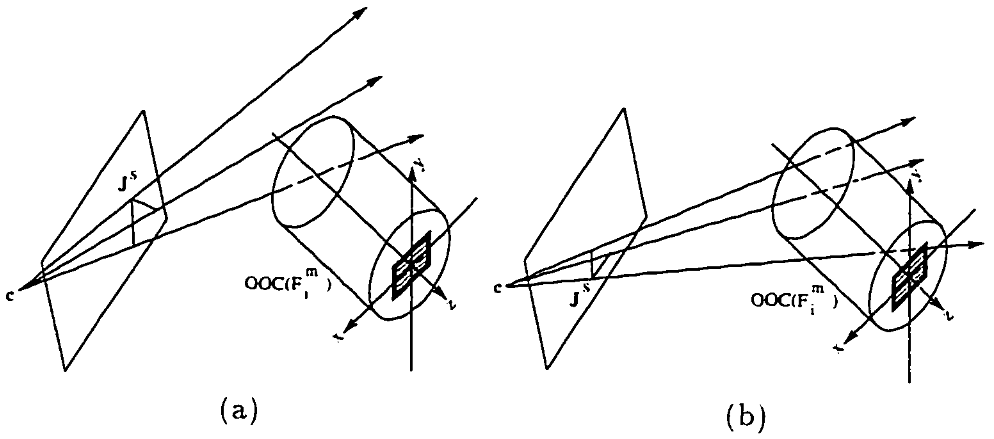


Figure 3.7: Application of object-occupancy constraint $OOC(F_i^m)$: (a) $OOC(F_i^m)$ is inconsistent with junction J^s, (b) $OOC(F_i^m)$ is consistent with junction J^s.

3.3 Filtering Techniques

In this section, we describe a couple of filtering techniques for reducing the number of model junctions that are actually matched with a scene one. Those techniques are especially important since the size of the junction database can be large. For a polyhedron with S surfaces, and E_S edges per surface, the total number of junctions in the junction database is of order $O(S^2 E_S)$ and $O(S^2 E_S^2)$ for S- and SE-patch cases, respectively. Section 3.3.1 presents a filtering technique which uses the object-occupancy constraint. In Section 3.3.2, we present the other technique, which derives bounds on transformation-invariant model attributes by utilizing the edge-angle constraint.

3.3.1 Object-Occupancy Constraint

For each match between a tactile feature, F^s, and a corresponding model one, F_i^m, we construct volume $OOC(F_i^m)$ off-line. This volume provides an approximate bound on the space occupied by $\mathcal{O}$ given that F_i^m corresponds to F^s (refer to Section 2.3.3). The set of all object-occupancy constraints $\mathcal{V} = \{OOC(F_i^m)\}$ is stored in a *volume database*. At run-time, we compute the intersection between the lines of sight of junction J^s, $L(\mathrm{v}^s)$ and $L(\mathrm{u}_k^s)$, and each volume $OOC(F_i^m)$. If *any* of those lines of sight does not intersect $OOC(F_i^m)$ (e.g., see Fig. 3.7(a)), then model feature F_i^m is inconsistent with the scene feature set. Accordingly, all model junctions in the set ${}^i\mathcal{J}^m$ are excluded from matching. On the other hand, if *all* lines of sight

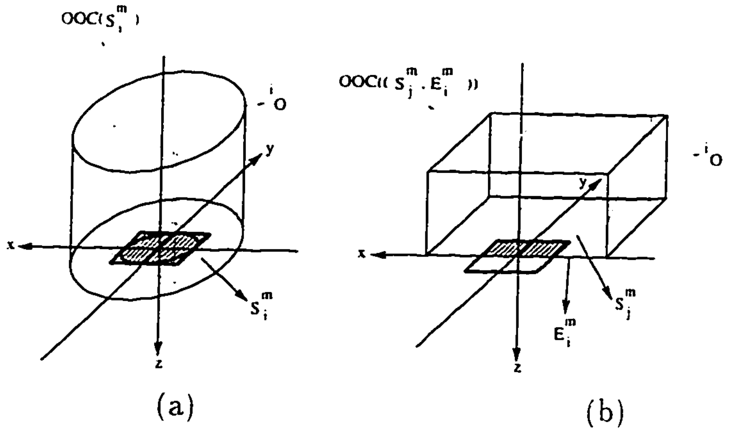


Figure 3.8: Object-occupancy constraints: (a) S-patch case, (b) SE-patch case.

intersect $OOC(F_i^m)$ (e.g., see Fig. 3.7(b)), then model junctions in ${}^i\mathcal{J}^m$ are considered for matching. This filtering technique is very useful in eliminating erroneous scene junctions that are distant from the tactile sensor.

Let $\mathcal{O}$ be a model object, and let ${}^i\mathcal{O}$ be the object after transformation by ${}^i\mathbf{T}$ (refer to (3.2)). In the S-patch case, the partially-transformed object ${}^i\mathcal{O}$ is constrained to move freely in the x–y plane such that S-surface S_i^m, assumed to be in contact with the tactile sensor, totally covers the sensor. Since θ_z is unknown off-line, the uncertainty about the volume occupied by ${}^i\mathcal{O}$ can be approximated by a cylinder along the z axis, $OOC(S_i^m)$, as shown in Fig. 3.8(a). Dimensions of such a cylinder are derived in Appendix C.1. In the SE-patch case, the partially-transformed object ${}^i\mathcal{O}$ is constrained to move along the x axis such that model edge E_i^m, of model surface S_j^m, totally covers the tactile edge. Hence, the uncertainty about the volume occupied by ${}^i\mathcal{O}$ can be approximated by a cuboid, $OOC((S_j^m, E_i^m))$, that is parallel to the principal axes (see Fig. 3.8(b)). Derivation of the dimensions of cuboid $OOC((S_j^m, E_i^m))$ is presented in Appendix C.2.

3.3.2 Constraints on Model Attributes

In this technique, transformation-invariant model attributes are utilized in order to reduce the number of scene/model matches. A model feature (a partially-transformed model junction in the junction database) is considered for matching with a given scene one (a visual junction), only if its attribute values lie within the respective scene bounds. Thus, an incompatible model junction can be eliminated from matching

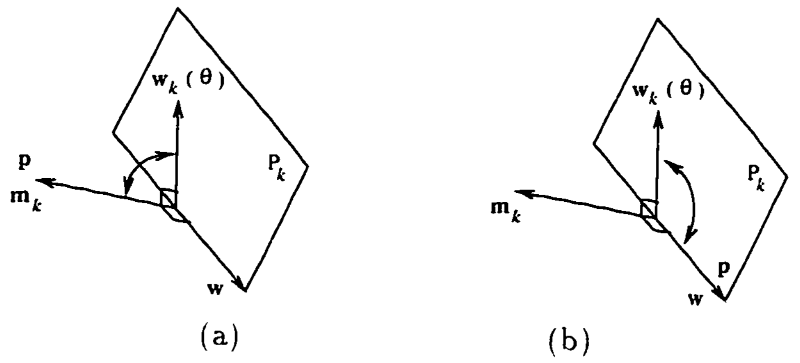


Figure 3.9: Extreme bounds on angle($\mathbf{p}, \mathbf{w}_k(\theta)$): (a) [90°, 90°], (b) $EAC(F_i^m, E_k^s)$.

using very few comparison operations[5]. Notice that the tactile polygon is implicitly included in the process by choosing the tactile-feature frame as the reference one.

The invariant attributes of a model junction can be determined as follows. For an S-patch, since only θ_z, t_x and t_y need to be determined on-line, it is easy to show that angle($\hat{\mathbf{z}}, {}^i\mathbf{e}_k^m$) and ${}^iv_z^m$ are invariant attributes. Similarly, for an SE-patch, since the only DOF that needs to be determined on-line is t_x, it can be easily shown that angle($\hat{\mathbf{x}}, {}^i\mathbf{e}_k^m$), angle($\hat{\mathbf{y}}, {}^i\mathbf{e}_k^m$), angle($\hat{\mathbf{z}}, {}^i\mathbf{e}_k^m$), ${}^iv_y^m$ and ${}^iv_z^m$ are invariant attributes. Thus, for a scene junction J^s, the number of corresponding attribute bounds is $(1 + \mid J_s \mid)$ and $(2 + 3 \mid J_s \mid)$ for S- and SE-patch cases, respectively.

Now, let us determine scene bounds on the above model attributes. For angular attributes, we first compute the edge-angle constraint $EAC(F_i^m, E_k^s)$, for each model feature F_i^m and junction edge E_k^s (refer to Section 2.3.3). Let us assume that model features F_i^m and ${}^iE_k^m$ match the corresponding scene ones. In such a case, we can represent the direction of ${}^fE_k^m$ by vector $\mathbf{w}_k(\theta)$, defined in (B.1), where $\theta \in EAC(F_i^m, E_k^s)$. Bounds on angle($\mathbf{p}, \mathbf{w}_k(\theta)$), where $\mathbf{p}$ is substituted by $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ or $\hat{\mathbf{z}}$, are obtained by determining the minimum and maximum values of angle($\mathbf{p}, \mathbf{w}_k(\theta)$), subject to the constraint $\theta \in EAC(F_i^m, E_k^s)$. Details of such a process are presented in Appendix D.

It is interesting to note that the tightness, and accordingly the robustness, of the bounds on angle($\mathbf{p}, \mathbf{w}_k(\theta)$) vary considerably depending on the direction of $\mathbf{p}$ relative to $\mathbf{m}_k$ and $\mathbf{w}$. For example, those bounds are [90°, 90°] if $\mathbf{p}$ is aligned with $\mathbf{m}_k$ (Fig.

[5]It is also possible to index the invariant model attributes, thereby significantly improving speed by enabling direct access to compatible model junctions.

3.9(a)), but $EAC(F_i^m, E_k^s)$, if **p** is aligned with **w** (Fig. 3.9(b)).

Bounds on the distance attributes can be determined as follows. From (3.16), it is easy to obtain bounds on attribute ${}^i v_\omega^m$ (ω is substituted by either y or z):

$$\text{minmax}(L(v_\omega^s, \alpha^{min}), L(v_\omega^s, \alpha^{max}))$$

where $\text{minmax}(\{x\}) = [\min(\{x\}), \max(\{x\})]$, and $VDC(S^m, V^s) = [\alpha^{min}, \alpha^{max}]$.

3.4 Experimental Results

In this section, we present a number of simulation and real experiments to demonstrate the performance of the proposed vision/touch-based pose-generation technique.

3.4.1 Uncertainty Handling

Thresholds which are used throughout the system are automatically determined based on the following error model:

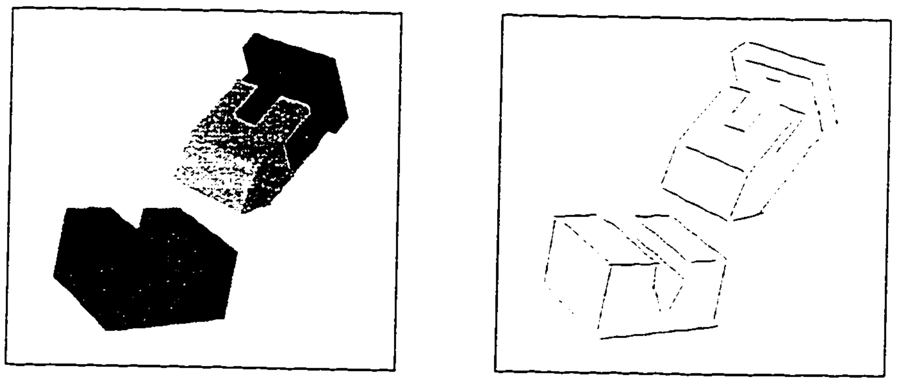
1. *Tactile-Sensor Frame:* The actual origin of the frame is assumed to be within some distance, Δr^f, of the estimated one. The actual principal axes are assumed to be obtained from the estimated ones by rotating them about an arbitrary axis, where the rotation angle is between 0 and some threshold, $\Delta\theta^f$.

2. *Tactile Vertex:* The actual tactile vertex, of a tactile edge, is assumed to be within some distance, Δr^t, of the estimated one.

3. *Visual Vertex:* The actual visual vertex is assumed to be within some pixel distance, Δr^v, of the estimated one.

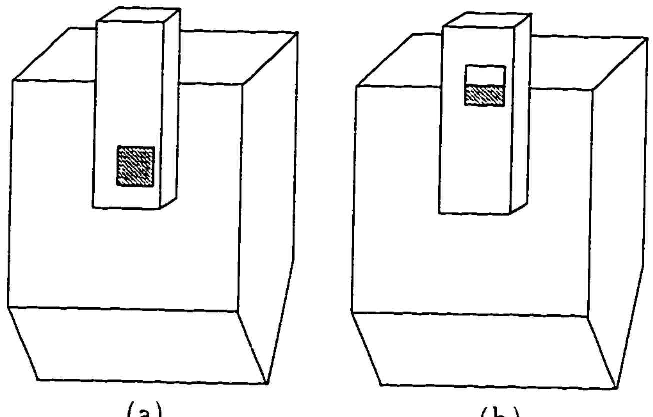
3.4.2 Simulation Experiments

Fig. 3.10(a) shows a scene of a three-fingered robot hand inserting a peg to a slot. Fig. 3.10(b) shows results of the feature extraction process. Nine three-edge and 21 two-edge junctions are extracted from such a scene. Since it is hard to determine whether a given junction belongs to the model object, we use all of those junctions for matching. A tactile sensor of dimensions 10 mm $\times$ 10 mm is assumed to exist on the middle finger of the robot hand. Fig. 3.11 shows the tactile images obtained by



(a) (b)

Figure 3.10: A synthetic scene of a peg being inserted to a slot: (a) scene, (b) extracted edges.



(a) (b)

Figure 3.11: Positions of the tactile sensor: (a) surface contact, (b) surface-edge contact.

Feature Set	Scene Features	Model Features			Generated Hypos.
		Total	Filtered 1	Filtered 2	
J-feature	25	60	N/A	N/A	40.3
S-patch + junction	30	420	280.2	124.1	17.4
SE-patch + junction	30	1800	780.0	10.6	2.2

Table 3.1: Peg-to-a-slot scene: Generated hypotheses.

assuming that the tactile sensor is mounted at two different positions on the middle finger.

To perform a quantitative comparison between vision/touch-based and vision-based localization, we use a technique similar to that presented by Kanatani [49] and Wong and Kittler [74] to visually generate pose hypotheses. In their techniques, the visual feature used for matching is a three-edge junction, where one of the junction edges is fully visible. This feature will be referred to as a *J-feature*. A junction edge is considered fully visible if the other end is also a junction. Thus, for n three-edge scene junctions, the total number of J-features can be up to $3n$.

For each type of scene feature sets (J-feature, S-patch (Fig. 3.11(a)) + junction, SE-patch (Fig. 3.11(b)) + junction), localization is performed 100 times. In each run, the sensory data are randomly perturbed, according to the bounded-error model outlined above, assuming $\Delta\theta^f = 5°$, $\Delta r^f = 5$ mm, $\Delta r^t = 0.75$ mm, and $\Delta r^v =$ 1 pixel. The results obtained in all runs, for each type of scene sets, are averaged and summarized in Table 3.1. Columns in this table are:

1. *Feature Set:* the scene feature set used to generate hypotheses
2. *Scene Features:* number of scene feature sets
3. *Model Features (Total):* total number of model feature sets that correspond to the scene set
4. *Model Features (Filtered 1):* average number of model sets, per scene set, that are accepted by the object-occupancy constraints
5. *Model Features (Filtered 2):* average number of model sets, per scene set, that are accepted by both the object-occupancy constraints and the constraints on

model transformation-invariant attributes

6. *Generated Hypotheses:* average number of generated hypotheses, per scene set

From Table 3.1, we can observe the following:

1. The two filtering techniques, individually, succeed in eliminating a large number of model junctions. On the average, 33% and 57% of the model junctions are eliminated by the object-occupancy constraints for S- and SE-patches, respectively. On the other hand, the constraints on invariant model attributes succeed in eliminating many of the model junctions that have passed the object-occupancy-constraint test (on the average, 56% and 99% of the model junctions are eliminated for S- and SE-patch cases, respectively). Clearly, the filtering techniques are more powerful in the SE-patch case, due to the presence of extra 3-D information provided by the tactile edge.

2. Combined together, the two filtering techniques are effective in reducing the number of model junctions that are passed to the matcher. On the average, only 30% and 0.6% of the model junctions are passed to the matcher for S- and SE-patches, respectively. This makes the number of model sets that are actually used in matching comparable to the case when only visual features are used (124.1 and 10.6 model sets for S- and SE-patches, respectively, compared to 60 model sets in the vision-only case). This is despite the fact that the total number of model sets is much higher when a tactile feature is included in the scene set (420 and 1800 for S- and SE-patches, respectively, but only 60 for J-features).

3. Not all matches between model and scene junctions are successful in generating pose hypotheses. This is because the constraints utilized by the filtering techniques are rather "loose"; i.e., they do not guarantee that a model junction that is accepted by them will be consistent with the scene junction. An interesting research problem would be to develop tighter constraints that can still be computed efficiently, as the ones developed here.

4. Incorporating a tactile feature in the scene feature set results in a considerably smaller number of generated hypotheses, per scene set, than when only visual

Feature Set	Error		
	Rotational	Translational	Depth
J-feature	2.3°	34.3 mm	33.1 mm
S-patch + junction	1.3°	4.5 mm	3.0 mm
SE-patch + junction	1.9°	6.6 mm	4.1 mm

Table 3.2: Peg-to-a-slot scene: Errors in estimating the object pose.

features are used. The extent of reduction is 57% and 95% for S- and SE-patches, respectively. This is because the tactile feature provides more explicit information about the object pose. This enables the development of filtering techniques that limit model sets which are passed to the matcher. Furthermore, the presence of 3-D data in the scene feature set reduces the number of successful scene/model matches. Notice that 67% of the matches are successful when only visual features, J-features, are used, compared to 14% and 21% success rate for S- and SE-patches, respectively[6].

Next, we compare the accuracy of the poses generated visually with those generated by integrating visual and tactile data. Table 3.2 shows such a comparison, which is also obtained by running the vision- and vision/touch-based localization systems 100 times under different perturbation values and averaging. Columns in this table are:

1. *Rotational Error:* average angle between the quaternion representing actual rotation and those of the estimated ones

2. *Translational Error:* average distance between the origin of actual object frame and those of the estimated ones

3. *Depth Error:* average absolute error in estimating the object depth with respect to the visual sensor (translation along the optical axis)

From Table 3.2, we observe that the error in rotation is quite similar in all cases. We also observe that the rotational error in the SE-patch case is slightly larger than that in the S-patch one. This is because the amount of uncertainty in the tactile

[6]Notice that the success rate in the vision/touch case would be even lower if no filtering is applied.

edge is relatively high compared to that of the visual edges. Recall that one of the rotational DOF's (θ_z, see Section 3.2) is determined by matching visual and tactile edges in the S- and SE-patch cases, respectively. This observation is quite realistic since the resolution of tactile sensors is usually much lower than that of visual ones. For example, in our lab, the resolution of the tactile sensor is two orders of magnitude lower than that of the visual one. If the tactile edge is too noisy, then it should be ignored: i.e., the SE-patch should be considered as an S-patch. However, the price is paid in the generation of a larger number of hypotheses that need to be verified.

For translational error, we observe that it is an order of magnitude worse when only visual features are used. Observing the Depth-Error column in Table 3.2, we can see that, in the vision-only case, most of the translational error is due to poor object-depth estimation. This observation can be explained as follows. Visually, the object depth is determined by matching a fully-visible scene edge with a model one [49, 74]. It can be shown that the accuracy of such a process is proportional to the actual object depth, along with the relative error in determining the edge length. Thus, we can expect poorer depth estimation for short fully-visible edges, and for objects that are not close to the image plane (in this experiment, the depth of the peg is 925.0 mm). On the other hand, in the vision/touch case, depth is estimated by determining the intersection of the tactile plane (or the tactile edge) with a line of sight corresponding to a junction vertex. The error in estimating the location of the intersection point depends on the following factors:

1. Rotational and translational uncertainties of the tactile-feature frame

2. Distance between the tactile-feature frame and the line of sight (notice that the error component due to the rotational uncertainty of the tactile-feature frame is proportional to the above distance)

3. Uncertainty of the line of sight

The accuracy of the intersection point, and accordingly that of depth estimation, is weakly dependent on the object depth because of the following reasons:

1. The uncertainties of the tactile-feature frame are independent of the object depth.

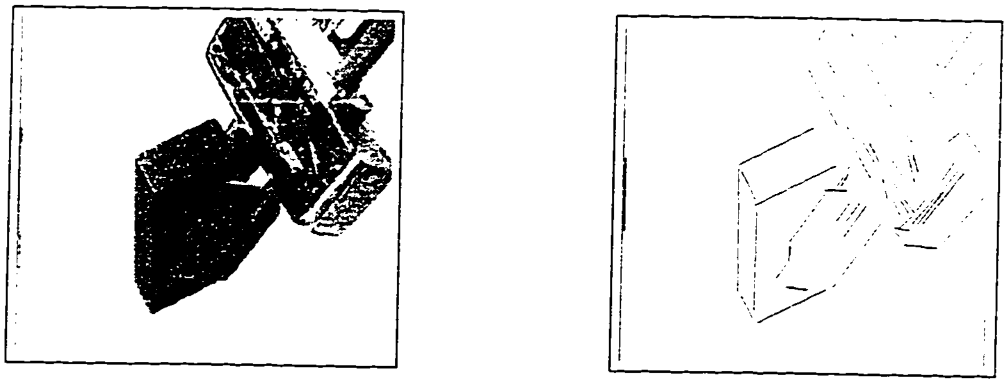
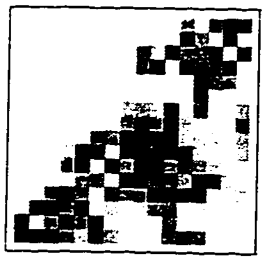

(a) (b)

Figure 3.12: First real scene: (a) scene, (b) extracted edges.

2. The distance between the tactile-feature frame and the line of sight is bounded by the object size.

3. The uncertainty of the line of sight is relatively very small, which makes its error contribution insignificant. To illustrate this observation, let us represent this uncertainty by a cone whose axis is the line of sight and angle is determined by the uncertainty of the corresponding vertex, Δr^v (see Section 3.4.1). Experimentally, we have found that such an angle is very small, which makes the divergence rate of the cone very slow. For example, assuming $\Delta r^v = 1$ pixel, we have found that the cone radius at the peg depth (925.0 mm) is about 0.7 mm[7].

It should be noted that we are not trying to draw general conclusions about the accuracy of pose estimates generated visually compared to those generated by integrating visual and tactile data. Obviously, this depends on many factors including amount of uncertainty in both visual and tactile data, geometric relationships between features, and object depth. Our objective is simply to highlight some common situations (in particular, when the object is not close to the image plane) in which vision/touch-based depth estimation can be much more accurate than vision-based estimation.



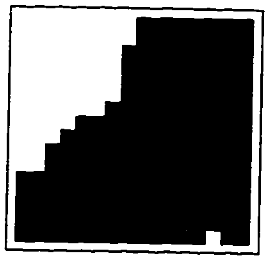


Figure 3.13: First real scene: tactile image: (a) original. (b) thresholded.

Feature Set	Scene Features	Model Features			Generated Hypos.
		Total	Filtered 1	Filtered 2	
J-feature	7	48	N/A	N/A	48
S-patch + junction	17	288	271.1	73.4	24
SE-patch + junction	17	1152	768.0	1.65	1.65

Table 3.3: First real scene: Generated hypotheses.

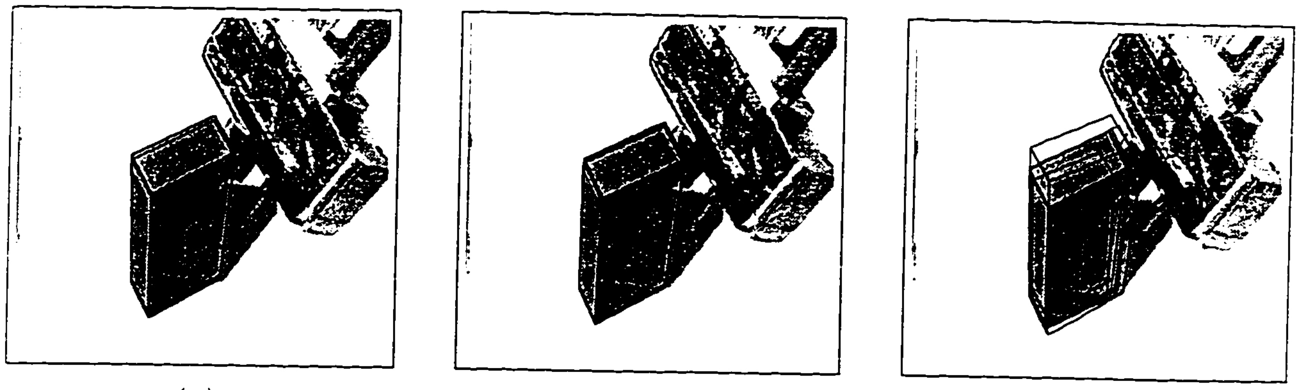


Figure 3.14: Valid hypotheses for the: (a) S-patch case (four hypotheses). (b) SE-patch case (four hypotheses). (c) vision-only case (six hypotheses).

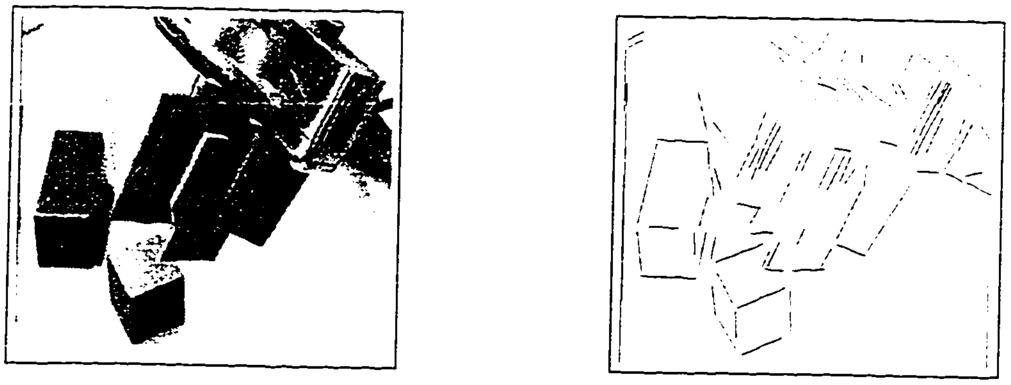

(a) (b)

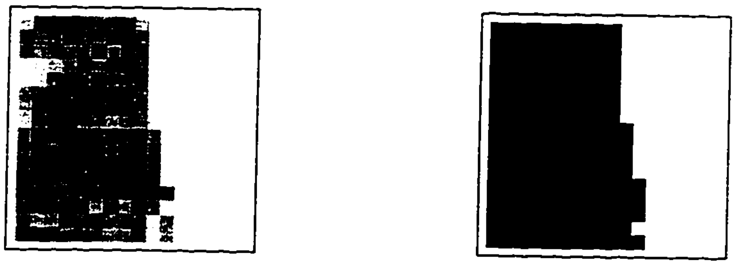
Figure 3.15: Second real scene: (a) scene, (b) extracted edges.

3.4.3 Real Experiments

The proposed technique has also been tested using real data. The experimental setting is that described in Section 2.4.3. Fig. 3.12(a) shows a scene of a parallel-jaw gripper grasping a rectangular polyhedron. Results of the feature extraction process are shown in Fig. 3.12(b). Three three-edge junctions and 14 two-edge ones are extracted from such a scene. Fig. 3.13(a) shows the original tactile image. The darkness of each tactile element (tactel) in this image is proportional to the pressure on it. The tactile image is filtered and thresholded to form the binary image shown in Fig. 3.13(b). This binary image is analyzed to detect an SE-patch. To test the system performance in the S-patch case, we also consider the center of the sensor as an S-patch of negligible dimensions (notice that the sensor center lies inside the tactile polygon). Pose-generation results are shown in Table 3.3. From this table, similar conclusions can be drawn as in the simulation experiments. Fig. 3.14 shows wireframes of the valid generated hypotheses (i.e., those generated from valid scene sets), for each of the three scene feature sets. Notice the variance in the accuracy of the visually-generated poses, compared to those that are generated by integrating vision and touch. As explained before, this variance is mainly due to poor depth estimation.

Our second real scene, Figs. 3.15, is quite similar to the first one, except that the object is heavily occluded. In such a scene, there is insufficient visual information

[7] The assumed camera model is that of an NEC CCD camera. This camera is the one used in the real experiments, which are presented in the next section.

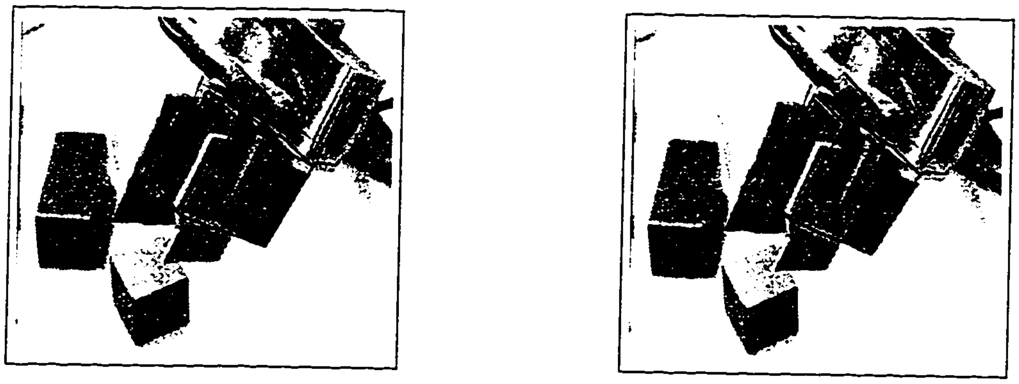


(a) (b)

Figure 3.16: Second real scene: tactile image: (a) original, (b) thresholded.

Feature Set	Scene Features	Model Features			Generated Hypos.
		Total	Filtered 1	Filtered 2	
J-feature	11	48	N/A	N/A	30.5
S-patch + junction	34	288	251.2	63.5	23.5
SE-patch + junction	34	1152	700.2	8.8	4.2

Table 3.4: Second real scene: Generated hypotheses (all visually-generated hypotheses are erroneous).



(a) (b)

Figure 3.17: Valid hypothesis for: (a) S-patch case, (b) SE-patch case.

to determine the object pose (only one valid two-edge junction is extracted). Fig. 3.16 shows the obtained tactile data, from which an SE-patch is extracted. As in the previous scene, the sensor center is selected as an S-patch of negligible dimensions to test the system performance in the S-patch case. Results are summarized in Table 3.4. The obtained results further enforces the conclusions drawn earlier. Figs. 3.17(a) and 3.17(b) show the valid generated hypothesis for S- and SE-patch cases, respectively. This experiment demonstrates the capability of the proposed vision/touch-based technique to determine object pose under heavy occlusion.

3.5 Conclusions

A novel technique has been presented for generating hypotheses by integrating visual and tactile data. A hybrid set of visual and tactile features (a visual junction and a tactile polygon) is used to generate pose hypotheses. The matching process first determines a subset of the object's DOF's using the tactile feature. The remaining DOF's, which cannot be determined from the tactile feature, are then obtained from the visual one. A couple of filtering techniques are developed to reduce the number of model feature sets that are actually compared with a given scene set. Superiority of the proposed technique over vision-based pose generation has been demonstrated in its capability of determining the object pose under heavy occlusion, its generation of a smaller number of pose hypotheses, and its accuracy in estimating object depth.

Although only two cases of tactile contact are considered here, other cases can be handled in a similar fashion. For example, assume that we have an edge contact. Four DOF's can be determined form the tactile edge (rotation about, and translation along the edge can not be determined). The remaining two DOF's can then be obtained by matching a visual feature such as a junction, for example.

The proposed approach can be extended to handle non-polyhedral objects as well. For example, assume that we are given a cylindrical model object. A tactile edge generated through contact between the cylinder and the tactile sensor constrains all of the cylinder's DOF's except for translation along the edge. This parameter can then be determined from the visual data by matching the cylinder's circular ends with corresponding visual features.

Chapter 4

Touch-Based Hypothesis Verification

4.1 Introduction

A hypothesis is verified by performing exhaustive comparison between scene and model features. This can be done in either of two fashions:

- *Model-Driven:* For each model feature, we look for a compatible scene one.
- *Data-Driven:* For each scene feature, we look for a compatible model one.

A naive implementation of either model- or data-driven verification would have time complexity of order $O(|\mathcal{F}^s||\mathcal{F}^m|)$, where $\mathcal{F}^s$ and $\mathcal{F}^m$ are sets of scene and model features, respectively. The choice of either model- or data-driven verification seems to be arbitrary. However, there are situations in which data-driven verification is more efficient:

- A subset of sensory data is known to belong to the object of interest (e.g., the tactile data in the task of recognizing an object in a hand). In such a case, hypotheses which are inconsistent with *any* piece of the sensory data can be eliminated immediately, without any further scene/model comparisons.
- It is possible to extract perceptual structures from the sensory data (i.e., determine groups of scene features that are likely to belong to the same object). Hypotheses generated by matching features in some perceptual structure are expected to be consistent with the rest of features in the structure. Hypotheses

inconsistent with any of those features can be eliminated, or at least given a lower priority than the consistent ones.

Performance of model-driven verification can be improved by indexing the scene features [56]. Similarly, indexing of model features can improve data-driven verification. For example, Chen and Kak [22] associate with each model feature a principal direction, which can represent a characteristic position or orientation of the feature. Those principal directions, expressed with respect to an object coordinate frame ($\mathcal{F}_O$), are indexed by mapping them onto a tesselated sphere, called feature sphere. At run-time, the principal direction of the scene feature under consideration is computed, transformed from a world frame ($\mathcal{F}_W$) to $\mathcal{F}_O$, and mapped to the feature sphere. The scene feature is then compared with only those model features whose principal directions are mapped to the same feature-sphere cell. Neighboring cells can also be considered to handle the uncertainties in computing both the pose hypothesis and the principal direction of the scene feature. The main limitation of feature-sphere indexing is that it *partially* utilizes the constraints provided by the scene feature: constraints on only two of the object's six DOF's are utilized. Constraints on the remaining DOF's, as well as viewpoint-invariant attributes (e.g., dihedral angle of an edge, radius of a cylinder) are not utilized in order to give the index more discriminative power.

In this chapter, we present an indexing scheme for touch-based data-driven verification. This scheme attempts to utilize *all* constraints provided by a tactile feature in order to build a robust index with high discriminative power. A set of standard tactile features is considered for verification (e.g., surface patch, edge segment). For each considered feature type, we construct a set of constraints that bounds the object poses which are supported by the tactile feature. A pose constraint (or a *filter*) is simply a group of numeric bounds on each of the six pose parameters. The filters are made invariant to the location of the tactile feature in the world frame by expressing them with respect to a tactile-feature frame, $\mathcal{F}_\mathcal{D}$. This enables us to construct the filters, as well as index them off-line. Each filter corresponds to a bound on the object poses that lead to a match between a model feature, F^m, and the tactile feature F^s. Since, generally, we do not have *a priori* knowledge about which model feature F^m

corresponds to F^s, a filter is constructed for *every* possible F^m. For example, if the scene feature is a surface patch and the model object is a rectangular polyhedron, then there will be six filters, one for every model surface. At run-time, we determine the consistency of hypothesis H with feature F^s by first transforming H from $\mathcal{F}_W$ to $\mathcal{F}_D$. The index, called *filter index*, corresponding to the type of F^s is then traversed using the pose parameters of H as keys. Based on this traversal, a small, possibly empty, subset of filters is retrieved. For hypothesis H to be supported by feature F^s, it must be accepted by at least one of the retrieved filters; i.e., each pose parameter of H must lie within the respective bound of a filter.

Main characteristics of the proposed indexing scheme are:

1. Almost all the constraints provided by a scene feature are utilized to build a robust index with high discriminative power. As will become clear later, this is quite easily achieved by selecting $\mathcal{F}_D$ as the reference frame rather than $\mathcal{F}_O$, as in [22].

2. The scheme can handle more complex feature primitives than the ones in [22]. For example, as will be shown later, a polygonal surface patch along with one of its edges (an SE-patch) can be considered a single primitive in our scheme. This capability further improves the discriminative power of indexing.

3. The filter index is designed to index *ranges* and not *values*. This makes it capable of handling cases when we only have loose bounds on a pose parameter (e.g., the translation along the x axis lies between -10 mm and 5 mm). This also makes it capable of explicitly handling uncertainty in estimating a parameter (e.g., the rotation about the x axis $= 45° \pm 3°$).

4. The filter index consists of several levels, where each level corresponds to a DOF. The use of a multi-level index has the advantage of requiring a space of polynomial complexity. In addition, viewpoint-invariant attributes can be incorporated into the index by simply adding more levels.

The remainder of the chapter is organized as follows. The next section presents a detailed description of the proposed indexing scheme. As in the previous chapter, S- and SE-patches will be considered as representative tactile features. Computational complexity of the scheme is analyzed in Section 4.3. Section 4.4 shows some experimental results using simulated and real data. Finally, Section 4.5 provides the conclusions[1].

4.2 Proposed Indexing Scheme

4.2.1 Overview

The consistency of scene feature F^s with hypothesis ${}^{\mathcal{W}}\mathbf{T}_{\mathcal{O}}$ is determined as follows:

1. In the off-line stage, filters are constructed by matching S- and SE-patches with corresponding model features (Sections 4.2.3 and 4.2.4, respectively). The constructed filters are then indexed (Section 4.2.5).

2. Hypothesis ${}^{\mathcal{W}}\mathbf{T}_{\mathcal{O}}$ is transformed from $\mathcal{F}_{\mathcal{W}}$ to the tactile-feature frame associated with feature F^s, $\mathcal{F}_{\mathcal{D}}$, to form ${}^{\mathcal{D}}\mathbf{T}_{\mathcal{O}}$.

3. The six pose parameters of ${}^{\mathcal{D}}\mathbf{T}_{\mathcal{O}}$ are extracted (Section 4.2.2).

4. The extracted parameters of ${}^{\mathcal{D}}\mathbf{T}_{\mathcal{O}}$ are used to navigate the filter index to retrieve a set of filters.

5. If the set selected by the filter index is empty, then ${}^{\mathcal{D}}\mathbf{T}_{\mathcal{O}}$ is considered inconsistent with F^s; otherwise, it is compared with each filter in the retrieved set.

6. If ${}^{\mathcal{D}}\mathbf{T}_{\mathcal{O}}$ is not accepted by any filter, then the hypothesis is considered inconsistent with feature F^s. Otherwise, for each accepting filter, a *fine matching* step takes place by comparing F^s with the model feature corresponding to the accepting filter in order to decide whether F^s supports ${}^{\mathcal{D}}\mathbf{T}_{\mathcal{O}}$ (Section 4.2.6).

4.2.2 Extraction of Pose Parameters

As in the previous chapter, a 3-D pose of an object, $\mathcal{O}$, with respect to a frame, $\mathcal{W}$, is expressed by three rotations about the principal axes, followed by three translations

[1] A preliminary version of this work was published in *Proc. IEEE Int. Conf. Multisensor Fusion and Integ. for Intell. Sys.*, Las Vegas, Nevada, October 1994 [14].

along the same axes. Given a pose. $\mathbf{K}$. in the form of a transformation matrix.

$$\mathbf{K} = \begin{pmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{4.1}$$

a unique interpretation is obtained as follows: Comparing (3.1) and (4.1). we obtain

$$\theta_y = \arcsin(-k_{31}).$$

We choose the solution of θ_y that lies in the range $[-90°, 90°]$. Assuming $|\theta_y| \neq 90°$. θ_x and θ_z are given by

$$\begin{aligned} \theta_x &= \text{arctan2}(k_{32}, k_{33}) \\ \theta_z &= \text{arctan2}(k_{21}, k_{11}). \end{aligned}$$

On the other hand, if $\theta_y = \pm 90°$, we get

$$\theta_x \mp \theta_z = \text{arctan2}(\pm k_{12}, \pm k_{13}). \tag{4.2}$$

respectively. Equation (4.2) implies that if $|\theta_y| = 90°$. then θ_x and θ_z are dependent. To get a unique interpretation. we set $\theta_x = 0$. and then solve for θ_z from (4.2). The translation parameters are directly obtained as follows:

$$\begin{pmatrix} t_x & t_y & t_z \end{pmatrix} = \begin{pmatrix} k_{14} & k_{24} & k_{34} \end{pmatrix}.$$

4.2.3 Filters of an S-patch

In this section and the next one, we will present how *ideal* filters are computed for each of the two considered feature types. Uncertainty in the sensory data is handled by dilating the ideal-filter bounds (in the current implementation, the extent of dilation is determined empirically). Notice that since the filter index is constructed off-line. worst-case uncertainty should be considered.

For the S-patch feature. filters are constructed by matching each model surface. S_i^m. with the scene S-patch (recall that the reference frame is $\mathcal{F}_D$). This matching process determines θ_x, θ_y and t_z exactly and obtains bounds on t_x and t_y, but cannot determine or find a bound on θ_z.

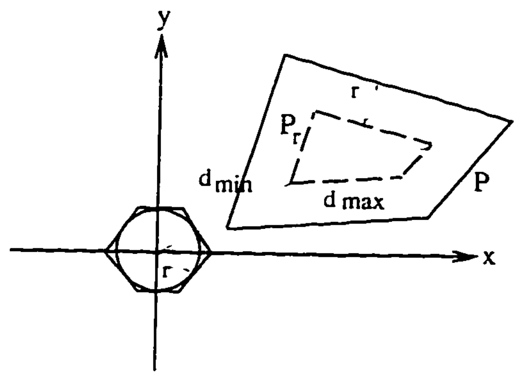


Figure 4.1: Computing bounds on t_x and t_y.

Derivation of θ_{ix}, θ_{iy} and t_{iz} is presented in Section 3.2.2 (equations (3.7)-(3.9)). After the transformation of surface S_i^m by those parameters, the problem is reduced to a 2-D localization problem in the $x-y$ plane. Although there is insufficient information to compute t_{ix} and t_{iy} exactly, it is still possible to obtain bounds on those parameters. In order to simplify this process, we approximate the tactile polygon by a circle of radius r inside it. The partially transformed surface S_i^m in the $x-y$ plane can be represented by a binary function $P(x,y)$ which is 1 if point (x,y) lies inside the surface polygon, and 0 otherwise. Define $P_r(x,y)$ as the polygon formed by "shrinking" $P(x,y)$ by r (see Fig. 4.1). Assuming $\theta_{iz} = 0$, it can be shown that the range of possible translations (t_{ix}, t_{iy}) such that the sensor circle is entirely inside $P(x,y)$ is any point inside $P_r(-x,-y)$. Since the rotation of the object by θ_{iz}, which is unknown off-line, about the z axis results in the same rotation of $P_r(x,y)$, we obtain the following bounds on t_{ix} and t_{iy}:

$$\mid t_{ix} \mid, \mid t_{iy} \mid \in [0, d_{max}]$$

where d_{max} is the maximum distance between the vertices of $P_r(x,y)$ and the origin. A tighter constraint can be obtained, if we use bounds on $\sqrt{t_{ix}^2 + t_{iy}^2}$, instead of t_{ix} and t_{iy} separately:

$$\sqrt{t_{ix}{}^2 + t_{iy}{}^2} \in [d_{min}, d_{max}] \qquad (4.3)$$

where d_{min} is the minimum distance between $P_r(x,y)$ and the origin (notice that d_{min} is 0, if the origin is inside $P_r(x,y)$).

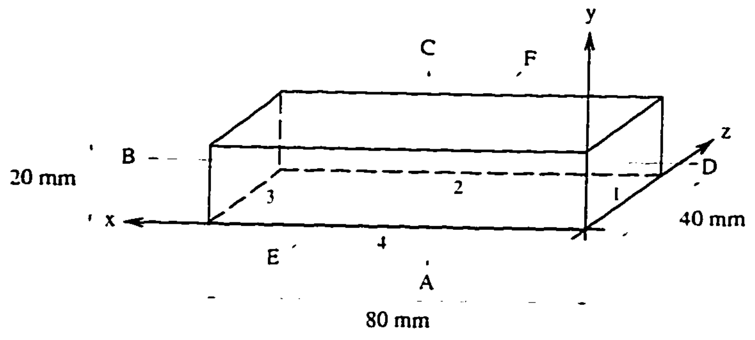


Figure 4.2: A rectangular polyhedron.

Surface	θ_x	θ_y	t_z	$\sqrt{t_x{}^2 + t_y{}^2}$
A	$(-90, -90) \mp \delta\theta_x$	$(0, 0) \mp \delta\theta_y$	$(0, 0) \mp \delta t_z$	$(7.07, 82.76) \mp \delta t_{xy}$
B	$(0, 0) \mp \delta\theta_x$	$(-90, -90) \mp \delta\theta_y$	$(-80, -80) \mp \delta t_z$	$(7.07, 38.08) \mp \delta t_{xy}$
C	$(90, 90) \mp \delta\theta_x$	$(0, 0) \mp \delta\theta_y$	$(-20, -20) \mp \delta t_z$	$(7.07, 82.76) \mp \delta t_{xy}$
D	$(0, 0) \mp \delta\theta_x$	$(90, 90) \mp \delta\theta_y$	$(0, 0) \mp \delta t_z$	$(7.07, 38.08) \mp \delta t_{xy}$
E	$(180, 180) \mp \delta\theta_x$	$(0, 0) \mp \delta\theta_y$	$(0, 0) \mp \delta t_z$	$(7.07, 76.49) \mp \delta t_{xy}$
F	$(0, 0) \mp \delta\theta_x$	$(0, 0) \mp \delta\theta_y$	$(-40, -40) \mp \delta t_z$	$(7.07, 76.49) \mp \delta t_{xy}$

Table 4.1: Filters associated with the model object shown in Fig. 4.2. assuming a 10 mm × 10 mm square S-patch ($\delta\theta_x$, $\delta\theta_y$, δt_z and δt_{xy} are thresholds to cope with the uncertainty in the sensory data).

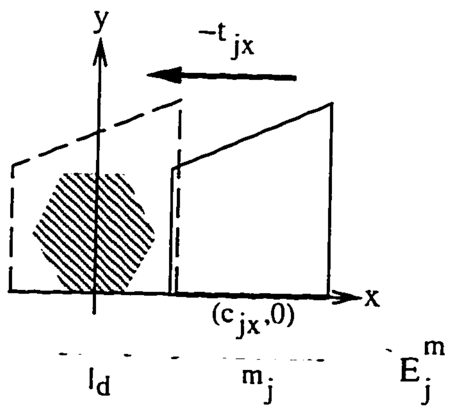


Figure 4.3: Translation of model edge E_j along the x axis.

Equations (3.7)-(3.9) and (4.3) are the set of bounds that constitute the ideal filter associated with surface S_i^m. As an illustration of the above procedure, consider the rectangular polyhedron shown in Fig. 4.2, and a 10 mm $\times$ 10 mm square S-patch. Filters associated with this object are shown in Table 4.1. Each row in this table represents the filter corresponding to the match between the surface in the first column of the table and the S-patch.

4.2.4 Filters of an SE-patch

Filters of an SE-patch are constructed by matching each model SE-surface, (S_i^m, E_j^m), with the scene SE-patch. This matching process determines θ_x, θ_y, θ_z, t_y and t_z exactly, and obtains a bound on t_x.

The parameters θ_{ix}, θ_{iy}, t_{iz}, θ_{jz} and t_{jy} are derived in Section 3.2.2 (see equations (3.7)-(3.9), and (3.13)-(3.14)). After transformation by the above parameters, it can be shown that surface S_i^m becomes coincident with $x-y$ plane, and edge E_j^m becomes aligned with the x axis. Although there is insufficient information to determine t_{jx} exactly, we can still obtain bounds on it. Translation along the x axis can be split into two steps; the first to translate the center of E_j^m to the origin, and the second to further translate E_j^m so that the scene edge remains fully inside it. Let $(c_{jx}, 0)$ be the center of edge E_j^m, m_j the length of E_j^m, and l_d the length of the scene edge (see Fig. 4.3). The second translation component is bounded by $\mid \frac{(m_j - l_d)}{2} \mid$. Therefore, we

Edge	θ_z	t_y	t_x
1	$(90,90) \mp \delta\theta_z$	$(0,0) \mp \delta t_y$	$(0,40) \mp \delta t_x$
2	$(180,180) \mp \delta\theta_z$	$(40,40) \mp \delta t_y$	$(0,80) \mp \delta t_x$
3	$(-90,-90) \mp \delta\theta_z$	$(80,80) \mp \delta t_y$	$(-40,0) \mp \delta t_x$
4	$(0,0) \mp \delta\theta_z$	$(0,0) \mp \delta t_y$	$(-80,0) \mp \delta t_x$

Table 4.2: Filters associated with surface A of the rectangular polyhedron in Fig. 4.2, for the SE-patch case, and assuming $l_d = 0$. The bounds on θ_x, θ_y and t_z are given by the first row of Table 4.1 ($\delta\theta_z$, δt_y and δt_x are thresholds to cope with the uncertainty in the sensory data).

have

$$t_{jx} \in [-c_{jx} - \frac{(m_j - l_d)}{2}, -c_{jx} + \frac{(m_j - l_d)}{2}]. \tag{4.4}$$

It can be seen that the bounds on t_{jx} in (4.4), apart from all the other ones, depend on a parameter that is only known at run-time, l_d. To overcome this difficulty, we compute these bounds off-line assuming l_d is equal to 0, or the minimum possible length of an extracted edge. Then, at run-time, it is possible to update the lower and upper bounds of t_{jx} at a negligible cost by adding $\pm\frac{l_d}{2}$, respectively. Note that if $l_d > m_j$, then SE-surface (S_i^m, E_j^m) can not match the scene SE-patch, and accordingly the corresponding filter is eliminated.

Equations (3.7)-(3.9), (3.13)-(3.14) and (4.4) are the set of bounds that constitute the ideal filter associated with SE-surface (S_i^m, E_j^m). As an illustration of the above procedure, Table 4.2 shows four of the 24 filters of the rectangular polyhedron shown in Fig. 4.2 (only those associated with the edges of surface A).

4.2.5 The Filter Index

For a polyhedron with S surfaces, and an average of E_S edges per surface, the number of model filters is S and SE_S for S- and SE-patches, respectively. Being *invariant* to the location of the scene feature in $\mathcal{F}_W$, these filters can be indexed off-line. Indexing greatly reduces the number of filters that are actually compared with a given pose hypothesis, hence achieving significant improvement in performance.

Assume that we have a set of filters, $\mathcal{T}$, on a set of pose parameters, $\mathcal{R}$, where

$$\mathcal{R} = \begin{cases} \{\theta_x, \theta_y, t_z, \sqrt{t_x^2 + t_y^2}\} & \text{for an S-patch, and} \\ \{\theta_x, \theta_y, t_z, \theta_z, t_y, t_x\} & \text{for an SE-patch.} \end{cases}$$

Given the set of filters $\mathcal{T}$, our objective is to build a multi-level index on the parameters of $\mathcal{R}$, to efficiently reduce the *filter uncertainty* of an input pose hypothesis. The *filter uncertainty* of a hypothesis, H, is defined as the minimal subset of $\mathcal{T}$ known to contain the filter that accepts H, if such a filter exists. Clearly, the *initial* filter uncertainty of any hypothesis is $\mathcal{T}$.

The proposed index can be viewed as a tree of depth $m-1$, where

- $m = 4$ and 6 for S- and SE-patches, respectively,
- Each node in the tree is an array of pointers,
- Each pointer in a non-leaf node is either pointing to a child node or nil, and
- Each pointer in a leaf node is either pointing to a set of filters or nil.

For each tree node, N, there is an associated set of filters, $\mathcal{U} \subseteq \mathcal{T}$, representing the filter uncertainty of any hypothesis mapped to that node. The function of N is to reduce this *input* filter uncertainty by utilizing the bounds of $\mathcal{U}$ on a pose parameter, $r_j \in \mathcal{R}$, where $j-1 \in [0, m-1]$ is the depth of N. A filter $U_i \in \mathcal{U}$ can be represented as

$$U_i = \{(lb_{ik}, ub_{ik}) : k = 1, m\}$$

where (lb_{ik}, ub_{ik}) are the lower and upper bounds of U_i on r_k. Reduction of filter uncertainty is performed by node N as follows:

- The range of the bounds of $\mathcal{U}$ on r_j, $(\min_i(lb_{ij}), \max_i(ub_{ij}))$, is uniformly mapped to the entries of N. For example, let us assume that $\mathcal{U}$ is the set of filters shown in Table 4.1. The range of the bounds on θ_x is $[-180°, 180°]$. If we allocate an array of 36 entries to N, then each node entry, N_k, will correspond to the range $[-180° + 10°(k-1), -180° + 10°k]$, and N will be the root node shown in Fig. 4.4.

- For each entry, N_k, in N, there is an associated set of filters, $\mathcal{V}_k$, representing the *output* filter uncertainty of this entry. Set $\mathcal{V}_k$ is the subset of filters in $\mathcal{U}$ with the bound on r_j overlapping with the range mapped to N_k. An example of

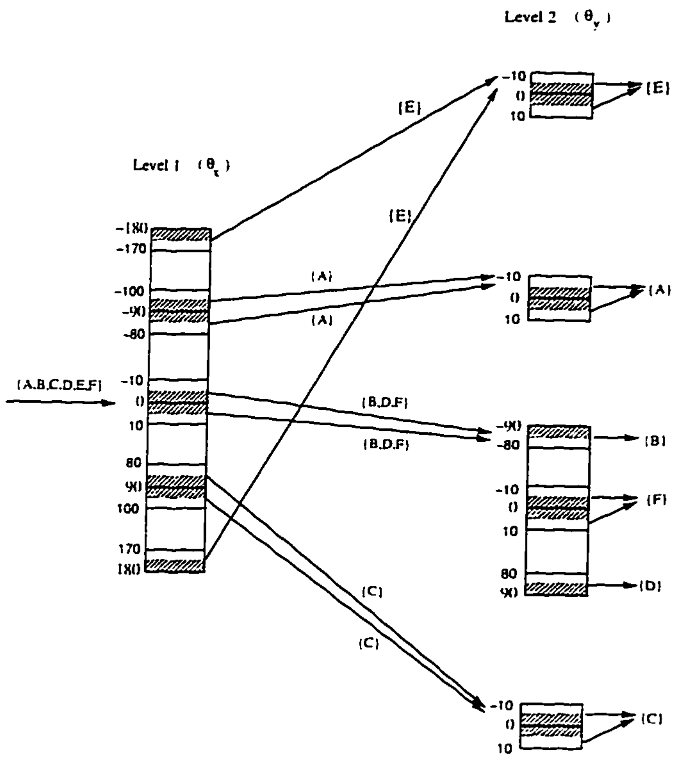
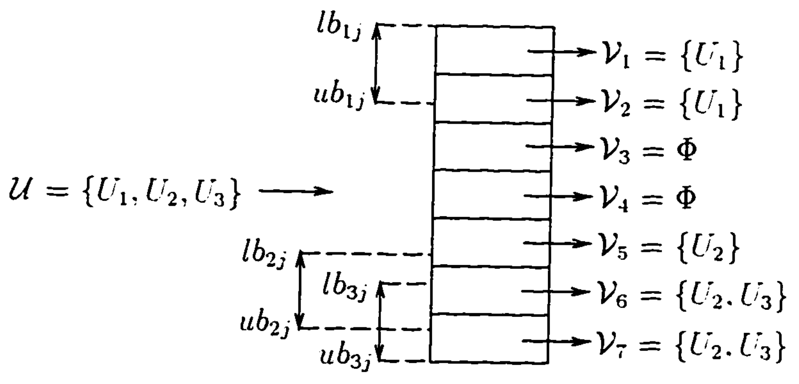


Figure 4.4: A filter index on the filters shown in Table 4.1 (only two levels are shown for the sake of clarity). It is assumed that $\delta\theta_x = \delta\theta_y = 5°$. Arrows are labelled with sets representing the filter uncertainty at the different stages of the index.



lb_{1j}
ub_{1j}
$\mathcal{U} = \{U_1, U_2, U_3\}$
lb_{2j}
lb_{3j}
ub_{2j}
ub_{3j}
$\mathcal{V}_1 = \{U_1\}$
$\mathcal{V}_2 = \{U_1\}$
$\mathcal{V}_3 = \Phi$
$\mathcal{V}_4 = \Phi$
$\mathcal{V}_5 = \{U_2\}$
$\mathcal{V}_6 = \{U_2, U_3\}$
$\mathcal{V}_7 = \{U_2, U_3\}$

Figure 4.5: Reduction of filter uncertainty by a tree node of depth $j - 1$.

this process is shown in Fig. 4.5. It can seen that the filters are "dilated" when indexed, where the extent of dilation depends on the resolution of the index arrays.

- If $\mathcal{V}_k = \Phi$, then $\mathcal{N}_k$ is set to nil. Otherwise, if $\mathcal{V}_k \neq \Phi$ and $\mathcal{N}$ is a non-leaf node, then $\mathcal{N}_k$ points to a child node that is constructed by assuming that $\mathcal{V}_k$ is its input uncertainty. Finally, if $\mathcal{V}_k \neq \Phi$ and $\mathcal{N}$ is a leaf node, then $\mathcal{N}_k$ points to $\mathcal{V}_k$. Entries of $\mathcal{N}$ with the same output filter uncertainty point to the same child node.

An example of a filter index is shown in Fig. 4.4.

At run-time, the initial filter uncertainty $\mathcal{T}$ of a hypothesis H is successively reduced by traversing the filter index. Index traversal is performed by mapping the pose parameters of H into the corresponding index entries. If the traversal encounters a nil pointer, then H is considered inconsistent with the scene feature. Otherwise, H is compared with the retrieved set of filters, associated with a leaf-node entry, in order to decide whether it is inconsistent with the scene feature.

4.2.6 Fine Matching

Since model filters provide *gross* bounds on the object pose, especially in the S-patch case, an extra verification step is needed to perform *fine* matching between the scene feature and the hypothesized model object. Fine matching is performed as follows. Model feature F^m assumed to correspond to scene feature F^s is transformed from $\mathcal{F}_O$ to $\mathcal{F}_D$. Identity of F^m is directly given by the filter that has accepted the hypothesis (recall that each filter is constructed by matching F^s with some corresponding model feature). The hypothesis is accepted only if the scene patch is entirely covered by the model surface and the scene edge, in the SE-patch case, is entirely covered by a model one. Notice that the identity of this model edge is also directly determined by the accepting filter. To perform this check, we compute the minimum distance between the scene-patch vertices and the model surface. Furthermore, in the SE-patch case, we compute the minimum distance between the scene edge vertices and the corresponding model edge. The hypothesis is accepted only if the maximum of those distances is less than some threshold.

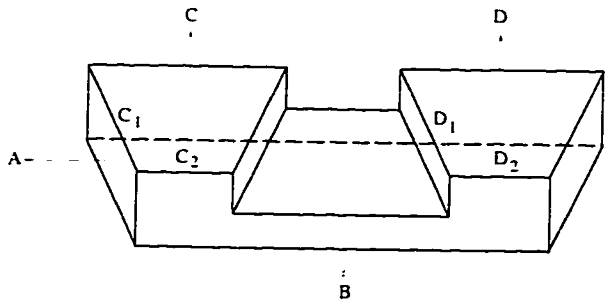


Figure 4.6: A polyhedral model object.

Note that there may be more than one filter accepting a hypothesis, if there exists overlapping between filters. For example, suppose that a model object has several surfaces with the same equation. In such a case, the S-patch filters corresponding to those surfaces will have identical bounds on θ_x, θ_y and t_z, and possible overlapping between the bounds on $\sqrt{t_x^2 + t_y^2}$. If there is more than one filter accepting the hypothesis, then the verification process described above is repeated for each accepting filter.

4.3 Complexity Analysis: A Practical Approach

In this section, we investigate the computational complexity of the proposed verification scheme. In order to obtain practical results, we will base the following analysis on a number of assumptions, stated below, that are expected to be valid in a wide range of applications.

Assume that the model object is a polyhedron with S surfaces and an average of E_S edges per surface. A possible naive verification technique would be as follows. The equation of the scene patch is first compared with those of the model surfaces. This stage has a time complexity of order $O(S)$. For each model surface that has the same equation as that of the scene patch, the model surface and the scene patch are compared. The hypothesis is accepted only if the scene patch is inside the model surface and the scene edge, for SE-patches only, is inside a model one. This process has a worst-case time complexity of order $O(pE_SE_D)$, where p is the maximum number of model surfaces having the same equation, and E_D is the number of edges in the scene patch. Thus, the whole verification process has a time complexity that ranges from $O(S)$ to $O(S + pE_S E_D)$. In contrast, we will show below that, under reasonable assumptions, our verification scheme has a complexity that ranges from $O(1)$ (i.e., constant time *independent* of the number of model features) to $O(pE_S E_D)$ and $O(E_S E_D)$, for S- and SE-patches, respectively.

For the filtering step, the sequential comparison of a given pose hypothesis with model filters has a time complexity of order $O(S)$ for an S-patch, and $O(SE_S)$ for an SE-patch. To estimate the reduction in time complexity achieved by filter indexing, we make the following reasonable assumptions (refer to the model object in Fig. 4.6):

1. Non-similar model surface equations (e.g., equations of surfaces A and B in Fig. 4.6) are sufficiently different so as not to cause overlapping between the corresponding sub-filters on θ_x, θ_y and t_z (a sub-filter is simply a subset of filter bounds). This assumption implies that the application of the sub-filters on θ_x, θ_y and t_z will reduce the surface uncertainty (uncertainty about the model surface corresponding to the scene patch) to at most those surfaces that have the same equation (e.g., surfaces C and D).

2. For SE-surfaces having the same surface equation, non-similar edge equations (e.g., equations of edges C_1 and C_2) are sufficiently different so as not to cause overlapping between the corresponding sub-filters on θ_z and t_y. Notice that the edge equation is the one defined in (3.10). This assumption together with the previous one implies that the application of the sub-filters on θ_x, θ_y, t_z, θ_z and t_y in the SE-patch case will reduce the surface-edge uncertainty to at most those pairs that have the same surface and edge equations (e.g., the SE-surfaces (C, C_2) and (D, D_2)).

3. For SE-surfaces having the same surface and edge equations, the edges are sufficiently far from each other so as not to cause overlapping between the corresponding bounds on t_x. This assumption together with the previous ones implies that the application of the filters in the SE-patch case will reduce the surface-edge uncertainty to at most a *single* SE-surface.

4. The resolution of the filter-index arrays is sufficient to separate between non-overlapping bounds. This assumption ensures that the filter index is able to

capture the discriminative power of the filters.

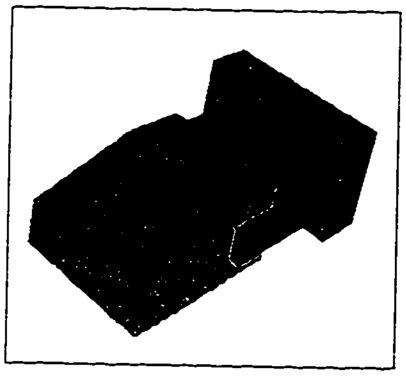
Let us obtain an upper limit on the size of filter uncertainty (number of filters). For an S-patch, assumptions 1 and 4, imply that a three-level index on θ_x, θ_y and t_z will reduce the filter-uncertainty size to at most p. Extending the index to include the bounds on $\sqrt{t_x^2 + t_y^2}$ will not help in reducing the upper limit on the uncertainty size, due to the possible overlapping between these bounds. For an SE-patch, assumptions 1–4 imply that a six-level index, on all the pose parameters, will reduce the filter uncertainty size to at most one. Thus, filter indexing reduces the time complexity of filtering from $O(S)$ and $O(SE_S)$ to $O(p)$ and $O(1)$, for S- and SE-patches, respectively.

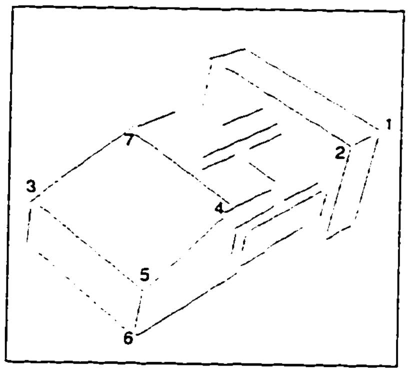
Having *a priori* knowledge about the model object can help in reducing the space complexity of the filter index, while keeping the same upper limit on the filter-uncertainty size. For example, assume that the model object is a convex polyhedron. Since all surfaces of a convex polyhedron have different orientations, it can be easily shown that, under the abovementioned assumptions, a two-level index on θ_x and θ_y will reduce the filter-uncertainty size in the S-patch case to at most one. In addition, since edges that belong to the same surface have different orientations along the surface plane[2], it can be easily shown that a three-level index on θ_x, θ_y and θ_z in the SE-patch case will also reduce the uncertainty size to at most one.

For the fine-matching step, the cost of comparing a model surface with a scene patch has a time complexity of order $O(E_S E_D)$. Thus, under the abovementioned assumptions, the proposed verification scheme has a complexity that ranges from $O(1)$ to $O(pE_S E_D)$ and $O(E_S E_D)$, for S- and SE-patches, respectively.

It is interesting to compare the complexity bounds of our scheme with those of the feature-sphere indexing scheme, described in Section 4.1. It can be easily shown that the complexity of feature-sphere indexing ranges from $O(1)$ to $O(qE_S E_D)$, for both S- and SE-patches, where q is the maximum number of model surfaces having the same *orientation*. Notice that the complexity of verification is the same for both S- and SE-patches, since feature-sphere indexing can not utilize the knowledge of a surface patch along with one of its edges simultaneously. Since $q \geq p$, we can see that our verification scheme has better complexity bounds. In the next section, we will

[2]Notice that surfaces of a convex polyhedron are convex polygons.


(a)



(b)

Figure 4.7: A synthetic scene: (a) scene, (b) extracted junctions.

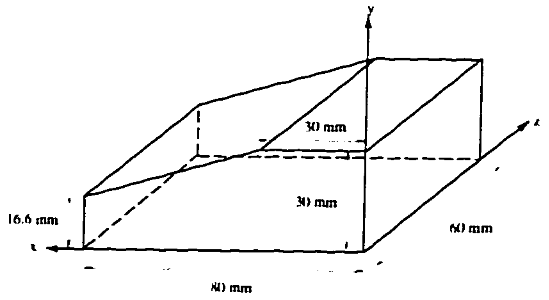


Figure 4.8: A mouse-like object.

demonstrate its superiority with respect to the discriminative power of indexing.

4.4 Experimental Results

The proposed verification scheme has been tested using simulation and real experiments. To demonstrate the performance of the scheme, we use vision to generate hypotheses, while touch to verify them. The technique used for vision-based hypothesis generation is the same as the one used in the previous chapter (see Section 3.4.2).

4.4.1 Simulation Experiments

Fig. 4.7(a) shows a scene of a three-fingered robot hand holding a computer mouse-like object (see Fig. 4.8). Fig. 4.7(b) shows results of the feature extraction process. Seven three-edge junctions are extracted. From those junctions, 18 J-features are used to generate pose hypotheses. A 10 mm $\times$ 10 mm planar-array tactile sensor is

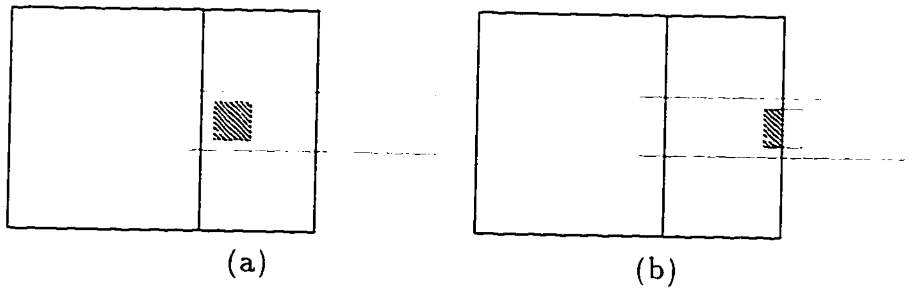


Figure 4.9: A top view of the mouse showing the tactile sensor mounted on the middle finger: (a) first position, (b) second position.

assumed to be mounted on the middle finger. Fig. 4.9 shows a top view of the mouse and the tactile images obtained by assuming that the tactile sensor is mounted at two different positions. Uncertainty is introduced into the tactile data in a manner similar to that presented in the previous chapter (see Section 3.4.2), assuming $\Delta r^f = 5$ mm, $\Delta\theta^f = 5°$, and $\Delta r^t = 0.5$ mm. The error thresholds for the filter index are empirically chosen to be 7° for rotational DOF's, and 13 mm for translational ones. The array resolution for each stage of the filter index is set to half the corresponding error threshold.

Results of verifying the visually-generated hypotheses are summarized in Table 4.3. Columns in this table are:

1. *Sensor Position:* the assumed sensor position as shown in Fig. 4.9
2. *Data Type:* type of extracted tactile data (S- or SE-patch)
3. *Initial Hypotheses:* number of visually-generated hypotheses
4. *Hypotheses Accepted by Index:* number of hypotheses for which navigation of the filter index retrieves one or more filters
5. *Hypotheses Accepted by Filtering:* number of hypotheses that are accepted by a filter
6. *Hypotheses Accepted by Fine Matching:* number of hypotheses that pass the fine-matching test
7. *Valid Hypotheses:* number of valid hypotheses among those that have passed the fine-matching test

Sensor Position	Data Type	Initial Hypos.	Hypos. Accepted by			Valid Hypos.
			Index	Filtering	Fine Matching	
Fig. 4.9(a)	S-patch	880	50	24	12	10
Fig. 4.9(b)	SE-patch	880	22	11	11	10

Table 4.3: Results of the simulation experiments.

The following can be observed from the table:

- The filter index is very efficient in eliminating hypotheses that are inconsistent with the tactile data in a constant time, without any comparison with filters. Out of the 868 (869) hypotheses that are inconsistent with the S-patch (SE-patch), only 38 (11) are accepted by the index (notice that number of inconsistent hypotheses = total number of hypotheses - number of hypotheses accepted by the fine matcher). Thus, the filter index succeeds in eliminating 96% and 99% of the inconsistent hypotheses for S- and SE-patches, respectively. As explained in Section 4.2, this robustness is due to the fact that the filter index captures the constraints provided by the tactile feature. In addition, notice that the filter index is slightly more robust in the SE-patch case, because of the extra constraints on θ_z, t_x and t_y.

- There are a few hypotheses that are accepted by the filter index, but not by the filters (26 and 11 for S- and SE-patches, respectively). This is because of the slight dilation of the filters when indexed (see Section 4.2.5).

- A very small number of the inconsistent hypotheses are passed to the computationally expensive fine matcher (12 and none for the two contact positions, respectively). In addition, notice that the fine matcher does not eliminate any hypotheses in the SE-patch case (Fig. 4.9(b)). However, this should not mislead the reader that the fine-matching stage is redundant in the SE-patch case. This is simply because the filters do not fully capture the shape of the tactile polygon.

- Very few erroneous hypotheses are found consistent with the tactile data (two and one hypotheses for the two contact positions, respectively). Further verifi-

Hypothesis	Valid	First Position		Second Position	
		Filtering	Fine Matching	Filtering	Fine Matching
a	no	rejected		rejected	
b	no	rejected		rejected	
c	no	accepted	rejected	rejected	
d	no	accepted	accepted	rejected	
e	no	accepted	accepted	accepted	accepted
f	yes	accepted	accepted	accepted	accepted

Table 4.4: Results of verifying the pose hypotheses shown in Fig. 4.10.

cation can be done by utilizing other available tactile features, as well as visual data.

To demonstrate the robustness of the filter index compared to that of the feature sphere [22], we compare the discriminative power of both. Since θ_x and θ_y represent the orientation of a surface patch, a two-level filter index on θ_x and θ_y would have a very similar discriminative power to that of the feature sphere. Using this two-level index in verification, we have found that, on the average, only about 62% of the inconsistent hypotheses are rejected by the index, for both contact positions. This is considerably lower than the discriminative power of the proposed filter index (recall that 96% and 99% of the inconsistent hypotheses are eliminated, for the two contact positions, respectively). In addition, notice that an SE-patch would be treated as two independent features, an S-patch and an edge, in the feature-sphere scheme. Thus, determining the consistency of an SE-patch with a hypothesis would involve traversing two indices of relatively low discriminative power, not just one index of high discriminative power, as in our case.

It is interesting to visualize what hypotheses are accepted or rejected by the two verification stages (filtering and fine matching). Fig. 4.10 shows a sample of six visually-generated hypotheses. Results of verifying those hypotheses are shown in Table 4.4. Notice that the hypothesis shown in Fig. 4.10(c) is accepted by the filtering process in the S-patch case (see Fig. 4.9(a)), although there is no model surface in contact with the tactile sensor. This is because the filters of an S-patch provide only loose constraints on t_x and t_y, and no constraint on θ_z (refer to Section 4.2.3).

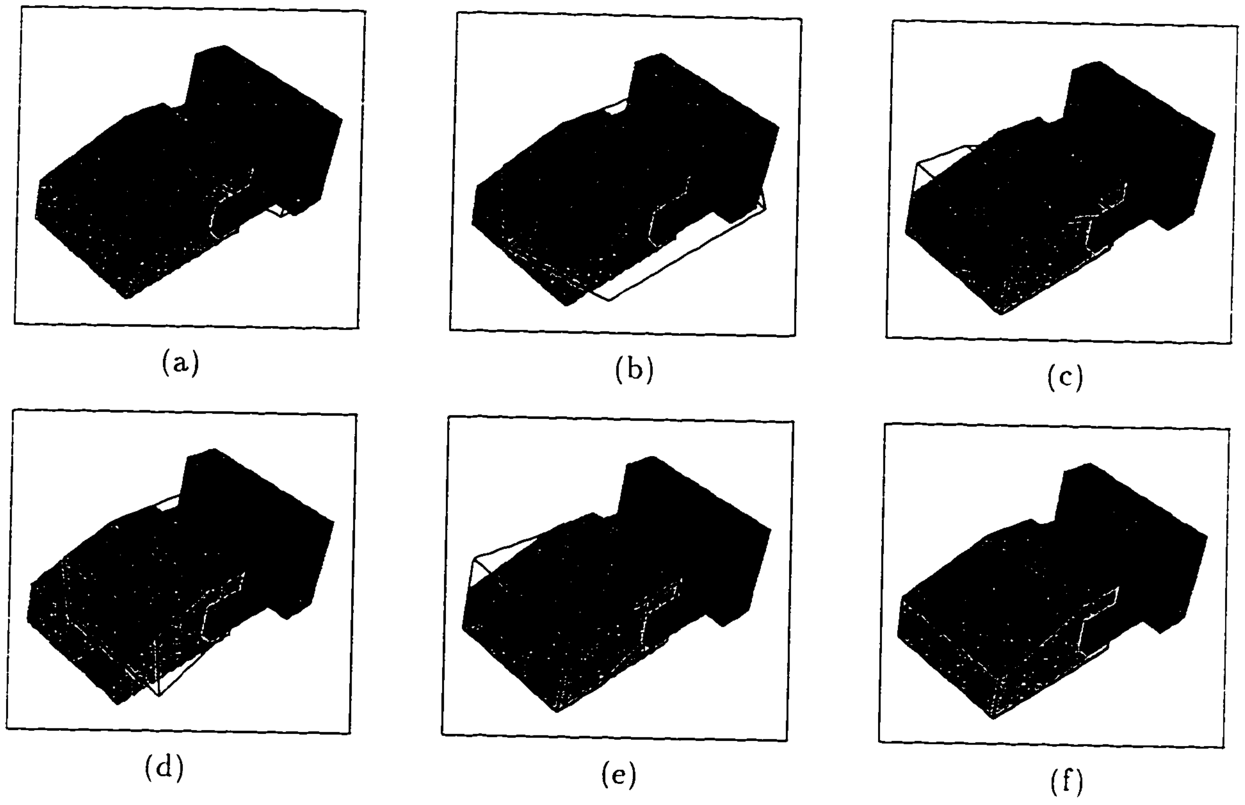


Figure 4.10: A sample of six visually-generated pose hypotheses.

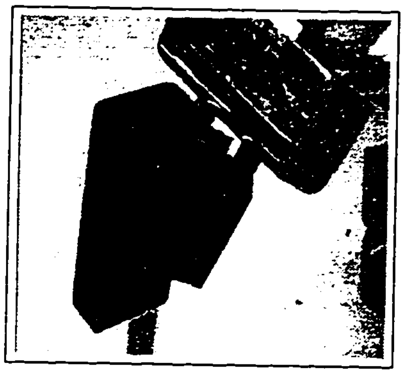
Data Type	Initial Hypos.	Hypos. Accepted by			Valid Hypos.
		Index	Filtering	Fine Matching	
SE-patch	36	2	2	2	2

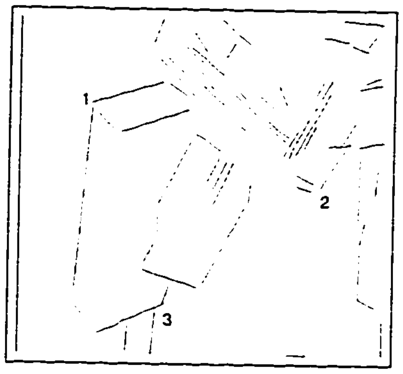
Table 4.5: Results of the real experiment.

4.4.2 Real Experiments

Fig. 4.11(a) shows a scene of a parallel-jaw gripper grasping a rectangular polyhedron. Three three-edge junctions are extracted, from which five J-features are used to generate pose hypotheses. To avoid the generation of equivalent hypotheses due to the symmetry of the model object[3], we only use any two adjacent model junctions for matching, instead of using the eight junctions. Fig. 4.12(a) shows the original tactile image. This image is filtered and thresholded to form the binary image shown in Fig. 4.12(b). An SE-patch is detected in this image. The error thresholds for the filter index are empirically chosen to be 5° for rotational DOF's, and 7 mm for

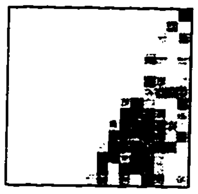
[3]Each pose of a rectangular polyhedron with three different dimensions can be described by four different transformation matrices.


(a)



(b)

Figure 4.11: A real scene: (a) scene. (b) extracted junctions.


(a)

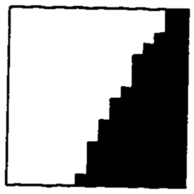

(b)

Figure 4.12: A real tactile image: (a) original, (b) thresholded.

translational ones. As in the simulation experiments, the array resolution for each stage of the filter index is set to half the corresponding error threshold.

The obtained results are summarized in Table 4.5. From these results, similar conclusions can be drawn as in the simulation experiments. In addition, we have used a two-level filter index, on θ_x and θ_y, for verification, in order to estimate the performance of feature-sphere indexing. We have found that the two-level filter index is only capable of eliminating 53% of the inconsistent hypotheses, as opposed to 100% for the six-level filter index in our case. This further demonstrates the robustness of the proposed filter index.

The flexibility of multi-level indexing in utilizing the available constraints can be seen in the task considered here (localizing an object in a parallel-jaw gripper). A constraint that can be utilized in such a task is the distance between the parallel jaws, D_J. This can be done by simply adding an extra level to the filter index, corresponding to D_J. In this case, the output filter uncertainty (refer to Section 4.3) for a particular D_J is the set of filters associated with surfaces that have parallel ones at distance D_J.

4.5 Conclusions

In this chapter, we have presented a robust indexing scheme for data-driven touch-based hypothesis verification. Robustness is achieved by utilizing almost all the constraints provided by tactile features, thus leading to an index with higher discriminative power than previous approaches. Utilization of these constraints is made feasible by selecting a tactile-feature frame as the reference frame. Such a selection enables the construction and indexing of an invariant set of constraints on object poses that are supported by the tactile feature. At run-time, the consistency of a hypothesis with the scene feature is efficiently determined by traversing the filter index corresponding to the feature type, using pose parameters of the hypothesis as keys. For the two types of tactile features considered in this work, S- and SE-patches, it has been shown that our scheme has a better time complexity than previous ones. The superiority of its discriminative power has been demonstrated using simulated and real experiments.

Although only two types of features are considered in this work, other types of 3-D features (e.g., edges, corners, cylinders) can be handled in a similar fashion.

Chapter 5

Vision-Based Hypothesis Verification

5.1 Introduction

Hypothesis verification techniques can be classified into two categories: feature-based and pixel-based. In feature-based verification, which is the more common approach, a hypothesis is validated by comparing model and scene features (e.g., [28, 41, 74]). A problem with this type of verification is that the feature-comparison step can be relatively complex, because of the large number of involved parameters. For example, suppose that we are comparing straight edges. In such a case, the parameters involved are edge equation, edge end points, and an indication of whether each edge end is *true* or *false*. An edge end is said to be either true or false, depending upon if it is likely to correspond to a model vertex or not, respectively. An example of true and false edge ends is shown in Fig. 5.1. Furthermore, the uncertainty in estimating each

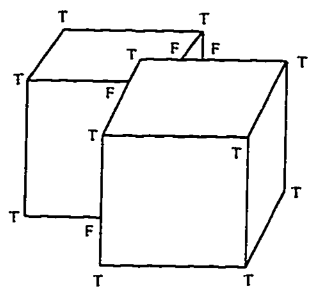


Figure 5.1: An example of true and false edge ends.

of the above parameters should be considered in the matching step as well. Another problem with feature-based techniques is that they are unsuitable for implementation on parallel graphics/vision hardware, since this type of hardware is geared towards pixel-based operations.

In pixel-based verification, the approach adopted in this chapter, an object hypothesis is verified by synthesizing an image of it, and comparing it, pixel wise, with the input image or a direct derivative of it (e.g., the edge image). This type of verification has several advantages over feature-based verification. Firstly, the implementation is much simpler. Secondly, the time complexity of verifying a hypothesis is independent of the scene complexity (number and types of scene features). Thirdly, it is insensitive to imperfections of the feature extraction process (e.g., missing or broken edges). Finally, the technique is suitable for parallel implementation on parallel graphics/vision hardware. This can be performed by using the graphics hardware to synthesize an image of the hypothesized object, and then using the vision hardware to compare scene and model images.

Pixel-based verification has been commonly used in systems that interpret range images (e.g., [12, 34]). The difficulty in vision-based 3DOR, in contrast to range-based 3DOR, is the complexity of forming realistic images of model objects. This difficulty is due to the large number of parameters involved in forming an intensity image (e.g., surface geometry, surface reflectance, and illumination [10]). For this reason, the few vision-based systems reported in the literature that perform pixel-based verification operate on the edge image instead of the intensity one. Most of these methods precompute a "distance" image, where the value of each pixel is its distance from the nearest edge pixel (e.g., [5, 13]). This distance image is then compared with the model edge image, a process that is commonly known as Chamfer matching. The goodness of the match is evaluated by computing the average distance between model and scene edges. Safranek *et al.* [66] use an approach that is based on Dempster-Shafer theory. For each model edge pixel, evidence for the hypothesis is gathered depending on two factors: the magnitude of the corresponding scene edge pixel, and the angle between the direction of the model pixel and that of the scene one. Evidence from all model pixels is then combined using Dempster-Shafer theory, in order to decide whether to accept the hypothesis.

There is a couple of problems with the above approaches. Firstly, they do not consider the uncertainty of model edge pixels, which is due to the error in object-pose estimation. As a consequence, a valid but noisy object hypothesis might be classified as invalid by the matching process. In order to build a robust matcher, it is important to accommodate object-pose uncertainty. Secondly, the chosen hypothesis-selection criteria can be misleading in some cases. For example, suppose that the object to be recognized is partially occluded. In such a case, an erroneous hypothesis of a small narrow object that is located along an edge (e.g., the hypothesis in Fig. 5.7(b)) might be selected as the valid one.

In this chapter, we present a pixel-based verification technique that attempts to overcome the above problems. Knowledge about the uncertainties of both scene and model edge pixels is utilized by appropriately *dilating* the scene edge image. An analytical method is presented to determine the extent of dilation, assuming some bounded error on the object pose. The dilated edge image can be viewed as a binary distance image, where the value of each pixel is either zero or one, depending on whether or not it corresponds to an edge, respectively. To overcome the inadequacy of the above hypothesis-selection criteria, we use one that depends on both average edge distance and hypothesized-object size.

The remainder of the chapter is organized as follows. The next section presents the verification technique in detail. Experimental results on real data are presented in Section 5.3. Finally, conclusions are drawn in Section 5.4[1].

5.2 Pixel-Based Verification

In this section, we first present an overview of the verification technique. This will be followed by a discussion on the uncertainties of scene and model edge pixels. Finally, we present an analytical method for estimating the extent of dilation of the scene edge image, given an error bound on the object pose.

[1]A preliminary version of this work was published in *Proc. IEEE Int. Conf. Robotics and Automation*, Minneapolis, Minnesota, April 1996 [17].

5.2.1 Overview

The verification algorithm can be written as follows:

```
Construct edge image, $I^s$, from the input scene image.
Dilate $I^s$ to accommodate scene and model uncertainties.
for each generated object hypothesis do
    Construct edge image, $I^m$, of the hypothesis.
    Set $hit\_count$, $miss\_count$ and $total\_count$ to 0.
    for each edge pixel in $I^m$ do
        $total\_count = total\_count + 1$
        if corresponding pixel in $I^s$ is an edge then
            $hit\_count = hit\_count + 1$
        else
            $miss\_count = miss\_count + 1$.
Accept the hypothesis with maximum $\rho = visibility\_ratio * size\_ratio$ , where
  $visibility\_ratio = \frac{hit\_count}{total\_count}$, and
  $size\_ratio = \frac{hit\_count}{\max\{hit\_count\}}$.
```

From the above algorithm, we observe that verification is merely a simple pixel-comparison process. In such a process, each model edge pixel that matches a scene one votes in favor of the hypothesis; otherwise, it votes against it. We define a hypothesis-selection criterion, ρ, that depends on two factors:

1. Size of the visible-part of the object (i.e., number of matched pixels) relative to the total object size (*visibility_ratio*).

2. Size of the visible-part relative to the largest visible-part of all object hypotheses (*size_ratio*).

Thus, the criterion favors less occluded and larger object hypotheses. It is easy to see that hypothesis validity is directly proportional to ρ, which lies in the range $[0, 1]$. Notice that $(1 - visibility\_ratio)$ can be interpreted as the average edge distance, if, as mentioned in the previous section, the dilated edge image is viewed as a distance image.

5.2.2 Scene and Model Uncertainties

Assume that we have a scene edge point, $\mathbf{e}^s$, that is the projection of some model edge point, $\mathbf{e}^m$. Ideally, we should have

$$\mathbf{e}^s = \text{proj}(\mathbf{e}^m) \tag{5.1}$$

where $\text{proj}(\mathbf{e}) = \begin{pmatrix} f\frac{e_x}{e_z} & f\frac{e_y}{e_z} \end{pmatrix}^t$ and f is the camera focal length. Practically, however, we only have estimates of $\mathbf{e}^s$ and $\mathbf{e}^m$, which we will denote by $\bar{\mathbf{e}}^s$ and $\bar{\mathbf{e}}^m$, respectively. The discrepancy between $\bar{\mathbf{e}}^s$ and $\mathbf{e}^s$ is referred to as *scene uncertainty*, while that between $\text{proj}(\bar{\mathbf{e}}^m)$ and $\text{proj}(\mathbf{e}^m)$ is referred to as *model uncertainty*. There are several factors that contribute to scene uncertainty such as image quantization, and imperfections of the edge-extraction process. Approaches for modeling and analyzing scene uncertainty can be found in [9, 47, 52]. Model uncertainty, a main focus of this chapter, is due to the error in object-pose estimation. The relationship between $\bar{\mathbf{e}}^m$ and $\mathbf{e}^m$ can be expressed as

$$\bar{\mathbf{e}}^m = \mathbf{E}\mathbf{e}^m \tag{5.2}$$

where $\mathbf{E}$ is a transformation matrix that represents the error in estimating the object pose. Assuming small rotational error, we can represent $\mathbf{E}$ as follows (see [63]):

$$\mathbf{E} = \begin{pmatrix} 1 & -\delta\theta_z & \delta\theta_y & \delta t_x d \\ \delta\theta_z & 1 & -\delta\theta_x & \delta t_y d \\ -\delta\theta_y & \delta\theta_x & 1 & \delta t_z d \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

where $\delta\theta_w$ is the rotational error about w axis, δt_w is the scale of the translational error along w axis ($w = x, y, z$), and d is the object depth (translation of the object center along the optical axis). The translational error is chosen to be linearly proportional to the object depth, since this is the general case in vision-based hypothesis generation (e.g., [49, 74]).

Assuming that the pose error is bounded, we can represent both scene and model uncertainties by *uncertainty regions*, $R^s(\bar{\mathbf{e}}^s)$ and $R^m(\bar{\mathbf{e}}^s)$, respectively, in the image plane. These regions provide spatial bounds on the locations of $\mathbf{e}^s$ and $\text{proj}(\bar{\mathbf{e}}^m)$, respectively, under some error assumptions. Formally,

$$\begin{aligned} R^s(\bar{\mathbf{e}}^s) &= \{\mathbf{e}^s : \mathbf{e}^s \text{ is a possible location of } \bar{\mathbf{e}}^s\} \\ R^m(\bar{\mathbf{e}}^s) &= \{\text{proj}(\bar{\mathbf{e}}^m) : \bar{\mathbf{e}}^m \in \mathbf{E}\mathbf{e}^m \text{ where } \text{proj}(\mathbf{e}^m) = \bar{\mathbf{e}}^s\}. \end{aligned}$$

Our objective is to dilate the scene edge image so as to accommodate both scene and model uncertainties. This can be done by replacing each scene edge pixel $\bar{\mathbf{e}}^s$ by region $R(\bar{\mathbf{e}}^s)$. This region is defined as

$$R(\bar{\mathbf{e}}^s) = R^m(\bar{\mathbf{e}}^s) \oplus R^s(\bar{\mathbf{e}}^s) \tag{5.4}$$

where $A \oplus B$ is the *dilation* of region A by region B (see [37])[2].

5.2.3 Determination of Model Uncertainty

In this section, we present an analytical method to determine the model uncertainty region $R^m(\bar{\mathbf{e}}^s)$. The method starts by computing $V^m(\mathbf{e}^m)$, a spatial bound on the 3-D location of $\bar{\mathbf{e}}^m$. Region $R^m(\bar{\mathbf{e}}^s)$ is then determined by simply projecting $V^m(\mathbf{e}^m)$ onto the image plane.

Assuming that the variation in the object depth is small relative to the actual depth, we can reasonably approximate d by e_z^m in the expression for E, defined in (5.3). Having done so, we get the following error equation from (5.2) and (5.3):

$$\delta\mathbf{e}^m = \mathbf{A}\delta\mathbf{p} \tag{5.5}$$

where

$$\begin{aligned}
\delta\mathbf{e}^m &= \bar{\mathbf{e}}^m - \mathbf{e}^m \\
\delta\mathbf{p} &= \begin{pmatrix} \delta\theta \\ \delta\mathbf{t} \end{pmatrix} \\
\mathbf{A} &= \begin{pmatrix} 0 & e_z^m & -e_y^m & e_z^m & 0 & 0 \\ -e_z^m & 0 & e_x^m & 0 & e_z^m & 0 \\ e_y^m & -e_x^m & 0 & 0 & 0 & e_z^m \end{pmatrix}.
\end{aligned} \tag{5.6}$$

The pose-error vector, $\delta\mathbf{p}$, is assumed to be bounded within a 6-D ellipsoid. This ellipsoid is represented by the equation

$$\delta\mathbf{p}^t\Lambda_p^{-1}\delta\mathbf{p} = 1 \tag{5.7}$$

where Λ_p is the diagonal matrix:

$$\Lambda_p = \text{diag}((\delta\theta_x^{\max})^2, (\delta\theta_y^{\max})^2, (\delta\theta_z^{\max})^2, (\delta t_x^{\max})^2, (\delta t_y^{\max})^2, (\delta t_z^{\max})^2). \tag{5.8}$$

[2]There is a slight approximation in (5.4) because it implicitly assumes that the shape of $R^m(\bar{\mathbf{e}}^s)$ is the same for any point in $R^s(\bar{\mathbf{e}}^s)$. As will be shown later, this is not the case. However, we consider this assumption to be reasonable since $R^s(\bar{\mathbf{e}}^s)$ is expected to be very small.

The error-bound vectors $\delta\theta^{\max}$ and $\delta \mathbf{t}^{\max}$ can be determined through knowledge of the hypothesis-generation algorithm. In some cases, the hypothesis generator produces hypotheses with no predetermined error bound. In such cases, the error bound in (5.8) can be interpreted as a "threshold" that differentiates between a valid noisy hypothesis and an invalid one.

From (5.5) and (5.7), it can be shown (see [33]) that $\delta \mathbf{e}^m$ lies within a 3-D ellipsoid defined by

$$(\delta \mathbf{e}^m)^t \Lambda_e^{-1} \delta \mathbf{e}^m = 1 \tag{5.9}$$

where

$$\Lambda_e = \mathbf{A}\Lambda_p\mathbf{A}^t. \tag{5.10}$$

Thus, the equation of ellipsoid $V^m(\mathbf{e}^m)$, which bounds $\bar{\mathbf{e}}^m$, is

$$(\mathbf{x} - \mathbf{e}^m)^t \Lambda_e^{-1} (\mathbf{x} - \mathbf{e}^m) = 1. \tag{5.11}$$

Projecting $V^m(\mathbf{e}^m)$ onto the image plane, we get an ellipse, which defines the model uncertainty region $R^m(\bar{\mathbf{e}}^s)$. The boundary of $R^m(\bar{\mathbf{e}}^s)$ is the projection of those points on $V^m(\mathbf{e}^m)$, $\mathbf{x}$, whose normals, $2\Lambda_e^{-1}(\mathbf{x} - \mathbf{e}^m)$, are perpendicular to the corresponding line of sight. Thus, these points are determined by the following equation:

$$\mathbf{x}^t \Lambda_e^{-1} (\mathbf{x} - \mathbf{e}^m) = 0. \tag{5.12}$$

Subtracting (5.12) from (5.11), we get the following plane equation:

$$(\mathbf{e}^m)^t \Lambda_e^{-1} \mathbf{x} = (\mathbf{e}^m)^t \Lambda_e^{-1} \mathbf{e}^m - 1. \tag{5.13}$$

Thus, $R^m(\bar{\mathbf{e}}^s)$ is the projection of the 3-D ellipse defined by the intersection of the above plane with $V^m(\mathbf{e}^m)$. The equation of the conic, $K^m(\mathbf{e}^m)$, that passes through such an ellipse is determined as follows. Squaring both sides of (5.13), we get

$$\mathbf{x}^t(\alpha \Lambda_e^{-1} \mathbf{e}^m (\mathbf{e}^m)^t \Lambda_e^{-1} \alpha)\mathbf{x} = 1 \tag{5.14}$$

where $\alpha = \frac{1}{(\mathbf{e}^m)^t \Lambda_e^{-1} \mathbf{e}^m - 1}$. Substituting from (5.13) into (5.11) and rearranging, we get

$$\mathbf{x}^t(\alpha \Lambda_e^{-1})\mathbf{x} = 1. \tag{5.15}$$

Subtracting (5.14) from (5.15), we get the equation of $K^m(\mathbf{e}^m)$:

$$\mathbf{x}^t \Lambda_k^{-1} \mathbf{x} = 0 \tag{5.16}$$

where

$$\Lambda_k^{-1} = \alpha \Lambda_e^{-1}(\mathbf{I} - \mathbf{e}^m(\mathbf{e}^m)^t \Lambda_e^{-1} \alpha). \tag{5.17}$$

The 2-D equation of ellipse $R^m(\bar{\mathbf{e}}^s)$ in the image plane is then obtained by substituting $\mathbf{x} = (\ x\ \ y\ \ f\ )^t$ in (5.16), to get a 2-D equation of the following form:

$$(\mathbf{x} - \mathbf{c})^t \mathbf{\Lambda}_{\mathbf{c}}^{-1} (\mathbf{x} - \mathbf{c}) = 1 \tag{5.18}$$

where $\mathbf{c}$ is the center of $R^m(\bar{\mathbf{e}}^s)$ in the image plane, and $\mathbf{\Lambda}_{\mathbf{c}}^{-1}$ is a 2×2 symmetric matrix. Note that $R^m(\bar{\mathbf{e}}^s)$ is *independent* of the length of $\mathbf{e}^m$, $\| \mathbf{e}^m \|$, which is unknown *a priori*. This fact can be proved by observing the following:

1. Matrix $\mathbf{A}$ in (5.6) can be represented as the product of $\frac{1}{\|\mathbf{e}^m\|}$ and a matrix whose elements are dependent on only the *direction* of $\mathbf{e}^m$. Accordingly, by observing (5.10) and (5.17), we can see that each of the matrices Λ_e^{-1} and Λ_k^{-1} is the product of $\frac{1}{\|\mathbf{e}^m\|^2}$ and a matrix that is independent of $\| \mathbf{e}^m \|$.

2. Equation of conic $K^m(\mathbf{e}^m)$, (5.16), is independent of the scale of Λ_k^{-1} since it is homogeneous. Accordingly, the equation of ellipse $R^m(\bar{\mathbf{e}}^s)$, (5.18), is also independent of the scale of Λ_k^{-1}.

From the above two observations, we can conclude that $R^m(\bar{\mathbf{e}}^s)$ depends on only the direction of $\mathbf{e}^m$, which is the same as that of $\bar{\mathbf{e}}^s$, and so it can be determined *off-line*.

5.2.4 Practical Considerations

It is quite complex to implement $R^m(\bar{\mathbf{e}}^s)$ as ellipses of various sizes and orientations, especially because, in general, the center of ellipse $R^m(\bar{\mathbf{e}}^s)$ does not coincide with $\bar{\mathbf{e}}^s$. For this reason, we approximate $R^m(\bar{\mathbf{e}}^s)$ by the smallest circle, $C^m(\bar{\mathbf{e}}^s)$, that is centered at $\bar{\mathbf{e}}^s$ and covers $R^m(\bar{\mathbf{e}}^s)$ (see Fig. 5.2). Define the following:

$$\begin{aligned} \delta_w &= \text{resolution of the image in the } w \text{ direction } (w = x, y) \text{, and} \\ \mathbf{\Delta} &= \text{diag}(\delta_x, \delta_y). \end{aligned}$$

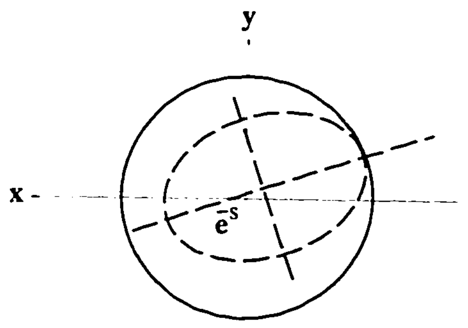


Figure 5.2: Approximation of $R^m(\bar{\mathbf{e}}^s)$ by a circle.

The equation of $R^m(\bar{\mathbf{e}}^s)$ with respect to the image coordinates can be expressed as follows:

$$(\mathbf{x}-\mathbf{d})^t \Lambda_{\mathbf{d}}^{-1}(\mathbf{x}-\mathbf{d}) = 1 \tag{5.19}$$

where $\Lambda_{\mathbf{d}}^{-1} = \mathbf{\Delta}^t \Lambda_{\mathbf{c}}^{-1} \mathbf{\Delta}$ and $\mathbf{d} = \mathbf{\Delta}^{-1}\mathbf{c}$. Without loss of generality, we can assume that $\Lambda_{\mathbf{d}} = \mathrm{diag}(a^2, b^2)$, and $\mathbf{d} = \mathbf{0}$. A point on such an ellipse can be parametrized by θ as

$$\mathbf{x}(\theta) = \begin{pmatrix} a\cos(\theta) \\ b\sin(\theta) \end{pmatrix}. \tag{5.20}$$

The radius of $C^m(\bar{\mathbf{e}}^s)$ is obtained by maximizing the distance between $\mathbf{x}(\theta)$ and $\bar{\mathbf{e}}^s$. That is,

$$F(\theta) = (\mathbf{x}(\theta) - \bar{\mathbf{e}}^s)^t(\mathbf{x}(\theta) - \bar{\mathbf{e}}^s). \tag{5.21}$$

Substituting from (5.20) in (5.21) and differentiating $F(\theta)$, we get

$$\frac{dF(\theta)}{d\theta} = a\bar{e}_x^s \sin(\theta) + (b^2 - a^2)\sin(\theta)\cos(\theta) - b\bar{e}_y^s\cos(\theta). \tag{5.22}$$

Functions $\sin(\theta)$ and $\cos(\theta)$ can be represented as follows:

$$\sin(\theta) = \frac{2t}{1+t^2}, \qquad \cos(\theta) = \frac{1-t^2}{1+t^2} \tag{5.23}$$

where $t = \tan(\frac{\theta}{2})$. Substituting from (5.23) in (5.22) and setting $\frac{dF(\theta)}{d\theta} = 0$, we get the following quartic equation:

$$(\bar{e}_y^s b)t^4 + 2(\bar{e}_x^s a - b^2 + a^2)t^3 + 2(\bar{e}_x^s a + b^2 - a^2)t - \bar{e}_y^s b = 0. \tag{5.24}$$

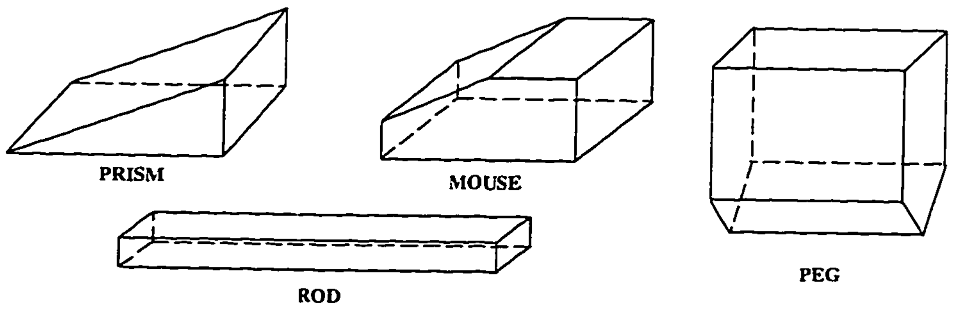


Figure 5.3: The chosen model database.

Solving (5.24), we get four values of θ. Substituting those values in (5.21), we get the radius of $C^m(\bar{e}^s)$:

$$r^m(\bar{e}^s) = \lceil \max_\theta \{\sqrt{F(\theta)}\} \rceil. \tag{5.25}$$

The above process can be computed off-line to construct a *radius image*. In this image, the value of each pixel $\bar{e}^s$ is

$$r(\bar{e}^s) = r^m(\bar{e}^s) + r^s(\bar{e}^s) \tag{5.26}$$

where $r^s(\bar{e}^s)$ is the uncertainty radius of $R^s(\bar{e}^s)$ (refer to (5.4)). At run time, each pixel $\bar{e}^s$ of the scene edge image is examined. If it is an edge pixel, then it gets replaced by a circle centered at $\bar{e}^s$ and of radius $r(\bar{e}^s)$.

5.3 Experimental Results

In this section, we present a couple of experiments using real images to demonstrate the effectiveness of the proposed verification technique. An NEC CCD camera is used in the experiments. This camera provides images of dimensions 480×512, with resolution 0.01 mm $\times$ 0.01255 mm in the x and y directions, respectively. The focal length of the camera is 16 mm.

Edges are extracted from an input image as follows. First, the image is smoothed by applying Gaussian operator of size 7×7. Edges are then detected by applying Sobel operators [37]. Pixels of gradient magnitude greater than some threshold are considered edges. Using such a technique, an actual edge is detected as a "thick" band of edges along the actual one. Since scene uncertainty is not formally analyzed

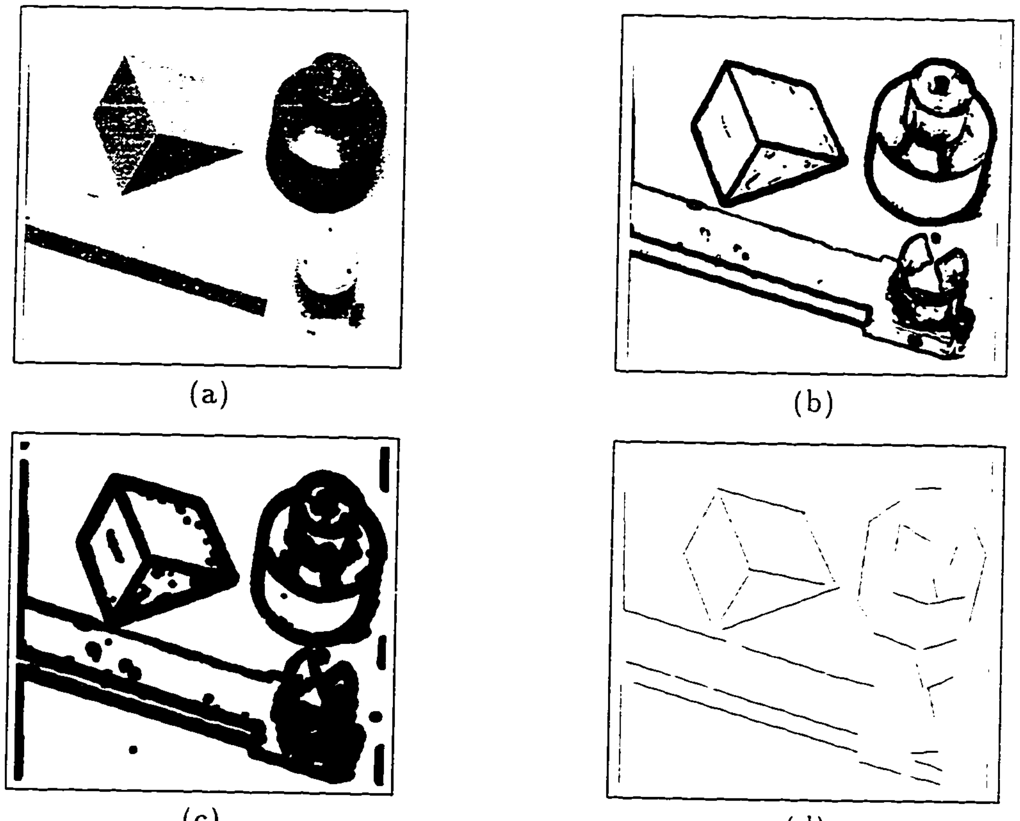

(a) (b) (c) (d)

Figure 5.4: First scene: (a) scene, (b) edge image, (c) dilated edge image, (d) extracted straight edges.

in this work, we have assumed that the resulting thickened edge image accommodates scene uncertainty. Accordingly, $r^s(\bar{e}^s)$ is set to 0, when dilating the thickened edge image (refer to (5.26)). This would not have been the case, had we chosen another edge-detection technique such as zero-crossings, for example.

For the experimental results presented in this section, the model error parameters are empirically chosen as follows: $\delta\theta^{\max} = \begin{pmatrix} 0.1^\circ & 0.1^\circ & 0.05^\circ \end{pmatrix}^t$ and $\delta\mathbf{t}^{\max} = \begin{pmatrix} 0.00125 & 0.00125 & 0.02 \end{pmatrix}^t$. For those parameters, $r^m(\bar{e}^s)$ ranges from four pixels, at the image center, to eight pixels, at any of the four image corners. The technique used in vision-based hypothesis generation is the same as the one described in Section 3.4.2. Finally, we have chosen a model database consisting of four simple polyhedral objects: PRISM, ROD, MOUSE and PEG (see Fig. 5.3).

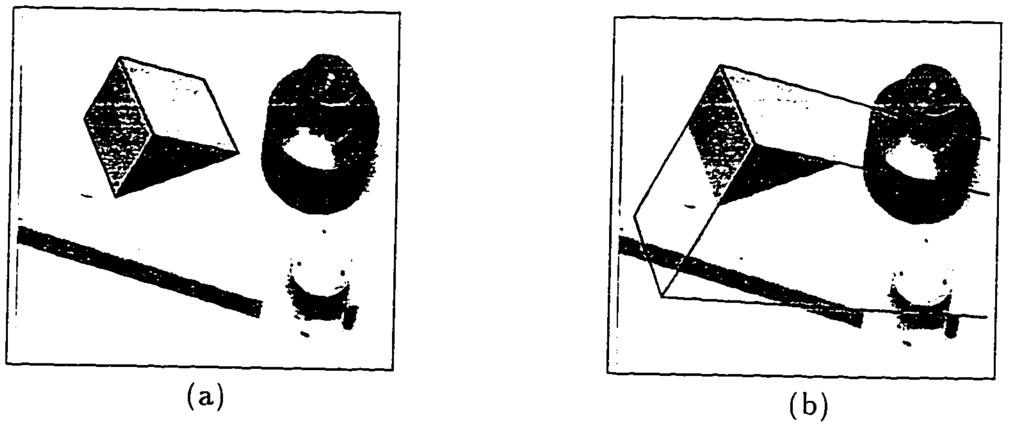


Figure 5.5: First scene: (a) valid hypothesis, (b) selected hypothesis when ρ = *size_ratio*.

5.3.1 An Unoccluded-Object Scene

Our first scene includes an unoccluded model object. PRISM (see Fig. 5.4(a)). The corresponding edge and dilated-edge images are shown in Figs. 5.4(b) and 5.4(c). respectively. The extracted straight edges are shown in Fig. 5.4(d). Five three-edge junctions are extracted, four of which belong to PRISM (see Fig. 5.4(d)). Out of the 15 junction edges, 12 are considered fully visible. Thus, we have 12 J-features for hypothesis generation, all of which are found to belong to PRISM. Since it is difficult, in general, to decide which junction belongs to the scene object of interest, we match all J-features with corresponding model ones. The generated hypotheses are then passed to the pixel-based verifier. Histograms of the hypothesis-selection criterion ρ for all model objects are shown in Table 5.1. From the results obtained, we have found that the 11 hypotheses in the highest non-empty slot, $\rho \in [0.7, 0.8]$, are valid, one of which is shown in Fig. 5.5(a). Thus, in this case, the verification algorithm successfully determines the valid hypotheses. The remaining valid one (notice that we have 12 valid J-features of scene object PRISM) is not as accurate as the others, and so it is mapped to the next slot, $\rho \in [0.6, 0.7]$. Notice that, although the scene object is not occluded, ρ's corresponding to the valid hypotheses are considerably less than 1. As will be seen below, this is because of the existence of an erroneous hypothesis with more raw number of votes, *hit_count*, than the valid hypotheses, thus reducing their size ratio, *size_ratio*, and accordingly ρ (refer to Section 5.2.1).

To demonstrate the importance of using both visibility and size ratios in the hypothesis-selection criterion ρ, we have run the experiment twice considering only one factor at a time. It has been found that the criterion $\rho = visibility\_ratio$ selects the right hypotheses, whereas $\rho = size\_ratio$ chooses an erroneous one which is shown in Fig. 5.5(b). This demonstrates the insufficiency of using the size ratio as the only selection criterion.

Object	0	1	2	3	4	5	6	7	8	9	Total
ROD	168	144	92	28	0	0	0	0	0	0	432
PRISM	41	96	99	43	18	3	1	11	0	0	312
MOUSE	96	151	131	53	23	2	0	0	0	0	456
PEG	100	144	105	37	22	0	0	0	0	0	408

Table 5.1: Histograms of ρ corresponding to each model object (first scene). Column i corresponds to $\rho \in [0.1i, 0.1(i+1)]$.

5.3.2 An Occluded-Object Scene

We have also tested the performance of the system under occlusion. Fig. 5.6(a) shows a scene of an occluded object, ROD. The corresponding edge image, dilated edge image and extracted straight edges are shown in Figs. 5.6(b)-(d), respectively. Five three-edge junctions are extracted, only one of which belongs to object ROD. Out of the 15 junction edges, only four are considered fully visible. Thus, there are four J-features for hypothesis generation, two of which are found to belong to ROD. Histograms of ρ are shown in Table 5.2. Eight hypotheses are found in the highest non-empty slot, $\rho \in [0.8, 0.9]$, all of them are valid (notice that, due to the symmetry of ROD, each object pose is described by four equivalent hypotheses). Fig. 5.7(a) shows one of the valid hypotheses. This experiment demonstrates the robustness of the system under object occlusion.

As in the first experiment, we have run the system twice considering one selection factor, $size\_ratio$ and $visibility\_ratio$, at a time. Contrary to the first experiment, we have found that the criterion $\rho = size\_ratio$ selects the right hypotheses, whereas $\rho = visibility\_ratio$ chooses an erroneous one which is shown in Fig. 5.7(b). This demonstrates the insufficiency of using the visibility ratio as the only criterion. Furthermore, this observation along with the one obtained in the first experiment support

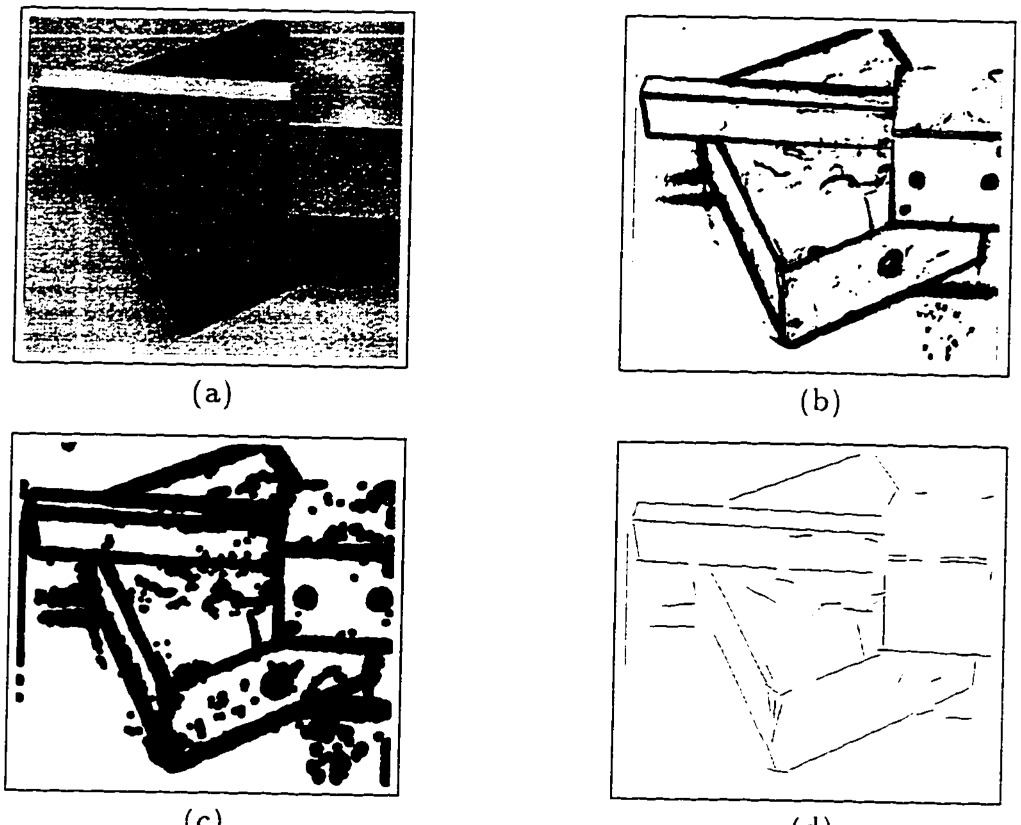


Figure 5.6: Second scene: (a) scene. (b) edge image. (c) dilated edge image. (d) extracted straight edges.

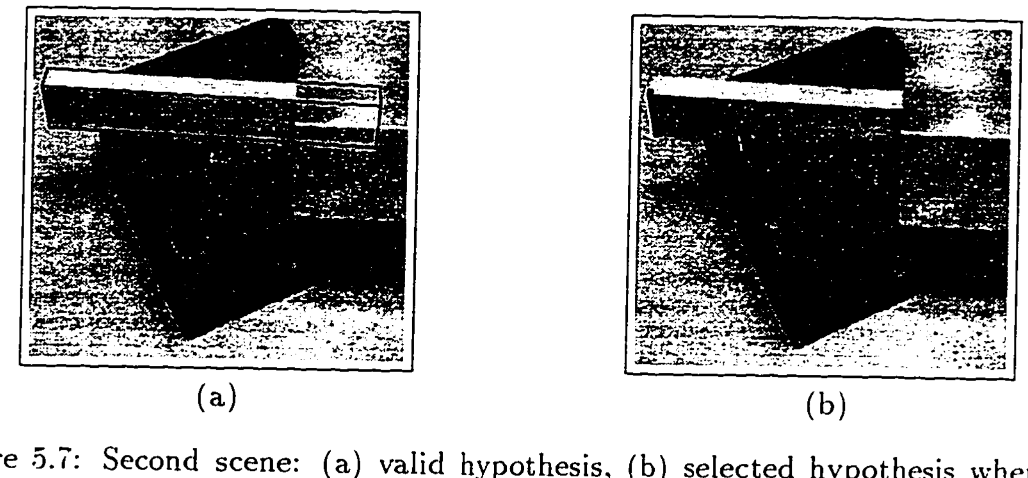


Figure 5.7: Second scene: (a) valid hypothesis, (b) selected hypothesis when $\rho = visibility\_ratio$.

the importance of combining both factors in the hypothesis-selection criterion.

Object	0	1	2	3	4	5	6	7	8	9	Total
ROD	36	28	12	4	0	8	0	0	8	0	96
PRISM	6	25	13	0	0	0	0	0	0	0	44
MOUSE	16	65	25	5	1	0	0	0	0	0	112
PEG	12	60	14	8	0	2	0	0	0	0	96

Table 5.2: Histograms of ρ corresponding to each model object (second scene). Column i corresponds to $\rho \in [0.1i, 0.1(i+1)]$.

It should be pointed out that it is not guaranteed that the valid hypotheses will always be found in the highest non-empty histogram slot (as in the presented two scenes). A further verification step might be needed to verify those hypotheses in the highest slot. One possibility is applying a feature-based verification technique. which utilizes the perceptual structures of the scene data. In case of failure, the same process should be applied to the hypotheses in the next highest slot, and so on. This situation can occur, for example, if the object of interest is heavily occluded.

5.4 Conclusions

A novel pixel-based technique has been presented for visually verifying 3-D object hypotheses. The visible-edge image of each hypothesized object is synthesized and compared, pixel wise, with the scene edge image. Hypotheses are then verified based on a criterion that favors less occluded and larger objects. Uncertainties in estimating the locations of both scene and model edges are handled by dilating the scene edge image. An analytical method is presented for determining the extent of dilation, assuming some error bound on the object pose.

In contrast to the common feature-based approaches, the proposed technique is very simple to implement, and is suitable for parallel implementation on graphics/vision hardware. Another important advantage is that the scene/model comparison step, being at the pixel level, is independent of both scene and model complexity (i.e., number and types of features). This makes the technique capable of handling arbitrarily complex scenes and model objects. Compared to existing pixel-based techniques, the proposed one has the advantage of being capable of handling uncertainties in both scene data and model pose.

An interesting point that is not covered in this chapter is performing experimental comparison between the proposed technique and similar pixel-based ones. This comparison. which is a subject of future work, is important to clearly demonstrate the performance of the proposed technique.

Chapter 6

Conclusions

In this chapter, we present the contributions of this dissertation in Section 6.1. In Section 6.2, we outline possible improvements to the developed work. Finally, directions for future work are presented in Section 6.3.

6.1 Contributions

The main thrust of this dissertation is the integration of 2-D and 3-D sensory data in the context of 3DOR. A vision/touch-based 3DOR system has been developed. This system consists of the following stages:

1. Vision/Touch-Based Screening: We have developed a screening technique that efficiently utilizes all the relevant visual and tactile data. This is achieved by employing a CSP framework which enables the integration of different types of sensory data. A novel method is presented for efficiently constructing a CSP corresponding to a combination of visual and tactile features. Efficiency is achieved by developing a number of approximate visual/tactile constraints, which enable the construction of visual/tactile binary constraints using very few comparison operations per constraint entry. The consistency of a model object with scene data is determined by first constructing the corresponding CSP, and then enforcing arc consistency, a low-order polynomial time process. We have shown that this process has a couple of advantages. Firstly, in case of arc-consistency failure, it can eliminate many erroneous model objects efficiently without either generating or verifying any pose hypotheses. Secondly, in case of

success. it can dramatically reduce the scene/model correspondence uncertainty. thus greatly reducing the computational load on subsequent recognition stages.

2. Vision/Touch-Based Hypothesis Generation: A novel technique is developed for generating pose hypotheses using a combination of visual and tactile features. in particular, a visual junction and a tactile polygon. The technique first determines a subset of the object DOF's from the tactile feature. The remaining DOF's, which cannot be determined from the tactile feature, are then obtained from the visual one. A couple of filtering techniques are developed to reduce the number of model feature sets that are actually compared with a given scene set. The proposed hypothesis-generation technique is essential in situations when visual and tactile data, individually, do not provide sufficient pose information. Experimentally, we have demonstrated its superiority over vision-based techniques in the following aspects: capability of determining the object pose under heavy occlusion, generation of a smaller number of pose hypotheses, and accuracy of estimating the object depth.

3. Touch-Based Hypothesis Verification: A highly-discriminative indexing scheme has been presented for touch-based data-driven verification. Selecting a tactile-feature frame as the reference frame, we have been able to obtain a *tight* and *invariant* set of constraints on object poses that are supported by the tactile feature. These constraints are indexed using a multi-level index. At run-time, pose parameters of the hypothesis to be verified are used as keys to traverse the index. The hypothesis is supported by the tactile feature only if the traversal is successful. Incorporating almost all the constraints provided a tactile feature, the proposed index has a higher discriminative power than similar indexing schemes. Experimentally, we have demonstrated its capability of eliminating the majority of inconsistent hypotheses in a constant time that is independent of the number of model features.

4. Vision-Based Hypothesis Verification: A pixel-based technique has been developed for visual hypothesis verification. The visible-edge image of a hypothesized object is synthesized and compared, pixel wise, with the scene edge image.

Based on this comparison, hypotheses are evaluated using a criterion that favors less occluded and larger objects. Uncertainties in estimating the locations of both scene and model edges are handled by dilating the scene edge image. An analytical method is presented for determining the extent of dilation, assuming some error bound on the object pose. The main advantage of this technique over similar pixel-based ones is its capability of accommodating the uncertainty of object-pose estimation.

6.2 Possible Improvements

The presented system can be improved in a number of ways.

1. Vision/Touch-Based Screening: The screening technique is totally dependent on the reliability of the perceptual-organization process, that of segmenting the visual features into structures that are perceived to belong to the same object. One way of improving this reliability is to utilize the tactile data, which are known to belong to the object of interest. This process, which can be called "touch-guided perceptual organization", would eliminate every visual feature that can not be consistent with *all* tactile features. Passing only the consistent visual features to the original perceptual-organization process would, hopefully, yield a more reliable output.

2. Vision/Touch-Based Hypothesis Generation: The filtering techniques used to reduce the number of scene/model matches are *ad hoc*. Although they are effective, more rigorous approaches are needed. A possible approach is to index model sets using vision/touch-based viewpoint-invariant attributes. Invariants are abound for 3-D feature sets, and scarce for 2-D sets. In fact, it is known that invariants do not exist for general 2-D feature sets [20], but only for special ones (e.g., four coplanar lines intersecting at a point [55], a pair of coplanar conics [35]). An interesting research problem is investigating whether invariants exist for sets that are composed of 2-D and 3-D features, as in our case.

3. Touch-Based Hypothesis Verification: Uncertainty in the touch-based indexing scheme is handled by dilating the pose constraints (filters). In the developed

work. the extent of dilation is determined empirically. A more formal approach is needed to determine the extent of dilation considering the two sources of uncertainty involved in the process: the tactile feature (scene uncertainty), and the estimated object pose (model uncertainty). A more robust approach would be to dilate the pose constraints considering only scene uncertainty. Model uncertainty can then be handled using the pose parameters along with their uncertainty parameters as keys into a more sophisticated index. This would give the index a *dynamic* discriminative power depending on the amount of uncertainty in the hypothesis to be verified.

4. Vision-Based Hypothesis Verification: The discriminative power of the vision-based verification scheme can be improved as follows. Rather than using a bounded-error model. we can use a probabilistic one by assigning probability distributions to the uncertainties of both scene and model edge pixels. In such a scheme. the value of each pixel in the scene edge image would be the probability that it corresponds to an edge. Another improvement to the matching process can be achieved by incorporating the difference between the estimated direction of the scene edge pixel and that of the corresponding model one. Obviously. these improvements would come at the expense of increasing the implementation complexity.

6.3 Directions for Future Work

First. let us analyze the 3DOR architecture adopted in this work, that of screening candidate model objects followed by generating and verifying object/pose hypotheses. As mentioned. this screening-based architecture has the advantage of utilizing the scene perceptual structures, early in the recognition process, in order to reduce the number of scene/model matches. However. it has the important limitation of being heavily dependent on the reliability of the perceptual-organization process. Reliable extraction of perceptual structures can be extremely difficult depending on the extent of noise and clutter in the scene data. Furthermore, since model objects are considered in turn. the best-case time complexity is linearly proportional to the number of model objects. This can be a problem for large model databases.

Next, let us consider the effect of eliminating the screening stage, and, instead, employing an indexing scheme for fast retrieval of model feature sets that are likely to be consistent with a particular local scene set (e.g., [7, 21, 45, 59, 70]). Paradoxically, this indexing-based architecture would not only considerably reduce the dependency on the validity of the perceptual structures, but also improve the best-case time complexity. This best case would take place in the following scenario:

- A selected set of local scene features belongs to the object of interest.
- The scene set is inconsistent with all model feature sets except for the valid one.
- Indexing into the model database, using viewpoint-invariant attributes, retrieves only the valid model set.
- Matching scene and model sets generates a hypothesis that is subsequently validated by the hypothesis verifier.

It can be easily seen that, thanks to indexing, an object can be recognized in a time that is *independent* of the model-database size. However, the fact that local features are used in indexing can make the actual complexity much worse than that of the screening-based architecture. This is especially true if we have a large number of model objects with similar local features.

An interesting research problem is to develop a 3DOR architecture that combines the advantages of the previous ones. In particular, it is interesting to develop an architecture that utilizes both the information provided by a perceptual structure (as in the screening-based one) and the power of indexing (as in the indexing-based one). This can be achieved as follows. Given a perceptual structure, we can extract a set of attribute values whose number is proportional to the structure size. Ordering those values according to some perceptual rule (e.g., adjacency), we can use them as a key into a robust index that can accept *variable-length* keys. Such a capability can be achieved by constructing a multi-level index, similar to the one presented in Chapter 4, whose number of levels is equal to the maximum possible key length. Every entry in the index, internal or external, points to a number of model-feature sets whose attribute values are similar to those corresponding to the entry. This indexing scheme is a major derivation from most existing ones that can only accept keys of fixed length.

The proposed indexing scheme has several advantages. Firstly, the richness of a perceptual structure is utilized in order to make a more discriminative query, where the extent of discrimination depends on the amount of information provided by the structure. Secondly, erroneous perceptual structures are expected to be discovered very quickly, since indexing would likely retrieve no, or very few, model sets. Thirdly, the lower bound of the time complexity is no longer linearly dependent on the model-database size, as in the screening-based architecture. Finally, the average-case complexity can be expected to be very weakly dependent, or even independent, on the model-database size. This is a property that is not possessed by any 3DOR system to the best of my knowledge.

The obvious problem with the above scheme is the space complexity of the index, which appears to be combinatorial in the number of model-object features. One way of controlling this complexity is to stop expanding an index node when the number of model feature sets that are mapped to such a node becomes less than some threshold (such a threshold determines the upper bound of generated hypotheses). Thus, leaf nodes of the proposed index would be of variable depth, where shallower nodes correspond to rare model feature sets, while deeper ones correspond to common sets.

Another important issue is the choice of scene attributes. The most common attributes that are used in indexing are viewpoint-invariant ones. This type of attributes provides constraints which are not as tight as the object-rigidity constraint. Worse yet, as mentioned in the previous section, invariants do not exist for general 2-D feature sets. It is also possible that this is the case for a combination of 2-D and 3-D features. Direct incorporation of the rigidity constraint will overcome the above problems. This can be accomplished by dedicating the first few index levels to viewpoint-variant scene attributes that suffice to estimate the object pose with respect to a viewer-centered frame, $\mathcal{F}_V$. Given such an estimate, the next index levels can then freely incorporate both viewpoint-invariant as well as viewpoint-variant attributes, thus leading to an index of high discriminative power.

The obvious problem with the above approach is that the "pose-estimation" index levels would have to implicitly discretize the six object DOF's. This is prohibitively expensive in terms of space requirements. We can handle this problem by selecting a data-centered frame, $\mathcal{F}_{\mathcal{D}}$, as the reference frame. Such a selection can considerably reduce the DOF's that need to be discretized. This can be demonstrated by the following examples:

- It can be shown that an $\mathcal{F}_{\mathcal{D}}$ that is defined with respect to (w.r.t.) a tactile patch and a visual junction constrains four DOF's. An additional DOF is required in order to determine the location of $\mathcal{F}_{\mathcal{D}}$ w.r.t. $\mathcal{F}_{\mathcal{V}}$. Accordingly, only three DOF's would need to be discretized.

- In case of range data, it is easy to find an $\mathcal{F}_{\mathcal{D}}$ that constrains all DOF's of the scene object (e.g., a frame that is defined w.r.t. three non-parallel surfaces). However, two DOF's will still be required to determine the orientation of $\mathcal{F}_{\mathcal{D}}$ w.r.t. $\mathcal{F}_{\mathcal{V}}$; those are the only ones that need to be discretized.

- In case of visual data, an $\mathcal{F}_{\mathcal{D}}$ that is defined w.r.t. a visual junction constrains three DOF's. Selecting the origin of $\mathcal{F}_{\mathcal{D}}$ to be the same as that of $\mathcal{F}_{\mathcal{V}}$, and considering that feature visibility is not affected by the rotation of $\mathcal{F}_{\mathcal{V}}$ [48], we can assume that $\mathcal{F}_{\mathcal{D}}$ is the same as $\mathcal{F}_{\mathcal{V}}$. Thus, we would have to discretize only three DOF's.

Bibliography

[1] P. K. Allen. Integrating vision and touch for object recognition tasks. *Int. J. of Robotic Res.*. 7(6):15–33. 1988.

[2] P. K. Allen and P. Michelman. Acquisition and interpretation of 3-D sensor data from touch. *IEEE Trans. on Robotics and Automat.*, 6(4):397–404, 1990.

[3] F. Arman and J. K. Aggarwal. CAD-based object recognition in range images using pre-compiled strategy trees. In A. K. Jain and P.J. Flynn, editors, *Three-Dimensional Object Recognition Systems*, pages 115–134. Elsevier Science Publishers, 1993.

[4] F. Arman and J. K. Aggarwal. Model-based object recognition in dense-range images - A review. *ACM Comput. Surveys*, 25(1):5–43, 1993.

[5] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proc. fifth Int. Joint Conf. Artificial Intelligence*, pages 659–663, Cambridge, MA, 1977.

[6] A. Basu and J. Aloimonos. A robust, correspondence-less, translation-determining algorithm. *Int. J. of Robotic Res.*, 9(5):35–59, 1990.

[7] J. S. Beis and D. G. Lowe. Learning indexing functions for 3-D model-based object recognition. In *IEEE Conf. Comput. Vision Patt. Recogn.*, pages 275–280. Seattle, WA. 1994.

[8] R. Bergevin and M. D. Levine. Generic object recognition: Building and matching coarse descriptions from line drawings. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 15(1):19–36, 1993.

[9] V. Berzins. Accuracy of laplacian edge detectors. *Comput. Vision Graphics Image Process.*, 27(2):195–210, 1984.

[10] P. J. Besl and R. C. Jain. Three-dimensional object recognition. *ACM Comput. Surveys*, 17(1):75–145, 1985.

[11] B. Bhanu and C. Ho. CAD-based 3D object representation for robot vision. *IEEE Computer*, 20(8):19–35, 1987.

[12] R. C. Bolles and P. Horaud. 3DPO: A three-dimensional part orientation system. *Int. J. of Robotic Res.*, 5(3):3–26, 1986.

[13] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 10(6):849–865, 1988.

[14] M. Boshra and H. Zhang. Use of tactile sensors in enhancing the efficiency of vision-based object localization. In *Proc. IEEE Int. Conf. Multisensor Fusion and Integ. for Intell. Sys.*, pages 243–250, Las Vegas, Nevada, 1994.

[15] M. Boshra and H. Zhang. Use of visual and tactile data for generation of 3-D pose hypotheses. In *Proc. IEEE/RSJ/GI Int. Conf. Intelligent Robots and Systems*, pages 73–80, Munchen, Germany, 1994.

[16] M. Boshra and H. Zhang. A constraint-satisfaction approach for 3-D vision/touch-based object recognition. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 368–373, Pittsburgh, PA, 1995.

[17] M. Boshra and H. Zhang. An efficient pixel-based technique for visual verification of 3-D object hypotheses. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 3472–3477, Minneapolis, Minnesota, 1996.

[18] R. A. Brooks. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intell.*, 17:285–348, 1981.

[19] R. A. Browse and J. C. Rodger. Techniques for combining tactile and visual perception for robotics. In M.A. Goodale, editor, *Vision and Action: The Control of Grasping*, pages 295–320. Ablex Publishing Corporation, 1990.

[20] J. B. Burns, R. S. Weiss, and E. M. Riseman. View variation of point-set and line-segment features. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 15(1):51–68, 1993.

[21] A. Califano and R. Mohan. Multidimensional indexing for recognizing visual shapes. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 16(4):373–392, 1994.

[22] C. H. Chen and A. C. Kak. A robot vision system for recognizing 3-D objects in low-order polynomial time. *IEEE Trans. on Syst., Man and Cybern.*, 19(6):1535–1563, 1989.

[23] H. H. Chen. Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 13(6):530–541, 1991.

[24] R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. *ACM Comput. Surveys*, 18(1):67–108, 1986.

[25] R. Dechter. Enhancements schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intell.*, 41(3):273 –312, 1989/90.

[26] R. Dechter. From local to global consistency. *Artificial Intell.*, 55(1):87–107, 1992.

[27] M. Dhome and T. Kasvand. Polyhedra recognition by hypothesis accumulation. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 9(3):429–439, 1987.

[28] M. Dhome, M. Richetin, J. Lapreste, and G. Rives. Determination of the attitude of 3-D objects from a single perspective view. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 11(12):1265–1278, 1989.

[29] S. J. Dickinson, A. P. Pentland, and A. Rosenfeld. 3-D shape recovery using distributed aspect matching. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 14(2):174–198, 1992.

[30] R. E. Ellis. Planning tactile recognition paths in two and three dimensions. *Int. J. of Robotic Res.*, 11(2):87–111, 1992.

[31] T. Fan. G. Medioni. and R. Nevatia. Recognizing 3-D objects using surface descriptions. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 11(11):1140-1157, 1989.

[32] O. D. Faugeras and M. Hebert. The representation, recognition and locating of 3-D objects. *Int. J. of Robotic Res.*, 5(3):27–52, 1986.

[33] Olivier Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint.* The M.I.T. Press., 1993.

[34] P. Flynn and A.K. Jain. BONSAI: 3-D object recognition using constrained search. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 13(10):1066–1075, 1991.

[35] D. Forsyth. J. L. Mundy, A. Zisserman, C. Coelho. A. Heller, and C. Rothwell. Invariant descriptors for 3-D object recognition and pose. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 13(10):971–991, 1991.

[36] C. Goad. Special-purpose automatic programming for 3-D model-based vision. In *Proc. Image Understanding Workshop*, pages 94–104. 1983.

[37] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison Wesley, 1992.

[38] K. D. Gremban and K. Ikeuchi. Appearance-based vision and the automatic generation of object recognition programs. In A. K. Jain and P. J. Flynn, editors. *Three-Dimensional Object Recognition Systems*, pages 229–258. Elsevier Science Publishers, 1993.

[39] W. E. L. Grimson and T. Lozano-Perez. Model-based recognition and localization from sparse range or tactile data. *Int. J. of Robotic Res.*, 3(3):3–35, 1984.

[40] W. E. L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 9(4):469–482, 1987.

[41] W.E.L. Grimson and D. P. Huttenlocher. On the verification of hypothesized matches in model-based recognition. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 13(12):1201–1213, 1991.

[42] C. Hansen and T. C. Henderson. CAGD-based computer vision. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 11(11):1181–1193, 1989.

[43] B. K. P. Horn. Extended Gaussian images. *Proc. IEEE*, 72:1656–1678, 1984.

[44] S. A. Hutchinson and A. C. Kak. Multisensor strategies using Dempster-Shafer belief accumulation. In M.A. Abidi and R.C. Gonzalez, editors, *Data Fusion in Rob. and Mach. Intell.*, pages 165–209. Academic Press, 1992.

[45] D. W. Jacobs. The space requirements of indexing under perspective projections. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 18(3):330–333, 1996.

[46] R. Joshi and A. C. Sanderson. Shape matching from grasp using a minimal representation size criterion. In *IEEE Int. Conf. on Robotics and Automation*, pages 442–449, Atlanta, Georgia, 1993.

[47] R. Kakarala and A.O. Hero. On achievable accuracy in edge localization. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 14(7):777–781, 1992.

[48] K. Kanatani. Camera rotation invariance of image characteristics. *Comput. Vision Graphics Image Process.*, 39:328–354, 1987.

[49] K. Kanatani. Constraints on length and angle. *Comput. Vision Graphics Image Process.*, 41(1):28–42, 1988.

[50] S. B. Kang and K. Ikeuchi. The complex EGI: A new representation for 3-D pose determination. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 15(7):707–721, 1993.

[51] W. Kim and A. C. Kak. 3-D object recognition using bipartite matching embedded in discrete relaxation. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 13(3):224–251, 1991.

[52] J. V. Kittler. On the accuracy of the sobel edge detector. *Image and Vision Computing*. 1(1):37–42, 1983.

[53] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[54] S. Lee and H. Hahn. An optimal sensing strategy for recognition and localization of 3-D natural quadric objects. *IEEE Trans. on Pattern Anal. and Mach. Intell.*. 13(10):1018–1037. 1991.

[55] G. Lei. Recognition of planar objects in 3-D space from single perspective views using cross ratio. *IEEE Trans. on Robotics and Automat.*, 6(4):432–437, 1990.

[56] D. G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intell.*, 31(3):355–395, 1987.

[57] R. C. Luo and W. Tsai. Object recognition using tactile-array sensors. In *Proc. IEEE Int. Conf. Rob. and Aut.*, pages 1248–1253. San Francisco, CA. 1986.

[58] A. K. Mackworth. Consistency in networks of relations. *Artificial Intell.*, 8(1):99–118. 1977.

[59] J. Mao, P. J. Flynn, and A. K. Jain. Integration of multiple feature groups and multiple views into a 3D object recognition system. *Comput. Vision Image Understand.*, 62(3):309–325, 1995.

[60] J. Maver and R. Bajcsy. Occlusions as a guide for planning the next view. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 15(5):417–433, 1993.

[61] R. Mohr and T. C. Henderson. Arc and path consistency revisited. *Artificial Intell.*, 28:225–233, 1986.

[62] D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 16(5):449–459, 1994.

[63] R. P. Paul. *Robot Manipulators: Mathematics, Programming, and Control.* Cambridge, MA: M.I.T. Press., 1981.

[64] A. P. Reeves and R. W. Taylor. Identification of three-dimensional objects using range information. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 11(4):403–410, 1989.

[65] R. Safaee-Rad, I. Tchoukanov, K. C. Smith, and B. Benhabib. Three-dimensional location estimation of circular features for machine vision. *IEEE Trans. on Robotics and Automat.*, 8(5):624–640, 1992.

[66] R. J. Safranek, S. Gottschlich, and A. C. Kak. Evidence accumulation using binary frames of discernment for verification vision. *IEEE Trans. on Robotics and Automat.*, 6(4):405–417, 1990.

[67] J. L. Schneiter and T. B. Sheridan. An automated tactile sensing strategy for planar object recognition and localization. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 12(8):775–786, 1990.

[68] T. M. Sobh and R. Bajcsy. Autonomous observation under uncertainty. In *IEEE Int. Conf. on Robotics and Automation*, pages 1792–1798, Nice, France, 1992.

[69] S. A. Stansfield. A robotic perceptual system utilizing passive vision and active touch. *Int. J. of Robotic Res.*, 7(6):138–161, 1988.

[70] F. Stein and G. Medioni. Structural indexing: Efficient 3-D object recognition. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 14(2):125–145, 1992.

[71] S. Ullman. Analysis of visual motion by biological and computer systems. *IEEE Computer*, 14(8):57–69, 1981.

[72] P. Whaite and F. P. Ferrie. From uncertainty to visual exploration. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 13(10):1038–1049, 1991.

[73] A. K. C. Wong, S. W. Lu, and M. Rioux. Recognition and shape synthesis of 3-D objects based on attributed hypergraphs. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 11(3):279–290, 1989.

[74] K. C. Wong and J. Kittler. Recognizing polyhedral objects from a single perspective view. *Image and Vision Computing*, 11(4):211–220, 1993.

[75] Y. Wu, S. S. Iyengar, R. Jain, and S. Bose. A new generalized computational framework for finding object orientation using perspective trihedral angle constraint. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 16(10):961–975, 1994.

[76] S. Zhang, G. D. Sullivan, and K. D. Baker. The automatic construction of view-independent relational model for 3-D object recognition. *IEEE Trans. on Pattern Anal. and Mach. Intell*, 15(6):531–543, 1993.

Appendix A

Determination of Continuous Visual/Tactile Constraints

A.1 Cylinder $OOC(S_i^m)$

Let h_j be the distance between model vertex V_j^m and model surface S_i^m in the direction of $\mathbf{n}_i^m$: i.e.,

$$h_j = \mathbf{n}_i^m \mathbf{v}_j^m - d_i^m. \tag{A.1}$$

It can be shown that the dimensions of cylinder $OOC(S_i^m)$ are

$$[h_i^{min}, h_i^{max}] = (\min_j(h_j), \max_j(h_j)) \tag{A.2}$$
$$r_i = \max_{j,k}(\| (\mathbf{v}_j^m - h_j\mathbf{n}_i^m) - \mathbf{v}_k^m \|) \tag{A.3}$$

where $[h_i^{min}, h_i^{max}]$ are the cylinder's heights measured from the plane of S^s along $\mathbf{n}^s$. r_i is the cylinder's radius, and V_k^m is a vertex of surface S_i^m. Notice that the dimensions of the cylinder do not depend on the location of S^s, and so they can be computed off-line.

A.2 Edge-Angle Constraint $EAC(S_i^m, E_j^s)$

The equation of line of sight $L(V_j^s)$ can be represented as

$$L(V_j^s, \alpha) = \mathbf{c} + \alpha \mathrm{dir}(\mathbf{v}_j^s - \mathbf{c}) \tag{A.4}$$

where $\mathbf{c}$ is the camera viewpoint, $\mathrm{dir}(\mathbf{x}) = \frac{1}{\|\mathbf{x}\|}\mathbf{x}$, and α is the distance from $\mathbf{c}$ to $L(V_j^s, \alpha)$ in the direction of $\mathrm{dir}(\mathbf{v}_j^s - \mathbf{c})$.

It can be shown that the bounds of $EAC(S_i^m, E_j^s)$, $[\theta^{min}, \theta^{max}]$, are

$$\begin{aligned}
\theta^{min} &= \text{angle}(\mathbf{v}_{j1}^s - \mathbf{c}, L(V_{j2}^s, \alpha_2^{max}) - L(V_{j1}^s, \alpha_1^{min})) \\
\theta^{max} &= \text{angle}(\mathbf{v}_{j1}^s - \mathbf{c}, L(V_{j2}^s, \alpha_2^{min}) - L(V_{j1}^s, \alpha_1^{max})).
\end{aligned}$$

where $EDC(S_i^m, E_j^s) = ([\alpha_1^{min}, \alpha_1^{max}], [\alpha_2^{min}, \alpha_2^{max}])$, and angle($\mathbf{u}, \mathbf{v}$) is the angle between vectors $\mathbf{u}$ and $\mathbf{v}$.

A.3 Edge-Length Constraint $ELC(S_i^m, E_j^s)$

It can be shown that the bounds of $ELC(S_i^m, E_j^s)$, $[d^{min}, d^{max}]$, are

$$\begin{aligned}
d^{min} &= \min\Big(\bigcup_{u,v=min,max} \text{dis}(\alpha_1^u, \alpha_2^v), \text{dis}(\alpha_1^\perp, \alpha_2^{min}), \text{dis}(\alpha_1^{min}, \alpha_2^\perp)\Big) \\
d^{max} &= \max\Big(\bigcup_{u,v=min,max} \text{dis}(\alpha_1^u, \alpha_2^v)\Big)
\end{aligned}$$

where

- $EDC(S_i^m, E_j^s) = ([\alpha_1^{min}, \alpha_1^{max}], [\alpha_2^{min}, \alpha_2^{max}])$,
- $\text{dis}(\alpha_1, \alpha_2) = \| L(V_{j1}^s, \alpha_1) - L(V_{j2}^s, \alpha_2) \|$, and
- $L(V_{jk}^s, \alpha_k^\perp)$, $k = 1, 2$, is the perpendicular projection of $L(V_{j(3-k)}^s, \alpha_{3-k}^{min})$ on $L(V_{jk}^s)$.

Note that the term that contains $\alpha_k^\perp$ in the expression for d^{min} is only included if $\alpha_k^\perp \in [\alpha_k^{min}, \alpha_k^{max}]$.

Appendix B

Mathematical Definitions of Scene and Model Attributes

B.1 Attributes of Model Pair (S^m, E^m)

1. $L^m = \| \mathbf{v}_1^m - \mathbf{v}_2^m \|$, where $(\mathbf{v}_1^m, \mathbf{v}_2^m)$ are the end vertices of E^m

2. $A^m = \text{angle}(\mathbf{v}_2^m - \mathbf{v}_1^m, \mathbf{n}^m)$

3. $P_l^m = \mathbf{n}^m \mathbf{v}_l^m - d^m$

4. $DB_l^m = \{\| \mathbf{u} - \mathbf{v}_l^m \| : \mathbf{u} \in S^m\}$

B.2 Attributes of Scene Pair (S^s, E^s)

1. $LB^s(S^m) = ELC(S^m, E^s)$

2. $AB^s(S^m)$: The direction of E^m, $\mathbf{w}(\theta)$, can be represented by a rotation of vector

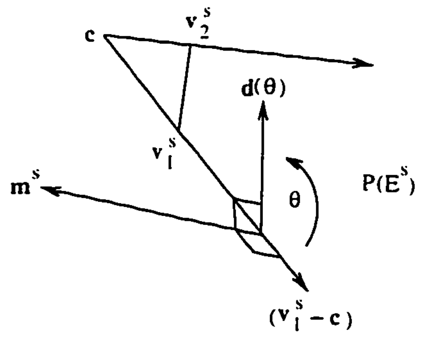


Figure B.1: Representation of vectors that lie inside $P(E^s)$.

$\mathrm{dir}(\mathbf{v}_1^s - \mathbf{c})$ about the normal of plane of sight $P(E^s)$, $\mathbf{m}^s$ (see Fig. B.1). That is,

$$\mathbf{w}(\theta) = \mathrm{rot}(\mathbf{m}^s, \theta)\mathrm{dir}(\mathbf{v}_1^s - \mathbf{c}) \tag{B.1}$$

where θ is the rotation angle, $\mathbf{m}^s = \mathrm{dir}((\mathbf{v}_1^s - \mathbf{c}) \times (\mathbf{v}_2^s - \mathbf{v}_1^s))$ and (V_1^s, V_2^s) are the end vertices of E^s. The rotation matrix, $\mathrm{rot}(\mathbf{m}, \theta)$, is defined as [63]

$$\mathrm{rot}(\mathbf{m}, \theta) = \begin{pmatrix} m_x m_x v\theta + c\theta & m_y m_x v\theta - m_z s\theta & m_z m_x v\theta + m_y s\theta \\ m_x m_y v\theta + m_z s\theta & m_y m_y v\theta + c\theta & m_z m_y v\theta - m_x s\theta \\ m_x m_z v\theta - m_y s\theta & m_y m_z v\theta + m_x s\theta & m_z m_z v\theta + c\theta \end{pmatrix} \tag{B.2}$$

where $c\theta = \cos(\theta)$, $s\theta = \sin(\theta)$ and $v\theta = 1 - c\theta$. In our case, θ is restricted to lie in the range $EAC(S^m, E^s)$. Thus, $AB^s(S^m)$, is defined as

$$AB^s(S^m) \;=\; \{\mathrm{angle}(\mathbf{w}(\theta), \mathbf{n}^s) : \theta \in EAC(S^m, E^s)\}.$$

3. $PB_l^s(S^m) = \{\mathbf{n}^s L(V_l^s, \alpha) - \mathbf{n}^s \mathbf{p}^s : \alpha \in VDC(S^m, V_l^s)\}$, where $L(V_l^s, \alpha)$ is defined in (A.4).

4. $DB_l^s(S^m) = \{\| \mathbf{p}^s - L(V_l^s, \alpha) \| : \alpha \in VDC(S^m, V_l^s)\}$

Appendix C

Determination of the Dimensions of Volume $OOC(F_i^m)$

C.1 Cylinder $OOC(S_i^m)$

It can be easily shown that the heights of cylinder $OOC(S_i^m)$ are the same as in the point-like surface patch case (see (A.2)). To compute the radius of cylinder $OOC(S_i^m)$, r_i, we approximate the tactile polygon by a circle inside it of radius t (e.g., a $2t \times 2t$ tactile square can be approximated by a circle of radius t). The radius of $OOC(S_i^m)$ is the maximum distance between the vertices of $\mathcal{O}$ and the z axis such that the tactile circle remains inside S_i^m. It can be shown that this maximum distance occurs when a vertex in U_i^m, the polygon obtained by "shrinking" S_i^m by t, translates to the center of the tactile circle. Thus,

$$r_i = \max_{j,k}(\| (\mathbf{v}_j^m - h_j \mathbf{n}_i^m) - \mathbf{u}_k^m \|)$$

where $\mathbf{u}_k^m \in U_i^m$ and h_j is as defined in (A.1).

C.2 Cuboid $OOC((S_j^m, E_i^m))$

It can be easily shown that the range of translations along the x axis that keep the tactile edge inside model edge E_i^m is

$$[-p_x - \frac{(l_i - l_T)}{2}, -p_x + \frac{(l_i - l_T)}{2}]$$

where l_T is the length of the tactile edge, l_i is the length of edge E_i^m, and $\mathbf{p}$ is the center of E_i^m after transforming it by ${}^i\mathbf{T}$ (notice that $p_y = p_z = 0$). Dimensions of cuboid $OOC((S_j^m, E_i^m))$ are determined as follows:

$$
\begin{aligned}
[d_{ix}^{min}, d_{ix}^{max}] &= [\min_k({}^i o_{kx}) - p_x - \frac{(l_i - l_T)}{2}, \max_k({}^i o_{kx}) - p_x + \frac{(l_i - l_T)}{2}] \\
[d_{iy}^{min}, d_{iy}^{max}] &= \text{minmax}_k({}^i o_{ky}) \\
[d_{iz}^{min}, d_{iz}^{max}] &= \text{minmax}_k({}^i o_{kz})
\end{aligned}
$$

where $[d_{i\omega}^{min}, d_{i\omega}^{max}]$ is the range of the cuboid along ω axis ($\omega = x, y, z$), and ${}^i\mathbf{o}_k \in {}^i\mathcal{O}$.

It can be seen that one can compute all the dimensions of $OOC((S_j^m, E_i^m))$ off-line except for $[d_{ix}^{min}, d_{ix}^{max}]$, since they depend on l_T that is only known at run-time. However, $[d_{ix}^{min}, d_{ix}^{max}]$ can be computed off-line assuming $l_T = 0$, and then updated on-line at a negligible cost by adding $\pm\frac{l_T}{2}$ to d_{ix}^{min} and d_{ix}^{max}, respectively.

Appendix D

Determination of the Bounds on angle($\mathbf{p}, \mathbf{w}_k(\theta)$)

Define

$$\begin{aligned}\beta(\theta) &= \text{angle}(\mathbf{p}, \mathbf{w}_k(\theta)) \\ &= \arccos(\mathbf{p} \cdot \mathbf{w}_k(\theta)) \qquad \text{(D.1)}\end{aligned}$$

where $\theta \in EAC(F_j^m, E_k^s)$, $\mathbf{p}$ is a unit vector, and $\mathbf{w}_k(\theta)$ is as defined in (B.1). Substituting from (B.1) and (B.2) into (D.1), we get

$$\beta(\theta) = \arccos(a \sin(\theta) + b\cos(\theta))$$

where $a = \mathbf{p}.(\mathbf{m} \times \mathbf{w})$ and $b = \mathbf{p} \cdot \mathbf{w}$.

It can be easily shown that the extreme point of $\beta(\theta)$ occurs at θ^e, where

$$\theta^e = \arctan \frac{a}{b}.$$

We choose the solution of θ^e that falls in the range $[0°, 180°]$. The bounds on $\beta(\theta)$ are

$$[\beta^{min}, \beta^{max}] = \begin{cases} \text{minmax}(\beta(\theta^e), \beta(\theta^{min}), \beta(\theta^{max})) & \text{if } \theta^e \in EAC(F_j^m, E_k^s) \\ \text{minmax}(\beta(\theta^{min}), \beta(\theta^{max})) & \text{otherwise} \end{cases}.$$

Notice that the value of $\tan(\theta^e)$ is undefined if both a and b are zero. This occurs when $\mathbf{p}$ is perpendicular to plane P_k. In this case, we have $\beta(\theta) \in [90°, 90°]$.