Nonlinear Probabilistic Estimation of 3-D Geometry from Images

by

Ali Jerome Azarbayejani

S.B., Aero. Eng., Massachusetts Institute of Technology (1988)
S.M., Aero. Eng., Massachusetts Institute of Technology (1991)
S.M., EECS, Massachusetts Institute of Technology (1991)

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1997

Author ____

Program in Media Arts and Sciences
School of Architecture and Planning
10 January 1997

Certified by ____

Alex Pentland
Academic Head and Toshiba Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by ____

Stephen A. Benton
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Nonlinear Probabilistic Estimation of 3-D Geometry from Images

by

Ali Jerome Azarbayejani

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on 10 January 1997, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis addresses the problem of designing practical vision systems for computing 3-D representations of scenes from images. A theoretical understanding of the basic geometric problems has been emerging over decades, yet difficulties associated both with automatically extracting 2-D geometric features from images and in computing stable 3-D geometric descriptions from 2-D features have impeded progress in building reliable vision systems that perform useful tasks outside a laboratory setting.

The 3-D computational problems are geometrically poorly leveraged by the image features, involve nonlinear relationships, and have non-Euclidean state domains. To model such domains, a manifold-tangent framework is developed which allows non-Euclidean state manifolds to be locally described by Euclidean tangent spaces. Coupled with local probabilistic uncertainty models in tangent space, the manifold-tangent representation is used to generalize classic iterative (Levenberg-Marquardt) and recursive (Kalman) estimators to operate on nonlinear implicit constraints involving non-Euclidean domains, such as those found in the 3-D vision geometry problems.

Using this framework, a well-behaved model is developed for the classic problem of recovering 3-D motion and structure from point correspondences, augmented to include recovery of camera focal parameters as well. Extensive computer simulations are carried out to characterize performance in a controlled setting. Several prototype applications are developed to verify performance in the field and to illustrate the practical capabilities afforded by reliable 3-D geometry estimation from uncalibrated imagery; these include a 3-D head tracking system for human-computer interface, a video-based tool for building 3-D texture-mapped models, and a film post-production tool for matching 3-D camera motions.

The automatic feature extraction issue that lingers for point-based systems motivates analysis of a second class of problem, that of computing 3-D geometry from blob correspondences. Blob features are moment-based spatial descriptions of objects in images that, unlike traditional features, can be stably and efficiently extracted from noisy image streams in real-time. The modeling principles used for points are used in a similar way to develop a computational model for blobs. Performance in estimating 3-D camera geometry and 3-D blob geometry is characterized through computer simulations and a real-time self-calibrating 3-D person-tracker application developed using the model.

Thesis Supervisor: Alex Pentland
Title: Academic Head and Toshiba Professor of Media Arts and Sciences
Program in Media Arts and Sciences

Doctoral Dissertation Committee

Thesis Supervisor

Alex Pentland
Academic Head and Toshiba Professor of Media Arts and Sciences
Program in Media Arts and Sciences

Thesis Reader

W. Eric L. Grimson
Professor of Computer Science and Engineering
Department of Electrical Engineering & Computer Science

Thesis Reader

Takeo Kanade
Director, The Robotics Institute
U.A. and Helen Whitaker Professor of Computer Science
Carnegie Mellon University

Dedicated to
Martin Friedmann
1965 – 1995

Contents

List of Figures

Chapter 1

Introduction

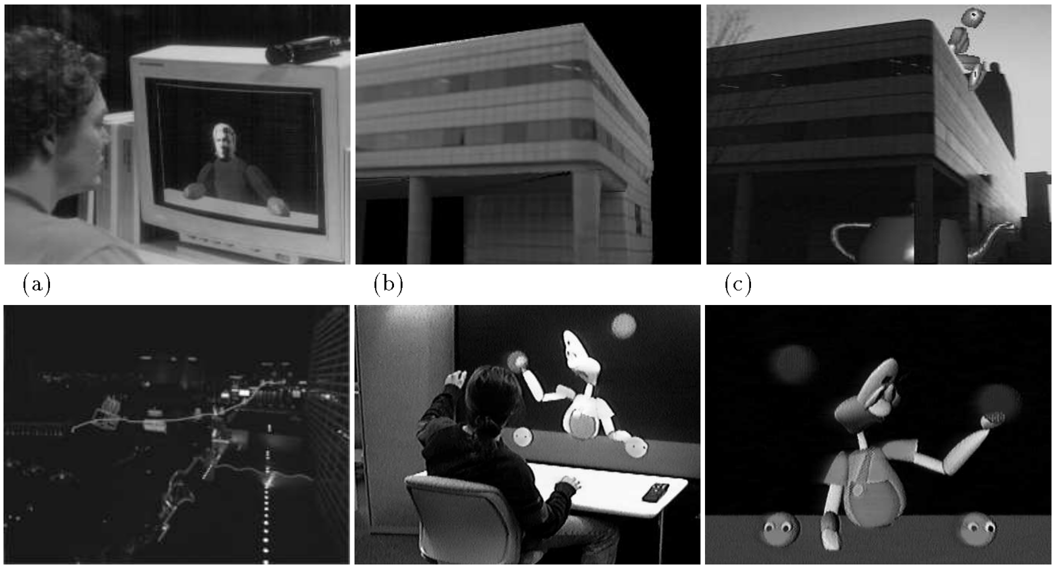
People have been fascinated with vision for centuries, but until the recent advent of the computer age have not had the tools with which to build their own "seeing machines". Electronic imaging now provides machines with the capability to capture images of light measurements in very much the same way as the human retina, and digital computers allow machines to process the images potentially much like the human mind. The hope is that machines might someday through these tools obtain a visual perception of their surroundings much like that which most humans depend on every day for most aspects of their lives.

Motivations behind building seeing machines are as varied as the overlapping disciplines associated with the field of computer vision. Cognitive scientists want to build seeing machines to investigate theories of human perception and cognition. Artificial intelligence (AI) scientists want to build machines that have perceptual and mental capabilities of people, presumably so the machines can do more sophisticated work for us. Robotics scientists want their machines to be more autonomous through vision and therefore more useful. Other computer scientists simply want better interfaces to their computing machines. And many computer users who manipulate large amounts of imagery — including stills, video, and film — would like their computers to more intelligently search, understand, and manipulate imagery instead of merely pushing around bits.

However, building a general-purpose seeing machine apparently involves understanding the entire problem of how intelligent behavior works. Since we do not currently have a sufficient understanding of how the human mind represents the world, how it remembers and recognizes objects, or how it classifies and labels things, it is not surprising that we have not succeeded in building a significantly general-purpose seeing machine. But there are many aspects of vision that can adequately be separated from the rest of the thorny issues of intelligence and it has been disappointing that even these special-purpose visual tasks have been difficult to implement reliably.

In particular, a great deal of computer vision work over the last two decades has been devoted to obtaining 3-D spatial descriptions of scenes from images. In fact, many scientists originally considered the prime function of vision to be building these 3-D descriptions, upon which other higher-level functions, such as object recognition, could function. Today's philosophies about vision are more diverse, yet the scientific questions about human 3-D perception and the engineering questions about machine 3-D perception persist while the success at building 3-D seeing machines remains quite limited. It is this lack of applicability and reliability of the basic vision algorithms that is the core motivation for the theoretical innovation in this thesis.

This research, for example, was originally motivated by the desire to develop a visually-based human-computer interface, and the first task was to be that of tracking a person's



(a) (b) (c) (d) (e) (f)

Figure 1-1: The applications driving the research and serving as benchmarks for the underlying theoretical treatment include (a) 3-D head tracking for human-computer interface, (b,c) models-from-video for computer graphics and video post-production, (d) 3-D motion matching for film post-production, (e,f) camera self-calibration for visual interface systems and 3-D person-tracking for human-computer interaction and visually controlled graphics.

head in 3-D. The location of the person's head could be used for rendering a model of the head with the correct motion, leading to a paradigm for low-bandwidth teleconferncing, or for controlling the viewpoint of the user's display, which leads to a paradigm for visually controlled graphics, or "virtual holography".

A traditional approach to this problem, like many other geometry-based problems in vision, is tracking 2-D features in images of the person's head and using some internal model of the imaging process to then estimate 3-D motion from these features. For this system, special-purpose commercial hardware was used to track feature points on the face in real time and the estimation of 3-D motion from these feature tracks was the principal research problem.

When the camera and the head shape are known, a straightfoward modeling and estimation approach works very well. In general, however, these are not known and the resulting "structure-from-motion" problem arises, which has been considered a fundamentally difficult problem in our field. Despite receiving a great deal of attention in the literature and attracting a wide variety of modeling and estimation solutions, practical applications have been limited due to poor reliability and accuracy.

An investigation into the nature of the problem when neither motion, object shape, nor focal length is known has led to most of the basic elements of the modeling and estimation approach of this thesis. Most importantly, considerations of numerical conditioning fostered a novel approach to modeling of pointwise structure and camera orientation and a careful analysis of 3-D rotational representation and estimation motivated the manifold-tangent framework for modeling the curved state spaces that occur in 3-D vision geometry problems. Together these resulted in development of the nonlinear probabilistic framework for 3-D geometry estimation from point correspondences that is a basis for recovering 3-D motion, 3-D pointwise structure, and camera focal parameters from uncalibrated imagery.

The benchmark of success of this framework has been the successful application of the new formulation to real-world problems including the real-time head-tracking system and several interactive systems for processing film and video. The *models-from-video* application consists of building texture-mapped polygonal models from moving video of objects for the purpose of obtaining realistic and metrically correct computer graphics models for virtual environments. The *match-move* application consists of recovering camera trajectories of moving cameras for the purposes of image stabilization, integration of film and computer graphics, and other film post-production tasks.

However, the demonstrated success of the estimation strategy for points is somewhat undermined by the lingering problem that points are difficult to reliably correspond in real imagery, particularly in the low-resolution video quality images that most computer vision systems must work with. In the head-tracking system, special-purpose commercial hardware did the job, but the range of motion that could be successfully tracked was somewhat limited. In the models-from-video application, a similar template-based tracking strategy was used, but because of jerky camera motions, a large 3-D viewpoint change, and minimalist implementation, many automatic tracking mistakes either had to corrected by hand using a graphical user interface or resulted in the feature track being automatically discarded. Likewise, the film post-production system was mostly automatic but required human interaction at keyframes to constrain the tracking. These interactive systems are perfectly acceptable for the imagery production applications, but for the grander goal of autonomous vision systems, further progress on geometrical feature extraction is required.

In particular, to make further progress on human-computer interface it has been necessary to take a new look at feature extraction. The template-based point tracking used in the head tracking system had the advantage of using high-resolution close-up images of an essentially rigid object. To track people moving freely about a room or even sitting at a desk requires tracking nonrigid articulated bodies through a larger field of view, thus at lower image resolution.

To solve this, a new formulation of 3-D geometry estimation has been devised using a new type of feature that, unlike traditional features—points, edges, lines, and contours—has been shown to be reliably and cheaply extractable from even low-quality images of complex objects such as moving people. Blob features were motivated in fact by a laboratory prototype system for real-time 2-D visual person tracking. A probabilistic modeling approach using color and spatial distributions is the basis for the system which reliably extracts image regions corresponding to parts of a person from noisy low-resolution video. People using this system can move freely without catastrophic failure of the tracking system.

These blobs can be modeled as moment-based distributions, both in 3-D and 2-D, and can represent either probability of occupancy or solid-body geometry. Geometric relationships between the 3-D distributions and corresponding 2-D distributions facilitate the subsequent estimation of parameters of the 3-D object blobs from the 2-D "feature" blobs. Many of the estimation and modeling issues are the same as in the points problem and the basic framework developed for points extends readily to the blob problem.

The success of the blobs formulation is again benchmarked by the success of the applications, including a scheme for self-calibration of a multi-camera user interface and real-time 3-D tracking of a moving person. The person-tracker uses two cameras and two standard workstations, runs at 30Hz and has been used by hundreds of people. The self-calibration technique operates by obtaining blob correspondences from a moving person and is therefore very simple to use.

1.1 Contributions

The research results presented in this document offer several potentially important contributions to the field of computer vision, outlined here.

3-D geometry is a non-Euclidean manifold First is the concept that the 3-D geometry used in vision cannot be adequately represented as a Euclidean state space. The 3-D rotation in particular is non-Euclidean in nature and can only be represented as a curved manifold. Another way of saying this is that any numerical parameterization of the problem must be augmented with constraints, which would lead to a traditional *constrained nonlinear optimization* approach to the problem. These two interpretations are somewhat equivalent, but the manifold formulation is more useful in practical problems and is more naturally extended to probabilistic recursive estimation.

Most previous work has applied standard batch or recursive estimation techniques, which are all based on Euclidean domains, to the 3-D geometry estimation problems by modifying them to normalize the rotation quaternion at each iteration (for batch estimation) or each time step (for recursive estimation). This can be thought of as a poor ad hoc method of doing constrained optimization and could, e.g., be improved by using more sophisicated traditional constrained optimization formulations.

However, the problems become more complicated when probabilistic representations are required. Traditional constrained optimization techniques are not based on probabilistic models. For recursive estimators, not only does the estimator have to maintain a single state on a manifold but it also has to describe a probability distribution over the state space. Since traditional probabilistic methods are based on Euclidean domains, usually Gaussian distributions, they simply do not apply straightforwardly. There are no standard distributions, like the Gaussian, that span the rotation manifold.

The solution offered in this document is the manifold-tangent model for curved state spaces, in which state manifolds are augmented by a set of tangent hyperplanes at each state, each of which is a local Euclidean domain in which traditional probabilistic models and estimation strategies apply.

The model is locally Euclidean and therefore facilitates natural generalization to curved manifolds of most of the classic nonlinear estimation algorithms, which are themselves based on local properties. For the purposes of this research, the classic batch (Levenberg-Marquardt) and recursive (Kalman) estimators are generalized in this way. These generalizations have some precedent — e.g. the generalized batch algorithm is another way of looking at constrained optimization — but for the computer vision field, they constitute a new way of solving the probabilistic estimation problem for 3-D geometry and may lead to new ways of thinking about the problem.

Virtual plane camera model One of two contributions of this thesis to advancement of thinking regarding the classic point-based structure-from-motion estimation problem is a novel model for the perspective camera internal and relative orientation which is designed to remain numerically well-conditioned for any focal length, including an infinite one (orthographic projection).

This type of camera model is particularly important when focal length is unknown and was motivated by the problem of estimating geometry from uncalibrated imagery. The model formulation also has implications for choosing the scaling of parameters which may prove to be important in future work.

Reference frame point structure model The second contribution to the point-based problem is the characterization of tracked points as a location in a reference image (typically the first image in which the features were selected) plus a depth.

The model derives from the way features are actually tracked in practice, i.e. by identifying an image point in a reference image and tracking it. For a group of image points, the unknowns are the depths of those image points in the reference frame, the focal parameters of the camera, and the relative orientations of subsequent frames. The measurements are the locations of those points in subsequent frames.

The model has many fewer parameters than traditional object-based formulations and in particular has more constraints than unknowns at each step of a recursive filter. The consequence is that the model describes the same geometry, yet is more stable and less reliant on initial conditions, leading to greater success in application to practical problems.

Blob modeling and 3-D estimation Another potentially significant contribution is the introduction of the "blob" geometric primitive for 3-D estimation. Blobs are interesting because they can be reliably and efficiently extracted and tracked in real imagery and because they can provide a probabilistic, rather than solid-body, interpretation of 3-D geometry.

The motivation for using blobs for 3-D geometry arose from an actual video processing system that extracted blobs as features on moving people in a complex background environment. Using the geometric information contained in 2-D blobs this research has led to a means for obtaining 3-D geometric properties of blobs from corresponding 2-D blobs.

Blobs represent a departure from traditional thinking about features because instead of using sharp local geometric features such as points, edges, lines or contours of an object, blobs represent the spatial moments of the entire object directly and use these as the pertinent geometric information.

Furthermore, by representing 2-D and 3-D objects in this way, there is a spatial probabilistic interpretation of the objects which represents a departure from traditional notions of solid-body geometry or surfaces as 3-D geometric primitives. The blob moments can be mathematically interpreted as the statistics of a Gaussian distribution, which can in turn be physically interpreted as the probability distribution of occupancy for either a 3-D object or 2-D feature.

It remains to be seen what additional benefits or uses of this interpretation might be, but for the present purposes, the representation is convenient and very useful because it contains the required parameters for the location, orientation, and dimensions of the object.

Applications Finally, the applications that have been demonstrated as part of this research are probably the most visible, and perhaps most useful, contributions to the field because they have demonstrated some of the potential practical impact of the underlying technology, and of 3-D computer vision technology in general.

For example, a prototype 3-D system for image-based modeling has recently been field tested at a Hollywood production studio on several film shots for which the 3-D motion of the camera was unknown and required for post-production tasks. The successful extraction of the 3-D motion using the prototype system was used to verify the performance of the technique on completely uncontrolled imagery. The successful demonstration of computer vision technology may foster entirely new ideas about how to approach many media production problems because many of the industry's expensive computer tools are based around the assumption that 3-D modeling of imaged scenes is too difficult. The results of this research effort with Rhythm & Hues Studios show that not only can it be practical, but that the precision obtained can live up to the required standards of high-end film production.

In the mainstream computer vision field, the assumption that visually tracking people — especially in 3-D, in real-time, and without special hardware — is a very difficult problem may be challenged by the results of the prototype person-tracking application. The prototype system built as part of this research was recently demonstrated at the Siggraph '96 interactive exhibition where hundreds of people used the vision system to interact with six different interactive virtual worlds.

A critical part of making the person tracking system practical was the self-calibration process, which operates entirely by tracking blob features on the person. Demonstration of this application could change the way many people think about solving problems with computer vision because there is currently a prevailing assumption that camera calibration is a difficult and painful process. This assumption drives many aspects of computer vision research and is a large part of the motivation behind view-based techniques, for example. For many vision tasks, however, centimeter precision is sufficient, and since it has now been demonstrated that a calibration technique can achieve this in a matter of seconds and requires no props, the practical options for solving vision problems, especially human-computer interaction, may change substantially.

Chapter 2

Background

This research has precedent in a long history of work on 3-D geometry in vision and stands among a growing array of related current research.

This chapter attempts first to clarify the context of 3-D computational geometry in the study of vision, making the argument that 3-D representations of some sort are the most appealing theoretically and the most useful in practice.

Among the various approaches to modeling and estimating 3-D geometry, the second section argues further that a feature-based approach using nonlinear, dynamic, and probabilistic models offers the highest potential for a general framework that will be effective for a wide range of computer vision applications.

Finally, the last section reviews the major accomplishments in the field on visually tracking people to motivate the blob-based approach that is part of this research.

2.1 The role of 3-D geometry in vision

Obtaining a 3-D geometric representation of the visible world has always been one of the principal goals of computer vision, but it is worth trying to understand the role of 3-D geometry in vision and applications before entering into how people have tried to compute it. The important questions include whether a 3-D description is necessary and whether such a description is useful.

The pioneers of computer vision and artificial intelligence (AI) assumed without a great deal of hesitation that the primary role of vision is indeed to recover a 3-D representation of the visible world which could then be used to recognize objects and reason about the world. For example, Marr defined the vision process in a very geometrically oriented way:

> What does it mean, to see? The plain man's answer (and Aristotle's too) would be, to know what is where by looking. In other words, vision is the *process* of discovering from images what is present in the world, and where it is. (Marr [46])

and, indeed, this interpretation of the visual process pervaded the field in the 1980's as much computer vision research focused on the recovery of 3-D geometry as the basis for higher levels of processing, including recognition and reasoning.

For example, some early hallmark achievements included the basic understanding of recovering 3-D scene structure from stereo (e.g. Marr & Poggio [47]) and from motion (e.g. Ullman [78, 79]). Both of these geometric solutions relied on correspondence of features in images taken from different points of view. Other image cues including shading, retinal velocity fields (optical flow), and de-focus were also used to extract 3-D shape information [36].

In nearly all of this work the problems neatly separate into two sequential stages: feature extraction (point correspondence, image gradients, optical flow) and shape estimation. In the case of correspondence-based methods, both stages have been well understood theoretically for years but nonetheless have presented difficulties to those trying to build implementations. This meager progress towards building practical systems, in part, has led many researchers to doubt the requirement for 3D analysis as the initial step in vision off of which everything else builds.

In particular, it is unlikely that 3-D representations can be computed before segmentation and recognition because extraction of features is so intimately related to these processes. Understanding and organizing an image is not purely a matter of signal analysis, as "the correspondence problem" has usually been approached, because there are always an infinite number of 3-D interpretations for each signal; the correct one corresponds to regularities in the natural world which, in the human visual system, are either learned or acquired genetically [57]. Thus, ultimately 3-D analysis cannot happen in isolation; as a practical matter, however, it is difficult to study the entire visual perception problem at once, so 3-D analysis continues to be studied in isolation despite these theoretical drawbacks.

Other researchers have rebutted the role of 3-D analysis further and have suggested that 3-D representations may not be required at all. Two important emerging schools of thought include that of non-metric 3-D geometry and that of view-based (2-D) representation.

Non-metric geometric descriptions are 3-D descriptions, but not in the way we usually describe 3-D properties, i.e. locations, distances, and angles (Euclidean geometry). Recent developments on the recovery of non-metric structure invariants [39, 25, 64] arise from certain convenient linear relationships between points and lines in displaced cameras. The resulting descriptions of geometrical structure encode somewhat abstract relationships between 3-D points rather than what we consider to be actual physical locations of 3-D points.

This work is partially a reaction to the difficulty of the numerical estimation part of recovering 3-D geometry from traditional correspondences and has some attractive properties, primarily that computation is algebraically elegant and simple. However, this line of research still has all the problems associated with getting the correspondences in the first place. Also, since non-metric representations are somewhat abstract and do not have simple physical meaning, they are difficult to use as an end-product and difficult to compute with. For example, the solutions to certain numerical estimation problems involve using probabilistic models for describing uncertainty in physical parameters; the numeric values for these models are easy to specify for most problems since they correspond to real physical quantities that we as humans understand. Physically-based parameters also provide a common language that is useful for both relating computation to the real world and communicating information between visual subsystems.

The other school challenging traditional 3-D geometry is that of view-based representations. The basic tenet is that, since several different views of an object encode the 3-D geometry of the object, the views themselves constitute a representation of the object. This approach is attractive from a practical standpoint because the difficulties of 3-D estimation can be avoided completely and only images need be stored. This approach has been used successfully for some recognition tasks [77, 49, 9, 85, 20, 13, 83] and has even been proposed as a way of doing synthesis (computer graphics) [10].

However, the tasks that have been successfully solved using these techniques are either essentially 2-D in nature, for which a 2-D representation is entirely appropriate, or require a large number of images from various points of view to solve problems with a significant 3-D component. In the latter case, certain 3-D geometry is implicitly being used (presumed known) to specify the viewpoints of the multiple images; it is unclear that it would be easier to recover these viewpoints than to build an equivalently descriptive 3-D model from the images. It is also quite certain that a 3-D model containing the same information in the multiple images would be a far more compact description.

Thus, despite the branching of intellectual thought on internal representations for vision, there is still no alternate approach to implementing 3-D visual perception that is more theoretically appealing than the process of representing and computing 3-D geometry. The slow progress of the last decade or so can probably be attributed to various circumstances other than fundamental difficulty, some of which are addressed in this thesis. First, numerical estimation on traditional problems has proven difficult even when the geometry is well understood; the mathematical modeling issues of this thesis highlight the role of representation in achieving stable numerical estimation for pointwise geometry. Second, traditional features (points, lines, contours) upon which classic formulations are based are difficult to find and not particularly well-suited for natural imagery such as that of people; the investigation of this thesis into the geometric properties of blob features arises from trying to formulate a new estimation approach based on geometric features which have been shown ahead of time to be reliably extracted from real video imagery. Third, traditional formulations have over-emphasized the importance of precision and under-emphasized the importance of stable representations. Fourth, the traditional placement of 3-D recovery before segmentation and recognition is unrealistic. Fifth, integration of geometric information acquired from various cues (correspondence, shading, motion, internal models) has largely been ignored, leaving each separate module somewhat fragile.

These areas of continuing research, among others, suggest that 3-D geometrical representation and computation are subject to vast improvement and will indeed continue to play an important role in understanding human and computer visual perception. But, aside from studying the role of 3-D geometry in intelligence, there are many computer applications that drive this sort of technology.

Obvious applications include vision-based modeling, i.e. developing models from pictures for computer-aided design (CAD), computer graphics (CG), virtual environments (VE), and architectural surveying [6]. In these cases, the very goal is the 3-D geometric model and the only available input are the images. Although these problems closely resemble the vision problem, there is no debate as to the final representation because a 3-D representation is the stated goal.

Other applications include computer vision systems which do not need to be fully intelligent, but do need to be able to automatically and correctly interpret a scene and react accordingly. For example, there is great interest in tracking people in rooms for various reasons, including security and monitoring, marketing, and computer-human interface. It is not clear that 3-D representations are required for all tasks; for example, recognizing certain gestures in an interface application can often be done with 2-D processing. However, other actions, such as pointing at an object requires a 3-D analysis except in special restricted cases.

Since a 3-D description contains all the information of any set of 2-D projections and is more compact, there is little reason not to use a 3-D description if it is available. Thus for most of the cases in which there is a choice of representations, 3-D approaches will likely win out eventually because they are more general and their output is more readily useful to users.

2.2 Modeling and estimation of 3-D geometry

In computer vision there have been many ways of approaching mathematical modeling and estimation of 3-D geometry. The choices of modeling critically affect what kind and quality of information can be extracted and what kind and quality of input can be processed.

This section motivates the use of corresponding geometric features in multiple images along with nonlinear, dynamic, and probabilistic modeling as the most fruitful approach to solving practical problems.

2.2.1 Single versus multiple images

The first way of distinguishing the various techniques is whether they attempt to infer 3-D geometry from one viewpoint of a scene or from multiple viewpoints.

The information contained in a single 2-D image of a scene is inherently ambiguous in the third dimension. If a computer is given no rules about the natural structure of the 3-D world, there is an infinite number of valid interpretations of how geometry projects into any image and 3-D interpretation is impossible [57]. The reason people can understand 3-D scenes from single images, such as photographs, is because we have an understanding of 3-D natural constraints and have been trained through our experience in the world to visually interpret them properly.

The implication for computational geometry is that 3-D interpretation from single images requires some prior internal modeling of the 3-D world. Indeed, some of the early work in computer vision, most notably that of Roberts [58], used arrangements of features in the image along with known 3-D properties of solid objects to infer the 3-D pose of 3-D objects from a single image.

However, the "blocks world" of these early experiments dealt with a highly limited toy world and the desire for processing a wider variety of natural scenes inspired work towards more general representations, including work using multiple images from different viewpoints.

Traditional *stereo* (inspired from animate vision) consists of simultaneous pictures from two closely displaced viewpoints, *motion* consists of multiple pictures from closely displaced times taken from a single camera of a moving scene (or from a moving camera, or both), and *wide baseline stereo* consists of a collection of pictures from widely spaced viewpoints.

Binocular stereo and motion are both inspired by animate visual systems and therefore have some natural relevance to the visual and intelligence process. Wide baseline stereo is not natural, but very useful for computers, as in the fields of photogrammetry and surveying where multiple pictures are taken of a scene to build 3-D models to high precision.

The underlying geometrical reason that multiple images from binocular sensors or motion are powerful is that, unlike with single images, 3-D geometry can be recovered from image-image correspondences alone with no prior models or understanding of the 3-D world. Therefore, some sorts of general 3-D representations can theoretically be built without the need to have an internal database of all types of objects in the scene. In fact, this idea was largely the basis for Marr's bottom-up theory of computational vision [46].

In practice the approach is not so easily implemented, partly because the numerical problems are poorly conditioned [30], but because the use of multiple images provides a more general framework for studying and understanding 3-D geometric estimation it has become a more popular vein of research and is the basis for the research presented here. Using specific world knowledge, as required in single image analysis, is much more difficult because it is intimately related to so many other aspects of intelligence, including object recognition, representation, memory, knowledge, and learning. Therefore, it is currently practical only in computer systems operating in highly limited visual domains.

Model-based and image-based analysis can benefit from each other and ultimately should be merged in applications where world object types are limited in number. This research focuses on the problems of image-based analysis alone, seeking reliability and probabilistic measures of precision that can eventually be used to merge these techniques.

2.2.2 Symbolic modeling

Among the many ways of processing multiple images, the first principal distinction between methods is whether they use *iconic* or *symbolic* representations to describe the images.

In traditional stereo, *disparity* measurements are an iconic representation of image-image correspondences at each pixel [36]. In motion analysis, *optical flow* is an iconic representation of image-image correspondence at each pixel [36]. In both domains, symbolic *features* are an alternative that provide sparse scene-oriented correspndences.

With only two images, the two representations of correspondence are somewhat equivalent, but with more images, the two types of image measurements provide different quality of information. The right choice for a particular application depends largely on the goals of the application.

Iconic correspondence is image-centered whereas symbolic correspondence is scene-centered. The advantage of iconic correspondence is that a large number of pixels contribute to the compuation. (A disadvantage on serial computers is that all of these contributions must be computed!) The disadvantage is that the lack of a scene-centered representation results in image-centered representations of 3-D, i.e. depth maps. Although these are useful instantaneous representations, there are two problems to going beyond the depth map.

The first arises from the well-known scale ambiguity [36] in 3-D geometry recovery from two images. In motion problems, this is usually expressed as "only the direction of translation can be recovered", i.e. the magnitude of the translational motion is ambiguous, as is the scale of the scene. For any optical flow field, and associated 3-D interpretation of camera motion and scene structure, the same flow field occurs if the translational motion and scene structure are scaled by the same factor [36]. Thus, rotational motion and direction of velocity relative to the instantaneous scene description can be determined uniquely, but scene structure and absolute motion cannot.

For stereo, there is typically a known baseline, so scale factor is not a problem and absolute instantaneous motion can be determined. However, there is still the problem that the scene representation is image-based. Again, a single depth map is a good instantaneous representation, but if the goal is to refine the scene geometry or to track absolute 3-D motion, a 3-D scene representation must be instantiated as in [33].

Thus, iconic modeling can be useful in some limited cases, but when 3-D scene structure or absolute 3-D motion tracking is the goal, symbolic modeling is required. For that reason, feature-based techniques are generally more useful. They are also more common because they are usually more computationally efficient, particularly on serial computers.

There are good perceptual reasons for favoring symbolic representations as well. People consciously perceive objects in the world as 3-D objects and not as projections on retinal images coupled with a depth map. We quite clearly think and reason in 3-D and we generally want our computers to do so as well because we can relate to the 3-D representation. The first step in arriving at such a representation is segmentation of the scene into symbolic entities.

This research focuses solely on symbolic modeling and recovery of parameters of the symbolic representations from multiple images. This is primarily because of the application areas of interest, in which absolute 3-D shapes and motion are the goal. For other applications, particularly navigation of mobile robots, the iconic approaches may be appropriate for much of the visual functionality.

2.2.3 Nonlinear modeling

The relationship between 3-D geometric parameters and 2-D geometric observations in the images is nonlinear. A natural approach to solving the problems, then, is to mathematically model the nonlinear system and use one of a variety of nonlinear search techniques [61, 21] to solve for the 3-D parameters from the 2-D observations.

That is, an equation of the form

$$y = h(x) \tag{2.1}$$

is set up where y contains 2-D geometric observations, x contains desired 3-D parameters, and h is a nonlinear mapping. An error function,

$$e(x, y) = h(x) - y \tag{2.2}$$

is the basis for an objective function, such as square error

$$F(x, y) = e^T(x, y)e(x, y) \tag{2.3}$$

which is minimized with respect to x to find the optimal set of 3-D parameters for the observations y.

For point geometry, a simple formulation for two images leads to an observation vector y containing (u_L, v_L, u_R, v_R) pairs of corresponding locations in the images, the state vector x contains six relative motion parameters and (X, Y, Z) triplets for each scene point, and

$$h : \mathbb{R}^{6+3N} \mapsto \mathbb{R}^{4N} \tag{2.4}$$

where N is the number of point correspondences.

In fact, the *bundle adjustment* approach of photogrammetry [8, 17, 38, 65] is an iterative search procedure based on this objective function and weighted Gauss-Newton iterations [61] and has been used for high-precision measurement of 3-D points from images since the 1950's. This iterative procedure results from deterministic or statistical least-square-error optimization criteria.

This approach is so straightfoward that, although "iterative methods" [36] were used in early computer vision research to solve the geometric equations, little specific attention was paid to the difficulties associated with the modeling or estimation process. Instead, far more energy in computer vision was spent on the problem of obtaining the correspondences from the images, which was a prerequisite to performing any 3-D estimation.

Later approaches in computer vision [14, 16, 41, 81, 72] returned to the nonlinear problem largely as a result of poor performance of "linear" techniques (discussed below) but not with the typical success exhibited by work in photogrammetry. This is partially due to the fact that vision problems contain a dynamic component, involve closely spaced images, and are expected to be a great deal more general and automatic. In photogrammetry, the images are discrete, spaced widely, and initial conditions can be set on a case-by-case basis by hand. The relatively unfavorable conditions that occur in the vision problems call for a more careful treatment of the modeling and estimation problems.

A key event in computational geometry research in computer vision was the discovery of a "linear" technique for recovery of 3-D geometry from point correspondences in two images [76, 43]. The *epipolar constraint* results in an error function

$$e^{(epi)}(x, y) = 0 \tag{2.5}$$

where x contains only the six motion parameters, y again contains the image point correspondences and

$$e^{(epi)} : \mathbb{R}^6 \times \mathbb{R}^{4N} \mapsto \mathbb{R}^N \tag{2.6}$$

contains one scalar constraint per point.

Although it is not a complete model of the geometry—it neglects the scene parameters and uses a constraint which ignores errors along epipolar lines—the attraction to this formulation is that it can be written in a linear form

$$M(y)E(x) = 0 \tag{2.7}$$

where

$$M \in \mathbb{R}^{N\times 9} \tag{2.8}$$

and

$$E \in \mathbb{R}^{9} \tag{2.9}$$

contains a set of nine parameters, called *essential parameters*, that depend only on the motion parameters, translation and rotation.

There is a scale ambiguity in E, so there are eight free parameters. Eight or more corresponding points can be used to recover E up to a scale factor, as long the 3-D points do not lie in a number of degenerate configurations. The nine parameters of E form a (3,3) matrix which can be decomposed into two matrices, one containing translation parameters and the other containing a rotation matrix [76]. Thus, motion can be recovered with linear techniques and independent of depth parameters.

The essential matrix is the basis for each row of Eqn. 2.7, which can be written

$$p_L^T E p_R \tag{2.10}$$

where p_L and p_R are image points of the form (u, v, f) where f is the focal length of the camera.

A more general form of this epipolar constraint is written

$$p_L^T F p_R \tag{2.11}$$

where (3,3) matrix F is called the *fundamental matrix* [24] and the projected points p_L and p_R are expressed as 2-D projective points $(u, v, 1)$. This formulation has led to a great deal of research in obtaining 3-D geometric parameters in a *projective* coordinate frame [25]. Again, solutions are obtained using linear techniques. Since this whole class of solutions to the point geometry problem, refered to as *linear* or *epipolar* solutions, has dominated the computer vision field in the last decade, it deserves special attention as a competing approach to nonlinear modeling.

The arguments for using linear formulations include that linear systems are easy to solve and that, since scene structure is eliminated from the state, motion can be estimated without the need for maintaining a representation of the scene. Scene structure can be obtained later, if desired, by using the recovered motion and the point corrspondences. The further argument that nonmetric projective solutions are desirable is that useful projective descriptions of objects can be obtained with linear techniques without the need even for a calibrated camera. Many robotics tasks, for instance, might be solvable without recovering metric geometry [25].

The resulting mathematics from these formulations have a lot of structure and beauty to them because they are linear and can manipulated easily. However, there are serious flaws with the entire approach for metric 3-D estimation which stems from the fact that the epipolar constraint does not capture all of the geometric information present in the available point correspondences. This causes the entire range of techniques based on this formulation to suffer from extreme sensitivity to measurement noise [81, 32], unnecessary degenerate cases [76, 67], and the inability to solve for absolute scene structure or absolute motion in a natural or accurate way.

The underlying reason for sensitivity to measurement noise is that the active constraint for a pair of corresponding points is actually valid for one of the points and the entire corresponding epipolar line in the other image. Thus noise properties in the image are not isotropic, with any noise along the epipolar line completely unpenalized in the epipolar constraint. (See Section 4.1.4 for a discussion and experimental analysis of this issue.) Thus solutions will tend to produce high residual errors along the epipolar lines, which makes subsequent scene reconstruction very poor.

The sensitivity properties are also drastically affected by the particular configuration of cameras, with complete degeneration ocurring in any configuration with zero relative translation between the two cameras. This is disasterous for motion estimation because the most trivial case of zero motion cannot even be solved! Since the sensitivity degrades for small motions, enormous numerical errors can occur in image sequence processing where interframe motions are small. Thus these techniques are very poorly suited for real-time image sequences or any motion analysis where there is not a known minimum displacement between frames.

A further problem arises in sequence processing from the fact that scene structure is discarded entirely. This problem is similar to the problem discussed in Section 2.2.2 in that every pair of frames will have a different scale factor associated with it and they cannot be rectified without resorting to a 3-D representation of the scene, which as proposed above is very poorly conditioned with these formulations. Dynamic sequence analysis has been tried [67, 51], but the systems are complicated, probably fragile, and almost certainly inaccurate.

If stereo is used, the problem with scale ambiguity disappears because the cameras can be self-calibrated and scene models can be built up as in [25]. But the numerical problems in obtaining accurate scene models persist and a large part of the reason is that, with traditional stereo systems, the self-calibration errors using epipolar constraints will be most serious in the directions which most critically affect scene reconstruction. This is a direct result of scene structure being removed completely from the objective function in the first place.

Thus these "linear" formulations pose a tradeoff between simplicity of computation coupled with algebraic beauty versus accurate 3-D scene reconstruction and absolute 3-D motion estimation. The nonlinear techniques do not have the same sensitivity problems so the only remaining reason not to use the nonlinear techniques are the questions of global convergence, which are difficult to obtain with nonlinear systems.

A large part of this thesis is understanding the state space structure of the nonlinear vision geometry problems and formulating solution techniques that are appropriate and stable. In this way, the advantages of full modeling of the geometry remain.

Other advantages of pursuing the nonlinear modeling issue is that not all 3-D geometry problems are as simple as points. Many of the same geometric issues encountered with points are found in other classes of vision geometry and thus a successful nonlinear approach can have broad applicability. An example is the blob geometry that forms the second class of problems addressed by nonlinear modeling in this thesis.

2.2.4 Dynamic modeling

The principal difference between vision and photogrammetry is that vision has a *dynamic* component whereas photogrammetry deals with collections of pictures usually with no temporal coherence. Dynamic image streams present special problems and also special opportunities for image processing and estimation.

A dynamic image stream is not an arbitrary collection of multiple pictures but a temporally ordered sequence of pictures, usually with a short interframe time interval, such as from video (.017–.040 sec) or film (.042 sec). Enforcing temporal continuity on the image stream provides a useful constraint.

This is especially necessary for "actual" vision systems that need to operate and compute in real time (as opposed to photogrammetry or other image analysis that can be processed offline and non-causally). In this case, causality is a strict constraint and all 3-D information at the current time must be based only on information at the current time instant and earlier. Even for offline systems, however, dynamic models can be useful for processing temporal image sequences. Post-processing of video and film clips, for example, is one class of applications where real-time operation is not necessary, but the imagery is nevertheless dynamic and dynamic modeling can add important constraint to the estimation. Also, long sequences are solvable in principle using batch estimation, but are solved much more efficiently recursively.

In computer vision, much work has employed the Kalman filter for this purpose. The Kalman filter is an optimal linear filtering technique which relies not only on a linear observation model but also on a linear dynamic model. The extended Kalman filter (EKF) is an extension to nonlinear systems which uses local linearization and has been the basis for a variety of dynamic estimation problems in visual geometry [1, 15, 14, 22, 26, 33, 48, 86, 5, 4, 3].

Although it has several computational and practical advantages, the requirement for causality often causes the estimation to be unstable under noisy conditions. This stability issue is the original motivation for much of the analysis in this research, because the lack of reliability exhibited by these formulations had made it difficult to use them in real vision systems.

One reaction in the theoretical domain to the perceived performance problems of EKF methods has been to resort to batch optimization based on more sophisticated iterative estimation strategies, such as Levenberg-Marquardt [81, 41, 72]. Other methods include batch minimization using epipolar constraints [68], a linear solution called the *factorization method* which applies to orthographic projection [75], and an extension of this to paraperspective projection [55]. These all have better convergence properties than the EKF but, of course, they are non-causal and they do not take advantage of the temporal coherence of the image sequences, thus they do not address the critical real-time issues.

Dynamic modeling is a foundation of this work because of the application domains of interest—film, video, and real-time interfaces. The implications of dynamic problems on modeling, particularly in the context of real-time systems, include the need for recursive processing, which in turn necessitates representations for information, memory, and learning.

For dynamic problems, image sequences are of arbitrary and unknown length. Therefore, observation data cannot be accumulated indefinitely and somehow information has to be compressed and represented in the 3-D domain if it is to be accumulated. Since current information has to be computed at every frame, estimation has to be recursive. The combination of recursive computation and compression and memory of information is the basis for recursive estimation, such as the Kalman filter.

The medium for compressing and maintaining information in the Kalman filter is probability models.

2.2.5 Probabilistic modeling

The principal difference between batch processing and recursive processing is that in batch processing all the observation data is available at once and in recursive processing only a portion is available at any time and, furthermore, previous data must be discarded at some point.

Thus there is a need in recursive processing to record all learned information in a compact form. The way that probabilistic modeling can do this is through *uncertainty modeling* [1, 70, 48, 71, 82, 66]. The current value of all relevant parameters is stored along with parameters of an uncertainty model, which is usually a Gaussian distribution centered at the value of the estimated parameters.

This is, in fact, the basis of the Kalman filter techniques. However, the KF models are only strictly valid for linear systems mapping between linear vector spaces. This thesis analyzes the problems encountered by vision geometry problems when these models cannot be used.

2.3 Person tracking

Most of the work discussed above is associated with point correspondences in multiple images. Although points are the simplest geometry and can serve as a building block for many types of problems, there are applications of 3-D geometry estimation in which point correspondences are not a useful starting point.

For example, an important application of interest for human-computer interface is visually tracking moving people. Besides the traditional problems of finding corresponding points across two images, there is the additional problem that people typically do not have a large number of fixed points that can be identified, and certainly not enough identifiable points on each rigid body segment so that 3-D estimation could take place even if points could be found.

For this reason and others, person tracking has been a formidably difficult problem in computer vision. There have been only a few serious attempts at performing this task, which are reviewed here. The blob-based method developed in this thesis solves many of the outstanding problems associated with these and has made possible the first real-time visually-based full-body human-computer interfaces that does not use special hardware or special markers.

2.3.1 Top-down approaches

O'Roarke and Badler

In 1980, O'Roarke and Badler [52] proposed a "top-down" approach to visual analysis of human motion, using a complex parameterized 3-D model of the human body and constraint propagation. The research concentrates on the 3-D modeling aspect using only synthetic imagery as input.

The computational framework requires predicted 3-D regions for the location of several features on the human body at each image, but does not treat the initialization problem. The system verifies these locations from the (synthetic) image, resulting in updated (and smaller) 3-D regions, interprets these feature movements as semantic body movements which drive a constrained internal model yielding full body position and new predictions for the next image frame.

The body model is a detailed articulated human body consisting of segments, joints, and surfaces. The rigid segments correspond to human limbs, the joint constraints include anatomical restrictions, and the surfaces, representing flesh, are modeled using overlapping spheres.

The observations are 2-D spatial locations of model features, which include head, neck, shoulders, elbows, wrists, hands, waist, hips, knees, ankles, and feet. In this paper, only synthetic images are used, however, so these features are not actually extracted from realistic imagery.

This system is a highly model-based approach which, as discussed in Section 2.2.1, can be very effective in cases where the subject of the images is known ahead of time. However, just as the processing approach is top-down, so is the motivating force of this work and consequently the important low-level vision-oriented problems of relating image features to model features is completely ignored. In the PFINDER work discussed below, many of the image features required by the O'Roarke system can be found, but perhaps not to the precision assumed by their system. Yet, there is potentially a fruitful marriage between coarse blob tracking and strong internal modeling of people, as being explored in [84].

Hogg

The work published by Hogg in 1983 [34] also uses a top-down procedure, but with real imagery. Due to the difficulty of handling real data, the application domain is much more limited, i.e. a walking person in profile.

The computational framework consists of both an initialization and a tracking stage. The initialization step uses change detection, which assumes the person is the major moving scene component. This results in a range of possible initial poses. The tracking procedure follows a hypothesize-and-verify strategy. Hypothesis poses are generated in a portion of state space that is constrained by spatio-temporal continuity and connectedness of parts in the internal model. Verification is achieved by choosing the hypothesis pose with the highest plausibility measure, which is generated for each pose by comparing image edges to predicted edges.

The model, inspired by Marr and Nishihara [45] (and earlier from Binford [11]), consists of elliptic-cylindrical body segments connected hierarchically. A PERSON consists of TORSO, HEAD, two ARMS, and two LEGS. Each ARM consists of UPPER-ARM and LOWER-ARM, which consists of FOREARM and HAND. Similarly for LEG.

The observations are occluding edges in the images, which can work as features in well constrained environments, but noisy video makes localized features such as edges, lines, points, and contours difficult to extract reliably.

Rohr

Research published later in 1994 by Rohr [59] follows a similar approach to Hogg's using the same internal model, but with various improvements. Where the Hogg experiments were performed partially interactively, the Rohr system is more automatic. The Rohr system is more sophisticated in the way it performs model verification (removing hidden surfaces) and in the way it enforces spatio-temporal continuity (with the Kalman filter).

2.3.2 Probabilistic approach

Unlike the work described above, the PFINDER person-tracking system [84] (see Fig. 2-1) was designed as a user interface system and thus bore the firm task requirement that it must operate in real-time and with people dressed in ordinary clothing undergoing general unpredictable motion. These constraints imposed that stability, speed, and generality of the scene were paramount concerns relative to accuracy or generality of function. As a result, the system is fast, stable, and can readily accommodate various users but it is limited in the configurations which it can make a correct interpretation.

The system assumes a static background so that it can first classify the person in the foreground from the rest of the scene, resulting in a silhouette region. It then uses classification over spectral and spatial dimensions within the silhouette region to track "blobs" which represent the head, hands, feet, and upper- and lower-body. The blob-tracking is continuously bootstrapped and stabilized by analysis on the contour of the silhouette.

The analysis is not completely 3-D as it represents the person as being in a vertical plane, but the blob features are the inspiration for the approach to true 3-D tracking using stereo cameras that is the topic of the research presented in Chapter 5.

Figure 2-1: The PFINDER system extracts regions using heirarchical clustering based on color classification and models these regions as blobs. The left image depicts a typical video image, the center depicts the clustered regions, and the right depicts the computed 2-D blob features.

2.3.3 Volumetric geometry

Although the blob objects used in this thesis are treated as probabilistic distributions, they could be interpreted as a simple sort of volumetric primitive since they contain the important basic geometric properties of objects—the static dimensions, and the dynamic motion including location and orientation.

There has been much work on recovering other types of volumetric models from images, including pioneering work in the 1970's on the use of generalized cones and cylinders [11, 50, 45], and work in the 1980's and 1990's on superquadrics [53, 73, 63]. These are generally based on measurement of contours or 3-D range data, where as the blob models are based directly on the more stable measurements of 2-D moments.

The principal problem with applying these in practical vision systems is that the required contour features are difficult and computationally intensive to compute. Recently, work on recovering 3-D shape and motion of a person's upper body from multiple camera viewpoints has achieved good results [28], but the multiple viewpoints were very favorable (four orthogonal views), the subjects wore carefully designed tight clothing, and the processing still had to be performed offline because of the expense.

For real systems, it is unusual that orthogonal views are available; the more common situation of two relatively close cameras is a less well-conditioned problem, indicating that the numerical estimation problems are important ones to confront. Also, to be more generally useful, systems should work with people dressed in ordinary clothing and to be applicable to human-computer interface they should work in real-time. These are some of the issues successfully addressed by the blob-based technique.

Chapter 3

Mathematical modeling and optimization

This chapter formulates a *manifold-tangent* state model in which state space is parameterized as a curved manifold and the local tangent hyperplane is a Euclidean domain. Functions on the curved domain can be expanded and linearized so that first-order (linear) differential properties of the function are on the Euclidean tangent space.

In this way, existing nonlinear estimation routines, which only apply to Euclidean domains, can be simply and naturally extended for problems with curved domains. In particular, this chapter shows how a generalized Levenberg-Marquardt strategy and a generalized extended Kalman filter arise from this modeling approach.

Further mathematical background is provided for the purposes of error analysis. Finally, several generic manifold-tangent models associated with vision-based 3-D geometry are described for subsequent use in the chapters on point and blob geometry.

3.1 Manifold-tangent state space modeling

Traditional optimization frameworks assume state space is $\mathbb{R}^n$. If the natural domain of the state is a curved manifold, special techniques are employed for *constrained optimization* by specifying additional functions which encode the constraints defining the manifold. The job of *modeling* the system consists merely of specifying the dimension of the state space and any applicable constraint equations and it is the job of the *optimization* strategy to effectively apply those constraints.

In this section, a modeling paradigm is introduced in which the structure of the state space is encoded in the model so that the issue of taking steps in state space is a property of the model and not of the particular optimization technique. In this way, both recursive and iterative optimization techniques can be developed using a single interface to the model and the critical numerical conditioning issues associated with the model can be taken care of in the modeling stage instead of being discovered in the estimation stage.

The manifold-tangent modeling paradigm consists of not just a parameter space but a structure

$$\mathcal{M} = \{\mathcal{P}_S, \mathcal{S}, \{\mathcal{P}_T(x), \mathcal{T}(x), \mathcal{Z}(x) | x \in \mathcal{S}\}, \mathcal{A}\} \tag{3.1}$$

where any particular model $\mathcal{M}^{(i)}$ contains

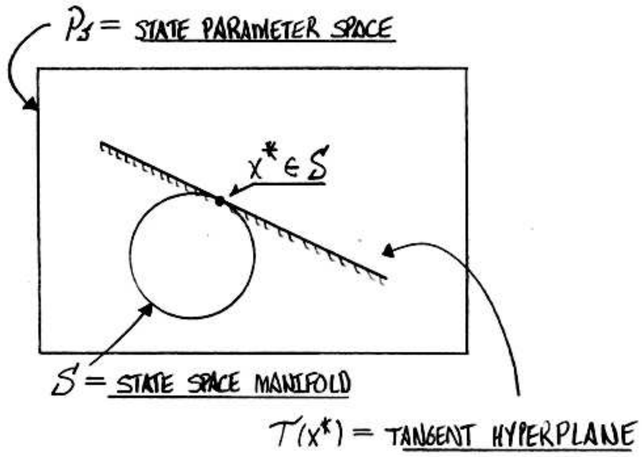


Figure 3-1: The manifold-tangent model for curved state spaces. State parameter space is a Euclidean space, i.e. $\mathcal{P}_S \equiv \mathbb{R}^m$, but the state space manifold itself is not, i.e. $\mathcal{S} \subset \mathcal{P}_S$. At each state $\hat{x} \in \mathcal{S}$ a tangent hyperplane $\mathcal{T}(\hat{x})$ provides a reduced-dimension Euclidean domain in which to perform local function linearization. A local homeomorphic transform between tangent space and state space facilitates this.

$\mathcal{M}^{(i)}$:

$$\begin{aligned}
\mathcal{P}_S^{(i)} &= \mathbb{R}^m, \quad \text{state parameter space} \\
\mathcal{S}^{(i)} \subseteq \mathcal{P}_S^{(i)} &= \{z \in \mathcal{P}_S^{(i)} | z \text{ is a physical state}\} \\
\mathcal{P}_T^{(i)}(x), x \in \mathcal{S}^{(i)} &= \mathbb{R}^n, \quad \text{tangent parameter space} \\
\mathcal{T}^{(i)}(x), x \in \mathcal{S}^{(i)} &= \{z \in \mathcal{P}_S^{(i)} | z \text{ is in } n\text{-D tangent hyperplane of } \mathcal{S}^{(i)} \text{ at } x\} \\
\mathcal{Z}^{(i)}(x), x \in \mathcal{S}^{(i)} &= \text{homeomorphism between } \mathcal{P}_T^{(i)}(x) \text{ and } \mathcal{T}^{(i)}(x) \\
\mathcal{A}^{(i)} &= \text{addition function : } \mathcal{S}^{(i)} \times \mathcal{P}_T^{(i)} \mapsto \mathcal{S}^{(i)}
\end{aligned} \tag{3.2}$$

where $\mathcal{P}_S$ is the *state parameter space*, $\mathcal{S}$ is the state space manifold, $\mathcal{T}(x)$ is the local tangent hyperplane at state x, $\mathcal{P}_T(x)$ is a tangent parameter space homeomorphic to $\mathcal{T}(x)$, $\mathcal{Z}(x)$ is the homeomorphism, and $\mathcal{A}$ is an "addition" or "composition" function which adds together a state and a tangent state to form a new state. These are discussed further below.

3.1.1 State parameter space and state space manifolds

State space is a mathematical space associated with the physical state of the world. Every point in state space should correspond to a physical state of the system being modeled. It is preferable that the relationship is also one-to-one.

For example, the physical location of an object in a scene can be described by a point $x \in \mathbb{R}^3$, by associating the three dimensions of $\mathbb{R}^3$ with three orthogonal physical directions and associating the values of the components of x with the physical distance (at some scale) along each of these directions from a fixed point.

The orientation of an object, however, cannot be so simply modeled because the set of all rotations does not form a Euclidean vector space such as $\mathbb{R}^n$. In vision geometry problems,

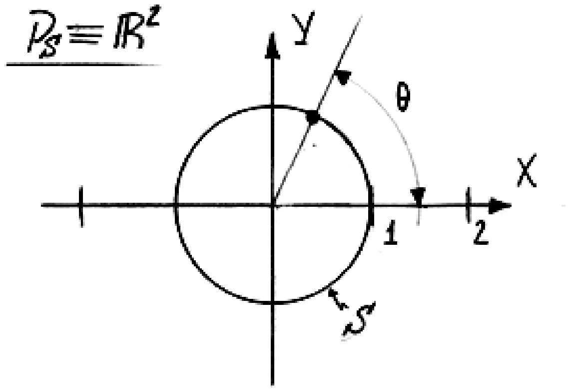


Figure 3-2: A possible state manifold for parameterizing 2-D rotation is the unit circle, $(x, y) = (\cos\theta, \sin\theta)$.

3-D rotation in particular requires special treatment and inspires the following generalized formulation of state spaces.

A state vector x is a member of *state space* $\mathcal{S}$ which is in general an n-D manifold embedded in a larger m-D *state parameter space* $\mathbb{R}^m$

$$x \in \mathcal{S} \subseteq \mathcal{P}_S \equiv \mathbb{R}^m \tag{3.3}$$

where, of course, $n \leq m$. Although state parameter space is $\mathbb{R}^m$, not all m-vectors represent *feasible* states. Only those points on the state space manifold $\mathcal{S}$ are feasible and are called "states". An m-vector which is not on the state space manifold is not physically meaningful.

Example For illustration purposes, consider 2-D rotation, which has one degree of freedom and may be parameterized in $\mathbb{R}^2$ as $(\cos(\theta), \sin(\theta))$ (see Fig. 3-2). In this case, state parameter space is $\mathbb{R}^2$ and state space $\mathcal{S}$ is the unit circle, a 1-D manifold. That is,

$$\mathcal{P}_S \equiv \mathbb{R}^2 \tag{3.4}$$

$$\mathcal{S} = \{(x, y) | x^2 + y^2 = 1\} \subset \mathcal{P}_S \tag{3.5}$$

Any 2-vector on the unit circle is a feasible state; any other 2-vector is physically meaningless. ■

3.1.2 Tangent spaces

To extend existing general methodologies of estimation, it is necessary to formulate a way of taking numeric steps to get from one state $\hat{x} \in \mathcal{S}$ to another nearby state $x \in \mathcal{S}$. In traditional estimation, steps are taken by using vector addition, but vector addition does not apply on curved manifolds.

A poor ad hoc method of extending existing methods is to take a step using vector addition and then apply the manifold constraint to project back onto state space. A better idea, taken from constrained optimization, is to take a step in the tangent hyperplane and then project onto state space. The state space model $\mathcal{M}$ includes a set (bundle) of *tangent* *hyperplanes*, one for each state,

$$\{\mathcal{T}(x) | x \in \mathcal{S}\} \tag{3.6}$$

as part of the model specification.

The tangent hyperplane

$$\mathcal{T}(x) \subseteq \mathcal{P}_S \tag{3.7}$$

defined at $x \in \mathcal{S}$ is a tangent hyperplane to $\mathcal{S}$ through x and is homeomorphic to $\mathbb{R}^n$. Equality holds only in the case when the whole parameter space is state space.

A basis for $\mathcal{T}$ in state parameter space, $\mathbb{R}^m$ will consist of n mutually orthogonal unit m-vectors which are each tangent to $\mathcal{S}$. If $\mathcal{S}$ is a parameterized manifold defined by a vector-valued constraint equation, for example,

$$c(x) = 0 \in \mathbb{R}^{m-n} \tag{3.8}$$

then any full rank matrix $Z \in \mathbb{R}^{m \times n}$ of orthogonal column (unit) vectors that satisfy

$$J_c(x) Z = 0 \in \mathbb{R}^{(m-n) \times n} \tag{3.9}$$

can serve as a basis set of $\mathcal{T}(x)$, where

$$J_c(x) = \left. \frac{\partial c(\xi)}{\partial \xi} \right|_{\xi = x} \tag{3.10}$$

is the Jacobian matrix of partial derivatives of $c(x)$ at x.

This matrix Z can define the homeomorphism

$$\mathcal{P}_T(x) \equiv \mathbb{R}^n \rightleftharpoons \mathcal{T}(x) \tag{3.11}$$

between tangent parameter space $\mathcal{P}_T(x)$ and the tangent hyperplane $\mathcal{T}(x)$. However, in general, any homeomorphic function

$$\mathbb{R}^n \quad \overset{\mathcal{Z}}{\rightleftharpoons} \quad \mathcal{T}(x) \tag{3.12}$$

can be used and thus for generality the model contains a set of homeomorphic functions, one for each state,

$$\{\mathcal{Z}(x) | x \in \mathcal{S}\} \tag{3.13}$$

such that

$$\mathcal{P}_T(x) \quad \overset{\mathcal{Z}(x)}{\rightleftharpoons} \quad \mathcal{T}(x) \tag{3.14}$$

holds, where $\mathcal{P}_T(x) \equiv \mathbb{R}^n$ is a linear vector space.

Then a step $\delta x \in \mathcal{P}_T(x)$ can be taken in the tangent parameter space and the equivalence

$$z = x + \mathcal{Z}(x)(\delta x) \tag{3.15}$$

holds where $z \in \mathcal{T}(x) \subset \mathcal{P}_S$ and $\delta x \in \mathcal{P}_T(x)$.

Example For example, in the 2-D rotation parameterization of Fig. 3-2, the tangent space $\mathcal{T}(x)$ at any state x on the unit circle is the line through x orthogonal to the vector from the origin to x, producing a 1-D tangent space $\mathcal{T}(x)$ parameterized by the scalar δx. Thus,

$$\mathcal{P}_S \equiv \mathbb{R}^2 \tag{3.16}$$

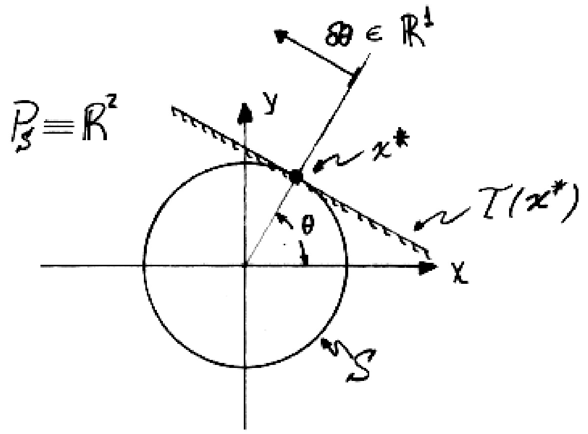


Figure 3-3: The tangent space of a 2-D rotation manifold.

$$\mathcal{S} = \{x \in \mathrm{I\!R}^2 | x_1^2 + x_2^2 = 1\} \subset \mathcal{P}_S \tag{3.17}$$

$$\mathcal{T}(\hat{x}) = \{x \in \mathrm{I\!R}^2 | \langle x, \hat{x} \rangle = 1\} \subset \mathcal{P}_S \tag{3.18}$$

and

$$\mathcal{P}_T \equiv \mathrm{I\!R}^1 \tag{3.19}$$

is the tangent parameter space.

The homeomorphism

$$\delta x \in \mathrm{I\!R} \quad \overset{\mathcal{Z}(x)}{\mapsto} \quad \delta x \begin{pmatrix} x_2 \\ -x_1 \end{pmatrix} \tag{3.20}$$

specifies a step along the tangent line that defines a move from x to

$$z = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \delta x \begin{pmatrix} x_2 \\ -x_1 \end{pmatrix} \tag{3.21}$$

which is a point on the tangent line a distance δx away from x. ■

3.1.3 Composition function

To complete the model interface, we add a projection function after the tangent step to create a state composition function

$$\mathcal{A} : \mathcal{S} \times \mathcal{P}_T \mapsto \mathcal{S} \tag{3.22}$$

which parallels vector addition for taking steps in linear vector spaces.

The tangent step is taken in a local tangent hyperplane and the resulting tangent point is projected onto state space. That is,

$$\mathcal{A}(x, \delta x) = P(x + \mathcal{Z}(x)(\delta x), x) \tag{3.23}$$

where the projection function

$$P(z(x, \delta x), x) \tag{3.24}$$

$$P : \mathcal{T}(x) \times \mathcal{S} \mapsto \mathcal{S} \tag{3.25}$$

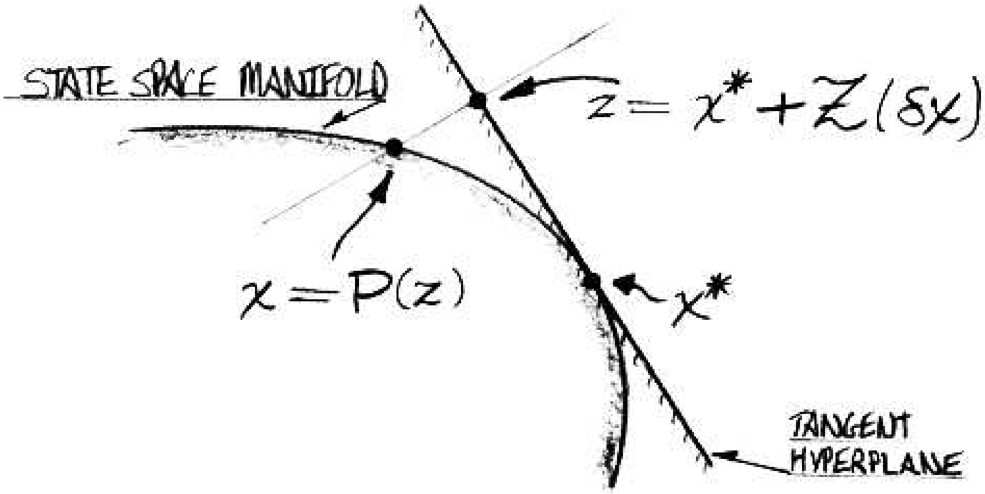


Figure 3-4: The composition function combines a local "tangent basis" homeomorphism $\mathcal{Z}(x) : \mathcal{P}_T \mapsto \mathcal{T}(x)$ with a local "projection" homeomorphism $P(x) : \mathcal{T}(x) \mapsto \mathcal{S}$ to create an invertible "addition" function $\mathcal{A} : \mathcal{S} \times \mathcal{P}_T \mapsto \mathcal{S}$.

projects z onto $\mathcal{S}$ along a perpendicular to the tangent hyperplane $\mathcal{T}(x)$.

The difference function

$$\mathcal{A}^{-1} : \mathcal{S} \times \mathcal{S} \mapsto \mathcal{P}_T \tag{3.26}$$

is an inverse of the composition function and maps two states to a tangent step relative to one of the two states.

Specifically,

$$\mathcal{A}^{-1}(x, \hat{x}) = \mathcal{Z}^T(\hat{x})(P^{-1}(x; \hat{x}) - \hat{x}) \tag{3.27}$$

consists of projecting $x \in \mathcal{S}$ onto the tangent hyperplane $\mathcal{T}(\hat{x})$, subtracting from $\hat{x}$ and putting through the homeomorphic mapping to get a result in $\mathcal{P}_T$.

These properties of $\mathcal{A}$

$$\begin{aligned}
\mathcal{A}(x, 0) &= x && (3.28)\\
\mathcal{A}^{-1}(x, x) &= 0 && (3.29)\\
\mathcal{A}^{-1}(\mathcal{A}(x, \delta x), x) &= \delta x && (3.30)\\
\mathcal{A}^{-1}(x, \mathcal{A}(x, \delta x)) &= -\delta x && (3.31)\\
\mathcal{A}(\hat{x}, \mathcal{A}^{-1}(x, \hat{x})) &= x && (3.32)
\end{aligned}$$

follow directly from the definitions.

It will be necessary for various reasons to linearize these two functions. To this end the following Taylor series expansions apply:

$$\mathcal{A}(x, \delta x) = x + \left.\frac{\partial \mathcal{A}(\xi, \delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = x \\ \delta\xi = 0}} \delta x + \cdots \tag{3.33}$$

and

$$\mathcal{A}^{-1}(x,\hat{x}) = 0 + \left.\frac{\partial \mathcal{A}^{-1}(\xi_1,\xi_2)}{\partial \xi_1}\right|_{\substack{\xi_1 = \hat{x} \\ \xi_2 = \hat{x}}} (x - \hat{x}) + \cdots \tag{3.34}$$

which are obtained straightforwardly with use of the properties above. The abbreviations

$$\frac{\partial \mathcal{A}}{\partial \delta x} = \left.\frac{\partial \mathcal{A}(\xi,\delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = x \\ \delta\xi = 0}} \in \mathbb{R}^{m \times n} \tag{3.35}$$

$$\frac{\partial \mathcal{A}^{-1}}{\partial x} = \left.\frac{\partial \mathcal{A}^{-1}(\xi_1,\xi_2)}{\partial \xi_1}\right|_{\substack{\xi_1 = \hat{x} \\ \xi_2 = \hat{x}}} \in \mathbb{R}^{n \times m} \tag{3.36}$$

are used for brevity when the context is clear.

Some other useful properties include

$$\left.\frac{\partial \mathcal{A}^{-1}(\xi,\zeta)}{\partial \xi}\right|_{\substack{\xi = x \\ \zeta = \hat{x}}} \left.\frac{\partial \mathcal{A}(\xi,\delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = x \\ \delta\xi = 0}} = \frac{\partial \delta x}{\partial \delta x} = I_{n,n} \tag{3.37}$$

$$\left.\frac{\partial \mathcal{A}^{-1}(\xi,\zeta)}{\partial \zeta}\right|_{\substack{\xi = x \\ \zeta = \hat{x}}} \left.\frac{\partial \mathcal{A}(\xi,\delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = \hat{x} \\ \delta\xi = 0}} = -\frac{\partial \delta x}{\partial \delta x} = -I_{n,n} \tag{3.38}$$

which arise from Eqn. 3.30 and Eqn. 3.31. These relationships, since they produce identity matrices, are pseudoinverse relationships (because in general the matrices are not square).

An example of a nontrivial state composition function is developed in Section 3.7.2 for modeling 3-D rotation.

3.1.4 Euclidean state models

The state model developed above has gone to great extent to generalize the concept of state space beyond a simple Euclidean vector space. A Euclidean vector space is, however, a trivial case of this model in which state space is equivalent to the state parameter space

$$\mathcal{P}_S \equiv \mathcal{S} \equiv \mathbb{R}^m \tag{3.39}$$

or, in other words, every element of state parameter space $\mathcal{P}_S$ corresponds to a physically meaningful state of the system.

The tangent space is coincident with the state space

$$\mathcal{P}_T(x) \equiv \mathcal{T}(x) \equiv \mathbb{R}^n \equiv \mathbb{R}^m, \forall x \in \mathcal{S} \tag{3.40}$$

and the homeomorphism

$$\mathcal{Z}(x) \equiv I, \forall x \in \mathcal{S} \tag{3.41}$$

is therefore trivial.

The state composition function

$$\mathcal{A}(x,\delta x) \doteq x + \delta x \tag{3.42}$$

where

$$x, \delta x \in \mathbb{R}^n \tag{3.43}$$

reverts to Cartesian vector addition and the difference function

$$\mathcal{A}^{-1}(x, \hat{x}) \doteq x - \hat{x} \tag{3.44}$$

is simply Cartesian vector subtraction.

Whenever a state model consists of n independent parameters, it can be modeled using a standard Euclidean model. For convenience, the notation

$$\mathcal{M} \equiv \mathcal{M}^n_{Euc} \tag{3.45}$$

will indicate a n-D Euclidean state model.

3.1.5 Composite and component state models

Often, parts of the state space can be modeled separately and independently. It is straightfoward to combine a set of N component state models $\mathcal{M}^{(i)}, i = 1 \ldots N$ into a *composite state model*, expressed as

$$\mathcal{M} = \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(N)} \tag{3.46}$$

where

$$\mathcal{P}_S = \mathcal{P}_S^{(1)} \times \cdots \times \mathcal{P}_S^{(N)} \tag{3.47}$$
$$\mathcal{S} = \mathcal{S}^{(1)} \cup \cdots \cup \mathcal{S}^{(N)} \tag{3.48}$$
$$\mathcal{P}_T = \mathcal{P}_T^{(1)} \times \cdots \times \mathcal{P}_T^{(N)} \tag{3.49}$$
$$\mathcal{T}(x) = \mathcal{T}^{(1)}(x^{(1)}) \cup \cdots \cup \mathcal{T}^{(N)}(x^{(N)}) \tag{3.50}$$

prescribes that the composite parameter space is the Cartesian product of the component parameter spaces, the composite tangent parameter space is the Cartesian product of the component tangent parameter spaces, the composite state space set is the union of the component state space sets, and the composite tangent hyperplane set is the union of the component tangent hyperplane sets.

The component $\mathcal{Z}^{(i)}$ and $\mathcal{A}^{(i)}$ functions can be combined straightfowardly to form composite $\mathcal{Z}$ and $\mathcal{A}$ functions.

3.2 Functions having manifold-tangent domains

The intent with manifold-tangent models is to model the state on a manifold and first-order perturbations on the tangent hyperplane. To this end, we would like to define real functions on manifold-tangent spaces

$$\mathcal{M} \mapsto \mathcal{E} \tag{3.51}$$

where $\mathcal{E} \in \mathbb{R}^p$ such that if $\mathcal{M} = \{\mathcal{S}, \mathcal{P}_T(x), \ldots\}$ then there is a forward function

$$g : \mathcal{S} \mapsto \mathcal{E} \tag{3.52}$$

mapping from the state space manifold, but the first-order differential properties at some reference state $\hat{x}$, encoded in a Jacobian matrix,

$$J(\hat{x}) : \mathcal{P}_T(\hat{x}) \mapsto \mathcal{E} \tag{3.53}$$

map from the local tangent space. This can be accomplished through a straightforward compound linearization including g and the manifold-tangent composition function $\mathcal{A}$.

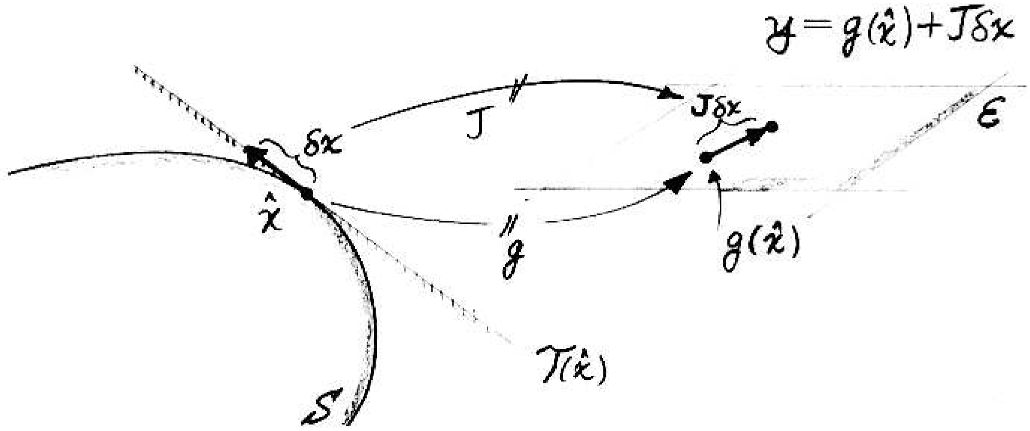


Figure 3-5: The manifold-tangent state model allows functions to have a $\mathcal{S}$ domain and a linearization that has a $\mathcal{P}_T$ domain. Here, a function $g : \mathcal{S} \mapsto \mathcal{E}$ has a manifold domain but its Jacobian $J : \mathcal{P}_T(\hat{x}) \mapsto \mathcal{E}$ has a tangent domain.

3.2.1 Linearization

If a function

$$g : \mathcal{S} \mapsto \mathcal{E} \tag{3.54}$$

maps a curved state space $\mathcal{S}$ onto a Euclidean space $\mathcal{E}$, then a Taylor series expansion around a reference state $\hat{x}$

$$\begin{aligned} g(x) &= g(\mathcal{A}(\hat{x}, \delta x)) && (3.55)\\ &= g(\underbrace{\mathcal{A}(\hat{x}, 0)}_{\hat{x}} + \frac{\partial \mathcal{A}}{\partial \delta x}\mathcal{A}^{-1}(x, \hat{x}) + \cdots) && (3.56)\\ &= g(\hat{x}) + \underbrace{\frac{\partial g}{\partial x}\frac{\partial \mathcal{A}}{\partial \delta x}}_{J}\underbrace{\mathcal{A}^{-1}(x, \hat{x})}_{\delta x} + \cdots && (3.57) \end{aligned}$$

can be set up involving the tangent parameters δx and leads to the linearized system

$$\delta g \approx J(\hat{x})\delta x \tag{3.58}$$

where

$$\begin{aligned} \delta g &\doteq g(x) - g(\hat{x}) && (3.59)\\ \delta x &\doteq \mathcal{A}^{-1}(x, \hat{x}) \in \mathcal{P}_T(\hat{x}) && (3.60)\\ J(\hat{x}) &\doteq \left.\frac{\partial g(\xi)}{\partial \xi}\right|_{\xi = \hat{x}} \left.\frac{\partial \mathcal{A}(\xi, \delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = \hat{x}\\ \delta\xi = 0}} && (3.61) \end{aligned}$$

hold by definition.

3.2.2 Dual domains

We will be concerned with dual-domain functions of the form

$$g : \mathcal{S}^{(1)} \times \mathcal{S}^{(2)} \mapsto \mathcal{E} \tag{3.62}$$

in which case a similar expansion of $g(x, y)$ results in the linearized system

$$\delta g \approx J_x(\hat{x}, \hat{y})\delta x + J_y(\hat{x}, \hat{y})\delta y \tag{3.63}$$

where

$$J_x(\hat{x}, \hat{y}) \doteq \left.\frac{\partial g(\xi, \zeta)}{\partial \xi}\right|_{\substack{\xi = \hat{x} \\ \zeta = \hat{y}}} \left.\frac{\partial \mathcal{A}(\xi, \delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = \hat{x} \\ \delta\xi = 0}} \tag{3.64}$$

$$J_y(\hat{x}, \hat{y}) \doteq \left.\frac{\partial g(\xi, \zeta)}{\partial \zeta}\right|_{\substack{\xi = \hat{x} \\ \zeta = \hat{y}}} \left.\frac{\partial \mathcal{A}(\zeta, \delta\zeta)}{\partial \delta\zeta}\right|_{\substack{\zeta = \hat{y} \\ \delta\zeta = 0}} \tag{3.65}$$

hold by definition.

3.3 Batch optimization for manifold-tangent state models

With these tools, we can now state a general least-squares optimization algorithm in terms of manifold-tangent state models and show how all major batch optimization algorithms on continuous linear and nonlinear systems relate to it. The discussion of nonlinear algorithms leads to a generalized formulation for manifold-tangent models of the Levenberg-Marquardt strategy, which will be used in Chapter 5 for the static problem of self-calibration from a batch of blob correspondences. For completeness, the illuminating but somewhat peripheral discussion of linear algorithms can be found in Appendix B.

3.3.1 General manifold-tangent least squares

The following is a general iterative strategy for least-squares optimization expressed in terms of a manifold-tangent state model.

Algorithm 1 (General Manifold-tangent Least Squares) *Given a manifold-tangent state model $\mathcal{M}$, an observation space $\mathcal{O} \equiv I\!R^q$, and an implicit error function $e(x, y) = 0$ where*

$$e : \mathcal{S} \times \mathcal{O} \mapsto \mathcal{E} \equiv I\!R^p$$

the least squares objective function is

$$F(x, y) = e^T(x, y)R^{-1}e(x, y)$$

where $x \in \mathcal{S}$, $y \in \mathcal{O}$ and $R \in I\!R^{p\times p}$ is a positive definite weighting matrix. A general strategy for finding the solution $\hat{x}$ that minimizes $F(x, y)$ is

input x_0, y
for $k = 0, 1, \ldots$ repeat
  expand and linearize
    $e(x, y) = e(x_k, y) + J_k \delta x + \cdots = 0$
    $\delta e \doteq -e(x_k, y) \approx J_k \delta x, \delta x \doteq \mathcal{A}^{-1}(x, x_k)$
  for $i = 0, 1, \ldots$ repeat
    choose P_i^{-1}
    solve
      $\hat{\delta x_i} = (J_k^T R^{-1} J_k + P_i^{-1})^{-1} J_k^T R^{-1} \delta e$
    move
      $\hat{x}_i = \mathcal{A}(x_k, \hat{\delta x_i})$
  end until $F(\hat{x}_i, y) < F(x_k, y)$
  update $x_{k+1} = \hat{x}_i$
end until $\| J_k^T R^{-1} \delta e \| <$ tolerance

where

$$J_k \doteq \left. \frac{\partial e(\xi, \zeta)}{\partial \delta \xi} \right|_{\substack{\xi = x_k \\ \zeta = y}}$$

and P^{-1} is a positive semi-definite matrix.

■

The "general" least squares strategy is so-called because it contains the classic nonlinear estimators as special cases when the state space is chosen to be Euclidean and P^{-1} and R^{-1} are chosen appropriately. The strategy also contains all the classic optimal linear estimators when in addition to Euclidean spaces a linear constraint is used.

The special cases of linear estimators are reviewed in Appendix B. The special cases of nonlinear estimators are reviewed here to motivate the generalization of Levenberg-Marquardt.

3.3.2 Euclidean nonlinear systems

For Euclidean nonlinear systems, Alg. 1 can be simplified in that the step δx becomes simply $x - x_k$, the Jacobian becomes simply $\partial e / \partial x$, and the move becomes simply vector addition. These simplifications lead to a general approach, outlined in Alg. 2 below in which various interpretations of the weighting matrices lead to traditional nonlinear optimization techniques including Newton's method, the Gauss-Newton method, and various so-called *modified Newton's methods* such as the Levenberg-Marquardt method.

The nonlinear techniques look a lot like linear techniques (see Appendix B) in the inner loop, but are extended to allow iterative re-linearization in the outer loop. The manifold-tangent algorithm can then be seen as a further extension following the same general approach but taking steps through the homeomorphic tangent space rather than through state parameter space.

The following algorithm specializes Alg. 1 to Euclidean domains and the subsequent sections explain how choices of weighting matrices result in the traditional nonlinear optimization techniques. Arrows indicate where Alg. 2 specializes from Alg. 1.

Algorithm 2 (General Euclidean Nonlinear Least Squares) *Given a Euclidean state space $\mathcal{S} \equiv I\!R^n$, an observation space $\mathcal{O} \equiv I\!R^q$, and an implicit error function $e(x, y) = 0$ where*

$$e : \mathcal{S} \times \mathcal{O} \mapsto \mathcal{E} \equiv I\!R^p$$

the least squares objective function is

$$F(x, y) = e^T(x, y)R^{-1}e(x, y)$$

where $x \in \mathcal{S}$, $y \in \mathcal{O}$ and $R \in I\!R^{p \times p}$ is a positive definite weighting matrix. A general strategy for finding the solution $\hat{x}$ that minimizes $F(x, y)$ is

input x_0
for $k = 0, 1, \ldots$ repeat
  expand and linearize
    $e(x, y) = e(x_k, y) + J_k \delta x + \cdots = 0$
$\Rightarrow$ $\delta e \doteq -e(x_k, y) \approx J_k \delta x$, $\delta x \doteq x - x_k$
  for $i = 0, 1, \ldots$ repeat
    choose P_i^{-1}
    solve
      $\widehat{\delta x}_i = (J_k^T R^{-1} J_k + P_i^{-1})^{-1} J_k^T R^{-1} \delta e$
    move
$\Rightarrow$ $\hat{x}_i = x_k + \widehat{\delta x}_i$
  end until $F(\hat{x}_i, y) < F(x_k, y)$
  update $x_{k+1} = \hat{x}_i$
end until $\| J_k^T R^{-1} \delta e \| <$ tolerance

where

$$\Rightarrow \qquad J_k \doteq \left. \frac{\partial e(\xi, \zeta)}{\partial \xi} \right|_{\substack{\xi = x_k \\ \zeta = y}}$$

and P^{-1} is a positive semi-definite matrix.

■

Alg. 2 contains several classic optimization algorithms as special cases.

Newton's method

In the most basic *Newton's method*, with a weighted least-squares objective function, P^{-1} stands for the second derivative terms of the Taylor series expansion of $e(x, y)$, i.e. $P^{-1} = S$ where

$$S = \sum_{i=1}^{p} (R^{-1}e)_i \underbrace{\left[\frac{\partial}{\partial x} \frac{\partial e_i}{\partial x} \right]}_{\text{Hessian}} \tag{3.66}$$

where $R^{-1}e$ represents the weighted error vector.

Newton's method is easily derived by considering a Taylor's series expansion on the gradient of the objective function $F(x, y)$

$$\frac{\partial F}{\partial x}(x, y) = \frac{\partial F}{\partial x}(\hat{x}, y) + \underbrace{\left. \frac{\partial}{\partial x} \frac{\partial F}{\partial x} \right|_{x = \hat{x}}}_{\text{Hessian matrix}} (x - \hat{x}) + \cdots \tag{3.67}$$

$$g(x) = g(\hat{x}) + H(\hat{x}) \delta x \tag{3.68}$$

where the gradient vector of F at x has been abbreviated $g(x)$, the Hessian matrix of F at x has been abbreviated $H(x)$, and the assumption of Newton's method that F is quadratic in form near the solution has eliminated the higher-order terms.

The minimum of F requires that $g(x) = 0$ leading to

$$H(\hat{x})\delta x = -g(\hat{x}) \tag{3.69}$$

or

$$\delta x = -H^{-1}(\hat{x})g(\hat{x}) \tag{3.70}$$

after performing the inverse. (Note that a singular Hessian means trouble.)

With a least squares error function and its derivatives

$$\begin{aligned}
F(x,y) &= e^T(x,y)R^{-1}e(x,y) \qquad \text{(scalar objective)} && (3.71)\\
\frac{\partial F}{\partial x}(x,y) &= 2J^T(x,y)R^{-1}e(x,y) \qquad \text{(gradient vector)} && (3.72)\\
\frac{\partial}{\partial x}\frac{\partial F}{\partial x}(x,y) &= 2J^T(x,y)R^{-1}J(x,y) + 2S(x,y) \qquad \text{(Hessian matrix)} && (3.73)\\
& && (3.74)
\end{aligned}$$

where S is as in Eqn. 3.66, it is easy to recover the solution step

$$\begin{aligned}
\delta x &= (J^T(\hat{x})R^{-1}J(\hat{x}) + S(\hat{x}))^{-1}J^T(\hat{x})R^{-1}(-e(\hat{x})) && (3.75)\\
&= (J^T R^{-1} J + S)^{-1} J^T R^{-1} \delta e && (3.76)
\end{aligned}$$

of Alg. 2.

Since S is usually expensive to evaluate and can fail to be postive definite, other methods ignore or replace it with a useful approximation that is positive definite.

The Gauss-Newton method

For instance, the *Gauss-Newton* method ignores S completely, using $P^{-1} = 0$ resulting in

$$\hat{\delta x} = (J^T J)^{-1} J^T \delta y \tag{3.77}$$

where R^{-1} has been chosen to be scaled identity. This is the Moore-Penrose pseudoinverse (see Section B.5). It can be singular, however, and is therefore not generally reliable.

Modified Newton's methods

Since the matrix $J^T J$ can be singular and non-invertible, it is preferable to replace S with something other than zero. The class of *modified Newton's methods* [61] look like Newton's method, but replace S with a positive definite P^{-1} to ensure invertibility.

For example, the *Levenberg-Marquardt* solution uses $P^{-1} = \mu_k I$ where μ_k is a real scalar that is modulated up and down according to progress being made in reducing the error. When μ_k is large, the iteration step becomes a small step in the steepest descent direction. When μ_k is zero, the iteration is the same as Gauss-Newton.

The Levenberg-Marquardt strategy is simple and quite effective in practice and thus it is chosen as the basis for a batch algorithm for manifold-tangent domains.

3.3.3 A generalized Levenberg-Marquardt algorithm

Following is the basic Levenberg-Marquardt optimization strategy [61] generalized to manifold-tangent state models. The line numbering is in reference to Alg. 1.

Algorithm 3 (Generalized Levenberg-Marquardt) *Given a manifold-tangent state model $\mathcal{M}$, an observation space $\mathcal{O} \equiv I\!R^q$, and an implicit error function $e(x,y) = 0$ where*

$$e : \mathcal{S} \times \mathcal{O} \mapsto \mathcal{E} \equiv I\!R^p$$

the least squares objective function is

$$F(x,y) = e^T(x,y)R^{-1}e(x,y)$$

where $x \in \mathcal{S}$, $y \in \mathcal{O}$ and $R \in I\!R^{p \times p}$ is a positive definite diagonal weighting matrix. A strategy for finding the solution $\hat{x}$ that minimizes $F(x,y)$ is

1 input x_0, y
  choose $\mu_0, \nu = 10$
2 for $k = 0, 1, \ldots$ repeat
3   expand and linearize
4     $e(x,y) = e(x_k, y) + J_k \delta x + \cdots = 0$
5     $\delta e \doteq -e(x_k, y) \approx J_k \delta x, \delta x \doteq \mathcal{A}^{-1}(x, x_k)$
6   for $i = 0, 1, \ldots$ repeat
7   choose $P_i^{-1} = \mu_k I$
8   solve
9       $\hat{\delta x}_i = (J_k^T R^{-1} J_k + P_i^{-1})^{-1} J_k^T R^{-1} \delta e$
10   move
11       $\hat{x}_i = \mathcal{A}(x_k, \hat{\delta x}_i)$
    if $F(\hat{x}_i, y) \geq F(x_k, y)$ then
      set $\mu_k = \mu_k \nu$
    end
12   end until $F(\hat{x}_i, y) < F(x_k, y)$
13   update $x_{k+1} = \hat{x}_i$
  set $\mu_{k+1} = \mu_k / \nu$
14 end until $\| J_k^T R^{-1} \delta e \| <$ tolerance

where

$$J_k \doteq \left. \frac{\partial e(\xi, \zeta)}{\partial \delta \xi} \right|_{\substack{\xi = x_k \\ \zeta = y}}$$

is the Jacobian.

■

When Euclidean assumptions are placed on the model, the algorithm is the classic Levenberg-Marquardt algorithm [61, 56]. The major difference is the use of a tangent step for updating the state estimate at each iteration. Without the manifold-tangent state model, either an inefficient ad hoc technique would have to be used in conjunction with Levenberg-Marquardt or one would have to resort to some kind of constrained optimization technique.

3.4 Probabilistic error modeling in manifold-tangent domains

The constraint model contains an idealized physical model of the process, but in order to obtain a numerically stable solution and to weight various parameters in a physically meaningful way, it is often necessary to model the process errors, the deviation from ideality.

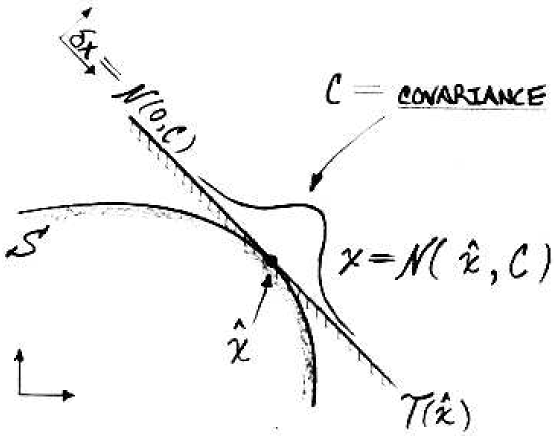


Figure 3-6: Parameter uncertainty modeling in manifold-tangent domains can be accomplished by a zero-mean distribution in local tangent space.

Traditionally, for Euclidean domains, uncertainty models are based on Gaussian distributions in state space (cf. Section B.2 and Section B.4). For curved domains, Gaussians cannot be applied directly, but they can be applied in the local tangent space, leading to a way of using manifold-tangent models to extend recursive probabilistic techniques to vision problems.

For a manifold-tangent model, we can define the notation

$$x = \mathcal{N}(\hat{x}, P) \tag{3.78}$$

to specify a manifold random variable

$$x = \mathcal{A}(\hat{x}, \delta x) \tag{3.79}$$

described by the normal probability distribution on Euclidean random variable

$$\delta x = \mathcal{A}^{-1}(x, \hat{x}) \tag{3.80}$$

where

$$p(\delta x) = \frac{1}{(2\pi)^{n/2}|P|^{1/2}} \exp\left(-\frac{1}{2}\delta x^T P^{-1} \delta x\right) \tag{3.81}$$

is a zero-mean Gaussian distribution in $\mathcal{P}_T(\hat{x})$.

Using this model for local probability distributions, we can describe local uncertainty in state space parameters. In conjunction with the function linearizations of Section 3.2, this leads to a means for generalizing the extended Kalman filter strategy to one which applies to recursive estimation on a curved state space.

3.5 Recursive optimization for manifold-tangent state models

The Kalman filter is based on an observation model (external constraint), a dynamic model (internal constraint), and probability distributions for characterizing modeling error and measurement error as random variables. Here the Kalman filter framework is generalized in two principal directions.

First, the state space is treated as a manifold-tangent model and, second, implicit relationships rather than forward models are used for the internal and external constraints.

The same implicit error function as used in batch optimization is used here

$$e(x(t), y(t)) + v(t) = 0 \tag{3.82}$$

but the arguments are now dynamic and the extra term $v(t)$ is a zero-mean random variable expressing modeling error in the external constraint function.

In addition, dynamic constraint is expressed as an internal constraint implicit function

$$i(x(t), x(t+1)) + w(t) = 0 \tag{3.83}$$

where $w(t)$ is a random variable expressing modeling error in the internal constraint function.

3.5.1 Internal constraint and state transition function

Internal constraint implicit function

For dynamic problems, one way of modeling temporal constraints in discrete time is through constraints imposed by temporally neighboring states. In general, the implicit function

$$i(x(t), x(t+1)) + w(t) = 0 \tag{3.84}$$

where

$$i : \mathcal{S} \times \mathcal{S} \mapsto \mathcal{I} \equiv \mathrm{I\!R}^n \tag{3.85}$$

can impose a constraint between $x(t)$ and $x(t+1)$.

Existing versions of the Kalman filter, which use Euclidean domains, model this constraint using a state transition function

$$f : \mathcal{S} \mapsto \mathcal{S} \tag{3.86}$$

such that

$$x(t+1) = f(x(t)) + w(t) \tag{3.87}$$

where $x(t), x(t+1), w(t) \in \mathcal{S}$, leading to

$$i(x(t), x(t+1)) \doteq f(x(t)) - x(t+1) \tag{3.88}$$

as an equivalent implicit function.

Although all the problems treated in this thesis are modeled using simple state transition functions, the implicit function formulation is presented to maintain generality.

For the general case, the linearization is obtained from the Taylor series expansion about two adjacent reference states $\hat{x}(t)$ and $\hat{x}(t+1)$ and proceeds as in Eqn. 3.63

$$\begin{aligned}
i(x(t), x(t+1)) + w(t) = & \qquad (3.89)\\
& \left(i(\hat{x}(t), \hat{x}(t+1) + \frac{\partial i(\xi,\zeta)}{\partial \xi} \frac{\partial \mathcal{A}}{\partial \delta x} \mathcal{A}^{-1}(x(t), \hat{x}(t)) \right. \qquad (3.90)\\
& \left. + \frac{\partial i(\xi,\zeta)}{\partial \zeta} \frac{\partial \mathcal{A}}{\partial \delta x} \mathcal{A}^{-1}(x(t+1), \hat{x}(t+1)) + \cdots \right) + w(t) \qquad (3.91)\\
= \quad & 0 \qquad (3.92)
\end{aligned}$$

leading to a linearized system

$$\delta i \approx J_t \delta x(t) + J_{t+1} \delta x(t+1) + w(t) \tag{3.93}$$

where

$$\begin{aligned}
\delta i &\doteq -i(\hat{x}(t), \hat{x}(t+1)) && (3.94)\\
\delta x(\tau) &\doteq \mathcal{A}^{-1}(x(\tau), \hat{x}(\tau)) && (3.95)\\
J_t &\doteq \left.\frac{\partial i(\xi,\zeta)}{\partial \xi}\right|_{\substack{\xi = \hat{x}(t)\\ \zeta = \hat{x}(t+1)}} \left.\frac{\partial \mathcal{A}(\xi,\delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = \hat{x}(t)\\ \delta\xi = 0}} && (3.96)\\
J_{t+1} &\doteq \left.\frac{\partial i(\xi,\zeta)}{\partial \zeta}\right|_{\substack{\xi = \hat{x}(t)\\ \zeta = \hat{x}(t+1)}} \left.\frac{\partial \mathcal{A}(\xi,\delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = \hat{x}(t+1)\\ \delta\xi = 0}} && (3.97)
\end{aligned}$$

as in Eqn. 3.64 and Eqn. 3.65.

Error propagation

Part of the Kalman filter is to propagate error parameters forward in time using the internal dynamic constraint. For the implicit constraint equation this means computing

$$x(t+1) = \mathcal{N}(\hat{x}(t+1), P(t+1)) \tag{3.98}$$

given

$$x(t) = \mathcal{N}(\hat{x}(t), P(t)) \tag{3.99}$$

and the implicit constraint relationship $i(x(t), x(t+1))$.

This can be done to first-order using the linearization of Eqn. 3.93. We can rearrange to obtain

$$-J_{t+1}\delta x(t+1) = -\delta i + J_t \delta x(t) + w(t) \tag{3.100}$$

where

$$\begin{aligned}
\delta x(t) &= \mathcal{N}(0, P(t)) && (3.101)\\
w(t) &= \mathcal{N}(0, Q(t)) && (3.102)
\end{aligned}$$

are the initial error and noise statistics.

This leads to

$$\delta x(t+1) = J_{t+1}^{\dagger}\delta i + (-J_{t+1}^{\dagger})(J_t \delta x(t) + w(t)) \tag{3.103}$$

where $J_{t+1}^{\dagger}$ is some pseudoinverse of J_{t+1}.

The new tangent space mean is

$$E\delta x(t+1) = J_{t+1}^{\dagger}\delta i \tag{3.104}$$

leading to

$$\delta x(t+1) = \mathcal{N}(J_{t+1}^{\dagger}\delta i, P(t+1)) \tag{3.105}$$

where

$$P(t+1) = (J_{t+1}^{\dagger}J_t)P(t)(J_{t+1}^{\dagger}J_t)^T + (J_{t+1}^{\dagger})Q(t)(J_{t+1}^{\dagger})^T \tag{3.106}$$

is the covariance.

Equivalently, we can say

$$x(t+1) = \mathcal{N}(\mathcal{A}(\hat{x}(t+1), J_{t+1}^{\dagger}\delta i), P(t+1)) \tag{3.107}$$

where $\hat{x}(t+1)$ is provided from the prediction stage of the Kalman filter.

State transition function

When the constraint is in terms of a state transition function

$$f : \mathcal{S} \mapsto \mathcal{S} \tag{3.108}$$

such that

$$x(t+1) = f(x(t)) \tag{3.109}$$

and the state space is not Euclidean, the implicit constraint becomes

$$i(x(t), x(t+1)) = \mathcal{A}^{-1}(f(x(t)), x(t+1)) \tag{3.110}$$

leading to

$$\left.\frac{\partial i(\xi,\zeta)}{\partial \xi}\right|_{\substack{\xi = \hat{x}(t)\\ \zeta = \hat{x}(t+1)}} = \left.\frac{\partial \mathcal{A}^{-1}(\xi,\zeta)}{\partial \xi}\right|_{\substack{\xi = f(\hat{x}(t))\\ \zeta = \hat{x}(t+1)}} \left.\frac{\partial f(\xi)}{\partial \xi}\right|_{\xi = \hat{x}(t)} \tag{3.111}$$

$$\left.\frac{\partial i(\xi,\zeta)}{\partial \zeta}\right|_{\substack{\xi = \hat{x}(t)\\ \zeta = \hat{x}(t+1)}} = \left.\frac{\partial \mathcal{A}^{-1}(\xi,\zeta)}{\partial \zeta}\right|_{\substack{\xi = f(\hat{x}(t))\\ \zeta = \hat{x}(t+1)}} \tag{3.112}$$

and to the relationship

$$\hat{x}(t+1) = f(\hat{x}(t)) \tag{3.113}$$

which together lead to

$$\begin{aligned} \delta i &\doteq -i(\hat{x}(t), \hat{x}(t+1)) && (3.114)\\ &= -f(\hat{x}(t)) + \hat{x}(t+1) && (3.115)\\ &= 0 && (3.116) \end{aligned}$$

and to the linearized system

$$J_t \delta x(t) + J_{t+1}\delta x(t+1) + w(t) = 0 \tag{3.117}$$

where

$$J_t = \left.\frac{\partial \mathcal{A}^{-1}(\xi,\zeta)}{\partial \xi}\right|_{\substack{\xi = f(\hat{x}(t))\\ \zeta = \hat{x}(t+1)}} \left.\frac{\partial f(\xi)}{\partial \xi}\right|_{\xi = \hat{x}(t)} \left.\frac{\partial \mathcal{A}(\xi,\delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = \hat{x}(t)\\ \delta\xi = 0}} \tag{3.118}$$

$$J_{t+1} = I \tag{3.119}$$

which arise from Eqn. 3.96, Eqn. 3.97, Eqn. 3.111, Eqn. 3.112, and the identity of Eqn. 3.38.

Thus, if we start with

$$x(t) = \mathcal{N}(\hat{x}(t), P(t)) \tag{3.120}$$

the error propagation rule is

$$x(t+1) = \mathcal{N}(f(\hat{x}(t)), P(t+1)) \tag{3.121}$$

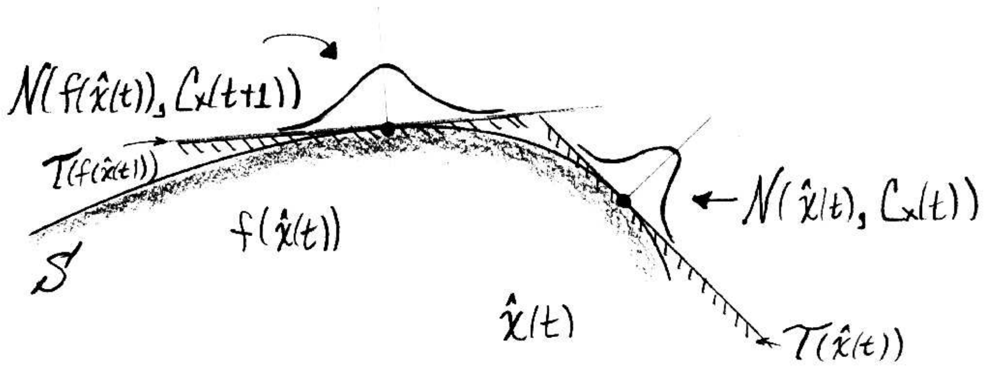


Figure 3-7: Error propagation with a state transition dynamic constraint

where

$$P(t+1) = J_t(\hat{x}(t))P(t)J_t^T(\hat{x}(t)) + Q(t) \tag{3.122}$$

and J_t

$$J_t(\hat{x}(t)) \doteq \left.\frac{\partial \mathcal{A}^{-1}(\xi,\zeta)}{\partial \xi}\right|_{\substack{\xi = f(\hat{x}(t)) \\ \zeta = f(\hat{x}(t))}} \left.\frac{\partial f(\xi)}{\partial \xi}\right|_{\xi = \hat{x}(t)} \left.\frac{\partial \mathcal{A}(\xi,\delta\xi)}{\partial \delta\xi}\right|_{\substack{\xi = \hat{x}(t) \\ \delta\xi = 0}} \tag{3.123}$$

which follows directly from Eqn. 3.118 and Eqn. 3.113.

3.5.2 Observation and modeling errors

There are two sources of errors that will prevent the constraint equation $e(x, y)$ from vanishing exactly: measurement errors (noise) and modeling errors. Neither can be avoided in general, but both types of errors can be modeled probabilistically.

The measurement is modeled as a random variable

$$y(t) = \mathcal{N}(\bar{y}(t), R(t)) \tag{3.124}$$

or

$$y(t) = \bar{y}(t) + \delta y(t) \tag{3.125}$$

where $y(t)$ is the recorded measurement, $\bar{y}(t)$ is a noiseless measurement, and $\delta y(t)$ is a random variable added to a noiseless measurement representing noise in the measurement process.

The random variable $\delta y(t)$ thus is zero-mean with

$$E\delta y(t) = 0 \tag{3.126}$$

and

$$R(t) = E\delta y(t)\delta y^T(t) \tag{3.127}$$

and is Gaussian distributed with covariance matrix $R(t)$.

Constraint modeling error is modeled as a random variable $v(t)$ added to the noiseless constraint:

$$e(x(t), y(t)) + v(t) = 0 \tag{3.128}$$

where $e(x, y)$ is the noiseless constraint.

The random variable $v(t)$ is usually modeled as zero-mean with

$$Ev(t) = 0 \tag{3.129}$$

and

$$S(t) = Ev(t)v^T(t) \tag{3.130}$$

and is modeled as Gaussian distributed with a covariance matrix $S(t)$.

It will be convenient to express both the measurement and modeling errors in error space. To this end, the linearization Eqn. 3.63 leads to

$$\delta e(\hat{x}(t), y(t)) = J_x(\hat{x}(t), y(t))\delta x + \underbrace{J_y(\hat{x}(t), y(t))\delta y + v(\hat{x}(t), y(t))}_{v'(\hat{x}(t), y(t))} \tag{3.131}$$

in which v' is a new random variable. Based on the linearization, the mean and covariance of v' can be computed as

$$Ev'(\hat{x}(t), y(t)) = J_y(\hat{x}(t), y(t))E\delta y + Ev(\hat{x}(t), y(t)) = 0 \tag{3.132}$$

$$Ev'(\hat{x}(t), y(t))v'(\hat{x}(t), y(t))^T = J_y(\hat{x}(t), y(t))R(t)J_y^T(\hat{x}(t), y(t)) + S(t) \tag{3.133}$$

a result which will be used later.

3.5.3 A generalized EKF for manifold-tangent models

A generalization of the extended Kalman filter for manifold-tangent models is relatively straightforward using the tangent-space linearizations discussed in Section 3.2 and the probability models discussed in Section 3.4. Below is an algorithm for a *generalized extended Kalman filter* (GEKF) based on these properties of the manifold-tangent models. It follows the same approach as the traditional EKF but represents state probability distributions in tangent spaces and uses implicit constraints for internal and external constraint.

The algorithm is summarized here, and its relationship to the original Kalman filter (KF) for Euclidean linear systems and the extended Kalman filter (EKF) for Euclidean nonlinear systems are taken up in the following two sections.

Algorithm 4 (GEKF) *Given a manifold-tangent state model $\mathcal{M}$ and internal and external constraint models*

$$\begin{aligned} i(x(t), x(t+1)) + w(t) = 0 \quad &\leftarrow \quad \text{internal constraint} \\ e(x(t), y(t)) + v(t) = 0 \quad &\leftarrow \quad \text{external constraint} \end{aligned}$$

the following random variables

$$\begin{aligned} x(t_1|t_2) &\doteq \mathcal{N}(\hat{x}(t_1|t_2), P(t_1|t_2)), \forall t_1, t_2 \\ \delta y(t) &\doteq \mathcal{N}(0, R(t)) \\ v(t) &\doteq \mathcal{N}(0, S(t)) \\ w(t) &\doteq \mathcal{N}(0, Q(t)) \end{aligned}$$

and the following abbreviations

$$\begin{aligned} \delta e(t) &\doteq -e(\hat{x}(t|t-1), y(t)) \\ J_x(t) &\doteq J_x(\hat{x}(t|t-1), y(t)) \qquad \leftarrow \text{(Eqn. 3.64)} \end{aligned}$$

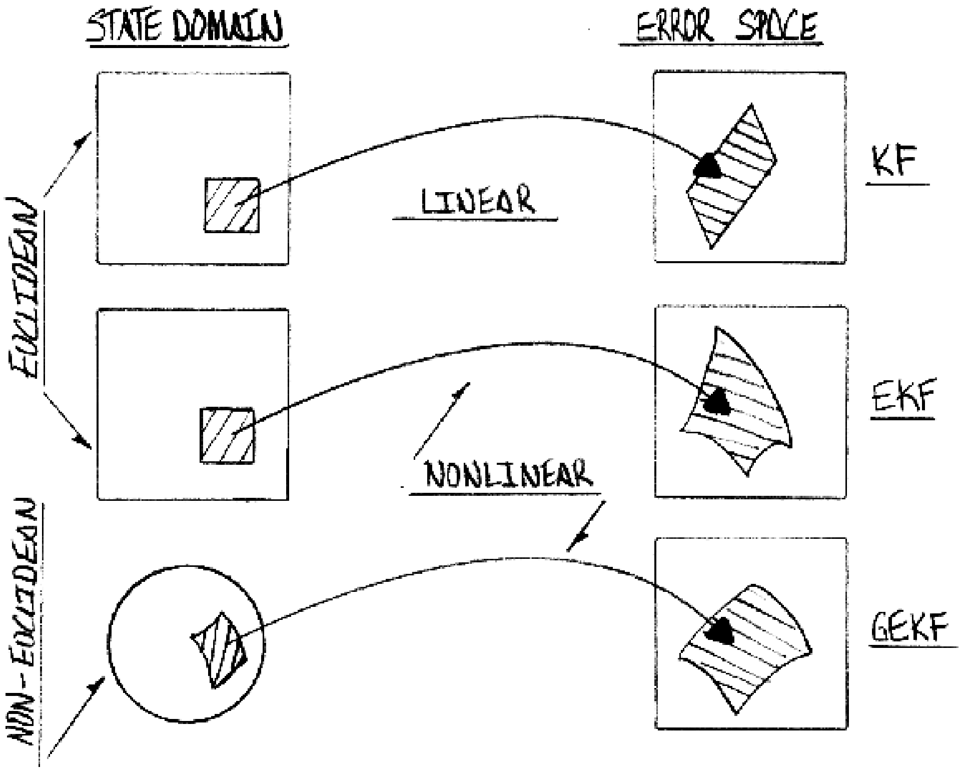


Figure 3-8: The Kalman filter (KF) is for a linear function mapping a Euclidean space to an error metric. The extended Kalman filter (EKF) is for a nonlinear function mapping a Euclidean space to an error metric. The generalized EKF is for a nonlinear function mapping a non-Euclidean space to an error metric.

$$
\begin{aligned}
J_y(t) &\doteq J_y(\hat{x}(t|t-1), y(t)) \qquad \leftarrow \text{(Eqn. 3.65)} \\
J_t(t) &\doteq J_t(\hat{x}(t|t)) \qquad \leftarrow \text{(Eqn. 3.123)}
\end{aligned}
$$

the GEKF can be summarized as follows:

input $\hat{x}(1|0), P(1|0)$
for each $t = 1, 2, \ldots$
  input $y(t), R(t), S(t), Q(t)$
  solve linearized system
    $K(t) = P(t|t-1)J_x^T(t)\left[J_x(t)P(t|t-1)J_x^T(t) + J_y(t)R(t)J_y^T(t) + S(t)\right]^{-1}$
    $\widehat{\delta x}(t|t-1) = K(t)\delta e(t)$
  update
    $\hat{x}(t|t) = \mathcal{A}(\hat{x}(t|t-1), \widehat{\delta x}(t|t-1))$
    $P(t|t) = (I - K(t)J_x(t))P(t|t-1)$
  predict
    $\hat{x}(t+1|t) = f(\hat{x}(t|t))$
    $P(t+1|t) = J_t(t)P(t|t)J_t^T(t) + Q(t)$
end

■

The notation $\hat{x}(t_1|t_2)$ is meant to indicate an estimate of state vector $x(t)$ at $t = t_1$ given measurements through time t_2.

Note that covariance matrices $R(t)$, $S(t)$, and $Q(t)$ are inputs to the system. They can be considered static priors, or determined dynamically by the measurement process (R) and an internal model (S,Q).

3.5.4 Euclidean nonlinear systems and the EKF

For Euclidean systems, the tangent space is coincident with the state space and so the state error probability distribution covariances

$$\begin{aligned} P(t|t-1) &\doteq E(x(t)-\hat{x}(t|t-1))(x(t)-\hat{x}(t|t-1))^T && (3.134)\\ P(t|t) &\doteq E(x(t)-\hat{x}(t|t))(x(t)-\hat{x}(t|t))^T && (3.135)\\ P(t+1|t) &\doteq E(x(t)-\hat{x}(t+1|t))(x(t)-\hat{x}(t+1|t))^T && (3.136)\end{aligned}$$

can be expressed as covariances on vector differences in the state space.

In the KF and EKF formulations, only direct forward models

$$y(t) = h(x(t)) + v'(t) \tag{3.137}$$

are considered. This equation can be rewritten as

$$\begin{aligned} e(x(t), y(t)) + v(t) &= \overbrace{h(x(t)) - y(t) + \underbrace{\delta y(t)}_{}}^{e(x,\,y-\delta y)}\underbrace{{}+v(t)}_{v'(t)} && (3.138)\\ &= 0 && (3.139)\end{aligned}$$

to elucidate the relationship with the generalized version. It is clear that the Jacobian matrix

$$\begin{aligned} J_x(t) &\doteq \left.\frac{\partial e(\mathcal{A}(x,\delta x), y)}{\partial \delta x}\right|_{x\,=\,\hat{x}(t|t-1)} && (3.140)\\ &= \left.\frac{\partial h}{\partial x}\right|_{x\,=\,\hat{x}(t|t-1)} && (3.141)\end{aligned}$$

becomes simply the Jacobian of the forward model $h(x)$ and the Jacobian matrix

$$\begin{aligned} J_y(t) &\doteq \frac{\partial e(\mathcal{A}(x,\delta x), y)}{\partial y} && (3.142)\\ &= -I && (3.143)\end{aligned}$$

becomes the (negative) identity matrix.

In the traditional EKF, the measurement noise $\delta y(t)$ and measurement modeling error $v(t)$ are bunched up into a single term $v'(t)$ with a covariance

$$R'(t) = Ev'(t)v'(t)^T \tag{3.144}$$

which can be done here because both errors enter the equation in the same place. (Actually, the way the KF is usually described, the assumption is that the model is perfect but there is measurement noise; since mathematically it does not matter whether the errors are due to measurement noise or modeling error, it is not really an issue. But in the GEKF, the modeling error and measurement error enter in different locations and are therefore represented separately.) In any case, in relation to the GEKF the covariance matrix

$$R'(t) = J_y R(t) J_y^T + S(t) \tag{3.145}$$

represents all error in the measurement equation where $J_y = -I$. One can therefore assume that there are perfect measurements ($R = 0$) in which case $R' = S$ describes modeling error, or that there is a perfect model ($S = 0$) in which case $R' = R$ describes measurement error, or that neither is perfect and that $R' = R + S$ is a combination of both.

The "innovation"

$$\begin{aligned} \delta e(t) &\doteq -e(\hat{x}(t|t-1), y(t)) && (3.146)\\ &= y(t) - h(\hat{x}(t|t-1)) && (3.147) \end{aligned}$$

becomes the usual vector difference between observed and predicted measurement vectors.

The Jacobian of the dynamic constraint

$$\begin{aligned} J_t(t) &\doteq J_t(\hat{x}(t|t)) && (3.148)\\ &= \frac{\partial f(x)}{\partial x} && (3.149) \end{aligned}$$

becomes simply the Jacobian of f.

And the dynamic process noise

$$Q(t) \doteq E w(t) w^T(t) \tag{3.150}$$

can be written as a covariance matrix in state space where the dynamic constraint

$$x(t+1) = f(x(t)) + w(t) \tag{3.151}$$

can be written as vector addition in state space.

These specializations of the GEKF yield the following algorithm, which is the traditional *extended Kalman filter* [18, 29].

Algorithm 5 (EKF) *Given a dynamic model*

$$x(t+1) = f(x(t)) + w(t)$$

with $Ew(t) = 0$ *and* $Ew(t)w^T(t) = Q(t)$ *and an measurement model*

$$y(t) = h(x(t)) + v'(t)$$

with $Ev(t) = 0$ *and* $Ev'(t)v'^T(t) = R'(t)$ *and a state error covariance*

$$P(t|t') = E(x(t) - \hat{x}(t|t'))(x(t) - \hat{x}(t|t'))^T$$

and the following abbreviations

$$\begin{aligned} J_x(t) &\doteq \left.\frac{\partial h(x)}{\partial x}\right|_{x = \hat{x}(t|t-1)}\\ \delta e(t) &\doteq y(t) - h(\hat{x}(t|t-1)) \end{aligned}$$

$$J_t(t) \doteq \left.\frac{\partial f(x)}{\partial x}\right|_{x = \hat{x}(t|t)}$$

the EKF can be summarized as follows:

```
input x̂(t|t−1), P(t|t−1)
for each t = 1, 2, ...
   input y(t), R(t)
   solve linearized system
      K(t) = P(t|t−1) J_x^T(t) [J_x(t) P(t|t−1) J_x^T(t) + R'(t)]^{−1}
      δx̂(t|t−1) = K(t) δe(t)
   update
      x̂(t|t) = x̂(t|t−1) + δx̂(t|t−1)
      P(t|t) = (I − K(t) J_x(t)) P(t|t−1)
   predict
      x̂(t+1|t) = f(x̂(t|t))
      P(t+1|t) = J_t(t) P(t|t) J_t^T(t) + Q(t)
end
```

■

3.5.5 Euclidean linear systems and the KF

Finally, if the constraint and dynamic models are both linear

$$h(x(t)) = J_x(t)x(t) \tag{3.152}$$

$$f(x(t)) = J_t(t)x(t) \tag{3.153}$$

where $x \in \mathbb{R}^n$, $J_x \in \mathbb{R}^{p\times n}$, and $J_t \in \mathbb{R}^{n\times n}$, then we recover the traditional Kalman filter, which is a provably optimal filter for Euclidean linear systems in the sense of producing the minimum variance estimate [29, 18].

The algorithm is summarized here.

Algorithm 6 (Kalman filter (KF)) *Given a linear dynamic model*

$$x(t+1) = J_t(t)x(t) + w(t)$$

with $Ew(t) = 0$ *and* $Ew(t)w^T(t) = Q(t)$ *and a linear measurement model*

$$y(t) = J_x x(t) + v'(t)$$

with $Ev(t) = 0$ *and* $Ev'(t)v'^T(t) = R'(t)$ *and a state error covariance*

$$P(t|t') = E(x(t) - \hat{x}(t|t'))(x(t) - \hat{x}(t|t'))^T$$

the Kalman filter (KF) can be summarized as follows:

GEKF	EKF	KF
$x \in \mathbb{R}^m, \delta x \in \mathbb{R}^n, m \geq n$	$x, \delta x \in \mathbb{R}^n$	
Perturbation:		
$\mathcal{A}(x_1, \delta x) = x_2$	$x_1 + \delta x = x_2$	
Inverse Perturbation:		
$\mathcal{A}^{-1}(x_1, x_2) = \delta x$	$x_2 - x_1 = \delta x$	
Linearized Perturbation:		
$\frac{\partial \mathcal{A}}{\partial \delta x} \in \mathbb{R}^{m \times n}$	$\frac{\partial \mathcal{A}}{\partial \delta x} = I_{n \times n}$	
Linearized Inverse Perturbation:		
$\frac{\partial \mathcal{A}^{-1}}{\partial x} \in \mathbb{R}^{n \times m}$	$\frac{\partial \mathcal{A}^{-1}}{\partial x} = I_{n \times n}$	
Dynamic Model:		
$x(t+1) = \mathcal{A}(f(x(t)), w(t))$	$x(t+1) = f(x(t)) + w(t)$	$x(t+1) = J_t x(t) + w(t)$
Observation Model:		
$e(x(t), y(t)) + v(t) = 0$	$h(x(t)) - y(t) + v(t) = 0$	$J_x(t)x(t) - y(t) + v(t) = 0$

Figure 3-9: Comparison between the models used in the traditional KF and EKF versus the generalized version used here.

input $\hat{x}(t|t-1), P(t|t-1)$
for each $t = 1, 2, \ldots$
  input $y(t), R(t)$
  solve linearized system
    $K(t) = P(t|t-1)J_x^T(t)\left[J_x(t)P(t|t-1)J_x^T(t) + R'(t)\right]^{-1}$
    $\hat{\delta x}(t|t-1) = K(t)\delta e(t)$
  update
    $\hat{x}(t|t) = \hat{x}(t|t-1) + \hat{\delta x}(t|t-1)$
    $P(t|t) = (I - K(t)J_x(t))P(t|t-1)$
  predict
    $\hat{x}(t+1|t) = f(\hat{x}(t|t))$
    $P(t+1|t) = J_t(t)P(t|t)J_t^T(t) + Q(t)$
end

■

The table in Fig. 3-9 summarizes the GEKF and its relationship to the KF and EKF.

3.6 Model and error analysis

This section formulates a probabilistic method for estimating state covariances and evaluating constraint models.

3.6.1 Probabilistic error propagation

The basic mathematical problem is to evaluate the quality of a state estimate with respect to set of measurements given a constraint model

$$e(x, y) = 0 \tag{3.154}$$

relating the state x to the measurements y. This section derives the basic mathematical relationships which can then be used as a tool to evaluate various models with respect to a given physical state ("model analysis") or to evaluate various state estimates with respect to a given model ("error analysis").

The linearization of Eqn. 3.63 leads to the relationship

$$e(\hat{x}, \hat{y}) + J_x(\hat{x}, \hat{y})\delta x + J_y(\hat{x}, \hat{y})\delta y + \cdots = 0 \tag{3.155}$$

where $\hat{x}$ is a state estimate compatible with measurement vector $\hat{y}$. In overdetermined systems, $e(\hat{x}, \hat{y})$ will not vanish exactly, but it will be small. The linearization leads to

$$\begin{aligned} J_x \delta x &= -e - J_y \delta y & (3.156) \\ J_x^T J_x \delta x &= -J_x^T e - J_x^T J_y \delta y & (3.157) \end{aligned}$$

which is the system of normal equations for the first-order variations in the system.

The Moore-Penrose pseudoinverse is

$$\begin{aligned} J_x^\dagger &= (J_x^T J_x)^{-1} J_x^T & (3.158) \\ &= (V\Sigma U^T U \Sigma V^T)^{-1} V \Sigma U^T & (3.159) \\ &= V \Sigma^{-1} U^T & (3.160) \end{aligned}$$

where

$$J_x = U \Sigma V^T \tag{3.161}$$

is the singular value decomposition (see Section B.6) of the Jacobian matrix and exists if and only if Σ^{-1} exists.

3.6.2 Nonsingular normal equations

If the pseudoinverse exists, the relationship

$$\delta x = -J_x^\dagger e - J_x^\dagger J_y \delta y \tag{3.162}$$

expresses the forward relationship between δy and δx. Using this relationship, if we model δy as a zero-mean random variable with covariance $R = E\delta y \delta y^T$, then the random variable δx can be characterized as

$$E\delta x = -J_x^\dagger e \tag{3.163}$$

$$\begin{aligned} P &= E(\delta x + J_x^\dagger e)(\delta x + J_x^\dagger e)^T & (3.164) \\ &= (J_x^\dagger J_y) R (J_x^\dagger J_y)^T & (3.165) \end{aligned}$$

where P is the covariance of the state variation δx. This matrix can be related to the *Fisher information matrix* and the *Cramér-Rao inequality* expressing a lower bound on parameter uncertainty [42].

Thus,

$$\delta y \sim \mathcal{N}(0, R) \quad \Rightarrow \quad \delta x \sim \mathcal{N}(-J_x^\dagger e, (J_x^\dagger J_y) R (J_x^\dagger J_y)^T) \tag{3.166}$$

expresses the propagation of error from measurements to state.

3.6.3 Singular normal equations

However, if the diagonal matrix Σ is singular, then a singular analysis must take place. If the normal system (Eqn. 3.157) is singular then some number of the diagonal elements of Σ

will be small compared to the rest and the matrix can be expressed as

$$\Sigma = \begin{pmatrix} \Sigma' & \\ & 0 \end{pmatrix} \in \mathbb{R}^{n \times n} \tag{3.167}$$

where

$$\begin{aligned} \Sigma' &\in \mathbb{R}^{m \times m} && (3.168) \\ 0 &\in \mathbb{R}^{n-m \times n-m} && (3.169) \end{aligned}$$

where $m < n$ and Σ' is invertible.

The orthonormal matrices

$$V = \begin{pmatrix} V' & V^0 \end{pmatrix} \in \mathbb{R}^{n \times n} \tag{3.170}$$

$$U = \begin{pmatrix} U' & U^0 \end{pmatrix} \in \mathbb{R}^{p \times n} \tag{3.171}$$

can be decomposed in a corresponding way, where

$$\begin{aligned} V' &\in \mathbb{R}^{n \times m} && (3.172) \\ U' &\in \mathbb{R}^{p \times m} && (3.173) \end{aligned}$$

represent the nonsingular subspaces and with some algebra lead to

$$\begin{aligned} \delta x &= (V')(\Sigma')^{-1}(U')^T(-e - J_y \delta y) && (3.174) \\ (V')^T \delta x &= (\Sigma')^{-1}(U')^T(-e - J_y \delta y) && (3.175) \end{aligned}$$

where the second form emphasizes that only a subspace of state tangent space is actually constrained by the measurements.

The rest of the space, represented by V^0, is not reached by the measurements and therefore remains completely unconstrained, with infinite variance along these directions. Any state parameters represented in the basis set V^0 will be unreliable.

3.6.4 Numerical model analysis

Given a state and a measurement that are known to be physically well-conditioned, we can use the above mathematics to see which of several models is best conditioned numerically.

Each model along with the known state $\hat{x}$ and associated measurement $\hat{y}$ will yield $e(\hat{x}, \hat{y}) = 0$. Each nonsingular model will have a precision relationship

$$\delta y \sim \mathcal{N}(0, R) \quad \Rightarrow \quad \delta x \sim \mathcal{N}(0, (J_x^\dagger J_y) R (J_x^\dagger J_y)^T) \tag{3.176}$$

and can be compared to see which one is more precise. If one constraint model produces lower parameter variances than another model for the same state and measurement pair, one can say that the first model is more precise with respect to that region of state space.

If one model is nonsingular and the other is singular, the nonsingular model clearly performs better in that region of state space.

If the physical situation is not singular, the goal of modeling should be first to find a numerically nonsingular model for the regions of state space that are of interest. Secondary modeling steps can be taken to find models that are better conditioned numerically.

This model analysis is used in Section 4.1.4 to compare linear and nonlinear models of the same geometry.

3.6.5 Numerical error analysis

Given a model, a set of measurements, and a state estimate, we can use the same mathematics to compute the precision of the state parameter estimates.

If the normal system is nonsingular, then the measurements sufficiently constrain the state and the precision can be computed using Eqn. 3.166.

Assuming the constraint model is well-designed and sufficient, a singular normal system indicates that the measurements are insufficient to constrain the state. The state may be partially constrained and the constraint structure can be seen in the SVD. The important information is which directions of state space are degenerate and is contained in the null space matrix V^0. Any state tangent parameters represented in the $n-m$ vectors of V^0 are unreliable and can be considered to have infinite, or very large, variance.

3.6.6 Analytical Jacobian analysis

A sufficient (but not necessary) condition for the normal equations to be singular is if J_x has deficient column rank (i.e. rank $< n$). Since the Jacobian matrix J_x can usually be described analytically, this condition can be checked analytically, and singular cases can be spotted in the modeling process.

If the Jacobian is described via a chain of matrices, e.g.

$$J_x = J_1 J_2 \cdots J_N \tag{3.177}$$

then each matrix in the chain must have rank at least n.

Although linear dependencies can be checked for analytically, the easiest case to spot by far is when a column of J_x can become zero. This technique is used in Section 3.7.5 to develop a representation for relative orientation of a camera that remains nonsingular over the desired range of parameters.

3.7 Some models for vision-based geometry

This section develops model parameterizations for important geometric entities, including translation, rotation, and camera.

3.7.1 3-D translation model

The translation model is a parameterization for the physical concept of a translated 3-D reference frame. The translation parameters correspond directly to the distance in one reference frame of the origin of another.

Fig. 3-10 illustrates the physical meaning of the three parameters associated with 3-D translation.

Since every finite point in $\mathbb{R}^3$ is a possible translation, the translation state space can be modeled as a simple Euclidean space with three parameters, i.e.

$$\mathcal{M}^{(tra)} \equiv \mathcal{M}^3_{Euc} \tag{3.178}$$

or, explicitly, the model specification

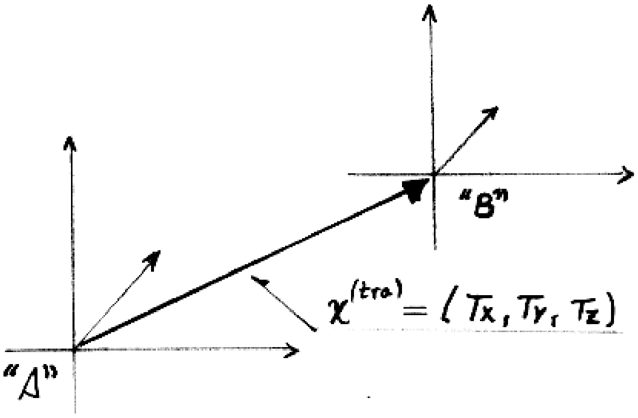


Figure 3-10: Physical model of reference frame translation.

$\mathcal{M}^{(tra)}$:

$$\begin{aligned}
\mathcal{P}_S^{(tra)} &= \mathrm{I\!R}^3 \\
\mathcal{S}^{(tra)} \subseteq \mathcal{P}_S^{(tra)} &= \mathrm{I\!R}^3 \\
\mathcal{P}_T^{(tra)}(x), x \in \mathcal{S}^{(tra)} &= \mathrm{I\!R}^3 \\
\mathcal{T}^{(tra)}(x), x \in \mathcal{S}^{(tra)} &= \mathrm{I\!R}^3 \\
\mathcal{Z}^{(tra)}(x), x \in \mathcal{S}^{(tra)} &= I \\
\mathcal{A}^{(tra)} &= +_3, \quad \text{i.e. 3-D Cartesian vector addition}
\end{aligned} \tag{3.179}$$

summarizes the state space.

Under this parameterization, a translation function

$$h^{(tra)} : \mathcal{S}^{(tra)} \times \mathrm{I\!R}^3 \mapsto \mathrm{I\!R}^3 \tag{3.180}$$

maps the coordinates of a physical point relative to a translated reference frame "B" into the coordinates of the same physical point relative to reference frame "A" according to

$$p_A = h^{(tra)}(x^{(tra)}, p_B) \tag{3.181}$$
$$= x^{(tra)} + p_B \tag{3.182}$$

where the 3-D translation represents the physical location of the "B" reference frame relative to the "A" reference frame (see Fig. 3-10).

3.7.2 3-D rotation model

The rotation model is a parameterization for the physical concept of a rotated 3-D reference frame. The rotation parameters correspond indirectly to a 3-D axis of rotation and an angle of rotation.

Fig. 3-11 illustrates the physical meaning of an axis-angle parameterization of rotation, where n represents a 3-D vector called the axis and θ represents the angle of rotation.

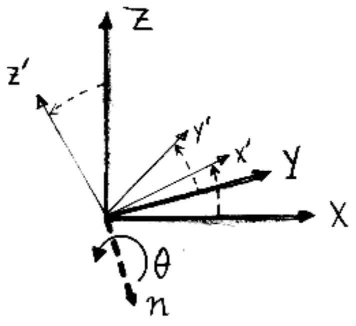


Figure 3-11: Physical model of reference frame rotation

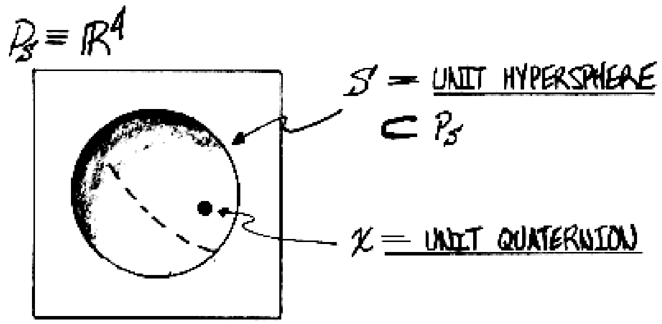


Figure 3-12: Depiction of the rotation parameter space and the state manifold. The rotation quaternion inhabits a 4-D unit hypersphere.

Hamilton's *unit quaternion* representation [31, 35] (see Appendix A) consists of four parameters which bear a simple relationship to the physical interpretation. The scalar "real part" of the unit quaternion

$$q_0 = \cos(\theta/2) \tag{3.183}$$

and the vector "imaginary part"

$$\begin{pmatrix} q_X \\ q_Y \\ q_Z \end{pmatrix} = n \sin(\theta/2) \tag{3.184}$$

combine to form a 4-vector of unit length.

Topologically, the unit 4-vector parameterizes the set of all points on a 4-D unit sphere. However, since q and $-q$ represent the same physical state, the state space can be thought of as a hemi-4-sphere with a one-to-one mapping or a 4-sphere with a two-to-one mapping between parameter states and physical states.

Fig. 3-12 depicts the relationship between state parameter space and the state manifold.

We can use quaternion algebra to find a suitable and physically meaningful specification of tangent space at a given rotational state and to specify a rotation composition function, $\mathcal{A}^{(rot)}$.

First, consider the composition of any rotation q^* with an additional small rotation quaternion δq to produce a new rotation q. Consider the axis/angle interpretation of the quaternion

$$\delta q = (\cos(\theta/2), n\sin(\theta/2)) \tag{3.185}$$

where n is a unit 3-D rotational axis and θ is the magnitude of rotation around that axis, and is small (say, less than 30 degrees). Then δq can be approximated as

$$\delta q = \left(\sqrt{1-\epsilon}, \omega_X/2, \omega_Y/2, \omega_Z/2\right) \tag{3.186}$$

where

$$\epsilon = (\omega_X^2 + \omega_Y^2 + \omega_Z^2)/4 \tag{3.187}$$

and $\omega_X = n_X\theta$, $\omega_Y = n_Y\theta$, $\omega_Z = n_Z\theta$ are approximately Euler angles, provided the magnitude θ remains small.

The composition of q^* with δq is

$$\begin{aligned}
q &= \begin{pmatrix} q_0^* & -q_1^* & -q_2^* & -q_3^* \\ q_1^* & q_0^* & q_3^* & -q_2^* \\ q_2^* & -q_3^* & q_0^* & q_1^* \\ q_3^* & q_2^* & -q_1^* & q_0^* \end{pmatrix} \begin{pmatrix} \delta q_0 \\ \delta q_1 \\ \delta q_2 \\ \delta q_3 \end{pmatrix} && (3.188) \\
&= \begin{pmatrix} q^* & v_1 & v_2 & v_3 \end{pmatrix} \begin{pmatrix} \delta q_0 \\ \delta q_1 \\ \delta q_2 \\ \delta q_3 \end{pmatrix} && (3.189) \\
&= \begin{pmatrix} q^* \end{pmatrix} + \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} \begin{pmatrix} \omega_X/2 \\ \omega_Y/2 \\ \omega_Z/2 \end{pmatrix} + (\sqrt{1-\epsilon} - 1) \begin{pmatrix} q^* \end{pmatrix} && (3.190) \\
&\doteq P^{(rot)} \left(\begin{pmatrix} q^* \end{pmatrix} + \begin{pmatrix} v_1 & v_2 & v_3 \end{pmatrix} \begin{pmatrix} \omega_X/2 \\ \omega_Y/2 \\ \omega_Z/2 \end{pmatrix} \right) && (3.191) \\
&\doteq P^{(rot)}(q^* + \mathcal{Z}^{(rot)}(q^*)(\Omega)) && (3.192) \\
&\doteq \mathcal{A}^{(rot)}(q^*, \Omega) && (3.193)
\end{aligned}$$

where

$$\Omega = \begin{pmatrix} \omega_X \\ \omega_Y \\ \omega_Z \end{pmatrix} \tag{3.194}$$

represents the tangent space parameters.

With these results from quaternion algebra, the following specification

$\mathcal{M}^{(rot)}$:

$$\begin{aligned}
\mathcal{P}_S^{(rot)} &= \mathbb{R}^4 \\
\mathcal{S}^{(rot)} \subseteq \mathcal{P}_S^{(rot)} &= \{z \in \mathcal{P}_S \,|\, \langle z, z \rangle = 1\} \\
\mathcal{P}_T^{(rot)}(x), x \in \mathcal{S}^{(rot)} &= \mathbb{R}^3 \\
\mathcal{T}^{(rot)}(x), x \in \mathcal{S}^{(rot)} &= \{z \in \mathcal{P}_S \,|\, \langle z, x \rangle = 1\} \\
\mathcal{Z}^{(rot)}(x), x \in \mathcal{S}^{(rot)} &= \text{Defined by Eqn. 3.191, Eqn. 3.192, and Eqn. 3.187} \\
\mathcal{A}^{(rot)} &= \text{Defined by Eqn. 3.193}
\end{aligned} \tag{3.195}$$

summarizes the manifold-tangent model for rotation.

Under this parameterization, a rotation function

$$h^{(rot)} : \mathcal{S}^{(rot)} \times \mathbb{R}^3 \mapsto \mathbb{R}^3 \tag{3.196}$$

maps the coordinates of a physical point relative to a rotated reference frame "B" into the coordinates of the same physical point relative to reference frame "A" according to

$$p_A = h^{(rot)}(x^{(rot)}, p_B) \tag{3.197}$$

$$= \begin{pmatrix} q_0^2 + q_X^2 - q_Y^2 - q_Z^2 & 2(q_X q_Y - q_0 q_Z) & 2(q_X q_Z + q_0 q_Y) \\ 2(q_X q_Y + q_0 q_Z) & q_0^2 - q_X^2 + q_Y^2 - q_Z^2 & 2(q_Y q_Z - q_0 q_X) \\ 2(q_X q_Z - q_0 q_Y) & 2(q_Y q_Z + q_0 q_X) & q_0^2 - q_X^2 - q_Y^2 + q_Z^2 \end{pmatrix} \begin{pmatrix} p_B \end{pmatrix} \tag{3.198}$$

where

$$x^{(rot)} = \begin{pmatrix} q_0 \\ q_X \\ q_Y \\ q_Z \end{pmatrix} \tag{3.199}$$

represents the physical 3-D rotation of the "B" reference frame relative to the "A" reference frame (see Fig. 3-11).

3.7.3 Camera model—interior orientation

The interior orientation is a parameterization for the physical concept of a pinhole projection imaging system, which is an adequate approximation to the lens-based imaging systems used in practice. The interior orientation parameterization should specify the internal imaging geometry, i.e. the extent of the perspective distortion induced by the projection.

Fig. 3-13 illustrates the physical model of the pinhole imaging process. All light rays emanating from some point in the scene that pass through the pinhole become imaged at a particular point on a a physical surface by following a straight-line path. Typically, a rectangle centered on the optical axis and lying in a plane normal to the optical axis is the image sensor, which records the intensity and spectral characteristics of the light. These rectangular spatial distributions of light measurements, quantized in space and time, constitute the digital images which we use as the visual measurements for computer vision.

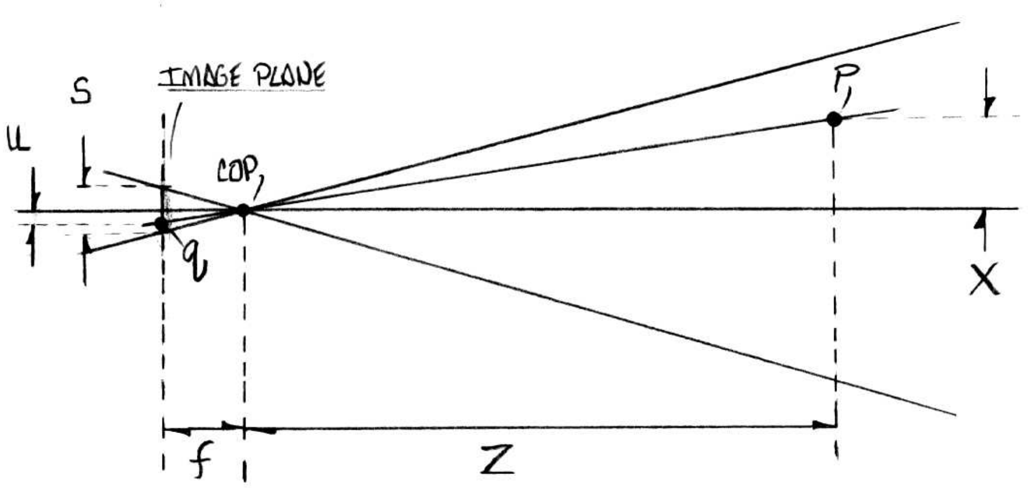


Figure 3-13: Camera pinhole model and the geometry of similar triangles.

COP-based coordinate system

The important geometric relationship between a scene point p and its associated image point q in the pinhole model is captured by similar triangles. As depicted in Fig. 3-13, if O is the COP, p' is the projection of p on the optical axis, and q' is the center of the image rectangle, then the triangle similarity relationship

$$Oqq' \sim Opp' \tag{3.200}$$

holds for every point p and its associated projection q.

Analytically, we can establish a natural coordinate system where the origin is the COP, the z-axis is along the optical axis, and the xy-plane is parallel to the image plane. Then, if p has coordinates (X, Y, Z) and q has coordinates $(-u, -v, -f)$, the relationship

$$\begin{pmatrix} u/f \\ v/f \end{pmatrix} = \begin{pmatrix} X/Z \\ Y/Z \end{pmatrix} \tag{3.201}$$

encodes the similar triangles relationship.

Virtual plane coordinate system

There is another, perhaps more useful, interpretation of the geometry, which starts with a virtual image and establishes a "collinearity" relationship between each point in the virtual plane and the scene point that lies on the same ray. Fig. 3-14 depicts the geometric concept in which a virtual image plane normal to the optical axis is constructed a distance Z_f on the scene side of the COP. Each optical ray from a scene point p intersects the physical image plane at q and the virtual image at a point Q. The scene point

$$p = Q + \alpha\overline{qp} \tag{3.202}$$

can be expressed as being located some distance α along the ray passing through Q, q, and p.

Analytically, we can establish a coordinate system in which the origin is at the inter-

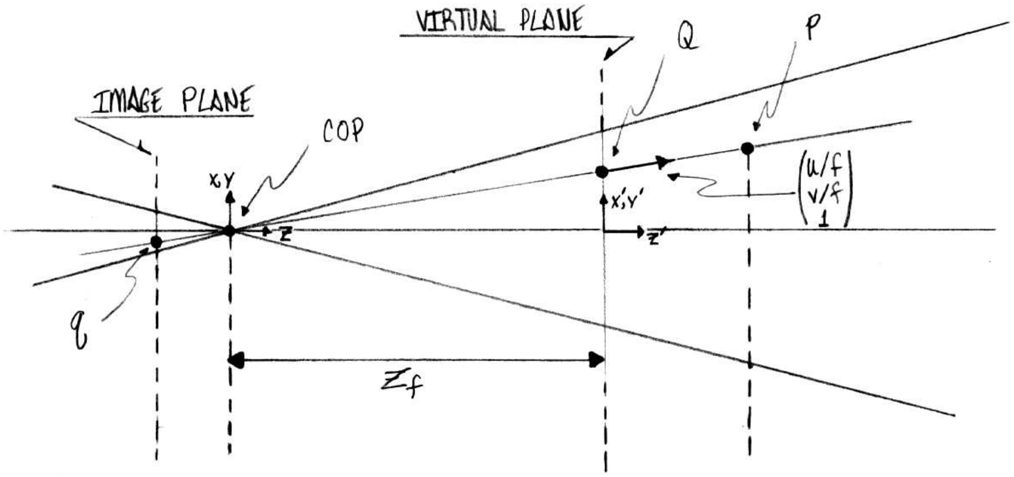


Figure 3-14: Camera pinhole model and the virtual plane coordinate system.

section of the optical axis and the virtual image plane, the XY-plane is the virtual image plane, and the Z-axis is along the optical axis. In this coordinate system, if $Q = (U, V, 0)$, the scene point

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} U \\ V \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} u/f \\ v/f \\ 1 \end{pmatrix} \tag{3.203}$$

can be described as a point plus a scaled ray. The relationship

$$\begin{pmatrix} u/f \\ v/f \end{pmatrix} = \begin{pmatrix} U/Z_f \\ V/Z_f \end{pmatrix} \tag{3.204}$$

is used to express the ray in terms of the internal camera geometry only.

The advantage of this *virtual-plane* "collinearity" interpretation over the *COP-based* "similar triangles" interpretation is that there is no necessity for a finite COP, which e.g. admits the theoretically interesting special case of *orthographic projection*, in which all rays from the scene intersect the image rectangle orthogonally. Geometrically the image-based interpretation (Fig. 3-14) is well-suited to describe orthographic projection as well as perspective projection of any focal length by simply specifying the the ray direction at each virtual image point. In the COP-based interpretation (Fig. 3-13), orthographic projection occurs only in the limit as the COP is infinitely far from both the image plane and the scene, in which the image size and scene size must both become zero!

Indeed, analytically we can see that Eqn. 3.201 becomes degenerate when $f \to \infty$ whereas Eqn. 3.203 does not.

More importantly than the theoretical interest of orthography is the practical concern of estimation with well-conditioned parameters. The COP-based model, in which the values for Z can be an order of magnitude or more different than the values for X and Y can present numerical conditioning problems. This problem is especially of concern when the value of f is unknown, because the conditioning of the problem will vary with f.

The scale issue and derivation of the camera model

Consider a physical scene point with coordinates (X, Y, Z) projecting to a physical image sensor point with coordinates $(-u, -v, -f)$ in an ideal pinhole imaging system of focal length f where the size of the image sensor is s. All the above distance values are physical distances (say, meters) and the points are in the COP-based coordinate system (see Fig. 3-13).

From Eqn. 3.201, we know the relationship

$$\frac{u}{f} = \frac{X}{Z} \tag{3.205}$$

holds. (A similar relationship holds for the v coordinate and all the following derivation applies in a parallel way to the v coordinate.)

In practice, however, we do not know the physical size of the image sensor, so we do not measure u directly. Instead, we can only measure u/s, which is an image-frame-normalized value. By dividing each parameter in the left-hand ratio by s we obtain the relationship

$$\frac{u/s}{f/s} = \frac{X}{Z} \tag{3.206}$$

in terms of the measured image coordinate u/s. The important focal parameter becomes f/s rather than f to maintain the ratio.

We also do not know the physical scale of the world and cannot hope to recover it from image geometry alone because any scaling of the world produces the same image. So, instead, we can choose a virtual image rectangle at a depth Z_f which has a finite and fixed physical size d. Regardless of what value we choose for Z_f, we can be assured that

$$d/Z_f = s/f \tag{3.207}$$

or, equivalently,

$$(Z_f/d)(s/f) = 1 \tag{3.208}$$

by geometry of similar triangles. In practice, we want to choose Z_f to be a value close to the Z values of the scene points so that d will assume a value on the same order as the size of the actual object. This can be done, e.g. by choosing $Z_f = Z_0$ where Z_0 is the depth of one of the points.

The derivation continues, however, for any choice of Z_f, leaving the question of how to set scale to the particular problem being solved. The similar triangle relationship can be written as

$$\frac{u/s}{f/s} = \frac{X/d}{Z/d} \tag{3.209}$$

where d is the size of the virtual image rectangle located at a depth Z_f.

There is a numerical problem with this equation when f/s gets large as both sides of the equation vanish. The solution to this numerical has two parts, corresponding to keeping the representation of focal parameter finite and keeping the representation of depth (Z) finite for long focal lengths ($f/s \to \infty$).

First, we would like to establish a finite focal parameter for describing long focal lengths. In particular, we would like to accommodate the range of focal lengths roughly

$$f/s \in [.3, \infty) \tag{3.210}$$

because these represent a realistic range of physical imaging parameters. Although this range of f/s is extremely large, the range of the inverse

$$s/f \in (0, 3.33] \tag{3.211}$$

is finite and relatively compact. Thus, s/f is a useful parameter for the interior orientation and the state vector for interior orientation

$$x^{(int)} = \{s/f\} \in \mathcal{M}^1_{Euc} \tag{3.212}$$

is an element of a one-parameter Euclidean space.

Second, we would like the description of scene points to remain finite for long focal lengths. To solve this, we can move the coordinate system from the COP to the virtual plane so that new plane-based coordinates

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} \doteq \begin{pmatrix} X \\ Y \\ Z - Z_f \end{pmatrix} \tag{3.213}$$

have the z-coordinate shifted by Z_f.

The coordinate frame shift results in

$$\frac{u/s}{f/s} = \frac{X/d}{Z_f/d + (Z - Z_f)/d} \tag{3.214}$$

$$= \frac{X/d}{Z_f/d + Z'/d} \tag{3.215}$$

which can be rewritten as a forward imaging equation

$$u/s = \frac{X/d}{(Z_f/d)(s/f) + ((Z - Z_f)/d)(s/f)} \tag{3.216}$$

$$= \frac{X'/d}{1 + (Z'/d)(s/f)} \tag{3.217}$$

mapping image coordinates (X', Y', Z') and focal parameter s/f to an image coordinate u/s. Here Eqn. 3.208 has been used to simplify the denominator.

If this derivation is now carried out for both image coordinates, the relationship

$$\begin{pmatrix} u/s \\ v/s \end{pmatrix} = \begin{pmatrix} X'/d \\ Y'/d \end{pmatrix} \frac{1}{1 + (Z'/d)(s/f)} \tag{3.218}$$

results.

This relationship can be used to construct a mapping

$$h^{(proj)} : \mathbb{R}^3 \times \mathbb{R}^1 \mapsto \mathbb{R}^2 \tag{3.219}$$

$$\begin{pmatrix} X'/d \\ Y'/d \\ Z'/d \end{pmatrix}, (s/f) \quad \overset{h^{(proj)}}{\mapsto} \quad \begin{pmatrix} u/s \\ v/s \end{pmatrix} \tag{3.220}$$

of scene point parameters and focal parameter to image point parameters.

Alternatively, it represents a mapping

$$h^{(proj')} : \mathbb{R}^3 \mapsto \mathbb{R}^2 \tag{3.221}$$

$$\begin{pmatrix} X'/d \\ Y'/d \\ (Z'/d)(s/f) \end{pmatrix} \overset{h^{(proj')}}{\mapsto} \begin{pmatrix} u/s \\ v/s \end{pmatrix} \tag{3.222}$$

of projectively scaled scene point parameters to image point parameters.

The motivation for the second alternative is associated with the estimation properties when the focal length is unknown, as discussed below.

Degeneracy of Jacobian at long focal lengths

To see the degeneracy, consider the constraint equation Eqn. 3.219 and its Jacobian matrix

$$\frac{\partial h^{(proj)}(x)}{\partial x} = \begin{pmatrix} \frac{1}{D} & 0 & \frac{-(X/d)(s/f)}{D^2} & \frac{-(X/d)(Z'/d)}{D^2} \\ 0 & \frac{1}{D} & \frac{-(Y/d)(s/f)}{D^2} & \frac{-(Y/d)(Z'/d)}{D^2} \end{pmatrix} \tag{3.223}$$

where

$$D = 1 + (Z'/d)(s/f) \tag{3.224}$$

is the denominator.

For long focal lengths, or orthographic imaging, the third column vanishes for all points resulting in rank-deficient Jacobian. As discussed in Section 3.6, this will make the intermediate parameter Z'/d extremely sensitive to noise at long focal lengths.

For the alternative model, Eqn. 3.221, the Jacobian becomes

$$\frac{\partial h^{(proj')}(x)}{\partial x} = \begin{pmatrix} \frac{1}{D} & 0 & \frac{-(X/d)}{D^2} \\ 0 & \frac{1}{D} & \frac{-(Y/d)}{D^2} \end{pmatrix} \tag{3.225}$$

and the degeneracy disappears. Thus, for estimation with an unknown focal length, the intermediate parameters in the domain of $h^{(proj')}$ will be used in combination with the relative orientation parameters and forward function. This is discussed further in Section 3.7.4 and Section 3.7.5

3.7.4 Camera model—relative orientation

The relative orientation is a parameterization for the physical concept of the displacement and orientation of one camera with respect to another camera (in the case of stereoscopy), or with respect to itself at a different time (in the case of motion). The relative orientation parameters correspond to displacement and rotation of one 3-D reference frame with respect to another.

The relative orientation of two cameras consists of a physical translation and rotation of virtual plane reference frames of the two cameras.

Using the forward functions based on translational and rotational parameterization developed in Section 3.7.1 and Section 3.7.2, the composite relative orientation can be described as

$$p_A = h^{(tra)}(T, h^{(rot)}(q, p_B)) \tag{3.226}$$

$$\begin{pmatrix} X_A \\ Y_A \\ Z_A \end{pmatrix} = \begin{pmatrix} T_X \\ T_Y \\ T_Z \end{pmatrix} + \begin{pmatrix} & & \\ & R(q) & \\ & & \end{pmatrix} \begin{pmatrix} X_B \\ Y_B \\ Z_B \end{pmatrix} \tag{3.227}$$

indicating the transformation of a 3-D point p_B from one virtual plane reference frame "B" to another virtual plane reference frame "A" where $(T_X, T_Y, T_Z) \equiv x^{(tra)}$ and $q \equiv x^{(rot)}$ and $R(q)$ is the rotation matrix corresponding to the unit quaternion as in Eqn. A.48.

If we choose the same virtual plane depth Z_f for each of the two camera reference frames we get the same scalar d in which to parameterize the coordinates of both p_A and p_B resulting in the relationship

$$\begin{pmatrix} X_A/d \\ Y_A/d \\ Z_A/d \end{pmatrix} = \begin{pmatrix} T_X/d \\ T_Y/d \\ T_Z/d \end{pmatrix} + \begin{pmatrix} & & \\ & R(q) & \\ & & \end{pmatrix} \begin{pmatrix} X_B/d \\ Y_B/d \\ Z_B/d \end{pmatrix} \tag{3.228}$$

which requires a translation scaling of the same degree.

We will want to use this function in conjunction with the interior orientation model discussed in Section 3.7.3 and thus the parameters on the left should be in the domain of the function $h^{(proj')}$ (Eqn. 3.221). To this end, the equation can be rewritten

$$\begin{pmatrix} X_A/d \\ Y_A/d \\ (Z_A/d)(s/f) \end{pmatrix} = \begin{pmatrix} T_X/d \\ T_Y/d \\ (T_Z/d)(s/f) \end{pmatrix} + \begin{pmatrix} 1 & & \\ & 1 & \\ & & s/f \end{pmatrix} \begin{pmatrix} & & \\ & R(q) & \\ & & \end{pmatrix} \begin{pmatrix} X_B/d \\ Y_B/d \\ Z_B/d \end{pmatrix} \tag{3.229}$$

and thus the parameters $x^{(rel)}$ of relative orientation become

$$\begin{aligned} \left.\begin{matrix} T_X/d \\ T_Y/d \\ (T_Z/d)(s/f) \end{matrix}\right\} &= \text{translation (projectively scaled)} \\ \left.\begin{matrix} q_0 \\ q_X \\ q_Y \\ q_Z \end{matrix}\right\} &= \text{rotation (unit quaternion)} \\ s/f &= \text{focal parameter} \end{aligned} \tag{3.230}$$

and the relationship Eqn. 3.229

$$h^{(rel)} : x^{(rel)} \times \mathbb{R}^3 \mapsto \mathbb{R}^3 \tag{3.231}$$

can be combined with the projection relationship Eqn. 3.221 to obtain a composite function for the projection of a 3-D point in one camera frame into the image of another camera frame.

3.7.5 Composite camera model

We can combine the parameterizations discussed in Section 3.7.3 and Section 3.7.4 into a composite camera model that describes both relative and internal orientation and is a well-behaved parameterization when focal length is unknown.

The following specification

$\mathcal{M}^{(cam)}$:

$$\begin{array}{rcl}
\mathcal{P}_S^{(cam)} & = & \mathbb{R}^8 \\
\mathcal{S}^{(cam)} \subseteq \mathcal{P}_S^{(cam)} & = & \mathbb{R}^4 \times \mathcal{S}^{(rot)} \\
\mathcal{P}_T^{(cam)}(x), x \in \mathcal{S}^{(cam)} & = & \mathbb{R}^7 \\
\mathcal{T}^{(cam)}(x), x \in \mathcal{S}^{(cam)} & = & \mathbb{R}^4 \times \mathcal{T}^{(rot)}(x) \\
\mathcal{Z}^{(cam)}(x), x \in \mathcal{S}^{(cam)} & = & I_4 \times \mathcal{Z}^{(rot)} \\
\mathcal{A}^{(cam)} & = & +_4 \times \mathcal{A}^{(rot)}
\end{array} \tag{3.232}$$

summarizes the manifold-tangent parameterization $\mathcal{M}^{(cam)}$. The notation $+_4$ indicates Euclidean vector addition in $\mathbb{R}^4$.

The state parameters of $x^{(cam)}$ can be summarized as

$$\begin{array}{rcl}
s/f & = & \text{interior orientation} \\
\left.\begin{array}{r} T_X/d \\ T_Y/d \\ (T_Z/d)(s/f) \end{array}\right\} & = & \text{relative orientation, translation} \\
\left.\begin{array}{r} q_0 \\ q_X \\ q_Y \\ q_Z \end{array}\right\} & = & \text{relative orientation, rotation}
\end{array} \tag{3.233}$$

where the first four parameters form a 4-D Euclidean space and the last four parameters form a rotation manifold, as modeled in Section 3.7.2.

The associated composite forward mapping from scene parameters in one frame to image parameters in another

$$h^{(cam)} : x^{(cam)} \times \mathbb{R}^3 \mapsto \mathbb{R}^2 \tag{3.234}$$

is the composite

$$h^{(cam)} = h^{(proj')} \circ h^{(rel)} \tag{3.235}$$

of the projection function $h^{(proj')}$ (Eqn. 3.221) and the relative orientation function $h^{(rel)}$ (Eqn. 3.231).

Chapter 4

Point Geometry

This chapter develops an image-based point geometry model that is motivated by and well-suited to the problem of "structure-from-motion". More precisely, the idea is that an image sequence is taken from a moving camera (the "motion") and the 3-D geometry (the "structure") is obtained by estimating camera interior and relative parameters and 3-D locations of points.

The first section treats the modeling of the pointwise geometry and combines this with the camera model developed in Section 3.7.5 to obtain a complete estimation model for the problem.

Various aspects of the performance, in various regimes of state space, are systematically evaluated by a set of computer simulation experiments in the subsequent two sections.

The final three sections describe three applications based on the estimation model and present a realistic picture of the utility of the fundamental technique and the quality of performance in the field.

4.1 Modeling

4.1.1 Image-based point structure

An image-based structure model is appropriate to the paradigm of motion-based estimation, in which feature points are selected in the first image and tracked in subsequent images. The first image is the structure reference frame and the unknown parameters are the depths of the selected points in this reference image.

Specifically, each 3-D point p_i is represented in the virtual-plane coordinate system (see Section 3.7.3) of the reference image by a depth parameter Z_i/d, where

$$Z_i \doteq Z_i^{(COP)} - Z_f^{(COP)} \tag{4.1}$$

is the distance from the virtual reference plane to the point, $Z_f^{(COP)}$ is the COP-centered depth of the plane, $Z_i^{(COP)}$ is the COP-centered depth of the point, and

$$d = Z_f^{(COP)}(s/f) \tag{4.2}$$

is the induced dimension of the virtual image rectangle (see Fig. 4-1).

The virtual plane $Z_f^{(COP)}$ is chosen by fixing the depth of one of the points, say $i = 1$,

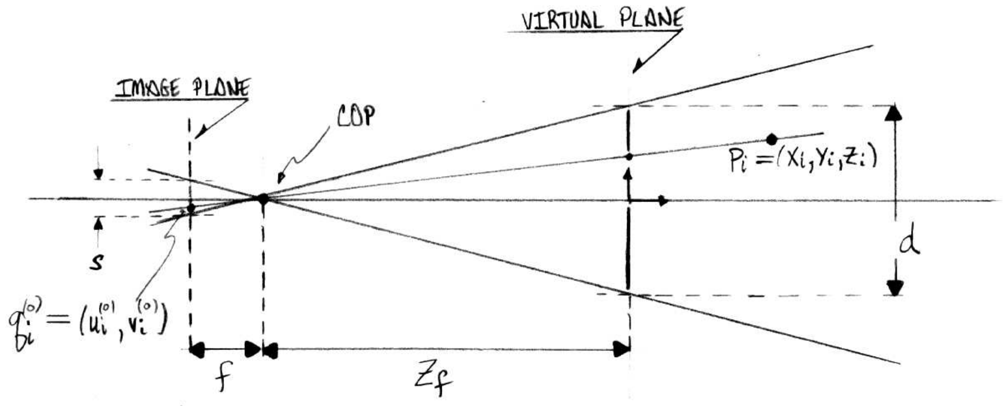


Figure 4-1: Image-based pointwise structure model.

to be zero, i.e. let

$$Z_1^{(COP)} - Z_f^{(COP)} = 0 \tag{4.3}$$

hold for p_1.

The 3-D coordinates of point p_i can be expressed in terms of the unknown depth parameter Z_i/d as

$$\begin{pmatrix} X_i/d \\ Y_i/d \\ Z_i/d \end{pmatrix} = \begin{pmatrix} (1 + (Z_i/d)(s/f))(u_i^{(0)}/s) \\ (1 + (Z_i/d)(s/f))(v_i^{(0)}/s) \\ (Z_i/d) \end{pmatrix} \tag{4.4}$$

which follows directly from the forward projection equation (Eqn. 3.218) where

$$q_i^{(0)} \doteq \begin{pmatrix} u_i^{(0)}/s \\ v_i^{(0)}/s \end{pmatrix} \tag{4.5}$$

is the selected image location that defines p_i.

The expression in Eqn. 4.4 can be interpreted as a *back-projection* mapping

$$h^{(back)} : \mathbb{R}^2 \times \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}^3 \tag{4.6}$$

or

$$\begin{pmatrix} u_i^{(0)}/s \\ v_i^{(0)}/s \end{pmatrix} \times (s/f) \times (Z_i/d) \quad \overset{h^{(back)}}{\mapsto} \quad \begin{pmatrix} X_i/d \\ Y_i/d \\ Z_i/d \end{pmatrix} \tag{4.7}$$

from a selected 2-D image location to a 3-D point in the reference image frame. The mapping depends on the internal orientation of the reference camera and the depth of the point in the reference frame, both of which are typically unknown and have to be estimated along with the motion parameters.

If we model each point p_i, then, with the parameterization

$$x^{(3pt)} = \{Z_i/d\} \in \mathcal{M}^1_{Euc} \tag{4.8}$$

which is a single Euclidean parameter, a composite mapping can be made between the defining reference image location $q_i^{(0)}$ and the projection $q_i(t)$ in the camera image at time t

$$q_i(t) = h^{(cam)}\left(x^{(cam)}(t), h^{(back)}\left(q_i^{(0)}, x^{(int)}, x^{(3pt)}\right)\right) \tag{4.9}$$

which is a function of the interior orientation of the reference image $x^{(int)}$, the 3-D point $x^{(3pt)}$, and the camera parameters of the current camera $x^{(cam)}(t)$. The function $h^{(cam)}$ is defined in Eqn. 3.234 and $x^{(cam)}(t)$ in Eqn. 3.233.

We define the composite mapping of Eqn. 4.9

$$h^{(pt)} : \mathbb{R}^2 \times \mathcal{P}_S^{(int)} \times \mathcal{P}_S^{(cam)} \times \mathcal{P}_S^{(3pt)} \mapsto \mathbb{R}^2 \tag{4.10}$$

as the point correspondence mapping for a single point. It maps the reference image point to its location in the current frame as a function of the interior orientation of both cameras, the relative orientation and the depth of the point.

4.1.2 Complete model for motion-based point geometry

For the motion-based geometry estimation problem, we are interested in solving the 3-D geometry for the case in which N points are selected in the first image and tracked through an arbitrarily long image sequence.

To do this with the GEKF algorithm for this points problem, we require a state model $\mathcal{M}^{(pts)}$, an external constraint model $e(x(t), y(t))$, an internal dynamic model $i(x(t), x(t+1))$, and an error model consisting of error characterization of initial knowledge $P(t|t-1)$, measurement error $R(t)$, constraint modeling error $S(t)$, and dynamic modeling error $Q(t)$.

State model

The state model for this problem

$$\mathcal{M}^{(pts)} = \mathcal{M}^{(cam)} \times \prod_{i=1}^{N} \mathcal{M}^{(3pt)} \tag{4.11}$$

consists of a camera state and N instances of the 3-D image-based point model discussed above.

The state parameters

$$\begin{aligned}
s/f &= \text{interior orientation} \\
\left.\begin{array}{r} T_X/d \\ T_Y/d \\ (T_Z/d)(s/f) \end{array}\right\} &= \text{relative orientation, translation} \\
\left.\begin{array}{r} q_0 \\ q_X \\ q_Y \\ q_Z \end{array}\right\} &= \text{relative orientation, rotation} \\
Z_1/d &= \text{structure point 1} \\
&\vdots \\
Z_N/d &= \text{structure point N}
\end{aligned} \tag{4.12}$$

collectively form the state vector $x^{(pts)} \in \mathbb{R}^{8+N}$.

The manifold-tangent model specification

$\mathcal{M}^{(pts)}$:

$$\begin{aligned}
\mathcal{P}_S^{(pts)} &= \mathbb{R}^{8+N} \\
\mathcal{S}^{(pts)} \subseteq \mathcal{P}_S^{(pts)} &= \mathbb{R}^{4+N} \times \mathcal{S}^{(rot)} \\
\mathcal{P}_T^{(pts)}(x), x \in \mathcal{S}^{(pts)} &= \mathbb{R}^{7+N} \\
\mathcal{T}^{(pts)}(x), x \in \mathcal{S}^{(pts)} &= \mathbb{R}^{4+N} \times \mathcal{T}^{(rot)}(x) \\
\mathcal{Z}^{(pts)}(x), x \in \mathcal{S}^{(pts)} &= I_{4+N} \times \mathcal{Z}^{(rot)} \\
\mathcal{A}^{(pts)} &= +_{4+N} \times \mathcal{A}^{(rot)}
\end{aligned} \tag{4.13}$$

summarizes the state space manifold.

[Note: technically the parameter s/f is a non-negative ratio and thus the state space could be further restricted to only non-negative reals for that parameter.]

Constraint model

A constraint model can be built on the composite forward model for the set of N points

$$h^{(pts)} : \mathcal{P}_S^{(pts)} \times \mathbb{R}^{2N} \mapsto \mathbb{R}^{2N} \tag{4.14}$$

which maps the state parameters $x^{(pts)}$ and a vector $q^{(0)}$ of defining reference image locations to the set of image locations $q(t)$ in the current frame.

Symbolically,

$$q(t) = h^{(pts)}(x^{(pts)}(t), q^{(0)}) \tag{4.15}$$

which can be broken down into a set of parallel single point relationships

$$q_1(t) = h^{(pt)}(x^{(pts)}, q_1^{(0)}) \tag{4.16}$$

$$\vdots \tag{4.17}$$

$$q_N(t) = h^{(pt)}(x^{(pts)}, q_N^{(0)}) \tag{4.18}$$

where $q_i(t) \in \mathbb{R}^2$ represents one of the N feature measurements and $h^{(pt)}$ is from Eqn. 4.10.

Since we have a forward model, the most appropriate composite constraint is then the error vector

$$e^{(pts)}(x^{(pts)}(t), y^{(pts)}(t)) = h^{(pts)}(x^{(pts)}(t), q^{(0)}) - y^{(pts)}(t) \tag{4.19}$$

where $y^{(pts)}(t)$ represents the measurement at time t of the feature point locations $q(t)$.

The internal constraint function is based on an identity state transition function, i.e.

$$\begin{aligned}
x(t+1) &= f(x(t)) && (4.20) \\
&= x(t) && (4.21)
\end{aligned}$$

which is exact for the static parameters of $x^{(cam)}$ and $x^{(str)}$ and expresses temporal coherence for the dynamic parameters. The extent of the temporal coherence is dictated by the variances in the $Q(t)$ matrix discussed below.

Error model

The final requirement for the Kalman filter implementation using the GEKF algorithm, is to specify the various error characteristics of the model.

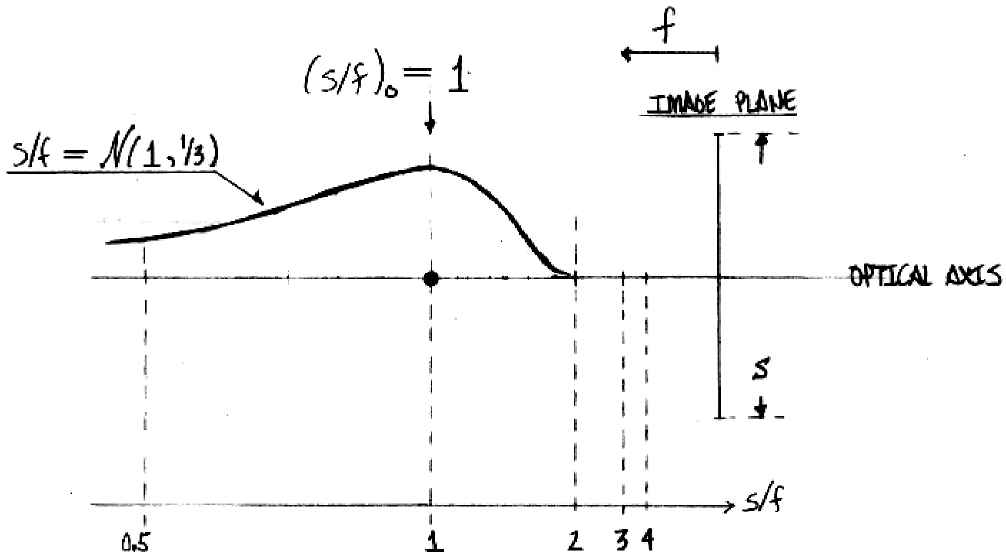


Figure 4-2: Camera error model. A Gaussian distribution in (s/f) space with a standard deviation equal to one-third the focal parameter gives a distribution on focal length like the one above. Here $(s/f)_0 = 1$.

In any practical application, knowledge of the domain may allow better estimates of error parameters, but in general, with the scaled distance parameters and the virtual plane coordinate system of the model $\mathcal{M}^{(pts)}$, a quite generic set of error variances apply to a wide range of imagery.

First, we must specify the initial state

$$x(t|t-1) = \mathcal{N}(\hat{x}(t|t-1), P(t|t-1)) \tag{4.22}$$

where x can be partioned into four parts corresponding to interior orientation, relative translation, relative rotation, and point depths

$$\hat{x}(t|t-1) = \begin{pmatrix} (s/f)_0 \\ 0_{(3)} \\ q_I \\ 0_{(N)} \end{pmatrix} \tag{4.23}$$

and the error covariance matrix $P(t|t-1)$

$$P = \begin{pmatrix} P^{(int)}_{(1\times1)} & & & \\ & P^{(tra)}_{(3\times3)} & & \\ & & P^{(rot)}_{(3\times3)} & \\ & & & P^{(pt)}_{(N\times N)} \end{pmatrix} \tag{4.24}$$

can be partitioned into four corresponding blocks on the diagonal. The notation

$$q_I = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{4.25}$$

is the identity unit quaternion indicating no initial rotation and $(s/f)_0$ is an initial focal length estimate.

The values of the error covariance matrix P can be set quite generically as

$$\begin{aligned} P^{(int)} &= (\frac{1}{3}(s/f)_0)^2 && (4.26) \\ P^{(tra)} &= 0_{(3\times3)} && (4.27) \\ P^{(rot)} &= 0_{(3\times3)} && (4.28) \\ P^{(pt)} &= \begin{pmatrix} 0 & \\ & \sigma^2_{depth} I_{(N-1\times N-1)} \end{pmatrix} && (4.29) \end{aligned}$$

where $\sigma_{depth} \in [2, 10]$, depending on what is known of depth of field.

These values express that the focal length parameter is between orthographic and twice the initial value with 98% certainty, that the initial motion is known to be exactly zero, and that the depth of each point $p_i, i \neq 1$ is not known well at all, except that it is unlikely to be greater than σ_{depth} times the field of view away from the virtual reference plane. There are very few scenes in practice where the depth of the object is greater than five or ten times its depth.

Recall from Section 4.1.1 that the reference plane is physically defined by setting $Z_1/d = 0$, which is why the variance for p_1 is exactly zero.

The internal constraint covariance matrix $Q(t)$ can also be set quite generically as

$$\begin{aligned} Q^{(int)} &= 0 && (4.30) \\ Q^{(tra)} &= (\frac{1}{3}(\text{MAXVEL})\Delta t)^2 I && (4.31) \\ Q^{(rot)} &= (\frac{1}{3}(\text{MAXROT})\Delta t)^2 I && (4.32) \\ Q^{(pt)} &= 0, \forall i = 1 \dots N && (4.33) \end{aligned}$$

where MAXVEL is the maximum translational velocity expected for the sequence, MAXROT is the maximum rotational velocity expected, and Δt is the interframe time interval.

Coupled with an identity state transition function

$$f(x(t)) = x(t) \tag{4.34}$$

this choice of values expresses that the focal parameter of the camera and the location of the points in the reference frame are static parameters and that the velocities of the translation and rotation are not likely to be larger than certain limits MAXVEL and MAXROT.

Finally, the measurement covariance matrix $R(t)$ can be set up as

$$R(t) = \sigma^2_{image} I \tag{4.35}$$

where σ^2_{image} is the measurement noise variance for each measured coordinate and can be set as

$$\sigma_{image} = \frac{1}{3}\left(\frac{\text{PIXACC}}{\text{PIXRES}}\right) \tag{4.36}$$

and PIXACC expresses the coordinate accuracy in terms of maximum (98% probability) error in pixels, and PIXRES is the resolution of the image in pixels, i.e. s measured in pixels. For subpixel accuracy tracking in video resolution imagery, $\sigma_{image} = .0005$. A more realistic value in noisy video is $\sigma_{image} = .0015$.

The external constraint modeling error $S(t)$ can be set to zero or to a small diagonal matrix to augment the constraint error induced by $R(t)$. The only practical reason for adjusting S independent of R is if lens distortion moves the model considerably away from a pinhole projection. But in this case, S should depend on the image location of the measurement (higher variance further out), or the projection model should be replaced to introduce lens distortion parameters.

4.1.3 Comparison to traditional nonlinear models

The computational model developed above arises from a particular way of looking at and understanding the physical geometrical problem. Other viewpoints can lead to other nonlinear formulations. All of them are geometrically correct, but some may perform better numerically.

The most similar nonlinear modeling approaches are found in several batch estimation approaches [8, 72, 41, 81, 16] and recursive approaches [14, 5]. There are significant differences between these and the proposed model, each of which is important for different reasons.

Camera model

Our camera model is based on a nondimensional parameter s/f and the virtual plane coordinate system described in Section 3.7.3. Other than [72], which uses a similar camera model to ours, all other formulations assume the camera focal length f is known relative to s and use a coordinate system based at the COP.

The motivation behind using the virtual plane coordinate system and the inverted ratio of focal length to image size came from the desire to estimate focal length in uncalibrated imagery and to allow for the widest range of realistic focal parameters. The details behind why the traditional model is inappropriate are discussed in Section 3.7.3.

Since our model addresses a more general class of problem, no direct comparison can be made. In the case of known focal length, the effects on computational performance are probably small. But to address the more general problem, a straightforward generalization of previous formulations (i.e. using f as a parameter) will encounter numerical problems in certain ranges of state space, particularly for long focal lengths.

Even if it is known, however, that the focal length is not long, the arbitrary choice of scaling that is required by dimensional representations of lengths can cause poor and uneven conditioning throughout the state space. For example, a popular choice of arbitrary scaling is to choose $f = Z_f = 1$, $s = d$, and to set d somehow by fixing the coordinates of a point or the centroid of points. Scaling issues may be part of the problem in the poor numerical estimation performance exhibited by earlier formulations.

Translation model

Our translational model is based on nondimensional parameters scaled to a distance d that is meaningful to the scene. In particular, it is the width of the field of view of the reference frame camera at the depth of one of the scene points. Therefore it is a distance on the order of the size of the scene and one can be sure that, regardless of the physical scale of the scene, the translational magnitude will usually be less than 2 and almost certainly less than 10 for any realistic sequence containing a fixed set of points.

Additionally, the z-coordinate of the relative translation is further scaled to the focal parameter of the camera. This is important for long focal length estimation because of degeneracies that arise in this case, which is discussed in detail in Section 3.7.3.

For most other models, the camera translational parameters are a scaled representation of the vector between the COPs of two camera stations. The degeneracy at long focal lengths is that this vector will become infinite in length. When the focal length is known to be small, the difference in parameterization is probably not significant.

Rotation model

Our rotational model is a manifold-tangent model based on the unit quaternion manifold and its tangent hyperplanes.

Other models use the quaternion but generally use Euclidean-based estimation techniques. Batch approaches [72, 41, 81] use the Levenberg-Marquardt method or conjugate gradient [16], while recursive approaches [14] use the EKF, none of which were designed for curved spaces.

In [14], based on the EKF, a set of constant velocity parameters are in the state rather than the quaternion, so the space is Euclidean, but the formulation only applies to constant velocity.

In [16], based on conjugate gradient, the entire quaternion is in the state vector and no fixed motion constraint is imposed. However, the 4-D rotation parameter subspace is assumed to be Euclidean at each step and the quaternion is normalized after each iteration to pull the state estimate back onto the rotation manifold. The authors propose that such normalization will have little effect, but produce results which exhibit particularly curious oscillating behavior in the rotation parameters.

A possibility is that this apparent "limit cycling" may be partially due to the fact that the estimator is allowed to choose a non-physical set of parameters; the normalization will produce a physical set of parameters on the rotation manifold, but since the other state parameters are not corrected, they will remain optimized to the inconsistent non-physical rotation parameters. An alternative explanation of the phenomenon by the authors, which is not inconsistent with the above, is that the oscillating behaviour is probably due to trading off of errors between the rotation parameters and the slowly converging structure parameters. Either or both explanations could be true.

The batch approaches [72, 81, 41] use the Levenberg-Marquardt method, which is based on a Euclidean state space and, in the absence of any discussions of manifold modeling, appear to use the same post-estimation normalization technique. Although no catastrophic problems were reported, the practice of estimating rotation in a 4-D Euclidean space and then normalizing the quaternion is at best a needless inefficiency (because the excursions off the state manifold and back will probably not be the most efficient route to the solution), and is at worst a destabilizing force that can cause the solution to be extremely fragile.

The manifold-tangent model guarantees that the estimated state will be near the manifold and physically meaningful, as long as the step size remains bounded. The region-of-trust bound can be determined (it is $|\omega| = 2$) and enforced.

Scene-based point structure

Probably the most computationally significant difference from previous formulations is the image-based representation of points. All the formulations discussed above use what might be called a scene-based representation.

Philosophically, the image-based model arises naturally from the scenario that the vision system will choose a set of points in the first image, and, based on appearance, track the points through subsequent images and use the feature track information to recover 3-D parameters. Thus, features are defined by their image locations in the reference frame and the only unknown spatial parameter in the reference frame is their depths.

The alternative approach is a scene-based representation in which the structure model parameterizes the 3-D points relative to some arbitrary 3-D reference frame and treats each image as a set of measurements of the 3-D points.

The 3-D location of a point can be expressed as a 3-vector

$$x^{(3pt)} = (X_o, Y_o, Z_o) \tag{4.37}$$

in some fixed scene-centered coordinate system, resulting in a simple Euclidean state model

$$\mathcal{M}^{(3pt)} \equiv \mathcal{M}^3_{Euc} \tag{4.38}$$

for a single point, with the above parameters. A composite model

$$\mathcal{M}^{(str)} = \prod_{i=1}^{N} \mathcal{M}^{(3pt:i)} \equiv \mathcal{M}^{3N}_{Euc} \tag{4.39}$$

for a set of N points is simply a set of $3N$ parameters representing the 3-D coordinates of each of the points.

Although geometrically equivalent, this representation is burdened with many extra parameters which apparently cause slow convergence and stability problems.

In recursive systems, this parameterization results in a column-deficient Jacobian which makes the success of the estimator highly dependent upon initial conditions and initial specification of the state error and measurement statistics P and R.

In batch systems, this parameterization can lead to a bloated parameter space which is difficult to navigate and expensive to compute in. In [72], it is reported that convergence fails for an overly large number of frames without a decent initial condition leading to an ad hoc strategy to combine estimates based on smaller batches of frames.

The bloat problem is well-illustrated by considering a long sequence of 1700 frames used in one of our applications (see Section 4.6). A total of around 75 features are used with an average of over 10 feature points per frame, leading to a system with over 34,000 measurements and over 10,000 states. Whereas a batch formulation is intractable—the matrices cannot even be stored without special sparse matrix representations—the recursive formulation solves all 1700 frames of the same problem, with an average of 20 measurements per frame and 17 states per frame, in a matter of seconds.

Relationship between image- and scene-based models

The scene-based model has three parameters per feature point, corresponding to the three 3-D coordinates, whereas the image-based model has only one parameter per feature point, corresponding to the depth of the point in the reference image frame.

Although both interpretations assume zero-mean measurements, the different interpretations lead to widely disparate number of parameters. The question is: how can the same physical problem be represented by two sets of parameters with widely different dimensionality?

One answer is that the two interpretations are based on different assumptions regarding feature tracking. The image-based model defines each point relative to the reference image frame and assumes it is tracked with zero-mean error. The scene-based model defines each point relative to a scene-based reference frame and assumes it is tracked with zero-mean error.

The first is a rational assumption for causal sequences in which feature tracking is based on appearance of a feature template taken from the reference image. Thus it is natural to assume that tracking errors of the selected image point will be zero mean.

The second is a rational assumption for an arbitrary batch of images in which feature tracking errors are based on some external model of what the feature looks like. For example, in [5] a known 3-D model of a black and white target is used as the template for tracking rather than a template taken from an image.

The typical complaint about the first model is that not all the images are treated equally. The final result will have zero residual in the reference image and finite residual in all other images. Therefore, if a different image were chosen as the reference image, the result would be slightly different. With the scene-based model, all the images are treated equally and will have a finite residual.

Indeed, all the images are not treated equally, and for a good reason. The reference image is the one in which feature points are defined and the template is extracted for tracking. It is generally the first of a causally related sequence of images. This model most accurately reflects the actual process of feature selection and tracking in image sequences as there is rarely a 3-D model of the object available (and if there were, one would not need to estimate structure anyway).

One particular instance in which one could argue that a 3-D model is implicitly used for tracking and that the scene-based model of tracking errors is therefore more appropriate is when the feature measurements are generated by a person looking at the images, as is usually the case in photogrammetry. In this case, it is arguable that a symbolic entity in the scene is being matched to the image and one should interpret that the image point selection in the first image is actually a *measurement* of what the person believes is the image location of the symbolic scene entity rather than the true location of the scene entity. This is in contrast to the typical computerized feature selection process in which the image signal of the first image is used to find candidate features for tracking, which may or may not correspond to meaningful scene points, but are represented completely by their appearance.

An example is tracking the corner of a building. If a person is doing the tracking, he might locate the corner of the building with some error in the first image, but subsequently his goal will always be to locate the corner of the building, so one can assume that overall, including the first image, the errors are random and zero-mean. If a computer is doing the tracking, however, the assumption is that if a point off the actual corner was selected in the first image that, based on the given template, subsequent feature tracking will precisely locate the same point off the actual corner and will have no reason to bias itself toward the actual corner because it does not know what a corner is; the only defining characteristic of the feature is the original image template.

(Of course, real feature tracking, especially with a lot of 3-D rotation in the scene, is not necessarily all that true to the original point, but there is generally no better assumption as to which way it will err, so on average a zero-mean assumption is the best one can do without a 3-D model of the scene.)

Whatever the assumptions about feature tracking, it may be desirable for some reason to obtain a solution equivalent to the scene-based model in which a finite residual results for each image in the collection. In this way, the relationship between the seemingly disparate representations can be understood and a new representation equivalent to the traditional scene-based model will result but with a new parameterization that makes it much easier to use.

To arrive at the new representation we can alter Eqn. 4.4 replacing the feature point specification $q^{(0)}$ with a measurement $\hat{q}^{(0)}$ which has some error b in it. Thus, the 3-D coordinates of point p can be expressed in terms of the unknown depth parameter Z/d and the image-plane measurement errors $(b_u/s, b_v/s)$ as

$$\begin{pmatrix} X/d \\ Y/d \\ Z/d \end{pmatrix} = \begin{pmatrix} (1 + (Z/d)(s/f))(\hat{u}^{(0)}/s + b_u/s) \\ (1 + (Z/d)(s/f))(\hat{v}^{(0)}/s + b_v/s) \\ (Z/d) \end{pmatrix} \tag{4.40}$$

where

$$q^{(0)} = \hat{q}^{(0)} + b \tag{4.41}$$

and

$$b = \begin{pmatrix} b_u \\ b_v \end{pmatrix} \tag{4.42}$$

is the bias on the original feature.

The unknown parameters form the component state vector

$$x^{(3pt)} = \begin{pmatrix} b_u/s \\ b_v/s \\ Z/d \end{pmatrix} \in \mathcal{M}^3_{Euc} \tag{4.43}$$

which is equivalent to the scene-based representation. The inversion of Eqn. 4.40 is

$$\begin{pmatrix} b_u/s \\ b_v/s \\ Z/d \end{pmatrix} = \begin{pmatrix} \dfrac{X/d}{1 + (Z/d)(s/f)} - \hat{u} \\ \dfrac{Y/d}{1 + (Z/d)(s/f)} - \hat{v} \\ Z/d \end{pmatrix} \tag{4.44}$$

which expresses that the biases are the difference between the true point projections and the measured ones.

The representation is easier to use than a parameterization relative to an arbitrary 3-D reference frame because the parameterization is based in the virtual plane coordinate system and represents physically meaningful quantities for which it is straightforward to assign initial error statistics. For example, the bias error is similar to the image-plane tracking error and the depth error can be placed at a large value to indicate no knowledge.

The representation also reduces straightfowardly to the unbiased image-based structure representation of Section 4.1.1 by setting the value of the bias to zero and reducing the uncertainty in the bias to zero.

4.1.4 Comparison to linear models

Using the modeling framework of Chapter 3 and the error analysis techniques discussed in Section 3.6, the numerical performance of nonlinear modeling can be compared directly to the numerical performance of a linear model based on the same geometry.

Fig. 4-3 depicts the 3-D geometry used to compare the two types of models. The geometry reflects a typical geometry of a wide baseline stereo camera system, such as that used in the person tracking applications discussed in Chapter 5. The view is intended to be of the XZ-plane, where the baseline is along the X-axis and there is zero displacement along the Y-axis and Z-axis. The cameras are verged to view the same volume, about 1.7m in depth, by rotating around the Y-axis only. There is no rotation around the X-axis or Z-axis.

The experiment is intended to plainly illustrate the deficiency of the linear constraints by applying a probabilistic-based model analysis. Since the linear constraints depend upon

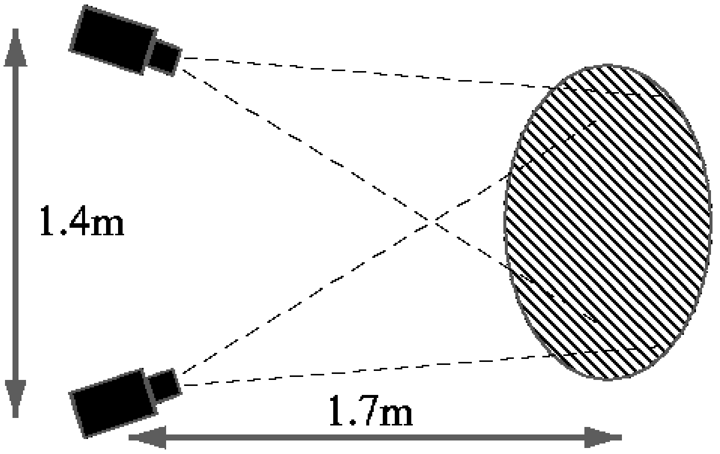


Figure 4-3: Geometry used for comparing linear and nonlinear numerical models. This diagram depicts a view of the XZ-plane, with the 1.4m baseline being in the X-direction and the 1.7m depth of the scene being in the Z-direction. Camera rotation is around the Y-axis only. The shaded area represents the view volume that is visible to both cameras.

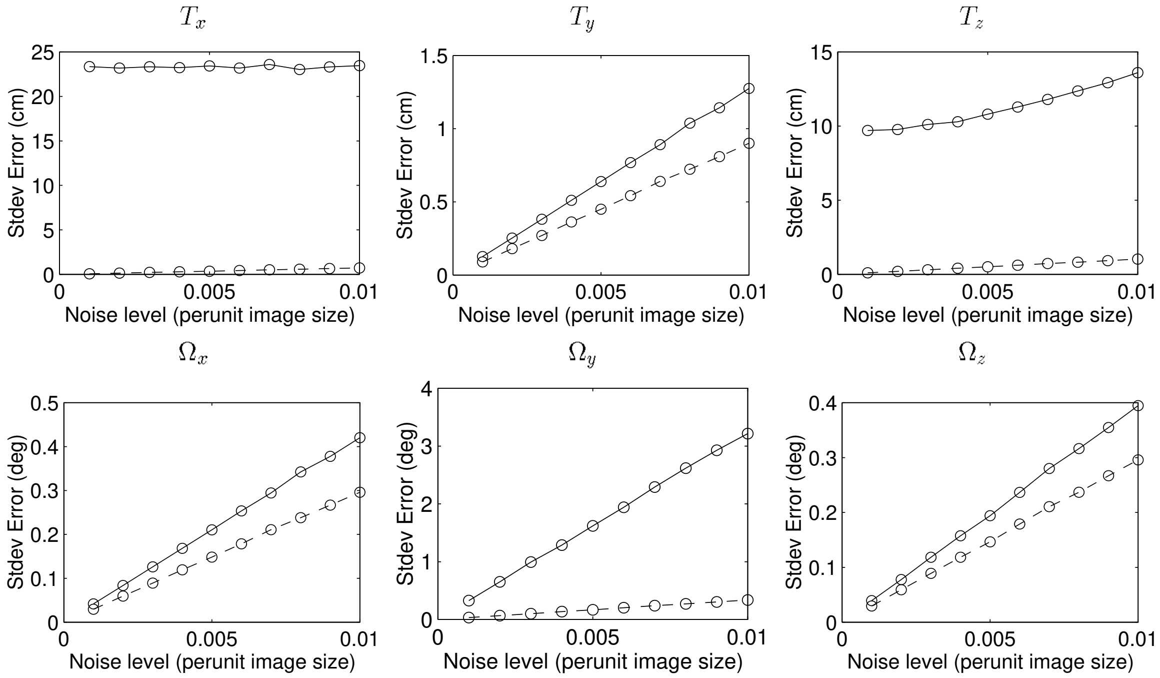


Figure 4-4: Precision analysis comparing linear models and nonlinear models. Error as a function of noise level. The graphed values represent average over all trials of the minimum RMS error achievable on each 3-D parameter given the trial scene structure and the given level of noise on the measurements.

constraining corresponding points only to an epipolar plane, as described in Section 2.2.3, and they cannot apply further constraint within the epipolar plane, we should expect measurement errors within epipolar planes to go unpenalized in any optimization procedure that uses the linear constraints only.

For example, in the geometry of Fig. 4-3, all epipolar planes will be close to, but not exactly, parallel to the XZ-plane. Therefore, we should expect the linear constraints to impose good translational constraint of the camera geometry in the Y-direction and good orientational constraint of the cameras about the X-axis and Z-axis, but impose poor translational constraint in the X- and Z-directions and poor orientational contraint about the Y-axis.

Indeed, the empirical model analysis clearly shows this behavior in the numerical precision computation.

The experiment performed here consists of ten increasing levels of noise and one thousand trials performed at each level of noise. For each trial, twenty random 3-D points were chosen in the viewing volume, indicated by the shaded region in Fig. 4-3, and the precision value for each translational and each orientational degree of freedom of relative orientation is computed. The results are illustrated in the graphs of Fig. 4-4.

Fig. 4-4 contains graphs of the standard deviations of the 3-D geometric parameters as a function of increasing noise level. The values express the RMS error of the 3-D parameter constraints relative to the level of feature point measurement error and thus the larger the value the less precise. The solid line represents the error resulting from linear constraints and the dashed line represents the error resulting from the full nonlinear constraints.

It can be clearly seen in these numerical results that, indeed, the precision of the linear constraints is very poor in the X- and Z-translation and in the Y-rotation while the precision of the nonlinear constraints remains good in all parameters. The parameters for which poor precision is observed are those that correspond to camera motion within a "principal epipolar plane". A "principal epipolar plane" is one which is roughly parallel to all the epipolar planes; in this case the XZ-plane. Such a principal plane can in general be the plane that is formed by the two camera COPs and the center of the scene.

The conclusion that can be drawn is that there is a serious limitation to the precision that can be obtained from linear constraints. This conclusion, along with the other benefits of a nonlinear framework, as discussed in Chapter 2, support the argument of using the nonlinear framework as a general approach to a wide range of vision problems.

4.2 Performance analysis: Simulations

The purpose of these simulations is to illustrate the performance of the model and estimator over a broad range of parameters.

4.2.1 General experimental procedure

The general procedure involves rendering a sequence of point correspondences by projecting a set of 3-D points onto the virtual image plane of a simulated moving camera and then estimating the 3-D geometry from the point correspondences. Up to a global scale factor, the estimator can recover the 3-D points, 3-D motion, and camera focal length used to generate the simulated images. Since we know the original scale factor in the computer simulations, the estimation result can be scaled to the same factor as the generative geometry for direct absolute comparison.

Pseudorandom noise and bias is added to the measurements to simulate tracking errors and the corrupted data is fed to a GEKF estimator (Section 3.5) based on the point geometry

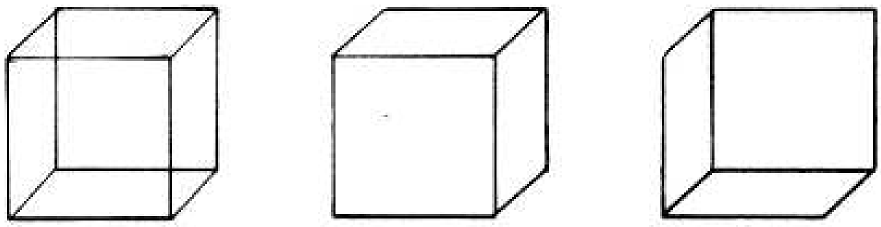


Figure 4-5: The Necker cube (left) illustrates how a set of features can be interpreted in two "depth-reversed" ways under orthographic projection. Perspective helps disambiguate, but the depth-reversed (wrong) solution still represents a strong minimum which can trap the estimator even under perspective. A simple solution to this problem is to check the depth-reversed solution to see if it is a better fit.

model developed above (Section 4.1).

The initial condition on the motion is known to be identity, i.e. zero translation and identity rotation. Therefore, initial error variances on the motion are exactly zero.

A moderate focal parameter $s/f = .5$ (28 deg field of view) is chosen as the initial condition on the focal length. The initial error variance is chosen as described in the model (Section 4.1.2).

The initial condition on the structure is that all points except for two are in a plane parallel to the image at a fixed depth, i.e.

$$(Z_i/d) = \alpha_0, \forall i = 1 \ldots N, i \neq i_0, i_1 \tag{4.45}$$

and of the remaining two,

$$(Z_{i0}/d) = \alpha_0 - \delta\alpha \tag{4.46}$$

$$(Z_{i1}/d) = \alpha_0 + \delta\alpha \tag{4.47}$$

one is pushed deeper and one is pulled closer. The one pulled closer to the camera is a point that is closest to the center of the 2-D cluster of points. The one pushed further away is a point that is furthest from the center of the cluster of points. This choice encourages a convex interpretation of the object and discourages an erroneous "depth-reversed" interpretation, a phenomenon discussed in further detail below.

Initial structure and the "depth reversal" phenomenon

Of the infinitude of choices for initial depths, the obvious choice, assuming nothing at all is known about the structure, would be equal depths for all points, i.e. all points in a plane parallel to the virtual image plane. The convergence failure that is bound to happen in this situation, however, is a phenomenon called "depth reversal".

Depth reversal is a perceptual phenomenon that can be convincingly illustrated with use of the Necker cube (Fig. 4-5), an orthographic projection of the edges of a cube. The cube can be interpreted in two ways equally well: the left square is the front face, facing up and left, or the right square is the front face, facing down and right. In real images perspective, occlusion, shading and other cues settle the issue of which interpretation is correct, but with just line or point features the geometric ambiguity persists.

When all points are in a plane parallel to the image plane, even with some amount of perspective effect, the system is on the boundary between the two interpretations and can easily go either way, leading either to the correct solution, or to a wrong "depth reversed" solution. One way to avoid this ambiguous initial condition is to bias the solution towards the correct interpretation. In real vision, other cues from the image provide the correct bias. In our experiments, all we have is the points, so we have to make an assumption.

In the simulations, we know the correct structure, so we can easily establish a favorable initial condition. Instead, a slightly more generally applicable assumption of convexity is assumed. By pulling a central point closer to the camera and pushing a peripheral point away from the camera, the convex interpretation has an initial advantage over the erroneous concave interpretation. In situations where the object is not actually convex, this will favor the erroneous solution and a different initial condition would be preferred.

The initialization procedure with regards to "depth reversal" is not a great concern because erroneous depth-reversed solutions can easily be detected and corrected automatically. Two estimators could be biased towards opposite interpretations and the one with lower residual error can be chosen as correct.

It is also not a great practical concern because for any given real situation a good assumption can usually be made, e.g.: for the head tracking application described in Section 4.4 a convexity assumption always works, for the models-from-video application of Section 4.5 a user can choose convexity or concavity, and for the film post-production application of Section 4.6 the user can specify deep and shallow points.

Moreover it is not a great theoretical concern because the depth reversal phenomenon occurs naturally in the human visual system as well, as supported by Fig. 4-5. Humans have no problems with real scenes because other cues exist.

Of much greater concern, then, in our simulations are other possible errors and convergence problems. Therefore, for the purpose of evaluating the interesting aspects of performance, the correct assumption about depth polarity is always enforced in the initial condition.

Alternatively, the experiments could have been run with a purely planar initial condition and depth reversals could have been detected and corrected. However, there is little to be gained except for complexity; the former approach is simpler to implement and explores the same regime of control variables.

4.2.2 Experiment 1: Noise analysis, perspective camera

The purpose of this experiment is to evaluate the estimation performance of the model and optimizer with a "normal" set of camera parameters and a wide range of noise, from zero up to extreme levels. The results should give a reasonable idea of what performace will be like using a real camera.

Procedure

In the first experiment, the generative 3-D structure is a set of 26 points occupying a spherical surface. The camera is a perspective projection with $s/f = 1$ (53 deg field of view). The motion consists of a 180 deg rotation around the 3-D object, i.e. the camera moves in a circular path around the object while rotating to point directly at the object (this can also be interpreted as the object spining in view of a static camera) (see Fig. 4-6). The generative motion parameters are illustrated in Fig. 4-7(c). Each sequence consists of

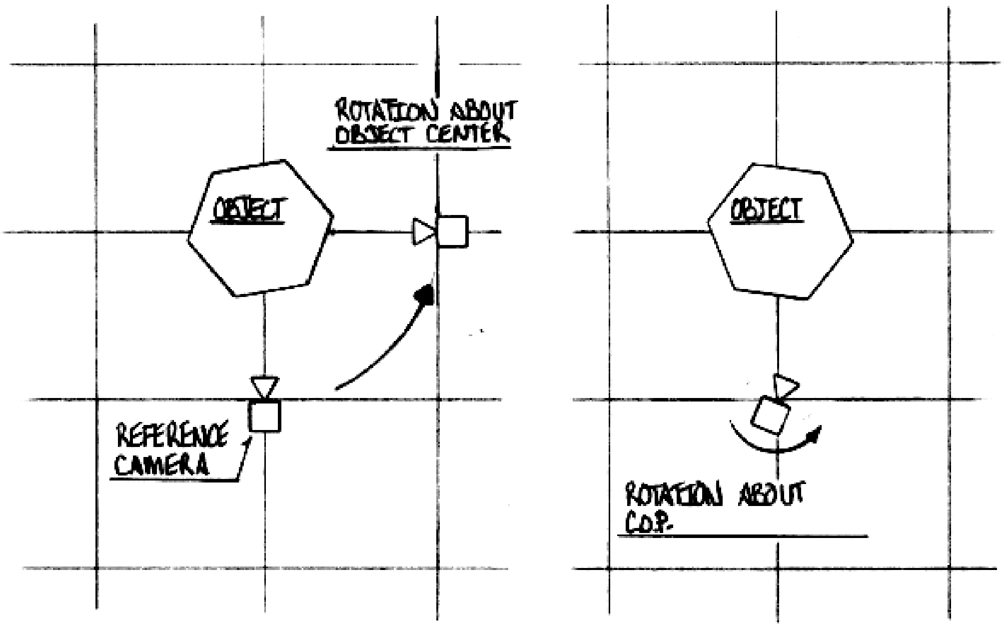


Figure 4-6: Experimental procedure for Exp. 1, Exp. 2, and Exp. 3 is to rotate a camera around the object's center. Experimental procedure for Exp. 4 is to rotate the camera around its own COP; this is a physically degenerate motion in which scene structure cannot be recovered.

this motion over 100 discrete frames.

Zero-mean uniformly-distributed noise over the interval $[-\gamma, \gamma]$ is added to each coordinate of each measurement, where for each run of the experiment γ is chosen to be $0, 2, 4, 6$, or 12 pixels, based on an image size of (512,512). Since the object is less than one third the size of the image, these noise intervals correspond to roughly 2.5%, 5%, 7.5%, and 15% of the size of the object.

The initial condition on the camera is $s/f = .5$ with an error variance of $(.125)^2$ (see Section 4.1.2). The initial depth plane is chosen to be $\alpha_0 = 2$ to match the scale with the generative structure. A convex initial condition is chosen with $\delta\alpha = 1$ (see Section 4.2).

Results

Fig. 4-7 contains two typical parameter estimation results and a table with cumulative statistics for various noise levels. Fig. 4-7(a) contains the estimation result for the case of no added noise. Fig. 4-7(b) contains the estimation result for an extreme level of noise, a distribution of noise up to ± 6pixels (s = 512pixels). Fig. 4-7(c) contains the actual generative motion parameters for comparison. The generative values for the static structure and camera parameters are the ones they appear to converge to. In the translation plot, the solid line is T_X/d, the dashed line is $(T_Z/d)(s/f)$. In the rotation plot, the solid line is q_0, the dashed line is q_Y.

The table in Fig. 4-7(d) demonstrates quantitatively the gradual degradation in performance with increased noise. For the motion estimates, (m_t,σ_t) are the mean and RMS errors for translation and, likewise, (m_q,σ_q) are mean and RMS error-cone angular errors for the rotation (in radians).

For structure and camera estimates, a rate of convergence is computed by fitting a decaying exponential function to the absolute error. Thus, for each trial the mean structure error is closely described by $A_s \exp(-r_s t)$ and the camera parameter error by $A_c \exp(-r_c t)$, where t is the frame number. Thus, for example, at the 40th frame in Exp. 1, the mean structure error is approximately 0.68 (.34% of mean depth) for the zero-noise case and 1.07 (.54% of mean depth) for the 6-pixel noise case.

4.2.3 Experiment 2: Noise analysis, orthographic camera

The purpose of this experiment is to evaulate the estimation performance in the extreme case of orthographic projection.

Procedure

The experimental parameters here are the same as in Exp. 1 except that a generative focal parameter of $s/f = 0$ is used, representing an orthographic projection. The size of the sphere is also reduced so that it will fit in the reduced field of view.

Although the motion is the same, note that the actual motion parameters in Fig. 4-8(c) are slightly different in that the $(T_Z/d)(s/f)$ parameter is identically zero. Since $s/f = 0$, the value of the motion T_Z/d is numerically indeterminate as it is physically indeterminate, as discussed in Section 3.7.5. The $(T_Z/d)(s/f)$ parameter is however fully determinate with a value of zero.

Results

The results in Fig. 4-8 are exactly analagous to those of Exp. 1. Fig. 4-8(a) contains the estimation result for the case of no added noise. Fig. 4-8(b) contains the estimation result for an extreme level of noise, a distribution of noise up to $\pm$6pixels ($s = 512$pixels). Fig. 4-8(c) contains the actual generative motion parameters for comparison. The generative values for the static structure and camera parameters are the ones they appear to converge to. In the translation plot, the solid line is T_X/d, the dashed line is $(T_Z/d)(s/f)$. In the rotation plot, the solid line is q_0, the dashed line is q_Y.

The table in Fig. 4-8(d) contains statistical measures of accuracy and convergence rate as explained in Exp. 1.

4.2.4 Experiment 3: Biased feature tracking

The purpose of this experiment is to evaluate the performance of the estimator in the presence of tracking bias under the assumption of zero-mean tracking.

Procedure

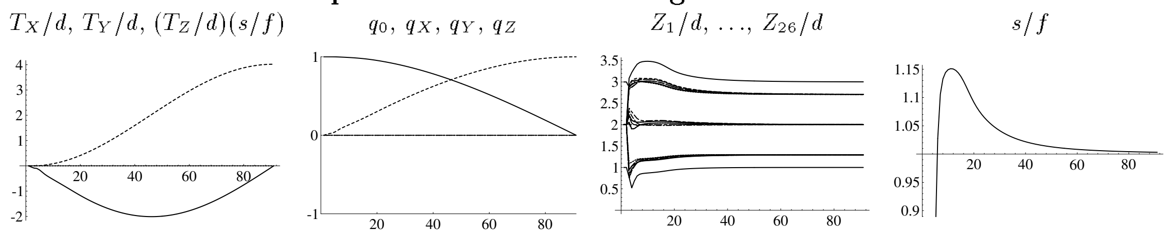
The experimental parameters here are the same as in Exp. 1 except that a fixed level of noise is used for all trials and a random tracking bias is introduced at various levels of magnitude.

For each trial, a noise level of 4pixels ($s = 512$pixels) is used. Random biases are added to each coordinate of each feature track where each bias is chosen from a unform distribution over the interval $[-\gamma, \gamma]$, where we choose γ to be 0, 2, 4, 8, or 12 pixels.

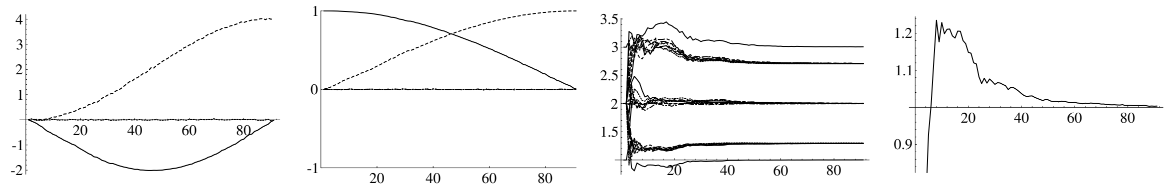
Results

The results in Fig. 4-9 are analagous to those of Exp. 1. Fig. 4-9(a) contains the estimation result for the case of a moderate level of added bias, $\pm$2pixels. Fig. 4-9(b) contains the

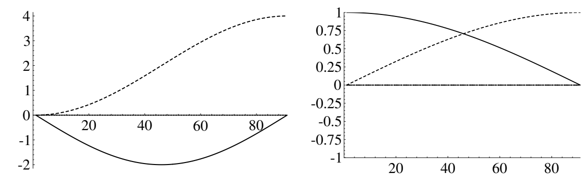
Experiment 1: Increasing Noise Level



(a) Estimated parameters, no added noise.



(b) Estimated parameters, added noise uniform over ±6 pixels (7.5% of object size)



(c) Actual motion parameters.

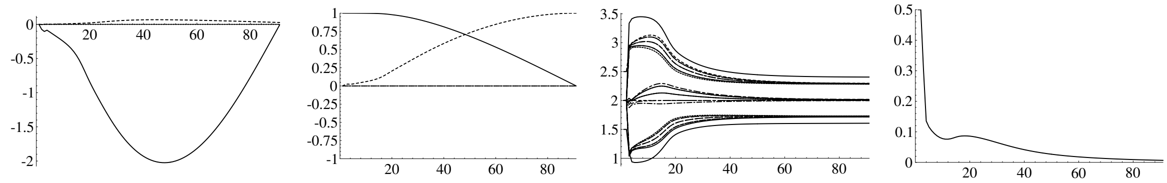
	Estimation Error				Convergence Rate	
Noise	Motion				Structure	Camera
pixels	m_t	σ_t	m_q	σ_q	$r_s(\mathrm{fr}^{-1})$	$r_c(\mathrm{fr}^{-1})$
0.	-0.0084	0.0250	0.0199	0.0273	0.1052	0.1262
2.	-0.0092	0.0252	0.0206	0.0280	0.0999	0.1042
4.	-0.0098	0.0256	0.0208	0.0283	0.1002	0.0946
6.	-0.0108	0.0283	0.0219	0.0299	0.0937	0.0850
12.	-0.0177	0.0464	0.0310	0.0421	0.0634	0.0571

(d) Statistics for several noise levels.

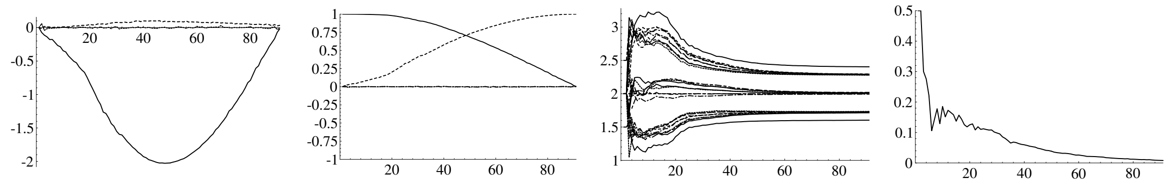
Figure 4-7: Experiment 1, synthetic data with random noise added. Increasing measurement noise results in increasing estimation error on dynamic variables (motion) and slower convergence rate of static variables (structure, camera). Focal length is 1.0 (53 deg field of view), pixels are based on the image being (512,512). Initial mean structure error, $A_s = .4577$, initial camera error $A_c = .5$.

Experiment 2: Orthographic

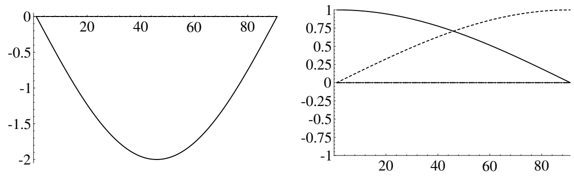
T_X/d, T_Y/d, $(T_Z/d)(s/f)$ — q_0, q_X, q_Y, q_Z — Z_1/d, ..., Z_{26}/d — s/f



(a) Estimated parameters, no added noise.



(b) Estimated parameters, added noise uniform over ±6 pixels (7.5% of object size)



(c) Actual motion parameters.

	Estimation Error				Convergence Rate	
Noise	Motion				Structure	Camera
pixels	m_t	σ_t	m_q	σ_q	$r_s(\mathrm{fr}^{-1})$	$r_c(\mathrm{fr}^{-1})$
0.	-0.0441	0.1144	0.0987	0.1320	0.0213	0.2293
2.	-0.0481	0.1168	0.1032	0.1358	0.0197	0.1399
4.	-0.0470	0.1115	0.1003	0.1314	0.0204	0.1004
6.	-0.0456	0.1066	0.0971	0.1271	0.0214	0.0902
12.	-0.0536	0.1169	0.1093	0.1405	0.0186	0.0621

(d) Statistics for several noise levels.

Figure 4-8: Experiment 2, synthetic data with random noise added. As in the perspective projection case, increasing measurement noise results in increasing estimation error on dynamic variables (motion) and slower convergence rate of static variables (structure, camera). Focal length is ∞ (0 deg field of view), pixels are based on the image being (512,512). Initial mean structure error, $A_s = .1908$, initial camera error $A_c = .5$.

estimation result for an extreme level of bias, a distribution of random bias up to ± 8pixels. Fig. 4-9(c) contains the actual generative motion parameters for comparison. The generative values for the static structure and camera parameters are the same as in Exp. 1 (the true focal parameter $s/f = 1$).

The table in Fig. 4-9(d) contains statistical measures of accuracy and convergence rate as explained in Exp. 1.

4.2.5 Experiment 4: Degenerate case — rotation about COP

In a general perspective projection, all optical rays pass through the center of projection (COP). Since traditional camera models have their origin at the COP, a rotation about the COP is often referred to as a "pure rotation". (Our camera model does *not* have its origin at the COP, so the term is somewhat of a misnomer here. Rotation about the COP will have a translation component relative to the virtual image-plane coordinate system, but geometrically the same degeneracy exists.)

Theoretically, in a rotational motion around the COP, the structure cannot be recovered at all, yet the focal length and the motion parameters corresponding to the rotation about the COP *can* be recovered. Fig. 4-10 confirms that our estimator in fact recovers what is estimable and fails to recover the structure and the structure-dependent translational component.

Procedure

The experimental procedure here is the same as in Exp. 1 except that a wider focal parameter is used and the sphere is pushed farther away from the camera so that a significant amount of rotation about the COP can take place with the object remaining in full view. The generative focal parameter is $s/f = 2$ and the trajectory consists of rotating about the vertical axis 22 deg and then -44 deg to return to an absolute rotation of -22 deg. The generative motion parameters are depicted in Fig. 4-10(c) and pictorially in Fig. 4-6.

Results

The results in Fig. 4-10 are analagous to those of Exp. 1. Fig. 4-10(a) contains the estimation result for the case of zero added measurement noise. Fig. 4-10(b) contains the estimation result for an extreme level of noise. Fig. 4-10(c) contains the actual generative motion parameters for comparison.

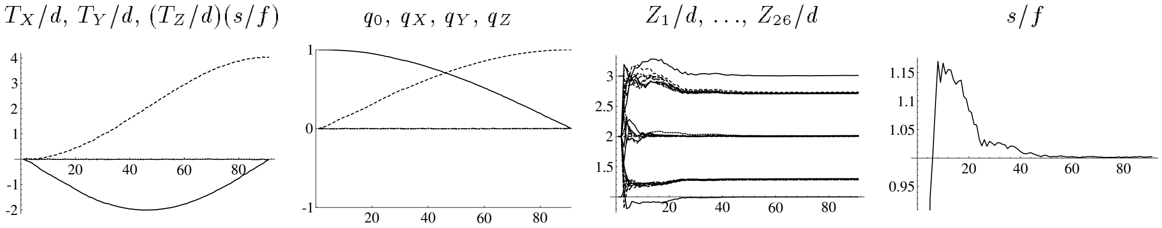
The table in Fig. 4-10(d) contains statistical measures of accuracy and convergence rate for several noise levels as explained in Exp. 1.

The principal qualitative result is that the rotation and focal parameters can indeed be estimated, as predicted, and the structure and components of the translation cannot. In the no-noise case, the ambiguity of the structure and translation is subdued because of the stabilizing probabilistic structure of the Kalman filter. When an extreme level of noise beyond the modeled level is introduced, as in Fig. 4-10(b), the estimate of the ambiguous variables becomes correspondingly noisy.

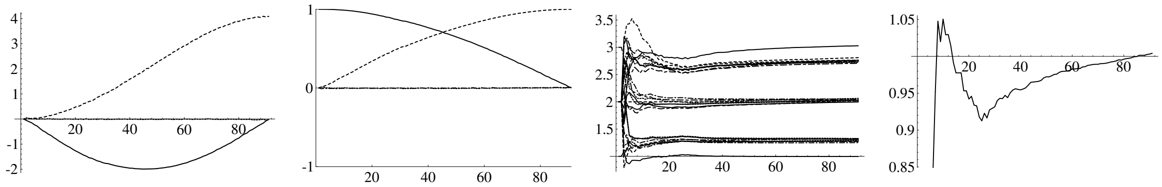
4.2.6 Experiment 5: Monte Carlo results

The purpose of this set of experiments is to evaluate the performance of the model and estimator over a large number of trials consisting of various motions, focal parameters, and noise levels.

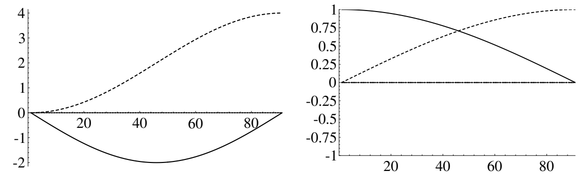
Experiment 3: Biased Measurements



(a) Estimated parameters, biases uniform over ± 2 pixels (2.5% of object size), noise level ± 4 pixels (5% of object size).



(b) Estimated parameters, biases uniform over ± 8 pixels (10.0% of object size), noise level ± 4 pixels (5% of object size).



(c) Actual motion parameters.

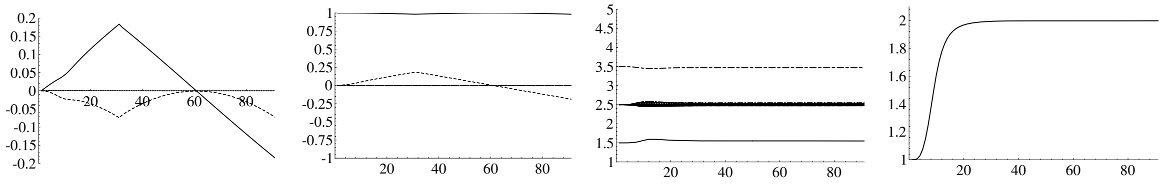
	Estimation Error				Convergence Rate	
Bias	Motion				Structure	Camera
pixels	m_t	σ_t	m_q	σ_q	$r_s(\mathrm{fr}^{-1})$	$r_c(\mathrm{fr}^{-1})$
0.	-0.0098	0.0256	0.0208	0.0283	0.1002	0.0946
2.	-0.0071	0.0206	0.0117	0.0185	0.1541	0.1433
4.	-0.0045	0.0185	0.0024	0.0122	0.1964	0.1977
8.	-0.0002	0.0252	-0.0134	0.0249	0.1505	0.2296
12.	0.0044	0.0375	-0.0262	0.0430	0.0842	0.1450

(d) Statistics for several bias levels.

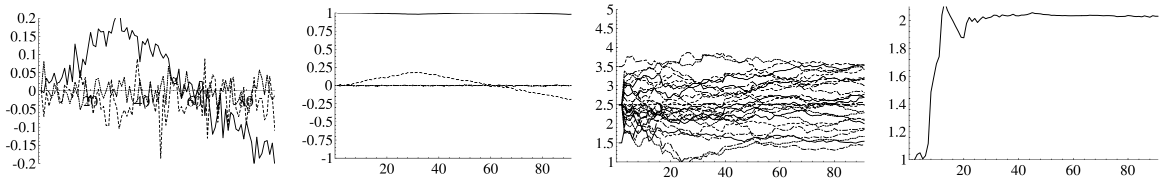
Figure 4-9: Experiment 3, synthetic data with random biases and noise added. Even large biases have only a moderate effect on accuracy. Focal length is 1.0 (53 deg field of view), noise level ± 4 pixels (5% of object size), pixels are based on the image being (512,512). Initial mean structure error, $A_s = .4577$, initial camera error $A_c = .5$.

Experiment 4: Degenerate case — Rotation about COP

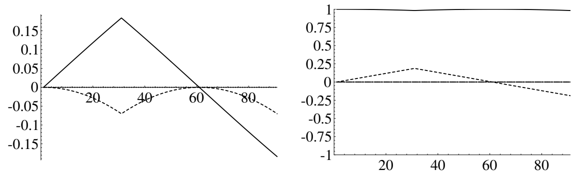
T_X/d, T_Y/d, $(T_Z/d)(s/f)$ $\quad$ q_0, q_X, q_Y, q_Z $\quad$ Z_1/d, …, Z_{26}/d $\quad$ s/f



(a) Estimated Trajectory, no added noise.



(b) Estimated Trajectory, added noise uniform over ± 6 pixels (7.5% of object size)



(c) Actual motion parameters.

	Estimation Error				Convergence Rate	
Noise	Motion				Structure	Camera
pixels	m_t	σ_t	m_q	σ_q	$r_s(\mathrm{fr}^{-1})$	$r_c(\mathrm{fr}^{-1})$
0.	0.0019	0.0039	0.0026	0.0064	(-0.0004)	0.1098
2.	0.0020	0.0186	0.0020	0.0094	(-0.0028)	0.1136
4.	0.0020	0.0284	0.0005	0.0121	(-0.0048)	0.1155
6.	0.0018	0.0361	-0.0006	0.0136	(-0.0072)	0.1119
12.	0.0011	0.0654	-0.0038	0.0218	(-0.0104)	0.0863

(d) Statistics for several noise levels.

Figure 4-10: Experiment 4, synthetic data with random noise added. The degenerate case of rotation around the COP yields no information about structure. These data confirm that the "pure rotational" motion and the focal length *can* be recovered, while the structure and absolute translation *cannot* be recovered. The estimator remains well-conditioned due to the formulation. Focal length is .5 (90 deg field of view), pixels are based on the image being (512,512). Initial mean structure error, $A_s = .4577$, initial camera error $A_c = 1.0$.

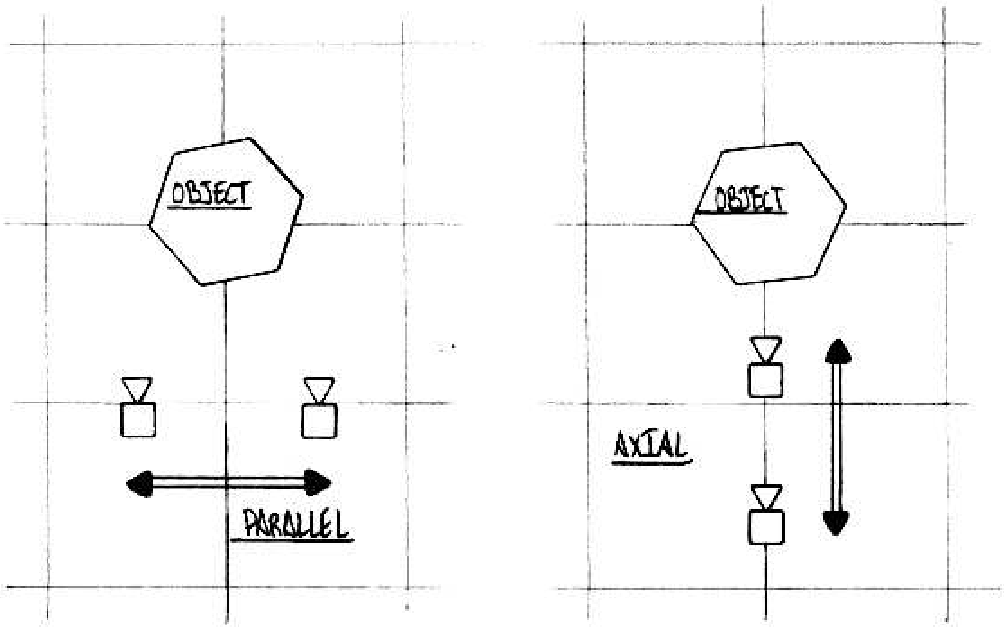


Figure 4-11: The parallel and axial motions used in Exp. 5.

Procedure

In addition to the wide rotation used in the previous experiments, a parallel motion and an axial motion are introduced, as depicted in Fig. 4-11. For each of the three motions, a number of trials are performed with varying focal parameters and varying amount of added noise. For each trial, a new random structure is chosen. The random structure consists of 20 points where the three coordinates are chosen randomly from a uniform distribution in a cubical volume.

For each of the three motions, the range of focal parameter was varied from $s/f = 0$ (orthographic) to $s/f = 2$ (90 deg field of view) holding the noise level constant at 1pixel (in a 512pixel image). A number of trials with random structures was performed at each set of control variables.

Also for each of the three motions, the noise level was increased from zero to 10pixels holding the focal parameter at $s/f = 1$ (53 deg field of view). A number of trials with random structures was performed at each set of control variables.

Results

Fig. 4-12, Fig. 4-13 and Fig. 4-14 show the experimental Monte Carlo error results of motion, structure, and focal length estimation for each of the three types of motion. For each type of motion, error statistics are presented for various focal parameters and various levels of added noise. Each bar on a graph represents mean error over 15 trials at that focal length and noise level, with a different, randomly-chosen structure used in each trial.

Structure errors are reported as percent of mean depth, where the errors are averaged over all points over all trials at each focal length and noise level. Camera errors are plotted as errors in terms of the camera parameter, s/f, but are quantified in terms of field of view error; the isolines on the graph represent constant field of view, which goes as

$2\arctan((s/f)/2)$. Translation errors are reported in terms of percent of the mean depth of the points. Rotation errors are reported as radians error from the true rotation.

Fig. 4-12(a) shows results for fields of view ranging from 10 deg to 60 deg and Fig. 4-12(b) shows results for noise levels up to 5 pixels, for the *Rotational* motion. Structure error is less than 1% of mean depth, translational motion errors are less than 1%, rotation errors within a .5 deg error cone, and field of view errors also within a .5 deg error cone.

For the *Parallel* motion, Figures 4-13(a) and 4-13(b) show that structure errors are somewhat larger (as expected), in the 2% – 7% range, translation errors are in in the 2% – 6% range, rotations remain in a .5 deg error cone, and the camera error bound increases to about 2.5 deg.

Finally, for the *Axial* motion, Figures 4-14(a) and 4-14(b) show that structure errors are 1% – 4%, translation is around 2%, rotation errors are around .2 deg, and camera error is in the 2 deg – 5 deg range.

Qualitatively we can observe from the controlled experiments that rotation estimation is relatively unaffected by the type of motion whereas structure, camera, and translation are much poorer for the parallel and axial motions. This is expected, since these motions do not provide as much difference in viewing angle to each of the points.

In response to increasing field of view, we find that rotation estimation is only slightly worse for wide angles as it is for orthographic projection. This is because in orthographic projection, rotation cannot be confused with translations; only rotations alter the relative positions of the points in the image. With wider fields of view, the potential for confusion under conditions of noise increases. Structure estimation improves with wider angles because the total change in viewpoint is greater for all motions. The camera estimate errors become numerically larger in s/f, but as seen by comparing to the field of view isolines, the field of view error does not change dramatically. Since translation errors are tied to the s/f parameter, they tend to go as the numerical error in s/f. The translational errors reflect primarily T_Z-error (along the optical axis).

In response to increasing noise, rotational estimation degrades smoothly in an expected way. Camera, structure, and translation appear to degrade more slightly with noise. Structure and translation are much more affected by the camera error, which in turn is much more affected by type of motion than by noise.

4.3 Performance analysis: Calibrated video sequence

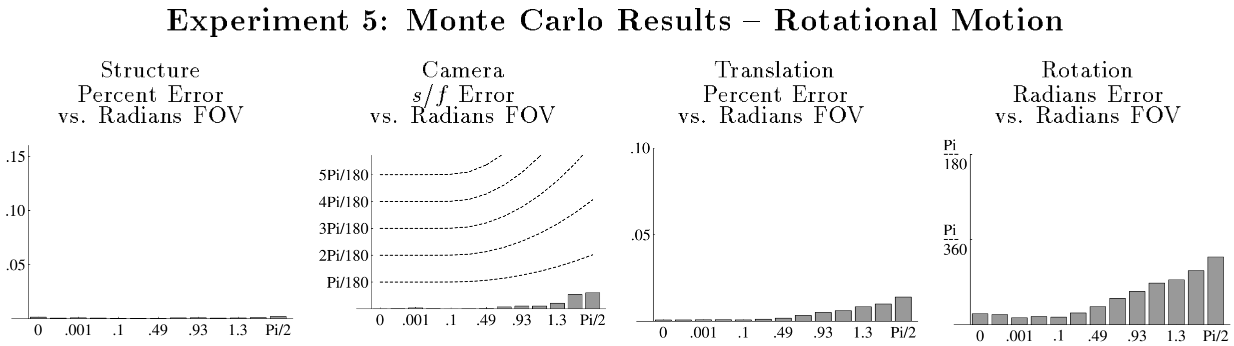
To provide a means for comparing our estimator performance to others, we have performed a final experiment on a publicly available sequence. This is the BOX sequence from the UMass database [23].

The basic conclusion of the experiment, however, is that using real imagery and a measured motion is actually a very poor way of comparing results of different methods. This is due to the fact that experimental procedure is not standardized and the fact that there are unknown errors in "ground truth" which make it impossible to separate various kinds of errors from the estimation residuals.

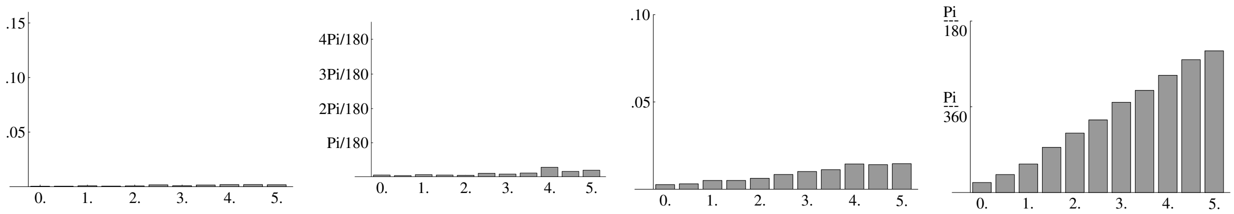
4.3.1 Experiment 6: Calibrated imagery

Procedure

The experimental data set consists of a set of "ground truth" parameters for 3-D points, 3-D motion, and the focal length of the camera in addition to the 2-D feature points through 9 images. The quotes on "ground truth" are to emphasize that these are not actually true physical values, as one would have in computer simulations, but are some set of measure-

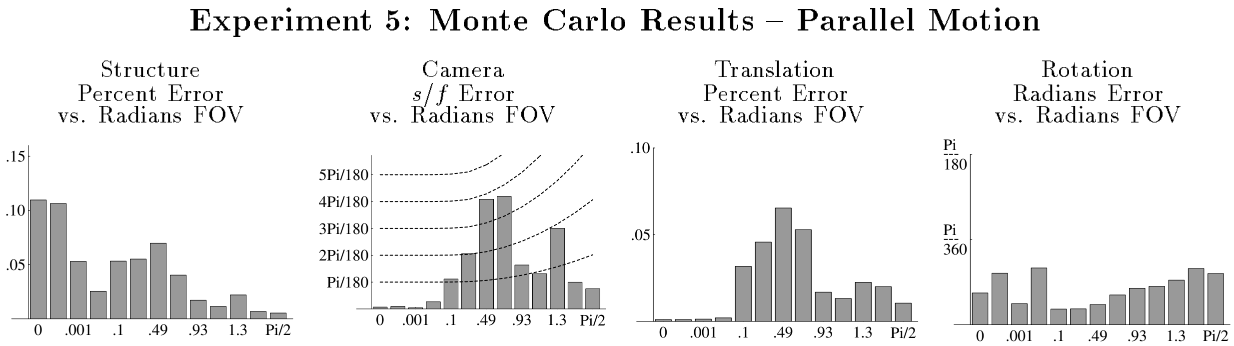


(a) The "rotate" motion, accuracy vs. field of view. The abscissa is field of view of the actual camera, ranging from 0 deg (orthographic) to 90 deg.

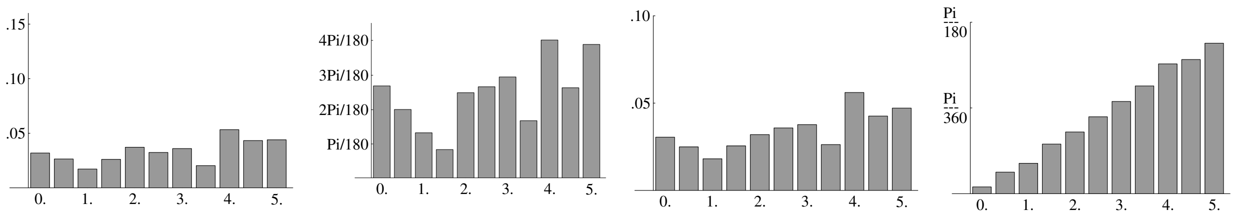


(b) The "rotate" motion, accuracy vs. noise level. The abscissa is the range in pixels of the added uniform noise, ranging from zero to 5 pixels.

Figure 4-12: Experiment 5, Rotation motion. Accuracy versus field of view and noise level.



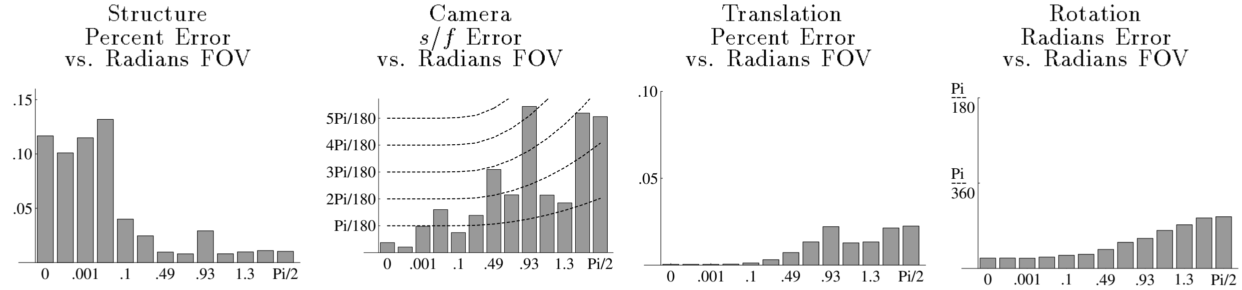
(a) The "parallel" motion, accuracy vs. field of view. The abscissa is field of view of the actual camera, ranging from 0 deg (orthographic) to 90 deg.



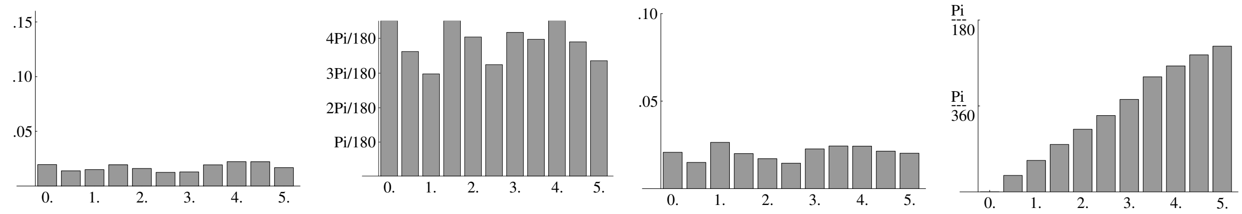
(b) The "parallel" motion, accuracy vs. noise level. The abscissa is the range in pixels of the added uniform noise, ranging from zero to 5 pixels.

Figure 4-13: Experiment 5, Parallel motion. Accuracy versus field of view and noise level.

Experiment 5: Monte Carlo Results – Axial Motion



(a) The "axial" motion, accuracy vs. field of view. The abscissa is field of view of the actual camera, ranging from 0 deg (orthographic) to 90 deg.



(b) The "axial" motion, accuracy vs. noise level. The abscissa is the range in pixels of the added uniform noise, ranging from zero to 5 pixels.

Figure 4-14: Experiment 5, Axial motion. Accuracy versus field of view and noise level.

ments of the physical values which will themselves have error in them.

Our experiment proceeds as usual, estimating 3-D points, 3-D motion, and focal parameters from the feature tracks. Analysis then takes place based on absolute error (relative to "ground truth" parameters) and on residual error.

Fig. 4-15(a) shows the first and last frames of the 9-frame sequence with features overlaid as white boxes. Fig. 4-15(b) shows the "ground truth" motion parameters and Fig. 4-15(c) shows the parameters recovered by our estimator.

Results

The absolute error using the published "ground truth" as the reference consists of a rigid motion RMS error of 2.5 % and a structure RMS error of 0.7%. The estimated focal length was 4% different from the manufacturer's specification reported in the data set.

The residual error produced by our estimate has a 0.50 pixel mean RMS value. The residual error is based on the difference between the given measurements and the re-synthesized features.

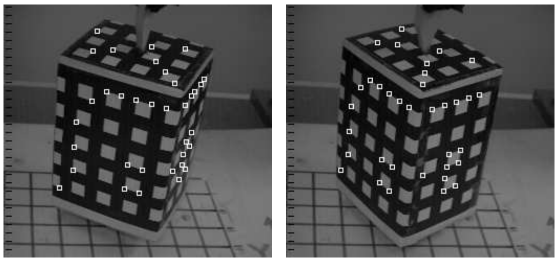
Several other researchers have used this sequence as a test for geometry estimation [51, 80, 40], and have reported rigid motion RMS errors of 0.1% and structure RMS errors of 0.2 to 0.3%. They did not estimate focal length.

To compare our results to the results of these published experiments is difficult because the authors have either performed post-estimation transforms (rigid and scale, or affine) to fit the actual data before computing error statistics [51, 80], or used radically different *a priori* information [40].

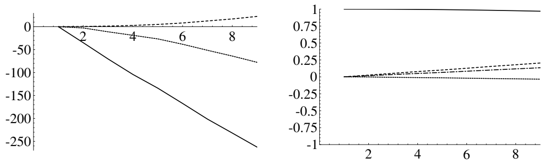
The evaluation with respect to "ground truth" is also not particularly useful because, in addition to the usual modeling errors and measurement errors, there are also unknown "ground truth" errors. It is impossible to separate the three sources of error so any residual of

Experiment 6: Calibrated BOX Sequence

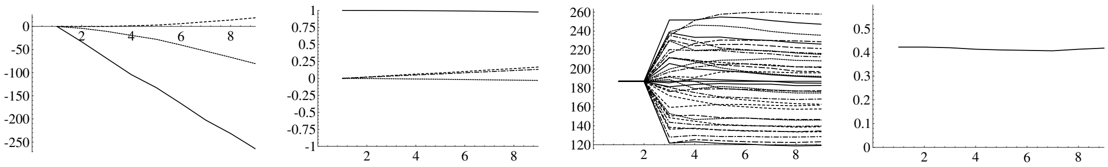
Frame 1 Frame 9



(a) Frames from the sequence



(b) Reference translation and rotation.



(c) Estimated translation, rotation, structure and focal parameters.

Figure 4-15: Experiment 6: The original images, the actual motion parameters, and estimated parameters. The estimated translation and structure have been scaled to match the numerical values of the "actual" parameters. For comparison to other papers using this sequence, the "alpha" depths are from the image plane; distance to COP is an additional 447.3 units.

the estimation could as likely be due to "ground truth" error as to modeling or measurement error.

Assuming the modeling and measurement errors are small we can show that there is indeed a substantial amount of "ground truth" error by computing the residual error of the "ground truth" parameters. The re-synthesis of image locations based on "ground truth" produces a mean RMS image-plane error, over all frames and all features, of 1.45 pixels. This figure is substantially larger than results from our estimate and is substantially larger than what one would expect with a nearly ideal model and subpixel-error feature tracking.

With subpixel tracking, one should expect a distribution of measurement errors with an RMS value of .17 (Gaussian distributed) to .25 (uniformly distributed). This is based on assuming that tracking errors on each image coordinate are in the range $[-.5, .5]$ pixels of the true location (with 98% probability in the Gaussian case). That level of error might be doubled by a small degree of modeling error leading to an expected value for the residual error on the same order as our result (.50 pixels).

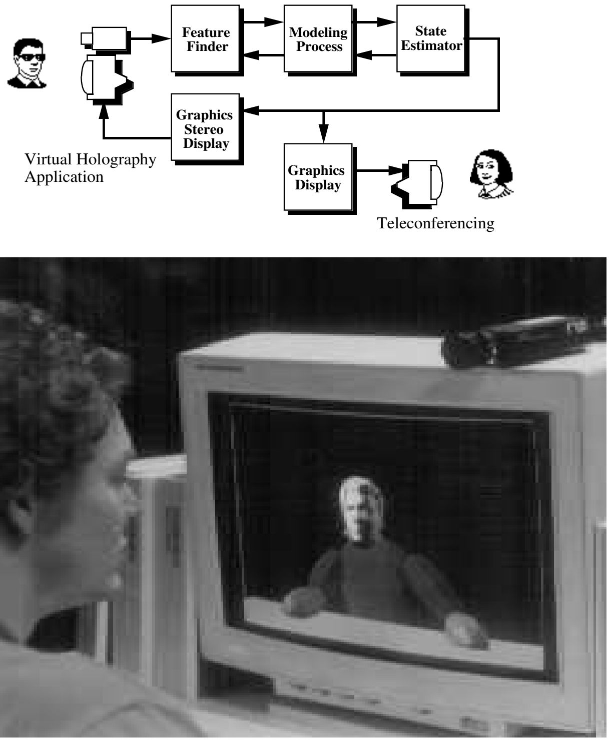


Figure 4-16: A real-time 3-D head tracking system.

4.4 Performance analysis: Head tracking application

The original motivation for pursuing visual 3-D estimation was to build a visual human-computer interface that would understand the 3-D motions of a user. This experimental head-tracking system, built in 1992, was one of the first attempts to demonstrate real-time control of a display based on the user's 3-D motions.

A diagram of the human-computer interface system is shown in Fig. 4-16. The idea is that a 3-D interpretation of the user could be used to control his own display or to control a display of a remote user.

In the local control mode, an example application would be "virtual holography", in which the 3-D viewing angle of the monitor could be estimated from tracking the head and the rendering view of a computer graphics scene could change accordingly to give the same effect as a holographic display.

In the remote control mode, an example application is teleconferencing where the 3-D position and orientation of the user's head can control a computer generated image of the user on the remote display. In this way, a remote shape model only has to be transmitted once and subsequent estimated motion parameters can be transmitted compactly at low bandwidth.

4.4.1 Prototype system description

The system consists of a Sun 4/330 and a Cognex 4400 image processing subsystem, which allows video images from a CCD camera to be digitized to RAM and processed at 30 frames per second. The total system frame rate has reached 10 frames per second (fps) with our implementation.

Selection of distinct features for tracking is based on image characteristics only; we use points where the image intensity surface $I(u,v)$ has a large Hessian, i.e.

$$\left[\left(\frac{d^2I(u,v)}{du^2}\right)\left(\frac{d^2I(u,v)}{dv^2}\right)-\left(\frac{d^2I(u,v)}{dudv}\right)^2\right]>\epsilon$$

for some threshold ϵ. Such points are peaks, saddle points, and pits in the image intensity surface, and correspond to features which can be localized in 2-D without "aperture" problems [44]. On heads, these features often (but not always) correspond to the corners of eyes, pupils, nostrils, etc.

For each frame, a set of features satisfying the Hessian criterion are selected for tracking. Since there are 6 motion parameters at each frame and since each tracked feature provides two measurements at each frame (its 2-D coordinates) only 3 points are theoretically required to recover motion. However, many more (at least 10–20) are typically used in practice to overdetermine the solution and reject noise. In our system, we typically track 5–10 features.

Tracking of features is performed using normalized correlation. Correlation templates are extracted for each feature from the original image and from subsequent images in which a substantially novel viewpoint occurs. Viewpoint changes are detected explicitly by using the 3-D motion estimates from the Kalman filter and implicitly by monitoring degradation of the correlation indices of the features.

The head tracking system uses the ThingWorld modeling system [54] to control a Tektronics stereoscopic display. As with most graphics systems, there can be a significant delay between the time ThingWorld receives object information and the time that it can render the new view. Such lags in updating the user's view can cause anything from a feeling of system sluggishness to actual motion sickness.

This problem can be alleviated by inserting a process between the head tracker and ThingWorld which predicts the position of the head one frame in advance, giving the ThingWorld renderer enough time to maintain synchronization with head position [27]. The optimal linear technique for such prediction is the Kalman filter. Since the head tracker is based on the Kalman filter, we can simply use its predictions of head position and velocity to maintain display synchronization.

The original implementation [4] of the estimation engine was based on an earlier mathematical model of the geometry which assumed both a calibrated camera and known structure of the 3-D points. The 2-D points in the first frame were back-projected onto a 3-D ellipsoid that was approximately the size of an average human head. This back-projection resulted in the set of "known" 3-D points, although this knowledge was obviously very approximate.

This implementation was also concerned with prediction of motion parameters and thus had a Newtonian dynamics model for the internal constraint rather than a simple identity model as described in Section 4.1. See [4] for details.

A later re-implementation [3] used the current formulation of Section 4.1 and estimated both the motion and the 3-D locations of the point features.

4.4.2 Example: Motion estimation, ellipsoidal shape model

To evaluate system performace, a person's head was tracked using the Polhemus sensor and the vision system simultaneously. The state rotational estimates were aligned to each other using *absolute orientation* techniques [35]. Scale and bias were removed by performing a linear regression of the Polhemus data to the vision estimate.

Fig. 4-17 displays some frames from the captured sequence along with plots containing the vision and Polhemus estimates. The captured sequence consists of 150 images with a

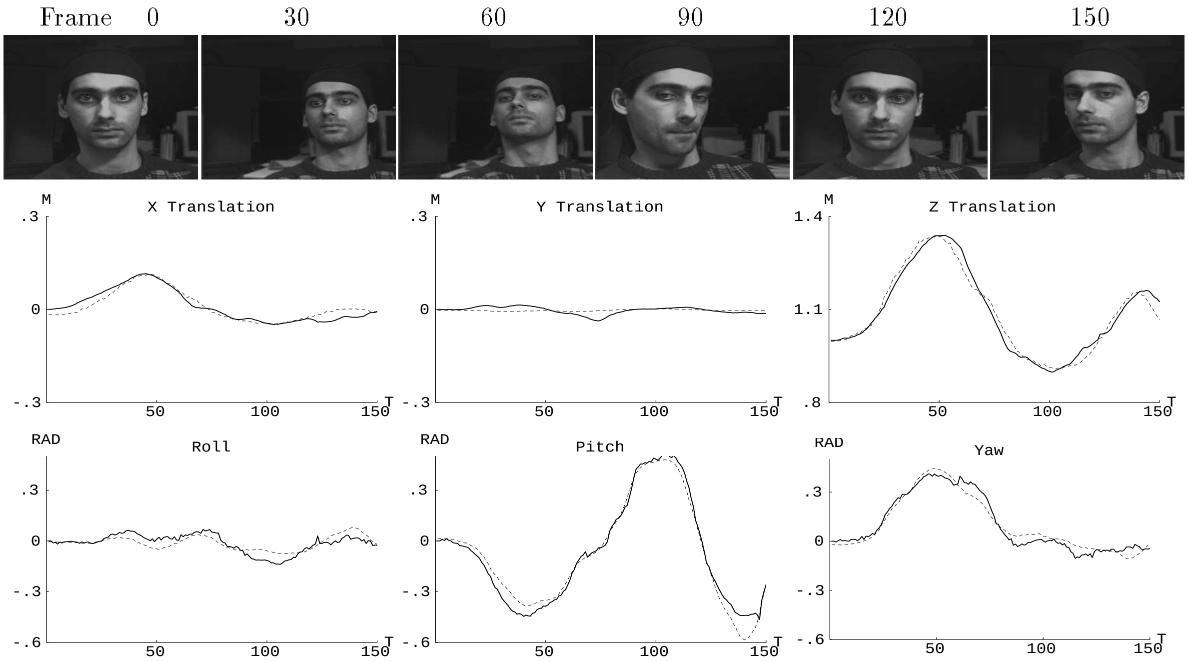


Figure 4-17: Performance analysis for the head tracking application. Structure is initialized by back projecting onto a 3-D ellipsoid. The solid line is the vision estimate, the dashed line is the Polhemus "ground truth". The coordinate frames of the vision and Polhemus estimates are rectified to each other by a rigid transform plus scaling.

large degree of 3-D head motion.

The RMS difference between vision and Polhemus estimates is 1.67 cm and 2.4 degrees. These statistics are comparable to the observed Polhemus accuracy, indicating that the vision estimate is as least as accurate as the Polhemus.

4.4.3 Example: Motion and structure estimation

Fig. 4-18 shows results from the same data set using a full structure and motion recovery. Note that the coordinate system and units for the translational parameters are different because the choice of virtual image plane was different. (In Fig. 4-17 the translation is that of the COP and the units are meters; in Fig. 4-18 the translation is of a scene-centered reference plane origin.) The rotational parameters, of course, have the same meaning and have very similar values.

The RMS difference in translation is 0.11 units and the RMS difference in rotation is 2.35 deg. The scale of translation is, of course, unknown, but is approximately 10–12cm per unit, yielding a RMS tracking error of approximately 1 cm.

4.5 Performance analysis: Models-from-video application

Whereas the head-tracking system utilizes only the motion parameters from the 3-D geometry estimator, the 3-D structure parameters can also lead to an interesting set of applications.

In this case, the application is motivated by the difficulty associated with developing

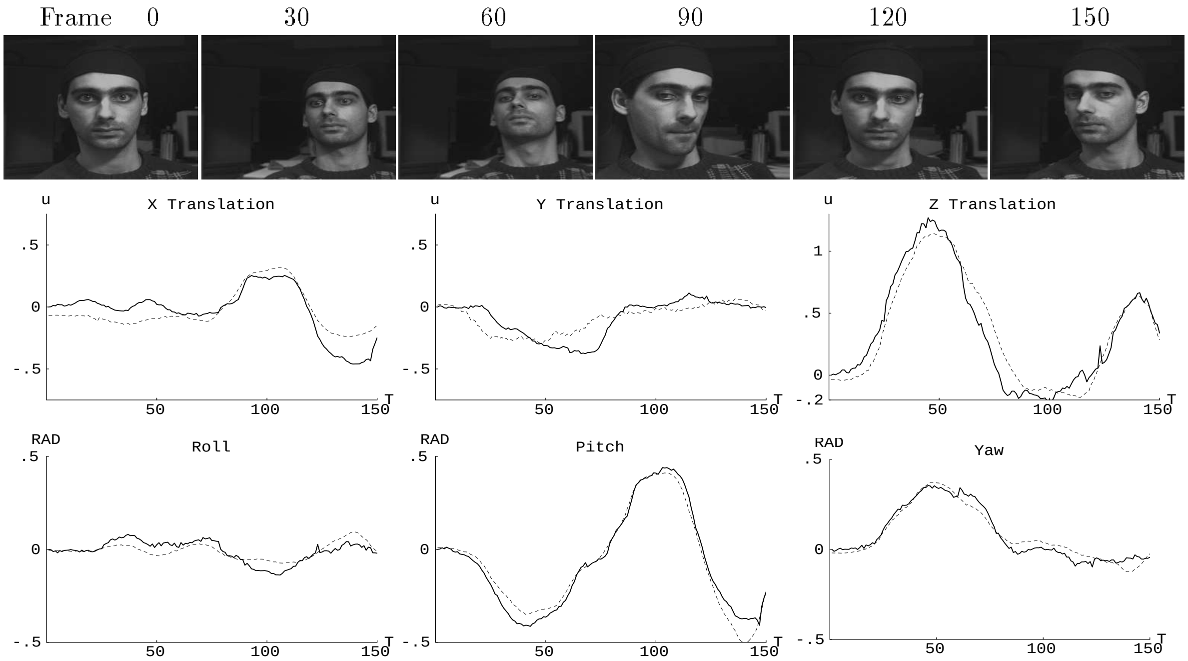


Figure 4-18: Performance analysis for the head tracking application. Structure is estimated along with motion. The solid line is the vision motion estimate, the dashed line is the Polhemus "ground truth". The coordinate frames of the vision and Polhemus estimates are rectified to each other by a rigid transform plus scaling. (The ground truth motion is physically identical to that shown in Fig. 4-17 but since the coordinate system origin is different, the translational parameters are different. The ground truth rotational parameters are identical because rotation is not affected by coordinate origin.)

content for 3-D computer graphics environments ("virtual environments"). The process of "modeling" in the context of computer graphics means building up polygonal models of objects for various uses. The art of building objects which appear at all realistic or complex requires a great deal of skill and patience, and as a practical concern is consequently very expensive.

In many cases, the modeler is trying to mimic a real object, which he does by looking at the object, or pictures of the object, and tries to encode all the geometry by hand. The hope is that, given enough pictures of the object, a computer can do this modeling automatically, saving a lot of time and energy for the artist.

The 3-D estimation capabilities developed in this chapter have led to the following prototype system for performing this task.

4.5.1 Prototype system description

The prototype system [2] is based on a computer program with an X-windows user interface. The computer program can select features automatically or the user can select features. Tracking is performed with an SSD (sum of squared distance) metric on templates, which are updated as the imagery moves.

The estimator produces the 3-D geometry estimate from the tracked feature measurements, resulting in an internal representation of 3-D camera motion, 3-D points, and the focal parameter.

The 3-D points are used to fit 3-D surfaces. This is accomplished by having the user group together all the points on a single planar surface. Experiments were performed on automatic segmentation using the 3-D estimate and the images, but for the purposes of this system, manual segmentation is adequate. Planar surface parameters are estimated for each group of planar features.

The user then selects the 2-D vertices of each planar polygon in any one of the images. The recovered 3-D motion and camera parameters are used to back-project the 2-D vertices onto the 3-D surfaces resulting in 3-D vertices. These vertices then define a set of 3-D polygons which describe the 3-D geometry of the building.

With this information in place, an automatic process is then invoked to recover texture maps for the 3-D polygons from the original images. Since the 3-D cameras and 3-D polygons are now known, the mapping between any location on any polygon and the image location in any image is defined. A high-resolution texture map can be generated for each polygon by painting each location on the polygon with a weighted sum of values obtained from a set of images.

4.5.2 Example: Wiesner building

In this example, a texture-mapped model of a building is extracted from a 20-second video clip of a walk-around outside a building (the Wiesner Building, housing the Media Laboratory, MIT). Every sixth video image was digitized, resulting in an image sequence of 90 frames. The top row of Fig. 4-19 shows some of the frames of the original digitized video.

The second row of Fig. 4-19 shows some of the twenty-one features on the building that were selected and tracked. These feature tracks constitute the measurements. The resulting estimates of 3-D camera geometry, camera motion, and pointwise structure are shown in Fig. 4-20. The GEKF is iterated once to remove the initial transient on the static parameters.

Recovered 3-D points are used to estimate the planar surfaces of the walls. The vertices are selected in an image by hand and back projected onto the planes to form 3-D polygons, depicted in wireframe in Fig. 4-19. The recovered motion and focal parameters are critical to being able to do this.

The resulting polygons, along with the recovered motion and focal length are used to warp and combine video from 25 separate frames to synthesize texture maps for each wall using a procedure developed by Galyean [6]. In the fourth row of Fig. 4-19, the texture-mapped model is shown rendered along the original trajectory.

In the last row of Fig. 4-19 a modified version of the original imagery is shown in which the 3-D geometric parameters are used to alter the original. The motion and focal length are used to render the artificial objects with the proper camera trajectory and projection parameters and the 3-D structure of the building is used to generate the proper mattes for occlusion.

The image-plane accuracy of the reconstructed model was assessed by examining the differences in the back-projected polygon textures. The result was that the image-plane errors were less than 1/2 pixel throughout the video sequence. (Since a single focal length and structure were used, the motion estimates for each frame in the sequence have to be accurate for the model and the video to line up after projection of the model.)

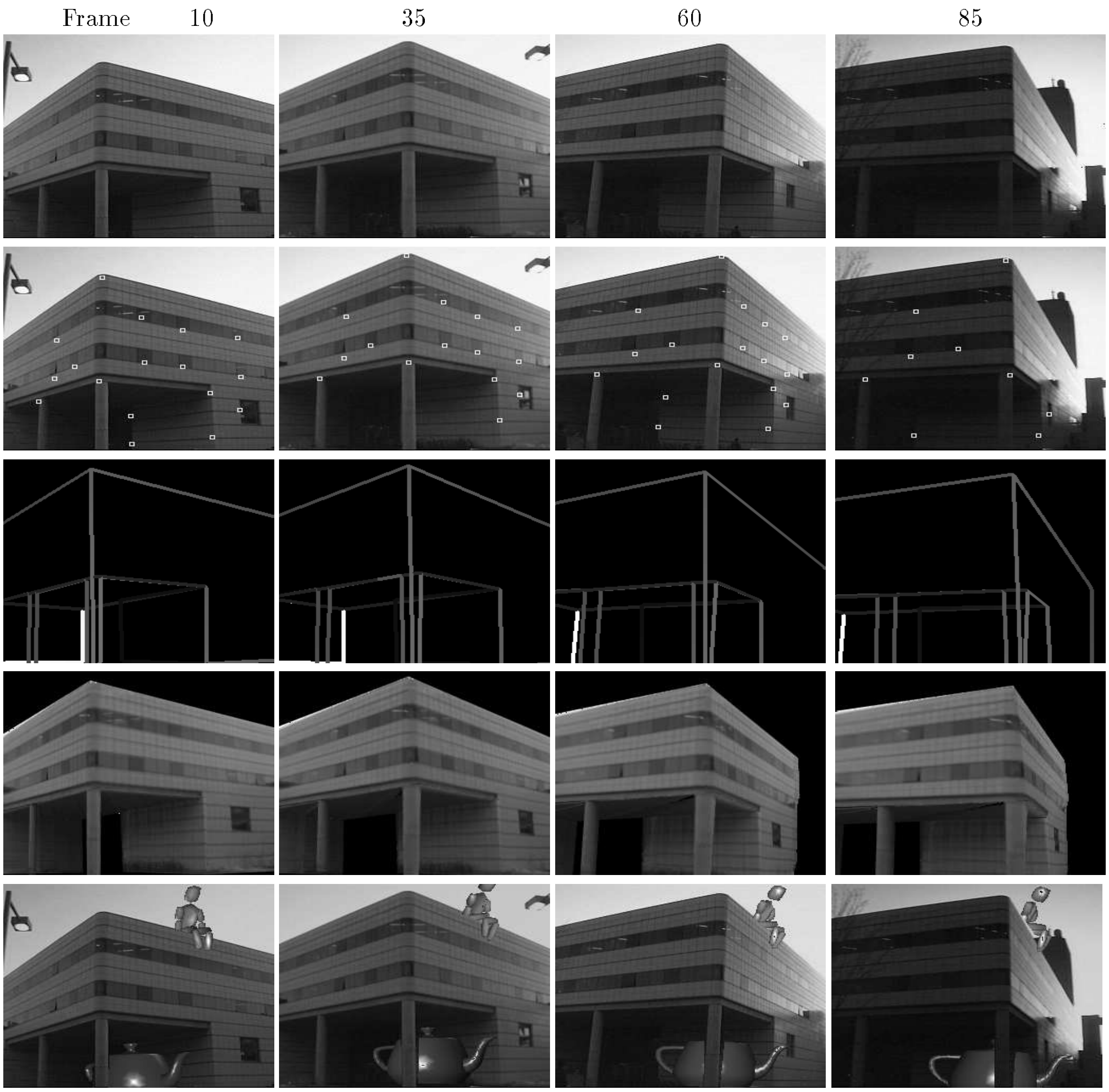


Figure 4-19: Recovering models from video. (a) Original frames. (b) The features are tracked in the video sequence using normalized correlation. (c) 3-D polygons are obtained by segmenting a 2-D image and back-projecting the vertices onto a 3-D plane. The plane for each polygon is computed from the recovered 3-D points corresponding to image features in the 2-D polygon. (d) Texture maps are obtained by projecting the video onto the 3-D polygons. The estimated motion and camera parameters are used to warp and combine the video from 25 separate frames to create the texture map for each polygon. (e) Recovered model and camera parameters are used to integrate 3-D computer graphics with the original imagery.

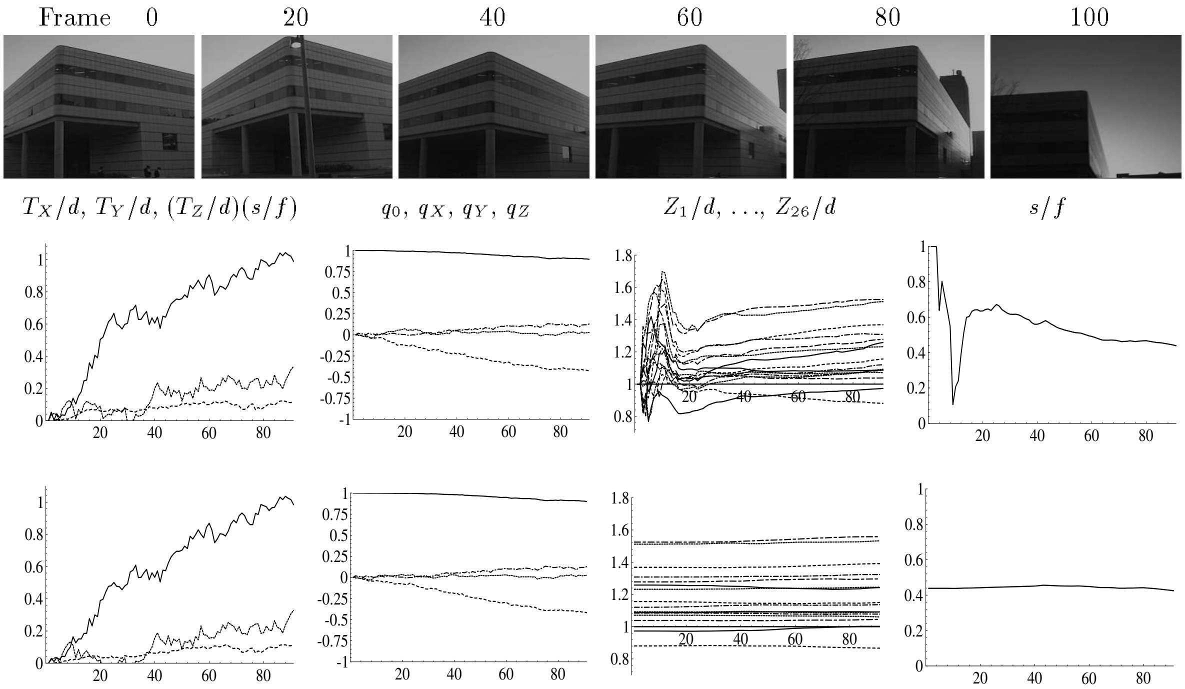


Figure 4-20: The dynamic parameter estimates for the Wiesner building example. The second estimate uses the final static parameters from the first as a starting point. The dynamic parameters are not greatly affected by the initial values of static parameters.

4.6 Performance analysis: Film post-production application

This application came about as the result of a real post-production problem in a major Hollywood studio and represents a possible alternative solution to the current established practice for composing real imagery with computer graphics in the film industry and entertainment industry at large.

Currently, filmed scenes are treated as merely 2-D images whereas computer graphics is usually done with 3-D models. Various techniques are used to integrate the filmed imagery with the rendered computer graphics imagery in the 2-D domain. But an alternative is to model the scene in 3-D so that integration can happen in the 3-D domain. Many integration problems become easier to solve this way.

Vision-based modeling of 3-D scenes from filmed imagery represents the key technology for 3-D integration. However, until recently, computer vision techniques have not been reliable or precise enough to achieve production-quality modeling results. Thus, the film industry has continued to rely on 2-D compositing techniques and, when the 3-D paradigm is used, has relied almost exclusively on skilled production artists to manually model imaged scenes using their own visual skills. The manual approach can produce good results but is laborious and time-consuming and therefore very expensive.

This section describes a system based on the modeling of Section 4.1 that has been used on a real Hollywood post-production project and has provided the precision and accuracy to meet the high demands of professional film production.

Frame 160 200 240 280 320 360 ...

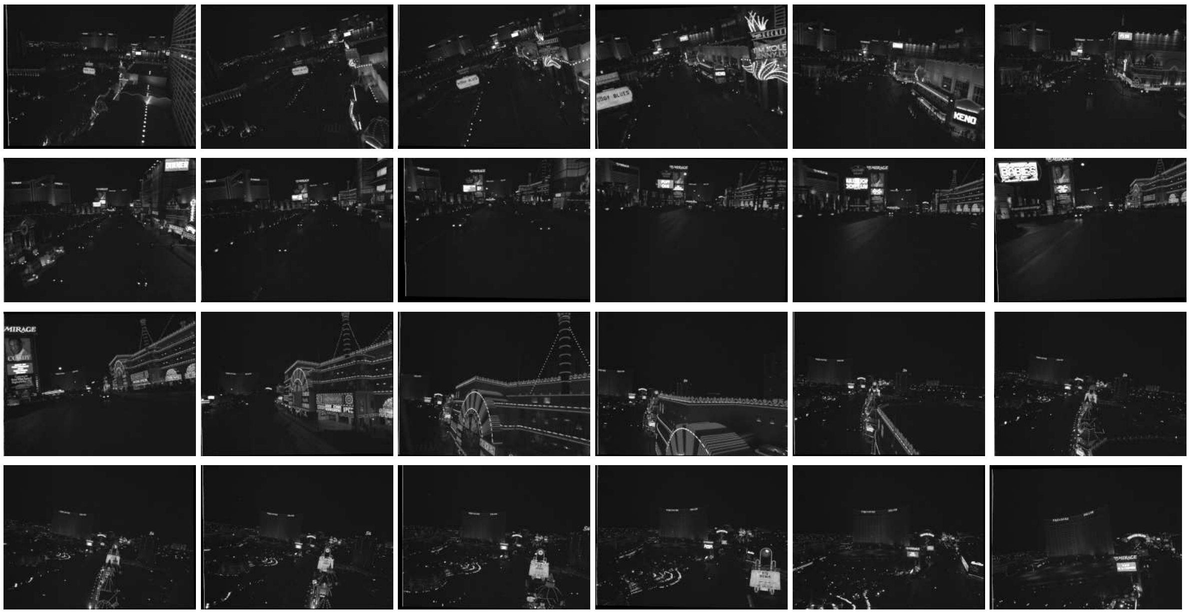


Figure 4-21: Some frames from one of the helicopter fly-through shots. This film shot, taken down the Las Vegas strip, is over one minute long and contains 1550 frames (at 24 fps).

4.6.1 Prototype system description

The system consists of a feature tracker and geometry estimator.

The feature tracker was designed by Rhythm & Hues Studios and consists of a manual keyframe selection interface and an automatic tracker. The user specifies a set of features in a number of keyframes and the automatic tracker fills in the rest.

The geometry estimator is augmented to deal effectively with the long sequences in which not all features can be identified in the first frame. In this case, new features are referenced to the first frame in which they occur and initialization bias is estimated as described in Section 4.1.3.

4.6.2 Example: Film post-production for a motion-based ride

The system is used in the production of a motion-based theme park ride, which consists of an audience situated on a large platform ("motion base" or "simulator") that moves in synchrony with the film footage projected on a large dome. The coordinated movement and visual stimulus is intended to present a rich and extremely realistic sensory illusion, but since the coordinated stimulus is so rich it must be done well or the ride will not be a pleasant experience.

In one scene of the ride, the desired illusion is of the audience being in a *Star Trek(TM)* shuttle craft flying through Las Vegas. For this scene, a helicopter-mounted camera was flown down the Las Vegas strip at 70 mph at altitudes ranging from 2 to 200 feet.

To produce the ride, the film imagery and motion base have to have a coordinated motion, which is complicated by the fact that the imagery must be accelerated. Large sudden motions must be removed for both audience comfort and dynamic limitations of the motion base, and yaw rotations must be removed because the platform does not yaw. Furthermore, 3-D computer graphics must be added to the imagery to realize the other

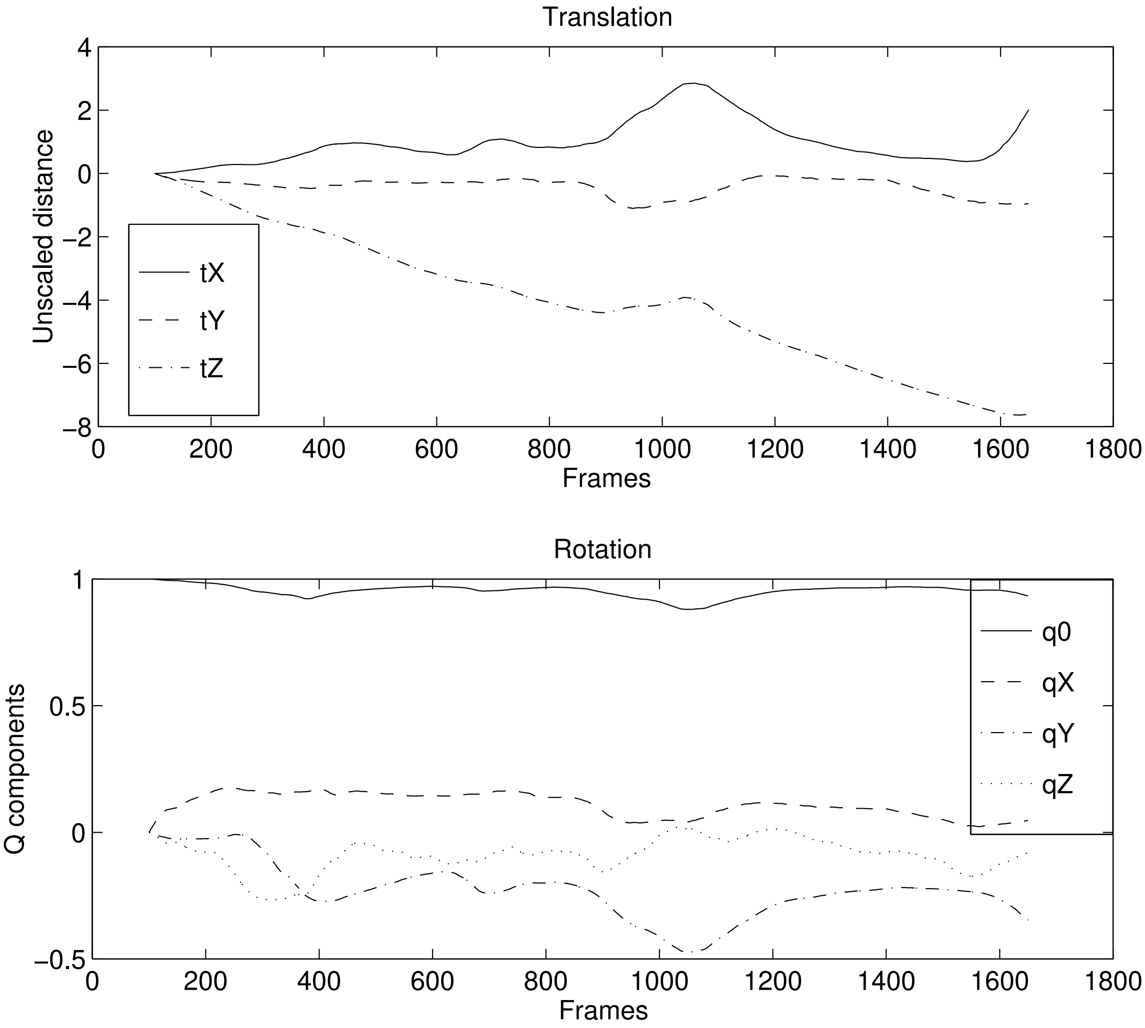


Figure 4-22: Finding the 3-D motion of a helicopter-based camera shot.

vehicles and explosions which are part of the ride.

These post-production goals present several challenges to the production studio. First, 3-D locations of objects in the scene must be recovered to situate 3-D computer graphics objects in the scene. Second, the 3-D camera motion must be recovered to program the motion base. And third, the 3-D motion of both the platform and the imagery must be altered in a coordinated way to meet the physical motion constraints of the ride. All of these problems can be solved using the 3-D geometry recovered from our point-based estimator.

Four shots required processing, the longest of which was 1700 frames, which is over 1 minute of film footage (24 fps). In this sequence, 35 keyframes are chosen (every 50th frame) and 10 features per keyframe are selected by the user.

The recovered 3-D motion, shown in Fig. 4-22 is used to derive the smoothed camera motion for the motion base and to correspondingly alter the imagery. In the general case, one needs to know the entire 3-D structure of the scene in order to properly alter the imagery in conjunction with an alteration of the 3-D camera motion. However, if only the rotational

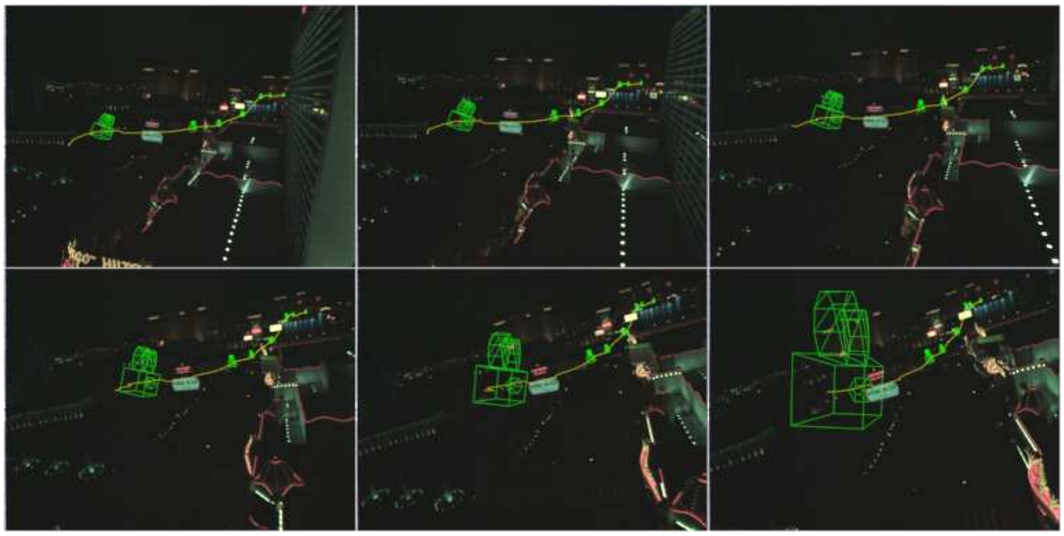


Figure 4-23: Depiction of the camera trajectory through the scene. The camera station of every 100'th frame is marked with a 3-D icon.

motion is being altered, scene structure is not required at all.

In our case, we needed to remove yaw (rotation about a vertical axis) and we needed to remove extreme accelerations in the imagery, which we assumed are due mostly to rotations of the mounted camera or rotation of the helicopter rather than translations. We have used the recovered rotation to obtain an acceptably smoothed trajectory and subsequently applied a 3-D rotational transform to the imaged scene to stabilize it in a way which will be coordinated with the motion base of the ride.

The recovered 3-D points are also used as keypoints in the scene to place other 3-D special effects into the imagery.

Chapter 5

Blob Geometry

This chapter develops a computational model for a new type of geometric features, called blobs. The use of blobs for representation of 3-D geometry was motivated by the ability to extract 2-D blob "features" efficiently and reliably in images. The moment-based representation of blob geometry leads to a probabilistic interpretation of 3-D geometry rather than the usual interpretation as surfaces or solids.

Since each 2-D blob feature contains geometric information about the position, orientation, and size of the corresponding 3-D object, it should be straightforward to use corresponding 2-D blobs to recover 3-D blob geometry much in the same way as 2-D point correspondences were used to recover 3-D geometry. Indeed, a physical interpretation and parameterization of blobs leads to formulation of a complete estimation model. The first section of this chapter develops a model for the case of an arbitrary number of cameras, N_C, and an arbitrary number of blob objects, N_O.

The second an third sections of this chapter focus on the driving application, which is a 3-D person tracker in which a pair of cameras watches a person and determines the 3-D motions. The performance analysis is geared towards this type of system in which $N_C = 2$. Simulations and typical results from the real-time system are presented for controlled experimental analysis and to characterize performance in the field.

5.1 Modeling

5.1.1 Blobs in 2-D

Two-dimensional blob observations are represented as 2-D spatial distributions in the image plane and parameterized using the low order sample moments of the distribution. The first moment is the mean of the distribution and represents the physical center of the blob in the image. The second (central) moments make up a 2-D covariance matrix which represents the dimensions and orientation of the blob feature.

Specifically, if an object occupies all 2-D points $q = (u, v)$ in a 2-D region $\mathcal{R}$, the first moment is the mean

$$\bar{q} = \int\int_{\mathcal{R}} dA\phi(q)dq \tag{5.1}$$

where dA is a differential area and ϕ is some mass distribution function. The mass distribution function can be chosen as uniform, or can be chosen to represent, e.g., the confidence of a spatial pixel location being part of the object.

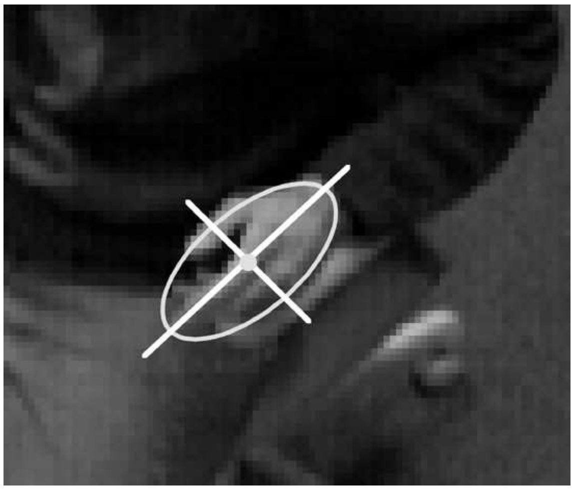


Figure 5-1: Two-dimensional moment-based blob modeling.

The mean clearly has two degrees of freedom

$$\bar{q} \doteq \begin{pmatrix} \bar{u} \\ \bar{v} \end{pmatrix} \in \mathbb{R}^2 \tag{5.2}$$

corresponding to the image coordinates of the mean.

The second (central) moment matrix

$$C_q = \int\int_{\mathcal{R}} dA\phi(q)(q-\bar{q})(q-\bar{q})^T \tag{5.3}$$

is called the covariance matrix. The covariance matrix

$$C_q = \begin{pmatrix} \sigma_u^2 & \sigma_{uv} \\ \sigma_{uv} & \sigma_v^2 \end{pmatrix} \tag{5.4}$$

contains sample variances of u and v on the diagonal and the sample covariance of u with v on the offdiagonals and is symmetric. It therefore has three degrees of freedom and can be represented by

$$\begin{pmatrix} \sigma_u^2 \\ \sigma_v^2 \\ \sigma_{uv} \end{pmatrix} \in \mathbb{R}^3 \tag{5.5}$$

as a 3-vector.

Since these are all independent parameters, we can build an observation model in which

$$y^{(2blob)} \in \mathcal{O}^{(2blob)} \equiv \mathbb{R}^5 \tag{5.6}$$

where

$$\begin{array}{rcl} \left.\begin{array}{r} \bar{u} \\ \bar{v} \end{array}\right\} & = & \text{image location of blob} \\ \left.\begin{array}{r} \sigma_u^2 \\ \sigma_v^2 \\ \sigma_{uv} \end{array}\right\} & = & \text{image orientation and size of blob} \end{array} \tag{5.7}$$

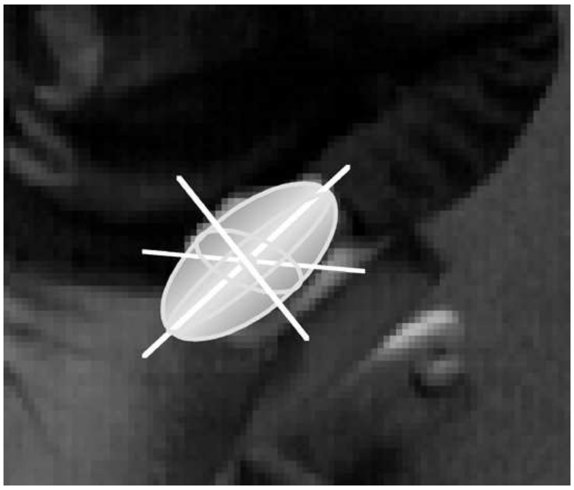


Figure 5-2: Three-dimensional moment-based blob modeling. The goal is to recover a 3-D blob representation of the object from multiple 2-D blob projections.

are the parameters of $y^{(2blob)}$.

5.1.2 Blobs in 3-D

Three-dimensional blob objects are represented as 3-D spatial distributions and parameterized using the low order moments of the distribution. The first moment is the mean of the distribution and represents the physical center of the object. The second (central) moments make up the covariance matrix which represents the dimensions and orientation of the object.

Specifically, if an object occupies all 3-D points p in a 3-D region $\mathcal{R}$, the first moment is the mean

$$\bar{p} = \int\int\int_{\mathcal{R}} dV \phi(p) p \tag{5.8}$$

where dV is an differential volume and ϕ is some mass distribution function.

The second (central) moment matrix is

$$C_p = \int\int\int_{\mathcal{R}} dV \phi(p)(p - \bar{p})(p - \bar{p})^T \tag{5.9}$$

and is called the covariance matrix.

By definition, the covariance matrix is symmetric and real, which means it can be diagonalized by orthonormal matrices [69], i.e.

$$C_p = \Phi D \Phi^T \tag{5.10}$$

where Φ is orthonormal and D is diagonal. The columns of Φ are eigenvectors of C_p and the associated diagonal elements of D are corresponding eigenvalues.

This decomposition has a physical meaning as the 3-D rotation of an axis-aligned distribution. The axis-aligned distribution has variances along the three coordinate axes equal to the diagonal elements of D and the three directional distributions are uncorrelated. The orthonormal matrix Φ can be thought of as a rotation matrix that rotates the axis-aligned distribution into the present coordinate system.

This decomposition yields a tripartite representation of 3-D blobs where diagonal matrix D represents the dimensionality of the object along object-centered principal axes, orthonormal matrix Φ represents the 3-D orientation of the object, and the mean vector $\bar{p}$ represents the location of the center of the object.

We can build a component state model for a single blob by combining a 3-D Euclidean parameterization for the dimensionality D with a 3-D Euclidean parameterization for the mean $\bar{p}$ and the manifold-tangent model of 3-D rotation $\mathcal{M}^{(rot)}$ developed in Section 3.7.2 (see Eqn. 3.195). That is,

$$\mathcal{M}^{(3blob)} = \mathcal{M}^3_{Euc} \times \mathcal{M}^3_{Euc} \times \mathcal{M}^{(rot)} \tag{5.11}$$

leading to the model specification

$\mathcal{M}^{(3blob)}$:

$$\begin{array}{rcl}
\mathcal{P}_S^{(3blob)} & = & \mathbb{R}^{10} \\
\mathcal{S}^{(3blob)} \subseteq \mathcal{P}_S^{(3blob)} & = & \mathbb{R}^6 \times \mathcal{S}^{(rot)} \\
\mathcal{P}_T^{(3blob)}(x), x \in \mathcal{S}^{(3blob)} & = & \mathbb{R}^9 \\
\mathcal{T}^{(3blob)}(x), x \in \mathcal{S}^{(3blob)} & = & \mathbb{R}^6 \times \mathcal{T}^{(rot)}(x) \\
\mathcal{Z}^{(3blob)}(x), x \in \mathcal{S}^{(3blob)} & = & I_6 \times \mathcal{Z}^{(rot)} \\
\mathcal{A}^{(3blob)} & = & +_6 \times \mathcal{A}^{(rot)}
\end{array} \tag{5.12}$$

where the ten parameters

$$\begin{array}{rcl}
\left.\begin{array}{c} d_x \\ d_y \\ d_z \end{array}\right\} & = & \text{dimensions of blob} \\
\left.\begin{array}{c} \bar{p}_x \\ \bar{p}_y \\ \bar{p}_z \end{array}\right\} & = & \text{scene location of blob} \\
\left.\begin{array}{c} q_0 \\ q_X \\ q_Y \\ q_Z \end{array}\right\} & = & \text{scene orientation of blob}
\end{array} \tag{5.13}$$

are the elements of $x^{(3blob)} \in \mathcal{P}_S^{(3blob)}$.

5.1.3 Complete blob-based 3-D geometry model

This section develops a complete manifold-tangent estimation model for the problem of observing N_O blob objects through a set of N_C static cameras. Thus the formulation is for multiple-baseline stereo and an arbitrary number of blob correspondences. In the applications and experiments of this thesis, we use two cameras, but the formulation is equally valid for more.

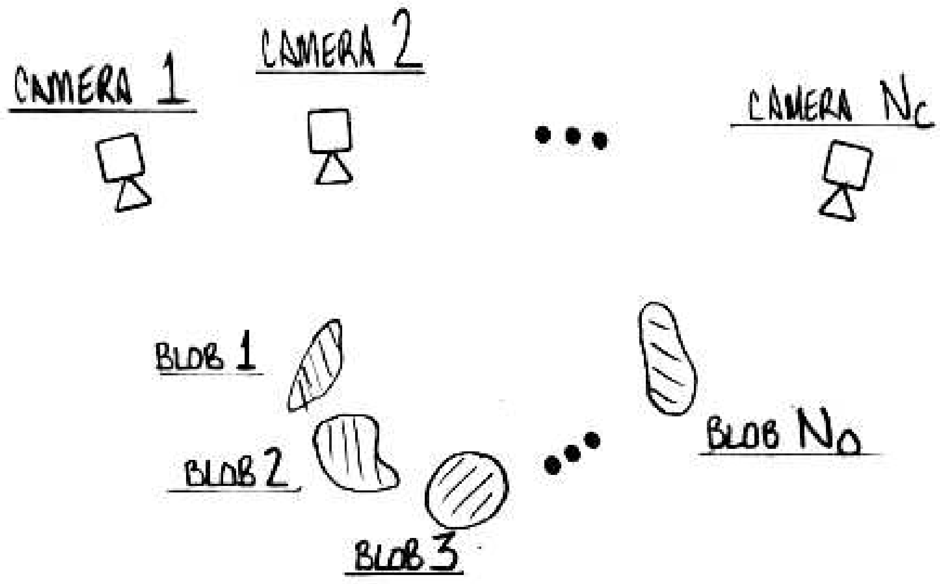


Figure 5-3: A system of N_C cameras and N_O blobs.

State model

The composite state model

$$\mathcal{M}^{(blobs)} = \prod_{k=1}^{N_C} \mathcal{M}_k^{(cam)} \times \prod_{i=1}^{N_O} \mathcal{M}_i^{(3blob)} \tag{5.14}$$

consists of a state model for each of N_C cameras and N_O blob objects where $\mathcal{M}^{(cam)}$ is from Eqn. 3.232 and $\mathcal{M}^{(3blob)}$ is from Eqn. 5.12.

Since $x^{(cam)} \in \mathbb{R}^8$ and $x^{(3blob)} \in \mathbb{R}^{10}$, then

$$x^{(blobs)} \in \mathcal{S}^{(blobs)} \subset \mathcal{P}_S^{(blobs)} \equiv \mathbb{R}^{8N_C+10N_O} \tag{5.15}$$

and has $7N_C + 9N_O$ total degrees of freedom.

Observation model

The observation vector

$$y^{(blobs)} = \begin{pmatrix} y_1^{(2blob)} \\ \vdots \\ y_M^{(2blob)} \end{pmatrix} \tag{5.16}$$

is a composite of M blob observations, where $y^{(2blob)}$ is defined in Eqn. 5.7.

Along with the observation vector is a correspondence function

$$C : \mathbb{Z} \mapsto \mathbb{Z}^2 \tag{5.17}$$

which maps the observation index $j = 1, \ldots, M$ to the corresponding blob object $i(j)$ and camera $k(j)$ such that

$$\left.\begin{array}{c} x_i^{(3blob)} \\ x_k^{(cam)} \end{array}\right\} \rightleftharpoons y_j^{(2blob)} \tag{5.18}$$

where

$$\left(\begin{array}{c} i(j) \\ k(j) \end{array} \right) = C(j) \tag{5.19}$$

expresses the index correspondence.

Constraint model

The external constraint function

$$e(x, y) = h(x) - y \tag{5.20}$$

decomposes into a forward model $h(x)$ subtracting the observation vector y and the forward model further decomposes into a parallel structure, one equation for each blob observation.

Explicitly,

$$e^{(blobs)}(x^{(blobs)}, y^{(blobs)}) = h^{(blobs)}(x^{(blobs)}) - y^{(blobs)} \tag{5.21}$$

where

$$h^{(blobs)}(x^{(blobs)}) = \left\{ \begin{array}{l} h^{(blob)}(x_{i(1)}^{(3blob)}, x_{k(1)}^{(cam)}) \\ \quad\vdots \\ h^{(blob)}(x_{i(M)}^{(3blob)}, x_{k(M)}^{(cam)}) \end{array} \right. \tag{5.22}$$

where

$$h^{(blob)} : \mathcal{M}^{(3blob)} \times \mathcal{M}^{(cam)} \mapsto \mathcal{O}^{(2blob)} \tag{5.23}$$

is the forward function mapping a 3-D blob object and a set of camera parameters to a 2-D blob observation.

The single-blob forward model is a composite

$$h^{(blob)} = h_p \circ h_t \circ h_b \tag{5.24}$$

including the blob transformation

$$h_b : \mathcal{M}^{(3blob)} \mapsto (\mathbb{R}^3 \times \mathbb{R}^{3\times 3}) \tag{5.25}$$

which maps blob parameters to a 3-D mean and covariance in world coordinates, the frame transformation

$$h_t : (\mathbb{R}^3 \times \mathbb{R}^{3\times 3}) \times \mathcal{M}^{(cam)} \mapsto (\mathbb{R}^3 \times \mathbb{R}^{3\times 3}) \tag{5.26}$$

which maps a 3-D mean and covariance in the world frame to a 3-D mean and covariance in the camera frame, and the projection

$$h_p : (\mathbb{R}^3 \times \mathbb{R}^{3\times 3}) \mapsto \mathcal{O}^{(2blob)} \tag{5.27}$$

which maps 3-D mean and covariance to 2-D observation parameters.

The blob transformation

$$(\bar{p}_W, C_W) = h_b(x^{(3blob)}) \tag{5.28}$$

is defined by the relationships

$$\bar{p}_W = \bar{p} \tag{5.29}$$

$$C_W = \Phi D \Phi^T \tag{5.30}$$

where $\bar{p}$ is the blob mean parameters from $x^{(3blob)}$

$$D = \begin{pmatrix} (d_X)^2 & & \\ & (d_Y)^2 & \\ & & (d_Z)^2 \end{pmatrix} \tag{5.31}$$

is a diagonal matrix of the squared blob dimensions and Φ is generated from the blob rotation quaternion according to Eqn. A.48.

The frame transformation from world to camera coordinates

$$(\bar{p}_C, C_C) = h_t\left((\bar{p}_W, C_W), x^{(cam)}\right) \tag{5.32}$$

is defined by the relationships

$$\bar{p}_C = T + R\bar{p}_W \tag{5.33}$$

$$C_C = RC_W R^T \tag{5.34}$$

where T is the translational component of $x^{(cam)}$ and R is the rotation matrix generated according to Eqn. A.48 using the rotation quaternion from $x^{(cam)}$ as the argument.

Finally, the projection

$$(\bar{q}, C_q) = h_p(\bar{p}_C, C_C) \tag{5.35}$$

is defined by the relationships

$$\bar{q} = h^{(proj)}(\bar{p}_C, x^{(cam)}) \tag{5.36}$$

$$C_q = JC_C J^T \tag{5.37}$$

where $h^{(proj)}$ is the projection equation Eqn. 3.219 and

$$J = \left.\frac{\partial h^{(proj)}(\xi, \zeta)}{\partial \xi}\right|_{\xi = \bar{p}_C} \tag{5.38}$$

is the Jacobian.

The internal constraint function is based on an identity state transition function, i.e.

$$\begin{aligned} x(t+1) &= f(x(t)) && (5.39)\\ &= x(t) && (5.40) \end{aligned}$$

which is exact for the static parameters of $x^{(cam)}$ and $x^{(3blob)}$ and expresses temporal coherence for the dynamic parameters. The extent of the temporal coherence is dictated by the variances in the $Q(t)$ matrix discussed below.

Error model

The state vector

$$x^{(blobs)} = \begin{pmatrix} x_1^{(cam)} \\ \vdots \\ x_{N_C}^{(cam)} \\ x_1^{(3blob)} \\ \vdots \\ x_{N_O}^{(3blob)} \end{pmatrix} \in \mathbb{R}^{8N_C+10N_O} \tag{5.41}$$

can be partitioned into N_C camera state vectors and N_O blob state vectors.

The error covariance matrix

$$P^{(blobs)} = \begin{pmatrix} P_1^{(cam)} & & & & & \\ & \ddots & & & & \\ & & P_{N_C}^{(cam)} & & & \\ & & & P_1^{(3blob)} & & \\ & & & & \ddots & \\ & & & & & P_{N_O}^{(3blob)} \end{pmatrix} \in \mathbb{R}^{(7N_C+9N_O)\times(7N_C+9N_O)} \tag{5.42}$$

can be correspondingly partitioned block-diagonally into camera covariance and blob covariance matrices.

Since the overall reference frame is arbitrary, one camera can serve as the reference frame for the others. Thus,

$$\begin{aligned} P_1^{(cam)} &= 0 && (5.43) \\ P_i^{(cam)} &\neq 0, \quad \forall i \neq 1 && (5.44) \end{aligned}$$

fixes the first camera frame and requires covariances on the positions of the remaining frames.

For each of the other frames, the camera covariance matrix

$$P_i^{(cam)} = \begin{pmatrix} P_i^{(int)} & & \\ & P_i^{(tra)} & \\ & & P_i^{(rot)} \end{pmatrix} \in \mathbb{R}^{7\times 7} \tag{5.45}$$

can be further partitioned into interior orientation, relative translation, and relative rotation. Although particular situations may provide better priors, the quite generic priors

$$\begin{aligned} P^{(int)} &= (\frac{1}{3}(s/f)_0)^2 \in \mathbb{R} && (5.46) \\ P^{(tra)} &= \begin{pmatrix} \sigma_p^2 & & \\ & \sigma_p^2 & \\ & & (\sigma_p(s/f)_0)^2 \end{pmatrix} \in \mathbb{R}^{3\times 3} && (5.47) \\ P^{(rot)} &= \begin{pmatrix} \omega_p^2 & & \\ & \omega_p^2 & \\ & & \omega_p^2 \end{pmatrix} \in \mathbb{R}^{3\times 3} && (5.48) \end{aligned}$$

where $(s/f)_0$ is initial focal parameter, $\sigma_p \in [2, 10]$, $\omega_p = 2/3$ express approximately zero initial knowledge of the values of the internal or relative orientation parameters.

The value of σ_p expresses that the scene footprint of every other camera is close to the first; if this were not the case, the cameras would not be looking at the same object and no correspondences would exist anyway. The value of rotation represents a standard deviation of about 40 deg; again, if the rotation were more than this, the camera would not be looking at the same object.

The blob submatrices

$$P_i^{(3blob)} = \begin{pmatrix} P_i^{(dim)} & & \\ & P_i^{(tra)} & \\ & & P_i^{(rot)} \end{pmatrix} \in \mathbb{R}^{9\times 9} \tag{5.49}$$

can also be decomposed into three 3×3 matrices representing the error covariance on blob dimensions, location and rotation.

To find the overall covariance, one can use the error analysis method described in Section 3.6 to project image error statistics into 3-D. Using the SVD pseudoinverse, values of infinity can be replaced by 10 for distances and 2/3 for rotations.

The dynamic constraints

$$Q^{(blobs)} = \begin{pmatrix} Q_1^{(cam)} & & & & & \\ & \ddots & & & & \\ & & Q_{N_C}^{(cam)} & & & \\ & & & Q_1^{(3blob)} & & \\ & & & & \ddots & \\ & & & & & Q_{N_O}^{(3blob)} \end{pmatrix} \in \mathbb{R}^{(7N_C+9N_O)\times(7N_C+9N_O)} \tag{5.50}$$

can be partitioned in a similar way where

$$Q_i^{(cam)} = 0 \in \mathbb{R}^{7\times 7} \tag{5.51}$$

because the cameras are assumed to be static.

Each blob covariance can be partitioned as well

$$Q_i^{(3blob)} = \begin{pmatrix} Q_i^{(dim)} & & \\ & Q_i^{(tra)} & \\ & & Q_i^{(rot)} \end{pmatrix} \in \mathbb{R}^{9\times 9} \tag{5.52}$$

where

$$Q^{(dim)} = 0 \in \mathbb{R}^{3\times 3} \tag{5.53}$$

$$Q^{(tra)} = \sigma_q^2 I \in \mathbb{R}^{3\times 3} \tag{5.54}$$

$$Q^{(rot)} = \omega_q^2 I \in \mathbb{R}^{3\times 3} \tag{5.55}$$

and

$$\sigma_q^2 = (\frac{1}{3}(\text{MAXVEL})\Delta t)^2 \tag{5.56}$$

and

$$\omega_q^2 = (\frac{1}{3}(\text{MAXROT})\Delta t)^2 \tag{5.57}$$

as in Section 4.1.2.

The measurement error covariance

$$R^{(blobs)} = \begin{pmatrix} R_1^{(2blob)} & & \\ & \ddots & \\ & & R_M^{(2blob)} \end{pmatrix} \tag{5.58}$$

can be partioned into 5×5 submatrices

$$R_j^{(2blob)} = \begin{pmatrix} \sigma_r^2 & & & & \\ & \sigma_r^2 & & & \\ & & \sigma_r^4 & & \\ & & & \sigma_r^4 & \\ & & & & \sigma_r^4 \end{pmatrix} \tag{5.59}$$

corresponding to each blob, $j = 1, \ldots, M$.

The σ_r value is related to the pixel accuracy and resolution

$$\sigma_r = \frac{1}{3} \frac{\text{PIXACC}}{\text{PIXRES}} \tag{5.60}$$

as in Section 4.1.2. Since the first two parameters of the measurement are distances and the last three are square distances (spatial covariances), the error variances for the second three parameters are squared variances.

The model error covariance matrix, S, can be set to zero. Alternatively, S can be used to add more variance to measurements when they are further toward the periphery because the linearized model is less accurate in those regions.

5.2 Performance analysis: Simulations

The purpose of the simulation experiments in this section is to empirically evaluate the performance of the model and estimator in recovering 3-D blob geometry from 2-D blob correspondences.

5.2.1 General experimental procedure

The experiments in this section all consist of two (virtual) cameras and a single blob. There is little loss in generality because each blob is independent of other blobs (as long as blobs do not occlude each other) and the performance for any set of more than two cameras will be equivalent or better than the performance for the configuration of any pair in the set.

The general procedure is to instantiate a pair of virtual cameras and a 3-D blob and to project the 3-D blobs into the 2-D images to obtain blob regions. Noise is added to the measurements and fed to either a GLM algorithm (see Section 3.3.3) for self-calibration or a GEKF algorithm (see Section 3.5.3) for dynamic blob tracking.

For all of these experiments, the generative system consists of two cameras with normal video-camera focal parameters placed about one meter apart and covering about a two-meter cubic volume centered about two meters in front of the cameras (see Fig. 5-4). These parameters closely resemble the parameters of the real imaging system used in the applications below for tracking people. Since ultimately the blob estimation is geared towards this type of practical application, this regime of parameters is the most interesting to examine at this time. A more favorable estimation results if the cameras have a better triangulation

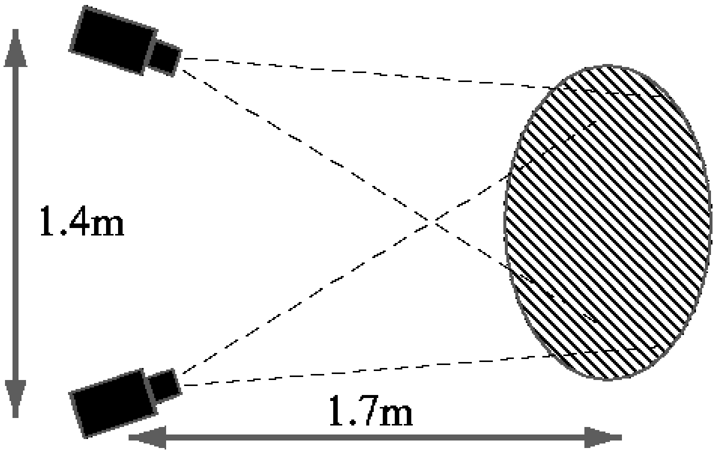


Figure 5-4: Experimental apparatus for blob geometry simulations. The baseline and depth of object is typical of a wide baseline camera system used for tracking a person sitting at a desk. The shaded area represents the viewable volume visible to both cameras.

angle, but this configuration is less practical.

5.2.2 Experiment 1: Self-calibration, relative orientation

The purpose of this experiment is to evaluate the performance of a GLM estimator with the blob manifold-tangent model $\mathcal{M}^{(blobs)}$ for calibrating the relative orientation of the two cameras. The expected result is that the relative orientation will be estimated well enough to accurately reconstruct blobs in the training volume.

Procedure

The procedure is to generate several trials and statistically evaluate the accuracy and the convergence properties. The procedure for each trial is to generate simulated images for a number of blob positions and orientations randomly chosen in the 3-D volume. The blob dimensions for each trial are chosen at random.

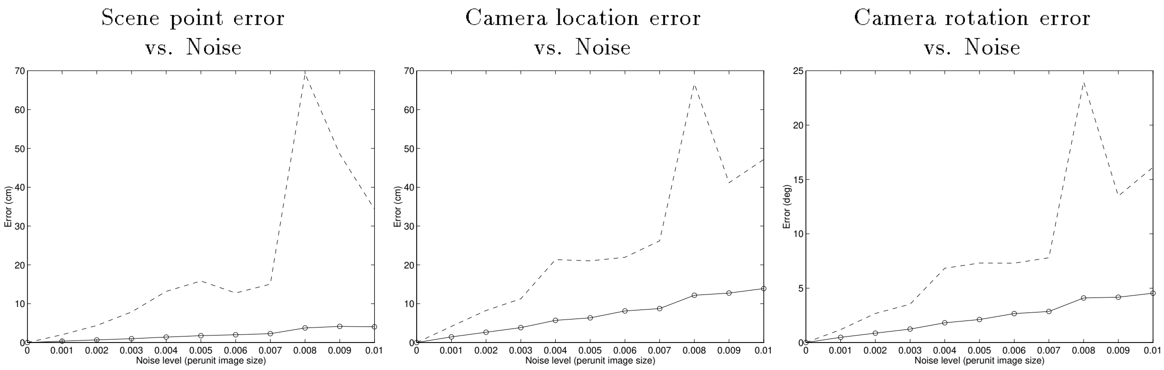
Two experiments were performed, the first with noise level as the control variable and the second with number of blob correspondences as the control variable.

For the noise level experiment, eleven noise levels were chosen, from zero to .010. The noise level represents the interval width of roughly uniformly distributed pseudorandom noise added to each coordinate of the mean of the blob image. The units are distance ratios of image coordinates to image size, i.e. $(u/s, v/s)$. For a 640×480 video image, the noise level of .010 corresponds to 6.4 pixels.

For each noise level, 100 trials were performed in which 20 blobs were randomly chosen in the 3-D viewing volume and the relative orientation was computed for the two cameras.

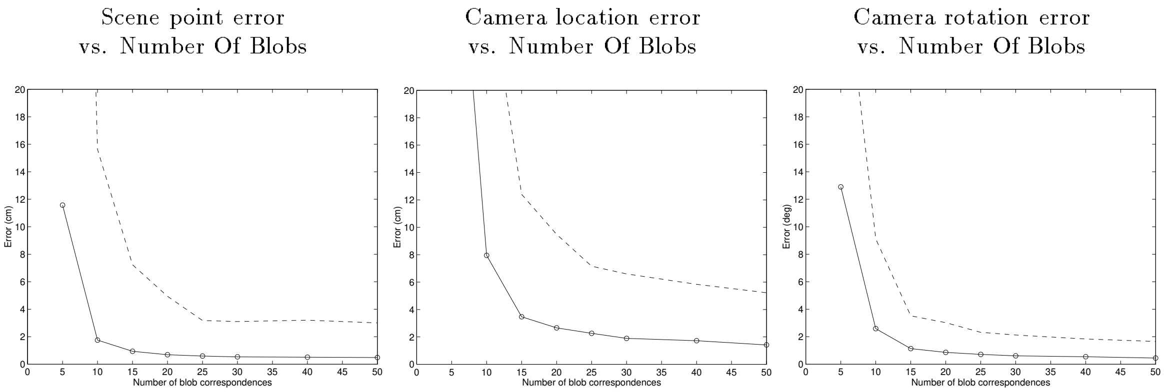
For the correspondence experiment, eight levels of correspondences were considered, from five blob correspondences to fifty blob correspondences. The noise level was chosen for all experiments to be .002. Again, 100 trials were performed at each level of correspondences.

Results



(a) (b) (c)

Figure 5-5: Blob relative orientation statistical error performance versus noise level. Over 100 trials the solid line is RMS error in centimeters, the dashed line is the maximum error. The noise level is reported in standard deviation relative to image size; a typical video processing error is .003 (σ = 2-pixels in a 640 image). Relative orientation error is reported in RMS error (a) in cm over all 3-D point locations, (b) in cm over the camera location, and (c) in deg over the camera rotation.



(a) (b) (c)

Figure 5-6: Blob relative orientation statistical error performance versus number of blob correspondences. Over 100 trials, the solid line is RMS error in centimeters, the dashed line is the maximum error. The number of blobs is the number of correspondences collected before computing the relative orientation. Relative orientation error is reported in RMS error (a) in cm over all 3-D point locations, (b) in cm over the camera location, and (c) in deg over the camera rotation.

The experimental results are summarized in Fig. 5-5 (noise level) and Fig. 5-6 (correspondence level). Each figure contains three plots, one for scene point reconstruction error, one for camera location error, and one for camera rotation error. For each plot, a solid line reflects the RMS error over the trials and a dashed line reflects the maximum error observed over the trials.

For the noise level experiment, the results show an expected gradual degradation of precision with increasing noise level. The dashed line indicates that even at low noise levels there is the opportunity for occasional poor results. The severity of the occasional poor result increases with increasing noise.

A qualitative analysis of the configurations of the failure cases reveals that most of these are due to particularly poor random configurations of blobs, such as all blobs clustered near one another or very nearly in a line.

The overall level of precision of scene reconstruction for reasonable noise levels can be summarized as being on the order of 1cm–2cm in successful cases. Success appears to be achievable by avoiding degenerate correspondence sets, which is a reasonable practical requirement. Although the camera location and orientation errors are slightly higher, it is usually the scene reconstruction that one is ultimately concerned with.

For the correspondence level experiment, poor performance is expected when only five correspondences are available (six are required in theory) and diminishing returns are expected when many more than twenty or so are available. Again, the occasional poor result, as indicated by the level of the dashed line, is typically due to a poorly chosen (random) configuration of blobs.

For a static camera system, a large number of correspondences can be collected over time. In a person-tracking system with several parts of the body being tracked, hundreds of correspondences can be obtained in a matter of seconds, so achieving the critical number is not at all the challenge. Rather, the challenge in collecting correspondences seems to be in achieving a well-distributed sampling of 3-D points. This lesson is carried over into the calibration procedure for the person tracking system described in the next section.

5.2.3 Experiment 2: Shape estimation

The purpose of this experiment is to evaluate the static shape parameters of a single blob from a pre-calibrated camera pair. The expected result is that the 3-D blob means will be stable and accurate but that there will be significant errors in the 3-D blob covariances in the direction of the optical axes of the cameras. Since the covariances are a composite of the blob dimensions and rotation, both sets of parameters will suffer.

However, since the blob dimensions are static and the orientation is dynamic, the errors should be separable in a dynamic estimation setting. In particular, once the static shape parameters converge, the physical rotation parameters should improve as well.

Procedure

The procedure here is similar to the previous experiments. The same nominal configuration as above (1.4m baseline) is used, except the baseline is used as a control variable and is varied from .2m to 2.0m. One thousand trials are performed at each baseline for each of two cases.

In the first case, all blob parameters are chosen at random, including the three blob dimensions independently, and all parameters are estimated with no initial knowledge of blob parameters. In the second case, the blob proportions are assumed known and only the motion parameters of the blob are free.

We expect to see a great improvement in the rotation parameters in the second case,

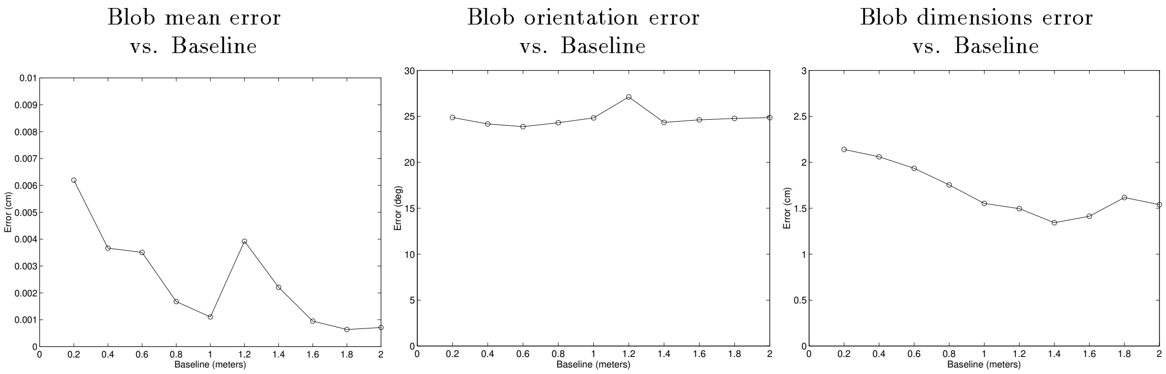


Figure 5-7: CASE 1: Blob shape and motion statistical error performance versus baseline. The errors are reported as RMS error in centimeters for mean and dimensions and in angular degrees for rotation. In this experiment, blob dimensions and blob orientation are unknown.

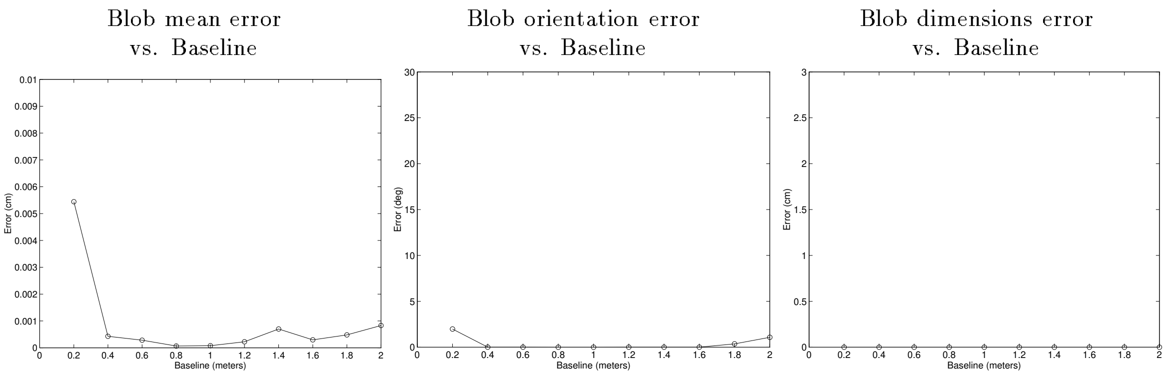


Figure 5-8: CASE 2: Blob shape and motion statistical error performance versus baseline. The errors are reported as RMS error in centimeters for mean and dimensions and in angular degrees for rotation. In this experiement, blob dimensions are known (thus zero error) and orientation performance is improved.

but, due to ambiguities in parameters arising from blob symmetry, additional manipulation must take place in the procedure for the second case to accurately measure the results.

Ambiguous shape parameters result from two aspects of symmetry associated with blob objects. The first is that when any two dimensions of the blob are close to each other, any rotation about the orthogonal axis does not change the physical blob shape. Thus the correct interpretation is ambiguous. The second is that, even with unequal proportions along the axes, there is a discrete set of coordinate axis renamings which can result in the same object, due to mirror symmetry of the object about each of its axes.

For these experiments, the first source of ambiguity is removed by constraining the shape to be in a 1:.75:.5 ratio; only the overall size of the blob is chosen at random. The second source of ambiguity is removed by performing an analysis of all 24 permutations of axes that result in an equivalent shape, and choosing the one with the smallest angular deviation from the actual parameters.

Results

The experimental results are summarized in Fig. 5-7 (unresticted motion and shape estimation) and Fig. 5-8 (motion estimation with known shape proportions). Each figure contains three plots, one each for blob location, blob orientation, and blob shape. The solid line represents RMS error over 1000 trials for each baseline. The two sets of plots (Fig. 5-7 and Fig. 5-8) are scaled the same for comparison. (In Fig. 5-8 the dimension errors are identically zero because shape proportions are known.)

In both cases, the blob location is exceedingly precise (fractions of millimeters) as might be expected when the cameras are known exactly, but the interest in these experiments is primarily in investigating the dimension and orientation estimation. As predicted, the orientation and shape parameters are very poor in CASE 1, largely due to symmetry ambiguities and local minima created due to these ambiguities, as predicted above.

Qualitative analysis of the particularly poor estimation trials reveals several reasons for failure of precise estimation. First, the z-dimension is often poorly leveraged by the measurements, resulting in a shape which explains the measurements well but does not represent well the physical 3-D volume that originally generated those measurements. In other words, the estimated 3-D covariance is simply wrong, but its projections into the two images are very close to those of the actual covariance. In this case, the poor shape estimation is reflected in both the dimension and orientation parameters.

Second, randomly chosen blobs often have uniform dimensionality in two or more axes. As discussed above, this can result in shape parameters which physically represent the actual 3-D object well, but do not correspond well to the actual parameters because of ambiguity. In other words, the 3-D covariance is physically correct, but the axes are rotated so different parameters result.

Third, mirror symmetries occur, and are mostly filtered out by analyzing equivalent permutations of axes as discussed above, but when other symmetries distort the parameters, this procedure can result in further errors. In this case, the 3-D covariance is again physically correct, but the representation is different because of flipped axes.

In CASE 2, many of these sources of errors are eliminated because ambiguities are removed with the known shape proportions. As a result, the orientation error is greatly improved.

The implication of CASE 2 is important for estimation in a dynamic setting. Since the shape parameters are static, a finite number of observations should result in a good shape estimate for each blob object. If shape estimation is successful, dynamic estimation of blob motion estimation should therefore be much more like CASE 2 than like CASE 1.

5.3 Application: Real-time self-calibrating person tracker

Finally, some quantitative results from the real-time system are presented to indicate performance in the field.

5.3.1 Prototype system description

The STIVE system is depicted in Fig. 5-9 and consists of a pair of cameras, each sending its video signal to an SGI Indy (200MHz R4400) computer. Each computer runs a simplified version of the PFINDER [84] program (FFINDER), which obtains spatial distributions in the image for the face and hands based only on a color classification. The 2-D blobs produced by these two programs are the input to a third program called SPFINDER which computes the 3-D blob geometry based on the 2-D blobs.

Since the camera configuration is essentially static, the real-time system for tracking people was built under the assumption of a calibrated rig to simplify the processing. The

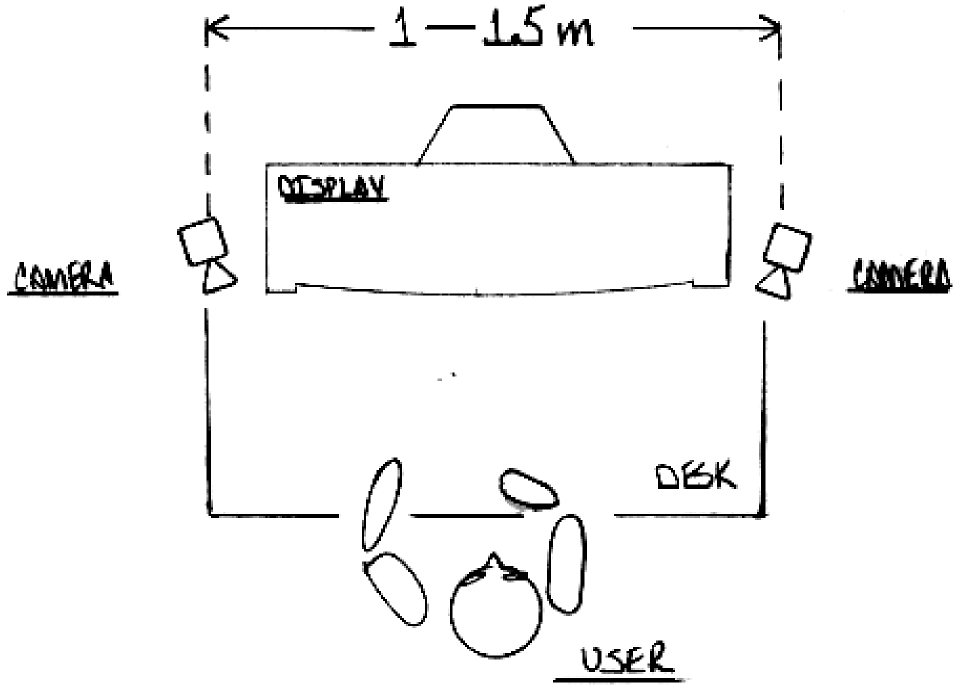


Figure 5-9: STIVE: STereo Interactive Video Environment. A wide-baseline stereo system utilizes two cameras and tracks image blobs representing parts of the person from each camera. The correspondences yield 3-D blobs in real time. The person's 3-D motions are used to allow interaction with characters in a virtual environment, viewed through the video display.

self-calibration ability of the formulation was designed into a second real-time system which is run one time when the STIVE system is set up to obtain the calibration. Both systems utilize the FFINDER front end for blob tracking, described below. Use of the blobs for obtaining 3-D geometry in the calibration and the tracking systems is described in the respective sections below.

Two-dimensional blob tracking: FFINDER

The simplified FFINDER program ('f' for 'fleshtone') discards the background removal and heirarchical clustering of PFINDER [84] and uses a single Gaussian color distribution to do single-class pattern classification [74] of human fleshtone for finding face and hands. The result is fast (up to 30Hz) feature processing of (320,240) images of moving people.

Such a color classification technique has proven to be quite robust and race-independent [62, 84, 19]. The color class that results from training on a variety of races of people has a small variance in chrominance and a large variance only in brightness. For example, in YUV color space, the luminence Y has a large variance and the chrominance subspace UV has small variances.

The color class is trained by taking sample pictures of people's hands and faces and computing the sample mean and covariance in color space. There are many ways of approaching the color modeling and classification, but a simple Gaussian model in YUV space has proven to work acceptably for a wide range of people.

More sophisticated color class modeling and classification would improve the reliability and precision of the system further but the lesson, of course, is that unlike point, line, or contour features, even such a simplistic blob feature tracking system has proven to work reliably enough so that hundreds of people have been able to use it to interact in real time [12] and the processing can be done using only standard commercial computers and cameras.

The FFINDER program is self-bootstrapping and contains two processing modes for real-time efficiency. The main program,

FFINDER program:

```
mode = INIT_MODE;
for( int frameCount = 0;; frameCount++ )
    {
        image = grab();
        switch( mode )
            {
                case INIT_MODE:
                    if( ! init( image ) )
                        mode = TRACK_MODE;
                    break;
                case TRACK_MODE:
                    if( track( image ) || collide() )
                        mode = INIT_MODE;
                    break;
            }
    }
```

contains an initialization mode and a tracking mode. The initialization mode assumes only that there is a single person in the scene and uses color classification along with some heuristics about hands and faces to segment out and label the three fleshtone entities. The tracking mode uses the previous location of each entity to efficiently find its current member pixels. The spatial moments are computed for each entity resulting in the blob representation of each hand and face.

The `init()` function fails when it cannot cluster three fleshtone objects, which is in practical terms due either to there being no person in the scene (too few blobs), there being an occlusion (too few separate blobs), or there being more than one person in the scene (too many blobs). In these cases, the system remains in initialization mode until it successfully finds three blobs which meet the heuristics for being face and hands. When it does, it switches to tracking mode. Each FFINDER system using (320,240) images runs at 30Hz when it is consistently in tracking mode, at 5–10 Hz when it is consistently in initialization mode.

The system remains in tracking mode until one or more blobs cannot be found or if two blobs begin to substantially occlude each other. The `collide()` function determines when two blobs are no longer distinguishable due to occlusion.

Three-dimensional blob tracking: SPFINDER

The tracker assumes the system has been self-calibrated and thus the camera geometry is known. The job of the system, then, is to estimate the shape of the 3-D blobs and track their 3-D motions.

The 3-D blob tracking program,

SPFINDER program:

```
for( int frameCount = 0;; frameCount++  )
    {
        ffinderClient.getLeftPerson();
        ffinderClient.getRightPerson();
        for( int which = 0; which < 3; which++ )
            {
                if( ! inited[ which ] )
                    {
                        computeInit( which );
                        regularizeBlob( which );
                        setCovariances( which );
                        inited[ which ] = 1;
                    }
                estimate( which );
            }
    }
```

computes the 3-D blob parameters from the 2-D blob moments. There is an initialization for each blob which consists of a static shape estimate using the GLM algorithm, followed by a heuristic regularization (each blob is cylinder-like, with one long dimension and two equal shorter dimensions). After that, a GEKF is used to refine the shape and track the 3-D motion.

5.3.2 Example: Self-calibration and tracking of a moving person

For self-calibration, corresponding 2-D blobs are collected from the two images over time and used to compute the 3-D camera geometry. To get a good set of correspondences for calibration, the person should ideally move his hands and head in order to maximally fill the view intersection of the two cameras. As revealed in the synthetic experiments of the previous section, degenerate clusters of blobs can result in a poor estimation. To avoid redundant measurements that can lead to a poor distribution of blobs, the system is programmed to wait for a significant motion (in the images) before collecting the next sample.

The process of generating correspondences for a self-calibration is illustrated in Fig. 5-10 in which the top two images show a stereo pair from the sequence overlaid with the entire sequence of correspondences (the center of each blob is indicated with a white square).

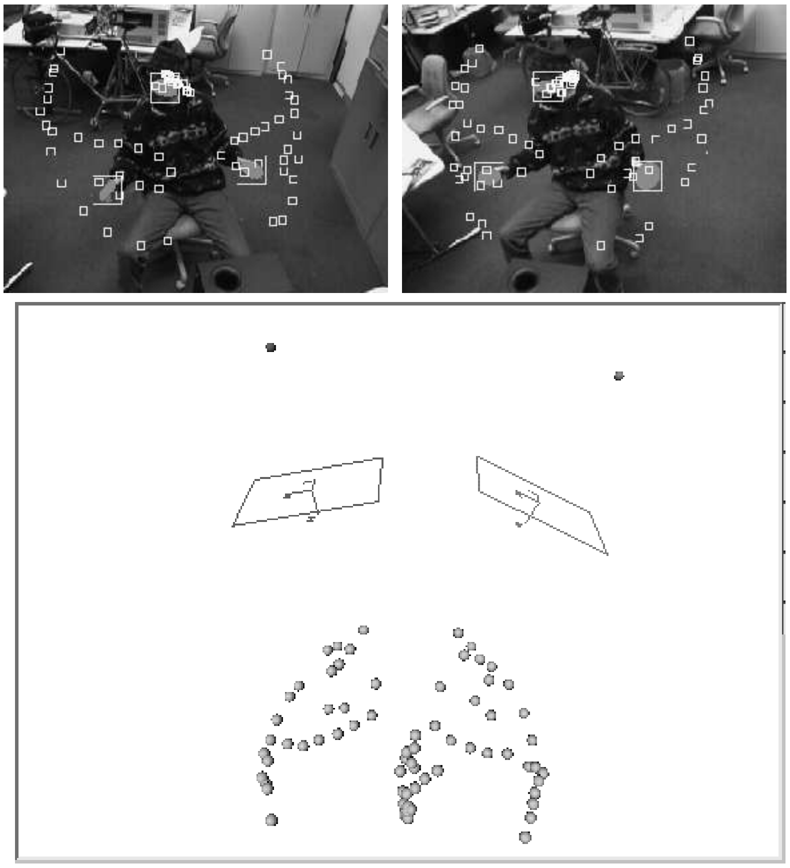
The result is an arbitrarily large set of 3-D blob correspondences. The manifold-tangent model for blobs can be used with the GLM algorithm (see Section 3.3.3) to estimate the camera parameters as a batch optimization. The bottom illustration of Fig. 5-10 depicts a top view of the 3-D relative orientation of the cameras and the 3-D locations of the tracked hands and head that have been recovered from that calibration sequence.

Absolute error analysis is difficult because the real parameters are unknown and not easily measured, but we can do a residual error analysis, we can evaluate absolute 3-D precision to some degree, and we can measure consistency.

To measure residual error, the self-calibration procedure was repeated for a number of trials, and the mean error of the 2-D blob center locations over all frames and all blobs was computed. The residual error over 20 trials, each of which consisted of 3 blobs (face and hands) in 30 frames (90 total correspondences per trial) was $.0044s$, where s is the width of the image. This translates to 1.4 pixels in a half-resolution (320,240) video image.

To measure absolute 3-D precision, the recovered self-calibration parameters were used

Stereo sequence for self-calibration



3-D view of self-calibration sequence

Figure 5-10: The blob representation can be used to facilitate stereo self-calibration. Here we illustrate the self-calibration of a stereo rig in real time, simply from watching a person moving. The stereo pair shows the feature tracks on the person. The 3-D view shows a roughly overhead view of the space including the recovered cameras and the 3-D feature tracks. RMS residual error is 1.5 pixels; RMS 3-D errors are on the order of 2.25cm.

to reconstruct the trajectory of the right hand moving a known distance in a straight line. The data was collected by moving the hand a fixed distance along the straight edge of a table. The data was analyzed by fitting the 3-D blob centers to a straight line and computing the length of the line segment and the RMS residual error of the fit. The RMS distance error is then scaled to the known length to obtain a RMS 3-D precision measure relative to a real distance. The 3-D precision obtained over 20 trials of self-calibration followed by data collection and analysis results in a mean RMS 3-D error of 2.4cm. Each trial consisted of 40 correspondences and blob reconstructions.

To measure consistency, a number of self-calibration trials were conducted with a static camera configuration. Precision results for each of the Euclidean parameters is obtained by computing the mean and covariance of the resulting estimates over all trials. Precision results for the rotation are obtained by iteratively searching for the "mean" rotation and computing rotation statistics in tangent space; this is done by computing second-order error statistics in the tangent parameter space of a starting guess (obtained by normalizing the

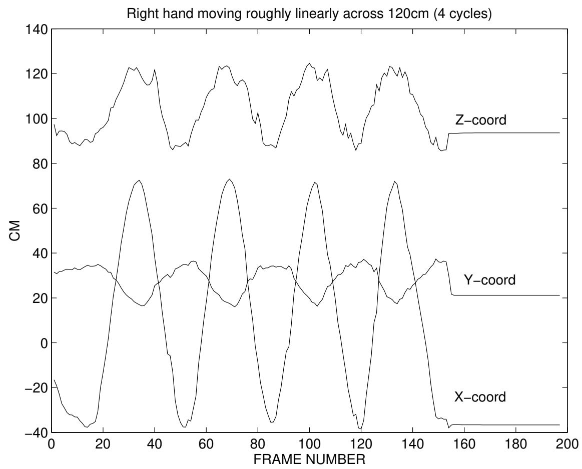
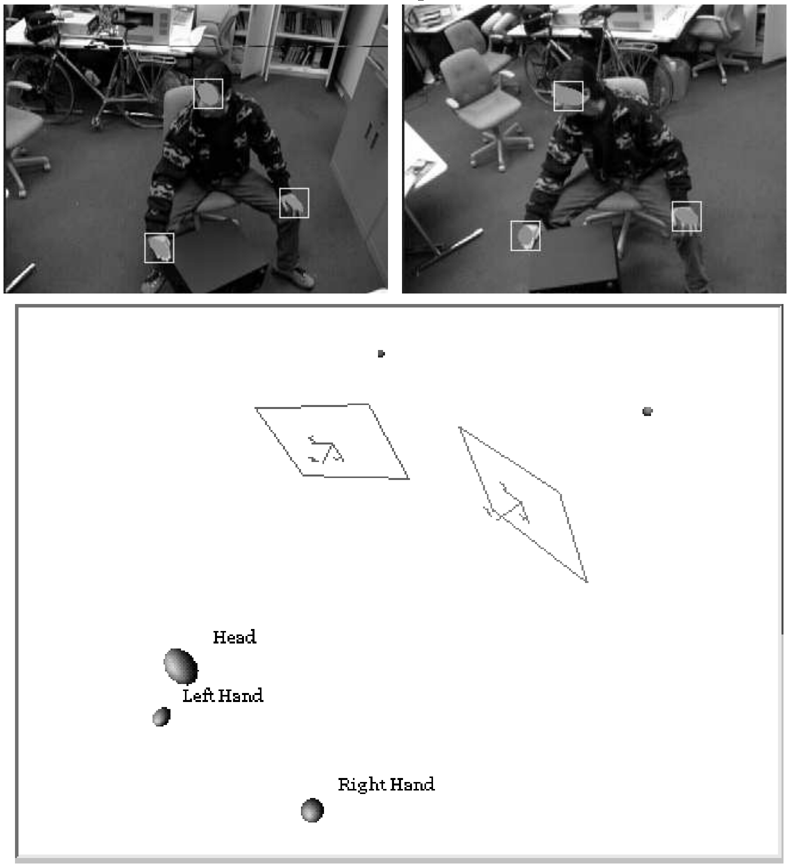


Figure 5-11: Translation of the right hand back and forth along a linear trajectory after self-calibration. Reconstruction of hand position has an RMS error of 2.25cm, resulting in a relative error of 1.8% over this trajectory.

mean quaternion) and performing an iterative search for the least square error in tangent parameters. Over 20 trials, the RMS residuals are 3.2cm in camera location and 1.8deg in camera orientation.

Stereo pair



3-D estimate

Figure 5-12: Real-time estimation of position, orientation, and shape of moving human head and hands. We find RMS errors of 1.5cm, 2 degrees, and 5% on translation, rotation, and shape, respectively along a linear 3-D trajectory.

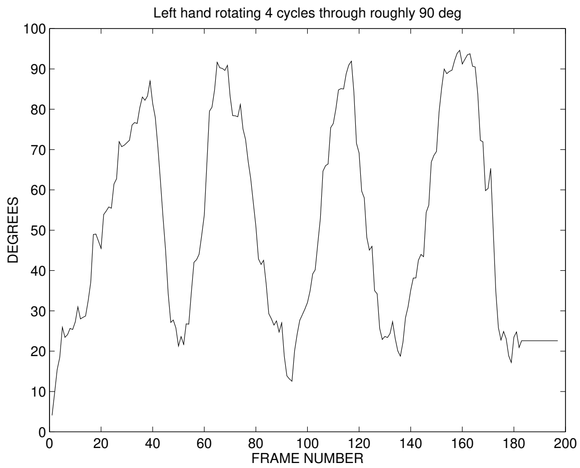


Figure 5-13: Rotation of the left hand back and forth through roughly 90 deg. An analysis of the jitter in the angular signal results in measures of 2 deg RMS error, or roughly 2.2 percent relative error.

Chapter 6

Conclusion

This thesis has explored some of the issues associated with making practical computer vision systems based on 3-D geometry estimation. The paradigm chosen for study was that of feature correspondences in multiple displaced camera images because it involves a formulation that requires only weak internal models of geometric primitives, which, in this study, are points and blobs.

The classic problem of point-based 3-D geometry estimation was taken up as a means of understanding the nonlinear systems associated with vision-based geometry. The basic insight is that not only is the mapping of 3-D points to 2-D points nonlinear, but the 3-D domain is also not a Euclidean space.

There are well-founded provably convergent optimization techniques for linear systems on Euclidean domains that apply to paradigms of both batch estimation (pseudoinverse methods) and recursive estimation (Kalman filter, recursive least squares). There are natural extensions of both of these to nonlinear systems on Euclidean domains for both batch estimation (modified Newton's methods) and recursive estimation (extended Kalman filter). In order to accommodate the non-Euclidean domains associated with 3-D geometry problems, this thesis has developed further extensions that generalize these local linearization methods to systems with non-Euclidean domains, resulting in a generalized Levenberg-Marquardt algorithm and a generalized extended Kalman filter.

Both of these generalizations are a result of modeling the non-Euclidean domain as a manifold embedded in a Euclidean space and modeling the system locally as a Euclidean domain homeomorphic to the local tangent hyperplane. This manifold-tangent model developed in this work allows simple natural extensions of established iterative search and recursive probabilistic optimization techniques.

The idea of modeling local perturbations in a tangent hyperplane is not a new one, as it is the basis for so-called projection methods for solving "nonlinearly constrained nonlinear optimization" problems. However, the idea of moving the hyperplane calculations from being the responsibility of the estimator designer to the responsibility of the model designer is an important practical issue. Many estimation failures occur because of poor conditioning of the specific model that is handed to the general-purpose estimation technique. Unfortunately, the estimation problems can only be fixed by modifying the model, not by modifying the estimator. Thus, the important topological issues of the system that affect conditioning should properly be handled in the modeling phase, not by the estimator.

This emphasis on modeling is the design philosophy that leads to the key result on point-based estimation. The problem of estimating 3-D geometry from point correspondences stably and accurately has long been considered a difficult problem, but with the well-conditioned model developed in Chapter 4, not only can the traditional structure and motion parameters be estimated well, but additional parameters such as focal length of the camera can be estimated as well.

The point-based estimation system that arose from this work is the basis for several prototype applications which were developed to test the algorithms. The first system, demonstrated in 1992, is a 3-D head tracking system that used real-time template-based feature tracking. The second system, demonstrated in 1993, uses points tracked in an uncalibrated hand-held video-camera walkaround of a building to generate a 3-D polygonal texture-mapped model of the building, which has been used to generate new views of the building and to modify the original imagery by adding 3-D objects. The third system, demonstrated in 1996, uses points tracked from film taken by a helicopter-based camera to recover the 3-D motion of the helicopter and stabilize the resulting imagery.

The insights and practical successes of the classic point geometry solution were then used as the basis for designing a practical system for 3-D tracking of people. Unfortunately a sufficient number of point features is impossible to find and track on moving people, as are edges and contours. But recent successes on stably tracking parts of moving people based on color similarity measures led to a new type of 2-D feature called blob features. Correspondences of these blob features in multiple displaced images yields information about the 3-D object that produced them, and this is the basis for the 3-D blob estimation formulation developed in Chapter 5.

The same modeling and analysis approach developed for points was applied to blobs, and the resulting experimental systems led to several additional practical applications associated with tracking people. The first system, demonstrated in 1995, consisted of 3-D tracking of a person's head and hands using a calibrated wide-baseline stereo camera system. In 1996, this system was extended to include the orientation of the person's hands. The second system, demonstrated in 1996, made the person-tracking system more practical by allowing it to self-calibrate from the dynamic correspondences of the head and hands. The third system, demonstrated in 1996, uses the 3-D person tracking as the basis for a character animation system, which in turn is used as a simple interactive game with wireless interaction. Several other applications of the 3-D person-tracker have grown from this as well.

6.1 Future research

This work builds on a long tradition of research in a variety of technical fields, including photogrammetry, computer vision, and estimation, and provides some of the first results in new application areas including vision-mediated processing for video and film production and real-time visual human-computer interfaces. The accomplishments of this research provide new directions for research in all these fields and application areas.

With respect to photogrammetric tasks, such as building scene models and computing camera motions, this work's successful application to dynamic imagery can profoundly change the way many surveying tasks are performed by greatly increasing the level of automation in these tasks. Future challenges include how to use the dynamic buildup of 3-D models to help perform automatic feature tracking in a way which exceeds the reliability of traditional open-loop 2-D image processing.

With respect to computer vision, the results obtained here advance the level of thinking about what quality of 3-D information can be obtained from imagery, provide some new tools with which to obtain 3-D information more accurately and reliably, and give new insights on how to evaluate different models and approaches to the 3-D geometry problems. To continue extending computer vision theory along these lines, the most critical challenges include determining how to extend the probabilistic framework to include a wider variety of information.

In the field of estimation, the framework yields some interesting insights on the relationship of traditional estimators to more complex estimation topologies. An interesting challenge from an estimation and optimization standpoint is to extend the spirit of abstraction further to include a unified abstract way of looking at the problems of optimization over different kinds of domains, such as discrete and continuous, deterministic and probabilistic, etc. Optimization over combination domains is a commonly occuring problem in vision, AI, and learning, and it would be useful to find a way of solving that kind of problem in an elegant way.

In the application areas of vision-assisted media production, this research shows the vast potential that can be achieved with vision techniques and points the way toward many exciting and valuable capabilities. With the integration of several techniques for analyzing scenes, software can be generated to ease many media production jobs, including, importantly, the composition of 3-D computer graphics with filmed imagery.

In the application area of human-computer interface, this research has also shown a great potential for a new paradigm of computer interaction and control. When computers are able to effortlessly observe and follow the actions of people, new ways of communication between people and machines can be actively experimented with. The blob-based representation and probabilistic estimation explored in this thesis points to the viability of future research focused on understanding various levels of representation and integrating various sources of information within a probabilistic optimization framework.

Overall, the prevalent themes implied by the successes in this work include probabilistic representations and the paradigms of processing that are made possible by such representations. In particular, these include the important topics of multiple-level and coarse-to-fine representations of scenes and of integration of multiple sources of information for the purposes of creating more effective, more reliable, and possibly more intelligent machines.

Appendix A

Quaternions

A.1 Quaternion basics

A quaternion is an ordered set of four scalar quantities. Alternatively, it can be thought of as a set containing one scalar and one 3-vector. The quaternion, $q \in \mathcal{Q}$ can be written as a 4-vector

$$q = \begin{pmatrix} q_0 \\ q_X \\ q_Y \\ q_Z \end{pmatrix} = \begin{pmatrix} q_0 \\ q_v \end{pmatrix} \tag{A.1}$$

where the first element q_0 is called the *scalar part* and the last three elements $q_v = (q_X, q_Y, q_Z)$ comprise the *vector part*.

Among alternative representations of quaternions is the "complex number with three imaginary parts",

$$q = q_0 + iq_X + jq_Y + kq_Z \tag{A.2}$$

where the relations

$$i^2 = -1 \tag{A.3}$$
$$j^2 = -1 \tag{A.4}$$
$$k^2 = -1 \tag{A.5}$$
$$ij = k \tag{A.6}$$
$$jk = i \tag{A.7}$$
$$ki = j \tag{A.8}$$
$$ji = -k \tag{A.9}$$
$$kj = -i \tag{A.10}$$
$$ik = -j \tag{A.11}$$

apply or, equivalently, as the the sum of a scalar and vector

$$q = q_0 + q_v \tag{A.12}$$

where $q_v \in \mathbb{R}^3$ and vector multiplication

$$\mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathcal{Q} \tag{A.13}$$

is noncommutative and defined by a relationship

$$q_v r_v = q_v \times r_v - q_v \cdot r_v \tag{A.14}$$

that maps the product to a quaternion.

For both of these alternative representations, the scalar part q_0 can be referred to as the "real" part and the remaining three elements (or the vector) comprise the "imaginary" part. These "complex number" representations allow quaternion arithmetic to directly follow from regular arithmetic operations. The multiplication formula discussed below, for example, can be found directly using Eqn. A.3–Eqn. A.11 or using Eqn. A.14.

The *dot product*

$$\mathcal{Q} \times \mathcal{Q} \mapsto \mathbb{R} \tag{A.15}$$

is defined by

$$\begin{aligned} p \cdot q &= p_0 q_0 + p_v \cdot q_v && \text{(A.16)} \\ &= p_0 q_0 + p_X q_X + p_Y q_Y + p_Z q_Z && \text{(A.17)} \end{aligned}$$

similar to the usual vector dot product.

The *magnitude* of a quaternion is the Euclidean norm

$$||q|| = \sqrt{q \cdot q} \tag{A.18}$$

and a *unit quaternion*, which is useful for representation of rotation, as described in Section A.3, has $||q|| = 1$.

Quaternions have *conjugates*,

$$q^* = \begin{pmatrix} q_0 \\ -q_v \end{pmatrix} = \begin{pmatrix} q_0 \\ -q_X \\ -q_Y \\ -q_Z \end{pmatrix} \tag{A.19}$$

in which, in analogy to conventional complex numbers, the scalar or "real" part is unchanged and the vector or "imaginary" part is negated.

A.2 Quaternion algebra and properties

The operations of *multiplication* and *scalar multiplication* can be meaningfuly defined for the set of quaternions $\mathcal{Q}$. Quaternion scalar multiplication

$$\mathbb{R} \times \mathcal{Q} \mapsto \mathcal{Q} \tag{A.20}$$

is defined as in a linear vector space

$$\alpha q = \begin{pmatrix} \alpha q_0 \\ \alpha q_X \\ \alpha q_Y \\ \alpha q_Z \end{pmatrix} \tag{A.21}$$

where $\alpha \in \mathbb{R}$ and $q \in \mathcal{Q}$.

Quaternion multiplication

$$\mathcal{Q} \times \mathcal{Q} \mapsto \mathcal{Q} \tag{A.22}$$

maps two quaternions to a new quaternion. Multiplication is not commutative and therefore the *premultiplier* and *postmultiplier* must be distinguished. The quaternion multiplicationis defined as

$$pq = \begin{pmatrix} p_0q_0 - p_Xq_X - p_Yq_Y - p_Zq_Z \\ p_Xq_0 + p_0q_X - p_Zq_Y + p_Yq_Z \\ p_Yq_0 + p_Zq_X + p_0q_Y - p_Xq_Z \\ p_Zq_0 - p_Yq_X + p_Xq_Y + p_0q_Z \end{pmatrix} \quad \text{(A.23)}$$

$$= \begin{pmatrix} p_0 & -p_X & -p_Y & -p_Z \\ p_X & p_0 & -p_Z & p_Y \\ p_Y & p_Z & p_0 & -p_X \\ p_Z & -p_Y & p_X & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_X \\ q_Y \\ q_Z \end{pmatrix} \doteq M_{pre}(p)q \quad \text{(A.24)}$$

$$= \begin{pmatrix} q_0 & -q_X & -q_Y & -q_Z \\ q_X & q_0 & q_Z & -q_Y \\ q_Y & -q_Z & q_0 & q_X \\ q_Z & q_Y & -q_X & q_0 \end{pmatrix} \begin{pmatrix} p_0 \\ p_X \\ p_Y \\ p_Z \end{pmatrix} \doteq M_{post}(q)p \quad \text{(A.25)}$$

where $p \in \mathcal{Q}$ and $q \in \mathcal{Q}$.

As demonstrated by the above expressions, quaternion multiplication can be converted into a matrix-vector multiplication by forming a matrix from one of the multiplicands and a vector from the other. If the matrix derives from the premultiplier, as does the matrix $M_{pre}(p)$ above, it is called the *quaternion premultiplier matrix* and if from the postmultiplier, as in $M_{post}(q)$ above, it is called the *quaternion postmultiplier matrix*.

The premultiplier and postmultiplier matrices for a given quaternion p are not the same, i.e. $M_{pre}(p) \neq M_{post}(p)$. (Otherwise, multiplication would be commutative.) Specifically, $M_{post}(p)$ varies from $M_{pre}(p)$ in that the lower right $(3,3)$ matrix of each is transposed from the other.

A useful property is that conjugating quaternions results in transposing the multiplier matrices, i.e.

$$M_{pre}(q^*) = M_{pre}^T(q) \quad \text{(A.26)}$$

$$M_{post}(q^*) = M_{post}^T(q) \quad \text{(A.27)}$$

hold.

The multiplier matrix for a multiplicand is meaningful because it is the partial derivative matrix of the product with respect to the other multiplicand. That is, if $r = pq$ then

$$\frac{\partial r}{\partial q} = M_{pre}(p) \quad \text{(A.28)}$$

is a Jacobian matrix and, similarly,

$$\frac{\partial r}{\partial p} = M_{post}(q) \quad \text{(A.29)}$$

is another Jacobian matrix.

A compact analytical formula for the multiplication operation is

$$pq = \begin{pmatrix} p_0q_0 - p_vq_v \\ p_0q_v + q_0p_v + p_v \times q_v \end{pmatrix} \quad \text{(A.30)}$$

which is equivalent to the matrix equation above. This expression can be more useful when performing analytical manipulations, whereas the other concept is more useful for performing numerical calculations.

The quaternion algebra also has a multiplicative identity element, the *identity quaternion*

$$e = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{A.31}$$

for which it can be easily verified that $eq = q$ and $qe = q$ for any quaternion $q \in \mathcal{Q}$.

Quaternions have *inverses* that satisfy $qq^{-1} = q^{-1}q = e$. The inverse is simply

$$q^{-1} = \frac{q^*}{q \cdot q} \tag{A.32}$$

leading to the fact that, for unit quaternions, $q^{-1} = q^*$.

[If addition of quaternions is defined as termwise addition of the elements, as for regular 4-vectors, then the above properties make the set of all quaternions a *noncommutative algebra with unit element*. The set of all (n, n) matrices is also such an algebra. The set of n-vectors is not an algebra because there is no multiplication operation. Hence, many powerful properties of algebras apply to the set of quaternions; this and the relation of quaternions to physical phenomena, such as 3-D rotations, motivated Hamilton's intense interest in quaternions.]

A.3 Representation of rotation

Sir Hamilton found many uses for quaternions in the analysis of physical problems. Unit quaternions, in particular, were found to be useful for representing the rotation of a 3-vector in 3-D space.

Quaternions can be used to compute the vector x' which results from rotation of the vector x about some axis $\hat{n}$ through an angle θ in some 3-D reference frame. Commonly, this rotation is computed by multiplying the original vector by a 3×3 orthonormal direction cosine matrix

$$x' = Rx \tag{A.33}$$

Alternately, Rodrigues's Formula provides a different representation of the relationship between the new vector x' and the original vector x. Rodrigues's Formula describes the vector that results from rotationg the original vector through an angle θ about the unit vector $\hat{n}$:

$$x' = \cos\theta x + \sin\theta \hat{n} \times x + (1 - \cos\theta)(\hat{n} \cdot x)\hat{n} \tag{A.34}$$

To derive this, consider a vector x rotating about the axis $\hat{n}$ (see Fig. A-1). The vector consists of components along $\hat{n}$ and perpendicular to it. The components can be separated as

$$x = \underbrace{(x \cdot \hat{n})\hat{n}}_{\text{along axis}} + \underbrace{(x - (x \cdot \hat{n})\hat{n})}_{\text{orthogonal to axis}} \tag{A.35}$$

Only the second term is affected by the rotation about the axis $\hat{n}$.

After a rotation through an angle θ, the second term becomes

$$(x - (x \cdot \hat{n})\hat{n}) \cos\theta + \hat{n} \times (x - (x \cdot \hat{n})\hat{n}) \sin\theta \tag{A.36}$$

The formula of Rodrigues results from re-combining this with the unchanged $(x \cdot \hat{n})\hat{n}$ to get

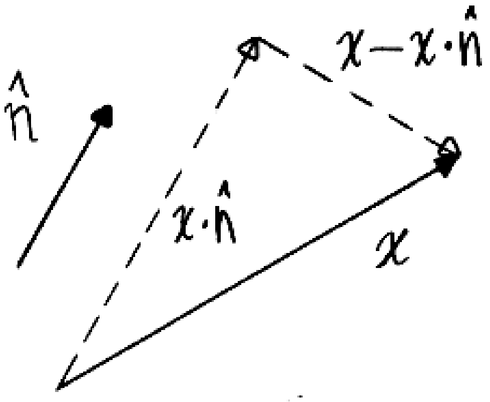


Figure A-1: When a vector x rotates about an axis $\hat{n}$, only the component of x orthogonal to the axis changes.

the expression in Eqn. A.34.

Consider now the quaternion formula

$$q\vec{x}q^{-1} \tag{A.37}$$

where q is a unit quaternion and

$$\vec{x} = \begin{pmatrix} 0 \\ x \end{pmatrix} \tag{A.38}$$

where $x \in \mathbb{R}^3$ is the vector part.

The product of the two quaternion multiplications is a quaternion with zero scalar part and a vector part equal to

$$(q_0^2 - q_v \cdot q_v)x + 2q_0 q_v \times x + 2(q_v \cdot x)q_v \tag{A.39}$$

The substitutions

$$q_0 = \cos(\theta/2) \tag{A.40}$$

and

$$q_v = \sin(\theta/2)\hat{n} \tag{A.41}$$

yield Rodrigues's Formula for the vector part.

Hence,

$$\vec{x}' = q\vec{x}q^{-1} \tag{A.42}$$

In this way, unit quaternions can be used to implement rotation of a vector. Specifically, the unit quaternion

$$q = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2)\hat{n} \end{pmatrix} \tag{A.43}$$

imparts a rotation to the vector x according to the angle θ and the axis $\hat{n}$.

The inverse transformation

$$\vec{x} = q^{-1}\vec{x}'q \tag{A.44}$$

results trivially from the forward tranformation through premultiplication by q^{-1} and post-multiplication by q.

A.3.1 Rotation matrices

Further perspective on the operation of rotation using unit quaternions results from expanding the expression in terms of pre- and post-multiplier matrices.

$$\begin{aligned} q\vec{x}q^{-1} &= (M_{pre}(q)\vec{x})q^{-1} && \text{(A.45)}\\ &= M_{post}^T(q)M_{pre}(q)\vec{x} && \text{(A.46)} \end{aligned}$$

The matrix $M_{post}^T(q)M_{pre}(q)$ has the form

$$M_{post}^T(q)M_{pre}(q) = \begin{pmatrix} q\cdot q & 0 & 0 & 0 \\ 0 & (q_0^2+q_X^2-q_Y^2-q_Z^2) & 2(q_Xq_Y-q_0q_Z) & 2(q_Xq_Z+q_0q_Y) \\ 0 & 2(q_Xq_Y+q_0q_Z) & (q_0^2-q_X^2+q_Y^2-q_Z^2) & 2(q_Yq_Z-q_0q_X) \\ 0 & 2(q_Xq_Z-q_0q_Y) & 2(q_Yq_Z+q_0q_X) & (q_0^2-q_X^2-q_Y^2+q_Z^2) \end{pmatrix} \tag{A.47}$$

where the lower right is the orthonormal rotation matrix.

Thus we obtain the formula

$$R(q) = \begin{pmatrix} q_0^2+q_X^2-q_Y^2-q_Z^2 & 2(q_Xq_Y-q_0q_Z) & 2(q_Xq_Z+q_0q_Y) \\ 2(q_Xq_Y+q_0q_Z) & q_0^2-q_X^2+q_Y^2-q_Z^2 & 2(q_Yq_Z-q_0q_X) \\ 2(q_Xq_Z-q_0q_Y) & 2(q_Yq_Z+q_0q_X) & q_0^2-q_X^2-q_Y^2+q_Z^2 \end{pmatrix} \tag{A.48}$$

for generating a rotation matrix from a unit quaternion.

It can be seen therefore, that the quaternion operation is entirely equivalent to rotation using direction cosine matrices. However, quaternions are preferable for some analytic manipulations and for numerical manipulations. In [35, 60], it is noted that composition of rotations requires less computation when quaternions are used in place of rotation matrices. And, perhaps more importantly, re-normalization of unit quaternions due to finite precision calculations is trivial compared to re-normalization of rotation matrices.

A.3.2 Composition of rotations

Composition of rotations is important in many applications where rotations are used. To illustrate how rotations are composed using unit quaternions, the property

$$p^*q^* = (qp)^* \tag{A.49}$$

is useful. This relation can be verified from direct evaluation of the two sides.

Now let p describe the rotation $\vec{x} \mapsto \vec{x}'$ and let q describe the rotation $\vec{x}' \mapsto \vec{x}''$. Then

$$\vec{x}' = p\vec{x}p^* \tag{A.50}$$

and

$$\begin{aligned} \vec{x}'' &= q\vec{x}'q^* && \text{(A.51)}\\ &= qp\vec{x}p^*q^* && \text{(A.52)}\\ &= r\vec{x}r^* && \text{(A.53)} \end{aligned}$$

where $r = qp$. Thus rotations are composed by multiplying the quaternions associated with each rotation in reverse order of the rotations.

References

See [7, 35, 36, 37, 60].

Appendix B

Linear estimators as specializations of manifold-tangent estimators

This appendix restates the traditional linear estimators in the terminology of the manifold-tangent estimators developed in Chapter 3 to make explicit the means by which they are a specialization of the broader framework and thereby to motivate the notion that the manifold-tangent framework is indeed a natural generalization of estabilished estimation strategies.

In linear systems, the iteration loops of Alg. 1 can be discarded and we are left with the general approach to linear least squares outlined in Alg. 7 below in which various interpretations of the weighting matrices results in weighted least squares (WLS) and its probabilistic interpretation, maximum likelihood (ML) estimation; recursive weighted least squares (RWLS) estimation and its probabilistic interpretation, maximum a posteriori (MAP) estimation; and the traditional linear least squares pseudoinverses, including the Moore-Penrose pseudoinverse based on direct inversion of the normal equations and the Golub-Reinsch pseudoinverse based on singular value decomposition (SVD) of the normal equations.

The following algorithm specializes Alg. 1 to a general procedure for linear Euclidean systems and the subsequent sections further specialize to the particular linear techniques.

Algorithm 7 (General Euclidean Linear Least Squares) *Given a manifold-tangent system model $\mathcal{M}$ with Euclidean state model, the least squares objective function is*

$$F(x, y) = e^T(x, y)R^{-1}e(x, y)$$

where $x \leftarrow \mathcal{M}$, $y \leftarrow \mathcal{M}_Y$, $e \leftarrow \mathcal{M}_C$ and R is a positive definite weighting matrix and

$$e(x, y) = Jx - y$$

and $J \in I\!R^{p \times n}$.

input x_0
compute x_1 (no iteration required)
  (constraint is already linear)
    $e(x,y) = Jx_0 - y + J(x - x_0)$
    $\delta e \doteq (y - Jx_0) = J(x - x_0)$
  compute $\hat{x}$ (no iteration required)
    choose P^{-1}
    solve
      $\hat{\delta x} = (J^T R^{-1} J + P^{-1})^{-1} J^T R^{-1} \delta e$
    move
      $\hat{x} = x_0 + \hat{\delta x}$
  end
  update $x_1 = \hat{x}$
end

where P^{-1} is a positive semi-definite matrix.

■

B.1 Weighted linear least squares (WLS) estimate

If the positive definite error weighting matrix is chosen diagonal

$$R^{-1} = \begin{pmatrix} 1/\sigma_1^2 & & \\ & \ddots & \\ & & 1/\sigma_p^2 \end{pmatrix} \tag{B.1}$$

then the "weighted square error" objective function we would like to minimize becomes

$$\begin{aligned} F(x,y) &= \sum_{j=1}^{p} \frac{1}{\sigma_j^2} e_j^2(x,y) && \text{(B.2)} \\ &= e^T(x,y) R^{-1} e(x,y) && \text{(B.3)} \\ &= (Jx - y)^T R^{-1} (Jx - y) && \text{(B.4)} \\ &= (J\delta x - \delta e)^T R^{-1} (J\delta x - \delta e) && \text{(B.5)} \end{aligned}$$

where $\delta x = x - x_0$ and $\delta e = y - Jx_0$ (as in Alg. 7) which expresses the weighted sum of squared constraint error. The minimum of this quadratic error function can be obtained by finding where the gradient vector

$$\frac{\partial F(x,y)}{\partial \delta x} = 2(J^T R^{-1}(J\delta x - \delta e))^T \tag{B.6}$$

is identically zero, i.e.

$$J^T R^{-1}(J\hat{\delta x}_{wls} - \delta e) = 0 \tag{B.7}$$

resulting in

$$\hat{\delta x}_{wls} = (J^T R^{-1} J)^{-1} J^T R^{-1} \delta e \tag{B.8}$$

and thus

$$\begin{aligned} \hat{x}_{wls} &= x_0 + \hat{\delta x}_{wls} && \text{(B.9)} \\ &= x_0 + (J^T R^{-1} J)^{-1} J^T R^{-1} (y - Jx_0) && \text{(B.10)} \end{aligned}$$

$$= (J^T R^{-1} J)^{-1} J^T R^{-1} y \tag{B.11}$$

which is the well known "weighted least squares" optimal estimate, where the value of x_0 is seen to be completely irrelevant to the estimate.

In Alg. 7, choosing P^{-1} to be zero, results in the single-step state estimate

$$\begin{aligned} x_1 &= \hat{x} && \text{(B.12)} \\ &= x_0 + (J^T R^{-1} J + P^{-1})^{-1} J^T R^{-1} (y - J x_0) && \text{(B.13)} \\ &= (J^T R^{-1} J)^{-1} J^T R^{-1} y && \text{(B.14)} \end{aligned}$$

which can be recognized as the WLS solution, where the initial state prediction x_0 can be seen algebraically to be irrelevant. Thus, Alg. 1 for a Euclidean state space with a linear constraint function, with R^{-1} chosen as positive definite diagonal, and P^{-1} chosen as zero, results in the linear WLS estimate.

B.2 Fisher's maximum likelihood (ML) estimate

A probabilistic interpretation leads to a principled and more general method for choosing the weighting matrix R^{-1}. Instead of simply minimizing the error vector (or arbitrarily weighted error vector), we can acknowledge that our measurements y or our constraint model $e(x, y)$ or both are inaccurate and we can try to model the errors.

We can model the residual error due to measurement noise and modeling error as a random variable v such that a new probabilistic constraint process

$$\begin{aligned} e(x, y) + v &= Jx - y + v && \text{(B.15)} \\ &= 0 && \text{(B.16)} \end{aligned}$$

is formed which is exactly satisfied.

If the error process v is chosen to be zero-mean and Gaussian-distributed such that

$$Ev = 0 \tag{B.17}$$

$$Evv^T = R \tag{B.18}$$

then we obtain

$$\begin{aligned} p(e(x, y)) &= p(v) && \text{(B.19)} \\ &= \frac{1}{(2\pi)^{p/2} |R|^{1/2}} \exp\left(-\frac{1}{2} e^T(x, y) R^{-1} e(x, y) \right) && \text{(B.20)} \end{aligned}$$

as the joint likelihood function.

The "maximum likelihood" estimate chooses the δx with the highest likelihood of producing the observed error, which is the same as choosing the δx to maximize the exponent, or equivalently the minimum of the negative exponent

$$e^T(x, y) R^{-1} e(x, y) \tag{B.21}$$

which can be recognized as the original objective function with R chosen to be the covariance of the model error. Therefore, minimization proceeds as in the WLS case, resulting in

$$\hat{x}_{ml} = (J^T R^{-1} J)^{-1} J^T R^{-1} y \tag{B.22}$$

which is the same as $\hat{x}_{wls}$ except with a firm probabilistic interpretation of R and a bit more generality (because R does not have to be diagonal corresponding to completely independent error characteristics along the principal axes of the state space). Thus, Alg. 1 for a Euclidean state space with a linear constraint function, with R the covariance of zero-mean Gaussian constraint error, and P^{-1} chosen as zero, results in the linear Gaussian ML estimate.

B.3 Recursive weighted linear least squares (RWLS) estimate

An additional term can be added to the objective function to penalize deviation from a specified point x_0 in state space. In recursive estimation, this is presumably because prior measurements have established confidence that the target state should be near x_0.

If the P^{-1} matrix in Alg. 7 is chosen diagonal

$$P^{-1} = \begin{pmatrix} 1/\varphi_1^2 & & \\ & \ddots & \\ & & 1/\varphi_n^2 \end{pmatrix} \tag{B.23}$$

and the objective function is augmented with a penalty term based on P^{-1} and x_0, we obtain

$$\begin{aligned} F(x,y) &= \sum_{j=1}^{p} \frac{1}{\sigma_j^2} e_j^2(x,y) + \sum_{i=1}^{n} \frac{1}{\varphi_i^2}(x_i - x_{0i})^2 && \text{(B.24)} \\ &= e^T(x,y) R^{-1} e(x,y) + (x - x_0)^T P^{-1} (x - x_0) && \text{(B.25)} \\ &= (Jx - y)^T R^{-1} (Jx - y) + (x - x_0)^T P^{-1} (x - x_0) && \text{(B.26)} \\ &= (J\delta x - \delta e)^T R^{-1} (J\delta x - \delta e) + \delta x^T P^{-1} \delta x && \text{(B.27)} \end{aligned}$$

which can be minimized by allowing the gradient

$$\frac{\partial F(x,y)}{\partial \delta x} = 2(J^T R^{-1}(J\delta x - \delta e))^T + 2(P^{-1}\delta x)^T \tag{B.28}$$

to vanish, i.e.

$$J^T R^{-1}(J\hat{\delta x}_{rwls} - \delta e) + P^{-1}\hat{\delta x}_{rwls} = 0 \tag{B.29}$$

resulting in

$$\hat{\delta x}_{rwls} = (J^T R^{-1} J + P^{-1})^{-1} J^T R^{-1} \delta e \tag{B.30}$$

and thus

$$\begin{aligned} \hat{x}_{rwls} &= x_0 + \hat{\delta x}_{rwls} && \text{(B.31)} \\ &= x_0 + (J^T R^{-1} J + P^{-1})^{-1} J^T R^{-1} (y - Jx_0) && \text{(B.32)} \\ & && \text{(B.33)} \end{aligned}$$

gives the recursive weighted least squares solution. Thus, Alg. 1 for a Euclidean state space with a linear constraint function, with R^{-1} chosen as positive definite diagonal, and P^{-1} chosen as positive definite diagonal, results in the RWLS estimate.

B.4 Bayesian maximum a posteriori (MAP) estimate

As in the ML case, a probabilistic interpretation of the weighting matrices yields the same result but with a probabilistic meaning.

If P is chosen as the prior covariance on the state, i.e.

$$E\delta x = 0 \tag{B.34}$$

$$E\delta x \delta x^T = P \tag{B.35}$$

and $\delta x \doteq x - x_0$ is Gaussian distributed, the prior distribution is

$$p(\delta x) = \frac{1}{(2\pi)^{n/2}|P|^{1/2}} \exp\left(-\frac{1}{2}\delta x^T P^{-1} \delta x\right) \tag{B.36}$$

which represents prior knowledge of x_0.

The likelihood function of Eqn. B.20 can be interpreted as a conditional probability distribution on δe given δx

$$p(\delta e|\delta x) = \frac{1}{(2\pi)^{p/2}|R|^{1/2}} \exp\left(-\frac{1}{2}(\delta e - J\delta x)^T R^{-1} (\delta e - J\delta x)\right) \tag{B.37}$$

where $\delta e = y - Jx_0$.

Using Bayes's rule

$$p(\delta x|\delta e) = \frac{p(\delta e|\delta x)p(\delta x)}{p(\delta e)} \tag{B.38}$$

and noting that δe is deterministic, we arrive at the posterior distribution

$$p(\delta x|\delta e) = C \exp\left(-\frac{1}{2}(\delta e - J\delta x)^T R^{-1} (\delta e - J\delta x)\right) \exp\left(-\frac{1}{2}\delta x^T P^{-1} \delta x\right) \tag{B.39}$$

where C is a scalar constant which will be irrelevant. The maximum a posteriori (MAP) estimate is obtained by maximizing the posterior distribution with respect to δx, which is equivalent to finding the minimum of

$$(J\delta x - \delta e)^T R^{-1} (J\delta x - \delta e) + \delta x^T P^{-1} \delta x \tag{B.40}$$

thus the MAP estimate

$$\hat{x}_{map} = x_0 + (J^T R^{-1} J + P^{-1})^{-1} J^T R^{-1} (y - Jx_0) \tag{B.41}$$

is identical to the RWLS estimate, but here, again, the weighting matrices have a probabilistic interpretation and are more general because they do not have to be diagonal. Thus, Alg. 1 for a Euclidean state space with a linear constraint function, with R equal to the constraint error covariance and P equal to the prior state covariace, results in the linear MAP estimate.

B.5 Moore-Penrose pseudoinverse

If $P^{-1} = 0$ and $R^{-1} = aI$ for any scalar a we obtain

$$\hat{\delta x}_{lls} = (J^T J)^{-1} J^T \delta e \tag{B.42}$$

which is the familiar Moore-Penrose pseudoinverse. The proof of optimality in the least-squares sense is the same as for WLS with the appropriate substitution for R. Thus, Alg. 1 reduces to traditional linear least squares with the assumptions of WLS plus an isotropic R matrix.

The solution is a direct inversion of the normal equations

$$J^T \delta e = J^T J \delta x \tag{B.43}$$

which can be singular. It is worthy to note that this pseudoinverse as well as WLS and ML will become unsolvable when J does not have full column rank while RWLS and MAP will remain well-behaved. This is because the extra information about the prior state numerically stabilizes the estimate, choosing one preferred solution from the infinite solutions that are a result of rank deficiency of J. The Golub-Reinsch pseudoinverse below accomplishes the same stability without overtly assuming any priors on x, but does effectively the same thing, choosing the δx that is closest to zero.

B.6 Golub-Reinsch pseudoinverse

When the normal equations Eqn. B.43 are singular, the Golub-Reinsch pseudoinverse, based on the singular value decomposition (SVD) [56, 61], can be used to find the "best" solution in the column space of J. If the SVD of J is

$$J = U \Sigma V^T \tag{B.44}$$

then the pseudoinverse is

$$J^\dagger = V(\Sigma^{-1})' U^T \tag{B.45}$$

where

$$U = \begin{pmatrix} u_1 & \cdots & u_n \end{pmatrix} \in \mathbb{R}^{p \times n} \tag{B.46}$$

$$V = \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix} \in \mathbb{R}^{n \times n} \tag{B.47}$$

have orthonormal columns and

$$\Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} \in \mathbb{R}^{n \times n} \tag{B.48}$$

is diagonal. The inverse $(\Sigma^{-1})'$ is modified from Σ^{-1} to replace infinite or large diagonal elements of Σ^{-1} with zero. It can be proven that

$$\hat{\delta x} = J^\dagger \delta e \tag{B.49}$$

selects the the smallest vector δx in the column space of J [56].

B.7 Relationship between batch and recursive steps

In both the recursive (Kalman-based) algorithms of this section and the batch (Newton-based) algorithms of Section 3.3, the key step in the algorithm is generating a step δx to a new state based on the "innovation" δe

$$\delta x = K\delta e \tag{B.50}$$

where the "gain matrix"

$$K \leftarrow P, R, J_x \tag{B.51}$$

depends on prior constraints on the state encoded in a weighting matrix P, characterization of measurement and modeling error encoded in a weighting matrix R, and first-order differential properties of the constraint equation J_x.

For the recursive algorithms, the gain matrix takes the form

$$K_r = PJ_x^T(J_xPJ_x^T + R)^{-1} \tag{B.52}$$

whereas for the batch algorithms, the gain matrix takes the form

$$K_b = (P^{-1} + J_x^TR^{-1}J_x)^{-1}J_x^TR^{-1} \tag{B.53}$$

primarily because of the different ways in which the methods were motivated and derived. However, if P and R are both invertible, the two forms can be shown to be the same thing through use of the *matrix inversion lemma*

$$(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1} \tag{B.54}$$

where A and D are square and invertible.

The lemma can be applied to the Kalman gain matrix

$$\begin{aligned}
K_r &= PJ_x^T(J_xPJ_x^T + R)^{-1} && \text{(B.55)}\\
&= PJ_x^T(R^{-1} - R^{-1}J_x(P^{-1} + J_x^TR^{-1}J_x)^{-1}J_x^TR^{-1}) && \text{(B.56)}\\
&= P(J_x^TR^{-1} - J_x^TR^{-1}J_x(P^{-1} + J_x^TR^{-1}J_x)^{-1}J_x^TR^{-1}) && \text{(B.57)}\\
&= P((P^{-1} + J_x^TR^{-1}J_x) - J_x^TR^{-1}J_x)(P^{-1} + J_x^TR^{-1}J_x)^{-1}J_x^TR^{-1} && \text{(B.58)}\\
&= (P^{-1} + J_x^TR^{-1}J_x)^{-1}J_x^TR^{-1} && \text{(B.59)}\\
&= K_b && \text{(B.60)}
\end{aligned}$$

to prove the equivalence. The matrix inversion lemma is proven in Section B.8.

In fact, the RWLS and MAP algorithms of Section B.3 and Section B.4 produce exactly this gain matrix with exactly the same probabilistic interpretation. Coupled with an equation to update the covariance matrix, the MAP estimator comprises the estimation half of the KF.

The implication of all these relationships between methods is that they all come down to the same unifying theme for taking a single step to a better estimate. Understanding this and how the manifold-tangent model extends this idea to state spaces on curved manifolds will help one to design the right estimation strategy for a particular problem and illuminates the capabilities and limitations of the existing methodologies.

B.8 Proof of Matrix Inversion Lemma

Lemma 1 (Matrix Inversion Lemma) *Given four matrices*

$$\begin{aligned} A &\in \mathbb{R}^{n\times n} \\ B &\in \mathbb{R}^{n\times m} \\ C &\in \mathbb{R}^{m\times n} \\ D &\in \mathbb{R}^{m\times m} \end{aligned}$$

where A^{-1} and D^{-1} exist, the following relationship

$$(A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}BD^{-1}(I - CA^{-1}BD^{-1})^{-1}CA^{-1}$$

holds.

Proof: If matrix Q were formed such that

$$Q = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathbb{R}^{n+m\times n+m} \tag{B.61}$$

then

$$Q^{-1} = \begin{pmatrix} E & F \\ G & H \end{pmatrix} \in \mathbb{R}^{n+m\times n+m} \tag{B.62}$$

exists where

$$E \in \mathbb{R}^{n\times n} \tag{B.63}$$
$$F \in \mathbb{R}^{n\times m} \tag{B.64}$$
$$G \in \mathbb{R}^{m\times n} \tag{B.65}$$
$$H \in \mathbb{R}^{m\times m} \tag{B.66}$$

are the submatrices and the relationships

$$QQ^{-1} = Q^{-1}Q = I \tag{B.67}$$

hold.

Thus

$$QQ^{-1} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} E & F \\ G & H \end{pmatrix} \tag{B.68}$$
$$= \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix} \tag{B.69}$$
$$= \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \tag{B.70}$$

leading to

$$\left.\begin{aligned} AE+BG &= I \\ AF+BH &= 0 \\ CE+DG &= 0 \\ CF+DH &= I \end{aligned}\right\} \Rightarrow \begin{aligned} A &= (I-BG)E^{-1} \\ B &= -AFH^{-1} \\ C &= -DGE^{-1} \\ D &= (I-CF)H^{-1} \end{aligned} \tag{B.71}$$

expressions for (A, B, C, D) in terms of $(A, B, C, D), (E, F, G, H)$.

Similarly,

$$Q^{-1}Q = \begin{pmatrix} E & F \\ G & H \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} \tag{B.72}$$

$$= \begin{pmatrix} EA+FC & EB+FD \\ GA+HC & GB+HD \end{pmatrix} \tag{B.73}$$

$$= \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} \tag{B.74}$$

leads to

$$\left.\begin{array}{r} EA+FC=I \\ EB+FD=0 \\ GA+HC=0 \\ GB+HD=I \end{array}\right\} \Rightarrow \begin{array}{l} E=(I-FC)A^{-1} \\ F=-EBD^{-1} \\ G=-HCA^{-1} \\ H=(I-GB)D^{-1} \end{array} \tag{B.75}$$

expressions for (E,F,G,H) in terms of $(A,B,C,D),(E,F,G,H)$.
Solving for (E,F,G,H) in terms of (A,B,C,D),

$$E = (I-EBD^{-1}C)A^{-1} \tag{B.76}$$

$$= A^{-1}+EBD^{-1}CA^{-1} \tag{B.77}$$

leads to

$$E = A^{-1}(I-BD^{-1}CA^{-1})^{-1} \tag{B.78}$$

$$= \left[(I-BD^{-1}CA^{-1})A\right]^{-1} \tag{B.79}$$

$$= (A-BD^{-1}C)^{-1} \tag{B.80}$$

$$F = -A^{-1}(I-BD^{-1}CA^{-1})^{-1}BD^{-1} \tag{B.81}$$

and

$$H = (I+HCA^{-1}B)D^{-1} \tag{B.82}$$

$$= D^{-1}+HCA^{-1}BD^{-1} \tag{B.83}$$

leads to

$$G = -D^{-1}(I-CA^{-1}BD^{-1})^{-1}CA^{-1} \tag{B.84}$$

$$H = D^{-1}(I-CA^{-1}BD^{-1})^{-1} \tag{B.85}$$

$$\tag{B.86}$$

giving (E,F,G,H) in terms of (A,B,C,D).
Now substituting to find (A,B,C,D) in terms of (A,B,C,D),

$$A = (I-BG)E^{-1} \tag{B.87}$$

$$AE = (I-BG) \tag{B.88}$$

$$A(A-BD^{-1}C)^{-1} = I-B\left[-D^{-1}(I-CA^{-1}BD^{-1})^{-1}CA^{-1}\right] \tag{B.89}$$

$$(A-BD^{-1}C)^{-1} = A^{-1}+A^{-1}BD^{-1}(I-CA^{-1}BD^{-1})^{-1}CA^{-1} \tag{B.90}$$

Q.E.D.■

Acknowledgements

First and foremost, I would like to acknowledge the support, leadership, and vision of my advisor, Sandy Pentland, who brought me to the Media Lab, encouraged innovative ideas, and rewarded me with the opportunity to work as part of a first class organization of people and facilities. I've learned a great deal from being under his wing, and this kind of research could only have taken place in the kind of environment he has established here at the Media Lab.

Among those people who made the Vision and Modeling research group what it was when I arrived are Ted Adelson, whose leadership of the group, along with Sandy's, was a great motivation for me coming to the lab, and the group of "old Vismod" people, including Eero Simoncelli, Bill Freeman, Matt Turk, Trevor Darrell, Irfan Essa, Stan Sclaroff, Martin Friedmann, Bradley Horowitz, John Wang, Roz Picard, Pawan Sinha, Thad Starner, John Martin and Laureen Chapman. I will always remember this group as the inspiration that it was in the first days I was here, with a spirit of camaraderie, teamwork, and creativity that made it a pleasure to be part of the group.

Special acknowledgements go to my officemate and co-conspirator in much of my early work, Bradley Horowitz, with whom I developed a great friendship and many of the ideas which were the seeds for much of the work in this thesis. And to Martin Friedmann, my other officemate, who was in life, and beyond life, a truly unique and inspirational person, and to whom this document is dedicated; memories of Marty continue to influence me and my outlook on everything in my own life.

Over the five years I was a part of Vismod an awful lot of bright and creative people have passed through these parts and have influenced both my research and my personal life in many positive ways. Among those people are Chris Perry, Lee Campbell, Aaron Bobick, Fang Liu, Monika Gorkani, Alex Sherstinsky, Sourabh Niyogi, Stephen Intille, Jeff Bilmes, Baback Moghaddam, Kris Popat, Steve Mann, Mike Casey, Tom Minka, Andy Wilson, Claudio Pinhanez, Yair Weiss, Judy Bornstein, Mike Johnson, Bruce Blumberg, David Allport, Chris Wren, Dave Becker, Ken Russell, Wasi Wahid, Martin Szummer, Jim Davis, Matt Brand, Flavia Sparacino, Laurie Ward, Matt Krom, Erik Trimble, Chahab Nastar, Nassir Navab, Josh Wachman, Michal Hlavac, Jeff Levine, Sumit Basu, Chloe Chao, Jen Healey, Nuria Oliver, Deb Roy, Jocelyn Riseberg, Francois Berard, Tony Jebara, Shawn Becker, and the 'evil queen' herself, Karen Navarro. I am sure there are many others. During this time Roz Picard and Aaron Bobick came on as new faculty and added a new dimension to the group. Aaron, in particular, contributed a good deal to my thinking in the early stages of my thesis by participating on my general exam committee as did Whitman Richards. My current officemate of several years, Lee Campbell, is always a source of useful information, good conversation, and good proofreading, including of this thesis, not to mention expert advice on chocolate. My past and future co-conspirator in film post-production, Chris Perry, has always been and continues to be a source of creative energy and new ideas. And special mention to the "free jazz quartet", consisting of me, Jim Davis, and David Allport, which brought profound musical concepts into my research. And speaking of music, my good friend and co-conspirator in music, imbibery, and other business matters, Mike Valdez, deserves thanks for providing a key creative outlet on many occasions over the years. And fellow trumpeteer and advisor in all things, from theses, to the ways of women, the philosophy of fashion, and, of course, all aspects of music, my good friend Dave Ricks, who exists currently only on the 'net, but nonetheless is a creative presence. And special thanks to Mitch, a bright light in my life and another great co-conspirator outside the laboratory walls.

I would also like to acknowledge here my esteemed doctoral committee members Eric Grimson and Takeo Kanade and thank them for taking the time and trouble to read and understand my work and provide me with the privilege of their expertise and guidance. It is much appreciated.

Finally, I would like to acknowledge the vast contributions of my parents and family over the past twenty-five years of my formal education, which is finally coming to an end. I am thankful for the opportunities, the wisdom, and the values that have been passed on to me and have aided me through this sometimes trying adventure, and I'm sure that the strength and knowledge I have gained as a result will help guide me through many new adventures to come.

Bibliography

[1] Nicholas Ayache and Olivier Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Trans. Robotics Automation*, 5(6):804–819, 1989.

[2] Ali Azarbayejani, Bradley Horowitz, Tinsley Galyean, and Alex Pentland. Method and apparatus for generating three-dimensional, textured models from plural video images. *U.S. Patent #5,511,153*, 23 April 1996.

[3] Ali Azarbayejani and Alex Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 17(6):562–575, June 1995.

[4] Ali Azarbayejani, Thad Starner, Bradley Horowitz, and Alex Pentland. Visually controlled graphics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(6):602–605, June 1993.

[5] Ali J. Azarbayejani. Model-based vision navigation for a free-flying robot. Master's thesis, Department of Aeronautics and Astronautics and Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, September 1991.

[6] Ali J. Azarbayejani, Tinsley Galyean, Bradley Horowitz, and Alex Pentland. Recursive estimation for CAD model recovery. In *2nd CAD-based Vision Workshop*, Los Alamitos, CA, February 1994. IEEE Computer Society, IEEE Computer Society Press. (Champion, PA).

[7] Richard H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics*. American Institute of Aeronautics and Astronautics, New York, NY, 1987.

[8] Horst A. Beyer. An introduction to photogrammetric camera calibration. In *Seminaire Orasis*, pages 37–42, St. Malo, September 1991.

[9] D. Beymer and T. Poggio. Image representations for visual learning. *Science*, 272(5270):1905–1909, june 1996.

[10] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. A.I. Memo No. 1431, MIT AI Lab, 1993.

[11] Thomas O. Binford. Visual perception by computer. In *Proceeding of the IEEE Conference on Systems and Control*, December 1971. (Miami).

[12] Brian Blau and Clark Dodsworth et al, editors. *Visual Proceedings, The Art and Interdisciplinary Programs of Siggraph '96*, New York, NY, August 1996. The Association for Computing Machinery, Inc., ACM Siggraph. (New Orleans, LA).

[13] Aaron Bobick and Jim Davis. An appearance-based representation of action. In *ICPR*, August 1996.

[14] Ted J. Broida, S. Chandrashekhar, and Rama Chellappa. Recursive estimation of 3-D motion from a monocular image sequence. *IEEE Trans. Aerosp. Electron. Syst.*, 26(4):639–656, July 1990.

[15] Ted J. Broida and Rama Chellappa. Estimation of object motion parameters from noisy images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(1):90–99, January 1986.

[16] Ted J. Broida and Rama Chellappa. Estimating the kinematics and structure of a rigid object from a sequence of monocular images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(6):497–513, June 1991.

[17] D. C. Brown. A solution to the general problem of multiple station analytical stereotriangulation. RCS TR 43, Patrick Air Force Base, 1958.

[18] Robert Grover Brown. *Introduction to Random Signal Analysis and Kalman Filtering.* John Wiley & Sons, New York, 1983.

[19] Trevor Darrell, Bruce Blumberg, Sharon Daniel, Bradley Rhodes, Pattie Maes, and Alex Pentland. Alive: Dreams and illusions. In *Visual Proceedings, ACM Siggraph.* ACM Siggraph, July 1995. (Los Angeles, CA).

[20] Trevor Darrell, Irfan Essa, and Alex Pentland. Gesture analysis in real time using interpolated views. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1996. (to appear).

[21] J. E. Dennis, Jr and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[22] Ernst Dieter Dickmanns and Volker Graefe. Dynamic monocular machine vision. *Machine Vision and Applications*, 1:223–240, 1988.

[23] R. Dutta, R. Manmatha, L. R. Williams, and E. M. Riseman. A data set for quantitative motion analysis. In *1989 IEEE Conference on Computer Vision and Pattern Recognition*, Los Alamitos, CA, June 1989. IEEE Computer Society, IEEE Computer Society Press. (San Diego, CA).

[24] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank. Camera self-calibration: Theory and experiments. In G. Sandini, editor, *Computer Vision, ECCV '92*, pages 321–334, Berlin, May 1992. DIST, University of Genova, Springer-Verlag. (Santa Margherita Ligure, Italy).

[25] Olivier Faugeras. What can be seen from an uncalibrated stereo rig? In *Proceedings of the European Conference on Computer Vision*, pages 563–578, June 1992. (Santa Margherita Ligure, Italy).

[26] Olivier D. Faugeras, Nicholas Ayache, and B. Faverjon. Building visual maps by combining noisy stereo measurements. In *Proc. IEEE Conf. on Robotics and Automation*, April 1986. (San Francisco, CA.).

[27] Martin Friedmann, Thad Starner, and Alex Pentland. Device synchronization using an optimal linear filter. In R. A. Earnshaw, M. A. Gigante, and H. Jones, editors, *Virtual Reality Systems*, chapter 9. Academic Press, London, 1993.

[28] Dariu M. Gavrila and Larry S. Davis. 3-D model-based tracking of humans in action: a multi-view approach. In *Proc. 1996 IEEE Conf. on Computer Vision and Pattern Rec.*, pages 73–80, Los Alamitos, CA, June 1996. IEEE Computer Society. (San Francisco).

[29] Arthur Gelb, editor. *Applied Optimal Estimation.* MIT Press, Cambridge, MA, 1974.

[30] W. Eric L. Grimson. Why stereo vision is not always about 3D reconstruction. A. I. Memo 1435, MIT AI Lab, Cambridge, MA, July 1993.

[31] Sir William Rowan Hamilton. *Elements of Quaternions*, volume 1,2. Longmans, Green, and Company, London, second edition, 1899 (1866).

[32] David J. Heeger and Allan D. Jepson. Subspace methods for recovering rigid motion I: Algorithm and implementation. *International Journal of Computer Vision*, 7(2):95–117, 1992.

[33] Joachim Heel. Temporally integrated surface reconstruction. In *ICCV '90*. IEEE, 1990.

[34] D. Hogg. Model-based vision: a program to see a walking person. *Image & Vision Computing*, 1(1):5–20, Feb 1983.

[35] Berthold K. P. Horn. Closed-form solution of absolute oreintation using unit quaternions. *Journal of the Optical Society of America*, 4:629, 1987.

[36] Berthold Klaus Paul Horn. *Robot Vision.* MIT Press, 1986.

[37] Peter C. Hughes. *Spacecraft Attitude Dynamics.* John Wiley & Sons, New York, 1986.

[38] H.M. Karara, editor. *Non-topographic Photogrammetry.* American Society of Photogrammetry and Remote Sensing, Falls Church, Virginia, second edition, 1989.

[39] J. J. Koenderink and A. J. van Doorn. Affine structure from motion. *Journal of the Optical Society of America*, 8:377–385, 1991.

[40] Rakesh Kumar, Harpreet S. Sawhney, and Allen R. Hanson. 3D model acquisition from monocular image sequences. In *1992 IEEE Conference on Computer Vision and Pattern Recognition*, pages 209–215, Los Alamitos, CA, June 1992. IEEE Computer Society, IEEE Computer Society Press. (Champaign-Urbana, IL).

[41] Ratnam V. Raja Kumar, Arun Tirumalai, and Ramesh C. Jain. A non-linear optimization algorithm for the estimation of structure and motion parameters. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 136–143, June 1989. (San Diego, CA.).

[42] Lennart Ljung. *System Identification.* Prentice-Hall, Inc, Englewood Cliffs, NJ, 1987.

[43] H. C. Longuet-Higgens. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[44] B. D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. 7th Intern. Joint Conf. Artif. Intell.*, 1981. (Vancouver.).

[45] D. Marr and K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Rroceedings of the Royal Society — London B*, 200:269–294, 1978.

[46] David Marr. *Vision.* Freeman, 1982.

[47] David Marr and Tomaso Poggio. A computational theory of human stereo vision. *Proc. R. Soc. Lond. B*, 204:301–328, 1979.

[48] Larry Matthies, Takeo Kanade, and Richard Szeliski. Kalman filter based algorithms for estimating depth from image sequences. *International Journal of Computer Vision*, 3(3):209–236, 1989.

[49] Baback Moghaddam and Alex Pentland. Probabilistic visual learning for object representation. In Shree Nayar and Tomaso Poggio, editors, *Early Visual Learning*, pages 99–130. Oxford University Press, 1996.

[50] Ramakant Nevatia and Thomas Binford. Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98, 1977.

[51] J. Oliensis and J. Inigo Thomas. Incorporating motion error in multi-frame structure from motion. In *IEEE Workshop on Visual Motion*, pages 8–13, Los Alamitos, CA, October 1991. IEEE Computer Society, IEEE Computer Society Press. (Nassau Inn, Princeton, NJ.).

[52] Joseph O'Roarke and Norman I. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(6):522–536, June 1980.

[53] Alex Pentland. Perceptual organization and the representation of natural form. *Artificial Intelligence*, 28(3):293–331, 1986.

[54] Alex Pentland and J. R. Williams. Good vibrations : Modal dynamics for graphics and animation. *ACM SIGGRAPH Conference Proceedings*, 23(4):215–222, 1989.

[55] Conrad J. Poelman and Takeo Kanade. A paraperspective factorization method for shape and motion recovery. CMU-CS 92-208, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-3890, October 1992.

[56] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: the art of scientific computing.* Cambridge University Press, Cambridge, U.K., second edition, 1992.

[57] W. Richards, editor. *Natural Computation.* MIT Press, 1988.

[58] L.G. Roberts. *Optical and electro optical information processing*, chapter Machine perception of three-dimensional solids, pages 159–197. MIT Press, Cambridge, MA, 1965.

[59] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding*, 59(1):94–115, Jan 1994.

[60] E. Salamin. Application of quaternions to computation with rotations. Technical report, Stanford University, Stanford, CA, 1979.

[61] L. E. Scales. *Introduction to Non-linear optimization.* Springer-Verlag, New York, 1985.

[62] Rolf Schuster. Color object tracking with adaptive modeling. In *Workshop on Visual Behaviors*, pages 91–96, Seattle, WA, June 1994. International Association for Pattern Recognition, IEEE Computer Society Press.

[63] Stan Sclaroff and Alex Pentland. On modal modeling for medical images: under-constrained shape description and data compression. In IEEE, editor, *Proc. IEEE Workshop on Biomedical Image Analysis*, June 1994. (Seattle).

[64] Amnon Shashua. Projective depth: A geometric invariant for 3-D reconstruction from two perspective/orthographic views and for visual recognition. In *Proceedings: Fourth International Conference on Computer Vision*, pages 583–590, Los Alamitos, CA, May 1993. IEEE Computer Society and Gesellschaft für Informatik, IEEE Computer Society Press. (Berlin, Germany).

[65] C. C. Slama, editor. *Manual of Photogrammetry.* American Society of Photogrammetry and Remote Sensing, Falls Church, Virginia, fourth edition, 1980.

[66] Smith and Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–58, 1987.

[67] S. Soatto, P. Perona, R. Fraezza, and G. Picci. Recursive motion and structure estimation with complete error characterization. In *1993 IEEE Conference on Computer Vision and Pattern Recognition*, pages 428–433, Los Alamitos, CA, June 1993. IEEE Computer Society, IEEE Computer Society Press. (New York).

[68] Minas Spetsakis and Yiannis Aloimonos. A multi-frame approach to visual motion perception. *International Journal of Computer Vision*, 6(3):245–255, August 1991.

[69] Gilbert Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley, MA, 1986.

[70] Richard Szeliski. Estimating motion from sparse range data without correspondence. *Proceedings of the IEEE*, 1988.

[71] Richard Szeliski. Bayesian modeling of uncertainty in low-level vision. *International Journal of Computer Vision*, 5(3):271–301, 1990.

[72] Richard Szeliski and Sing Bing Kang. Recovering 3D shape and motion from image streams using non-linear least squares. In *1993 IEEE Conference on Computer Vision and Pattern Recognition*, pages 752–753, Los Alamitos, CA, June 1993. IEEE Computer Society, IEEE Computer Society Press. (New York).

[73] Demetri Terzopolous and Dimitri Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[74] Charles W. Therrien. *Decision, Estimation, and Classification.* John Wiley & Sons, New York, NY, 1989.

[75] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, November 1992.

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

[77] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1), 1991.

[78] Shimon Ullman. The interpretation of structure from motion. *Proc. R. Soc. Lond. B*, 203:405–426, 1979.

[79] Shimon Ullman. *The Interpretation of Visual Motion.* MIT Press, Cambridge, MA, 1979.

[80] Daphna Weinshall and Carlo Tomasi. Linear and incremental acquisition of invariant shape models from image sequences. In *Proceedings: Fourth International Conference on Computer Vision*, pages 675–682, Los Alamitos, CA, May 1993. IEEE Computer Society and Gesellschaft für Informatik, IEEE Computer Society Press. (Berlin, Germany).

[81] Juyang Weng, Narendra Ahuja, and Thomas S. Huang. Optimal motion and structure estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(9):864–884, September 1993.

[82] Peter Whaite and Frank P. Ferrie. From uncertainty to visual exploration. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(10):1038–1049, October 1991.

[83] A. D. Wilson and A. F. Bobick. Learning visual behavior for gesture analysis. In *Proc. IEEE Int'l. Symp. on Comp. Vis.*, Coral Gables, Florida, November 1995.

[84] Christopher Wren, Ali Azarbayejani, Trevor Darrell, and Alex Pentland. Pfinder: Real-time tracking of the human body. In *2d International Conference on Automatic Face- and Gesture-Recognition.* IEEE, October 1996. Also, Media Lab TR 353, http://pfinder.www.media.mit.edu/projects/pfinder/.

[85] J. Yamato, J. Ohya, and K. Ishii. Recognizing human action in time-sequential images using hidden markov model. In *Proc. 1992 IEEE Conf. on Computer Vision and Pattern Rec.*, pages 379–385. IEEE Press, 1992.

[86] G-S. Young and Rama Chellappa. 3-D motion estimation using a sequence of noisy stereo images: Models, estimation and uniqueness. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(8):735–759, January 1990.