

To the Graduate Council:

I am submitting herewith a dissertation written by Richard Odell Eason entitled "Determining the Pose of Three-Dimensional Objects by Multisensor Fusion." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Electrical Engineering.

R. C. Gonzalez

Rafael C. Gonzalez, Major Professor

We have read this dissertation
and recommend its acceptance:

Donald W. Boulton

Mohan M. Trivedi

James M. Reche

Carl G. Wagner

Accepted for the Council:

Lew Mink

Vice Provost
and Dean of The Graduate School

**DETERMINING THE POSE OF THREE-DIMENSIONAL
OBJECTS BY MULTISENSOR FUSION**

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Richard Odell Eason

August 1988

Copyright © Richard Odell Eason, 1988
All rights reserved

DEDICATION

This dissertation is dedicated to my wife, Mary Anne, my two boys, Christopher and Parker, and to my parents, Mrs. Betty Eason and the late H. Odell Eason, Jr. To each I owe so much.

ACKNOWLEDGEMENTS

I wish to express my appreciation to my major advisor, Dr. R. C. Gonzalez, for his support, advice, and patience during the course of this research. I would also like to thank Dr. Donald W. Bouldin, Dr. Mohan M. Trivedi, Dr. J. M. Rochelle, and Dr. Carl G. Wagner for serving on my committee and reviewing this dissertation.

In the pursuit of this degree, I have also benefited along the way from the help of many fellow students, some now graduated. In particular, I wish to mention Dr. M. A. Abidi, Dr. Nicolas Alvertos, Joel Spears, Danny Newport, Mishi Saravi, Alan Wintenberg, Kevin Barnett, Ravindra V. Narkhede, Raj R. Kumar, P. R. Mukund, Hal Beck, and Brian Bernhard. Special thanks go to Joel Spears and Deborah Marsicek for proof-reading this dissertation. In addition, the help of several support staff, notably Janet Smith and Randy Mayhew, is also very much appreciated.

I am very grateful to the University of Tennessee for providing my education, along with many other opportunities, and I gratefully acknowledge the support for this research from the Department of Energy through contract DOE-DE-FG02-86NE37968.

Last, but certainly not least, I would like to thank my wife, Mary Anne, for her loving support and patient endurance through my many years of graduate study.

ABSTRACT

This research addresses the problem of determining the position and orientation (pose) of a known object, given partial pose information from different types of sensors. From our viewpoint each sensory measurement provides geometric information constraining the position and orientation of an object feature; e.g., if depth is unknown, then visual sensing of an object vertex reveals that the vertex lies somewhere on a certain line in space (we assume object features have already been matched to sensory data). Different sensors provide different forms of these constraints and also involve different measurement accuracies. The problem is to find a best guess of object pose when given such information from a variety of sensors.

The approach proposed in this research is of the Least Mean Square Error (LMSE) variety. It involves combining the parameters of each sensory measurement with the estimated errors in measurement to form a distance function which provides a measure of error for particular values of the measurement parameters. From the model of each type of sensing, we define a constraint relationship between the measurement parameters and the parameters describing the position and orientation of an object feature. These constraints and the distance function generated by the sensory measurement are used to generate a distance function of the object feature parameters. From this new distance function and the known object model, a third distance function, one of the pose parameters, is created. This local pose distance function describes an error as a function of the pose parameters based on a single sensory measurement. Finally, the global pose distance function is defined as the sum of the local pose distance functions. The minimum of this last function (found numerically) provides our best guess of object

pose. The advantage of this approach is in providing a consistent framework for the incorporation of noisy data from a variety of sensors.

The mathematics of this approach were developed for the sensing of point, line, and plane object features by a variety of sensing modalities. The method was implemented in software and tested using the robotics workstation at the University of Tennessee.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
1.1 Multiple Sensors	1
1.2 Sensing Object Pose	3
1.3 Background	6
1.4 An Approach to Multisensor Fusion	8
1.5 Outline of Remaining Chapters	12
2. TRANSFORMATION OF THE OBJECT MODEL	14
2.1 The Object Model	14
2.2 Transformed Object Features	16
2.3 The Coordinate Transformation	17
2.4 Chapter Summary	21
3. SENSORY FEATURES	22
3.1 A Classification of Sensory Data	22
3.2 Parameterization of Sensory Features	25
3.3 The Sensing Models	25
3.3.1 Point-Point Problems	26
3.3.2 Point-Line Problems	26
3.3.3 Point-Plane Problems	27
3.3.4 Line-Point Problems	27

CHAPTER	PAGE
3.3.5 Line-Line Problems	28
3.3.6 Line-Plane Problems	29
3.3.7 Plane-Point Problems	30
3.3.8 Plane-Line Problems	30
3.3.9 Plane-Plane Problems	31
3.3.10 Constraint Summary	32
3.4 Distance Functions for Sensory Features	32
3.4.1 Matrix Decomposition	34
3.4.2 Position Component of the Sensory Distance Function	35
3.4.3 Direction Component of the Sensory Distance Function	37
3.5 Chapter Summary	40
4. DETERMINING OBJECT POSE	41
4.1 Outline of the Method	41
4.2 Minimizing the Global Pose Distance Function	42
4.3 Evaluating Local Pose Functions	43
4.4 Evaluating Derivatives of Local Pose Functions	44
4.5 Object Feature Distance Functions	47
4.5.1 Point-Point Problems	47
4.5.2 Point-Line Problems	48
4.5.3 Point-Plane Problems	52
4.5.4 Line-Point Problems	58
4.5.5 Line-Line Problems	66
4.5.6 Line-Plane Problems	68
4.5.7 Plane-Point Problems	75

CHAPTER	PAGE
4.5.8 Plane-Line Problems	80
4.5.9 Plane-Plane Problems	87
4.6 Chapter Summary	89
5. IMPLEMENTATION AND EXPERIMENTAL RESULTS	90
5.1 Algorithm Implementation	90
5.2 Testbed for Experiments	92
5.2.1 Vision Sensing	96
5.2.2 Touch Sensing	100
5.2.3 Proximity Sensing	101
5.2.4 Range Sensing	102
5.3 Experimental Results	103
5.3.1 Testing the Individual Types of Sensing	104
5.3.2 Testing the Fusion of Multisensor Data	107
5.3.3 The Introduction of Bad data	109
5.3.4 Improving a Pose Estimate using Additional Measurements	111
5.4 A Graphics Display for Human Visualization	114
5.5 Chapter Summary	115
6. CONCLUSIONS AND FUTURE WORK	117
6.1 Potential Speed Improvements	118
6.2 Recommendations for Future Work	119
BIBLIOGRAPHY	121
VITA	127

LIST OF TABLES

TABLE	PAGE
5.1 The object model for the panel.	98
5.2 Summary of the types of sensing we perform and the descriptors for each.	105
5.3 Estimating Pose using a Single Type of Sensing.	105
5.4 Estimating pose using multiple types of sensing.	108
5.5 Estimating pose using multiple types of sensing: the case with bad data added.	110
5.6 Improving the pose estimate by including additional data.	112
5.7 Improving the pose estimate by including additional data given an initial bad measurement.	113

LIST OF FIGURES

FIGURE	PAGE
1.1 An object's pose.	4
2.1 (a) A simple object. (b) Parameters of some features.	16
2.2 Transformed object features of the plate.	18
3.1 The sensing relationships between sensory features and transformed ob- ject features.	23
3.2 (a) A simple object. (b) Point sensing of some object features. (c) Line sensing of some object features. (d) Plane sensing of some object features.	24
3.3 Constraint relationships between sensory features and object features.	33
5.1 The robotics workstation at the University of Tennessee.	93
5.2 The sensory package mounted on the end effector.	94
5.3 The object whose pose we are to determine.	97
5.4 Example line drawings of panel pose.	116

CHAPTER 1

INTRODUCTION

Throughout history man has strived to make tools, machines, and devices to extend his capabilities and improve his quality of life. Advancing technology has brought increasing sophistication to these devices and has endowed them with expanding capabilities. In the last few decades the addition of computers to these devices has given them the capability of autonomous or semi-autonomous behavior and has opened up new areas of application, but it has not in itself allowed this behavior in unconstrained environments. For this, sensors are also required.

1.1 Multiple Sensors

Sensors can provide many kinds of information about the environment, information which can be used by autonomous mechanisms in a variety of ways in the performance of a variety of tasks. The availability of *multiple* sensors is beneficial not only in providing a greater volume of information but also in increasing its breadth, thus allowing for response to many stimuli and ultimately enabling greater capabilities.

The broad base of information provided by multiple sensors is derived both directly, from the different data provided by different sensing modalities, and indirectly, from the synergistic combination of two or more pieces of data. Stereo vision, where 3-dimensional information is inferred from 2-dimensional images, is an example of the latter.

Increasing the breadth of information, however, is not the only benefit of multiple sensing; a *redundancy* of information can also be very important. Redundant information ensues not only from a multiplicity of identical sensors but also from multiple sensors which differ in their method of operation but which ultimately provide the same information. Nature provides us with many examples of this. Vision, for example, is a very powerful sensor for locating objects, but most organisms can augment this information by using other sensors: in insufficient light, humans use a sense of touch to locate objects, bats use sonar, dogs use their keen sense of smell, and some snakes locate prey using special sensors of infrared radiation [48].

The availability of redundant sources of information can be advantageous in providing for choices among various sensing strategies when information must be actively sought. This can be important for several reasons. Certain sensors operate poorly or not at all under certain conditions. For example vision usually fails to perform under poor illumination or when the scene is occluded by smoke or some solid object. In addition if a sensor becomes defective, failing completely, then operation can continue if other means of acquiring information are available. In other situations the choice of a sensor may be guided more by economics, as different sensors generally provide different levels of detail and incur different costs of use in terms of processing required or motion necessary for proper sensor positioning [44]. Alternate sensing is therefore useful in optimizing performance.

Because sensor measurements are seldom perfect, another major advantage to the use of redundant information provided by multiple sensors is that it allows measurement errors to be offset by comparing and combining measurements. If sensory information proves inconsistent due to a noise component in the sensory measurements, then some

form of averaging can be used to provide better estimates. When a piece of data differs significantly from what was expected by other measurements, then sensor failure may be indicated, and that datum can be rejected [12]. Sensor failure results not only from hardware failure, but also from a failure to properly interpret the data.

The availability of multiple sensors can thus extend the capabilities of autonomous mechanisms in addition to allowing for more reliable, accurate, and cost-effective operation. However, along with these benefits comes a great need to coordinate the sensors and to organize the flow of information. Generally applicable methods for combining multi-sensory information are not presently available. Many of the issues involved are poorly understood and will provide new challenges in the years to come [19].

1.2 Sensing Object Pose

In many applications an essential task of the sensory system is to determine the location of objects in the environment with which the host must interact. For rigid objects of known geometry, this reduces to a problem of determining the position and orientation of an object coordinate system with respect to some frame of reference (see Figure 1.1). We shall use the term *pose* to refer to the combined position and orientation parameters, of which there are six.¹

Model-based pose estimation is a process involving three components: object features provided by the object model, sensory features extracted from the sensory data, and a model of the sensing process. The problem contains three main subproblems:

¹ Determining the pose of a known object is fundamentally the same problem as determining the pose of a mobile sensory system with respect to its known environment. Although we refer only to the former problem, both problems are actually addressed.

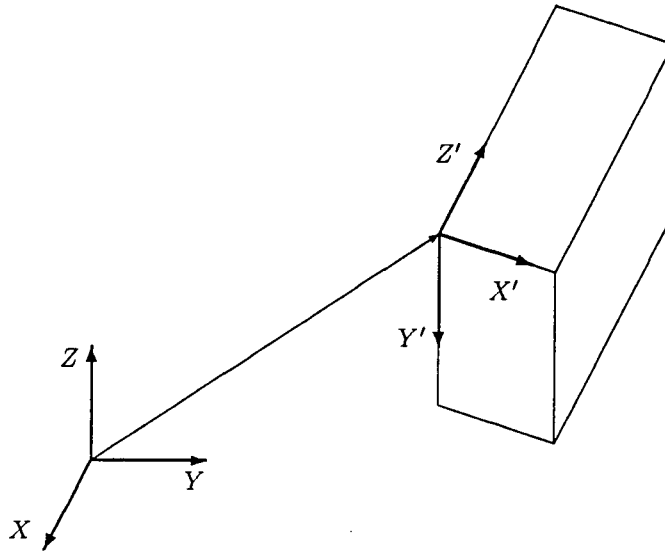


Figure 1.1: An object's pose.

feature extraction, determination of the correspondence between sensory and object features, and determination of pose using this correspondence and the model of the sensing process. These three subproblems are discussed in the following paragraphs.

The sensory information used in determining object pose can come from a variety of sources. Common examples include single camera images (both visual and infrared), depth maps produced by stereo vision or laser ranging devices, single point data generated by touch, proximity, or range sensors, and information provided by structured lighting or calibration marks used with visual images. Any type of sensing can be viewed as a mapping of object features to raw sensory data. Feature extraction is then a further mapping from the raw sensory data to a sensory feature space in which measurable parameters of sensory features are related to object features according to (hopefully simple) models. Feature extraction in some cases is simple (e.g., reporting the coordi-

nates of a touch sensor when contact is made with a surface) while in others it requires considerable preprocessing (e.g., finding an edge in a visual image).

In order to be of any value, an extracted sensory feature must be matched with the object feature which generated it. In some cases it is known what is sensed and the correspondence is automatic, while in others a best match must be found using a matching process (i.e., recognition²). A typical approach is to attempt to find pose using a partial match, and then use this pose estimate as an aid in predicting further correspondences [38,15,8]. Note that matching will usually involve the third subproblem listed above.

According to the sensory model, the parameters of a sensory feature are generated by the corresponding object feature as a function of object pose. Unfortunately the inverse is not unique in general; an infinite number of poses can generate a given sensory feature from an object feature. Each correspondence and its associated sensory model can therefore be viewed as providing a local (partial) estimate of object pose. Fusion of these local estimates is then performed to generate a global pose estimate.

Because sensors inherently contain error and do not report exact values, the local fits will not be perfect and an exact determination of object pose will not be possible; one can only hope to find an estimate which is "best" according to some criterion. When data is sparse or contains large errors, it becomes necessary to analyze the effects of estimated errors in the local data and provide an indication of how accurate a final global pose estimate is expected to be. If pose is not known accurately enough for a specific purpose, then examination of the errors can be used to guide in the collection of additional data to be used in reducing the inaccuracies [54].

² From a recognition point of view, pose is then found as a byproduct.

Most pose estimation is currently performed using ad hoc techniques with a single type of data. However, for reasons cited in the previous section, it can be advantageous to have available data from multiple sources. Data fusion, however, is complicated due to the variety of information. This research addresses a technique for the fusion of data from multiple sources. The problems of feature extraction and matching are not dealt with.

1.3 Background

The sensing of object pose has been a subject of considerable research for many years. By far the majority of this research has involved the use of a single sensor, notably vision (see for example [30,5,40,14]) and more recently range data (see surveys [6,41]). Work has also been done involving specific interactions between particular sensors, for example, vision and range [16,47,1,48] and vision and touch [2,3]; however, comparatively little has been reported on generally applicable methods of multisensor fusion to determine object pose.

Researchers at Utah [35,36,34,33,37] have described a Multisensor Kernel System (MKS) which can be used as a general framework for multisensor integration and data acquisition. In it they use a spatial proximity graph as a flexible means of organizing low-level 3-dimensional data collected by various logical sensors, from which various higher-level descriptions can be built. Logical sensors are used in a data flow network as a means of data abstraction, providing information without regard to its origin. Although their system has general applicability to multisensor problems, the determination of object pose has not been a focus of their work.

Durrant-Whyte [17,18,20,19] has recently reported a comprehensive effort at representing and merging uncertain geometric data from a variety of sensors. He uses Bayesian techniques in fusing probabilistic descriptions of sensory data to build and maintain a single world model.³ Although this approach is widely applicable, his work in applying these methods to the problem of determining object pose treats position and orientation separately without addressing the complexities of their nonlinear inter-relationship.

Bolle and Cooper [9] have described a method of optimally combining large sets of data. This is done by forming subproblems which can each be optimized individually and later merged to create a solution which is globally optimum. The method is general enough that it can be applied to a large class of problems. They applied it to the problem of determining pose using range data but have not as yet implemented it as a multisensor approach.

Some aspects of the research reported in this dissertation bear resemblance to other range data work reported by several researchers. For the purpose of recognition, Bilbro and Snyder [7] perform a best fit of an object model to segments extracted from a range image using a least square error approach. Their fit to planes is actually a subset of the sensing methods included in the present research; however, they do also include fits to general quadric surfaces and have managed to linearize the solution method for their problem. In a related approach Faugeras, et al. [23,4,22,24] estimate pose by minimizing the mean square error in the transformed parameters of object features. As with Bilbro and Snyder's work, the application is with dense range images. Grimson,

³ In a similar brief treatment by Luo [46], fusion is performed by *addition* of probability density functions rather than by the Bayesian method of multiplication.

et al. [32,26,31] localize objects using sparse range (or tactile) data. Rather than use a least square error approach, they check for values within certain bounds and focus on determining all possible legal interpretations of the data.

Research on combining multisensor information to determine the position of a mobile robot has been addressed at several institutions, notably Stanford [44,53], the Laboratoire d'Automatique et d'Analyse des Systèmes (France) [12,13,29], Purdue [42], and M.I.T. [10]. The focus here has been on consistent world modeling in the face of uncertain sensory data gathered after a sequence of moves in two dimensions. The researchers at Stanford and in France use probability functions to represent robot position, while the other two use bounding volumes. Because of the simple nature of their problems, averaging and intersection are used as the appropriate methods of fusion.

1.4 An Approach to Multisensor Fusion

This dissertation presents an approach to model-based pose estimation which can accommodate a variety of sensing modalities. The objective is to provide a framework in which partial pose information from a variety of sensors is treated in a consistent manner. The method can be described as one of the "Least Mean Square Error" (LMSE) variety. The problem is approached from the following viewpoint.

Each sensory measurement involves the set of parameter vectors, $\{\mathbf{P}, \mathbf{M}_i, \mathbf{X}_i, \mathbf{Y}_i\}$, where $\mathbf{P}$ represents the global pose parameters, $\mathbf{M}_i$ represents the parameters of an object feature in model space, $\mathbf{X}_i$ represents the parameters of the object feature in transformed space (i.e., the transformed object feature), $\mathbf{Y}_i$ represents parameters of the sensory feature, and i is the index of the the sensory measurement. Although later

chapters consider parameters which are purely geometric (e.g., the position of points, the direction of lines and surface normals), the generality of the fusion method given below does not require them to be so. For example, optical properties could also be included: the object model could contain surface reflectance properties, the pose vector could contain parameters describing environmental lighting conditions, the transformed object features could contain the intensity of reflected light, and the sensory features could contain gray level values.

These parameter sets are related to one another by the double mapping

$$M_i \xrightarrow{\mathcal{P}(\mathbf{P})} X_i \xrightarrow{\mathcal{S}_i} Y_i \quad (1.1)$$

where $\mathcal{P}(\mathbf{P})$ represents the transformation of the object feature from model space to the transformed space (a function of global pose) and $\mathcal{S}_i$ represents the sensory process which maps a transformed object feature to sensor feature space (a many-to-many mapping). From this perspective, feature extraction is the process of determining the set $\{Y_i\}$ from the raw data, correspondence is the problem of matching an M_i to each Y_i , and pose estimation is the problem of determining $\mathcal{P}(\mathbf{P})$ when given matched (sub)sets of $\{Y_i\}$ and $\{M_i\}$ and knowledge of the associated sensing processes $\{\mathcal{S}_i\}$. This research addresses the last of these problems.

With the approach presented here, each of the parameter vectors, $\mathbf{P}$, X_i , and Y_i , is described using a distance function rather than explicit values in order to incorporate estimates of sensing errors. In this manner, global pose is described by $E(\mathbf{P})$, (local pose estimates by $E_i(\mathbf{P})$), a transformed object feature by $f_i(X_i)$, and a sensory feature by $g_i(Y_i)$. Evaluation of each function for a given set of parameters indicates our estimate of “error” for that set, and the location of a function’s minimum represents our estimate of the most likely values. In particular our best guess of global pose is

given by the minimum of E . Note that the set $\{g_i(\mathbf{Y}_i)\}$ is extracted from the sensory data. Weighting of one sensory measurement over another is done by scaling of the individual functions.

Our task is to perform the inverse mapping given in the above relationship and create the error function for pose, $E(\mathbf{P})$, given the set $\{g_i(\mathbf{Y}_i)\}$ and models of the corresponding mappings. We begin by defining the functions $\{f_i(\mathbf{X}_i)\}$; i.e., for the object feature corresponding to each sensory measurement, we define an error as a function of its transformed feature parameters. We note that for a given instantiation of $\mathbf{X}_i$, there are possibly an infinite number of instantiations of $\mathbf{Y}_i$ which it could map to according to $\mathcal{S}_i$ (e.g., for a given position and orientation of a plane, there are an infinite number of points which could sample it). The $\mathbf{Y}_i$ that we choose to associate with that $\mathbf{X}_i$ is the one which is most likely (i.e., the one where the function g_i is minimum), and the error that we assign for that $\mathbf{X}_i$ will be the error for the $\mathbf{Y}_i$ so chosen. Thus we define

$$\begin{aligned} f_i(\mathbf{X}_i) &= \min_{\mathbf{Y}_i} g_i(\mathbf{Y}_i) \\ &\text{subject to } \mathbf{X}_i \xrightarrow{\mathcal{S}_i} \mathbf{Y}_i \end{aligned} \tag{1.2}$$

In words this states that the error assigned to a particular $\mathbf{X}_i$ is equivalent to the least error possible according to the corresponding sensory measurement and sensing model. This is analogous to defining the distance between point A and a line as “the least distance possible between the point A and point B where point B is constrained to lie on the line.”

Because the relationship $\mathcal{P}(\mathbf{P})$ defines $\mathbf{X}_i$ uniquely for a given $\mathbf{P}$, we can write $\mathbf{X}_i$ as a function of $\mathbf{P}$, and define the distance function for our local pose by

$$E_i(\mathbf{P}) = f_i(\mathbf{X}_i(\mathbf{P})) \tag{1.3}$$

This states that the local estimate of error for a given pose is the same as the error assigned to the resulting transformed object feature parameters. Globally, the error for a given pose is defined as the total error defined by the local pose estimates. Formally, this is written as

$$E(\mathbf{P}) = \sum_i E_i(\mathbf{P}) \quad (1.4)$$

where i varies from one to the number of sensor measurements involved.

Evaluation of this global pose function for an arbitrary pose reflects our estimate of how *unlikely* that pose is based on the sensory measurements. Several aspects of this function are of value. The location of its minimum provides the best guess of pose, while its value at the minimum provides a measurement of agreement among the sensory data. Although we do not address the matching problem in this research, bad correspondences will result in “abnormally” high evaluations of the associated local pose function at the minimum. In addition, the second partial derivatives of the function at the minimum (how quickly it accelerates in value) indicates how tightly constrained the global pose estimate is likely to be. The further constraining of pose as additional sensing is performed (assuming correspondences and modeling are correct) will be reflected in these partials.

Advantages of this method are its unified treatment of a wide variety of geometric sensory information and its incorporation of sensory errors into the pose estimation process. It does require that each sensory measurement be put in the form of a distance function, a process requiring a suitable model of the associated errors; however, any method which performs a similar function would require equivalent information.

A disadvantage is that in general it requires numerical techniques to evaluate the minimum of the required functions, so that the usual problems of convergence and global

minimums can apply. The nature of the solution method makes it more applicable to sparse problems where single-sensor techniques are not available. Suggestions given in the final chapter, however, can serve to reduce the severity of this problem for larger data sets.

1.5 Outline of Remaining Chapters

The remaining chapters of this dissertation present an application of this sensor fusion approach to the sensing of point, line, and plane object features.⁴ The parameters M_i and X_i will therefore describe the position and orientation of these object features with respect to the object and world coordinate systems respectively, P will represent the six degrees of freedom of object pose, and $\mathcal{P}_i(P)$ will represent the coordinate transformation from object to world coordinates. These parameterizations will be described in Chapter 2 along with the mapping (coordinate transformation) of object features from object to world coordinates as a function of object pose.

In order to provide a common basis for representing information from different sensors, Chapter 3 proposes a simple classification scheme which will accommodate a wide variety of sensing modalities. For each sensing class, the parameterizations of the sensory features are given along with the relationships of these parameters to the corresponding object features. Chapter 3 also describes the distance function which will be used to describe the estimated errors in the reported sensory features.

In Chapter 4 the details of the method of generating local pose functions are carried out for each of the sensing classes. For a few of these classes, a closed form of the local

⁴ The object may of course also contain other features, but only points, lines, and planes will be sensed here.

pose function will not be available, so we describe for these classes a procedure in which the required information can be found.

Chapter 5 describes an implementation of the algorithm using the robotics workstation at the University of Tennessee and presents the results from a series of experiments which demonstrate its operation.

Finally, Chapter 6 provides conclusions and recommendations for future work.

CHAPTER 2

TRANSFORMATION OF THE OBJECT MODEL

In the approach to multisensor fusion presented in Section 1.4, a double mapping in which sensory features are generated from the object model was described. The first half of that mapping,

$$M_i \xrightarrow{p(P)} X_i \quad (2.1)$$

represents how a feature of the object model is mapped to the transformed object feature as a function of pose. The purpose of this chapter is to describe this portion of the mapping as it applies to the present formulation of the sensor fusion algorithm.

2.1 The Object Model

There are a number of object properties which can be described by an object model (e.g., dimensions, weight, color, material) and many ways to represent that information. As there are no universal methods of representation, one must choose the method most appropriate for the problem at hand [49]. For our problem of determining object pose, we only require a description of the position and orientation with respect to an object coordinate system of the points, lines, and planes we will be sensing.¹ Our parameterizations of these features will be as follows: points will be described by the coordinates²

¹ Of course, additional information may also be included in the model for other purposes.

² Cartesian coordinates will be used throughout this thesis.

of the point, lines will be described by the coordinates of a point on the line and a unit vector in the direction of the line, and planes will be described by the coordinates of a point on the plane and the unit normal of the plane.³

To simplify the formulation that follows, the same notation will be used for each type of feature: $\mathbf{r}_i = (r_{i1}, r_{i2}, r_{i3})^T$ ⁴ will be used to refer to the coordinate vector of the point, and $\mathbf{w}_i = (w_{i1}, w_{i2}, w_{i3})^T$ will be used to refer to the unit vector.⁵ Thus for a sensory measurement indexed by i , the corresponding object feature in model space is parameterized by the six element vector,

$$\mathbf{M}_i = \begin{pmatrix} \mathbf{r}_i \\ \mathbf{w}_i \end{pmatrix} \quad (2.2)$$

Because point features do not have orientation components, their $\mathbf{w}_i$ can contain any arbitrary unit vector. It will later be ignored.

The simple example given in Figure 2.1 will serve as an illustration. Here a point on a plate is sensed by the first sensory measurement, a line on the plate is sensed by the second, and the surface of the plate is sensed by the third. An appropriate point and vector for the object feature corresponding to each measurement is shown in this figure. In each case the choice of point and vector is not unique, as our parameterizations contain more elements than the number of degrees of freedom represented.⁶

Note that the index i applies to the sensory measurement. When multiple measurements correspond to a single object feature (for example, several range points may

³ Either sense may be used for the vectors.

⁴ T indicates transpose.

⁵ Note we must keep track of the type of object feature represented by the index, i .

⁶ Each parameterization, $\mathbf{M}_i$, contains six elements while points features have three degrees of freedom, line features have four, and plane features have three.

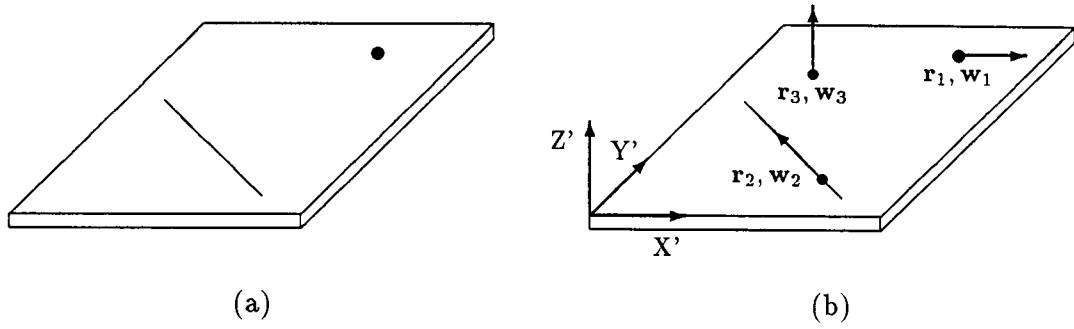


Figure 2.1: (a) A simple object. (b) Parameters of some features.

sample the same surface), each will generate its own instance of that object feature in the set $\{M_i\}$.⁷

2.2 Transformed Object Features

As was shown in Figure 1.1 on page 4, pose is defined as the position and orientation of the object coordinate system with respect to the world coordinate system. Features of our object have representations with respect to both systems: M_i represents the known position and orientation of an object feature with respect to the object coordinate system (values which are independent of pose), and X_i represents the position and orientation of the feature with respect to the world coordinate system. Whereas M_i used r_i and w_i , X_i will use x_i and u_i in an analogous fashion. Thus

$$X_i = \begin{pmatrix} x_i \\ u_i \end{pmatrix} \quad (2.3)$$

⁷ In actual practice each object feature could be described only once. Each sensory measurement would then simply generate a pointer to that feature.

where $\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3})^T$ is the coordinate vector of the point and $\mathbf{u}_i = (u_{i1}, u_{i2}, u_{i3})^T$ is the unit vector (see Figure 2.2).

2.3 The Coordinate Transformation

For our geometric problem, the mapping defined in Expression 2.1 represents the coordinate transformation of object feature parameters from the object coordinate system to the world coordinate system as a function of pose. Because both coordinate systems are Cartesian systems and no scaling is involved, the transformation of a point can be implemented as an orthogonal rotation followed by a translation, and the transformation of a vector can be implemented as an orthogonal rotation [43]. We can therefore write

$$\mathbf{x}_i = \mathbf{Q}\mathbf{r}_i + \mathbf{t} \quad (2.4)$$

and

$$\mathbf{u}_i = \mathbf{Q}\mathbf{w}_i \quad (2.5)$$

where $\mathbf{Q}$ is an orthonormal 3 by 3 rotation matrix⁸ and $\mathbf{t} = (t_1, t_2, t_3)^T$ is the translation. Physically, the first, second, and third columns of $\mathbf{Q}$ represent unit vectors in the world coordinate system in the direction of the X' , Y' , and Z' axes of the object coordinate system, and $\mathbf{t}$ represents the position in the world coordinate system of the origin of the object coordinate system.

We could use $\mathbf{t}$ and $\mathbf{Q}$ for our parameterization of pose— $\mathbf{t}$ to represent the three degrees of translational freedom and $\mathbf{Q}$ to represent the three degrees of rotational

⁸ i.e., we require $\mathbf{Q}\mathbf{Q}^T = \mathbf{I}$ and $\det(\mathbf{Q}) = 1$. This second condition will insure proper rotations, those which preserve the right-handedness of the axes. This rotation matrix preserves the lengths of vectors and the angle between vectors.

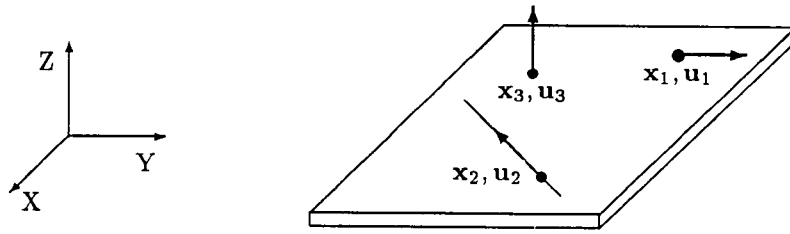


Figure 2.2: Transformed object features of the plate.

freedom. However, we note that the matrix representation of rotation involves nine parameters related by six constraining relationships imposed by its orthonormality⁹ (hence three degrees of freedom are represented). These six constraints must be satisfied later when we solve for object pose. Although the numerical techniques we will use later allow for constraint satisfaction, we will choose a representation of rotation involving fewer parameters and therefore fewer constraints.

There are a number of other ways to parameterize orientation in three dimensions. Some common methods are Euler angles, Gibbs vector, Caley parameters, axis and angle, linear parameters, Caley-Klein parameters, and quaternions [43,55]. We prefer a representation involving few parameters; however, it can be shown [56] that any representation using only three parameters (e.g., Euler angles, Gibbs vector, Caley parameters) can not be both global (every rotation has a finite-valued representation) and nonsingular (every rotation is uniquely defined). Quaternions [55,52,50], which involve four parameters and suffer from neither of these problems, seem to be an ideal choice: the only constraint binding the four parameters is that they form a four-dimensional (4-D) unit vector, and the space of all possible 3-D rotations is mapped uniformly onto

⁹ Ten equations are represented in Footnote 8, six of which are independent. Writing the six constraints using just six equations is not possible [55].

the resulting 4-D unit sphere [40]. Functions of the pose parameters which we will later be involved with will therefore be well-behaved and easy to work with.

Quaternions may be thought of as a special kind of complex number having a real part and three imaginary parts with units $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$. Using a circle above a symbol to denote a quaternion [40,39], we have

$$\overset{\circ}{\mathbf{q}} = q_0 + q_x \mathbf{i} + q_y \mathbf{j} + q_z \mathbf{k} \quad (2.6)$$

where q_0 , q_x , q_y , and q_z are real. If we interpret the units $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ as the basis vectors of a cartesian set of axes, we see that quaternions include real numbers ($q_x = q_y = q_z = 0$), complex numbers ($q_y = q_z = 0$), and vectors in 3-D space ($q_0 = 0$) [25]. Quaternions have many interesting and useful properties, but we need only be concerned with how they can be used to represent an orientation and how we can convert this representation to the rotation matrix form. For further information, the reader is directed to the sources cited above.

Our description of the quaternion representation of orientation begins with the "axis and angle" representation. To get from one orientation to another, an object can rotate an angle θ about some fixed axis in the direction $\mathbf{n} = (n_1, n_2, n_3)^T$. The sign of θ is determined using the "right hand rule." We can therefore represent the orientation of an object by the axis and angle which carries it to its position from an initial alignment with the reference axes. If we require $\mathbf{n}$ to be a unit vector and write it as

$$\mathbf{n} = n_1 \mathbf{i} + n_2 \mathbf{j} + n_3 \mathbf{k} \quad (2.7)$$

where $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are the basis vectors (unit vectors in the direction of the X , Y , and Z axes), then the quaternion representation of the orientation is given by

$$\overset{\circ}{\mathbf{q}} = \cos \frac{\theta}{2} + \mathbf{n} \sin \frac{\theta}{2} \quad (2.8)$$

The parameters of a quaternion generated in this manner,

$$(q_0, q_x, q_y, q_z) = \left(\cos \frac{\theta}{2}, n_1 \sin \frac{\theta}{2}, n_2 \sin \frac{\theta}{2}, n_3 \sin \frac{\theta}{2} \right) \quad (2.9)$$

are referred to as the Euler symmetrical parameters. It is easy to verify that the sum of the squares of these parameters is unity.

The rotation matrix corresponding to a quaternion defined in the above manner can be written in terms of the quaternion parameters as [39]

$$\mathbf{Q} = \begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_0 q_z) & 2(q_x q_z + q_0 q_y) \\ 2(q_y q_x + q_0 q_z) & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_0 q_x) \\ 2(q_z q_x - q_0 q_y) & 2(q_z q_y + q_0 q_x) & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix} \quad (2.10)$$

Substitution of this matrix into Equations 2.4 and 2.5 provides us with the relationship between $\mathbf{M}_i$ and $\mathbf{X}_i$ which is based on the pose parameterization

$$\mathbf{P} = \begin{pmatrix} \mathbf{t} \\ \mathring{\mathbf{q}} \end{pmatrix} \quad (2.11)$$

For convenience we will represent the quaternion $\mathring{\mathbf{q}}$ by the 4-D vector comprised of its parameters (i.e., $\mathring{\mathbf{q}} = (q_0, q_x, q_y, q_z)^T$) and therefore consider pose as a vector of seven elements. As stated above, these parameters are bound by a single constraint: the last four of these elements must form a vector of unit length.

As an example of the transformation given in this chapter, suppose that we have an object feature described in model space by $\mathbf{r}_i = (1, 2, -1)^T$ and $\mathbf{w}_i = (0, 1, 0)^T$ and a pose described by $\mathbf{t} = (3, 1, 4)^T$ and $\mathring{\mathbf{q}} = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0)^T$.¹⁰ Substituting the parameters

¹⁰ The reader can use Equation 2.9 to verify that this orientation represents a 90-degree rotation about the X -axis.

of the quaternion into Equation 2.10 we get

$$\mathbf{Q} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad (2.12)$$

Now, substitution of this matrix and the translation parameters, $\mathbf{t}$, into Equations 2.4 and 2.5 gives us $\mathbf{x}_i = (4, 2, 6)^T$ and $\mathbf{u}_i = (0, 0, 1)^T$ for our description of the object feature in world coordinates.

2.4 Chapter Summary

This chapter has presented the parameterizations of pose and of the object features in model and world coordinates. Equations 2.4, 2.5, and 2.10 defined the mapping of object features from one space to the other as a function of the pose parameters. These equations will later be used in solving for object pose.

CHAPTER 3

SENSORY FEATURES

As is represented by the second half of the double mapping shown in Section 1.4,

$$\mathbf{X}_i \xrightarrow{\mathcal{S}} \mathbf{Y}_i \quad (3.1)$$

the sensing model describes a relationship between the object features described in world coordinates ($\mathbf{X}_i$) and the sensory features extracted from the sensory data ($\mathbf{Y}_i$). Because of the variety of information provided by different sensing modalities, it is necessary that we find some common basis for representing these relationships. In an attempt to satisfy this requirement, we propose in this chapter a simple means of classifying sensor data which can accommodate a variety of sensing modalities. The parameterizations of the sensory features, $\mathbf{Y}_i$, will be given along with the relationships, $\mathcal{S}_i$, for each class of sensing. This chapter will also describe the distance functions we will use to describe the parameters of the sensory features.

3.1 A Classification of Sensory Data

The classification we will use requires that each sensory feature describe either a point, a line, or a plane in the world coordinate system. According to the sensory models we consider, each sensory feature is related to an object feature (described in world coordinates) by one of the following relationships: *contains*, *is contained in*, or *is*. The resulting nine classes of sensing are given in Figure 3.1 and examples of

		Transformed Object Feature		
		Point	Line	Plane
Sensory Feature	Point	<i>is</i>	<i>is contained in</i>	<i>is contained in</i>
	Line	<i>contains</i>	<i>is</i>	<i>is contained in</i>
	Plane	<i>contains</i>	<i>contains</i>	<i>is</i>
RELATIONSHIP				

Figure 3.1: The sensing relationships between sensory features and transformed object features.

each are given in Figure 3.2. Because each of the nine classes involves a particular relationship between the object features and sensory features, each will involve its own particular equations in generating the local pose distance function. The individual problems will be referred to using a two word descriptor, the first word referring to the type of sensory feature and the second referring to the type of object feature. For example, the *point-plane* problems represent those in which a sensor reports information about the coordinates of a point which lies on a planar surface of the object.

Although this classification of sensing may seem rather artificial and restrictive, a wide variety of sensing methods are actually represented. For example, touch or range sensing of a surface is a point-plane problem, a light stripe seen on a surface can be solved as a line-plane problem, visual sensing of a vertex is a line-point problem, and

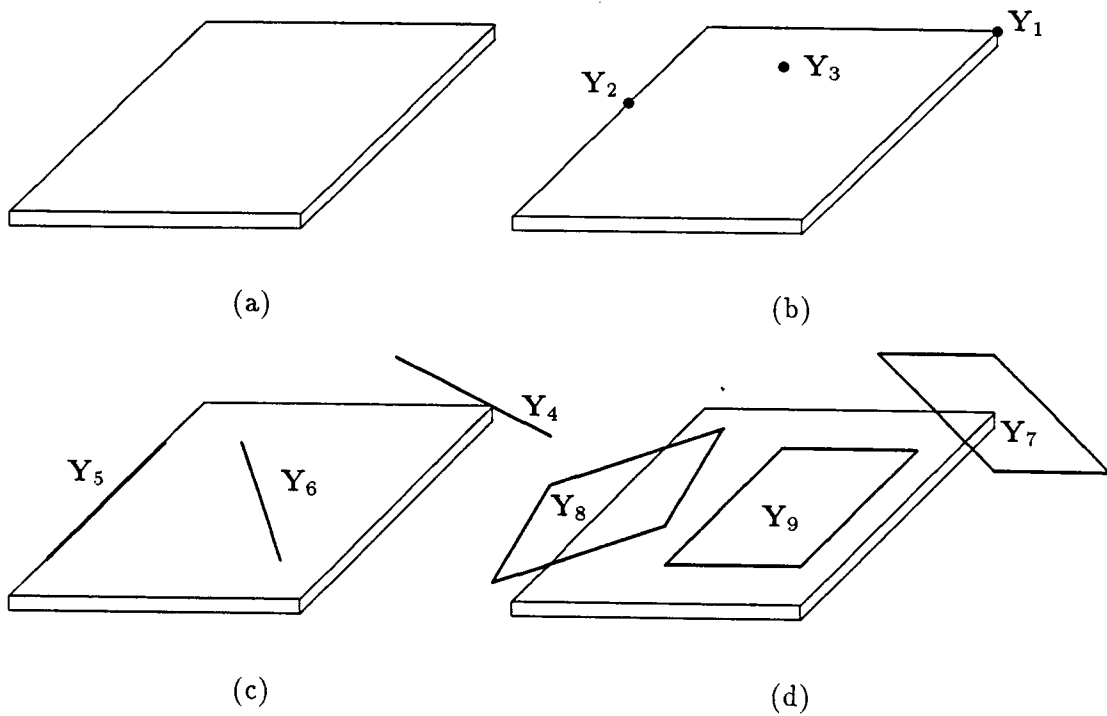


Figure 3.2: (a) A simple object. (b) Point sensing of some object features. (c) Line sensing of some object features. (d) Plane sensing of some object features.

visual sensing of an edge is a plane-line problem. Other examples given later in this chapter will serve to demonstrate its wide applicability.

3.2 Parameterization of Sensory Features

Just as a point and a vector were used to parameterize point, line, and plane object features, they will also be used for our parameterization of sensory features; i.e., point features will be described by the point and an arbitrary unit vector, line features will be described by a point on the line and a unit vector in the direction of the line, and plane features will be described by a point on the plane and the unit normal to the plane.¹ Each of these values will be with respect to the world coordinate system. For each type of sensory feature, we will use $\mathbf{y}_i = (y_{i1}, y_{i2}, y_{i3})^T$ to refer to the coordinate vector of the point and $\mathbf{v}_i = (v_{i1}, v_{i2}, v_{i3})^T$ to refer to the unit vector. The parameterization of each sensory feature is therefore

$$\mathbf{Y}_i = \begin{pmatrix} \mathbf{y}_i \\ \mathbf{v}_i \end{pmatrix} \quad (3.2)$$

analogous to those of the object features in the object and world coordinate systems.

3.3 The Sensing Models

The sensing model for each class of sensing describes a geometric relationship between an appropriate object feature in world coordinates and its corresponding sensory feature (also in world coordinates). Each of these relationships will be written as a set of constraint equations involving the parameters of the corresponding pair of object and

¹ Again either sense may be used for the unit vectors.

sensory features. Each of the following subsections will describe a class of sensing by providing an example and presenting the associated constraint equations.

3.3.1 Point-Point Problems

The point-point problems are those in which the sensor reports an estimate of the 3-D coordinates of a point feature on an object. An example of this type of sensing is where stereo vision is used to locate an identifiable point on the object such as a vertex, small hole, or light. Knowing the position and orientation of the two cameras and the direction of the object point from each, the 3-D coordinates of the point can be computed.

This is the simplest type of problem we will work with, as the mapping from the object feature (a point in world coordinates) to the sensory feature is in this case one-to-one (if we ignore the arbitrary unit vector specifying direction); i.e., the descriptions of the object feature and the sensory feature are unique, and they refer to the same thing. For this type of problem, we therefore write

$$\mathbf{y}_i = \mathbf{x}_i \quad (3.3)$$

as our constraint relationship between the parameters of object features and those of sensory features. Note that we do not need any constraints involving the arbitrary unit vectors, $\mathbf{u}_i$ and $\mathbf{v}_i$.

3.3.2 Point-Line Problems

Point-line problems are used when the sensor can estimate the coordinates of a point which lies on an object line. An example of this type of sensing is given by the detection of an edge on an object (formed by the intersection of two planar surfaces) by a "point sensor" such as a proximity or touch sensor.

For this type of sensing, the constraint dictates that the point sensory feature, $\mathbf{y}_i$, must be contained in the line object feature parameterized by $\mathbf{x}_i$ and $\mathbf{u}_i$. Recalling that $\mathbf{x}_i$ is a point on this line and $\mathbf{u}_i$ is a unit vector in the direction of the line, we can write this constraint as

$$\mathbf{y}_i = \mathbf{x}_i + k\mathbf{u}_i \quad (3.4)$$

where k is a (scalar) free variable.² Because $\mathbf{v}_i$ is arbitrary, no constraints need involve it.

3.3.3 Point-Plane Problems

The point-plane problems occur when a sensor, such as a touch or range sensor, reports the coordinates of an arbitrary point on the surface of the object.³ Our constraint for this problem is that the point sensory feature, $\mathbf{y}_i$, must lie on the plane which contains the point $\mathbf{x}_i$ and has the unit surface normal $\mathbf{u}_i$. This can be represented by

$$(\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \quad (3.5)$$

which essentially means that the vector from $\mathbf{x}_i$ to $\mathbf{y}_i$ must not have a component in the direction of $\mathbf{u}_i$, the normal to the plane. Note that once again the arbitrary unit vector, $\mathbf{v}_i$, does not enter into the constraint relationship.

3.3.4 Line-Point Problems

Line-point problems are used when a sensor determines the parameters of a line which contains a point feature. An example of this type of sensing is given by the visual

² The value of k may be interpreted as the signed distance from $\mathbf{x}_i$ to $\mathbf{y}_i$.

³ If a *known* point on the surface is sensed, then it would be better to use the point-point method, as that method places more restrictions on pose.

sensing of a point object feature. The image coordinates of the point, together with the position and orientation of the camera, can be used to determine the parameters of the line sensory feature.

The constraint relationship for the line-point problem is similar to that of the point-line problem. The difference is that the direction of the line is now defined by $\mathbf{v}_i$ instead of $\mathbf{u}_i$. Here our constraint relationship is written as

$$\mathbf{y}_i = \mathbf{x}_i + k\mathbf{v}_i \quad (3.6)$$

where k is once again a (scalar) free variable. The arbitrary unit vector $\mathbf{u}_i$ does not enter into the constraints for this type of problem.

3.3.5 Line-Line Problems

Line-line problems are used when a sensor is able to directly report an estimate of the parameters of a line object feature. An example of this type of sensing is given by the use of stereo vision to sense an edge of an object. Here the parameters of the line object feature can be computed from the depth information provided by the two images and the known positions of the two cameras.

Although the parameters of the sensory feature and the object feature describe the same line for this type of problem, these representations are not unique: the points $\mathbf{y}_i$ and $\mathbf{x}_i$ need not be the same, but they are required to both lie on a line whose direction is given by $\mathbf{u}_i$ (and $\mathbf{v}_i$). The associated constraint can be written as

$$\mathbf{y}_i = \mathbf{x}_i + k\mathbf{u}_i \quad (3.7)$$

where once again k is a (scalar) free variable. We note that $\mathbf{u}_i$ and $\mathbf{v}_i$ must represent the same line direction, but do not actually have to have the same sense (they could

point in *opposite* directions); however, because we will later represent $\mathbf{v}_i$ using a distance function which treats $+\mathbf{v}_i$ and $-\mathbf{v}_i$ the same, our constraint can force $\mathbf{v}_i$ to point in the same direction as $\mathbf{u}_i$. Our additional constraint relationship for this type of problem is therefore

$$\mathbf{v}_i = \mathbf{u}_i \quad (3.8)$$

3.3.6 Line-Plane Problems

Line-plane problems involve sensing in which an estimate can be made of the parameters of an arbitrary line which lies on a planar surface of the object. An example of this is where a sheet of light is projected on a surface, creating a light stripe, and this stripe is viewed by a camera. The parameters of the line can be computed from the intersection of two planes: one defined by the position and orientation of the camera, the location of the projection of the stripe on the camera image plane, and the camera model, and the other defined by the position and orientation of the source of the sheet of light.

Because the line sensory feature must be contained in the plane object feature, every individual point on the line, including $\mathbf{y}_i$, must be contained in this plane, and we can therefore write

$$(\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \quad (3.9)$$

as we did with the point-plane problems. We also require the direction of the line to be perpendicular to the normal to the plane and therefore write

$$\mathbf{v}_i^T \mathbf{u}_i = 0 \quad (3.10)$$

as an additional constraint.

3.3.7 Plane-Point Problems

With plane-point sensing a point object feature is determined to lie in a particular plane in space, as when a large planar surface on a robot makes contact with a vertex. As with the point-plane and line-plane problems, we will constrain $\mathbf{y}_i$ and $\mathbf{x}_i$ to lie in the same plane; however, this will now be a sensory-feature plane (normal given by $\mathbf{v}_i$) rather than an object-feature plane (normal given by $\mathbf{u}_i$). The constraint relationship for this type of problem is therefore

$$(\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i = 0 \quad (3.11)$$

which here means that the vector from $\mathbf{x}_i$ to $\mathbf{y}_i$ must not have a component in the direction of $\mathbf{v}_i$, the normal to the sensory plane. Once again, the arbitrary unit vector $\mathbf{u}_i$ does not enter into the constraint relationship.

3.3.8 Plane-Line Problems

Plane-line problems are those in which a particular plane in space is known to contain a certain line object feature. An example of this is when lines on an object are observed with a visual sensor. Here, the sensory plane is defined by the projection of the object line on the image plane, the center of the lens (position of the pin-hole using a pin-hole camera model), and the position and orientation of the camera.

To generate the constraints for this problem we note that every point on the line object feature, including $\mathbf{x}_i$, must be contained in the sensory plane parameterized by $\mathbf{y}_i$ and $\mathbf{v}_i$. We therefore write

$$(\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i = 0 \quad (3.12)$$

to represent this constraint. As with the line-plane problems, we require the direction

of the line to be perpendicular to the normal to the plane and therefore write

$$\mathbf{v}_i^T \mathbf{u}_i = 0 \quad (3.13)$$

as an additional constraint.

3.3.9 Plane-Plane Problems

Finally, the plane-plane problems are used when a sensor is able to directly report the parameters of a plane object feature. This occurs, for example, when planar surface of the object is sampled several times by a touch or range sensor (e.g., a range image). Although each individual measurement could be treated as a separate point-plane problem, it would be simpler to fit a plane through these points and replace these multiple problems with a single plane-plane problem.

Both the object feature parameters and sensory feature parameters refer to the same plane, but as was the case with the line-line problems, the description of this plane is not unique. $\mathbf{y}_i$ and $\mathbf{x}_i$ need not be the same point, but they are constrained to lie in a plane with a unit normal given by $\mathbf{u}_i$ (and $\mathbf{v}_i$). This constraint can be written as

$$(\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \quad (3.14)$$

We note that, as with the line-line problems, $\mathbf{u}_i$ and $\mathbf{v}_i$ must represent the same unit normal but do not necessarily have to have the same sense. Once again, however, the distance function we will use later to represent $\mathbf{v}_i$ will treat $+\mathbf{v}_i$ and $-\mathbf{v}_i$ the same and we can therefore require $\mathbf{u}_i$ and $\mathbf{v}_i$ to point in the same direction. This last constraint is now written as

$$\mathbf{v}_i = \mathbf{u}_i \quad (3.15)$$

3.3.10 Constraint Summary

In addition to the constraints listed above which apply to the individual problems, we note that the vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ must have unit length, and we therefore require

$$\mathbf{u}_i^T \mathbf{u}_i = 1 \quad (3.16)$$

and

$$\mathbf{v}_i^T \mathbf{v}_i = 1 \quad (3.17)$$

If we generate the object feature vectors, $\mathbf{u}_i$, from the unit vectors of the model, $\mathbf{w}_i$, using the coordinate transformation given in the previous chapter (as we later will), then the above constraint for $\mathbf{u}_i$ will be satisfied automatically. We must, however, keep the constraint for $\mathbf{v}_i$ in mind as we solve for pose. We will therefore add this constraint to the constraint sets given by Equations 3.3 through 3.15 for the individual problems. A summary of these constraints sets is given in Figure 3.3 for future reference.

3.4 Distance Functions for Sensory Features

As was mentioned in Section 1.4, the method of sensor fusion will require each sensor to report a distance function of the parameters of the sensory feature rather than specific values. The evaluation of this function for a given set of parameters produces a measure of how *unlikely* we estimate that parameter set to be. This will provide a unified method of incorporating estimates of measurement errors into the problem. This section describes the distance functions we will be using.

Each of our point, line, and plane sensory features is parameterized using a point, $\mathbf{y}_i$, and a unit vector, $\mathbf{v}_i$. For each sensory feature, regardless of its type, we will use a

		Transformed Object Feature		
		Point	Line	Plane
Sensory Feature	Point	$y_i = x_i$ $v_i^T v_i = 1$	$y_i = x_i + k u_i$ $v_i^T v_i = 1$	$(y_i - x_i)^T u_i = 0$ $v_i^T v_i = 1$
	Line	$y_i = x_i + k v_i$ $v_i^T v_i = 1$	$y_i = x_i + k u_i$ $v_i = u_i$ $v_i^T v_i = 1$	$(y_i - x_i)^T u_i = 0$ $v_i^T u_i = 0$ $v_i^T v_i = 1$
	Plane	$(y_i - x_i)^T v_i = 0$ $v_i^T v_i = 1$	$(y_i - x_i)^T v_i = 0$ $v_i^T u_i = 0$ $v_i^T v_i = 1$	$(y_i - x_i)^T u_i = 0$ $v_i = u_i$ $v_i^T v_i = 1$

CONSTRAINT SETS $\mathcal{C}_i$

Figure 3.3: Constraint relationships between sensory features and object features.

distance function of the form

$$g_i(y_i, v_i) = (y_i - \mu_i)^T C_i^{-1} (y_i - \mu_i) + v_i^T M_i v_i \quad (3.18)$$

where the 3-D vector μ_i and the 3 by 3 matrices, C_i^{-1} and M_i , contain coefficients which are generated as a combination of the sensory measurement and its estimated errors.⁴ We will require that C_i^{-1} and M_i be real, symmetric matrices. This means the eigenvalues of each will be real, and the set of eigenvectors for each will be orthogonal [43]. We will also require that C_i^{-1} be positive definite and that M_i be positive semidefinite.⁵

⁴ We represent this first matrix by C_i^{-1} rather than by C_i because of its close association to the inverse covariance matrix used with multivariable Gaussian probability functions and with the Mahalanobis distance used in pattern recognition [57].

⁵ If a matrix, A , is positive definite, then $x^T A x > 0$ for all nonzero vectors x . If it is positive semidefinite, then $x^T A x \geq 0$ for all nonzero vectors x , with the equality holding for at least some nonzero vectors x [43].

The distance function given above consists of a sum of two quadratic functions, one of $\mathbf{y}_i$ and the other of $\mathbf{v}_i$, each representing a “square error” function of its variables. The coefficients of these functions will be chosen so that the minimum of the function is zero, and the minimum occurs at the most likely values of the parameters. Note that the arbitrary sense we give to the direction vector, $\mathbf{v}_i$, makes no difference in the evaluation of the function (i.e., $\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i = (-\mathbf{v}_i)^T \mathbf{M}_i (-\mathbf{v}_i)$). Also note that for point sensory features, we can choose $\mathbf{M}_i$ to be a matrix of zeros, in order that the arbitrary unit vector, $\mathbf{v}_i$, not contribute to the distance function. The following subsections will discuss the two quadratic functions in some detail in order that the reader might gain an intuitive feel of their meaning and also to provide insight into how the coefficients can be generated.

3.4.1 Matrix Decomposition

Any real symmetric matrix⁶ $\mathbf{A}$ can be decomposed into

$$\mathbf{A} = \mathbf{R} \mathbf{\Lambda} \mathbf{R}^T = \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{pmatrix} \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix} \quad (3.19)$$

where the λ_j are the (necessarily real) eigenvalues of $\mathbf{A}$, and $\mathbf{R}$ is a 3 by 3 (proper) rotation matrix whose columns, $\mathbf{r}_j$, represent the corresponding normalized eigenvectors (i.e., an orthogonal set of unit vectors) [43]. If $\mathbf{A}$ is also positive definite (as is $\mathbf{C}_i^{-1}$), then all of its eigenvalues are positive, and we can write

$$0 < \lambda_1 \leq \lambda_2 \leq \lambda_3 \quad (3.20)$$

⁶ In this dissertation all matrices are 3 by 3, and all vectors are 3-dimensional unless otherwise specified.

where we have arbitrarily labeled our eigenvalues in order of increasing value. Now for any vector $\mathbf{x}$, we have

$$\begin{aligned}\mathbf{x}^T \mathbf{A} \mathbf{x} &= \mathbf{x}^T \mathbf{R} \mathbf{\Lambda} \mathbf{R}^T \mathbf{x} \\ &= \mathbf{z}^T \mathbf{\Lambda} \mathbf{z} \\ &= \lambda_1 z_1^2 + \lambda_2 z_2^2 + \lambda_3 z_3^2\end{aligned}\tag{3.21}$$

where $\mathbf{z} = (z_1, z_2, z_3)^T$, and we have used the substitution

$$\mathbf{z} = \mathbf{R}^T \mathbf{x}\tag{3.22}$$

Since $\mathbf{R}$ is a rotation matrix, $\mathbf{z}$ represents a description of $\mathbf{x}$ in a coordinate system rotated by $\mathbf{R}$ (see Equation 2.5 on page 17), and z_j is therefore the component of $\mathbf{x}$ in the direction of the basis vector $\mathbf{r}_j$. Equation 3.21 indicates that the evaluation of the quadratic $\mathbf{x}^T \mathbf{A} \mathbf{x}$ for a given vector $\mathbf{x}$ reduces to a linear combination of the squares of these components of $\mathbf{z}$. Furthermore, if we scale $\mathbf{x}$ (and therefore $\mathbf{z}$) by a factor of k , then $\mathbf{x}^T \mathbf{A} \mathbf{x}$ increases by a factor of k^2 . It is also important to note that because none of the eigenvalues are negative, the minimum of the function will be λ_1 .

A contour of $\mathbf{x}^T \mathbf{A} \mathbf{x}$, that is, a surface over which

$$\mathbf{x}^T \mathbf{A} \mathbf{x} = c = \text{constant}\tag{3.23}$$

provides a useful means of visualizing the distance function represented. Equation 3.21 indicates that each contour forms an ellipsoid which is oriented with the eigenvectors of $\mathbf{A}$, and the length of a semi-major axis oriented with eigenvector $\mathbf{r}_j$ is given by $\sqrt{c/\lambda_j}$.

3.4.2 Position Component of the Sensory Distance Function

Returning to the distance function for sensory features given in Equation 3.18 on page 33, we see that the first term, $(\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i)$, has these ellipsoidal contours, but there is also a displacement involved: a contour is centered on the point $\boldsymbol{\mu}_i$

rather than on the origin. This term therefore represents the scaled squared distance from μ_i to y_i where the scale factor depends on the direction. Note that if C_i^{-1} is equal to the identity matrix, then the contours are spheres, and the quadratic represents the Euclidean distance squared.

The coefficients μ_i and C_i^{-1} reflect knowledge about the value of the point y_i and the associated errors. The function $(y_i - \mu_i)^T C_i^{-1} (y_i - \mu_i)$ is minimum when y_i is equal to μ_i , so if the function is to reflect our knowledge of y_i , then μ_i should represent the most likely value of y_i . In most cases this value comes directly from the sensor reading and knowledge of the sensor position. For example, if we have point-plane sensing, as is given by a range sensor sampling a surface, then the most likely value of our point sensory feature, y_i , is the coordinates we calculate for the surface point based on the distance measurement and the sensor position and orientation. As another example, consider the visual sensing of a point (line-point sensing) where the line sensory feature is defined by the position and orientation of the camera, the camera focal length, and the image coordinates of the projection of the object point. A logical choice is to let y_i , the point on the sensory line, represent the position of the camera lens center. The most likely position of the lens center (errors prevent exact knowledge of position) is then used for μ_i .

The matrix C_i^{-1} is used to indicate how we expect y_i to vary from μ_i . The three orthogonal eigenvectors are chosen to be oriented with our error contours (the ellipsoids described above). r_1 is chosen to point in the direction in which the most variation in y_i is expected (remember we labeled our eigenvalues in order of increasing value), and r_3 points in the direction in which the least is expected. The eigenvalues are

chosen to reflect the relative variations in these directions.⁷ As an example, consider the coordinates of a point computed by stereo vision. We typically expect the error to be greatest in the “depth” direction, so $\mathbf{r}_1$ would be chosen to lie in this direction. If we expect the error to be least in a particular direction orthogonal to this, then we would choose $\mathbf{r}_3$ to lie in that direction and $\mathbf{r}_2$ to be orthogonal to both $\mathbf{r}_1$ and $\mathbf{r}_3$; otherwise $\mathbf{r}_2$ and $\mathbf{r}_3$ are chosen using, for example, the Gram-Schmidt orthogonalization procedure [43]. The eigenvalues for these directions are chosen to reflect the corresponding error.

3.4.3 Direction Component of the Sensory Distance Function

We now turn the discussion to the orientation term, $\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i$, of the distance function given in Equation 3.18 on page 33. Because $\mathbf{v}_i$ is restricted to be a unit vector, the contours of this quadratic have no physical meaning, so we desire some other means of visualizing what is represented by the quadratic. We will begin discussion on this topic with an explanation of why we restrict the matrix $\mathbf{M}_i$ to be positive *semidefinite*.

As we did with the matrix $\mathbf{C}_i^{-1}$, we can also decompose $\mathbf{M}_i$ according to Equation 3.19 and describe $\mathbf{v}_i$ using a coordinate system aligned with the eigenvectors. For the moment we will not place any restrictions on the eigenvalues (aside from assuming they are real), but we will label them in order of increasing value and write

$$\lambda_1 \leq \lambda_2 \leq \lambda_3 \quad (3.24)$$

If we let $\mathbf{A}$ represent $\mathbf{M}_i$ and $\mathbf{x}$ represent $\mathbf{v}_i$, then Equation 3.21 gives us

$$\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i = \lambda_1 z_1^2 + \lambda_2 z_2^2 + \lambda_3 z_3^2 \quad (3.25)$$

⁷ Note that if equal error is expected in all directions, then the eigenvalues are equal, and $\mathbf{R}$ may be chosen to be the identity matrix. The quadratic then represents the Euclidean distance squared.

were z_j again represents the component of $\mathbf{v}_i$ in the direction of the eigenvector $\mathbf{r}_j$. Note that if $\mathbf{v}_i$ is equal to plus or minus eigenvector $\mathbf{r}_j$ (both are unit vectors), then z_j will be plus or minus 1, the other two components of $\mathbf{z}$ will be zero, and the value of the quadratic will be λ_j .

Because $\mathbf{v}_i$ is a unit vector, $\mathbf{z}$ will be also, and we have

$$z_1^2 + z_2^2 + z_3^2 = 1 \quad (3.26)$$

This lets us rewrite Equation 3.25 as

$$\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i = \lambda_1 z_1^2 + \lambda_1 z_2^2 + \lambda_1 z_3^2 + \lambda_2 z_2^2 - \lambda_1 z_2^2 + \lambda_3 z_3^2 - \lambda_1 z_3^2 \quad (3.27)$$

$$= \lambda_1(z_1^2 + z_2^2 + z_3^2) + (\lambda_2 - \lambda_1)z_2^2 + (\lambda_3 - \lambda_1)z_3^2 \quad (3.28)$$

$$= \lambda_1 + \lambda'_2 z_2^2 + \lambda'_3 z_3^2 \quad (3.29)$$

where we have used λ'_j to represent $(\lambda_j - \lambda_1)$.

Several important observations can be made from Equation 3.29. First we note that as far as unit vectors are concerned, the quadratic contains a constant term, λ_1 , representing the minimum of the function, and two other terms, $\lambda'_2 z_2^2$ and $\lambda'_3 z_3^2$, which determine its "shape." Because we want our quadratic to represent a distance function, we would like for its minimum to be zero. This implies that λ_1 , our smallest eigenvalue, is zero and that $\mathbf{M}_i$ is positive semidefinite. We therefore write

$$0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \quad (3.30)$$

and have

$$\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i = \lambda_2 z_2^2 + \lambda_3 z_3^2 \quad (3.31)$$

Notice that by requiring λ_1 to be zero, we have not lost any control over the shape of

the quadratic (when evaluated for unit vectors), but have only specified its constant term.⁸

To help us visualize what is represented by $\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i$, we recall that z_j represents the component of $\mathbf{v}_i$ in the direction of eigenvector $\mathbf{r}_j$. z_2 and z_3 therefore describe the orthogonal projection of $\mathbf{v}_i$ onto the plane defined by eigenvectors $\mathbf{r}_2$ and $\mathbf{r}_3$ of the matrix, $\mathbf{M}_i$. It is convenient to think about the error represented by the quadratic in terms of this projection, because Equation 3.31 tells us that the error contours in this two dimensional space are concentric ellipses. When $\mathbf{v}_i$ lies in the direction of eigenvector $\mathbf{r}_1$ (the most likely direction for $\mathbf{v}_i$), then its projection is at $(z_2, z_3) = (0, 0)$, the point where the error is zero. As $\mathbf{v}_i$ moves in a plane from this orientation to an orthogonal direction where $\mathbf{z} = (0, z'_2, z'_3)$, its projection moves in a straight line from the origin to the point (z'_2, z'_3) towards this point. It is instructive to see how the error varies for this motion. If we let θ represent angle between $\mathbf{v}_i$ and $\mathbf{r}_1$, then the projection is at

$$\mathbf{z} = (0, \sin(\theta)z'_2, \sin(\theta)z'_3) \quad (3.32)$$

and the error represented is therefore

$$\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i = \sin^2 \theta (\lambda_2 z'^2_2 + \lambda_3 z'^2_3) \quad (3.33)$$

We see that for this motion, the error increases with the $\sin^2 \theta$ toward a maximum of $\lambda_2 z'^2_2 + \lambda_3 z'^2_3$.

We can use these results to help in selecting values for λ_2 , λ_3 , and $\mathbf{R}$ for the purpose of defining $\mathbf{M}_i$. We select $\mathbf{r}_1$ to be in the direction which our sensory measurement indicates is the most likely direction for $\mathbf{v}_i$, and $\mathbf{r}_3$ to be in an orthogonal direction

⁸ We could have achieved the same effect by defining our quadratic as $\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i - \lambda_1$, where λ_1 is the smallest eigenvalue.

towards which $\mathbf{v}_i$ is least likely to deviate.⁹ $\mathbf{r}_2$ of course is chosen to be orthogonal to both $\mathbf{r}_1$ and $\mathbf{r}_3$. For λ_2 and λ_3 we select values which reflect how quickly we want the error to increase as $\mathbf{v}_i$ moves from $\mathbf{r}_1$ towards $\mathbf{r}_2$ and $\mathbf{r}_3$, respectively, remembering these values are multiplied by $\sin^2 \theta$.

Note that the magnitude of these eigenvalues relative to those of $\mathbf{C}_i^{-1}$ must reflect our relative confidence in knowledge of orientation over knowledge of position for the sensory feature. Similarly, the relative confidence of one sensory measurement over another is achieved by the appropriate scaling of the corresponding sensory distance functions.

3.5 Chapter Summary

In this chapter we have described the sensory information which will be used by the sensor fusion method presented in the following chapter. We have proposed a scheme of classification of sensory data which will accommodate a variety of modalities for sensing object pose. We have given an example of each class of sensing and described the necessary relations between the sensory feature and sensed object feature. We have also described the distance function we will use to represent the sensory data and the estimated errors in measurement.

⁹ Use, for example, the Gram-Schmidt orthogonalization procedure if the accuracy is equal in all directions away from $\mathbf{r}_1$.

CHAPTER 4

DETERMINING OBJECT POSE

Section 3.4 described the distance function we will use to represent the parameters of a sensory feature. As was outlined in Section 1.4, we will convert each of these sensory distance functions to a distance function of the global pose parameters in order to estimate object pose. This chapter will present the details of this process.

4.1 Outline of the Method

Recall from Section 1.4 that we use the distance function of the transformed object features, $f_i(\mathbf{x}_i, \mathbf{u}_i)$, as an intermediate step in forming each local pose distance function, $E_i(\mathbf{t}, \hat{\mathbf{q}})$. Equation 1.2 on page 10 defined the value of the object feature distance function for a particular $\mathbf{X}_i$ (i.e., $\mathbf{x}_i, \mathbf{u}_i$) as the minimum value of the sensory distance function, $g_i(\mathbf{y}_i, \mathbf{v}_i)$, subject to the condition that that object feature could map to the sensory feature (i.e., choose the error assigned to the most likely sensory feature parameters). Section 3.3 described this mapping for a variety of types of sensing in terms of constraint equations involving the parameters of the two distance functions. We can therefore rewrite Equation 1.2 as

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i, \mathbf{v}_i} g_i(\mathbf{y}_i, \mathbf{v}_i) \\ &\text{subject to } \mathcal{C}_i(\mathbf{x}_i, \mathbf{u}_i) \end{aligned} \tag{4.1}$$

where the constraint sets for the individual problems, $\mathcal{C}_i(\mathbf{x}_i, \mathbf{u}_i)$, are given in Figure 3.3 on page 33.

Equation 1.3 on page 10 defined the local pose distance functions in terms of the object feature distance functions by writing the parameters of the object feature as functions of the pose parameters. For our parameterizations Equation 1.3 can be written as

$$E_i(t, \mathring{\mathbf{q}}) = f_i(\mathbf{x}_i(t, \mathring{\mathbf{q}}), \mathbf{u}_i(t, \mathring{\mathbf{q}})) \quad (4.2)$$

where the functions $\mathbf{x}_i(t, \mathring{\mathbf{q}})$ and $\mathbf{u}_i(t, \mathring{\mathbf{q}})$ are defined by Equations 2.4, 2.5, and 2.10 (pages 17 and 20). Similarly, Equation 1.4 on page 11, which defined the global pose function as the sum of the local pose functions, can be written as

$$E(t, \mathring{\mathbf{q}}) = \sum_i E_i(t, \mathring{\mathbf{q}}) \quad (4.3)$$

where the index, i , varies from 1 to the number of sensory measurements.

4.2 Minimizing the Global Pose Distance Function

The minimum of the global pose function will define our best guess of object pose, but in general this function will not be in a form in which the minimum can be found analytically; numerical techniques must be used instead. What we have is called a *nonlinear optimization* problem with a nonlinear equality constraint (remember $\mathring{\mathbf{q}}$ must have unit length) [28,45]. The function to be minimized, $E(t, \mathring{\mathbf{q}})$, is called the *objective* function, and the constraint is written in terms of a *constraint* function which must equal zero; that is

$$c(\mathring{\mathbf{q}}) = \mathring{\mathbf{q}}^T \mathring{\mathbf{q}} - 1 = 0 \quad (4.4)$$

There are commercially available software packages which are efficient at solving this type of problem. They typically require the user to provide routines for evaluating the

objective function and its derivatives and the constraint functions and their derivatives for given values of the parameters. For the package we will be using (NPSOL), only the first order derivatives are required.

For our problem, the constraint function, Equation 4.4 on page 42, and its derivatives,

$$\frac{\partial c(\mathbf{\hat{q}})}{\partial \mathbf{\hat{q}}} = 2 \mathbf{\hat{q}} \quad (4.5)$$

are easy to evaluate.¹ The evaluation of the objective function and its derivatives requires a bit more work.

Because the global pose function is defined as the sum of the local pose functions, we can evaluate it and its derivatives for a given $\mathbf{t}$ and $\mathbf{\hat{q}}$ by evaluating the individual local pose functions for that $\mathbf{t}$ and $\mathbf{\hat{q}}$ and then summing the results. That is, in addition to Equation 4.3, we have

$$\frac{dE(\mathbf{P})}{dP_k} = \sum_i \frac{dE_i(\mathbf{P})}{dP_k} \quad (4.6)$$

where P_k refers to the k th parameter of the vector $\mathbf{P}$.² The remainder of this chapter will focus on these local functions.

4.3 Evaluating Local Pose Functions

The evaluation of a local pose function for a given $\mathbf{t}$ and $\mathbf{\hat{q}}$ is fairly straightforward: find the object feature parameters, $\mathbf{x}_i$ and $\mathbf{u}_i$, using Equations 2.4, 2.5, and 2.10 (pages 17 and 20), and then evaluate $f_i(\mathbf{x}_i, \mathbf{u}_i)$ using Equation 4.1 together with the appropriate constraint set given in Figure 3.3 on page 33. For some of the problems, the

¹ i.e., $\frac{\partial c(\mathbf{\hat{q}})}{\partial q_m} = \frac{\partial}{\partial q_m} \sum_{k=1}^4 q_k^2 - 1 = \sum_{k=1}^4 2q_k \delta_{km} = 2q_m$

² For convenience we will often refer to the parameter vectors using $\mathbf{P}$, $\mathbf{X}_i$, and $\mathbf{Y}_i$.

constraint set leads to a closed form for $f_i(\mathbf{x}_i, \mathbf{u}_i)$, but for others numerical techniques must be used to evaluate this function for each particular $\mathbf{x}_i$ and $\mathbf{u}_i$. This evaluation will be discussed for each type of problem later in this chapter.

4.4 Evaluating Derivatives of Local Pose Functions

In order to find the derivative of a local pose function with respect to P_k we will use the chain rule with Equation 1.4 on page 11; i.e.,

$$\frac{dE_i}{d\mathbf{P}} = \frac{df_i}{d\mathbf{P}} = \frac{\partial \mathbf{X}_i^T}{\partial \mathbf{P}} \frac{\partial f_i}{\partial \mathbf{X}_i} + \frac{\partial f_i}{\partial \mathbf{P}} \quad (4.7)$$

where we have introduced the notation [51] $\frac{\partial \mathbf{y}^T}{\partial \mathbf{z}}$ to represent the m (dimension of $\mathbf{z}$) by n (dimension of $\mathbf{y}$) matrix

$$\begin{pmatrix} \frac{\partial y_1}{\partial z_1} & \frac{\partial y_2}{\partial z_1} & \dots & \frac{\partial y_n}{\partial z_1} \\ \frac{\partial y_1}{\partial z_2} & \frac{\partial y_2}{\partial z_2} & \dots & \frac{\partial y_n}{\partial z_2} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y_1}{\partial z_m} & \frac{\partial y_2}{\partial z_m} & \dots & \frac{\partial y_n}{\partial z_m} \end{pmatrix}$$

Because our function f_i does not directly contain the parameters of $\mathbf{P}$, the last term of Equation 4.7 is zero and we have

$$\frac{dE_i}{d\mathbf{P}} = \frac{\partial \mathbf{X}_i^T}{\partial \mathbf{P}} \frac{\partial f_i}{\partial \mathbf{X}_i} \quad (4.8)$$

which is a matrix product of the individual derivatives.³

We are left with two tasks: to determine $\frac{\partial \mathbf{X}_i^T}{\partial \mathbf{P}}$, which does not depend on the problem type, and to determine $\frac{\partial f_i}{\partial \mathbf{X}_i}$, which does. We will address the former problem here and will reserve the latter for later sections which deal with the sensing types individually.

³ Notice Equation 4.8 represents seven scalar equations.

To determine $\frac{\partial \mathbf{X}_i^T}{\partial \mathbf{P}}$ we will partition it into

$$\frac{\partial \mathbf{X}_i^T}{\partial \mathbf{P}} = \begin{pmatrix} \frac{\partial \mathbf{x}_i^T}{\partial \mathbf{t}} & \frac{\partial \mathbf{u}_i^T}{\partial \mathbf{t}} \\ \frac{\partial \mathbf{x}_i^T}{\partial \dot{\mathbf{q}}} & \frac{\partial \mathbf{u}_i^T}{\partial \dot{\mathbf{q}}} \end{pmatrix} \quad (4.9)$$

and find each partition individually using Equations 2.4, 2.5, and 2.10 (pages 17 and 20).

We begin with the partials with respect to $\mathbf{t}$, which are the easiest to determine.

Taking the partial of the transpose of Equation 2.4 (page 17) with respect to $\mathbf{t}$ we have

$$\frac{\partial \mathbf{x}_i^T}{\partial \mathbf{t}} = \frac{\partial}{\partial \mathbf{t}} (\mathbf{r}_i^T \mathbf{Q}^T + \mathbf{t}^T) \quad (4.10)$$

$$= \frac{\partial}{\partial \mathbf{t}} (\mathbf{r}_i^T \mathbf{Q}^T) + \frac{\partial}{\partial \mathbf{t}} (\mathbf{t}^T) \quad (4.11)$$

$$= \frac{\partial \mathbf{t}^T}{\partial \mathbf{t}} \quad (4.12)$$

where the first term was deleted because it did not involve $\mathbf{t}$, and its derivative with respect to $\mathbf{t}$ is therefore zero. The remaining term is clearly equal to the identity matrix⁴ and we have

$$\frac{\partial \mathbf{x}_i^T}{\partial \mathbf{t}} = \mathbf{I} \quad (4.13)$$

where $\mathbf{I}$ is the identity matrix.

For the partial of $\mathbf{u}_i$ with respect to $\mathbf{t}$, we see from Equation 2.5 (page 17) that $\mathbf{u}_i$ is not a function of $\mathbf{t}$ and can immediately write

$$\frac{\partial \mathbf{u}_i^T}{\partial \mathbf{t}} = \mathbf{0} \quad (4.14)$$

where $\mathbf{0}$ here represents a 3 by 3 matrix of zeros.

⁴ i.e., $\frac{\partial t_k}{\partial t_m} = \delta_{km}$ where δ_{km} is the Kronecker delta function defined by

$$\delta_{km} = \begin{cases} 0 & (k \neq m) \\ 1 & (k = m) \end{cases}$$

To determine the partial of $\mathbf{x}_i^T$ with respect to $\overset{\circ}{\mathbf{q}}$, we again use Equation 2.4, but this time we will first write the individual elements of $\mathbf{x}_i$ as

$$x_{i_k} = r_{i_1}Q_{k1} + r_{i_2}Q_{k2} + r_{i_3}Q_{k3} + t_k \quad (4.15)$$

where Q_{kl} refers to the element at row k , column l of the matrix $\mathbf{Q}$ given by Equation 2.10 on page 20. Taking the partial of x_{i_k} with respect to element m of $\overset{\circ}{\mathbf{q}}$ ⁵ we get

$$\frac{\partial x_{i_k}}{\partial q_m} = r_{i_1} \frac{\partial Q_{k1}}{\partial q_m} + r_{i_2} \frac{\partial Q_{k2}}{\partial q_m} + r_{i_3} \frac{\partial Q_{k3}}{\partial q_m} \quad (4.16)$$

as the entry at row m , column k of the 4 by 3 matrix $\frac{\partial \mathbf{x}_i^T}{\partial \overset{\circ}{\mathbf{q}}}$. Because of the simple quadratic form of the elements Q_{kl} , this operation can be done by inspection, although it is somewhat tedious. We will give the result only, as

$$\frac{\partial \mathbf{x}_i^T}{\partial \overset{\circ}{\mathbf{q}}} = 2 \begin{pmatrix} r_{i_1}q_0 - r_{i_2}q_z + r_{i_3}q_y & r_{i_1}q_z + r_{i_2}q_0 - r_{i_3}q_x & -r_{i_1}q_y + r_{i_2}q_x + r_{i_3}q_0 \\ r_{i_1}q_x + r_{i_2}q_y + r_{i_3}q_z & r_{i_1}q_y - r_{i_2}q_x - r_{i_3}q_0 & r_{i_1}q_z + r_{i_2}q_0 - r_{i_3}q_x \\ -r_{i_1}q_y + r_{i_2}q_x + r_{i_3}q_0 & r_{i_1}q_x + r_{i_2}q_y + r_{i_3}q_z & -r_{i_1}q_0 + r_{i_2}q_z - r_{i_3}q_y \\ -r_{i_1}q_z - r_{i_2}q_0 + r_{i_3}q_x & r_{i_1}q_0 - r_{i_2}q_z + r_{i_3}q_y & r_{i_1}q_x + r_{i_2}q_y + r_{i_3}q_z \end{pmatrix} \quad (4.17)$$

Using Equation 2.5 we can perform a similar feat to determine the partial of $\mathbf{u}_i^T$ with respect to $\overset{\circ}{\mathbf{q}}$. We note that the only difference between Equation 2.4 and Equation 2.5, as far as this partial is concerned, is the replacement of the vector $\mathbf{r}_i$ with the vector $\mathbf{w}_i$. We can therefore immediately write

$$\frac{\partial \mathbf{u}_i^T}{\partial \overset{\circ}{\mathbf{q}}} = 2 \begin{pmatrix} w_{i_1}q_0 - w_{i_2}q_z + w_{i_3}q_y & w_{i_1}q_z + w_{i_2}q_0 - w_{i_3}q_x & -w_{i_1}q_y + w_{i_2}q_x + w_{i_3}q_0 \\ w_{i_1}q_x + w_{i_2}q_y + w_{i_3}q_z & w_{i_1}q_y - w_{i_2}q_x - w_{i_3}q_0 & w_{i_1}q_z + w_{i_2}q_0 - w_{i_3}q_x \\ -w_{i_1}q_y + w_{i_2}q_x + w_{i_3}q_0 & w_{i_1}q_x + w_{i_2}q_y + w_{i_3}q_z & -w_{i_1}q_0 + w_{i_2}q_z - w_{i_3}q_y \\ -w_{i_1}q_z - w_{i_2}q_0 + w_{i_3}q_x & w_{i_1}q_0 - w_{i_2}q_z + w_{i_3}q_y & w_{i_1}q_x + w_{i_2}q_y + w_{i_3}q_z \end{pmatrix} \quad (4.18)$$

⁵ Note we use two different notations to refer to the elements of $\overset{\circ}{\mathbf{q}}$: when we think of it as a regular vector we use $\overset{\circ}{\mathbf{q}} = (q_1, q_2, q_3, q_4)^T$ for convenience; when we think of it as a quaternion we use $\overset{\circ}{\mathbf{q}} = (q_0, q_x, q_y, q_z)^T$ to reflect the meaning of each element.

Equations 4.13, 4.14, 4.17, and 4.18 are substituted into Equation 4.9 to complete our definition of the partial of $\mathbf{X}_i^T$ with respect to $\mathbf{P}$. The partial of f_i with respect to $\mathbf{X}_i$ for each type of sensing will be given in the following section.

4.5 Object Feature Distance Functions

At this point all that remains to be found is a method for each class of sensing for evaluating the object feature distance function, $f_i(\mathbf{x}_i, \mathbf{u}_i)$ and its derivatives,

$$\frac{\partial f_i}{\partial \mathbf{X}_i} = \begin{pmatrix} \frac{\partial f_i}{\partial \mathbf{x}_i} \\ \frac{\partial f_i}{\partial \mathbf{u}_i} \end{pmatrix} \quad (4.19)$$

for given values of $\mathbf{x}_i$ and $\mathbf{u}_i$. This is done using Equation 4.1 on page 41 along with the constraints given in Figure 3.3 on page 33, where the sensory feature distance functions, $g_i(\mathbf{y}_i, \mathbf{v}_i)$, are defined by Equation 3.18 on page 33 (recall the coefficients $\mathbf{C}_i^{-1}$, $\boldsymbol{\mu}_i$, and $\mathbf{M}_i$ are generated from the sensory data). Each class of sensing will create its own form of $f_i(\mathbf{x}_i, \mathbf{u}_i)$; some will have a closed form, while others will not. Each is discussed individually in the following subsections.

4.5.1 Point-Point Problems

The point-point problems are by far our simplest problems. Using Equations 4.1 and 3.18 (pages 41 and 33) and the constraints for point-point problems given in Figure 3.3 (page 33), our object feature distance function can be written as

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \\ &\text{subject to } \mathbf{y}_i = \mathbf{x}_i \end{aligned} \quad (4.20)$$

where we have dropped the term $\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i$ because, for this type of problem, $\mathbf{M}_i$ is set to the zero matrix.

Equation 4.20 represents a trivial minimization problem because the constraint completely binds the free variable, $\mathbf{y}_i$. We can therefore immediately write

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.21)$$

for the object feature distance function for this type of problem. Note this distance function is not actually a function of the arbitrary unit vector, $\mathbf{u}_i$.

All that remains is to determine the partial derivatives of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ with respect to $\mathbf{x}_i$ and $\mathbf{u}_i$. Because $\mathbf{C}_i^{-1}$ is a symmetric constant matrix, the partial of this quadratic with respect to $\mathbf{x}_i$ is given by⁶

$$\frac{\partial f_i}{\partial \mathbf{x}_i} = 2\mathbf{C}_i^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.22)$$

and the partial with respect to $\mathbf{u}_i$ is given by

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = 0 \quad (4.23)$$

because $f_i(\mathbf{x}_i, \mathbf{u}_i)$, for this type of problem, is not actually a function of $\mathbf{u}_i$.

4.5.2 Point-Line Problems

To generate the distance function for the point-line problems we will again use Equations 4.1 and 3.18 (pages 41 and 33) along with the appropriate constraint set given in Figure 3.3 (page 33) to produce

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \\ &\text{subject to } \mathbf{y}_i = \mathbf{x}_i + k\mathbf{u}_i \end{aligned} \quad (4.24)$$

⁶ We will often use in this thesis the relation [51]

$$\frac{\partial}{\partial \mathbf{z}} (\mathbf{h}^T \mathbf{A} \mathbf{h}) = 2 \frac{\partial \mathbf{h}^T}{\partial \mathbf{z}} \mathbf{A} \mathbf{h}$$

where $\mathbf{h}$ is a vector function of $\mathbf{z}$, and $\mathbf{A}$ is a symmetric matrix which is not a function of $\mathbf{z}$. In the above instance, $\mathbf{h}(\mathbf{z}) = \mathbf{z} + \mathbf{c}$, where $\mathbf{c}$ is a constant, so $\frac{\partial \mathbf{h}^T}{\partial \mathbf{z}} = \mathbf{I}$.

where we have again dropped the term $\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i$ because $\mathbf{M}_i$ is the zero matrix. We note that for the constraint for this problem, the free variable k represents the only degree of freedom of $\mathbf{y}_i$. This allows us to perform the substitution

$$\mathbf{y}_i = \mathbf{x}_i + k\mathbf{u}_i \quad (4.25)$$

and instead minimize over the free variable k . Thus we have

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \min_k (\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i) \quad (4.26)$$

which is an unconstrained minimization problem.

To perform this minimization we take the derivative of the objective function with respect to k , and set it equal to zero. Then we solve for k and substitute the resulting expression back into the objective function. In this manner, we have

$$\frac{d}{dk} \left((\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i) \right) = 0 \quad (4.27)$$

from which we carry out the derivative (see Footnote 6 on page 48) to obtain

$$2\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i) = 0 \quad (4.28)$$

For this scalar equation, we can solve for k by separating the terms,

$$k\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i = -\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.29)$$

and dividing by the scalar $\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i$ to get

$$k = -\frac{\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \quad (4.30)$$

Now substituting this into the objective function of Equation 4.26, we obtain

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \left(\mathbf{x}_i - \boldsymbol{\mu}_i - \frac{\mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right)^T \mathbf{C}_i^{-1} \left(\mathbf{x}_i - \boldsymbol{\mu}_i - \frac{\mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) \quad (4.31)$$

where we have regrouped the terms. Note that in terms of matrix multiplication, the product $k\mathbf{u}_i$ is actually carried as $\mathbf{u}_i k$.

To simplify this expression we factor out $(\mathbf{x}_i - \boldsymbol{\mu}_i)$ from each end,

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right)^T \mathbf{C}_i^{-1} \left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.32)$$

and expand,⁷

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \left(\mathbf{C}_i^{-1} - \frac{\mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} - \frac{\mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} + \frac{\mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{(\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i)^2} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.33)$$

We note that if we factor the scalar $\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i$ from the numerator and denominator of the term

$$\frac{\mathbf{C}_i^{-1} \mathbf{u}_i (\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i) \mathbf{u}_i^T \mathbf{C}_i^{-1}}{(\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i)^2} \quad (4.34)$$

then this term and one of the previous two terms will cancel, and we have

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \left(\mathbf{C}_i^{-1} - \frac{\mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.35)$$

as our final expression for $f_i(\mathbf{x}_i, \mathbf{u}_i)$ for the point-line problems.

The partial of this equation with respect to $\mathbf{x}_i$ is easy to obtain if we note the matrix expression in the middle is not a function of $\mathbf{x}_i$. Therefore we can again use the relation given in Footnote 6 on page 48 to write

$$\frac{\partial f_i}{\partial \mathbf{x}_i} = 2 \left(\mathbf{C}_i^{-1} - \frac{\mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.36)$$

as our expression for the partial of f_i with respect to $\mathbf{x}_i$.

The partial of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ with respect to $\mathbf{u}_i$ will require a little more work. First we write $f_i(\mathbf{x}_i, \mathbf{u}_i)$ of Equation 4.35 as

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i) - (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \left(\frac{\mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.37)$$

⁷ Remember $\mathbf{C}_i^{-1}$ is a symmetric matrix, so $(\mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1})^T = \mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T$.

and note the first term is not a function of $\mathbf{u}_i$, so it will not contribute to the partial.

We now have

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = -\frac{\partial}{\partial \mathbf{u}_i} \frac{(\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \quad (4.38)$$

Because $\mathbf{C}_i^{-1}$ is symmetric, the numerator of this function contains two identical scalar functions of $\mathbf{u}_i$:

$$(\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{u}_i = \mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.39)$$

We can therefore write the partial as

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = -\frac{\partial}{\partial \mathbf{u}_i} \frac{(\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i))^2}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \quad (4.40)$$

$$= -\frac{\partial}{\partial \mathbf{u}_i} \left((\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i))^2 (\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i)^{-1} \right) \quad (4.41)$$

which represents the derivative (with respect to $\mathbf{u}_i$) of the product of two scalar functions. Using the product rule for derivatives, this becomes

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = -\frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i)^{-1} (\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i))^2 - \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i))^2 (\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i)^{-1} \quad (4.42)$$

(note the partials must be kept on the left of each term). To take each partial, we first use the chain rule:

$$\begin{aligned} \frac{\partial f_i}{\partial \mathbf{u}_i} &= \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i) (\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i)^{-2} (\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i))^2 \\ &\quad - 2 \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)) (\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)) (\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i)^{-1} \end{aligned} \quad (4.43)$$

$$\begin{aligned} &= \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i) \left(\frac{\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right)^2 \\ &\quad - 2 \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)) \left(\frac{\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) \end{aligned} \quad (4.44)$$

To take the derivative appearing in the first term, we use the rule for taking the deriva-

tive of a scalar product,

$$\frac{\partial}{\partial \mathbf{u}_i} \mathbf{u}_i^T \mathbf{z} = \mathbf{z} \quad (4.45)$$

where $\mathbf{z}$ represents the vector $\mathbf{C}_i^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_i)$ (which is not a function of $\mathbf{u}_i$). For the derivative appearing in the second term, we use the identity given in Footnote 6 on page 48. This produces

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = 2\mathbf{C}_i^{-1} \mathbf{u}_i \left(\frac{\mathbf{u}_i^T \mathbf{C}_i^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right)^2 - 2\mathbf{C}_i^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_i) \left(\frac{\mathbf{u}_i^T \mathbf{C}_i^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) \quad (4.46)$$

as our final expression for $\frac{\partial f_i}{\partial \mathbf{u}_i}$. Note the big parentheses in this equation contain equivalent scalar functions.

4.5.3 Point-Plane Problems

To create the distance function for the point-plane problems we again use Equations 4.1 and 3.18 (pages 41 and 33) along with the appropriate constraint set given in Figure 3.3 (page 33). This produces

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \\ &\text{subject to } (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \end{aligned} \quad (4.47)$$

where the term $\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i$ has again been dropped because $\mathbf{M}_i$ is the zero matrix.

With this problem we cannot remove the constraint by simply substituting it into the objective function like we did for the point-line problem; we must solve this as a constrained minimization problem. Classical techniques employing Lagrange multipliers [28,45] can be used for this minimization. We will state the general method before applying it to this problem, in order that we may refer back to this subsection in solving other types of sensing.

We will first restate our general problem defined by Equation 4.1 using the parameters $\mathbf{X}_i$ and $\mathbf{Y}_i$ for notational convenience. We have defined $f_i(\mathbf{X}_i)$ by

$$\begin{aligned} f_i(\mathbf{X}_i) &= \min_{\mathbf{Y}_i} g_i(\mathbf{Y}_i) \\ &\text{subject to } \mathbf{h}_i(\mathbf{X}_i, \mathbf{Y}_i) = \mathbf{0} \end{aligned} \quad (4.48)$$

where $g_i(\mathbf{Y}_i)$ is defined by Equation 3.18, and the vector valued function $\mathbf{h}_i(\mathbf{X}_i, \mathbf{Y}_i)$ refers to the set of m constraint functions,

$$\mathbf{h}_i(\mathbf{X}_i, \mathbf{Y}_i) = \begin{pmatrix} h_{i1}(\mathbf{X}_i, \mathbf{Y}_i) \\ h_{i2}(\mathbf{X}_i, \mathbf{Y}_i) \\ \vdots \\ h_{im}(\mathbf{X}_i, \mathbf{Y}_i) \end{pmatrix} \quad (4.49)$$

which are represented in Figure 3.3 on page 33 (rewritten as functions which are to equal zero).

We note the minimization operation in effect causes the objective function to be evaluated for a specific value of $\mathbf{Y}_i$, and the value of $\mathbf{Y}_i$ chosen, $\mathbf{Y}_i^*$, will depend on both the objective function and the set of constraint functions. Because our constraints are functions of $\mathbf{X}_i$, $\mathbf{Y}_i^*$ will be also. We can therefore write

$$f_i(\mathbf{X}_i) = g_i(\mathbf{Y}_i^*(\mathbf{X}_i)) \quad (4.50)$$

The problem now is of course to determine $\mathbf{Y}_i^*(\mathbf{X}_i)$.

The method of Lagrange multipliers [28,45] provides criteria for selecting $\mathbf{Y}_i^*$ based on the objective and constraint functions. This method produces a set of equations which must be satisfied at the constrained minimum. We write this set as

$$\frac{\partial g_i}{\partial \mathbf{Y}_i} - \frac{\partial \mathbf{h}_i^T}{\partial \mathbf{Y}_i} \boldsymbol{\lambda} = \mathbf{0} \quad (4.51)$$

where $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m)$ is a vector of unknown Lagrange multipliers. We note that if $\mathbf{Y}_i$ is of dimension k , then this represents a set of k scalar equations. If we combine this set with the set of m equations represented by our constraint functions,

$$\mathbf{h}_i(\mathbf{X}_i, \mathbf{Y}_i) = \mathbf{0} \quad (4.52)$$

then we have a system of $k + m$ equations and $k + m$ unknowns ($\mathbf{Y}_i$ and λ ; remember $\mathbf{X}_i$ is assumed to be given). $\mathbf{Y}_i^*$ must be a solution to this system. When it is determined, it substituted into Equation 4.50.

For the point-plane problems, we apply this method to Equation 4.47 to get a system of equations consisting of the constraint equation,

$$(\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \quad (4.53)$$

which is equivalent to

$$\mathbf{y}_i^T \mathbf{u}_i = \mathbf{x}_i^T \mathbf{u}_i \quad (4.54)$$

and the three equations represented by

$$\frac{\partial}{\partial \mathbf{y}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) - \lambda_1 \frac{\partial}{\partial \mathbf{y}_i} (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \quad (4.55)$$

which is equivalent to

$$2\mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) - \lambda_1 \mathbf{u}_i = \mathbf{0} \quad (4.56)$$

and which can also be written as

$$(\mathbf{y}_i - \boldsymbol{\mu}_i) = \frac{\lambda_1}{2} \mathbf{C}_i \mathbf{u}_i \quad (4.57)$$

where we have premultiplied by $\frac{1}{2} \mathbf{C}_i$.⁸

⁸ Because $\mathbf{C}_i^{-1}$ is real, symmetric, and positive definite, it cannot be singular, and its inverse, $\mathbf{C}_i$, will therefore exist.

Now from the set of equations given by Equations 4.54 and 4.57 we will solve for $\mathbf{y}_i - \boldsymbol{\mu}_i$ in terms of $\mathbf{x}_i$, $\mathbf{u}_i$, $\mathbf{C}_i$, and $\boldsymbol{\mu}_i$ and then substitute this expression into the objective function of Equation 4.47 to generate $f_i(\mathbf{x}_i, \mathbf{u}_i)$. We first solve for $\lambda_1/2$ in terms of the other variables by premultiplying Equation 4.57 by $\mathbf{u}_i^T$ to get

$$\mathbf{u}_i^T(\mathbf{y}_i - \boldsymbol{\mu}_i) = \frac{\lambda_1}{2} \mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i \quad (4.58)$$

a scalar equation which can be divided by $\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i$ to produce

$$\frac{\lambda_1}{2} = \frac{\mathbf{u}_i^T(\mathbf{y}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.59)$$

Before we substitute this expression into Equation 4.57, we would like to somehow remove its dependence on $\mathbf{y}_i$. This can be done if we note that according to Equation 4.54, the scalar product $\mathbf{u}_i^T \mathbf{y}_i$ can be replaced by $\mathbf{u}_i^T \mathbf{x}_i$, and we therefore have

$$\frac{\lambda_1}{2} = \frac{\mathbf{u}_i^T(\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.60)$$

Now we can substitute this expression into Equation 4.57 to get

$$(\mathbf{y}_i - \boldsymbol{\mu}_i) = \mathbf{C}_i \mathbf{u}_i \frac{\mathbf{u}_i^T(\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.61)$$

$$= \frac{\mathbf{C}_i \mathbf{u}_i \mathbf{u}_i^T(\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.62)$$

where we have put the scalar term on the right of the vector $\mathbf{C}_i \mathbf{u}_i$ in order that we can carry out the multiplication in a matrix sense. This expression for $(\mathbf{y}_i - \boldsymbol{\mu}_i)$ can now be substituted into the objective function of Equation 4.47 to produce

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \left(\frac{\mathbf{C}_i \mathbf{u}_i \mathbf{u}_i^T(\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right)^T \mathbf{C}_i^{-1} \left(\frac{\mathbf{C}_i \mathbf{u}_i \mathbf{u}_i^T(\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right) \quad (4.63)$$

In performing the transpose on the left side, we note that because $\mathbf{C}_i^{-1}$ is a symmetric matrix, $\mathbf{C}_i$ will be also, so $\mathbf{C}_i^T$ is the same as $\mathbf{C}_i$. We therefore have

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \left(\frac{(\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right) \mathbf{C}_i^{-1} \left(\frac{\mathbf{C}_i \mathbf{u}_i \mathbf{u}_i^T(\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right) \quad (4.64)$$

which (noting the denominators are scalar) can be written as

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \frac{(\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i \mathbf{C}_i^{-1} \mathbf{C}_i \mathbf{u}_i \mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i)}{(\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^2} \quad (4.65)$$

$$= \frac{(\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i \mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i)}{(\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^2} \quad (4.66)$$

We can decompose the numerator of this function into three scalar functions,

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \frac{\left((\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{u}_i \right) \left(\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i \right) \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)}{(\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^2} \quad (4.67)$$

and divide both the numerator and denominator by the scalar $\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i$ to get

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \frac{\left((\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{u}_i \right) \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.68)$$

$$= \frac{\left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)^2}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.69)$$

as our final expression for $f_i(\mathbf{x}_i, \mathbf{u}_i)$.

Now we need to find the partial derivatives of this function with respect to $\mathbf{x}_i$ and $\mathbf{u}_i$. Just as it was with the point-line problems, the partial with respect to $\mathbf{x}_i$ is easy to obtain. If we write Equation 4.68 as

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \left(\frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.70)$$

and note the matrix term in the middle is not a function of $\mathbf{x}_i$, then we can use the relation given in Footnote 6 on page 48, to get

$$\frac{\partial f_i}{\partial \mathbf{x}_i} = 2 \left(\frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.71)$$

$$= 2 \frac{\mathbf{u}_i \mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.72)$$

as the expression for this partial with respect to $\mathbf{X}_i$.

To find the partial of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ with respect to $\mathbf{u}_i$, we first write (using Equation 4.69) the expression

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = \frac{\partial}{\partial \mathbf{u}_i} \frac{\left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)^2}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.73)$$

and note its similarity to Equation 4.40 on page 51, which applied to the point-line problems. The two equations are in the same form as far as the variable $\mathbf{u}_i$ is concerned, so the procedure for solving for the derivative will be identical. We will repeat the procedure here, but will abbreviate the explanation.

We first write the function as a product,

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = \frac{\partial}{\partial \mathbf{u}_i} \left(\left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)^2 (\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^{-1} \right) \quad (4.74)$$

and use the product rule for derivatives,

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = \frac{\partial}{\partial \mathbf{u}_i} \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)^2 (\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^{-1} + \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^{-1} \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)^2 \quad (4.75)$$

and then the chain rule,

$$\begin{aligned} \frac{\partial f_i}{\partial \mathbf{u}_i} &= 2 \frac{\partial}{\partial \mathbf{u}_i} \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right) \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right) (\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^{-1} \\ &\quad - \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i) (\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^{-2} \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)^2 \end{aligned} \quad (4.76)$$

The remaining partials are now computed to produce

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = 2(\mathbf{x}_i - \boldsymbol{\mu}_i) \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right) (\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^{-1} - 2\mathbf{C}_i \mathbf{u}_i (\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i)^{-2} \left(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i) \right)^2 \quad (4.77)$$

which can be written as

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = 2(\mathbf{x}_i - \boldsymbol{\mu}_i) \left(\frac{\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right) - 2\mathbf{C}_i \mathbf{u}_i \left(\frac{\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right)^2 \quad (4.78)$$

This completes the derivation of the object feature distance function and its first derivatives for the point-plane problems.

4.5.4 Line-Point Problems

The line-point problems provide our first instance of a sensory distance function which cannot be written in closed form. As we will later discover, this will also be the case for the plane-point and plane-line problems. We will begin this subsection with an outline of the method we will use for these three types of problems, giving the general information which is applicable to all three. Then we will provide the specific information for the line-point problem. As the other two types are encountered, we will refer back to this section and will only add the details for that problem type.

The procedure we will use to solve this type of problem requires that we first generate the Lagrange equations for the constrained minimization problem as described in Section 4.5.3. We will then remove λ from the system of equations (by writing its elements as functions of $\mathbf{X}_i$ and $\mathbf{Y}_i$ and then substituting these back into the equations). This transforms our original system of equations to one of k equations (k is the dimension of $\mathbf{Y}_i$) which we write as

$$\mathbf{N}(\mathbf{X}_i, \mathbf{Y}_i) = \mathbf{0} \quad (4.79)$$

For the other six types of sensing, we are able to solve this system of equations explicitly for $\mathbf{Y}_i$ (creating a function of $\mathbf{X}_i$), and can then substitute this *expression* into Equation 4.50 on page 53 to get an explicit expression for $f_i(\mathbf{X}_i)$. Derivatives of this expression are then taken to produce an explicit expression for $\frac{\partial f_i}{\partial \mathbf{X}_i}$. With the three types of sensing we are presently addressing, this is not possible.

We recall our objective is to be able to evaluate the function $f_i(\mathbf{X}_i)$ and its derivatives for *given* values of $\mathbf{X}_i$. We can therefore use numerical techniques⁹ on Equation 4.79

⁹ We can use for example Newton's Method or a Quasi-Newton Method [11].

to solve for $\mathbf{Y}_i^*$ for each given $\mathbf{X}_i$ and can then substitute this *value* into Equation 4.50 (page 53) to produce the desired value of $f_i(\mathbf{X}_i)$. This takes care of the evaluation of $f_i(\mathbf{X}_i)$, but we still need to evaluate its partial with respect to $\mathbf{X}_i$ for that given $\mathbf{X}_i$.

To evaluate the derivative for the problem as we have stated it, we can use the chain rule on Equation 4.50 to get

$$\frac{df_i}{d\mathbf{X}_i} = \frac{\partial \mathbf{Y}_i^T}{\partial \mathbf{X}_i} \frac{\partial g_i}{\partial \mathbf{Y}_i} \quad (4.80)$$

where the partial of g_i with respect to $\mathbf{X}_i$ is not included because it is zero. The objective function, g_i , which we refer to is the one given by Equation 3.18 on page 33; its partials are found to be

$$\frac{\partial g_i}{\partial \mathbf{Y}_i} = \begin{pmatrix} \frac{\partial g_i}{\partial \mathbf{y}_i} \\ \frac{\partial g_i}{\partial \mathbf{v}_i} \end{pmatrix} = \begin{pmatrix} 2\mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) \\ 2\mathbf{M}_i\mathbf{v}_i \end{pmatrix} \quad (4.81)$$

where $\mathbf{Y}_i$ is partitioned into $\mathbf{y}_i$ and $\mathbf{v}_i$. Note the values of $\mathbf{y}_i$ and $\mathbf{v}_i$ were generated in the numerical procedure which was used to evaluate $f_i(\mathbf{X}_i)$.

As we will see later in this section, the line-point problems will use a substitution which puts the objective function in the form

$$g_i(k, \mathbf{v}_i) = (\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1}(\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.82)$$

where $\mathbf{Y}_i$ is now partitioned into k and $\mathbf{v}_i$. For g_i in this form, we see that Equation 4.50 is rewritten as

$$f_i(\mathbf{X}_i) = g_i(\mathbf{X}_i; \mathbf{Y}_i^*(\mathbf{X}_i)) \quad (4.83)$$

and the chain rule operation will produce

$$\frac{df_i}{d\mathbf{X}_i} = \frac{\partial \mathbf{Y}_i^T}{\partial \mathbf{X}_i} \frac{\partial g_i}{\partial \mathbf{Y}_i} + \frac{\partial g_i}{\partial \mathbf{X}_i} \quad (4.84)$$

where the partial of g_i with respect to $\mathbf{X}_i$ is now included, because in general it is not zero for this g_i . Consequently, for line-point problems we see that the partial of g_i with

respect to $\mathbf{Y}_i$ is given by

$$\frac{\partial g_i}{\partial \mathbf{Y}_i} = \begin{pmatrix} \frac{\partial g_i}{\partial k} \\ \frac{\partial g_i}{\partial \mathbf{v}_i} \end{pmatrix} = \begin{pmatrix} 2\mathbf{v}_i^T \mathbf{C}_i^{-1}(\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) \\ 2k\mathbf{C}_i^{-1}(\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + 2\mathbf{M}_i\mathbf{v}_i \end{pmatrix} \quad (4.85)$$

and the partial of g_i with respect to $\mathbf{X}_i$ will be

$$\frac{\partial g_i}{\partial \mathbf{X}_i} = \begin{pmatrix} \frac{\partial g_i}{\partial \mathbf{x}_i} \\ \frac{\partial g_i}{\partial \mathbf{u}_i} \end{pmatrix} = \begin{pmatrix} 2\mathbf{C}_i^{-1}(\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) \\ \mathbf{0} \end{pmatrix} \quad (4.86)$$

As before the values of k and $\mathbf{v}_i$ were generated in the numerical procedure which was used to evaluate $f_i(\mathbf{X}_i)$.

What we still require, in order to use the chain rule given in Equation 4.80 or Equation 4.84, is a procedure for determining the partial of $\mathbf{Y}_i^T$ with respect to $\mathbf{X}_i$. For the three problems we are currently addressing, $\mathbf{Y}_i$ cannot be written as an explicit function of $\mathbf{X}_i$. Consequently, we cannot produce an explicit expression for its partial with respect to $\mathbf{X}_i$. We can, however, evaluate this partial if we use implicit differentiation on the set of equations relating $\mathbf{X}_i$ and $\mathbf{Y}_i$ (Equation 4.79 above). That is, we take the derivative of (the transpose of) this set with respect to $\mathbf{X}_i$ and get

$$\frac{d\mathbf{N}^T}{d\mathbf{X}_i} = \frac{\partial \mathbf{Y}_i^T}{\partial \mathbf{X}_i} \frac{\partial \mathbf{N}^T}{\partial \mathbf{Y}_i} + \frac{\partial \mathbf{N}^T}{\partial \mathbf{X}_i} = \mathbf{0} \quad (4.87)$$

which represents a k by k matrix equation (k is the dimension of $\mathbf{Y}_i$ and $\mathbf{N}$). We point out that the partials of $\mathbf{N}^T(\mathbf{X}_i, \mathbf{Y}_i)$ with respect to $\mathbf{X}_i$ and $\mathbf{Y}_i$ can be explicitly written as functions of $\mathbf{X}_i$ and $\mathbf{Y}_i$, and the values of $\mathbf{X}_i$ and $\mathbf{Y}_i$ are known.¹⁰ We therefore have a linear system of equations from which the partial of $\mathbf{Y}_i$ with respect to $\mathbf{X}_i$ can be

¹⁰ Recall that $\mathbf{Y}_i$ was found using numerical techniques on Equation 4.79.

computed using¹¹

$$\frac{\partial \mathbf{Y}_i^T}{\partial \mathbf{X}_i} = -\frac{\partial \mathbf{N}^T}{\partial \mathbf{X}_i} \left(\frac{\partial \mathbf{N}^T}{\partial \mathbf{Y}_i} \right)^{-1} \quad (4.88)$$

In summary, the above procedure requires that the methods which cannot generate $\mathbf{Y}_i$ as a function of $\mathbf{X}_i$, instead provide a system of equations, $\mathbf{N}(\mathbf{X}_i, \mathbf{Y}_i)$, from which $\mathbf{Y}_i$ can be computed for a given $\mathbf{X}_i$ using numerical techniques. Also required for these problems are the derivatives of $\mathbf{N}(\mathbf{X}_i, \mathbf{Y}_i)$ with respect to $\mathbf{X}_i$ and $\mathbf{Y}_i$ in order that they can be used with Equations 4.80 or 4.84 and 4.88 to evaluate the partial of $f_i(\mathbf{X}_i)$ with respect to $\mathbf{X}_i$.

We will now proceed to generate the required expressions for the line-point problems. Using Equations 4.1 and 3.18 (pages 41 and 33) along with the constraints for the line-point problems given in Figure 3.3 (page 33), we have

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i, \mathbf{v}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \\ &\text{subject to } \mathbf{y}_i = \mathbf{x}_i + k \mathbf{v}_i \\ &\mathbf{v}_i^T \mathbf{v}_i = 1 \end{aligned} \quad (4.89)$$

As was the case for the point-line problems, we can simplify the problem if we note that in the first constraint, the free variable k represents the only degree of freedom of $\mathbf{y}_i$ (and the second constraint does not involve $\mathbf{y}_i$). This allows us to perform the substitution

$$\mathbf{y}_i = \mathbf{x}_i + k \mathbf{v}_i \quad (4.90)$$

and minimize over the free variable k and $\mathbf{v}_i$ rather than $\mathbf{y}_i$ and $\mathbf{v}_i$. Thus we have

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{k, \mathbf{v}_i} (\mathbf{x}_i + k \mathbf{v}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k \mathbf{v}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \\ &\text{subject to } \mathbf{v}_i^T \mathbf{v}_i = 1 \end{aligned} \quad (4.91)$$

¹¹ This is more efficiently implemented using an LU decomposition rather than by taking the matrix inverse directly.

as our minimization problem. Notice that because $\mathbf{u}_i$ is an arbitrary unit vector which is absent from the constraint set, it does not enter into the equations for this type of problem.

Our first task is to use the method of Lagrange multipliers (described in the last subsection) to get a set of equations involving k and $\mathbf{v}_i$. For this type of problem, Equation 4.51 on page 53 produces the following set of four equations

$$\begin{pmatrix} \frac{\partial g_i}{\partial k} \\ \frac{\partial g_i}{\partial \mathbf{v}_i} \end{pmatrix} - \begin{pmatrix} \frac{\partial h_i^T}{\partial k} \\ \frac{\partial h_i^T}{\partial \mathbf{v}_i} \end{pmatrix} \lambda_1 = \mathbf{0} \quad (4.92)$$

where we have partitioned $\mathbf{Y}_i$ into k and $\mathbf{v}_i$, thereby partitioning the set of equations into a single equation and a set of three equations. If we substitute the expressions for g_i and h_i (the objective and constraint functions, respectively), these equations are written

$$\frac{\partial}{\partial k} \left((\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \right) - \frac{\partial}{\partial k} (\mathbf{v}_i^T \mathbf{v}_i - 1) \lambda_1 = 0 \quad (4.93)$$

for the first and

$$\frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \right) - \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{v}_i - 1) \lambda_1 = \mathbf{0} \quad (4.94)$$

for the remaining set of three. If we take the indicated partials and drop the zero terms, these become

$$2\mathbf{v}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) = 0 \quad (4.95)$$

and¹²

$$2k\mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + 2\mathbf{M}_i \mathbf{v}_i - 2\mathbf{v}_i \lambda_1 = \mathbf{0} \quad (4.96)$$

¹² Note Equation 4.95 indicates that $\frac{\partial g_i}{\partial k}$ will be zero in Equation 4.85.

These last two expressions, along with the constraint equation, make up our original set of equations from which $\mathbf{Y}_i$ and λ_1 can be found.

We now need to remove λ_1 from the equations. This is done by premultiplying Equation 4.96 by $\mathbf{v}_i^T$ to produce

$$2k\mathbf{v}_i^T \mathbf{C}_i^{-1}(\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + 2\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i - 2\mathbf{v}_i^T \mathbf{v}_i \lambda_1 = 0 \quad (4.97)$$

The first term of this equation is zero according to Equation 4.95, and the last is $2\lambda_1$ according to the constraint equation. This lets us write λ_1 as

$$\lambda_1 = \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.98)$$

If we substitute this value of λ_1 into Equation 4.96, we get

$$k\mathbf{C}_i^{-1}(\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i = \mathbf{0} \quad (4.99)$$

which can also be written as

$$k\mathbf{C}_i^{-1}(\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + (\mathbf{I} - \mathbf{v}_i \mathbf{v}_i^T) \mathbf{M}_i \mathbf{v}_i = \mathbf{0} \quad (4.100)$$

We can now write a system of equations not involving λ_1 as

$$\mathbf{N}(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) = \begin{pmatrix} \mathbf{N}_1(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) \\ \mathbf{N}_2(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) \end{pmatrix} = \mathbf{0} \quad (4.101)$$

where $\mathbf{N}_1$ represents the constraint equation,

$$\mathbf{N}_1(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) = \mathbf{v}_i^T \mathbf{v}_i - 1 \quad (4.102)$$

and $\mathbf{N}_2(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i)$ represents the last set of equations given above,

$$\mathbf{N}_2(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) = k\mathbf{C}_i^{-1}(\mathbf{x}_i + k\mathbf{v}_i - \boldsymbol{\mu}_i) + (\mathbf{I} - \mathbf{v}_i \mathbf{v}_i^T) \mathbf{M}_i \mathbf{v}_i \quad (4.103)$$

As was previously discussed, this set of equations is used to numerically solve for k and $\mathbf{v}_i$ for given values of $\mathbf{x}_i$ and $\mathbf{u}_i$.

Now we need to find the partials of this equation set with respect to each of the variables. That is, we require

$$\frac{\partial \mathbf{N}^T}{\partial \mathbf{Y}_i} = \begin{pmatrix} \frac{\partial \mathbf{N}_1^T}{\partial k} & \frac{\partial \mathbf{N}_2^T}{\partial k} \\ \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{v}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{v}_i} \end{pmatrix} \quad (4.104)$$

and

$$\frac{\partial \mathbf{N}^T}{\partial \mathbf{X}_i} = \begin{pmatrix} \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{x}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{x}_i} \\ \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{u}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{u}_i} \end{pmatrix} \quad (4.105)$$

We will determine these partials individually.

Because $\mathbf{N}_1$, given by Equation 4.102 is only a function of $\mathbf{v}_i$, its partials with respect to the other variables will be zero. We therefore have

$$\frac{\partial \mathbf{N}_1^T}{\partial k} = 0 \quad (4.106)$$

and

$$\frac{\partial \mathbf{N}_1^T}{\partial \mathbf{x}_i} = \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{u}_i} = 0 \quad (4.107)$$

The partial of this function with respect to $\mathbf{v}_i$ is easily seen to be

$$\frac{\partial \mathbf{N}_1^T}{\partial \mathbf{v}_i} = 2\mathbf{v}_i \quad (4.108)$$

We will now solve for the partials of $\mathbf{N}_2$, given by Equation 4.103. In taking the partial of the transpose of this function with respect to k , we omit the last term (because it does not involve k) and have

$$\frac{\partial \mathbf{N}_2^T}{\partial k} = \frac{\partial}{\partial k} \left(k \mathbf{C}_i^{-1} (\mathbf{x}_i + k \mathbf{v}_i - \boldsymbol{\mu}_i) \right)^T \quad (4.109)$$

$$= \frac{\partial}{\partial k} \left(k (\mathbf{x}_i + k \mathbf{v}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \right) \quad (4.110)$$

$$= \frac{\partial}{\partial k} \left(k^2 \mathbf{v}_i^T \mathbf{C}_i^{-1} + k(\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \right) \quad (4.111)$$

$$= 2k \mathbf{v}_i^T \mathbf{C}_i^{-1} + (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \quad (4.112)$$

$$= (2k \mathbf{v}_i + \mathbf{x}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \quad (4.113)$$

for this partial.

To find the partial of N_2 (given by Equation 4.103) with respect to $\mathbf{v}_i$, we begin with

$$\frac{\partial N_2^T}{\partial \mathbf{v}_i} = \frac{\partial}{\partial \mathbf{v}_i} \left(k \mathbf{C}_i^{-1} (\mathbf{x}_i + k \mathbf{v}_i - \boldsymbol{\mu}_i) + (\mathbf{I} - \mathbf{v}_i \mathbf{v}_i^T) \mathbf{M}_i \mathbf{v}_i \right)^T \quad (4.114)$$

$$= \frac{\partial}{\partial \mathbf{v}_i} \left(k \mathbf{C}_i^{-1} (\mathbf{x}_i + k \mathbf{v}_i - \boldsymbol{\mu}_i) + \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \right)^T \quad (4.115)$$

which, if we take the indicated transpose and note $\mathbf{C}_i^{-1}$ and $\mathbf{M}_i$ are symmetric, becomes

$$\frac{\partial N_2^T}{\partial \mathbf{v}_i} = \frac{\partial}{\partial \mathbf{v}_i} \left(k(\mathbf{x}_i + k \mathbf{v}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} + \mathbf{v}_i^T \mathbf{M}_i - \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T \right) \quad (4.116)$$

$$= \frac{\partial}{\partial \mathbf{v}_i} k^2 \mathbf{v}_i^T \mathbf{C}_i^{-1} + \frac{\partial}{\partial \mathbf{v}_i} \mathbf{v}_i^T \mathbf{M}_i - \frac{\partial}{\partial \mathbf{v}_i} \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T \quad (4.117)$$

where we have dropped the terms which do not involve $\mathbf{v}_i$. To take the partial of the last term with respect to $\mathbf{v}_i$, we factor it into a scalar and a vector function of $\mathbf{v}_i$ and then use the product rule for derivatives. This gives us

$$\frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T) = \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) (\mathbf{v}_i^T) \right) \quad (4.118)$$

$$= \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) (\mathbf{v}_i^T) + \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T) (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.119)$$

$$= 2 \mathbf{M}_i \mathbf{v}_i (\mathbf{v}_i^T) + \mathbf{I} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.120)$$

where the last term has become a scalar times the identity matrix. Now we can substitute this result into Equation 4.117 and use the relation

$$\frac{\partial \mathbf{v}_i^T}{\partial \mathbf{v}_i} = \mathbf{I} \quad (4.121)$$

to produce

$$\frac{\partial \mathbf{N}_2^T}{\partial \mathbf{v}_i} = k^2 \mathbf{C}_i^{-1} + \mathbf{M}_i - 2\mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T - \mathbf{I}(\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.122)$$

$$= k^2 \mathbf{C}_i^{-1} + \mathbf{M}_i(\mathbf{I} - 2\mathbf{v}_i \mathbf{v}_i^T) - \mathbf{I}(\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.123)$$

as our expression for this partial.

Finally, we require the partials of $\mathbf{N}_2^T$ (see Equation 4.103) with respect to $\mathbf{x}_i$ and $\mathbf{u}_i$. The partial with respect to $\mathbf{x}_i$ is given by

$$\frac{\partial \mathbf{N}_2^T}{\partial \mathbf{x}_i} = \frac{\partial}{\partial \mathbf{x}_i} \left(k \mathbf{C}_i^{-1} (\mathbf{x}_i + k \mathbf{v}_i - \boldsymbol{\mu}_i) \right)^T \quad (4.124)$$

$$= \frac{\partial}{\partial \mathbf{x}_i} k \mathbf{x}_i^T \mathbf{C}_i^{-1} \quad (4.125)$$

$$= k \mathbf{C}_i^{-1} \quad (4.126)$$

which is a scalar times a matrix. Because $\mathbf{N}_2^T$ is not a function of $\mathbf{u}_i$ we have

$$\frac{\partial \mathbf{N}_2^T}{\partial \mathbf{u}_i} = \mathbf{0} \quad (4.127)$$

for that partial.

This completes the derivation of the necessary equations for the line-point problems.

4.5.5 Line-Line Problems

Equations 4.1 and 3.18 (pages 41 and 33) and the constraint set for the line-line problems (given in Figure 3.3 on page 33) dictate the object feature distance function for line-line problems be defined by

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i, \mathbf{v}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \\ &\text{subject to } \mathbf{y}_i = \mathbf{x}_i + k \mathbf{u}_i \\ &\mathbf{v}_i = \mathbf{u}_i \\ &\mathbf{v}_i^T \mathbf{v}_i = 1 \end{aligned} \quad (4.128)$$

We immediately notice that with this minimization problem the second constraint completely binds the variable $\mathbf{v}_i$, and the third constraint is therefore unnecessary (recall $\mathbf{u}_i$ will always be a unit vector). In addition we see that, as with the point-line and line-point problems, the free variable k represents the only degree of freedom of $\mathbf{y}_i$. We can therefore perform the substitutions

$$\mathbf{v}_i = \mathbf{u}_i \quad (4.129)$$

and

$$\mathbf{y}_i = \mathbf{x}_i + k\mathbf{u}_i \quad (4.130)$$

in the objective function and change the minimization to be over the free variable k . Thus we have

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \min_k (\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i) + \mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i \quad (4.131)$$

which is an unconstrained minimization problem. Because the last term does not contain the parameter we are minimizing over, k , it can be brought outside the minimization operation, and we are left with

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i + \min_k (\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{x}_i + k\mathbf{u}_i - \boldsymbol{\mu}_i) \quad (4.132)$$

We can save ourselves considerable work if we recognize this function is the sum of two functions: one is a simple function of $\mathbf{u}_i$, and the other is identical to the expression which was applicable to the point-line problems (Equation 4.26 on page 49). The function $f_i(\mathbf{x}_i, \mathbf{u}_i)$ and its derivatives can therefore be found using the results from the point-line problem and adding the contribution from the term $\mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i$. In this way we first use Equation 4.35 on page 50 to write

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i + (\mathbf{x}_i - \boldsymbol{\mu}_i)^T \left(\mathbf{C}_i^{-1} - \frac{\mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.133)$$

as our object feature distance function for the line-line problems. We note that because the added term, $\mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i$, is not a function of $\mathbf{x}_i$, it will not contribute to the partial of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ with respect to $\mathbf{x}_i$, so we can replicate Equation 4.36 (page 50) to get

$$\frac{\partial f_i}{\partial \mathbf{x}_i} = 2 \left(\mathbf{C}_i^{-1} - \frac{\mathbf{C}_i^{-1} \mathbf{u}_i \mathbf{u}_i^T \mathbf{C}_i^{-1}}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) (\mathbf{x}_i - \boldsymbol{\mu}_i) \quad (4.134)$$

as the partial of our function with respect to $\mathbf{x}_i$ for the line-line problem. Finally, using Equation 4.46 on page 52 and the relation

$$\frac{\partial}{\partial \mathbf{u}_i} \mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i = 2 \mathbf{M}_i \mathbf{u}_i \quad (4.135)$$

we have

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = 2 \mathbf{M}_i \mathbf{u}_i + 2 \mathbf{C}_i^{-1} \mathbf{u}_i \left(\frac{\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right)^2 - 2 \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i) \left(\frac{\mathbf{u}_i^T \mathbf{C}_i^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i^{-1} \mathbf{u}_i} \right) \quad (4.136)$$

as our expression for the partial of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ with respect to $\mathbf{u}_i$.

4.5.6 Line-Plane Problems

To determine the object feature distance function for the line-plane problems, we again write Equation 4.1 (page 41), substituting in the expressions for the objective function (Equation 3.18 on page 33) and the constraint equations (Figure 3.3 on page 33), to get

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i, \mathbf{v}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \\ &\text{subject to } (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \\ &\mathbf{v}_i^T \mathbf{u}_i = 0 \\ &\mathbf{v}_i^T \mathbf{v}_i = 1 \end{aligned} \quad (4.137)$$

Before we proceed using Lagrange techniques to solve this constrained minimization problem, we observe this expression represents minimization of a separable sum of two

functions: the first term of the objective function and the first constraint contain $\mathbf{y}_i$ as the only free variable, and the second term of the objective function and the last two constraints contain $\mathbf{v}_i$ as the only free variable. We can therefore write this minimization as a sum of two independent minimizations. That is, we have

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = f'_i(\mathbf{x}_i, \mathbf{u}_i) + f''_i(\mathbf{u}_i) \quad (4.138)$$

where

$$\begin{aligned} f'_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \\ &\text{subject to } (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \end{aligned} \quad (4.139)$$

and

$$\begin{aligned} f''_i(\mathbf{u}_i) &= \min_{\mathbf{v}_i} \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \\ &\text{subject to } \mathbf{v}_i^T \mathbf{u}_i = 0 \\ &\quad \mathbf{v}_i^T \mathbf{v}_i = 1 \end{aligned} \quad (4.140)$$

If we compare the minimization defining f'_i with the minimization we perform for the point-plane problems (given in Equation 4.47 on page 52), we see they are identical. We can therefore use the results of the point-plane problem to define f'_i and its partials with respect to $\mathbf{x}_i$ and $\mathbf{u}_i$. In this section we only need to determine f''_i and its partials. To these values we can add the results from the point-plane problems in order to obtain the final results for the line-plane problems.

Because $f''_i(\mathbf{u}_i)$ is defined by a constrained minimization problem, we will as usual apply the method of Lagrange multipliers to solve it. Using Equation 4.51 (page 53), we first generate the set of equations,

$$\frac{\partial g_i}{\partial \mathbf{v}_i} - \frac{\partial \mathbf{h}_i^T}{\partial \mathbf{v}_i} \boldsymbol{\lambda} = \mathbf{0} \quad (4.141)$$

where g_i is defined by $\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i$, the vector-valued function, $\mathbf{h}_i$, represents the two con-

straint functions,

$$h_{i1}(\mathbf{u}_i, \mathbf{v}_i) = \mathbf{v}_i^T \mathbf{u}_i \quad (4.142)$$

and

$$h_{i2}(\mathbf{u}_i, \mathbf{v}_i) = \mathbf{v}_i^T \mathbf{v}_i - 1 \quad (4.143)$$

and λ is a vector of two dimensions. Substituting these expressions into Equation 4.141, we get the set of equations represented by

$$\frac{\partial}{\partial \mathbf{v}_i}(\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) - \frac{\partial}{\partial \mathbf{v}_i}(\mathbf{v}_i^T \mathbf{u}_i) \lambda_1 - \frac{\partial}{\partial \mathbf{v}_i}(\mathbf{v}_i^T \mathbf{v}_i - 1) \lambda_2 = 0 \quad (4.144)$$

which becomes

$$2\mathbf{M}_i \mathbf{v}_i - \mathbf{u}_i \lambda_1 - 2\mathbf{v}_i \lambda_2 = 0 \quad (4.145)$$

if we take the indicated partials. Notice this represents three equations and five unknowns—the other two equations are given by the constraint equations,

$$\mathbf{v}_i^T \mathbf{u}_i = 0 \quad (4.146)$$

and

$$\mathbf{v}_i^T \mathbf{v}_i - 1 = 0 \quad (4.147)$$

Our first task will be to remove λ_1 from the set of equations. If we premultiply Equation 4.145 by $\mathbf{u}_i^T$ then we have

$$2\mathbf{u}_i^T \mathbf{M}_i \mathbf{v}_i - \mathbf{u}_i^T \mathbf{u}_i \lambda_1 - 2\mathbf{u}_i^T \mathbf{v}_i \lambda_2 = 0 \quad (4.148)$$

The first constraint (Equation 4.146) indicates the last term of this equation is zero.

Therefore if we divide this equation by the scalar $\mathbf{u}_i^T \mathbf{u}_i$, we obtain

$$\lambda_1 = \frac{2\mathbf{u}_i^T \mathbf{M}_i \mathbf{v}_i}{\mathbf{u}_i^T \mathbf{u}_i} \quad (4.149)$$

If we substitute this expression for λ_1 into Equation 4.145 and divide by 2, we get

$$\mathbf{M}_i \mathbf{v}_i - \mathbf{u}_i \frac{\mathbf{u}_i^T \mathbf{M}_i \mathbf{v}_i}{\mathbf{u}_i^T \mathbf{u}_i} - \mathbf{v}_i \lambda_2 = \mathbf{0} \quad (4.150)$$

which can be written as

$$\mathbf{M}_i \mathbf{v}_i - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i \lambda_2 = \mathbf{0} \quad (4.151)$$

or

$$\left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i \lambda_2 = \mathbf{0} \quad (4.152)$$

It can be seen that this expression represents an eigenvalue problem. That is, the solutions for $\mathbf{v}_i$ must be given by the normalized eigenvectors of the matrix

$$\mathbf{B} = \left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \mathbf{M}_i \quad (4.153)$$

while the solutions for λ_2 will be the corresponding eigenvalues. Because $\mathbf{B}$ is a three-dimensional matrix, extraction of the normalized eigenvectors will leave us with three possible solutions for $\mathbf{v}_i$, one of which represents the constrained minimum to our problem.

The eigenvector we select must be the one which satisfies the constraint of Equation 4.146 and which produces the smallest value when inserted into the objective function ($\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i$). Although the constraint did enter into the formation of the eigenvalue problem, if the Lagrange multiplier, λ_1 , turns out to be zero for a particular solution, then this constraint may not be satisfied. This condition will occur at one of the solutions to our eigenvalue problem, one given by λ_2 equal to zero and $\mathbf{v}_i$ equal to $\mathbf{r}_1$ (the eigenvector of $\mathbf{M}_i$ for which the eigenvalue of $\mathbf{M}_i$ is zero).¹³ This solution will always exist for this eigenvalue problem, but it will be ignored unless it satisfies the constraints.

¹³ This is easily seen to be a solution if we note $\mathbf{M}_i \mathbf{r}_1 = \mathbf{0} \mathbf{r}_1 = \mathbf{0}$ and substitute it into Equation 4.152.

We note that if we premultiply Equation 4.152 by $\mathbf{v}_i^T$, we get

$$\mathbf{v}_i^T (\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i}) \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i^T \mathbf{v}_i \lambda_2 = 0 \quad (4.154)$$

which, if we note the constraint of Equation 4.147, can be written as

$$\lambda_2 = \mathbf{v}_i^T (\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i}) \mathbf{M}_i \mathbf{v}_i \quad (4.155)$$

$$= \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i^T \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \mathbf{M}_i \mathbf{v}_i \quad (4.156)$$

$$= \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i - \frac{\mathbf{v}_i^T \mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \mathbf{M}_i \mathbf{v}_i \quad (4.157)$$

Now for solutions in which the constraint of Equation 4.146 is satisfied the last term will be zero, and we have

$$\lambda_2 = \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.158)$$

This tells us that an eigenvalue of $\mathbf{B}$ (given by λ_2) automatically provides us with the evaluation of the objective function for the corresponding eigenvector. As a result, we can choose $\mathbf{v}_i$ to be equal to the eigenvector of $\mathbf{B}$ (for which the constraints are satisfied) which corresponds to the smallest eigenvalue of $\mathbf{B}$. Furthermore, we observe that as an evaluation of the objective function at the minimum, the selected eigenvector also provides us with the value of $f_i''(\mathbf{u}_i)$ (see Equation 4.140).

Now that we have a means of evaluating $f_i''(\mathbf{u}_i)$, we turn to the problem of determining its derivatives with respect to $\mathbf{x}_i$ and $\mathbf{u}_i$. We note that because $\mathbf{x}_i$ does not appear in the definition of f_i'' , the partial with respect to $\mathbf{x}_i$ will be zero, and we write

$$\frac{\partial}{\partial \mathbf{x}_i} f_i''(\mathbf{u}_i) = \mathbf{0} \quad (4.159)$$

To determine the partial of f_i'' with respect to $\mathbf{u}_i$ we will use the technique described in Section 4.5.4. We first write

$$f_i'' = g_i(\mathbf{v}_i) = \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.160)$$

where $\mathbf{v}_i$ is implicitly a function of $\mathbf{u}_i$ as defined by Equation 4.152. Using the chain rule on Equation 4.160 we obtain

$$\frac{\partial f_i''}{\partial \mathbf{u}_i} = \frac{\partial \mathbf{v}_i^T}{\partial \mathbf{u}_i} \frac{\partial}{\partial \mathbf{v}_i} g_i(\mathbf{v}_i) = 2 \frac{\partial \mathbf{v}_i^T}{\partial \mathbf{u}_i} \mathbf{M}_i \mathbf{v}_i \quad (4.161)$$

Now we need to determine the partial of $\mathbf{v}_i^T$ with respect to $\mathbf{u}_i$. We will first substitute the expression for λ_2 (Equation 4.158) into Equation 4.152 to get

$$\left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i}\right) \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i = \mathbf{0} \quad (4.162)$$

or

$$\mathbf{M}_i \mathbf{v}_i - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i = \mathbf{0} \quad (4.163)$$

which is a system of equations we will refer to by

$$\mathbf{N}(\mathbf{u}_i, \mathbf{v}_i) = \mathbf{0} \quad (4.164)$$

We can now use implicit differentiation on the transpose of this system of equations to determine $\frac{\partial \mathbf{v}_i^T}{\partial \mathbf{u}_i}$. That is, using the chain rule, we have

$$\frac{d\mathbf{N}^T}{d\mathbf{u}_i} = \frac{\partial \mathbf{v}_i^T}{\partial \mathbf{u}_i} \frac{\partial \mathbf{N}^T}{\partial \mathbf{v}_i} + \frac{\partial \mathbf{N}^T}{\partial \mathbf{u}_i} = \mathbf{0} \quad (4.165)$$

from which we can determine our partial as

$$\frac{\partial \mathbf{v}_i^T}{\partial \mathbf{u}_i} = -\frac{\partial \mathbf{N}^T}{\partial \mathbf{u}_i} \left(\frac{\partial \mathbf{N}^T}{\partial \mathbf{v}_i} \right)^{-1} \quad (4.166)$$

We must now find expressions for these two partials of $\mathbf{N}^T$.

To determine the partial of $\mathbf{N}^T$ with respect to $\mathbf{v}_i$ we write¹⁴

$$\frac{\partial \mathbf{N}^T}{\partial \mathbf{v}_i} = \frac{\partial}{\partial \mathbf{v}_i} \left(\mathbf{v}_i^T \mathbf{M}_i - \mathbf{v}_i^T \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} - \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T \right) \quad (4.167)$$

$$= \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{M}_i) - \frac{\partial}{\partial \mathbf{v}_i} \left(\mathbf{v}_i^T \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) - \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T) \quad (4.168)$$

¹⁴ Remember $\mathbf{M}_i$ is symmetric so $\mathbf{M}_i = \mathbf{M}_i^T$.

Taking the partials we get

$$\begin{aligned}\frac{\partial \mathbf{N}^T}{\partial \mathbf{v}_i} &= \frac{\partial \mathbf{v}_i^T}{\partial \mathbf{v}_i} \mathbf{M}_i - \frac{\partial \mathbf{v}_i^T}{\partial \mathbf{v}_i} \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \\ &\quad - \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \mathbf{v}_i^T - \frac{\partial}{\partial \mathbf{v}_i} \mathbf{v}_i^T (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i)\end{aligned}\quad (4.169)$$

$$= \mathbf{M}_i - \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} - (2\mathbf{M}_i \mathbf{v}_i) \mathbf{v}_i^T - \mathbf{I}(\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.170)$$

$$= \mathbf{M}_i \left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} - 2\mathbf{v}_i \mathbf{v}_i^T \right) - \mathbf{I}(\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.171)$$

where we have used the product rule for derivatives on the last term.

Finally, we need to determine the partial of $\mathbf{N}^T$ with respect to $\mathbf{u}_i$. We note that only one term of $\mathbf{N}(\mathbf{u}_i, \mathbf{v}_i)$ is a function of $\mathbf{u}_i$, so we have

$$\frac{\partial \mathbf{N}^T}{\partial \mathbf{u}_i} = -\frac{\partial}{\partial \mathbf{u}_i} \left(\mathbf{v}_i^T \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.172)$$

$$= -\frac{\partial}{\partial \mathbf{u}_i} \left((\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i \mathbf{u}_i^T) (\mathbf{u}_i^T \mathbf{u}_i)^{-1} \right) \quad (4.173)$$

If we apply the product rule and the chain rule, this becomes

$$\frac{\partial \mathbf{N}^T}{\partial \mathbf{u}_i} = -\frac{\partial}{\partial \mathbf{u}_i} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i \mathbf{u}_i^T) (\mathbf{u}_i^T \mathbf{u}_i)^{-1} - \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{u}_i)^{-1} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i \mathbf{u}_i^T) \quad (4.174)$$

$$\begin{aligned}&= -\left(\frac{\partial}{\partial \mathbf{u}_i} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i) \mathbf{u}_i^T + \frac{\partial}{\partial \mathbf{u}_i} \mathbf{u}_i^T (\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i) \right) (\mathbf{u}_i^T \mathbf{u}_i)^{-1} \\ &\quad + \frac{\partial}{\partial \mathbf{u}_i} (\mathbf{u}_i^T \mathbf{u}_i) (\mathbf{u}_i^T \mathbf{u}_i)^{-2} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i \mathbf{u}_i^T)\end{aligned}\quad (4.175)$$

$$\begin{aligned}&= -\left(\mathbf{M}_i \mathbf{v}_i \mathbf{u}_i^T + \mathbf{I}(\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i) \right) (\mathbf{u}_i^T \mathbf{u}_i)^{-1} \\ &\quad + 2\mathbf{u}_i (\mathbf{u}_i^T \mathbf{u}_i)^{-2} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i \mathbf{u}_i^T)\end{aligned}\quad (4.176)$$

$$= -\frac{\mathbf{M}_i \mathbf{v}_i \mathbf{u}_i^T + \mathbf{I}(\mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i)}{\mathbf{u}_i^T \mathbf{u}_i} + \frac{2\mathbf{u}_i \mathbf{v}_i^T \mathbf{M}_i \mathbf{u}_i \mathbf{u}_i^T}{(\mathbf{u}_i^T \mathbf{u}_i)^2} \quad (4.177)$$

We will use this last equation to evaluate the partial of $\mathbf{N}^T$ with respect to $\mathbf{u}_i$. This value and the value of the partial of $\mathbf{N}^T$ with respect to $\mathbf{v}_i$ (Equation 4.171) can now be substituted into Equation 4.166 in order to determine the value of $\frac{\partial \mathbf{v}_i^T}{\partial \mathbf{u}_i}$. Then this

value of $\frac{\partial \mathbf{v}_i^T}{\partial \mathbf{u}_i}$ can be substituted into Equation 4.161 to determine the partial of f_i'' with respect to $\mathbf{u}_i$, thus completing the procedure for finding the derivatives of f_i'' .

4.5.7 Plane-Point Problems

As we have done with previous problems, we will define the object feature distance function for the plane-point problems using Equations 4.1 and 3.18 (pages 41 and 33) along with the appropriate constraint set from Figure 3.3 (page 33). This gives us

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i, \mathbf{v}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \\ &\text{subject to } (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i = 0 \\ &\quad \mathbf{v}_i^T \mathbf{v}_i = 1 \end{aligned} \quad (4.178)$$

as the definition of our function. Notice the arbitrary unit vector $\mathbf{u}_i$ does not appear here.

For this constrained minimization problem, our first task is to apply the method of Lagrange multipliers. Using Equation 4.51 on page 53 to generate our equations, we get

$$\begin{pmatrix} \frac{\partial g_i}{\partial \mathbf{y}_i} \\ \frac{\partial g_i}{\partial \mathbf{v}_i} \end{pmatrix} - \begin{pmatrix} \frac{\partial \mathbf{h}_i^T}{\partial \mathbf{y}_i} \\ \frac{\partial \mathbf{h}_i^T}{\partial \mathbf{v}_i} \end{pmatrix} \boldsymbol{\lambda} = \mathbf{0} \quad (4.179)$$

where g_i is our objective function defined by

$$g_i(\mathbf{y}_i, \mathbf{v}_i) = (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.180)$$

the vector-valued function, $\mathbf{h}_i$, represents the two constraint functions,

$$h_{i1}(\mathbf{u}_i, \mathbf{v}_i) = (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i \quad (4.181)$$

and

$$h_{i2}(\mathbf{u}_i, \mathbf{v}_i) = \mathbf{v}_i^T \mathbf{v}_i - 1 \quad (4.182)$$

and λ is a vector of two dimensions. Substituting these expressions into Equation 4.179, produces a set of six equations represented by

$$\begin{aligned} \frac{\partial}{\partial \mathbf{y}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \right) \\ - \frac{\partial}{\partial \mathbf{y}_i} \left((\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i \right) \lambda_1 - \frac{\partial}{\partial \mathbf{y}_i} (\mathbf{v}_i^T \mathbf{v}_i - 1) \lambda_2 = \mathbf{0} \end{aligned} \quad (4.183)$$

which becomes

$$2\mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) - \mathbf{v}_i \lambda_1 = \mathbf{0} \quad (4.184)$$

for the first set of three equations and

$$\begin{aligned} \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \right) \\ - \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i \right) \lambda_1 - \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{v}_i - 1) \lambda_2 = \mathbf{0} \end{aligned} \quad (4.185)$$

which becomes

$$2\mathbf{M}_i \mathbf{v}_i - (\mathbf{y}_i - \mathbf{x}_i) \lambda_1 - 2\mathbf{v}_i \lambda_2 = \mathbf{0} \quad (4.186)$$

for the remaining set of three. These six equations are combined with the two constraint equations to give a system of eight equations and eight unknowns.

First we will remove λ_1 and λ_2 from our equation set by premultiplying Equations 4.184 and 4.186 by $\mathbf{v}_i^T$ and substituting in the constraints of Equations 4.181 and 4.182. For Equation 4.184 we get

$$2\mathbf{v}_i^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) - \mathbf{v}_i^T \mathbf{v}_i \lambda_1 = 0 \quad (4.187)$$

which can be written as

$$\lambda_1 = 2\mathbf{v}_i^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \quad (4.188)$$

For Equation 4.186 we get

$$2\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i^T (\mathbf{y}_i - \mathbf{x}_i) \lambda_1 - 2\mathbf{v}_i^T \mathbf{v}_i \lambda_2 = 0 \quad (4.189)$$

which can be written as

$$\lambda_2 = \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.190)$$

Now by substituting these expressions back into Equations 4.184 and 4.186 and dividing by two, we produce a system of six equations which we will refer to by

$$\mathbf{N}(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) = \begin{pmatrix} \mathbf{N}_1(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) \\ \mathbf{N}_2(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) \end{pmatrix} = \mathbf{0} \quad (4.191)$$

where $\mathbf{N}_1$ is given by the 3-D vector expression,

$$\mathbf{N}_1 = \mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) - \mathbf{v}_i \mathbf{v}_i^T \mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) \quad (4.192)$$

and $\mathbf{N}_2$ is given by the 3-D vector expression,

$$\mathbf{N}_2 = \mathbf{M}_i \mathbf{v}_i - (\mathbf{y}_i - \mathbf{x}_i) \mathbf{v}_i^T \mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) - \mathbf{v}_i \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.193)$$

As was indicated in Section 4.5.4, it will not be possible to write the object feature distance function for the plane-point problems in closed form. We will therefore use the method described in that section to evaluate $f_i(\mathbf{x}_i, \mathbf{u}_i)$ and its partials for given values of $\mathbf{x}_i$ (f_i is not actually a function of $\mathbf{u}_i$ for this type of problem). Recall that $\mathbf{y}_i$ and $\mathbf{v}_i$ are to be computed numerically from Equation 4.191, and the resulting values are substituted into the objective function to provide our value of $f_i(\mathbf{x}_i, \mathbf{u}_i)$.

The problem which remains in this subsection is to determine the values of the partials of $\mathbf{N}^T$ with respect to $\mathbf{Y}_i$ (partitioned into $\mathbf{y}_i$ and $\mathbf{v}_i$) and $\mathbf{X}_i$ (partitioned into $\mathbf{x}_i$ and $\mathbf{u}_i$), so they can be used with Equations 4.80 (page 59), 4.81 (page 59), and 4.88 (page 61) to determine the partials of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ with respect to $\mathbf{x}_i$ and $\mathbf{u}_i$ (note $\frac{\partial f_i}{\partial \mathbf{x}_i}$ will be zero). We will partition these partials into

$$\frac{\partial \mathbf{N}^T}{\partial \mathbf{Y}_i} = \begin{pmatrix} \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{y}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{y}_i} \\ \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{v}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{v}_i} \end{pmatrix} \quad (4.194)$$

and

$$\frac{\partial \mathbf{N}^T}{\partial \mathbf{X}_i} = \begin{pmatrix} \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{x}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{x}_i} \\ \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{u}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{u}_i} \end{pmatrix} \quad (4.195)$$

and will determine these partials individually.

For the partials of $\mathbf{N}_1$ with respect to $\mathbf{y}_i$ and $\mathbf{v}_i$ we use the transpose of Equation 4.192 to write

$$\frac{\partial \mathbf{N}_1}{\partial \mathbf{y}_i} = \frac{\partial}{\partial \mathbf{y}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} - (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \mathbf{v}_i^T \right) \quad (4.196)$$

$$= \frac{\partial}{\partial \mathbf{y}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{I} - \mathbf{v}_i \mathbf{v}_i^T) \right) \quad (4.197)$$

$$= \frac{\partial \mathbf{y}_i^T}{\partial \mathbf{y}_i} \left(\mathbf{C}_i^{-1} (\mathbf{I} - \mathbf{v}_i \mathbf{v}_i^T) \right) \quad (4.198)$$

$$= \mathbf{C}_i^{-1} (\mathbf{I} - \mathbf{v}_i \mathbf{v}_i^T) \quad (4.199)$$

for the partial with respect to $\mathbf{y}_i$ and

$$\frac{\partial \mathbf{N}_1}{\partial \mathbf{v}_i} = \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} - (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \mathbf{v}_i^T \right) \quad (4.200)$$

$$= - \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \mathbf{v}_i^T \right) \quad (4.201)$$

$$= - \frac{\partial}{\partial \mathbf{v}_i} \left(\left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{v}_i^T) \right) \quad (4.202)$$

$$= - \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{v}_i^T) - \frac{\partial}{\partial \mathbf{v}_i} \mathbf{v}_i^T \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.203)$$

$$= - \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \mathbf{v}_i^T - \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.204)$$

$$(4.205)$$

for the partial with respect to $\mathbf{v}_i$. Because $\mathbf{N}_1$ is not a function of $\mathbf{x}_i$ or $\mathbf{u}_i$, the partials with respect to these variables will be zero and we write

$$\frac{\partial \mathbf{N}_1}{\partial \mathbf{x}_i} = \mathbf{0} \quad (4.206)$$

and

$$\frac{\partial \mathbf{N}_1}{\partial \mathbf{u}_i} = \mathbf{0} \quad (4.207)$$

The partials of N_2 with respect to $\mathbf{y}_i$ and $\mathbf{v}_i$ are found using the transpose of Equation 4.193. We have

$$\frac{\partial N_2}{\partial \mathbf{y}_i} = \frac{\partial}{\partial \mathbf{y}_i} \left(\mathbf{v}_i^T \mathbf{M}_i; - (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T - \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T \right) \quad (4.208)$$

$$= - \frac{\partial}{\partial \mathbf{y}_i} \left(\left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{y}_i - \mathbf{x}_i)^T \right) \quad (4.209)$$

$$= - \frac{\partial}{\partial \mathbf{y}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{y}_i - \mathbf{x}_i)^T - \frac{\partial}{\partial \mathbf{y}_i} (\mathbf{y}_i - \mathbf{x}_i)^T \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.210)$$

$$= - \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T - \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.211)$$

for the partial with respect to $\mathbf{y}_i$ and

$$\frac{\partial N_2}{\partial \mathbf{v}_i} = \frac{\partial}{\partial \mathbf{v}_i} \left(\mathbf{v}_i^T \mathbf{M}_i; - (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T - \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T \right) \quad (4.212)$$

$$= \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{M}_i) - \frac{\partial}{\partial \mathbf{v}_i} \left(\left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{y}_i - \mathbf{x}_i)^T \right) - \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \mathbf{v}_i^T \right) \quad (4.213)$$

$$= \mathbf{M}_i - \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{y}_i - \mathbf{x}_i)^T - \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) (\mathbf{v}_i^T) - \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T) (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.214)$$

$$= \mathbf{M}_i - \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) (\mathbf{y}_i - \mathbf{x}_i)^T - 2 \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T - \mathbf{I} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.215)$$

for the partial with respect to $\mathbf{v}_i$. Similarly, the partial of N_2 with respect to $\mathbf{x}_i$ is derived using

$$\frac{\partial N_2}{\partial \mathbf{x}_i} = \frac{\partial}{\partial \mathbf{x}_i} \left(\mathbf{v}_i^T \mathbf{M}_i; - (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T - \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T \right) \quad (4.216)$$

$$= - \frac{\partial}{\partial \mathbf{x}_i} \left(\left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{y}_i - \mathbf{x}_i)^T \right) \quad (4.217)$$

$$= - \frac{\partial}{\partial \mathbf{x}_i} (\mathbf{y}_i - \mathbf{x}_i)^T \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.218)$$

$$= \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.219)$$

Finally, the partial of N_2 with respect to $\mathbf{u}_i$ is zero because N_2 is not a function of $\mathbf{u}_i$, and we have

$$\frac{\partial N_2}{\partial \mathbf{u}_i} = 0 \quad (4.220)$$

as the last required expression for the plane-point problems.

4.5.8 Plane-Line Problems

The object feature distance function for the plane-line problems will be defined in the usual way: using Equations 4.1 and 3.18 (pages 41 and 33) along with the appropriate constraint set given in Figure 3.3 (page 33) we have

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i, \mathbf{v}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \\ &\text{subject to } (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i = 0 \\ &\quad \mathbf{v}_i^T \mathbf{u}_i = 0 \\ &\quad \mathbf{v}_i^T \mathbf{v}_i = 1 \end{aligned} \quad (4.221)$$

for the definition of our function.

As usual we will apply the method of Lagrange multipliers to this constrained minimization problem. For the plane-point problems, Equation 4.51 on page 53 gives us

$$\begin{pmatrix} \frac{\partial g_i}{\partial \mathbf{y}_i} \\ \frac{\partial g_i}{\partial \mathbf{v}_i} \end{pmatrix} - \begin{pmatrix} \frac{\partial \mathbf{h}_i^T}{\partial \mathbf{y}_i} \\ \frac{\partial \mathbf{h}_i^T}{\partial \mathbf{v}_i} \end{pmatrix} \boldsymbol{\lambda} = \mathbf{0} \quad (4.222)$$

where g_i is our objective function defined by

$$g_i(\mathbf{y}_i, \mathbf{v}_i) = (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.223)$$

the vector-valued function, $\mathbf{h}_i$, represents the three constraint functions,

$$h_{i1}(\mathbf{u}_i, \mathbf{v}_i) = (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i \quad (4.224)$$

$$h_{i2}(\mathbf{u}_i, \mathbf{v}_i) = \mathbf{v}_i^T \mathbf{u}_i \quad (4.225)$$

and

$$h_{i3}(\mathbf{u}_i, \mathbf{v}_i) = \mathbf{v}_i^T \mathbf{v}_i - 1 \quad (4.226)$$

and λ is a vector of three dimensions. Substituting these expressions into Equation 4.222 produces a set of six equations, the first three of which are represented by

$$\begin{aligned} \frac{\partial}{\partial \mathbf{y}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \right) \\ - \frac{\partial}{\partial \mathbf{y}_i} \left((\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i \right) \lambda_1 - \frac{\partial}{\partial \mathbf{y}_i} (\mathbf{v}_i^T \mathbf{u}_i) \lambda_2 - \frac{\partial}{\partial \mathbf{y}_i} (\mathbf{v}_i^T \mathbf{v}_i - 1) \lambda_3 = \mathbf{0} \end{aligned} \quad (4.227)$$

which, if we take the partials, gives us

$$2\mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) - \mathbf{v}_i \lambda_1 = \mathbf{0} \quad (4.228)$$

The second set of three equations is written as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \right) \\ - \frac{\partial}{\partial \mathbf{v}_i} \left((\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{v}_i \right) \lambda_1 - \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{u}_i) \lambda_2 - \frac{\partial}{\partial \mathbf{v}_i} (\mathbf{v}_i^T \mathbf{v}_i - 1) \lambda_3 = \mathbf{0} \end{aligned} \quad (4.229)$$

which, if we take the partials, becomes

$$2\mathbf{M}_i \mathbf{v}_i - (\mathbf{y}_i - \mathbf{x}_i) \lambda_1 - \mathbf{u}_i \lambda_2 - 2\mathbf{v}_i \lambda_3 = \mathbf{0} \quad (4.230)$$

These six equations plus the three constraint equations provide us with a system of nine equations and nine unknowns. Our next task will be to remove the Lagrange multipliers from the system of equations by solving for them in terms of the other variables.

If we premultiply Equation 4.228 by $\mathbf{v}_i^T$ we get

$$2\mathbf{v}_i^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) - \mathbf{v}_i^T \mathbf{v}_i \lambda_1 = 0 \quad (4.231)$$

which can be written as

$$\lambda_1 = 2\mathbf{v}_i^T \mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) \quad (4.232)$$

if we use the third constraint (Equation 4.226). Similarly, by premultiplying Equation 4.230 by $\mathbf{v}_i^T$ we get

$$2\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i - \mathbf{v}_i^T (\mathbf{y}_i - \mathbf{x}_i) \lambda_1 - \mathbf{v}_i^T \mathbf{u}_i \lambda_2 - 2\mathbf{v}_i^T \mathbf{v}_i \lambda_3 = 0 \quad (4.233)$$

which becomes

$$\lambda_3 = \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \quad (4.234)$$

if we apply all three constraints. Finally, we can premultiply Equation 4.230 by $\mathbf{u}_i^T$ to get

$$2\mathbf{u}_i^T \mathbf{M}_i \mathbf{v}_i - \mathbf{u}_i^T (\mathbf{y}_i - \mathbf{x}_i) \lambda_1 - \mathbf{u}_i^T \mathbf{u}_i \lambda_2 - 2\mathbf{u}_i^T \mathbf{v}_i \lambda_3 = 0 \quad (4.235)$$

If we use the second constraint (Equation 4.225) and substitute the expression for λ_1 , this becomes

$$\mathbf{u}_i^T \mathbf{u}_i \lambda_2 = 2\mathbf{u}_i^T \mathbf{M}_i \mathbf{v}_i - \mathbf{u}_i^T (\mathbf{y}_i - \mathbf{x}_i) \lambda_1 \quad (4.236)$$

$$= 2\mathbf{u}_i^T \mathbf{M}_i \mathbf{v}_i - 2\mathbf{u}_i^T (\mathbf{y}_i - \mathbf{x}_i) \mathbf{v}_i^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \quad (4.237)$$

from which we have

$$\lambda_2 \frac{2}{\mathbf{u}_i^T \mathbf{u}_i} \left(\mathbf{u}_i^T \mathbf{M}_i \mathbf{v}_i - \mathbf{u}_i^T (\mathbf{y}_i - \mathbf{x}_i) \mathbf{v}_i^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \right) \quad (4.238)$$

Now after substituting these expressions for λ_1 , λ_2 , and λ_3 back into Equations 4.228 and 4.230 we are left with a system of six equations which we will refer to by

$$\mathbf{N}(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) = \begin{pmatrix} \mathbf{N}_1(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) \\ \mathbf{N}_2(\mathbf{x}_i, \mathbf{u}_i, k, \mathbf{v}_i) \end{pmatrix} = \mathbf{0} \quad (4.239)$$

Here N_1 comes from Equation 4.228 and is written as

$$N_1 = C_i^{-1}(y_i - \mu_i) - v_i v_i^T C_i^{-1}(y_i - \mu_i) \quad (4.240)$$

and N_2 comes from Equation 4.230 and is written as

$$\begin{aligned} N_2 &= M_i v_i - (y_i - x_i) v_i^T C_i^{-1}(y_i - \mu_i) \\ &\quad - u_i \frac{1}{u_i^T u_i} \left(u_i^T M_i v_i - u_i^T (y_i - x_i) v_i^T C_i^{-1}(y_i - \mu_i) \right) \\ &\quad - v_i v_i^T M_i v_i \end{aligned} \quad (4.241)$$

$$\begin{aligned} &= M_i v_i - (y_i - x_i) v_i^T C_i^{-1}(y_i - \mu_i) \\ &\quad - \frac{1}{u_i^T u_i} \left(u_i u_i^T M_i v_i - u_i u_i^T (y_i - x_i) v_i^T C_i^{-1}(y_i - \mu_i) \right) \\ &\quad - v_i v_i^T M_i v_i \end{aligned} \quad (4.242)$$

As Section 4.5.4 indicated, a closed form solution of this set of equations could not be found. Consequently, we will use the method presented in that section to provide values of the object feature distance function and its partials. That is, the set of equations is solved numerically for given values of x_i and u_i to determine the values of y_i and v_i at the minimum, and these values are substituted into the objective function to provide the value of $f_i(x_i, u_i)$.

What remains to be found in this section are the partials of N^T with respect to Y_i (y_i and v_i) and X_i (x_i and u_i). These can then be used with Equations 4.80 (page 59), 4.81 (page 59), and 4.88 (page 61) to determine the partials of $f_i(x_i, u_i)$ with respect to x_i and u_i .

We will partition the partials into

$$\frac{\partial N^T}{\partial Y_i} = \begin{pmatrix} \frac{\partial N_1^T}{\partial y_i} & \frac{\partial N_2^T}{\partial y_i} \\ \frac{\partial N_1^T}{\partial v_i} & \frac{\partial N_2^T}{\partial v_i} \end{pmatrix} \quad (4.243)$$

and

$$\frac{\partial \mathbf{N}^T}{\partial \mathbf{X}_i} = \begin{pmatrix} \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{x}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{x}_i} \\ \frac{\partial \mathbf{N}_1^T}{\partial \mathbf{u}_i} & \frac{\partial \mathbf{N}_2^T}{\partial \mathbf{u}_i} \end{pmatrix} \quad (4.244)$$

and will then determine each partition individually.

We can save ourselves some work if we note the expression for $\mathbf{N}_1$ for this problem is identical to the expression for $\mathbf{N}_1$ for the plane-point problems (see Equation 4.192 on page 77). We can therefore use the results from that section and write

$$\frac{\partial \mathbf{N}_1}{\partial \mathbf{y}_i} = \mathbf{C}_i^{-1}(\mathbf{I} - \mathbf{v}_i \mathbf{v}_i^T) \quad (4.245)$$

$$\frac{\partial \mathbf{N}_1}{\partial \mathbf{v}_i} = -\mathbf{C}_i^{-1}(\mathbf{y}_i - \boldsymbol{\mu}_i) \mathbf{v}_i^T - \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.246)$$

and

$$\frac{\partial \mathbf{N}_1}{\partial \mathbf{x}_i} = \frac{\partial \mathbf{N}_1}{\partial \mathbf{u}_i} = \mathbf{0} \quad (4.247)$$

In addition, the expression for $\mathbf{N}_2$ given in Equation 4.242 is similar to the expression for $\mathbf{N}_2$ for the plane-point problems (see Equation 4.193 on page 77). The current problem contains the additional term

$$\mathbf{N}'_2 = -\frac{1}{\mathbf{u}_i^T \mathbf{u}_i} \left(\mathbf{u}_i \mathbf{u}_i^T \mathbf{M}_i \mathbf{v}_i - \mathbf{u}_i \mathbf{u}_i^T (\mathbf{y}_i - \mathbf{x}_i) \mathbf{v}_i^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \right) \quad (4.248)$$

We can therefore determine the partials of $\mathbf{N}'_2$ and add them to the partials of $\mathbf{N}_2$ given for the plane-point problems to get the desired partials of $\mathbf{N}_2$ for the plane-line problems.

First taking the partial of $\mathbf{N}'_2{}^T$ with respect to $\mathbf{y}_i$, we get

$$\frac{\partial \mathbf{N}'_2}{\partial \mathbf{y}_i} = \frac{\partial}{\partial \mathbf{y}_i} \left(-\mathbf{v}_i^T \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} + (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.249)$$

$$= \frac{\partial}{\partial \mathbf{y}_i} \left(\left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{y}_i - \mathbf{x}_i)^T \right) \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \quad (4.250)$$

$$\begin{aligned}
&= \frac{\partial}{\partial \mathbf{y}_i} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{y}_i - \mathbf{x}_i)^T \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \\
&\quad + \frac{\partial}{\partial \mathbf{y}_i} (\mathbf{y}_i - \mathbf{x}_i)^T \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \quad (4.251)
\end{aligned}$$

$$= \left((\mathbf{C}_i^{-1} \mathbf{v}_i)(\mathbf{y}_i - \mathbf{x}_i)^T + \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \right) \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \quad (4.252)$$

Now adding this to the partial of $\mathbf{N}_2$ with respect to $\mathbf{y}_i$ for the plane-point problems (Equation 4.211 on page 79) we get

$$\begin{aligned}
\frac{\partial \mathbf{N}_2}{\partial \mathbf{y}_i} &= -\mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T - \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \\
&\quad + \left(\mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T + \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \right) \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \quad (4.253)
\end{aligned}$$

$$= - \left(\mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T + \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \right) \left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.254)$$

as our expression for $\frac{\partial \mathbf{N}_2^T}{\partial \mathbf{y}_i}$ for the plane-line problems.

Now for the partial of $\mathbf{N}_2'^T$ with respect to $\mathbf{v}_i$ we have

$$\frac{\partial \mathbf{N}_2'}{\partial \mathbf{v}_i} = \frac{\partial}{\partial \mathbf{v}_i} \left(-\mathbf{v}_i^T \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} + \mathbf{v}_i^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) (\mathbf{y}_i - \mathbf{x}_i)^T \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.255)$$

$$= \frac{\partial}{\partial \mathbf{v}_i} \mathbf{v}_i^T \left(-\mathbf{M}_i + \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) (\mathbf{y}_i - \mathbf{x}_i)^T \right) \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \quad (4.256)$$

$$= \left(-\mathbf{M}_i + \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) (\mathbf{y}_i - \mathbf{x}_i)^T \right) \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \quad (4.257)$$

Adding this to $\frac{\partial \mathbf{N}_2}{\partial \mathbf{v}_i}$ for the plane-point problems (Equation 4.215, page 79) we get

$$\begin{aligned}
\frac{\partial \mathbf{N}_2}{\partial \mathbf{v}_i} &= \mathbf{M}_i - \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) (\mathbf{y}_i - \mathbf{x}_i)^T - 2\mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T - \mathbf{I} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \\
&\quad + \left(-\mathbf{M}_i + \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) (\mathbf{y}_i - \mathbf{x}_i)^T \right) \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \quad (4.258)
\end{aligned}$$

$$\begin{aligned}
&= \left(\mathbf{M}_i - \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) (\mathbf{y}_i - \mathbf{x}_i)^T \right) \left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \\
&\quad - 2\mathbf{M}_i \mathbf{v}_i \mathbf{v}_i^T - \mathbf{I} (\mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i) \quad (4.259)
\end{aligned}$$

as our final expression for $\frac{\partial \mathbf{N}_2^T}{\partial \mathbf{v}_i}$.

For the partial of $N_2'^T$ with respect to $\mathbf{x}_i$ we have

$$\frac{\partial N_2'}{\partial \mathbf{x}_i} = \frac{\partial}{\partial \mathbf{x}_i} \left(-\mathbf{v}_i^T \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} + (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.260)$$

$$= \frac{\partial}{\partial \mathbf{x}_i} \left(\left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) (\mathbf{y}_i - \mathbf{x}_i)^T \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.261)$$

$$= \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \frac{\partial}{\partial \mathbf{x}_i} (\mathbf{y}_i - \mathbf{x}_i)^T \left(\frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.262)$$

$$= - \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \left(\frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.263)$$

We add this to the partial of N_2 with respect to $\mathbf{x}_i$ for the plane-point problems (Equation 4.219 on page 79) and get

$$\frac{\partial N_2}{\partial \mathbf{x}_i} = \mathbf{I} \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.264)$$

$$- \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \left(\frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.265)$$

$$= \left(\mathbf{I} - \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \left((\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i \right) \quad (4.266)$$

as our expression for $\frac{\partial N_2^T}{\partial \mathbf{x}_i}$.

Finally, we take the partial of $N_2'^T$ with respect to $\mathbf{u}_i$ using

$$\frac{\partial N_2'}{\partial \mathbf{u}_i} = \frac{\partial}{\partial \mathbf{u}_i} \left(-\mathbf{v}_i^T \mathbf{M}_i \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} + (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.267)$$

$$= \frac{\partial}{\partial \mathbf{u}_i} \left(\left(-\mathbf{v}_i^T \mathbf{M}_i + (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T \right) \frac{\mathbf{u}_i \mathbf{u}_i^T}{\mathbf{u}_i^T \mathbf{u}_i} \right) \quad (4.268)$$

The result of this partial is easily determined if we note the similarity of this equation to Equation 4.172 on page 74. The only difference between the two is that we have replaced the vector term $\mathbf{v}_i^T \mathbf{M}_i$ of that equation with

$$\left(\mathbf{v}_i^T \mathbf{M}_i - (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T \right) \quad (4.269)$$

Using the result of that partial operation given in Equation 4.177 (page 74), we can therefore make this same substitution to determine the partial of Equation 4.268. This gives us

$$\frac{\partial N'_2}{\partial \mathbf{u}_i} = -\frac{\mathbf{w}\mathbf{u}_i^T + \mathbf{I}(\mathbf{w}^T \mathbf{u}_i)}{\mathbf{u}_i^T \mathbf{u}_i} + \frac{2\mathbf{u}_i \mathbf{w}^T \mathbf{u}_i \mathbf{u}_i^T}{(\mathbf{u}_i^T \mathbf{u}_i)^2} \quad (4.270)$$

where $\mathbf{w}$ is given by

$$\mathbf{w} = \left(\mathbf{v}_i^T \mathbf{M}_i - (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} \mathbf{v}_i (\mathbf{y}_i - \mathbf{x}_i)^T \right) \quad (4.271)$$

Because the partial of N_2 with respect to $\mathbf{u}_i$ is 0 for the plane-point problems (Equation 4.220 on page 80), Equation 4.270 will also serve as our expression for $\frac{\partial N'_2}{\partial \mathbf{u}_i}$. This completes the derivation of the required partials for the plane-line problems.

4.5.9 Plane-Plane Problems

Finally, we turn to our last type of sensing, the plane-plane problems. To define the object feature distance function for these problems, we again use Equations 4.1 and 3.18 (pages 41 and 33) along with the appropriate constraint set given in Figure 3.3 (page 33). This gives us

$$\begin{aligned} f_i(\mathbf{x}_i, \mathbf{u}_i) &= \min_{\mathbf{y}_i, \mathbf{v}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) + \mathbf{v}_i^T \mathbf{M}_i \mathbf{v}_i \\ &\text{subject to } (\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0 \\ &\quad \mathbf{v}_i^T = \mathbf{u}_i \\ &\quad \mathbf{v}_i^T \mathbf{v}_i = 1 \end{aligned} \quad (4.272)$$

for the definition of our function.

As was the case for the line-line problems, we notice that the second constraint of this minimization problem completely binds the variable $\mathbf{v}_i$, and the third constraint is therefore unnecessary ($\mathbf{u}_i$ will always be a unit vector). We can therefore substitute $\mathbf{u}_i$

for $\mathbf{v}_i$ in the objective function, remove the last two constraints, and remove the minimization over $\mathbf{v}_i$. In addition, the second term of the objective function can be moved outside of the minimization operation because it does not contain $\mathbf{y}_i$, the parameter vector we are minimizing over. As a result, we can rewrite our distance function as

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i + \min_{\mathbf{y}_i} (\mathbf{y}_i - \boldsymbol{\mu}_i)^T \mathbf{C}_i^{-1} (\mathbf{y}_i - \boldsymbol{\mu}_i) \quad (4.273)$$

subject to $(\mathbf{y}_i - \mathbf{x}_i)^T \mathbf{u}_i = 0$

which is the sum of two functions.

The minimization term of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ is identical to the minimization we performed in Equation 4.47 on page 52 for the point-plane problems. $f_i(\mathbf{x}_i, \mathbf{u}_i)$ and its derivatives for the plane-plane problem can therefore be found by taking the results of the point-plane problem and adding the contribution from the term $\mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i$. In this way we use Equation 4.69 (page 56) and define

$$f_i(\mathbf{x}_i, \mathbf{u}_i) = \mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i + \frac{(\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i))^2}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.274)$$

for the plane-plane problems. For the partial of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ with respect to $\mathbf{x}_i$, we note that $\mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i$ is not a function of $\mathbf{x}_i$, so we can use Equation 4.72 (page 56) without modification and write

$$\frac{\partial f_i}{\partial \mathbf{x}_i} = 2 \frac{\mathbf{u}_i \mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \quad (4.275)$$

for our problem. Finally, we note that the partial of $\mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i$ with respect to $\mathbf{u}_i$ is given by

$$\frac{\partial}{\partial \mathbf{u}_i} \mathbf{u}_i^T \mathbf{M}_i \mathbf{u}_i = 2 \mathbf{M}_i \mathbf{u}_i \quad (4.276)$$

Adding this to Equation 4.78 (page 57) we get

$$\frac{\partial f_i}{\partial \mathbf{u}_i} = 2 \mathbf{M}_i \mathbf{u}_i + 2(\mathbf{x}_i - \boldsymbol{\mu}_i) \left(\frac{\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right) - 2 \mathbf{C}_i \mathbf{u}_i \left(\frac{\mathbf{u}_i^T (\mathbf{x}_i - \boldsymbol{\mu}_i)}{\mathbf{u}_i^T \mathbf{C}_i \mathbf{u}_i} \right)^2 \quad (4.277)$$

for the partial of this object feature distance function with respect to u_i . This completes the derivation of $f_i(\mathbf{x}_i, u_i)$ and its partials for the plane-plane problems.

4.6 Chapter Summary

In this chapter we have presented the mathematical details for evaluating the global pose distance function and its derivatives for given values of pose. This information, along with the constraint function and its derivatives (Equations 4.4 and 4.5 on page 42), can be used by commercially available software packages to minimize the pose function. Our estimate of pose is then given by the location of this minimum.

The technique used to evaluate the pose function and its derivatives involves first determining for each object feature, the parameters $\mathbf{x}_i$ and u_i (using Equations 2.4, 2.5, and 2.10 on pages 17 and 20) and their partials with respect to $\mathbf{t}$ and $\mathbf{q}$ (using Equations 4.13, 4.14, 4.17, and 4.18 in Section 4.4). The function $f_i(\mathbf{x}_i, u_i)$ and its partials are then evaluated using the methods of Section 4.5.¹⁵ The value of the global pose function can now be found using Equations 4.2 and 4.3 on page 42; its partials can be found using Equations 4.6 (page 43), 4.8 (page 44), 4.9 (page 45), and 4.19 (page 47).

¹⁵ The evaluation of $f_i(\mathbf{x}_i, u_i)$ in turn involves its own minimization problem which should not be confused with the larger minimization problem referred to above.

CHAPTER 5

IMPLEMENTATION AND EXPERIMENTAL RESULTS

Previous chapters in this dissertation have described a fusion algorithm for combining multiple sources of sensory information for the purpose of determining an object's pose. This chapter will describe an implementation of this algorithm and will present the results from a series of experiments which verify its operation.

5.1 Algorithm Implementation

The sensor fusion algorithm was implemented in FORTRAN and run on a VAX 11/785 computer. In this implementation, minimization of the global pose function was performed using the NPSOL software package from Stanford University [27]. As was described in Chapter 4, the use of this package required a routine for evaluating the objective function (i.e., the global pose function) and its derivatives and one for evaluating the constraint function and its derivatives for given values of pose. The routines which were written closely followed the procedure of Chapter 4 with one notable exception which is explained in the following paragraphs. The software is given in [21].

Although NPSOL searches for a minimum for which $\overset{\circ}{\mathbf{q}}$ is a unit vector, this constraint is not generally satisfied during the search process. This means that NPSOL will request evaluation of the global pose function and its derivatives for values of $\overset{\circ}{\mathbf{q}}$ which do not have unit length (i.e., $\overset{\circ}{\mathbf{q}}$ does not represent a true orientation—the matrix of Equation 2.10 on page 20 is not a valid rotation matrix). Mathematically this

presents no problem because the equations, as we have stated them, will still exhibit the correct solution point; however, the algorithm seemed to have greater stability when the vector $\mathring{\mathbf{q}}$ was normalized before it was substituted into the equations (i.e., $\mathring{\mathbf{q}}$ was replaced by $\mathring{\mathbf{q}}/\sqrt{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}}$). This could have been accomplished by redefining the matrix $\mathbf{Q}$ of Equation 2.10 on page 20, replacing each q_j with $q_j/\sqrt{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}}$. We achieved this same effect by redefining Equations 2.4 and 2.5 on page 17 as

$$\mathbf{x}_i = \mathbf{Q} \mathbf{r}_i \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) + \mathbf{t} \quad (5.1)$$

and

$$\mathbf{u}_i = \mathbf{Q} \mathbf{w}_i \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) \quad (5.2)$$

respectively, where $\mathbf{Q}$ maintains its original definition given in Equation 2.10.

It is important to note that in addition to affecting the computation of $\mathbf{x}_i$ and $\mathbf{u}_i$, this redefinition also changes the partials of $\mathbf{x}_i$ and $\mathbf{u}_i$ with respect to $\mathring{\mathbf{q}}$ (the partials with respect to $\mathbf{t}$ are unchanged). We compute the partial of $\mathbf{x}_i^T$ with respect to $\mathring{\mathbf{q}}$ by

$$\frac{\partial \mathbf{x}_i^T}{\partial \mathring{\mathbf{q}}} = \frac{\partial}{\partial \mathring{\mathbf{q}}} \left(\left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) \mathbf{r}_i^T \mathbf{Q}^T + \mathbf{t}^T \right) \quad (5.3)$$

$$= \frac{\partial}{\partial \mathring{\mathbf{q}}} (\mathbf{r}_i^T \mathbf{Q}^T) \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) + \frac{\partial}{\partial \mathring{\mathbf{q}}} \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) (\mathbf{r}_i^T \mathbf{Q}^T) \quad (5.4)$$

$$= \frac{\partial}{\partial \mathring{\mathbf{q}}} (\mathbf{r}_i^T \mathbf{Q}^T) \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) + \frac{\partial}{\partial \mathring{\mathbf{q}}} (\mathring{\mathbf{q}}^T \mathring{\mathbf{q}})^{-1} (\mathbf{r}_i^T \mathbf{Q}^T) \quad (5.5)$$

$$= \frac{\partial}{\partial \mathring{\mathbf{q}}} (\mathbf{r}_i^T \mathbf{Q}^T) \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) - \frac{\partial}{\partial \mathring{\mathbf{q}}} (\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}) (\mathring{\mathbf{q}}^T \mathring{\mathbf{q}})^{-2} (\mathbf{r}_i^T \mathbf{Q}^T) \quad (5.6)$$

$$= \frac{\partial}{\partial \mathring{\mathbf{q}}} (\mathbf{r}_i^T \mathbf{Q}^T) \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) - 2 \mathring{\mathbf{q}} (\mathring{\mathbf{q}}^T \mathring{\mathbf{q}})^{-2} (\mathbf{r}_i^T \mathbf{Q}^T) \quad (5.7)$$

$$= \frac{\partial}{\partial \mathring{\mathbf{q}}} (\mathbf{r}_i^T \mathbf{Q}^T) \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \right) - \frac{2}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \mathring{\mathbf{q}} \left(\frac{1}{\mathring{\mathbf{q}}^T \mathring{\mathbf{q}}} \mathbf{r}_i^T \mathbf{Q}^T \right) \quad (5.8)$$

Note that the partial in the first term of this expression can be computed using the old definition for $\frac{\partial \mathbf{x}_i^T}{\partial \dot{\mathbf{q}}}$ given in Equation 4.17 on page 46. Also note that the second term is equivalent to

$$\frac{2}{\dot{\mathbf{q}}^T \dot{\mathbf{q}}} \dot{\mathbf{q}} (\mathbf{x}_i - \mathbf{t})^T \quad (5.9)$$

where $\mathbf{x}_i$ was computed in Equation 5.1.

Our new definition of the partial of $\mathbf{u}_i$ with respect to $\dot{\mathbf{q}}$ is easily written if we note the similarity between Equations 5.1 and 5.2. That is, if we replace $\mathbf{r}_i$ with $\mathbf{w}_i$ and drop the variable $\mathbf{t}$ then we can immediately write our redefinition of Equation 4.18 on page 46 as

$$\frac{\partial \mathbf{u}_i^T}{\partial \dot{\mathbf{q}}} = \frac{\partial}{\partial \dot{\mathbf{q}}} (\mathbf{w}_i^T \mathbf{Q}^T) \left(\frac{1}{\dot{\mathbf{q}}^T \dot{\mathbf{q}}} \right) - \frac{2}{\dot{\mathbf{q}}^T \dot{\mathbf{q}}} (\dot{\mathbf{q}} \mathbf{u}_i^T) \quad (5.10)$$

where the partial in the first term is computed using the old definition for $\frac{\partial \mathbf{u}_i}{\partial \dot{\mathbf{q}}}$ given in Equation 4.18, and where $\mathbf{u}_i$ is determined using Equation 5.2.

5.2 Testbed for Experiments

Data collection for the experiments described later in this chapter was performed using the robotics workstation in the Electrical and Computer Engineering Department at the University of Tennessee, Knoxville. This section will describe the essential components of this workstation and their use in generating sensory information for testing the fusion algorithm.

The principal components of the robotics workstation are a Cincinnati Milacron T3-726 industrial robot (Figure 5.1), a VAX 11/785 host computer, and a sensory package mounted on the end effector of the robot (Figure 5.2). The robot is a fixed-base manipulator with six-degrees of freedom, capable of moving the sensory package

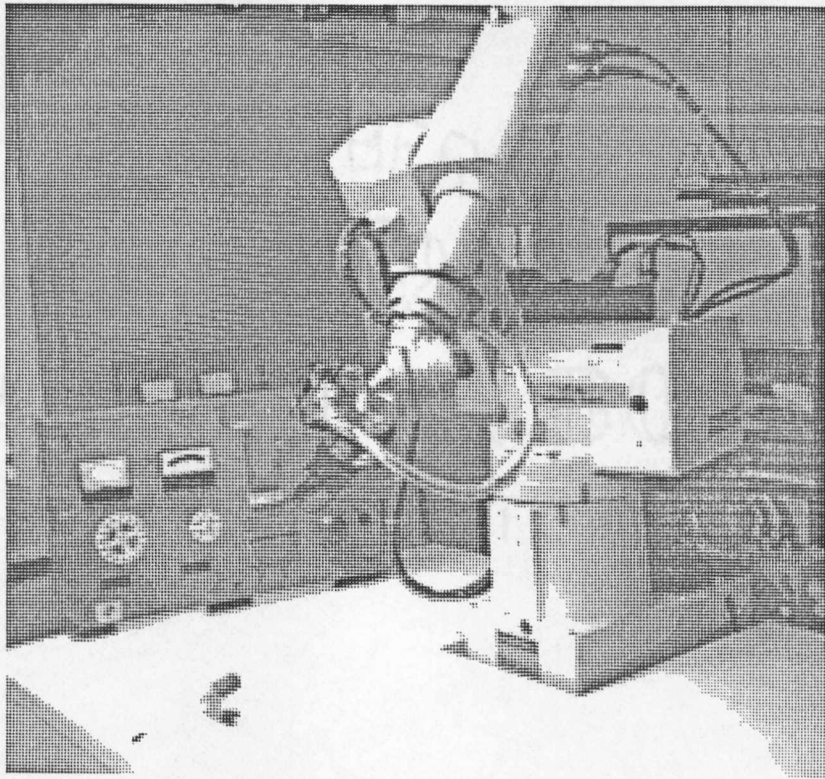


Figure 5.1: The robotics workstation at the University of Tennessee.

on the end effector to any position in its workspace with (practically) any orientation. The computer performs all high-level processing. It has direct control of the robot and direct access to all sensory information. Four sensors in the sensory package were used in our experiments: a vision system, a force/torque sensor (used for touch), a proximity sensor, and an ultrasonic range sensor. Before describing the use of these sensors, we will first provide some general information which is applicable to each.

Each of the sensory measurements provides a best guess of the point and unit vector parameters of a sensory feature. As was described in Section 3.4, these values must



Figure 5.2: The sensory package mounted on the end effector.

be converted to the form of a distance function (Equation 3.18 on page 33) reflecting sensor error in order for them to be used by the fusion algorithm. For our experiments, the conversion to a distance function was based on rough estimates rather than on precise calibration. The method used was to treat the position and orientation terms of Equation 3.18 independently and assign the coefficients of each term so that the contour for which the function has a value of one lies at the edge of the expected range.

For example, if a point, y_i , is expected to lie within a half inch of the reported value in the direction r_1 , then the eigenvalue in this direction is chosen to be four (see Section 3.4). If the point is expected to lie within a quarter inch in orthogonal directions r_2 and r_3 , then the eigenvalues in these directions are chosen to be 16. Now according to Equation 3.19 on page 34, the matrix C_i^{-1} is assigned a value of

$$C_i^{-1} = R \Lambda R^T = \begin{pmatrix} r_1 & r_2 & r_3 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 \\ 0 & 16 & 0 \\ 0 & 0 & 16 \end{pmatrix} \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix} \quad (5.11)$$

The parameter μ_i is of course set to the value of y_i reported by the sensor.

The direction term is generated in a similar manner. For example, if a unit vector is reported to lie in a direction of s_1 , and it is expected to vary by up to six degrees in an orthogonal direction s_2 and up to three degrees in a mutually orthogonal direction of s_3 , then we assign λ_1 a value of zero, λ_2 a value of $1/\sin^2(6^\circ) \approx 91.5$, and λ_3 a value of $1/\sin^2(3^\circ) \approx 365.1$. The matrix M_i is then given by

$$M_i = S \Lambda S^T = \begin{pmatrix} s_1 & s_2 & s_3 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 91.5 & 0 \\ 0 & 0 & 365.1 \end{pmatrix} \begin{pmatrix} s_1^T \\ s_2^T \\ s_3^T \end{pmatrix} \quad (5.12)$$

Each of the experiments we performed used the panel shown in Figure 5.3 as the object whose pose was to be determined. Table 5.1 shows the information stored in the object model for this object.¹ The point, line, and plane object features given in this model can be sensed using various sensors in the sensory package. Each of the following subsections describes a sensor in this package and the data it provided for our experiments.

5.2.1 Vision Sensing

The vision system consists of a black and white Fairchild CCD-3000 camera interfaced to a Perceptics 9200e image processor. When prompted by the host computer, this processor will digitize, store, and process individual frames from the camera video output.

To simplify the vision processing requirements for our experiments, we use the panel lights as the object features we sense with the camera. These lights can be turned on and off by software control. Therefore by taking two images, one with the panel lights on and one with the panel lights off, and then subtracting one from the other, the location of these lights in the image is easily determined. When taken individually, the panel lights represent point object features, and when taken in pairs, these lights can be used to simulate line object features.

Several of the classes of sensing defined in Chapter 3 can result from the use of visual data; we will briefly discuss four of these which were used in the experiments.

When a single camera image is used in sensing a point object feature, then the data can be presented to the fusion algorithm as a line-point problem. In this case we let y_i , a point on the line sensory feature, physically represent the actual location of the

¹ The feature description given in Table 5.1 is not actually stored as part of the object model, but is given here for convenience.

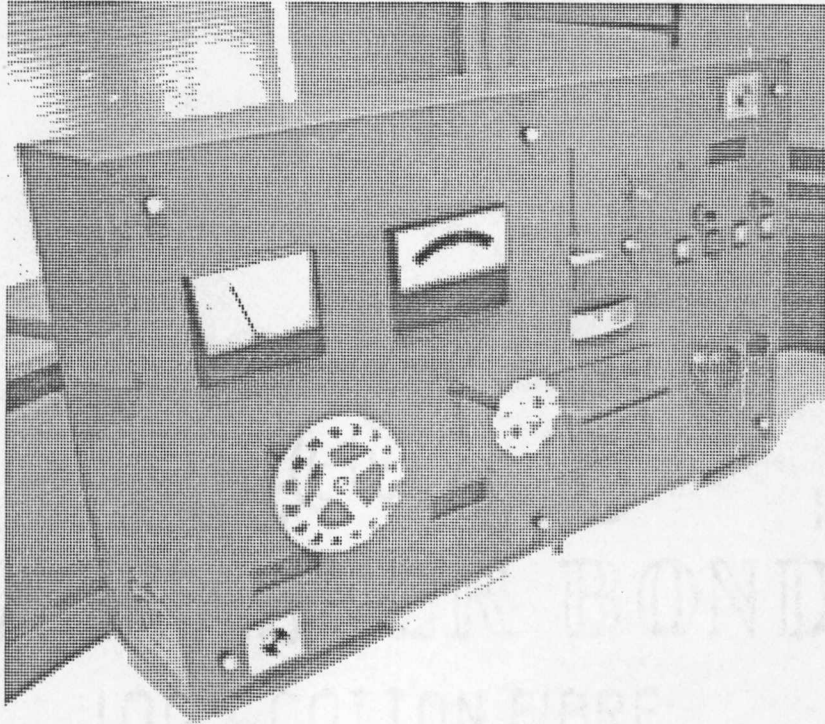


Figure 5.3: The object whose pose we are to determine.

Table 5.1: The object model for the panel.

Feature Description ^a	Feature Type	Position			Orientation		
		r_1	r_2	r_3	w_1	w_2	w_3
TL light	Point	1.5	18.5	0.55	0.0	0.0	1.0
BL light	Point	1.5	1.5	0.55	0.0	0.0	1.0
TC light	Point	16.5	18.5	0.55	0.0	0.0	1.0
BC light	Point	16.5	1.5	0.55	0.0	0.0	1.0
TR light	Point	31.5	18.5	0.55	0.0	0.0	1.0
BR light	Point	31.5	1.5	0.55	0.0	0.0	1.0
FBL vertex	Point	0.0	0.0	0.0	1.0	0.0	0.0
FBR vertex	Point	33.0	0.0	0.0	1.0	0.0	0.0
FTR vertex	Point	33.0	20.0	0.0	1.0	0.0	0.0
FTL vertex	Point	0.0	20.0	0.0	1.0	0.0	0.0
KBL vertex	Point	0.0	0.0	-6.2	1.0	0.0	0.0
KBR vertex	Point	33.0	0.0	-6.2	1.0	0.0	0.0
KTR vertex	Point	33.0	20.0	-6.2	1.0	0.0	0.0
KTL vertex	Point	0.0	20.0	-6.2	1.0	0.0	0.0
FB edge	Line	0.0	0.0	0.0	1.0	0.0	0.0
FT edge	Line	0.0	20.0	0.0	1.0	0.0	0.0
KT edge	Line	0.0	20.0	-6.2	1.0	0.0	0.0
KB edge	Line	0.0	0.0	-6.2	1.0	0.0	0.0
FL edge	Line	0.0	0.0	0.0	0.0	1.0	0.0
FR edge	Line	33.0	0.0	0.0	0.0	1.0	0.0
KR edge	Line	33.0	0.0	-6.2	0.0	1.0	0.0
KL edge	Line	0.0	0.0	-6.2	0.0	1.0	0.0
BL edge	Line	0.0	0.0	0.0	0.0	0.0	1.0
BR edge	Line	33.0	0.0	0.0	0.0	0.0	1.0
TR edge	Line	33.0	20.0	0.0	0.0	0.0	1.0
TL edge	Line	0.0	20.0	0.0	0.0	0.0	1.0
L surface	Plane	0.0	0.0	0.0	-1.0	0.0	0.0
R surface	Plane	33.0	0.0	0.0	1.0	0.0	0.0
B surface	Plane	0.0	0.0	0.0	0.0	-1.0	0.0
T surface	Plane	0.0	20.0	0.0	0.0	1.0	0.0
F surface	Plane	0.0	0.0	0.0	0.0	0.0	1.0
K surface	Plane	0.0	0.0	-6.2	0.0	0.0	-1.0

^aThe letters refer to the sides when facing the panel: L refers to the left side, R to the right, T to the top, B to the bottom, F to the front, and K to the back.

camera lens center; μ_i is then assigned a value equal to the reported position of this point. The matrix C_i^{-1} is chosen to indicate the expected accuracy of this position according to the method outlined above.

The vector portion of the sensory feature, v_i , represents the unit vector from the lens center to the object feature point. An estimate of this vector is computed from knowledge of the image coordinates of (the projection of) the object point, the camera imaging parameters, and the orientation of the camera. As was described above, we assign this as the value of the eigenvector of M_i which corresponds to the zero eigenvalue (s_1 in Equation 5.12). The eigenvalues corresponding to the eigenvectors in the orthogonal directions are chosen according to the estimated angular accuracy of this measurement using the method outlined above.

As was previously indicated, pairs of panel lights were used to simulate line object features (e.g., the edge of an object). When sensed from a single camera position, this represents an instance of plane-line sensing. We selected the position parameters of the sensory plane in the same way in which they were selected for the line in line-point sensing; i.e., y_i represents the camera lens center, μ_i is its reported position, and C_i^{-1} represents the estimated positional accuracy. The normal to the plane is defined by the the image coordinates of the two points, camera imaging parameters, and the orientation of the camera.² The computed value of this normal was assigned to the first eigenvector, s_1 . As before, the estimated orientation accuracy dictates the value of the other two eigenvalues.

² Because we are simulating line object features, the image coordinates of the two points are ignored; only the parameters of the image line they represent are of interest.

The mobility of our camera allows us to take images of an object feature from two distinct camera locations (i.e., stereo vision). If our object feature is a point, then triangulation can be used with the two images to compute the coordinates of the point, and we therefore have an instance of point-point sensing. The computed coordinates of the point were assigned to μ_i . The matrix C_i^{-1} was chosen to reflect the positional accuracy of this measurement; normally this accuracy is chosen to be less (i.e., the eigenvalue is smaller) in the depth direction. The matrix M_i was of course set to the zero matrix for this type of problem.

When a line object feature (simulated using two panel lights) is sensed with stereo vision, then the information is used as a line-line problem. Each image of the “line” defines a plane in world coordinates, and the intersection of the two planes defines our estimate of the line’s position. The point μ_i is chosen to be a point on this line midway between the two panel lights. As before, C_i^{-1} indicates the estimated positional accuracy of this point, and M_i indicates both the expected direction of the line and the estimated accuracy of this direction.

5.2.2 Touch Sensing

Touch sensing was implemented in our experiments using a force/torque wrist sensor manufactured by Lord Corporation. The technique involved picking up a probe with the gripper and using it to “poke” the object. On contact, the wrist sensor registered a force and the robot halted and reported the position of the probe tip.

Touch sensing involved several classes of sensing. Point-plane sensing occurred when a planar surface was touched; point-line sensing occurred when an edge was touched;

and point-point sensing occurred when a point object feature was touched.³ In each of these cases the sensory feature is a point. The reported coordinates of the probe tip were assigned to the variable μ_i in the sensory distance function. The matrix C_i^{-1} was chosen to reflect the accuracy of this position. Note that this accuracy is fairly high with touch sensing—almost as good as the robot positioning accuracy (probe “wobble” and nonzero probe size make it somewhat less accurate). The matrix M_i was set to zero in each of these cases as the direction component for point sensory features was not used.

By combining multiple point measurements of an object plane, other types of sensing were also implemented. Line-plane sensing was simulated by touching the plane in two distinct locations and then computing the parameters of the line through the two points. Similarly, plane-plane sensing was simulated by touching the plane in three places and computing the parameters of the plane through the three points. In both of these cases, the vector μ_i was chosen to be at the centroid of the points. The matrices C_i^{-1} and M_i were chosen to reflect the point accuracies.

5.2.3 Proximity Sensing

The gripper of the robot contains a set of infrared proximity sensors which serve to indicate when an object intrudes into its “sensitive volume.” The location of this sensitive volume as the object is detected yields the parameters of a point sensory feature.

We use proximity sensing in the same way we use touch sensing; the main difference is that the accuracy of the reported coordinates of the point is somewhat less with the proximity sensor. Proximity sensing therefore gives us another modality for

³ Note that in the collection of data for our experiments, human guidance is used to draw the probe towards these features.

acquiring point-point, point-line, point-plane, line-plane, and plane-plane data. For our experiments this is useful demonstrating the algorithm with data of different accuracies.

5.2.4 Range Sensing

The range sensor in our sensory package is an ultrasonic range sensor made by Polaroid Corporation. This sensor operates by transmitting an ultrasonic “chirp” and then listening to see how long it takes to receive an echo, the distance to the object being linearly proportional to the time delay.

Range sensing can be viewed as being equivalent to touch and proximity sensing, as the measured distance and the direction of the ultrasonic signal can be used to compute the coordinates of a point on the object; however, because of the beam spread, the accuracy characteristics are somewhat different with this sensor. This sensor is fairly accurate in reporting point coordinates in the depth direction and very inaccurate in the lateral directions. For our relatively small object, this implies the sensor is the most useful in sensing its “protruding” features (such as vertices).

In our experiments we used the range sensor as a plane-point sensor; that is, we assumed the (protruding) point feature of the object lay somewhere on a plane whose distance from the sensor was given by the distance measurement. μ_i was assigned the value of the computed world coordinates of the point at this distance from the sensor, and the most likely direction of the normal to the plane was given by the orientation of the beam.

5.3 Experimental Results

A series of experiments was performed to demonstrate that the fusion algorithm can determine object pose when given information from various combinations of sensors. In each experiment the panel was placed in a known position and orientation and a number of sensory measurements were made. The data was then passed to the fusion algorithm, and the computed pose was compared to the actual pose. We report the results of this comparison in terms of separate position and orientation errors: the position error represents the distance (in inches) between the positions of the computed and actual panel centers; the orientation error represents the angle of rotation required to get from the computed orientation to the actual orientation.⁴

In the experiments we performed, a human operator provided the sensing strategy, selecting which sensors to use and how to position them. The operator also performed the task of matching each element of sensory data to the appropriate object feature.

It should be noted that algorithm convergence issues were not of primary importance in these experiments. Optimization techniques cannot in general determine if a local minimum found is also a global minimum, and for the problems we are dealing with, multiple local minimums of equal (or nearly equal) value can exist if insufficient data is passed to the fusion algorithm. An example of this is given when all measurements come from a single visual image. In this case two minimums exist, one placing the panel in front of the camera and one placing it behind the camera and rotated 180 degrees (according to projective geometry, placing the panel in either pose would result in the same projection onto the image plane).

⁴ The dot product of the quaternions representing the two orientations yields the cosine of half this angle.

The initial guess presented to the fusion algorithm affects which local minimum is found. In our experiments we first used an initial guess in which the object coordinate axes were aligned with the world coordinate axes. If the algorithm converged on the wrong minimum, we then tried again with an estimate closer to the actual object pose in order that the algorithm favor the correct minimum. In a practical implementation of this algorithm, higher level information (knowledge of unacceptable poses) or further sensing should be used to insure the minimum found is the desired one.

In reporting our results we use a two character descriptor to refer to each particular type of sensing. The first character is a digit which represents the class of sensing, and the second character is a letter indicating the particular sensor which was used. Table 5.2 summarizes the types of sensing we performed and the descriptor used for each.

5.3.1 Testing the Individual Types of Sensing

In our first set of experiments, each experiment used only a single type of sensing to determine object pose. The purpose of this set of experiments was threefold: to demonstrate the operation of the fusion algorithm with single classes of sensors, to verify that the data collection method and software implementation were functioning properly, and to get a rough idea of the CPU time required to solve each type of problem on a VAX 11/785.

The results of this set of experiments are given in Table 5.3. Included in this table are the number of measurements used, the position error, the orientation error, the number of major iterations required by NPSOL, the Central Processing Unit (CPU) time in seconds, and the average CPU time per measurement per iteration.

Table 5.2: Summary of the types of sensing we perform and the descriptors for each.

Descriptor	Sensing Class	Sensor
1p	Point-point sensing	Proximity
1t	Point-point sensing	Touch
1v	Point-point sensing	Stereo vision
2p	Point-line sensing	Proximity
2t	Point-line sensing	Touch
3p	Point-plane sensing	Proximity
3t	Point-plane sensing	Touch
4v	Line-point sensing	Vision
5v	Line-line sensing	Stereo vision
6p	Line-plane sensing	Proximity
6t	Line-plane sensing	Touch
7r	Plane-point sensing	Range
8v	Plane-line sensing	Vision
9p	Plane-plane sensing	Proximity
9t	Plane-plane sensing	Touch

Table 5.3: Estimating Pose using a Single Type of Sensing.

Sensing Type	Number of Vectors	Position Error (inches)	Orientation Error (degrees)	Number of Major Iterations	CPU Time (sec)	Cycle Time (ms)
1p	15	0.12	0.46	36	5.8	10.7
1t	14	0.18	0.23	82	12.8	11.1
1v	13	1.24	1.59	30	4.5	11.5
2p	10	0.08	0.66	51	8.0	15.7
2t	19	0.14	0.46	53	11.7	11.6
3p	16	0.12	0.41	55	10.2	11.6
3t	16	0.20	0.04	51	9.2	11.3
4v	16	0.89	1.59	37	39.4	66.5
5v	14	1.20	0.37	45	7.7	12.3
6p	11	0.14	0.33	63	23.2	33.5
6t	11	0.09	0.20	59	23.3	35.9
7r	9	10.41	60.09	83	58.6	78.5
8v	19	0.93	0.50	19	42.3	117.3
9p	5	0.16	0.54	26	3.1	23.7
9t	5	0.28	0.20	25	2.9	23.2

As was described above, the position error refers to the difference between the computed and the actual position of the center of the panel, and the orientation error refers to the angle of rotation required to move from the computed orientation to the actual orientation. As can be seen in Table 5.3, object pose was computed with fairly good precision for each type of sensing with the exception of the plane-point sensing (type 7r). The large error with this type of sensing is likely due to the small amount of pose information in each measurement⁵ and the small number of measurements used.⁶ As should be expected, the errors were less for the "contact" sensors (i.e., proximity and touch) than they were for the "noncontact" sensors (range and vision); for the contact sensors the errors were within the measurement accuracy of the "actual" panel pose which was used as the reference.

The CPU times given reflect the total CPU time required to solve for pose on a VAX 11/785 computer. These execution times should be viewed with caution, since much depends on the the particular data which is used, the initial pose estimate given to NPSOL, and consequently, the particular search path followed by NPSOL to find the minimum.

The cycle time provides a better comparison between the sensing types. These times represent the total CPU time divided by the product of the number of measurements and the number of major iterations required by NPSOL. The cycle times therefore reflect the relative effort required to evaluate the local pose function and its derivatives; however, note that because each major iteration of NPSOL requires a variable number of these

⁵ Each plane-point measurement can constrain only one of the six degrees of freedom of object pose.

⁶ Our fixed-based robot could only be placed in a limited number of useful positions for obtaining these measurements.

evaluations, this measure will also depend to some extent on the data, the initial guess, and the search path taken.

5.3.2 Testing the Fusion of Multisensor Data

In a second set of three experiments, the fusion algorithm was presented with miscellaneous combinations of sensing types in order to demonstrate its operation with multisensor data. In each experiment, enough measurements were selected to completely constrain pose, but the number was kept small so that each measurement would have a significant affect on the result. Note that each of the nine types of sensing was used at least once in this set of experiments.

The results of these experiments are given in Table 5.4. For each experiment, the errors in position and in orientation are given along with the evaluations of the local pose functions at both the actual pose and at the pose computed from the data (each row represents a single measurement of the given type). Note that the global pose is represented by the sum of the individual functions.

In each of these experiments, the errors in position and orientation for the computed pose were relatively small, indicating the successful operation of the algorithm. Although the actual object pose would not be available in a practical implementation of this algorithm, for our experiments, the evaluation of a local pose function at this pose provides an indication of how accurate that measurement is and consequently how much that measurement can adversely affect the solution. If this value is small, then the measurement is right on target (assuming the actual pose was determined correctly); larger values indicate worse measurements (possibly the result of miscalibration) and will tend to pull the global pose estimate away from the actual pose. As can be seen

Table 5.4: Estimating pose using multiple types of sensing.

Experiment Number	Position Error (inches)	Orientation Error (degrees)	Sensing Type	Pose Function Value	
				at Actual Pose	at Estimated Pose
1	0.11	1.33	1t	0.66	0.86
			2p	0.04	0.05
			3t	0.01	0.03
			4v	0.71	0.60
			4v	0.59	0.50
			5v	1.33	0.92
			Total	3.34	2.97
2	0.43	1.85	1v	1.38	0.54
			2t	0.28	0.43
			6p	0.002	0.03
			7r	0.55	0.29
			9t	0.62	0.13
			1p	0.31	0.11
			Total	3.14	1.53
3	0.28	0.30	1t	0.41	5×10^{-5}
			2p	0.11	2×10^{-7}
			8v	0.004	5×10^{-4}
			Total	0.52	6×10^{-4}

in Table 5.4, most of the measurements were good in this respect with the exception of two measurements: 5v in the first experiment and 1v in the second.

The value of the global pose function at the estimated pose (i.e., the global pose minimum) provides a measure of agreement among the sensory data: the lower the value, the better the agreement. Furthermore, the contribution from each particular local pose function indicates how well that measurement agreed with the others. Note that, for the present experiments, the agreement was fairly good. In particular, it was very good for the third experiment. This was because the small amount of data used was barely enough to constrain the six degrees of freedom of object pose; the pose that was found was satisfactory for all of the data involved.

5.3.3 The Introduction of Bad data

This last set of experiments was then rerun with the same data, but an improperly-matched sensory feature was also included with each experiment in order to show the effect of bad data on pose estimation. In each rerun of the first two experiments, the added bad data was given a "strength" which was large enough to be significant, but not so large as to completely outweigh the good data (i.e., line-point data whose local pose function changes relatively little for large changes in object pose). In the third experiment, however, the added bad data was chosen so as to have considerable effect on the solution (i.e., point-point data whose local pose function changes significantly for small changes in pose).

The results of these experiments are given in Table 5.5. This table is in the same format as Table 5.4 and contains the same type of information: the pose error and the value of the local and global pose functions at the actual and estimated poses.

Table 5.5: Estimating pose using multiple types of sensing: the case with bad data added.^a

Experiment Number	Position Error (inches)	Orientation Error (degrees)	Sensing Type	Pose Function Value	
				at Actual Pose	at Estimated Pose
1	0.42	1.66	1t	0.66	2.22
			2p	0.04	0.04
			3t	0.01	0.04
			4v	0.71	0.48
			4v	0.59	0.58
			5v	1.33	0.41
			4v*	53.0	51.3
			Total	56.3	55.1
2	1.65	14.6	1v	1.38	1.18
			2t	0.28	0.24
			6p	0.002	0.86
			7r	0.55	0.18
			9t	0.62	11.2
			1p	0.31	2.02
			4v*	80.8	48.2
			Total	84.0	63.9
3	2.69	75.8	1t	0.41	19.8
			2p	0.11	1.66
			8v	0.004	30.5
			1v*	321.	192.
			Total	322.	244.

^aBad data is marked with an asterisk.

As shown in Table 5.5, the position and orientation were affected little by the bad data in the first two experiments, and they were considerably changed in the third. The relative strength of the bad data can be seen in the value of its local pose function at the actual pose. Note that in each case, its value at the estimated pose is also high, and when it considerably affects the pose estimate (i.e., in the third experiment), it also raises the values of the other local pose functions.

5.3.4 Improving a Pose Estimate using Additional Measurements

The purpose of our final two experiments was to demonstrate the increasing accuracy of the pose estimate as more data is added. In these experiments, a set of data was given to the fusion algorithm one measurement at a time, and with each new measurement, the pose was computed and the errors in position and orientation were recorded. In the first of these experiments all of the data was good data, and in the second the first measurement was bad and the rest were good (each used the same set of good data).

The results of these experiments are given in Tables 5.6 and 5.7. In each of these tables, each row represents a run of the fusion algorithm with one new measurement added. For the added measurement, the value of its local pose function at the actual pose is given as an indication of how good that measurement is (note that these values are independent of the other measurements). Also given in each row are the errors in position and in orientation for the computed pose.

Note that for each experiment, the position and orientation errors given in the first few rows are almost meaningless, as the algorithm does not yet have enough data to constrain the six degrees of freedom of object pose; the pose determined by the algorithm is largely dependent on the particular search path taken by the optimization algorithm.

Table 5.6: Improving the pose estimate by including additional data.^a

Number of Measurements	Data Added	Local Pose Function Value ^b	Position Error (inches)	Orientation Error (degrees)
1	1t	0.41	13.7	56.8
2	2p	0.11	28.2	116.
3	8v	0.004	0.28	0.30
4	9t	0.48	0.08	2.74
5	4v	0.59	0.18	0.58
6	6p	0.002	0.16	0.47
7	3t	0.49	0.19	0.86
8	5v	0.35	0.20	0.75
9	7r	0.06	0.19	0.79
10	3p	0.10	0.19	0.76
11	1v	1.38	0.25	0.34
12	6t	0.004	0.20	0.27
13	9p	0.01	0.19	0.28
14	2t	1.10	0.13	0.14
15	1p	0.49	0.13	0.26

^aEach row represents one run of the fusion algorithm and includes all data in the rows above it.

^bThis represents the evaluation of the local pose function at the actual object pose for the added data.

Table 5.7: Improving the pose estimate by including additional data given an initial bad measurement.^a

Number of Measurements	Data Added ^b	Local Pose Function Value ^c	Position Error (inches)	Orientation Error (degrees)
1	1v*	321.	7.14	168.
2	1t	0.41	10.8	97.5
3	2p	0.11	4.27	156.
4	8v	0.004	2.70	75.8
5	9t	0.48	2.52	39.1
6	4v	0.59	2.17	2.18
7	6p	0.002	2.16	2.06
8	3t	0.49	1.50	3.80
9	5v	0.35	1.49	3.73
10	7r	0.06	1.46	3.82
11	3p	0.10	1.46	3.82
12	1v	1.38	1.46	3.57
13	6t	0.004	0.98	3.24
14	9p	0.01	0.84	2.85
15	2t	1.10	0.78	1.37
16	1p	0.49	0.63	0.63

^aEach row represents one run of the fusion algorithm and includes all data in the rows above it.

^bThe bad measurement is marked with an asterisk.

^cThis represents the evaluation of the local pose function at the actual object pose for the added data.

For the first experiment, a relatively good value of pose was found by the third measurement.⁷ Because the initial data was fairly accurate, the improvement with additional measurements was small. Note that each additional measurement “yanks” the solution point (in pose space) in some direction, so there is an apparent random component to this improvement. The measurements having the greatest value of their local pose function generally have a greater affect on the solution point. Also note that as more measurements are added, the power of each measurement to move the solution point is diminished, so the fluctuation of the pose error will decrease. If the data is unbiased (zero mean) and the actual pose was correctly determined, then the errors should average to zero as more measurements are taken.⁸

In the second experiment, the steady improvement in the solution with additional data is much more apparent. This is because the initial piece of bad data (marked with an asterisk) forced the early pose estimates to a value far from the actual pose, and each new piece of good data then places a pull on the solution point in approximately the same direction in pose space, the direction in which the actual pose lies. By the last measurement, the pose estimation is reasonably accurate as the good data has outweighed the bad to a large extent.

5.4 A Graphics Display for Human Visualization

In order to help the user visualize the pose computed by the fusion algorithm, the implementation included an option for creating line drawings (displayed on the

⁷ Note the data for the first three measurements is the same as that used for the third experiment of the last set.

⁸ In this experiment the final error was close to the accuracy of the determined value of the actual pose.

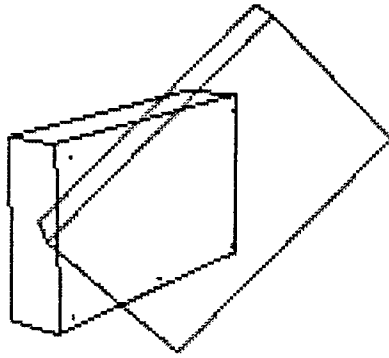
Perceptics Image Processor) of the panel in both its actual and estimated poses. In addition to displaying the edges of the panel, the six panel lights are also included to help distinguish between its front and back sides.

The user can change the viewing position and “camera focal length” and also has control over whether or not the hidden lines are to be included. The display is in color, with different colors used for the actual and estimated poses and for visible and hidden lines (if they are drawn).

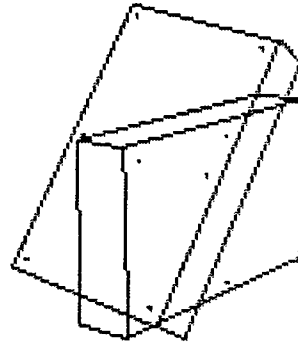
Figure 5.4 shows six examples of these line drawings which correspond to the first six poses computed in our final experiment (described by Table 5.7). Each drawing was made from the same viewpoint and with the same focal length. As can be seen in this figure, the poses computed after the first few measurements are far from the actual pose. This, of course, is due to the initial bad measurement and the small number of additional good measurements. It wasn't until the sixth measurement that the fusion algorithm had enough good data to arrive at a “reasonable” pose estimate in spite of the bad measurement.

5.5 Chapter Summary

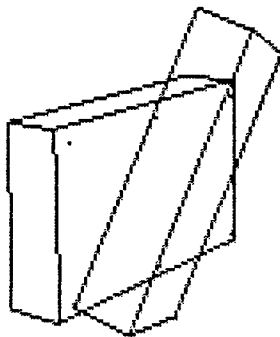
In this chapter we have described an implementation of the sensor fusion algorithm using the robotics workstation at University of Tennessee. We have also described a series of experiments which demonstrate that the fusion algorithm can determine object pose when given information from various combinations of sensory data. In addition, we demonstrated the use of the local pose functions in determining the degree of agreement among the sensory data.



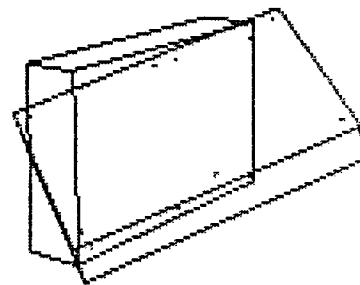
(a) 1 Measurement.



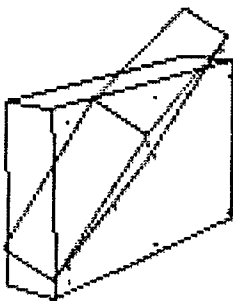
(b) 2 Measurements.



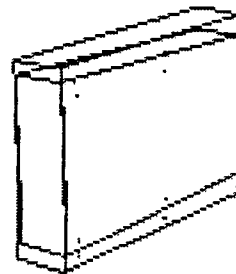
(c) 3 Measurements.



(d) 4 Measurements.



(e) 5 Measurements.



(f) 6 Measurements.

Figure 5.4: Example line drawings of panel pose. These drawings show the actual and computed panel poses after each of the first six runs of the final experiment.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This dissertation has described a method for combining local partial pose information from a variety of sensors to form a single (global) estimate of object pose. In this approach we used distance functions to represent estimated errors in uncertain parameters. The approach can be viewed in the following way: The parameters of each sensory measurement are written in terms of a *sensory feature* distance function which is based on both the measurement and the estimated errors in measurement. An *object feature* distance function is then defined in terms of this sensory feature distance function and the sensing model (written as a set of constraints describing the relationship between the parameters of the sensory feature and the object feature). Next a *local pose* distance function is defined in terms of the object feature distance function and the object model. Finally, the *global pose* function is defined as the sum of the local pose functions. The minimum of this global pose function (found numerically) provides the estimate of pose. Note that our method does not require that these functions be written explicitly in order to solve for pose; we only need to be able to evaluate them (and their first derivatives) for given values of their parameters.

An advantage of this approach is that it provides a consistent treatment of partial pose information originating from a variety of sensors. In this dissertation we developed the mathematics of this approach for certain classes of sensing, but other types of sensing could also be incorporated (by writing the sensory measurement as a distance function

and by describing the sensing model as a set of constraints relationships between the sensory parameters and the object feature parameters).

6.1 Potential Speed Improvements

The amount of time required by the algorithm to solve for global pose is roughly proportional to the number of contributing local measurements. Execution time could be reduced by using only a subset of the measurements; however, this can adversely affect the accuracy of the solution. There are several potential ways to reduce the execution time and maintain accuracy as the number of measurements increases.

In some instances multiple sensory measurements of a given object feature can be combined into a single measurement to reduce the number of local pose functions which must be evaluated at each iteration of the optimization routine. For example, a range image generated by a laser range scanner can produce a large number of point samples of the surface of an object. Instead of using each point as a separate point-plane measurement, we could fit a plane through the set of points and create a single plane-plane measurement.

In addition, the sensing class of the local pose function created by this technique is sometimes easier to work with than the sensing class of the functions it replaced. For example, in sensing a point object feature with stereo vision, two line-point problems (a sensing type for which iterative methods are required) can be converted to a single point-point problem (a type which is easily evaluated).

In some applications a current pose estimate needs to be updated (to improve accuracy) as new data is added. Because the resulting global pose function will be equivalent

to the old global pose function plus the new local pose functions, the computational requirements can be reduced by replacing the old pose function with its Taylor series approximation about the old solution point (an approximation to the matrix of second partials is provided by NPSOL). At each step of the iterative process, this approximate function is then evaluated in place of the local pose functions it represents.

Finally, we mention that this algorithm is easily implemented on a parallel architecture. Because the global pose is a sum of the local poses, the task of evaluating the local pose functions can be divided among processing nodes and the results from each simply added to a common accumulator. If a balanced partition is made, then the speed increase should be approximately proportional to the number of processors available.

6.2 Recommendations for Future Work

The classification of sensor data proposed in Chapter 3 allowed only for the sensing of point, line, and plane object features. One possible area of research would be to extend the use of the fusion method to include other object feature types (e.g., spheres, cylinders, and other quadric surfaces).¹ In addition, the present method allows only point, line, and plane sensory features; however, other types can also be of value. One example is applicable to our ultrasonic range sensor. In our experiments we assumed the detected object lay on a sensory plane; however, a spherical surface would provide a better approximation, especially if the beam spread is large.

Another possible extension of this method would be to incorporate statistical dependencies between sensory measurements (our treatment assumed the measurements

¹ Some special cases, such as spheres and cylinders sensed with point features, appear relatively easy to implement.

were statistically independent). An example of statistical dependence is when several object points are sensed in a single visual image—any errors in camera position and orientation would be the same for each measurement. Ignoring such information can sometimes lead to suboptimal results.

Another topic for future research is to develop a technique for interpreting the derivatives of the global pose function in the neighborhood of the minimum. These derivatives provide an indication of the the expected accuracy of our solution; the quicker the global pose function increases in value as the pose vector changes in a given direction, the smaller the expected pose error in that direction. This information can be useful in determining if pose is known well enough for a given application, and also in selecting further sensory measurements if pose is not known well enough. Interpretation is not simple due to the fact that pose is a seven dimensional quantity (with one constraint relationship) and the contribution from the orientation component is nonlinear.

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] J. K. Aggarwal and M. J. Magee. Determining Motion Parameters Using Intensity Guided Range Sensing. *Pattern Recognition*, 19, no. 2:169-179, 1986.
- [2] P. K. Allen. *Object Recognition Using Vision and Touch*. PhD thesis, University of Pennsylvania, Philadelphia, PA, September 1985.
- [3] P. K. Allen. Sensing and Describing 3-D Structure. In *International Conference on Robotics and Automation*, pages 126-131, San Francisco, California, April 1986.
- [4] N. Ayache, O. Faugeras, and B. Faverjon. A Geometric Matcher for Recognizing and Positioning 3-D Rigid Objects. In *Intelligent Robots and Computer Vision*, pages 152-159, 1984.
- [5] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [6] P. J. Besl and R. C. Jain. Three-Dimensional Object Recognition. *Computing Surveys*, 17, no. 1:75-145, March 1985.
- [7] G. L. Bilbro and W. E. Snyder. Linear Estimation of Object Pose from Local Fits to Segments. In *International Conference on Robotics and Automation*, pages 1747-1752, Raleigh, North Carolina, March 1987.
- [8] G. L. Bilbro and W. E. Snyder. *Linear Estimation of Object Pose from Local Fits to Segments*. Technical Report, Dept. of Electrical and Computer Engineering, North Carolina State University, December 1986.
- [9] R. M. Bolle and D. B. Cooper. On Optimally Combining Pieces of Information, with Application to Estimating 3-D Complex-Object Position from Range Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8, no. 5:619-638, September 1986.
- [10] R. A. Brooks. Visual Map Making for a Mobile Robot. In *International Conference on Robotics and Automation*, pages 824-829, March 1985.
- [11] R. L. Burden, J. D. Faires, and A. C. Reynolds. *Numerical Analysis*. Prindle, Weber & Schmidt, Boston, Massachusetts, 2 edition, 1981.
- [12] R. Chatila. Mobile Robot Navigation: Space Modeling and Decisional Processes. In *The Third International Symposium on Robotics Research*, pages 374-378, The MIT Press, Gouvieux, France, October 1986.
- [13] R. Chatila and J. Laumond. Position Referencing and Consistent World Modeling for Mobile Robots. In *International Conference on Robotics and Automation*, pages 138-145, St. Louis, Missouri, March 1985.

- [14] R. T. Chin and C. R. Dyer. Model-Based Recognition in Robot Vision. *Computing Surveys*, 18, no. 1:66-108, March 1986.
- [15] A. R. de Saint Vincent. A 3D Perception System for the Mobile Robot Hilare. In *International Conference on Robotics and Automation*, pages 1105-1111, San Francisco, California, April 1986.
- [16] R. O. Duda, D. Nitzan, and P. Barrett. Use of Range and Reflectance Data to Find Planar Surface Regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1, no. 3:259-271, July 1979.
- [17] H. Durrant-Whyte. Integration of Distributed Sensor Information: An Application to a Robot-System Coordinator. In *International Conference on Cybernetics and Society*, pages 415-419, Tucson, Arizona, November 1985.
- [18] H. F. Durrant-Whyte. Consistent Integration and Propagation of Disparate Sensor Observations. In *International Conference on Robotics and Automation*, pages 1464-1469, San Francisco, California, April 1986.
- [19] H. F. Durrant-Whyte. *Integration, Coordination and Control of Multi-Sensor Robot Systems*. Kluwer Academic Publishers, 1988.
- [20] H. F. Durrant-Whyte. Uncertain Geometry in Robotics. In *International Conference on Robotics and Automation*, pages 851-856, Raleigh, North Carolina, March 1987.
- [21] R. O. Eason. *Implementation of the Multisensor Fusion Algorithm for Determining Object Pose*. Technical Report TR-EE-88-51, Dept. of Electrical and Computer Engineering, University of Tennessee, August 1988.
- [22] O. D. Faugeras. New Steps Toward a Flexible 3-D Vision System for Robotics. In *Proceedings of the 7th International Conference on Pattern Recognition*, pages 796-805, Montreal, Canada, July-August 1984.
- [23] O. D. Faugeras and M. Hebert. A 3-D Recognition and Positioning Algorithm Using Geometrical Matching Between Primitive Surfaces. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 996-1002, Karlsruhe, West Germany, August 1983.
- [24] O. D. Faugeras and M. Hebert. The Representation, Recognition, and Locating of 3-D Objects. *The International Journal of Robotics Research*, 5, no. 3:27-52, Fall 1986.
- [25] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Sensing, Vision, and Intelligence*. McGraw-Hill Book Company, 1987.
- [26] P. C. Gaston and T. Lozano-Pérez. Tactile Recognition and Localization Using Object Models: The Case of Polyhedra on a Plane. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 257-266, May 1984.

- [27] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. *User's Guide for NPSOL (Version 2.1): A Fortran Package for Nonlinear Programming*. Technical Report SOL 84-7, Systems Optimization Laboratory, Department of Operations Research, Stanford University, September 1984.
- [28] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, New York, NY, 1981.
- [29] G. Giralt, R. Chatila, and M. Vaisset. An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots. In *The First International Symposium on Robotics Research*, pages 191–214, The MIT Press, Bretton Woods, NH, 1984.
- [30] R. C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison-Wesley Publishing Company, Reading, Massachusetts, 2 edition, 1987.
- [31] W. E. L. Grimson and T. Lozano-Perez. Localizing Overlapping Parts by Searching the Interpretation Tree. *IEEE Transactions on PAMI*, PAMI-9, no. 4:469–482, July 1987.
- [32] W. E. L. Grimson and T. Lozano-Pérez. Model-Based Recognition and Localization from Sparse Range or Tactile Data. *The International Journal of Robotics Research*, 3, no. 3:3–35, 1984.
- [33] T. Henderson, C. Hansen, and B. Bhanu. The Specification of Distributed Sensing and Control. *Journal of Robotic Systems*, 2, no. 4:387–396, 1985.
- [34] T. Henderson and E. Shilcrat. Logical Sensor Systems. *Journal of Robotic Systems*, 1, no. 2:169–193, 1984.
- [35] T. C. Henderson and W. S. Fai. *Pattern Recognition in a Multi-Sensor Environment*. Technical Report UUCS 83-001, University of Utah, July 1983.
- [36] T. C. Henderson, W. S. Fai, and C. Hansen. MKS: A Multisensor Kernel System. In *IEEE Transactions on Systems, Man, and Cybernetics*, pages 784–791, September/October 1984.
- [37] T. C. Henderson, C. D. Hansen, and W. S. Fai. Organizing Spatial Data for Robotics Systems. *Computers in Industry*, 6, no. 5:331–344, October 1985.
- [38] P. Horaud and R. C. Bolles. 3DPO's Strategy for Matching Three-Dimensional Objects in Range Data. In *IEEE International Conference on Robotics*, pages 78–85, 1984.
- [39] B. K. P. Horn. Closed-Form Solution of Absolute Orientation Using Unit Quaternions. *J. Opt. Soc. Am. A*, 4, no. 4:629–642, April 1987.
- [40] B. K. P. Horn. *Robot Vision*. McGraw-Hill Book Company, 1986.

- [41] R. A. Jarvis. A Perspective on Range Finding Techniques for Computer Vision. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 122-139, March 1983.
- [42] A. C. Kak, B. A. Roberts, K. M. Address, and R. L. Cromwell. Experiments in the Integration of World Knowledge With Sensory Information for Mobile Robots. In *International Conference on Robotics and Automation*, pages 734-740, Raleigh, North Carolina, March 1987.
- [43] G. A. Korn and T. M. Korn. *Mathematical Handbook for Scientists and Engineers*. McGraw-Hill Book Company, 1968.
- [44] D. J. Kriegman, E. Triendl, and T. O. Binford. A Mobile Robot: Sensing, Planning and Locomotion. In *International Conference on Robotics and Automation*, pages 402-408, Raleigh, North Carolina, March 1987.
- [45] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.
- [46] R. C. Luo, M. Lin, and R. S. Scherp. The Issues and Approaches of a Robot Multi-Sensor Integration. In *International Conference on Robotics and Automation*, pages 1941-1946, Raleigh, North Carolina, March 1987.
- [47] M. J. Magee, B. A. Boyter, C. Chien, and J. K. Aggarwal. Experiments in Intensity Guided Range Sensing Recognition of Three-Dimensional Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-7, no. 6:629-636, November 1985.
- [48] A. Mitiche and J. K. Aggarwal. Multiple Sensor Integration/Fusion Through Image Processing: A Review. *Optical Engineering*, 25, no. 3:380-386, March 1986.
- [49] B. T. Perricone and et al. *Representation and Processing of Spatial Knowledge*. Technical Report TR-1275, Department of Computer Science, University of Maryland, College Park, MD, May 1983.
- [50] E. Pervin and J. A. Webb. Quaternions in Computer Vision and Robotics. In *Computer Vision and Pattern Recognition*, pages 382-383, Washington, D.C., June 1983.
- [51] A. P. Sage. *Optimum Systems Control*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1968.
- [52] E. Salamin. Application of Quaternions to Computation with Rotations. 1979. Internal Working Paper, Stanford, AI Laboratory.
- [53] R. C. Smith and P. Cheeseman. *On the Representation and Estimation of Spatial Uncertainty*. Technical Report, SRI International, September 1985.

- [54] R. C. Smith and P. Cheeseman. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, 5, no. 4:56-68, 1986.
- [55] K. W. Spring. Euler Parameters and the Use of Quaternion Algebra in the Manipulation of Finite Rotations: A Review. *Mechanism and Machine Theory*, 21, no. 5:365-373, 1986.
- [56] J. Stuelpnagel. On the Parametrization of the Three-Dimensional Rotation Group. *SIAM Review*, 6, no. 4:422-430, October 1964.
- [57] J. T. Tou and R. C. Gonzalez. *Pattern Recognition Principles*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1974.

VITA

Richard Odell Eason was born in Anniston, Alabama on July 18, 1956. He was raised in Oak Ridge, Tennessee and graduated from Oak Ridge High School in June, 1974. The following September he entered The University of Tennessee, Knoxville. In 1975 he was the recipient of the Allen Medal given each year to the outstanding freshman mathematics student at the University, and in 1976 and 1977, he was a Fred M. Roddy Scholarship recipient. In March, 1978 he graduated with highest honors with a Bachelor of Science degree in Electrical Engineering.

After accepting a research assistantship and teaching assistantship at the University of Tennessee, Knoxville, he began his study toward a Master of Engineering in Electrical Engineering. This degree was awarded in June, 1980.

He then went to work at Zilog, Inc. in Cupertino, California as a VLSI design engineer. There he designed Zilog's Z8581 Clock Generator and Controller. In 1982 he returned to the University of Tennessee to enroll in the doctoral program. While in this program, he worked at the Robotics and Integrated Manufacturing Laboratory under the direction of Dr. R. C. Gonzalez. In 1984 he was the W. O. Leffell Scholarship recipient. The Doctor of Philosophy degree was awarded in August, 1988.

The author is a member of Phi Kappa Phi, Tau Beta Pi, and Eta Kappa Nu Honor Societies, the IEEE, and the IEEE Computer Society.