Stochastic Constraints for Vision-Aided Inertial Navigation

by

David D. Diel

Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of

Masters of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[February 2005]

January 2005

Author ....

Department of Mechanical Engineering

January 27, 2005

Certified by ....

Paul DeBitetto

Senior Member of Draper Technical Staff

Thesis Supervisor

Certified by ....

Derek Rowell

Professor of Mechanical Engineering

Thesis Supervisor

Accepted by........................ ....

Professor Lallit Anand

Chairman, Committee on Graduate Studies

Stochastic Constraints for Vision-Aided Inertial Navigation

by

David D. Diel

Submitted to the Department of Mechanical Engineering
on January 27, 2005, in partial fulfillment of the
requirements for the degree of
Masters of Science in Mechanical Engineering

Abstract

This thesis describes a new method to improve inertial navigation using feature-based constraints from one or more video cameras. The proposed method lengthens the period of time during which a human or vehicle can navigate in GPS-deprived environments. Our approach integrates well with existing navigation systems, because we invoke general sensor models that represent a wide range of available hardware. The inertial model includes errors in bias, scale, and random walk. Any camera and tracking algorithm may be used, as long as the visual output can be expressed as ray vectors extending from known locations on the sensor body.

A modified linear Kalman filter performs the data fusion. Unlike traditional Simultaneous Localization and Mapping (SLAM/CML), our state vector contains only inertial sensor errors related to position. This choice allows uncertainty to be properly represented by a covariance matrix. We do not augment the state with feature coordinates. Instead, image data contributes *stochastic epipolar constraints* over a broad baseline in time and space, resulting in improved observability of the IMU error states. The constraints lead to a relative residual and associated relative covariance, defined partly by the state history. Navigation results are presented using high-quality synthetic data and real fisheye imagery.

Thesis Supervisor: Paul DeBitetto
Title: Senior Member of Draper Technical Staff

Thesis Supervisor: Derek Rowell
Title: Professor of Mechanical Engineering

Acknowledgments

Let me not forget those who helped me on this journey. From a very young age, my mother gave me many things to observe, and the power of observation is still my most valued possession. My father never tired of answering my technical questions, so I never stopped asking them.

I cannot forget the great teachers who laid my early mathematical foundations: Mrs. Habel, Mrs. Pierce, Mrs. Geoielli, Mr. Frongello, Mike Bouch, and Larry Brown, who taught until he died in the service of his students.

From the University of Central Florida, I would like to thank Madi Dogariu, Jamal Nayfeh, Richard Zarda, and Niels da Vitoria Lobo for perceiving that I could have a career in research before I had any interest in such a thing.

In the pursuit of this thesis, several people have helped me directly. Special thanks to James Donna and Brent Appleby for describing classical inertial navigation models and concepts; J.-J. Slotine and Rami Mangoubi for insights on nonlinearities and error projections; Brock Bose for introducing me to spherical harmonics; John Leonard and Andrew Patrikalakis for use of their gantry; and Scott Rasmussen for his invaluable hardware assistance.

Thanks to Paul DeBitetto, Seth Teller, and Derek Rowell for their advisement, and for listening to me while I rambled on about issues unrelated to my work. Finally, I send brotherly love to my editors, Frank Goergen and Jeremy Parks.

Soli Deo Gloria!

Disclaimer

[illegible]

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Navigation is an old problem that periodically receives attention as new sensors are developed. Two inventions that have had a profound impact on navigation in the twentieth century are the Inertial Measurement Unit (IMU) and the Global Positioning System (GPS)[27]. A typical IMU consists of three accelerometers and three gyroscopes arranged on orthogonal axes such that the full motion of a rigid body can be measured. The GPS relies on a complex network of satellite transmitters to supply the sensing context for a number of receivers that each calculate their own position.

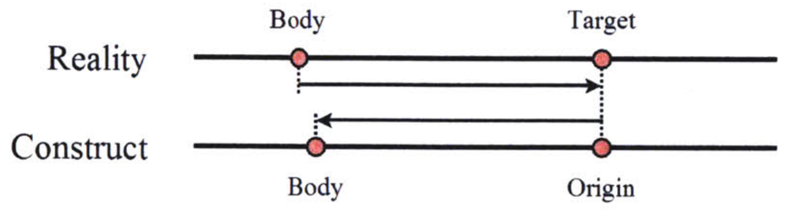
Many commercial navigation systems include an IMU and a GPS receiver. These devices support each other, so vendors will often sell them in a single package. Inertial sensors tend to capture quick motions, while the GPS receiver ensures long-term accuracy. However, circumstances such as sky occlusion, radio-reflective surroundings, and hostile jamming may deny access to GPS signals. And, when a GPS receiver fails, alternative sources of information become vital.

A video camera is a rich source of information when coupled with machine vision. During the past three decades, the practice of estimating scene Structure From Motion (SFM) has emerged, and two researchers in particular have demonstrated the current state of the art, namely Andrew Davison [13][14][15] and Alessandro Chiuso [10][11][12]. For many applications, SFM can produce a satisfactory model of local geometry. However, even the best algorithms require initialization, and as the camera moves far away from its starting point, computational requirements grow quickly and accuracy suffers.

Recently, vision-aided inertial navigation has been seriously addressed by several researchers [1][7][9][19][24][36][44][45][49]. We know that visual (optical) and inertial (vestibular) information *can* be combined, because people intuitively use these senses to navigate every day. A question of practical significance remains—*how* should we combine these senses within engineered systems, where accuracy and computation are costly resources? If we are willing to accept suboptimal performance, does our system need to maintain a map of all observed features in order to self-localize? The main problem with inertial navigation is drift. Can we simply use the camera to reduce inertial drift?

1.1 Navigation Problem

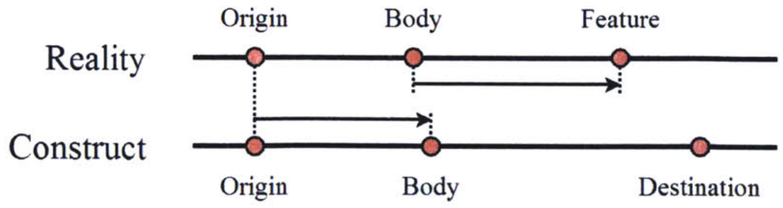
To describe our navigation problem, we begin with basic motion analysis. All motion is relative, so two or more references must be defined before distances can have meaning. The simplest system consists of two points in a single-dimensional space.



In the illustration above, the separation of physical reality from its mathematical construct has been emphasized in order to show the choice of origin and the possibility of inaccuracy. We call this scenario *target-relative navigation*, because the origin of the construct has been defined as the target itself. The body position is modeled by a single number—the directed distance from the origin to the body. If a range sensor were attached to the body, it would operate in the space of physical reality, measuring the directed distance from the body to the target.

Suppose someone wanted to bring the body in contact with the target[1]. Given range measurements and a means of actuation, a variety of control schemes could solve this problem. The accuracy and sampling frequency of the range sensor would affect the solution, but one can find real-world examples where the construct converges to reality and the body itself converges to the target. For an relevant example, see Huster's work on vision-aided robotic grasping [24]

A navigation system generally combines elements of self-localization, path planning, control, and actuation for the purpose of transporting people or material goods. In this thesis, we will focus on the self-localization part of the system. It is assumed that a person or intelligent agent will take care of the other components. Therefore, the single-dimensional version of our navigation problem looks like this:



We call this scenario *map-relative navigation*, because the destination is defined as a location on a map (one type of mathematical construct). The map origin corresponds to an invisible fixed point somewhere in the space of reality. Since our system relies on cameras,

[1]The body and target locations are concrete examples of states. In general, we could refer to the actual and desired states of any state-vector system.

we have introduced another reference—the point feature. Nothing is known about point features in advance, except that they do not move. Finally, an arrow has been drawn to show a hypothetical range measurement from the body to a feature[2].

Self-localization in reality means finding the six Degree-Of-Freedom (DOF) transformations $\boldsymbol{x}[t]$ that relate body poses to a map origin. Note that neither of our sensors are capable of observing the origin[3], so initial misalignment of the map cannot be corrected by runtime measurements.

We are specifically interested in terrestrial navigation. Likely environments include warehouses, factories, offices, homes, city streets, suburbs, highways, rural areas, forests, and caves. These environments happen to share some important visual characteristics: 1) Most of the scene remains stationary with respect to the planet's surface; and 2) For typical scenes, the ratio of body velocity to scene depth lies within a limited range. Other environments that meet these criteria include the sea floor and unexplored planets. The sky, outer space, and most seascapes do not fall in this category.

1.2 Unified Notation

The bodies of literature on navigation, estimation, and machine vision contain many conflicting notations. Given the limited number of characters in the Greek and Roman alphabets, one can easily see why variables are reused in different contexts. We present a unified notation for vision-aided inertial navigation that maintains moderate compatibility across fields. In selecting the symbols and ornaments shown in Tables 1.1 and 1.2, we have tried to avoid visual ambiguities, such as the confusion between the number 1 and the letter l.

1.3 Chapter Descriptions

In Chapter 2, we clarify our approaches toward vision and estimation. Constraint-based localization is presented as a computationally efficient alternative to feature mapping.

Chapter 3 defines common frames of reference, and describes classical inertial navigation relative to the Earth geoid. The problem is further clarified by an analysis of how inertial error propagates over time. Chapter 4 then gives the reader enough information to reproduce our vision system, which is otherwise treated as a black box.

Our method of data fusion is presented in Chapter 5. This is the glue that ties all of the other pieces together in what we call the *epipolar constraint filter*.

Finally, Chapters 6–7 analyze the response of the filter as it operates on real and synthetic data sets. Suggestions for future work are offered in Chapter 8.

[2]Most cameras measure radiant light, not range. Exceptions include active-focus and structured-light cameras, which are outside the scope of this thesis.

[3]In saying this, we are ignoring the marginal observability of latitude and height above the geoid through measurements of the gravity potential and of planetary rotation.

Symbol	Meaning	Units	Type
d	distance	$meters$	scalar
ℓ	world radiance	$lumens$	constant scalar function
t	time	$seconds$	scalar
v	gravitational potential	$\frac{meters^2}{second^2}$	constant scalar function
y_{ij}	pixel value	$lumens$	array of scalars
γ	geodetic latitude	$radians$	scalar
ϵ	relatively small number	$dependent$	scalar
κ_{ij}	corner strength	$\frac{lumens}{pixel^2}$	array of scalars
λ	geocentric latitude	$radians$	scalar
σ	statistical deviation	$dependent$	scalar
τ	interval of time	$seconds$	constant scalar
$\boldsymbol{b}$	bias	$dependent$	vector
$\boldsymbol{c}_{ij}$	camera ray	$none$	array of vectors
$\boldsymbol{f}$	specific force	$\frac{meters}{second^2}$	vector
$\boldsymbol{g}$	gravity	$\frac{meters}{second^2}$	vector
$\boldsymbol{h}$	measurement gain	$dependent$	vector
$\boldsymbol{k}$	Kalman gain	$dependent$	vector
$\boldsymbol{n}$	white noise	$dependent$	random vector
$\boldsymbol{p}$	general point	$meters$	vector
$\boldsymbol{q}$	kinematic joint state	$dependent$	vector
$\boldsymbol{s}$	scale	$none$	vector
$\boldsymbol{u}_{ij}$	normalized image coordinate	$none$	array of vectors
$\boldsymbol{x}$	body state	$radians, meters$	vector
$\boldsymbol{z}$	feature observation ray	$none$	vector
$\boldsymbol{\zeta}$	end-effector state (body frame)	$radians, meters$	vector
$\boldsymbol{\theta}$	Euler angles (body frame)	$radians$	vector
$\boldsymbol{\rho}$	imaging parameters	$none$	vector
$\boldsymbol{\psi}$	inertial sensor state	$mixed$	vector
$\boldsymbol{\omega}$	angular velocity (body frame)	$\frac{radians}{second}$	vector
$\boldsymbol{E}$	edge energy (Hessian)	$\frac{lumens}{pixel^2}$	matrix
$\boldsymbol{R}$	rotation	$none$	matrix
$\boldsymbol{\Lambda}$	covariance	$dependent$	matrix
$\boldsymbol{\Phi}$	state transition	$dependent$	matrix

Table 1.1: Definitions of mathematical symbols. Scalars are regular italic; vectors are bold lowercase; and matrices are bold uppercase. Values may vary with time unless noted.

Ornament	**Meaning**
$\diamond(\,)$	continuous samples
$\diamond[\,]$	discrete samples
$\diamond_T$	translation part
$\diamond_R$	rotation part (quaternion)
$\diamond^{\mathrm{T}}$	transpose
$\diamond^\circ$	units of $degrees$
$\dot{\diamond}$	1^{st} temporal derivative
$\ddot{\diamond}$	2^{nd} temporal derivative
$\vec{\diamond}$	unit magnitude
$\bar{\diamond}$	intermediate estimate
$\tilde{\diamond}$	residual
$\hat{\diamond}$	estimated
$\mathrm{d}\diamond$	derivative
$\partial\diamond$	partial derivative
$\delta\diamond$	error or perturbation
$\Delta\diamond$	change
$\nabla\diamond$	gradient

Table 1.2: Definitions of symbolic ornaments. The $\diamond$ symbol is a placeholder.

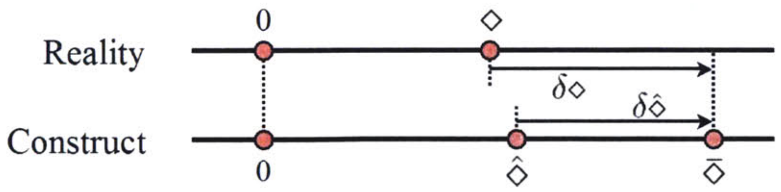


Figure 1-1: Diagram of selected symbolic ornaments.

Chapter 2

Approach

The task of self-localization can be posed as a problem in stochastic geometry [4][50]. Neither IMUs nor cameras produce geometric models, but their measurements are related to geometry. An IMU observes motion through a small window in time, while the camera observes projections of space through the complicated process of image formation [38][39]. Given noisy data from both sensors, we seek the most likely body path that could have caused the data. Some researchers have addressed this problem by minimizing reprojection error in the image space [32], but we attempt to minimize position error in the world space. Others define the body path incrementally through visual velocity estimates, but we will tackle pose estimation directly.

To make progress, we adopt some basic assumptions. The body begins at rest in a North-East-Down frame that is fixed to a geoid of known shape and gravitation. The initial position, orientation, and geodetic latitude of the body are given. We assume that the surroundings are mostly static and illuminated by steady ambient lighting. The optical system has been calibrated, and the error characteristics of all sensors have been determined by experimentation.

We also impose several constraints on what can be considered an acceptable solution. Similar to Chiuso, we only permit *causal* schemes—based on information from the past and present, but not the future [10][11]. Our system should operate without user intervention or access to external networks. The camera model should accommodate various lens types, including active lenses. The vision system should be robust to moderate lighting variation and changes in viewpoint. And finally, the data fusion algorithm should handle an occasional dropped frame or a temporary lens-cap situation.

2.1 Sensor Configurations

This development applies to a range of plausible sensor configurations. A single six Degree-Of-Freedom (DOF) IMU is rigidly attached to a body, such that the IMU and body coordinate frames are coincident. One or more cameras can then be attached to the body

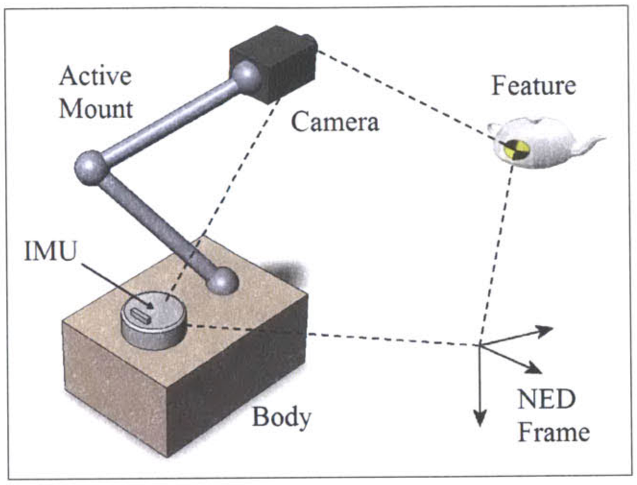


Figure 2-1: Sensor configuration with active camera mount.

via passive or active kinematic linkages. The physical mount will introduce an offset and a rotation for each camera, which can be represented by a simple transformation (see Section 5.2.4).

Outside of Chapter 4, our vision system will be treated as a black-box that produces ray vectors corresponding to tracked points. This layer of abstraction allows various camera types and projections to be accommodated. For example, infrared cameras could be used to handle low-light conditions, and parabolic and fisheye projections could be used together within the same system. Variable telephoto lenses are also covered by this abstraction, although a wide field-of-view lens is preferred because it allows more points to be tracked for longer periods of time.

2.1.1 Gyroscope Reliance

In this work, the body orientation is calculated by integration of the gyroscope (gyro) output. The gyros are used to compensate for camera rotation; however, visual data does not contribute the orientation estimate. In other words, our filter estimates and removes translation error only. This is not a huge limitation or barrier to implementation, because modern gyros can be trusted to provide orientation estimates for periods up to several minutes [9]. Nevertheless, the practical length of an excursion will be limited by the drift of the gyros.

For many hardware combinations, gyro accuracy exceeds imaging accuracy by orders of magnitude. Consider a typical wide-field video camera with a pixel separation of 0.25 degrees. A common flight control gyro can maintain similar angular precision for about 10 minutes with no visual assistance. Therefore, during an observation baseline of 2–3 minutes, the vision system could remove no more than a half-pixel of angular drift. However, after 20–30 minutes, the unmodeled gyro error could significantly degrade the performance of the whole system.

2.2 Applied Machine Vision

Several ideas from machine vision have potential relevance to inertial navigation. The list would certainly include stereo vision, optical flow, pattern recognition, and tracking of points, curves, and regions. Each of these methods produces a different kind of information. Stereo vision can produce dense relative depth estimates within a limited range. Optical flow can produce dense motion fields [6][22]. Pattern recognition can locate the direction to a unique landmark [17], which may be linked to absolute coordinates. And finally, various tracking methods can offer data association over multiple frames [41].

The scope of relevance can be narrowed by considering our requirements. Since the sensor platform might not be equipped with multiple cameras, we eliminate stereo vision. Pattern recognition demands a lot of memory to store the feature labels and world coordinates of numerous objects. In our unstructured environment, there would be no way to bound the memory requirements, so this option must also be eliminated. Therefore, we are left with tracking-based and flow-based approaches.

2.2.1 Tracking vs. Flow

Both optical flow and tracking provide local estimates of image motion. As a camera translates, light rays from the environment slice through a theoretical unit sphere centered at the camera's focus. Patterns of light appear to flow outward from a point source, around the sides of the sphere, and into a drain at the opposite pole. If the camera also rotates, then a vortex-like flow is superimposed on the flow induced by translation. Despite the naming conventions, both optical flow and tracking are supposed to track the flow. The difference between them lies in the selection of discrete coordinates, as demonstrated by Figure 2-2. Optical flow calculations are defined on a discrete regular mesh, with the assumption of underlying spatial and temporal continuity. In contrast, tracking methods tend to utilize discrete particles that retain their identity over a broad baseline in time. A particle could represent any kind of geometric entity, including points, lines, and curves [2][8].

We choose to track corner features, because they offer a powerful form of data association over time. Unlike smooth image regions, corner features have a high information content, and are easily tracked. The coordinates of a corner can be clearly defined and historically associated without regard to other parts of the image. Consequently, mistracked features in a local region will have little effect on the others in the system.

2.3 Data Fusion

Our data fusion strategy is to play the strength of the camera against the weakness of the IMU. Inertial estimates integrate small amounts of error over time, resulting in large amounts of long-term drift. In contrast, the accuracy of image data is mostly independent

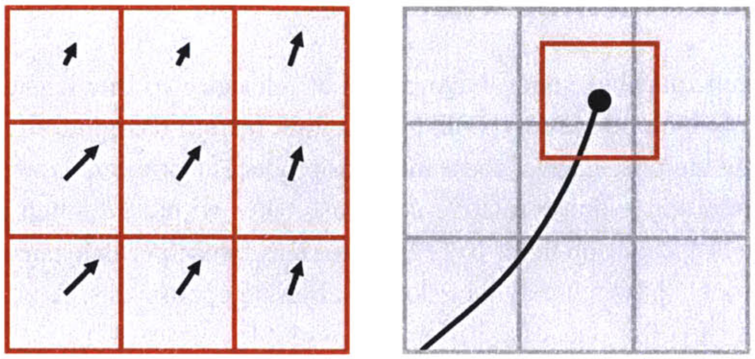


Figure 2-2: Visualizations of two approaches to motion extraction. Left—The displacement field calculated at a fixed number of discrete points on a grid, often called "Optical Flow" and associated with Euler. Right—The trajectory of a single particle, often called "Feature Tracking" and associated with Lagrange.

of time. Our concept is to anchor the current estimate of the body position to the state history through observations of visual constraints.

We will build our estimation algorithm around a single feature observation in a single camera. If multiple cameras and multiple features are available, then they will be handled by duplicating the process for one feature. This choice simplifies our work, although it may not result in the best estimator. The underlying assumption is that each feature measurement has independently distributed error. If that assumption is not true, then we can only hope that the errors are weakly correlated.

Probably the most common approach to combining feature observations with inertial data is Simultaneous Localization and Mapping (SLAM) [15][19][28]. Mapping offers the potential for *landmark recognition* and *loop closure*, but these extra benefits come at a high computational cost. Consider the overwhelming number of features that can be observed with a video camera. Several thousand salient features may appear during a one-minute excursion. And, it is possible that many of those features will be virtually indistinguishable in appearance. This leads to a web of uncertain data associations coupled with uncertain feature position estimates.

Our problem is essentially SLAM without mapping. Nevertheless, we can learn from solutions that involve mapping. For instance, SLAM can be solved by brute-force state augmentation and optimal filtering. Every observed feature can be added to a large state vector. Then, for a linear system with Gaussian measurement noise, the Kalman Filter (KF) recursively incorporates all of the information in each measurement [26], resulting in the most likely estimate of the body and feature states. For nonlinear systems, one can use the Extended Kalman Filter (EKF) without the same guarantee of optimality [25]. However, in the KF and EKF, computation scales quadratically with the number of states, and becomes intractable after approximately 300 features or 900 states have been included[1].

2.3.1 An Alternative to SLAM

Consider the one-dimensional map-relative navigation problem from the previous chapter. Suppose the body begins at $x(0) = 0$ and translates in the horizontal plane with a sinusoidal motion $x = \sin\left(\frac{\pi t}{20}\right)$. If a linear accelerometer is attached to the body, then it will measure $\bar{f} = \ddot{x} + n_w$, where $n_w \sim \mathcal{N}(0, \sigma_w)$ represents zero-mean Gaussian noise. Chapter 3 will present a more detailed error model, but for now, we will assume that the noise is uncorrelated over time.

Without any other information, the best estimate of the body position is given by integrating the accelerometer output, as in $\hat{x} = \bar{x} = \iint \bar{f}\,\mathrm{d}t = x + \delta x$. The error itself can also be expressed in state-space form:

$$\begin{bmatrix} \delta\dot{x} \\ \delta\ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta\dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ n_w \end{bmatrix} \tag{2.1}$$

The state-space equations document the propagation of uncertainty, but the error cannot be corrected without additional measurements. Ordinary integration is equivalent to *inertial dead-reckoning*, and it results in unbounded error growth, as shown in Figure 2-3.

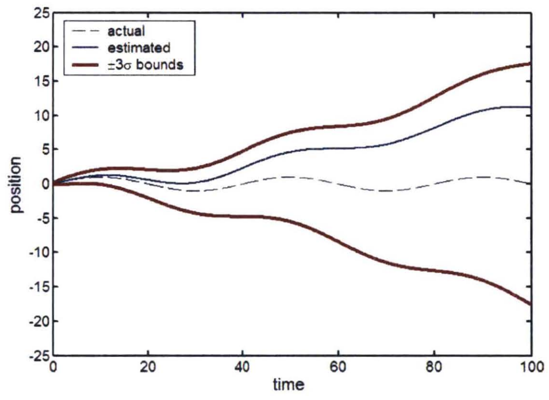


Figure 2-3: Position estimation by integrating accelerometer output.

Now, suppose that a single point feature exists at $p = 5$, and the body is equipped with a distance sensor. The sensor measures $\bar{d} = p - x + n_d$ with a sampling frequency of $10Hz$, where $n_d \sim \mathcal{N}(0, \sigma_d)$. Given this additional information, we can consider methods for fusing data from multiple sources. Two approaches are presented below; SLAM and constraint-based localization.

[1]Extensions such as *Atlas* [7] and *FastSLAM* [37] have been developed to manage large numbers of features by breaking them up into small groups, and these alternatives are worthy of further investigation.

A SLAM approach implies that the feature should be mapped, and this can be done by including p in the state-space. Since the feature does not move, its dynamics amount to $\dot{p} = 0$, or equivalently $\delta\dot{p}_o = 0$, where $\delta p_o \equiv \bar{p}[0] - p$. The expanded set of equations can be written

$$\begin{bmatrix} \delta\dot{x} \\ \delta\ddot{x} \\ \delta\dot{p}_o \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta\dot{x} \\ \delta p_o \end{bmatrix} + \begin{bmatrix} 0 \\ n_w \\ 0 \end{bmatrix} \tag{2.2}$$

Then, noticing that $\bar{d} = d + \delta d$, we can formulate a measurement model that is a function of the states

$$\delta d = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta\dot{x} \\ \delta p_o \end{bmatrix} + n_d \tag{2.3}$$

With these equations, the KF can be applied directly, and it will drive $\delta\hat{d} \rightarrow \delta d$, and therefore $\hat{d} \rightarrow d$. Figure 2-4 shows an example of the simulated filter response. Notice how the body position error becomes bounded, and the feature position estimate settles down to a constant value around $\hat{p} = 5.5$.

Constraint-based localization has much in common with SLAM. The noise models are identical, and the inertial error dynamics come from Equation 2.1. However, we introduce a subtle difference in the measurement equation:

$$\delta d = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta\dot{x} \end{bmatrix} + n_d \tag{2.4}$$

Equations 2.3 and 2.4 appear to contradict one another, but they can both be true if we change one of the noise distributions. In the second case, $n_d \sim \mathcal{N}(-\delta p_o, \sigma_d)$ balances the equation. However, since δp_o remains unknown, we cannot inform the KF of this new error distribution. So, in our new approach, we will run the filter with $n_d \sim \mathcal{N}(0, \sigma_d)$, and ignore the unmodeled bias.

Referring again to Figure 2-4, one can see that the result of constraint-based localization is not as good as state augmentation. In the latter simulation, an unwanted bias from the flawed noise model has perturbed the body position estimate. Why would anyone want to choose the latter option? First, recall that the camera is not a distance sensor, so the measurement specified here must be analogous to an attainable measurement from the camera (for more detail, see Section 5.1). Second, we can avoid the computation of p or δp. Therefore, requirements for memory and processing will grow linearly (not quadratically) with the number of visible features, and we never have to worry about points at infinity. Third, if multiple features are observed, then the effects of their unmodeled biases will balance one another. Specifically, given N independent simultaneous observations with equal uncertainty σ_d, their average will have an unmodeled uncertainty $\sigma_{avg} = \frac{\sigma_d}{\sqrt{N}}$. This reduction of uncertainty is very desirable, since images tend to contain a lot of features.

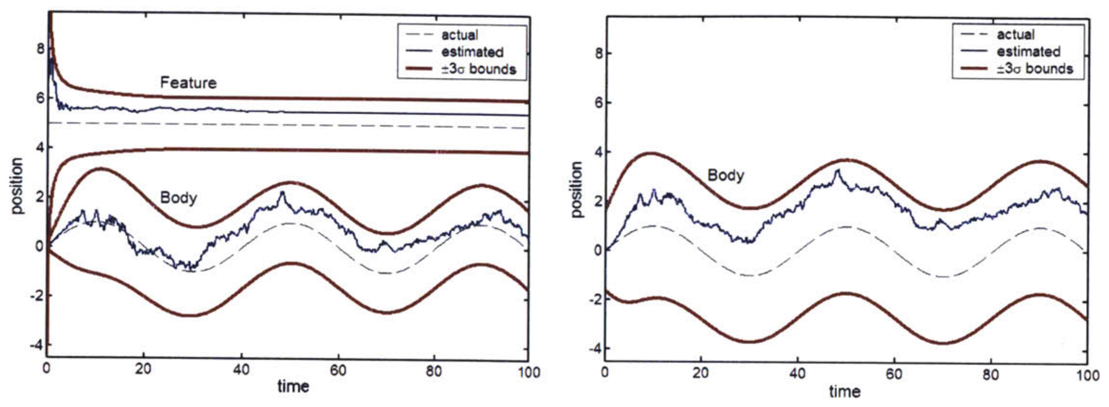


Figure 2-4: Left—SLAM solution by state augmentation. Right—Constraint-based localization.

2.3.2 System Block Diagram

The block diagram in Figure 2-5 foreshadows our concept for a class of vision-aided inertial systems. There are seven components, and most of the data flow is unidirectional. The left half of the diagram connects to the right half through a minimal exchange of information (typically on the order of 100kBPS), which might suggest dedicated hardware on each side. Besides power and an unobstructed view of the world, the system requires no external interface beyond what is shown.

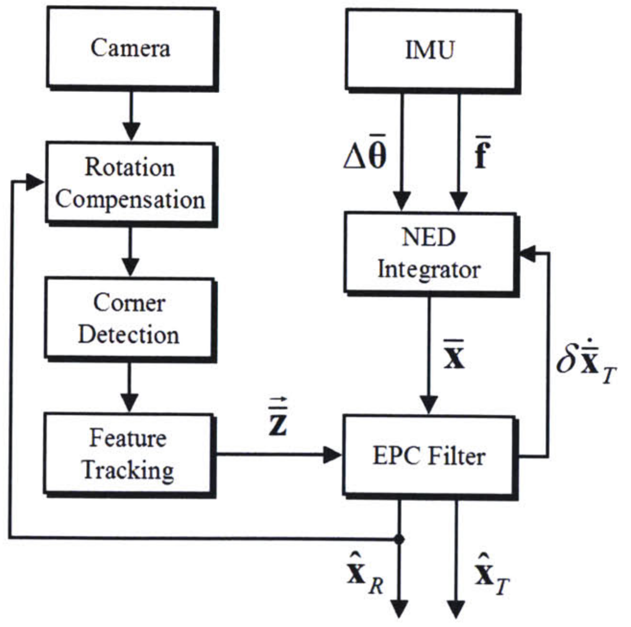


Figure 2-5: Proposed block diagram showing data fusion in the EPC Filter.

Chapter 3

Inertial Measurements

An Inertial Measurement Unit (IMU) senses rotation rates and specific forces associated with a moving body. A typical packaged IMU contains accelerometers and gyroscopes placed on on each of three orthogonal axes. They come in two common configurations: the *gimballed* type, which maintains a fixed orientation while the body rotates around it; and the *strapdown* type, which rotates with the body. In this chapter, we will characterize accelerometer error in a strapdown IMU[1].

Consider a model of an ideal inertial sensor. The gyros would output small changes in orientation $\Delta\boldsymbol{\theta}[t]$, which could be integrated to find an associated direction cosine matrix $\boldsymbol{R}(t)$. And, the accelerometers would output specific force, which is a combination of acceleration and apparent gravitation:

$$\boldsymbol{f} = \boldsymbol{R}^{-1}\left(\ddot{\boldsymbol{x}}_T - \boldsymbol{g}_{app}\right) \tag{3.1}$$

This model describes perfect sensors, but it contains hidden uncertainty in the term $\boldsymbol{g}_{app}$. Apparent gravity cannot be determined exactly. Instead, one must select a method to calculate its value, understanding that each method has limited accuracy. For example, one could simply assume a constant value $\boldsymbol{g}_{app} = [0\ 0\ 9.8]^{\mathrm{T}} \frac{meters}{second^2}$ near the Earth's surface. Another method is to let the inertial sensor sit still, and take gravity to be the average accelerometer output over a period of time.

Section 3.2 presents an expression for apparent gravity near the Earth that is accurate to the fifth decimal place (about 16 bits of information). It varies strongly with latitude and altitude, and weakly with velocity. We use this expression because its error is small enough to be neglected[2].

[1]The development for a gimballed IMU can be derived by analogy.

[2]Relative to a tactical-grade IMU.

3.1 Accelerometer Error

The basic concept of inertial navigation is to determine the location of a body by integrating IMU output with respect to time. Unfortunately, errors in the derivative measurements are integrated along with the desired signal.

Extensive studies of accelerometers have shown that their principal errors fall into three categories—bias, scale, and random walk. This is evidenced by manufacturers' reports [30], and verified by experts in the field [16]. These additive errors show up in the measurement of specific force as follows:

$$\bar{\boldsymbol{f}} = (\boldsymbol{I} + \text{diag}\,(\delta \boldsymbol{s}))\,\boldsymbol{f} + \delta \boldsymbol{b} + \boldsymbol{n}_w \tag{3.2}$$

where $\boldsymbol{n}_w$ represents Gaussian white noise. Rewriting Equation 3.2 as an acceleration measurement yields

$$\begin{aligned} \bar{\ddot{\boldsymbol{x}}}_T &= \boldsymbol{R}\bar{\boldsymbol{f}} + \boldsymbol{g}_{app} \\ &= \left(\ddot{\boldsymbol{x}}_T - \boldsymbol{g}_{app}\right) + \boldsymbol{R}\left(\text{diag}\,(\delta \boldsymbol{s})\,\boldsymbol{f} + \delta \boldsymbol{b} + \boldsymbol{n}_w\right) + \boldsymbol{g}_{app} \\ &= \ddot{\boldsymbol{x}}_T + \boldsymbol{R}\text{diag}\,(\boldsymbol{f})\,\delta \boldsymbol{s} + \boldsymbol{R}\delta \boldsymbol{b} + \boldsymbol{n}_w \end{aligned} \tag{3.3}$$

Therefore, omitting the second order effect of diag $(\delta \boldsymbol{f})\,\delta \boldsymbol{s}$, the error process may be defined

$$\delta \ddot{\boldsymbol{x}}_T \equiv \bar{\ddot{\boldsymbol{x}}}_T - \ddot{\boldsymbol{x}}_T = \boldsymbol{R}\text{diag}\left(\bar{\boldsymbol{f}}\right)\delta \boldsymbol{s} + \boldsymbol{R}\delta \boldsymbol{b} + \boldsymbol{n}_w \tag{3.4}$$

The bias and scale error terms can be further broken-down into a "turn-on" component and an "in-run" component:

$$\delta \boldsymbol{b} = \delta \boldsymbol{b}_{TurnOn} + \delta \boldsymbol{b}_{InRun} \tag{3.5}$$

$$\delta \boldsymbol{s} = \delta \boldsymbol{s}_{TurnOn} + \delta \boldsymbol{s}_{InRun} \tag{3.6}$$

The turn-on error is constant, but in-run errors vary slowly over time. The in-run characteristic can be modeled by passing Gaussian white noise through a low-pass filter:

$$\delta \dot{\boldsymbol{b}}_{InRun} = -\frac{1}{\tau_b}\delta \boldsymbol{b}_{InRun} + \boldsymbol{n}_b \tag{3.7}$$

$$\delta \dot{\boldsymbol{s}}_{InRun} = -\frac{1}{\tau_s}\delta \boldsymbol{s}_{InRun} + \boldsymbol{n}_s \tag{3.8}$$

Putting Equations 3.2–3.8 together leads to a Linear-Time-Varying (LTV) state-space model for the body position error. Note the nonlinearity with respect to external inputs $\boldsymbol{R}$ and $\bar{\boldsymbol{f}}$, though no elements of $\boldsymbol{\psi}$ appear in $\boldsymbol{\Phi}$:

$$\dot{\boldsymbol{\psi}} = \boldsymbol{\Phi}\boldsymbol{\psi} + \boldsymbol{n} \tag{3.9}$$

$$\boldsymbol{\psi} \equiv \begin{bmatrix} \delta\boldsymbol{x}_T \\ \delta\dot{\boldsymbol{x}}_T \\ \delta\boldsymbol{b}_{TurnOn} \\ \delta\boldsymbol{b}_{InRun} \\ \delta\boldsymbol{s}_{TurnOn} \\ \delta\boldsymbol{s}_{InRun} \end{bmatrix} \qquad \boldsymbol{n} = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{n}_w \\ \mathbf{0} \\ \boldsymbol{n}_b \\ \mathbf{0} \\ \boldsymbol{n}_s \end{bmatrix}$$

$$\boldsymbol{\Phi} = \begin{bmatrix} 0 & \boldsymbol{I} & 0 & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{R} & \boldsymbol{R} & \boldsymbol{R}\mathrm{diag}\left(\bar{\boldsymbol{f}}\right) & \boldsymbol{R}\mathrm{diag}\left(\bar{\boldsymbol{f}}\right) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{\boldsymbol{I}}{\tau_b} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{\boldsymbol{I}}{\tau_s} \end{bmatrix}$$

To calculate the body position without image data, one would twice integrate the first form of Equation 3.3. Given image data, one would also integrate Equation 3.9 and apply stochastic updates yet to be defined.

Identifier	**Accelerometers**						
	Bias			Scale			Walk
	σ_{TurnOn}	σ_{Steady}	τ	σ_{TurnOn}	σ_{Steady}	τ	σ
	$\frac{meters}{second^2}$	$\frac{meters}{second^2}$	$seconds$	$none$	$none$	$seconds$	$\frac{meters}{second^{\frac{3}{2}}}$
LN1	2.5×10^{-4}	9.8×10^{-5}	60	5.0×10^{-6}	0*	∞*	5.0×10^{-5}
LN2	2.0×10^{-3}	4.9×10^{-4}	60	3.0×10^{-4}	0*	∞*	5.0×10^{-4}
IMU1	2.0×10^{-2}	9.8×10^{-3}	60	1.3×10^{-4}	6.0×10^{-4}	60	3.3×10^{-4}
IMU2	2.5×10^{-1}	3.2×10^{-2}	60	3.0×10^{-3}	3.0×10^{-3}	60	1.5×10^{-3}

Identifier	**Gyroscopes**						
	Bias			Scale			Walk
	σ_{TurnOn}	σ_{Steady}	τ	σ_{TurnOn}	σ_{Steady}	τ	σ
	$\frac{radians}{second}$	$\frac{radians}{second}$	$seconds$	$none$	$none$	$seconds$	$\frac{radians}{\sqrt{second}}$
LN1	1.5×10^{-8}	1.5×10^{-8}	100	5.0×10^{-6}	0*	∞*	2.9×10^{-7}
LN2	4.8×10^{-6}	1.7×10^{-6}	100	1.0×10^{-4}	0*	∞*	2.0×10^{-5}
IMU1	1.5×10^{-5}	2.4×10^{-5}	100	7.0×10^{-5}	1.0×10^{-4}	100	1.5×10^{-5}
IMU2	9.7×10^{-4}	9.7×10^{-4}	100	1.0×10^{-3}	1.0×10^{-3}	100	8.7×10^{-4}

Table 3.1: Typical IMU parameter values derived from product manuals. These numbers have not been approved by any manufacturer. They are presented for order-of-magnitude comparison only. *Indicates a reasonable substitution for missing data.

3.1.1 Discrete Implementation

There are some small but important differences between the theory of the previous section and its implementation. First, the continuous model needs to be converted to a discrete one

$$\begin{aligned} \boldsymbol{\psi}[t] &= \text{expm}(\Delta t \boldsymbol{\Phi}[t-\Delta t])\, \boldsymbol{\psi}[t-\Delta t] \\ \boldsymbol{\Lambda}[t] &= \text{expm}(\Delta t \boldsymbol{\Phi}[t-\Delta t])\, \boldsymbol{\Lambda}[t-\Delta t]\, \text{expm}(\Delta t \boldsymbol{\Phi}[t-\Delta t])^{\mathrm{T}} + \Delta t \boldsymbol{\Lambda}_n \end{aligned} \tag{3.10}$$

Here, expm( ) represents the matrix exponential function, and Δt is a small time step, typically in the range of $\frac{1}{50}$ to $\frac{1}{1000}$ *seconds*. The initial uncertainty is given by

$$\boldsymbol{\Lambda}[0] = \text{blockdiag}\left(\boldsymbol{\sigma}^2_{\delta x}, \boldsymbol{\sigma}^2_{\delta \dot{x}}, \sigma^2_{bTurnOn}\boldsymbol{I}, \mathbf{0}, \sigma^2_{sTurnOn}\boldsymbol{I}\right) \tag{3.11}$$

The driving term $\boldsymbol{\Lambda}_n$ in Equation 3.10 corresponds to the noise $\boldsymbol{n}$. Since the noise is zero-mean Gaussian, uncorrelated with itself over time, its first-order expectation is $\text{E}[\boldsymbol{n}] = \mathbf{0}$, and its second-order expectation is the diagonal matrix

$$\boldsymbol{\Lambda}_n = \text{E}\left[\boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}\right] = \text{blockdiag}\left(\mathbf{0}, \sigma^2_w \boldsymbol{I}, \mathbf{0}, \sigma^2_b \boldsymbol{I}, \mathbf{0}, \sigma^2_s \boldsymbol{I}\right) \tag{3.12}$$

Now, the driving noise for a discrete system differs from that of a continuous system. Given the values in Table 3.1, the discrete versions of σ_b and σ_s can be derived (σ_w remains the same). Consider the one-dimensional bias variability equation:

$$\delta \dot{b}_{InRun} = -\frac{1}{\tau_b}\delta b_{InRun} + n_b \tag{3.13}$$

Noticing that the error dynamics are slow compared to the IMU time step ($\Delta t \ll \tau$), we choose a first-order approximation to the exponential function

$$\delta b_{InRun}[t] = \left(1 - \frac{\Delta t}{\tau_b}\right)\delta b_{InRun}[t-\Delta t] + \Delta t n_b \tag{3.14}$$

Then, consider the steady-state variance

$$\begin{aligned} \sigma^2_{bSteady} &= \lim_{t\to\infty} \text{E}\left[\delta b^2_{InRun}[t]\right] \\ &= \frac{\Delta t^2 \sigma_b^2}{1 - \left(1 - \frac{\Delta t}{\tau_b}\right)^2} \end{aligned} \tag{3.15}$$

Therefore, solving for the amplitude of the discrete driving noise n_b, we have

$$\sigma_b = \sigma_{bSteady}\sqrt{\frac{2}{\tau_b \Delta t} - \frac{1}{\tau_b^2}} \tag{3.16}$$

And, by analogy we also know the amplitude of n_s:

$$\sigma_s = \sigma_{sSteady}\sqrt{\frac{2}{\tau_s \Delta t} - \frac{1}{\tau_s^2}} \tag{3.17}$$

3.1.2 Hardware Cost

Identifier	Estimated Price
	dollars
LN1	80,000
LN2	20,000
IMU1	1,200
IMU2	500

Table 3.2: Typical IMU prices.

Navigation hardware can be expensive, so businesses are often interested in the tradeoff between cost and quality. For the right amount of money, greater than $\$1,000,000$, one can buy an IMU capable of circling the Earth with no visual assistance. On the other hand, there are sensors that cost much less, but could never be used for unaided navigation. We speculate that a highly profitable market exists for low-end IMUs coupled with vision.

3.2 Geological Effects

In this work, objects composing a scene are rigidly attached to a *rotating* planet. The visual navigation problem is most naturally expressed relative to the scene. However, inertial measurements are naturally expressed relative to an inertial frame. Three frames are involved all together; the inertial, the scene, and the body. Our goal is to understand the motion of a body frame relative to the inertial frame, as expressed in the scene frame.

Frame definitions are mostly, though not entirely, arbitrary. None of the sensors presented in this chapter are capable of measuring a planetary orbit around the sun, so we treat the geocentric (GC) frame as an inertial reference. The GC frame does not rotate, but the geoid rotates around it. For the body frame, we choose the forward-right-down convention. And for the scene reference, we choose a North-East-Down (NED) frame located at the origin of the body path. The fixed angle γ defines the geodetic latitude of the NED origin.

3.2.1 WGS 84 Geoid Model

To explain the specific forces associated with a body frame, one must first describe planetary geometry (geoids). There are few planetary models to choose from, and creating a

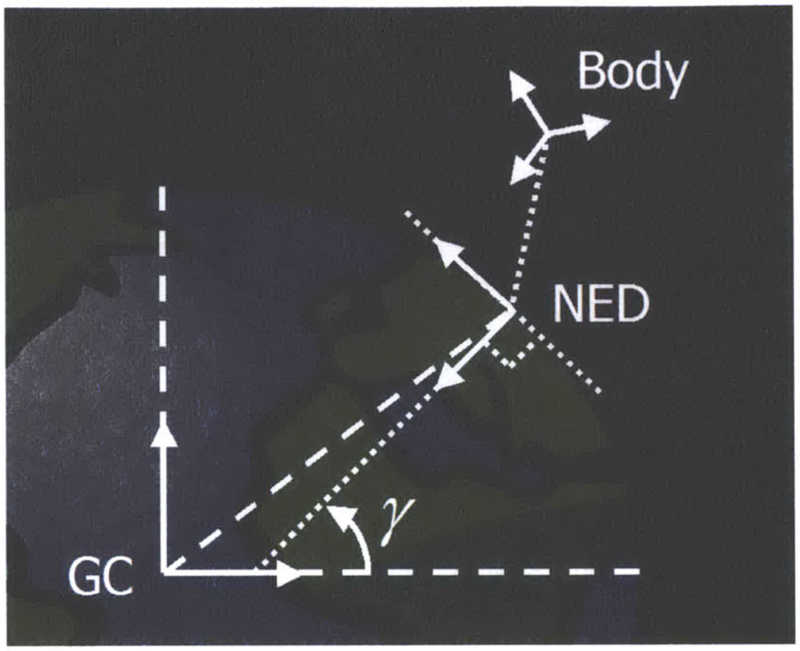


Figure 3-1: Navigation coordinate frames related to the ellipsoidal Earth model. The Earth rotates around the GC frame, and the NED frame is fixed to the Earth's surface. The body altitude is exaggerated to show its separation from the Earth.

new model would be far beyond the scope of this thesis. So, we appeal to a model standardized by the scientific community known as the 1984 World Geodetic System, or simply "WGS 84" [18][53]. Table 3.3 defines the WGS 84 parameters that will be used subsequent sections.

The standard model defines a spherical harmonic potential representing gravity plus centripetal acceleration. Spherical harmonics bear resemblance to a 3-dimensional fourier basis set. Iso-surfaces of the 1^{st} harmonic of the potential are spheres, and iso-surfaces of the potential formed by the first two harmonics are ellipsoids. For example, the physical surface of the Earth known as *sea level* is a particular iso-surface of the 2^{nd}-order potential. In practice, the first two spherical harmonics dominate the higher-order harmonics.

WGS 84 provides a model of the gravity potential, with centrifugal effects. Dropping all terms beyond the 2^{nd} harmonic results in the following potential:

$$v = \frac{GM}{\|\boldsymbol{p}\|} + \frac{\sqrt{5}GMr_e^2C_{20}}{2\|\boldsymbol{p}\|^3}\left(3\sin^2\lambda - 1\right) \tag{3.18}$$

where $\boldsymbol{p}$ is the 2D position of the body relative to the geoid center, and λ is the geocentric latitude of the body. Then, the gradient operator can be rewritten in the local NED frame

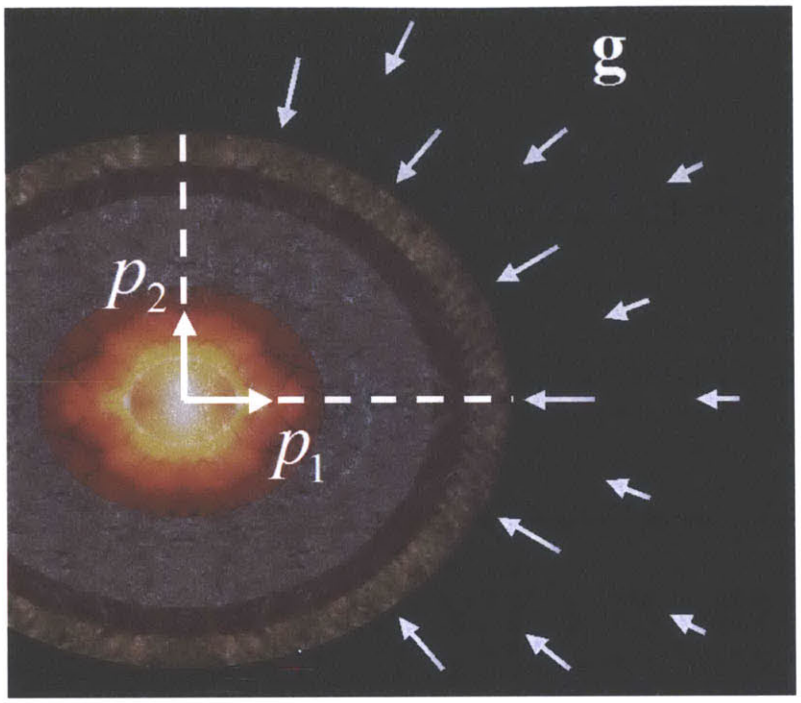


Figure 3-2: The gradient of the gravity potential, valid only outside the geoid and for $p_1 > 0$. Note that this diagram is defined in two dimensions by the planet's rotational axis and the body frame origin at an instant. By this convention, the out-of-plane component of the gradient is zero. The interior layers of the planet's crust are shown to emphasize the 2D slice through a 3D potential.

Parameter	**Meaning**	**Value**	**Units**	**Source**
ζ	inverse flattening ratio	$2.98257223563 \times 10^{2}$	$none$	[18]
r_e	equatorial radius	6.378137×10^{6}	$meters$	[18]
r_p	polar radius	$r_e\left(1-\frac{1}{\zeta}\right)$	$meters$	[18]
Ω	rotation rate	7.292115×10^{-5}	$\frac{radians}{seconds}$	[18]
GM	gravitational constant	3.986005×10^{14}	$\frac{meters^3}{seconds^2}$	[18]
C_{20}	2nd harmonic coefficient	-4.84166×10^{-4}	$none$	[18][53]
g_e	equatorial gravity	$-9.7803253359 - \Omega^2 r_e$	$\frac{meters}{seconds^2}$	[18][53]
g_p	polar gravity	-9.8321849378	$\frac{meters}{seconds^2}$	[53]

Table 3.3: 1984 World Geodetic System physical constants for navigation on Earth.

using the chain rule:

$$\nabla = \frac{\partial}{\partial \boldsymbol{x}_T} = \left[\begin{array}{cc} \frac{\partial \|\boldsymbol{p}\|}{\partial \boldsymbol{x}_T} & \frac{\partial \lambda}{\partial \boldsymbol{x}_T} \end{array} \right] \left[\begin{array}{c} \frac{\partial}{\partial \|\boldsymbol{p}\|} \\ \frac{\partial}{\partial \lambda} \end{array} \right] \tag{3.19}$$

Taking the gradient of the potential

$$\nabla v = \frac{1}{\|\boldsymbol{p}\|} \left[\begin{array}{ll} p_1 \sin\gamma & -p_2 \sin\gamma \\ 0 & 0 \\ p_1 \cos\gamma & -p_2 \cos\gamma \end{array} \right] \left[\begin{array}{c} -\frac{GM}{\|\boldsymbol{p}\|^2} \left(\left(\frac{3\sqrt{5} C_{20} r_e^2}{2\, \|\boldsymbol{p}\|^2} \right) \left(\frac{3p_2^2}{\|\boldsymbol{p}\|^2} - 1 \right) + 1 \right) \\ \frac{3\sqrt{5} C_{20} r_e^2 GM p_1 p_2}{\|\boldsymbol{p}\|^6} \end{array} \right] \tag{3.20}$$

leads to an equation for gravity by itself:

$$\boldsymbol{g} = \nabla v + \Omega^2 p_1 \left[\begin{array}{c} \sin\gamma \\ 0 \\ \cos\gamma \end{array} \right] \tag{3.21}$$

Note that $\boldsymbol{g}$ depends only on the variable $\boldsymbol{p}$ and several constants, so $\boldsymbol{g} = \boldsymbol{g}(\boldsymbol{p})$.

3.2.2 Geoid Rotation and Apparent Gravity

Inanimate objects near the surface of a geoid appear to be stationary, but they are actually hurling through space rather quickly. Given sensitive hardware, the rotation of the geoid can be measured[3], and a conflict may arise between what is seen and what is felt. This conflict can be resolved through an analysis of rigid-body dynamics.

The dynamic equations for a body in a rotating reference frame are well known. Here, we rewrite those basic equations in different frames, so that the effects of geoid rotation can be removed from the IMU output. The geoid rate can be subtracted from the gyro rates as follows:

$$\Delta\boldsymbol{\theta}[t] - \Delta t \boldsymbol{R}^{-1} \Omega \left[\begin{array}{c} \cos\gamma \\ 0 \\ -\sin\gamma \end{array} \right] \tag{3.22}$$

The effects of rotational acceleration can be lumped together with gravity. Let *apparent gravity* $\boldsymbol{g}_{app}$ be defined as the specific force felt by a body moving at a constant velocity relative to a NED frame:

[3]The procedure for finding true North using sensitive gyroscopes is called *geocompassing*.

$$\begin{aligned}
\boldsymbol{g}_{app} &= \boldsymbol{g} + \tilde{\boldsymbol{g}}_a + \tilde{\boldsymbol{g}}_b + \tilde{\boldsymbol{g}}_c \\
\tilde{\boldsymbol{g}}_a &= -\Omega^2 r_s \begin{bmatrix} \frac{1}{2}\sin 2\gamma \\ 0 \\ \cos^2\gamma \end{bmatrix} \\
\tilde{\boldsymbol{g}}_b &= \Omega^2 \begin{bmatrix} \sin^2\gamma & 0 & \frac{1}{2}\sin 2\gamma \\ 0 & 1 & 0 \\ \frac{1}{2}\sin 2\gamma & 0 & \cos^2\gamma \end{bmatrix} \boldsymbol{x}_{\mathrm{T}} \\
\tilde{\boldsymbol{g}}_c &= 2\Omega \begin{bmatrix} 0 & -\sin\gamma & 0 \\ \sin\gamma & 0 & \cos\gamma \\ 0 & -\cos\gamma & 0 \end{bmatrix} \dot{\boldsymbol{x}}_{\mathrm{T}}
\end{aligned} \tag{3.23}$$

This is gravity plus a few correction factors: $\tilde{\boldsymbol{g}}_a$ accounts for the centrepetal acceleration of the NED frame; $\tilde{\boldsymbol{g}}_b$ adds the centrepetal acceleration due to the body position in the NED frame; and $\tilde{\boldsymbol{g}}_c$ represents the Coreolis effect. In most cases, only $\tilde{\boldsymbol{g}}_a$ is necessary, but the other terms are presented for completeness. Finally, we need to define r_s, the distance from the geocentric origin to the NED origin:

$$r_s = \sqrt{\frac{r_e^4\cos^2\gamma + r_p^4\sin^2\gamma}{r_e^2\cos^2\gamma + r_p^2\sin^2\gamma}} \tag{3.24}$$

Chapter 4

Visual Measurements

Vision systems are designed to extract meaningful information from images. A successful system provides an answer to a question, which often can be expressed as a transformation from the image space to a space of higher-level information. In this work, a central question is, “What does a video stream say about a camera’s motion?” This could be answered if a transformation from the image space to the space of the camera path could be found. Unfortunately, information about the camera path is lost through the process of image formation.

People are often surprised by the simple questions that cannot be answered using current vision technology. The answer may be attainable, but only under a host of human assumptions that machines have not learned to make. Sometimes, the desired information is not there. One might ask, “Is a visible object the same as one that was previously observed?” In this case, the answer depends more on the assumptions and circumstances than the data itself—no general solution exists.

Instead of struggling to extract camera motion directly, or trying to identify landmarks, our system will assign labels to any distinct feature that enters the video stream. In other words, the goal will be to maintain local correspondences without high-level scene representation.

The left half of Figure 2-5 in Chapter 2 provides a graphical summary of the proposed vision system. Vision begins with the process of image formation, so we will consider how camera motion affects images in general. Next, we will show how to compensate for camera rotation through an information-conservative transformation. Then, we will examine the criteria for patterns of light to be distinct and trackable. A *corner strength* space will be presented as a quantitative measure of local information content. Finally, we will describe a tracking method that identifies and gives priority to points of highest information content.

4.1 Image Formation

There are many ways to model a camera in its environment, and choosing the right representation is essential to the development of this thesis. We begin with an achromatic ray-based model of light and its interaction with optical elements. An idealized achromatic ray can be represented by a unit vector radiating from a point. Likewise, an image projection can be represented by an array of unit vectors and a focal point. This implies a single focus for the optical elements associated with the camera [4]. Not all lenses fit this model, although pinholes and lenses with an infinite focal length do.

Structures and lights in the world create a web of radiant light $\ell\,(\,)$, which has been called the "plenoptic space" [38][39]. This space is parameterized by a viewpoint and a direction of observation, expressed in the visual (NED) frame[1]. If the camera frame is coincident with the body frame, then the viewpoint is $\boldsymbol{x}_T$, and the direction of observation is $\vec{\boldsymbol{z}} = \boldsymbol{R}\vec{\boldsymbol{c}}$. If the frames are not coincident, then the effect of the camera mount can be handled within the visual-inertial filter (see Section 5.2.4).

An ideal camera returns the image $y_{ij} = \ell\,(\boldsymbol{x}_T, \boldsymbol{R}\vec{\boldsymbol{c}}_{ij})$, where the subscripts correspond to a pixel location in an array. This is the generic single-focus camera model discussed earlier. To this, we add one of several specific camera projections, listed in Appendix B. All of the projections have the form $\boldsymbol{u} = \boldsymbol{u}\,(\vec{\boldsymbol{c}})$, where $\boldsymbol{u} \in [-1, 1]$ stretches to fill a rectangular array. All of the projections are invertible, so we also know $\vec{\boldsymbol{c}} = \vec{\boldsymbol{c}}\,(\boldsymbol{u})$. A forward-right-down frame is associated with the camera, such that $\vec{\boldsymbol{c}} = [1\ 0\ 0]^{\mathrm{T}}$ corresponds to the optical axis. This model is general enough to accommodate both active optics and any field-of-view from extreme telephoto to the full $360°$ imaging sphere.

At this stage, the only unknowns are the function $\ell\,(\,)$ and the camera position error $\delta\boldsymbol{x}_T$. Thousands of image measurements y_{ij} become available with each video frame, so it may be possible to learn about $\ell\,(\,)$ while estimating $\delta\boldsymbol{x}_T$. We have not modeled, nor do we plan to model, the explicit structure of the environment.

4.1.1 Calibration

This work relies on a calibrated camera, so the relationship $\boldsymbol{u}_{ij} = \boldsymbol{u}\,(\vec{\boldsymbol{c}}_{ij})$ must be known for all pixels, at all times. For a standard perspective lens, this relationship is determined by a single parameter that might be printed on the lens housing. For other radially symmetric camera models, Bakstein and Pajdla propose a simple calibration experiment, which we have recreated [5]. Their idea is to take a snapshot of a visual target and discover the intermediate relationship $r = r\,(c_1, \boldsymbol{\rho})$, which is a part of Equation B.9 in Appendix B. The target is printed so that it forms concentric rings at equiangular increments when inserted in a cylinder, as shown in Figure 4-1. The radial distance to each ring can then be measured, in *pixels*, by plotting the average image intensity with respect to radius and looking for

[1]By assuming that the environment is static and diffuse, we do not have to include time explicitly in the plenoptic expression.

local minima in the plot. Finally, the calibration parameters $\boldsymbol{\rho}$ can be found by a numerical routine, given several measurements and the known values of $c_1 = \cos(\alpha)$, where α is the angular deviation of each ring from the optical axis.

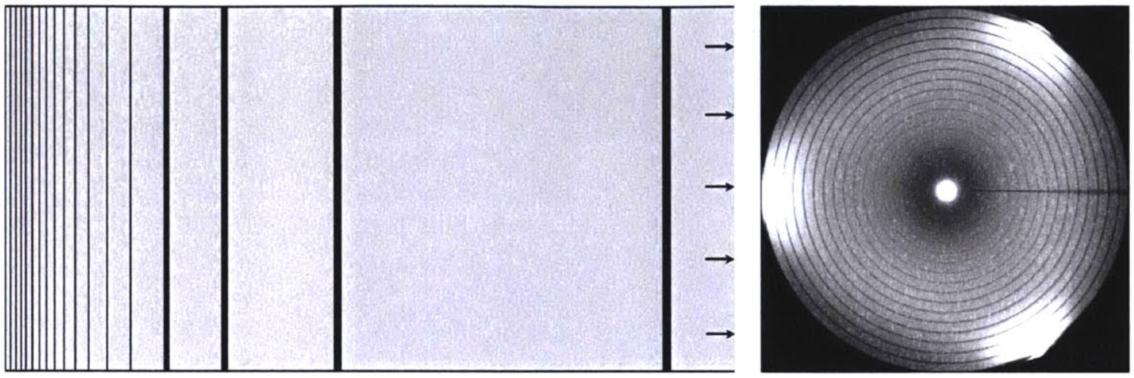


Figure 4-1: Left—Unwrapped calibration template designed as an insert for a cylinder. The lines are spaced at 5° intervals. Right—Calibration image taken with a commercially available fisheye lens mounted to a 2/3" CCTV camera (lens sold as Kowa LMVZ164, Rainbow L163VCS, or VITEK VTL-1634).

4.2 Rotation Compensation

The general motion of a camera can be separated into rotation and translation components. Typically, camera rotation causes most of the change that is perceived in the image plane, while the effect of translation is less significant. Since we want to keep track of bits of information, we would like the information to change slowly between frames. Therefore, we propose a transformation from the image space to a rotationally-fixed space.

An ideal output space would cover the imaging sphere evenly, with no variation in pixel density or directional preference[2]. The field of cartography has addressed this problem by producing numerous map projections. Each one is a flat representation of the surface of a sphere; a projection of the $SO(2)$ space onto the $\mathbb{R}^2$ space. We have used both the Gall isographic projection and the Apianus projection as output spaces, with similar results. Appendix B describes these and other useful projections.

To transform an image from one projection to another, each destination pixel is calculated as a function of the source image. The location of interest in the source space is $\boldsymbol{u} = \boldsymbol{u}(R^{-1}\vec{z}_{ij})$, where the subscripts ij correspond to the destination space. Since $\boldsymbol{u}$ is a non-integer, a bilinear weighted sum interpolates the value at the source location of interest. This transformation is almost exactly reversible, except for the slight blur introduced

[2]A fully 3D geodesic mesh would be ideal, but it is difficult to implement such a data structure.

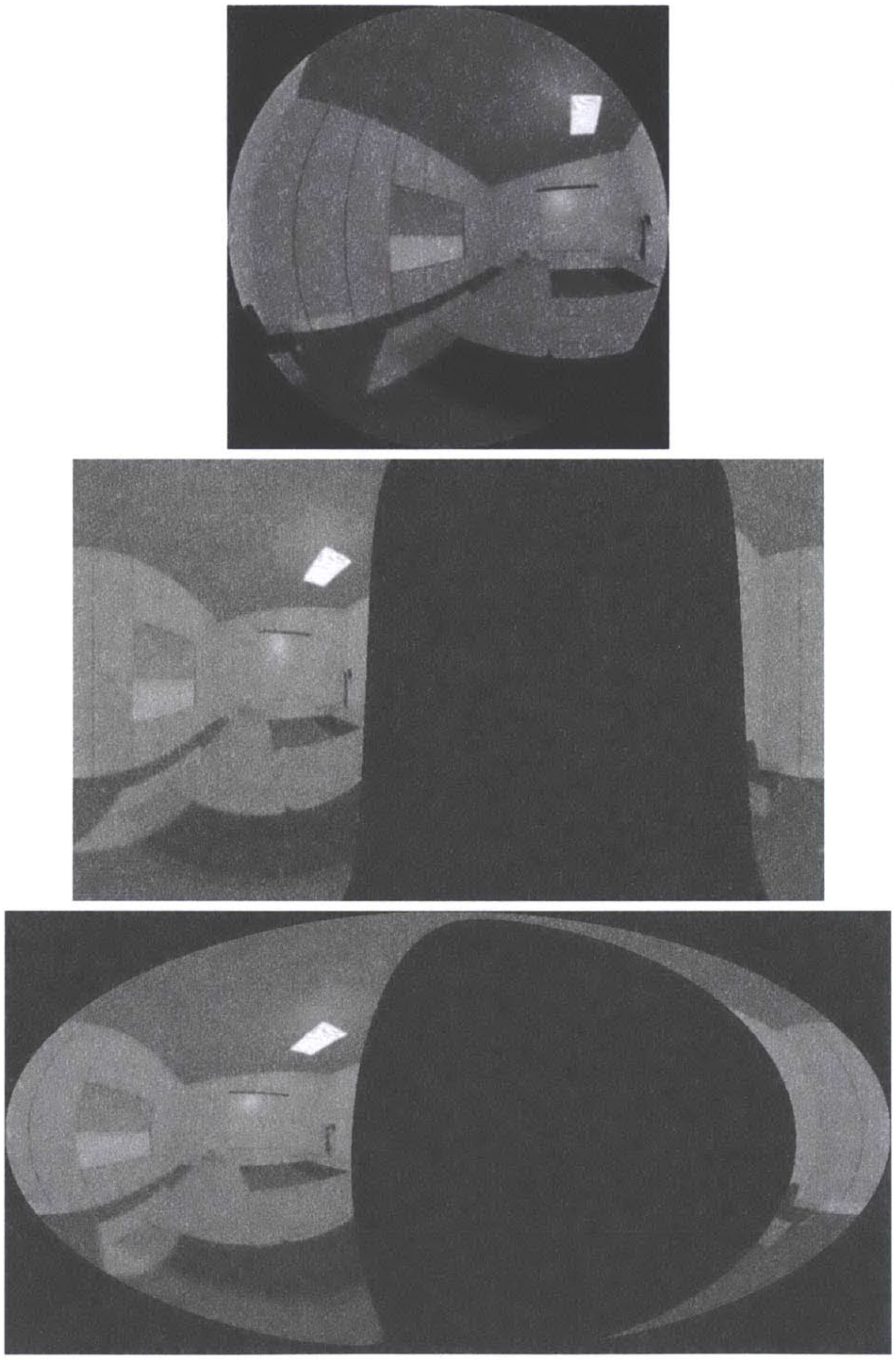


Figure 4-2: Top—Fisheye image before rotation compensation, with the camera pointing to the West. Middle—Rotation compensation with the Gall isographic reprojection. Bottom—Rotation compensation with the Apianus reprojection.

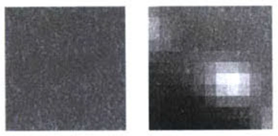
during interpolation. Therefore, the output space contains nearly all of the information in the original image.

For this application, a static map of $\vec{z}_{ij}$ can be pre-calculated for each destination pixel during algorithm initialization. To clarify, one can freely choose $\vec{z} = \vec{c}_{Gall}$, or $\vec{z} = \vec{c}_{Apianus}$, or $\vec{z} = \vec{c}_{other}$ as the destination projection.

4.3 Corner Strength

In this section, we describe a transformation from one image space to another of the same size. The input is the rotationally-fixed projection from the previous section, and the output is a local information metric, calculated for each pixel. This idea builds on corner detection theory [21][46], so we call it the *corner strength transformation.*

Some patterns of light contain more information than others. Consider the images below:



When compressed, the first image reduces by 95%, while the second reduces by only 41%. The first one is invariant to vertical and horizontal shifts, while the second varies substantially in both directions. Clearly, the second pattern is distinct, and it would be easier to track than the plain pattern.

There is a fundamental relationship between spatial variation and information content. This can be seen in the definition of the local image Hessian:

$$\boldsymbol{E} = \sum_{\boldsymbol{win}} \begin{bmatrix} \frac{\partial^2 y}{\partial i^2} & \frac{\partial y}{\partial i}\frac{\partial y}{\partial j} \\ \frac{\partial y}{\partial j}\frac{\partial y}{\partial i} & \frac{\partial^2 y}{\partial j^2} \end{bmatrix}_{\boldsymbol{win}} \mathcal{N}\left(\boldsymbol{win}; \boldsymbol{0}, \boldsymbol{I}\sigma_{smooth}\right) \tag{4.1}$$

where $\boldsymbol{win}$ represents a small region or window around each pixel of interest. Like a covariance matrix, the Hessian measures information content. Its eigenvalues represent gradient energy (ie. edges), and its eigenvectors define the major and minor gradient directions.

The ability to track a pattern depends on the eigenvalues of $\boldsymbol{E}$. If one eigenvalue is strong and the other is weak, then tracking will probably fail along the eigenvector corresponding to the weak direction. For this reason, Harris and Stephens proposed this corner strength metric that measures trackability [21]:

$$\kappa = \frac{\det\left(\boldsymbol{E}\right)}{\operatorname{tr}\left(\boldsymbol{E}\right) + \varepsilon} \tag{4.2}$$

Similarly, Shi and Tomasi reasoned that a "good feature to track" is determined by the minimum eigenvalue, which can be expressed in this concise form [46]:

$$\kappa = \text{MinEigVal}\left(\boldsymbol{E}\right) = \frac{1}{2}\left(E_{11} + E_{22} - \sqrt{\left(E_{11} - E_{22}\right)^2 + 4E_{12}^2}\right) \tag{4.3}$$

After several experiments with the two common metrics listed above, it became apparent that another factor contributes to tracking performance: balanced eigenvalues. When one eigenvalue dominates another, the shape of the pattern will be sharply defined in one direction, but elongated in the other. This problem can be avoided by using a corner strength metric that gives more weight to balanced eigenvalues:

$$\kappa = \frac{\det\left(\boldsymbol{E}\right)}{\text{tr}\left(\boldsymbol{E}\right)^n + \varepsilon} \tag{4.4}$$

We use this metric with $n = 1.5$, though any $n > 1$ is valid. Figure 4-3 shows the corner strength output, calculated by the balanced metric.

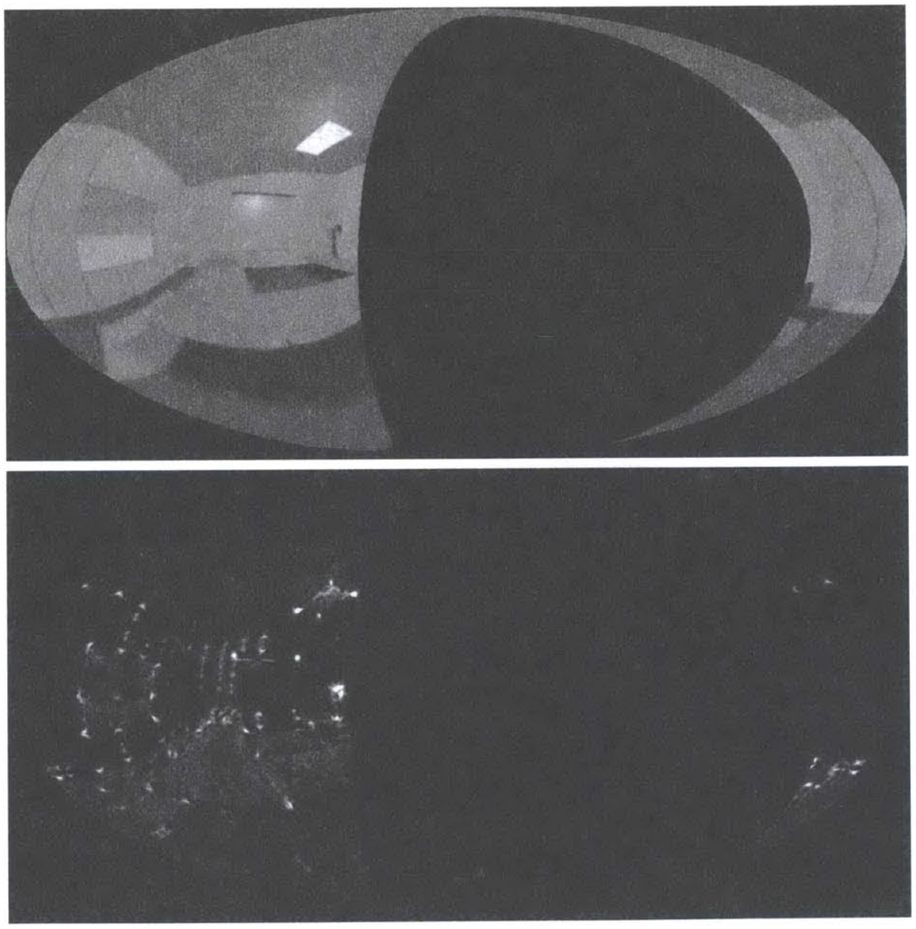


Figure 4-3: Demonstration of the corner strength transformation using the balanced eigenvalue metric.

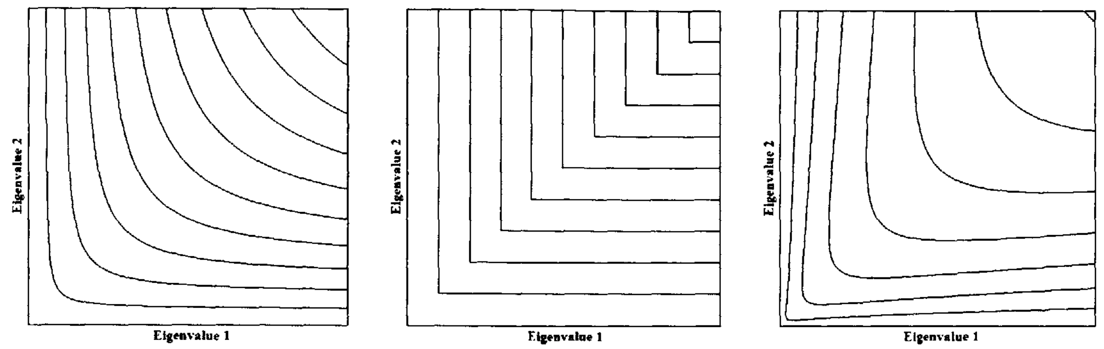


Figure 4-4: Contours of corner strength metrics, based on the eigenvalues of the edge energy Hessian: 1) Minimum eigenvalue, 2) Harris & Stephens, and 3) Balanced eigenvalue.

4.3.1 Feature Detection

When the number of features being tracked drops below a threshold, a detection algorithm can be executed to search for new ones. The corner strength metric makes feature selection trivial. First, a threshold is applied to highlight regions in the image that are trackable, and to reduce the amount of data being handled. The values above the threshold are then sorted from strongest to weakest, and feature selection begins with the strongest corner. A mask identifying valid image data is kept to constrain the region of feature selection. If the location of the candidate lies over a masked region, then it is selected, and a region around the feature is blacked out. This continues until the desired number of features are selected or until no more candidates are available.

Figure 4-5 shows a histogram from an actual corner strength image. Since the slope of the histogram is small in the region of the threshold, the algorithm is not overly sensitive to this parameter.

4.4 Feature Tracking

Images from the camera undergo rotation compensation and corner strength transformation before reaching the tracker. This final stage of the vision system produces historically linked observations of features. The observations are then passed to the inertial-visual filter in the form of ray vectors.

A feature can be tracked in either the rotation-compensated image space or the κ space. We found that κ has a slight advantage, because it tends to retain its appearance while the camera moves over large distances. When a feature is selected, a patch around its center is permanently stored to represent its appearance.

Features are tracked using the classical method of normalized cross-correlation [29]. This is essentially a template-matching method, where all of the feature positions in a small search region are evaluated, and the best match wins. By this method, features are located

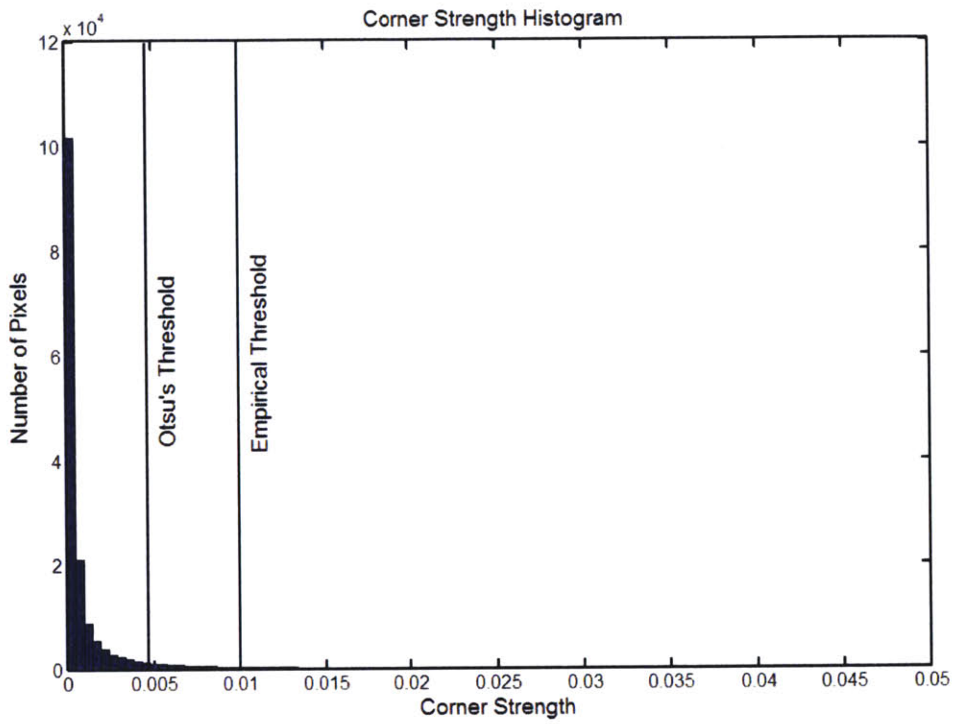


Figure 4-5: Histogram of a corner strength image and two reasonable thresholds: Otsu's binary classification threshold [40] and a threshold determined by human observation of its effect on several images.

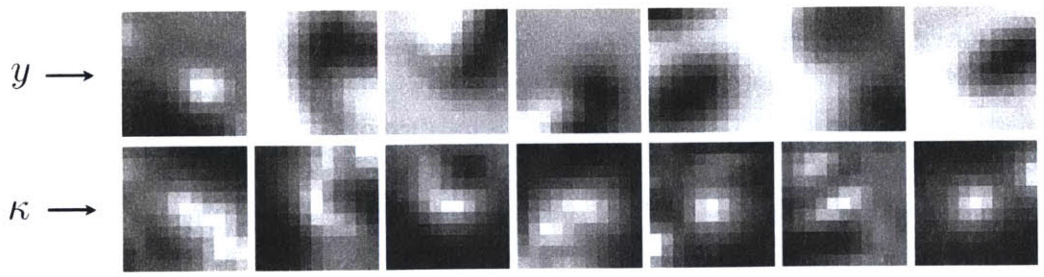


Figure 4-6: Examples of features and their corresponding corner strengths.

with single-pixel discrete accuracy. The normalization step makes the solution robust to lighting variation.

A feature can be lost in one of three ways: 1) No strong correlation exists within a local search region; 2) Another feature with a higher peak corner strength already occupies the space; or 3) It gets too close to the image boundary. If the feature survives, then an observation ray $\vec{z}[t]$ is stored along with the feature's current pixel location.

Although we recommend normalized cross-correlation, the choice of tracking algorithm is not critical. For example, the Kanade-Lucas-Tomasei (KLT) tracker uses a gradient descent solver to locate features with sub-pixel accuracy [33][51][52]. Any object tracker could substitute, as long as the result can be expressed as a set of points that are associated with ray vectors.

4.4.1 Feature Database

A feature is like a record that has a one-to-many relationship with a set of observations, so it makes sense to store features in a database structure. A record can be defined by the following fields: 1) A small, square, corner strength patch representing the first observation of the feature; 2) The current location of the feature in the image; 3) Pointers to ray vectors corresponding to historical observations of the feature; and 4) A status byte. In our implementation, the signed status byte can take on one of the following states:

(-2)	DEAD, another feature already occupies the space
(-1)	DEAD, feature could not be tracked
(0)	DEAD, no reason given
(1)	ALIVE, not included in filter, no reason given
(2)	ALIVE, not included in filter, camera has not moved much
(3)	ALIVE, not included in filter, feature has not moved much
(4)	ALIVE, not included in filter, outlier condition detected
(5)	ALIVE, included in filter

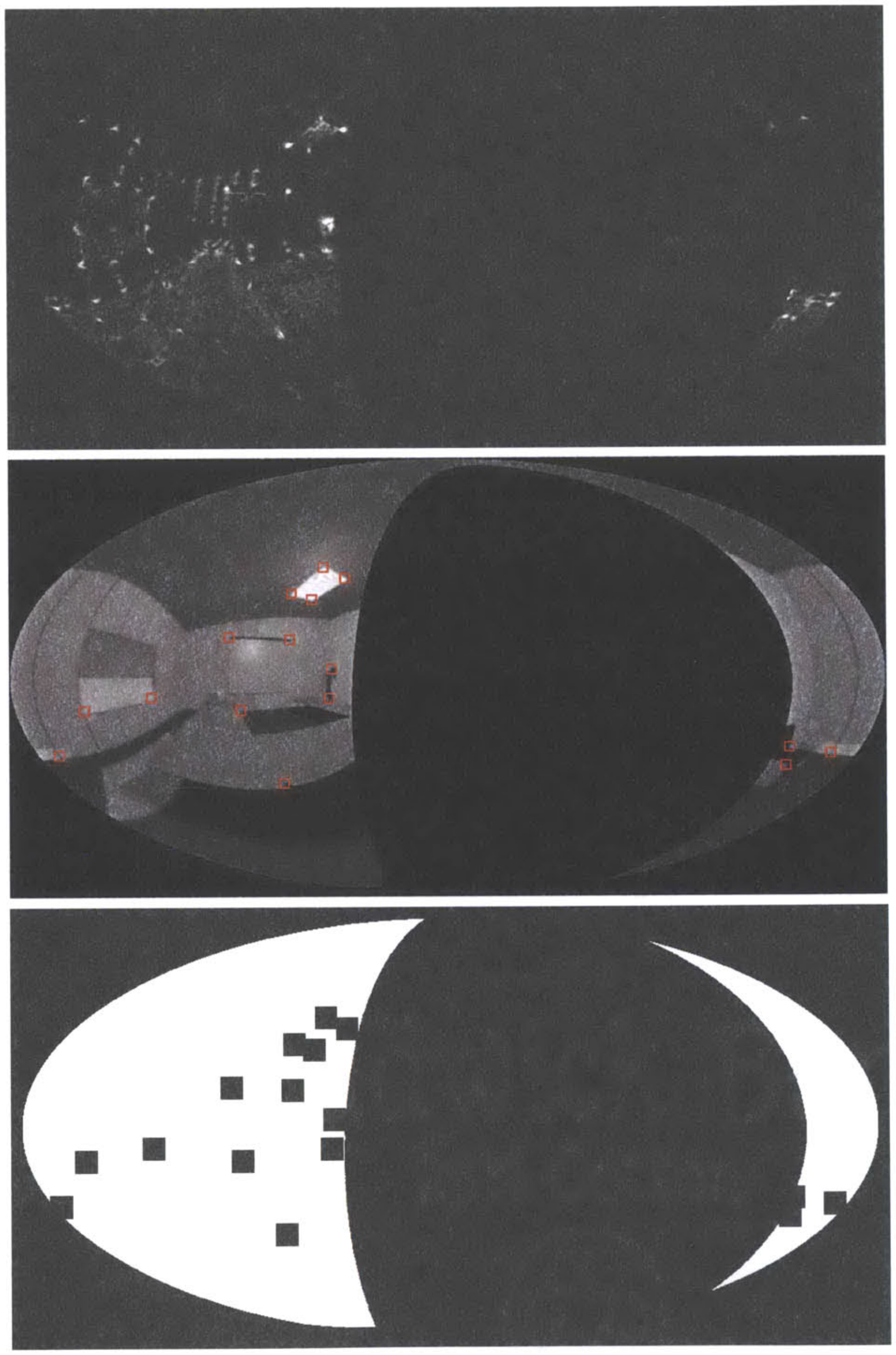


Figure 4-7: Top—Input to the tracking algorithm. Middle—Features that meet the corner threshold criteria. Bottom—Mask used during tracking to prevent feature overlap.

Chapter 5

Epipolar Constraint Filter

Consider a feature tracked by a single camera over multiple frames, as shown in Figure 5-1. The camera executes arbitrary 6-DOF motion $\boldsymbol{x}[t]$ while recording the ray observations $\vec{\boldsymbol{z}}[t]$. The spheres indicate that each measurement is rotation-compensated, so the orientation of the camera is irrelevant as long as the feature appears somewhere in the field-of-view. Referring to the figure again, notice that any two non-parallel rays define a planar subspace, and their cross product is orthogonal to the subspace. This well known relationship is often called the *epipolar constraint* [3][31][34][35], though it appears in various forms and under different names. In our notation, a basic form of the constraint is given by

$$(\vec{\boldsymbol{z}}[t_a] \times \vec{\boldsymbol{z}}[t_b]) \circ \Delta\boldsymbol{x}_T = 0 \tag{5.1}$$

where Δ represents a change of state between the time of feature acquisition t_a and the current time t_b. The constraint tells us that the camera did not move in the direction perpendicular to the plane.

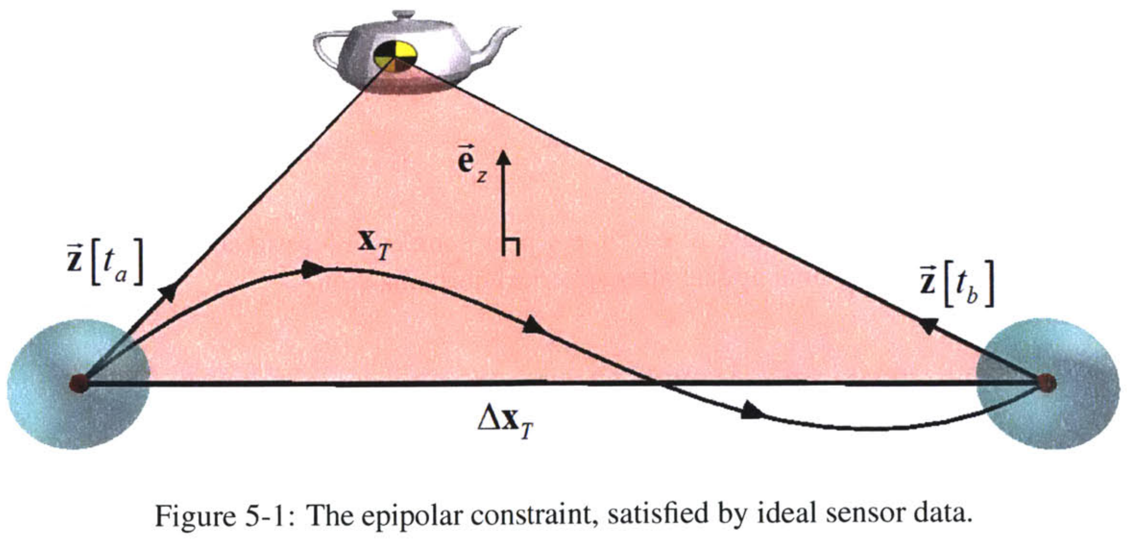


Figure 5-1: The epipolar constraint, satisfied by ideal sensor data.

5.1 Residual

Suppose we want to make a filter that incorporates the epipolar constraint. When noise perturbs the sensors, the left-hand-side of Equation 5.1 may become nonzero. The extra value that would have to be added to the right-hand-side to maintain equality is called a *residual*. We propose an alternative version of the constraint that yields a more meaningful residual. It is a projection of the out-of-plane body motion:

$$\tilde{\boldsymbol{x}}_T \equiv \left(I - \vec{\boldsymbol{e}}_x \vec{\boldsymbol{e}}_x^{\mathrm{T}}\right) \vec{\boldsymbol{e}}_z \vec{\boldsymbol{e}}_z^{\mathrm{T}} \Delta\bar{\boldsymbol{x}}_T \overset{?}{=} 0 \tag{5.2}$$

with

$$\vec{\boldsymbol{e}}_x = \frac{\Delta\bar{\boldsymbol{x}}_T}{\|\Delta\bar{\boldsymbol{x}}_T\|} \qquad \vec{\boldsymbol{e}}_z = \frac{\vec{\boldsymbol{z}}\,[t_a] \times \vec{\boldsymbol{z}}\,[t_b]}{\left\|\vec{\boldsymbol{z}}\,[t_a] \times \vec{\boldsymbol{z}}\,[t_b]\right\|}$$

The residual $\tilde{\boldsymbol{x}}_T$ of Equation 5.2 has several desirable properties: 1) It vanishes when the epipolar constraint is satisfied; 2) It depends only on directly measurable quantities, so scene depth does not appear explicitly; 3) It is a vector; and 4) Its direction is always perpendicular to the observation baseline. Figure 5-2 provides a graphical interpretation. Although this residual was chosen carefully, arguably better forms could exist.

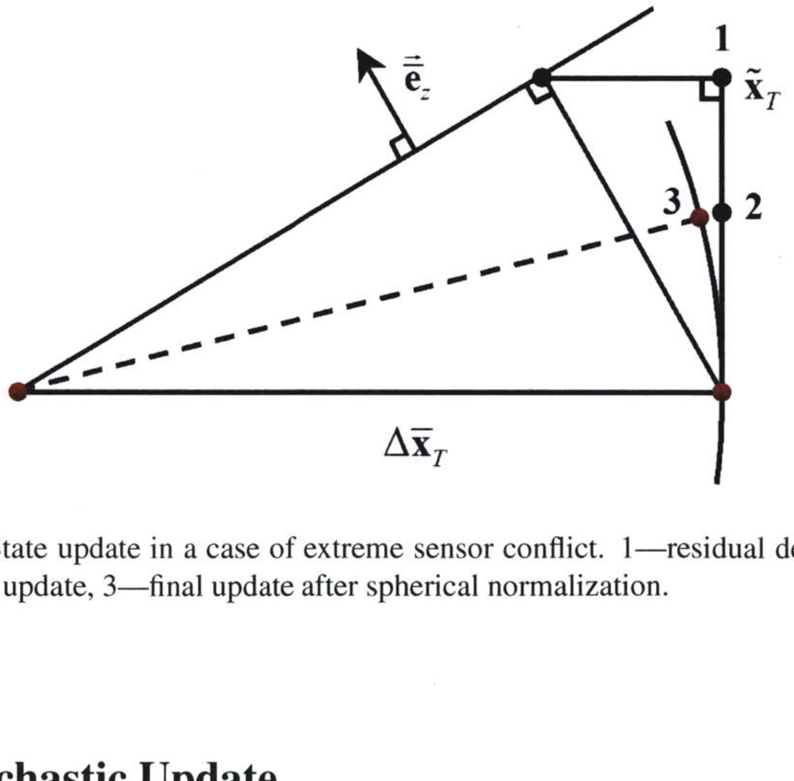


Figure 5-2: State update in a case of extreme sensor conflict. 1—residual definition, 2—Kalman state update, 3—final update after spherical normalization.

5.2 Stochastic Update

Nearly all of the tools are now in place to define a stochastic visual update. When error appears in the residual, we can rationally distribute its value between the image data and the inertial data. The IMU uncertainty is properly represented by a dynamically evolving Gaussian distribution, as in Equation 3.10. If we trust the residual direction and loosely assume Gaussian measurement noise associated with its magnitude, then the Bayes Least Squares (BLS) posterior estimates are given by

$$\boldsymbol{h} = \begin{bmatrix} \frac{\tilde{\boldsymbol{x}}_T^{\mathrm{T}}}{\|\tilde{\boldsymbol{x}}_T\|} & 0 \end{bmatrix} \qquad \text{(measurement gain)} \tag{5.3}$$

$$\boldsymbol{k} = \boldsymbol{\Lambda}^{-}\boldsymbol{h}^{\mathrm{T}}\left(\boldsymbol{h}\boldsymbol{\Lambda}^{-}\boldsymbol{h}^{\mathrm{T}} + \tilde{\sigma}^2\right)^{-1} \qquad \text{(Kalman gain)} \tag{5.4}$$

$$\boldsymbol{\psi}^{+} = \boldsymbol{\psi}^{-} + \boldsymbol{k}\,\|\tilde{\boldsymbol{x}}_T\| \qquad \text{(state update)} \tag{5.5}$$

$$\boldsymbol{\Lambda}^{+} = (\boldsymbol{I} - \boldsymbol{k}\boldsymbol{h})\,\boldsymbol{\Lambda}^{-} \qquad \text{(reduced uncertainty)} \tag{5.6}$$

and, as a matter of notation

$$\boldsymbol{\psi}^{+} = \hat{\boldsymbol{\psi}} = \begin{bmatrix} \delta\hat{\boldsymbol{x}}_T \\ \delta\hat{\dot{\boldsymbol{x}}}_T \\ \delta\hat{\boldsymbol{b}}_{TurnOn} \\ \delta\hat{\boldsymbol{b}}_{InRun} \\ \delta\hat{\boldsymbol{s}}_{TurnOn} \\ \delta\hat{\boldsymbol{s}}_{InRun} \end{bmatrix} \tag{5.7}$$

Admittedly, we do not know much about the measurement variance $\tilde{\sigma}^2$. It depends on the imaging hardware, the tracking algorithm, the body path, and the scene. From our simulations, we were able to determine strong dependence on the body path and the feature cross product.

$$\tilde{\sigma}^2 \approx \frac{\Delta\bar{\boldsymbol{x}}_T^2 \sigma_{angular}^2}{\left\|\vec{\boldsymbol{z}}\left[t_a\right] \times \vec{\boldsymbol{z}}\left[t_b\right]\right\|^2} + \sigma_{tol}^2 \tag{5.8}$$

Here, $\sigma_{angular}$ is the expected long-term deviation of the tracking algorithm, which we assume to be about six $pixels$ of angular separation, or $1.5°$. We also set the noise floor at $\sigma_{tol} = 0.01\ meters$.

5.2.1 Relative Covariance

Suppose the sensor platform has been traveling for some time in a pitch-black room. Suddenly, the lights are turned on, and newly acquired feature rays are stored. The body position estimate at that time would act as a reference for future visual constraints, and the state covariance would also be a reference. At any time, consider what happens to the covariance, given the exact body position:

$$\boldsymbol{\Lambda} \mid \boldsymbol{x}_T = \begin{bmatrix} 0 & 0 \\ 0 & \boldsymbol{\Lambda}_{4:18,4:18} - \boldsymbol{\Lambda}_{4:18,1:3}\boldsymbol{\Lambda}_{1:3,1:3}^{-1}\boldsymbol{\Lambda}_{1:3,4:18} \end{bmatrix} \tag{5.9}$$

Also, consider the algebraic change in covariance:

$$\Delta\boldsymbol{\Lambda} = \boldsymbol{\Lambda}\left[t_b\right] - \boldsymbol{\Lambda}\left[t_a\right] \tag{5.10}$$

Assuming that prior knowledge of the state $\boldsymbol{x}_T\left[t_a\right]$ does not affect the change in covariance, we make the following approximation:

$$\Delta\boldsymbol{\Lambda} \approx \boldsymbol{\Lambda}\left[t_b\right] \mid \boldsymbol{x}_T\left[t_a\right] - \boldsymbol{\Lambda}\left[t_a\right] \mid \boldsymbol{x}_T\left[t_a\right] \tag{5.11}$$

By doing this, we lose some fidelity of the noise model. However, we avoid the need to maintain complicated relationships between multiple features.

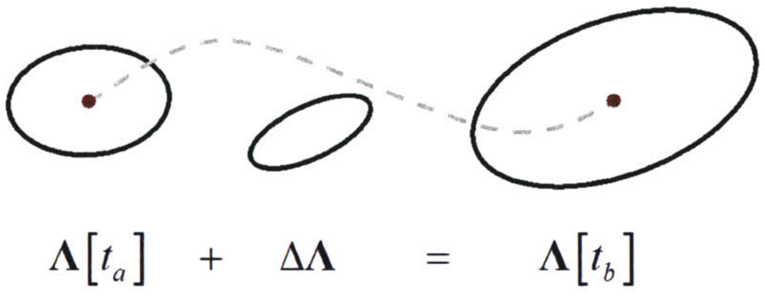

$\boldsymbol{\Lambda}[t_a]$ + $\Delta\boldsymbol{\Lambda}$ = $\boldsymbol{\Lambda}[t_b]$

Figure 5-3: The algebraic relative covariance.

Rearranging Equations 5.10 and 5.11 to put the unknown quantity on the left-hand-side yields

$$\begin{aligned} \boldsymbol{\Lambda}\left[t_b\right] \mid \boldsymbol{x}_T\left[t_a\right] &= \boldsymbol{\Lambda}\left[t_b\right] - \boldsymbol{\Lambda}\left[t_a\right] + \boldsymbol{\Lambda}\left[t_a\right] \mid \boldsymbol{x}_T\left[t_a\right] \\ &= \boldsymbol{\Lambda}\left[t_b\right] - \boldsymbol{\Lambda}_{ref}\left[t_a\right] \end{aligned} \tag{5.12}$$

with

$$\boldsymbol{\Lambda}_{ref} = \begin{bmatrix} \boldsymbol{\Lambda}_{1:3,1:3} & \boldsymbol{\Lambda}_{1:3,4:18} \\ \boldsymbol{\Lambda}_{4:18,1:3} & \boldsymbol{\Lambda}_{4:18,1:3}\boldsymbol{\Lambda}_{1:3,1:3}^{-1}\boldsymbol{\Lambda}_{1:3,4:18} \end{bmatrix} \tag{5.13}$$

Since the expression $\boldsymbol{\Lambda}\left[t_b\right] - \boldsymbol{\Lambda}_{ref}\left[t_a\right]$ includes an approximation, some of its eigenvalues could drop below zero. A covariance matrix by definition must be symmetric and positive semi-definite. Therefore, the approximate relative covariance for the BLS update is defined as follows

$$\boldsymbol{\Lambda}^- \equiv \text{EnforceSPD}\left(\boldsymbol{\Lambda}\left[t_b\right] - \boldsymbol{\Lambda}_{ref}\left[t_a\right]\right) \tag{5.14}$$

where EnforceSPD( ) brings all eigenvalues of its argument up to ϵ and enforces symmetry. The idea is to ignore states that are unobservable, given the temporal baseline of the measurement. After the BLS update, the reference uncertainty is then reinstated.

$$\boldsymbol{\Lambda}\left[t_b\right] = \boldsymbol{\Lambda}^+ + \boldsymbol{\Lambda}_{ref}\left[t_a\right] \tag{5.15}$$

5.2.2 Multiple Features

So far, the discussion has been limited to a single feature, but multiple features can be handled naturally. Each feature has its own reference time t_a, and all features share the current time t_b. We apply individual stochastic updates in sequential order, beginning with the oldest visible feature and iterating through all currently visible features. The order does matter, because of the slight nonlinearity of the updates. Each feature contributes one planar constraint, leading to state observability in multiple dimensions.

Since the inertial process runs continually, the filter is able to handle as few as zero visual observations. There are no special cases or minimum requirements on the number of features, so a temporary loss of vision will not crash the algorithm. However, for obvious reasons, the best performance is achieved with many features that are tracked over a long period of time.

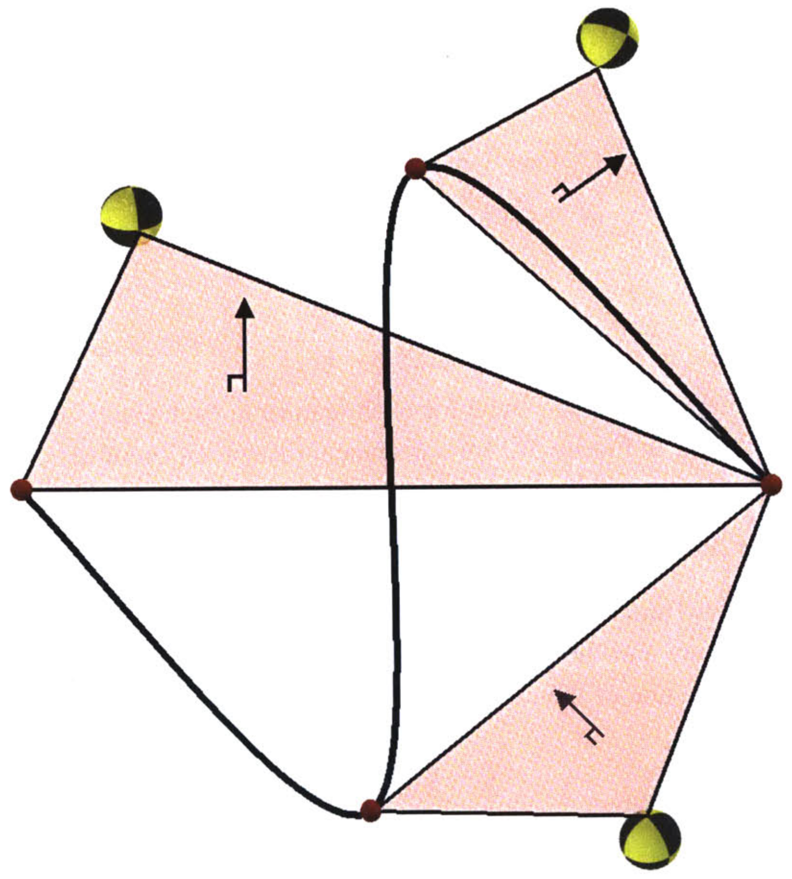


Figure 5-4: Multiple epipolar constraints can result in a fully constrained body position.

5.2.3 State Transfers

There are two kinds of state updates. The first update is the natural propagation of the state dynamics from Equation 3.10. After each IMU event, but before visual updates are applied, the values of the first six elements of $\psi[t]$ are *transferred*: The position error estimate goes into the position estimate, and the velocity error estimate is fed back into the IMU integration process according to these equations:

$$\bar{\boldsymbol{x}}_T - \delta\hat{\boldsymbol{x}}_T \Rightarrow \bar{\boldsymbol{x}}_T \;,\; \mathbf{0} \Rightarrow \delta\hat{\boldsymbol{x}}_T \tag{5.16}$$

$$\dot{\bar{\boldsymbol{x}}}_T - \delta\dot{\hat{\boldsymbol{x}}}_T \Rightarrow \dot{\bar{\boldsymbol{x}}}_T \;,\; \mathbf{0} \Rightarrow \delta\dot{\hat{\boldsymbol{x}}}_T \tag{5.17}$$

The second update comes from the vision system through the epipolar residual and Equation 5.5. Due to the nature of image projection, the residual contains no information about the magnitude of the body translation, but it does affect the direction of translation. Its direction is perpendicular to the translation baseline. Therefore, the visual updates are actually incremental rotations of point $\bar{\boldsymbol{x}}_T[t_b]$ about point $\bar{\boldsymbol{x}}_T[t_a]$. To be sure that the magnitude of translation is not affected, the state transfer must be constrained to the surface of a sphere of radius $\|\Delta\bar{\boldsymbol{x}}_T\|$, as follows:

$$\bar{\boldsymbol{x}}_T[t_a] + \vec{\boldsymbol{e}}_{upd}\|\Delta\bar{\boldsymbol{x}}_T\| \Rightarrow \bar{\boldsymbol{x}}_T[t_b] \;,\; \mathbf{0} \Rightarrow \delta\hat{\boldsymbol{x}}_T \tag{5.18}$$

$$\dot{\bar{\boldsymbol{x}}}_T - \left(\boldsymbol{I} - \vec{\boldsymbol{e}}_{upd}\vec{\boldsymbol{e}}_{upd}^{\mathrm{T}}\right)\delta\dot{\hat{\boldsymbol{x}}}_T \Rightarrow \dot{\bar{\boldsymbol{x}}}_T \;,\; \mathbf{0} \Rightarrow \delta\dot{\hat{\boldsymbol{x}}}_T \tag{5.19}$$

with

$$\vec{\boldsymbol{e}}_{upd} = \frac{\Delta\bar{\boldsymbol{x}}_T - \delta\hat{\boldsymbol{x}}_T}{\|\Delta\bar{\boldsymbol{x}}_T - \delta\hat{\boldsymbol{x}}_T\|}$$

This visual update happens often—once for each feature in each image from any number of cameras. Again, refer to Figure 5-2 for a graphical interpretation of these equations.

5.2.4 Camera Mount

Sometimes, the camera frame is not coincident with the IMU frame, due to a fixed or active mount between the body and the camera. In general, there could be a set of active joints $\boldsymbol{q}(t)$ that determine the state of the camera relative to the body.

$$\boldsymbol{\zeta} = \boldsymbol{\zeta}(\boldsymbol{q}(t)) \tag{5.20}$$

The camera state can be broken into an offset $\boldsymbol{\zeta}_T$ and an orientation $\boldsymbol{R}_{cam} = \boldsymbol{R}_{cam}(\boldsymbol{\zeta}_R)$, and each part has a separate role in the system. The role of the camera orientation is straightforward. Rotation compensation simply takes on an extra term, so $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{R}^{-1}\vec{\boldsymbol{z}}_{ij})$ becomes $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{R}_{cam}^{-1}\boldsymbol{R}^{-1}\vec{\boldsymbol{z}}_{ij})$. The camera offset has a more complicated effect, because it is embedded in the filter equations. From Figure 5-1, notice that the epipolar plane constrains the motion of the camera, which can differ from the motion of the body. This has an effect on the definition of the residual, so Equation 5.2 becomes:

$$\tilde{\boldsymbol{x}}_T \equiv \left(\boldsymbol{I} - \vec{\boldsymbol{e}}_x \vec{\boldsymbol{e}}_x^{\mathrm{T}}\right) \vec{\boldsymbol{e}}_z \vec{\boldsymbol{e}}_z^{\mathrm{T}} \Delta \left(\boldsymbol{x}_T + \boldsymbol{R}\boldsymbol{\zeta}_T\right) \overset{?}{=} \boldsymbol{0} \tag{5.21}$$

with

$$\vec{\boldsymbol{e}}_x = \frac{\Delta\left(\bar{\boldsymbol{x}}_T + \boldsymbol{R}\boldsymbol{\zeta}_T\right)}{\|\Delta\left(\bar{\boldsymbol{x}}_T + \boldsymbol{R}\boldsymbol{\zeta}_T\right)\|} \qquad \vec{\boldsymbol{e}}_z = \frac{\vec{\boldsymbol{z}}[t_a] \times \vec{\boldsymbol{z}}[t_b]}{\|\vec{\boldsymbol{z}}[t_a] \times \vec{\boldsymbol{z}}[t_b]\|}$$

Similarly, the state transfer equations must incorporate the camera offset. The updates, analogous to Equations 5.18 & 5.19, become:

$$\bar{\boldsymbol{x}}_T[t_a] - \Delta\left(\boldsymbol{R}\boldsymbol{\zeta}_T\right) + \vec{\boldsymbol{e}}_{upd}\,\|\Delta\left(\bar{\boldsymbol{x}}_T + \boldsymbol{R}\boldsymbol{\zeta}_T\right)\| \Rightarrow \bar{\boldsymbol{x}}_T[t_b]\ ,\ \boldsymbol{0} \Rightarrow \delta\hat{\boldsymbol{x}}_T \tag{5.22}$$

$$\dot{\boldsymbol{x}}_T - \left(\boldsymbol{I} - \vec{\boldsymbol{e}}_{upd}\vec{\boldsymbol{e}}_{upd}^{\mathrm{T}}\right)\delta\dot{\hat{\boldsymbol{x}}}_T \Rightarrow \dot{\boldsymbol{x}}_T\ ,\ \boldsymbol{0} \Rightarrow \delta\dot{\hat{\boldsymbol{x}}}_T \tag{5.23}$$

with

$$\vec{\boldsymbol{e}}_{upd} = \frac{\Delta\left(\bar{\boldsymbol{x}}_T + \boldsymbol{R}\boldsymbol{\zeta}_T\right) - \delta\hat{\boldsymbol{x}}_T}{\|\Delta\left(\bar{\boldsymbol{x}}_T + \boldsymbol{R}\boldsymbol{\zeta}_T\right) - \delta\hat{\boldsymbol{x}}_T\|}$$

5.3 Outlier Protection

Even the best tracking algorithms occasionally mistrack, and when they do, the types of errors observed can be difficult to characterize. Although many factors lead to measurement error, we can identify some measurements as being highly unlikely. In a statistical framework, events that almost never happen are called outliers, and it is common practice to ignore them when they are identified.

Here, we identify three criteria that must be true for a reasonable measurement. The first rule says that the magnitude of the angle-of-conflict between the IMU data and image data should not exceed $\frac{\pi}{4}$ *radians*:

$$\left|\vec{\boldsymbol{e}}_x \circ \vec{\boldsymbol{e}}_z\right| < \cos\left(\frac{\pi}{4}\right) \tag{5.24}$$

The second rule constrains the angle-of-conflict to a half-space, such that the camera and feature observation move in opposing directions:

$$\vec{\boldsymbol{e}}_x \circ \left(\vec{\boldsymbol{z}}[t_b] - \vec{\boldsymbol{z}}[t_a]\right) < 0 \tag{5.25}$$

Refer to Figures 5-1 & 5-2 for a graphical interpretation of these inequalities. Finally, the residual magnitude should have a reasonable probability of being observed, where anything beyond 2.5 standard deviations could be considered highly unlikely. In practice, one can test the residual against the joint uncertainty

$$\|\tilde{\boldsymbol{x}}_T\| < 2.5\sqrt{\boldsymbol{h}\boldsymbol{\Lambda}^{-}\boldsymbol{h}^{\mathrm{T}} + \bar{\sigma}^2} \tag{5.26}$$

or against the state uncertainty alone (more conservative)

$$\|\tilde{\boldsymbol{x}}_T\| < 2.5\sqrt{\boldsymbol{h}\boldsymbol{\Lambda}^{-}\boldsymbol{h}^{\mathrm{T}}} \tag{5.27}$$

If any of these criteria are not met, then the current measurement is excluded from the filter.

5.4 Numerical Caveats

There are four divide-by-zero traps to avoid in the proposed system. The first two appear when the triangle in Figure 5-1 becomes degenerate. If either of the denominators in Equation 5.2 become small, we exclude the corresponding feature from the update phase. The third trap appears when the magnitude of the residual drops below ϵ. In this case, the filter is doing well. The measurement should be included, but calculation of its direction may be numerically unstable. To work around this, we replace Equation 5.3 with

$$\boldsymbol{h} = \begin{bmatrix} \vec{e}_z^{\,\mathrm{T}} & 0 \end{bmatrix} \tag{5.28}$$

This works because the constraint plane normal and the residual point in approximately the same direction when the residual becomes small. The fourth trap appears when the inversion in Equation 5.13 becomes ill-conditioned. We prevent this by adding σ_{tol}^2 to the diagonal elements of $\boldsymbol{\Lambda}_{1:3,1:3}$ during inversion.

Chapter 6

Experiments

In this chapter, we present hardware, data, and results that demonstrate the capability of the proposed system. Our aim is not to test any particular aspect of the filter, but to encourage future interest in the field of inertial-visual sensing by showing the potential of this combination. To that end, we have created both custom hardware and an extensive simulation system that are equally valid platforms for future experimentation.

The synthetic visual scenes are rendered with POV-Ray, a free ray-tracing system, and the corresponding inertial data is produced by Monte-Carlo simulation. By correctly modeling rough textures, transparency, reflected light, and pixel sampling, a ray-traced image can closely resemble reality. In the world of simulation, everything is known, from the exact body path to the camera projection and the structure of the environment. However, it is simple enough to separate the images from those variables that would be unknown under real circumstances.

Although synthetic data can be very accurate, our real data differed from the simulations in several ways. In the real data: 1) Pixel saturation was more common; 2) There were about three times as many visual features per image; 3) The lens deviated from the calibrated model by as much as ± 0.5 degrees; 4) The initial conditions were estimated from the data itself, while the sensor was at rest; and 5) The time synchronization of the images relative to the IMU could only be determined up to about ± 0.03 seconds.

The same code was applied to all data sets. Besides initial conditions, the only parameters that had to be adjusted were the camera calibration and $\sigma_{angular}$. To the best of our knowledge, the results presented here are typical.

6.1 Hardware

The hardware used to collect real data consists of an IMU, a camera, a laptop, custom data acquisition devices, and rigid mounting frames. Two complete packages were assembled, with different mounts and data acquisition devices, although only one of them was used. Figure 6-1 shows both versions.

The camera offset was $\boldsymbol{\zeta}_T = \begin{bmatrix} 0 & 0 & -0.145 \end{bmatrix}^{\mathrm{T}}$, and the relative camera orientation was $\boldsymbol{\zeta}_R = \begin{bmatrix} 0.5 & -0.5 & -0.5 & 0.5 \end{bmatrix}^{\mathrm{T}}$, the quaternion form of $\boldsymbol{R}_{cam}$. The lens focal length was fixed at $1.6\ mm$, and the aperture was set to its maximum. A Windows DirectShow application called AmCap was used to set the camera parameters and store video data. Images were taken at $640 \times 480\ pixels$ resolution (subsampled by 2), with an exposure time of $8.3\ ms$ and a gain of 4.3. The requested frame rate was $12.27\ Hz$, but the actual rate was measured at $12.60\ Hz$. As a check of the lens and camera together, we were able to verify that the image was circular.

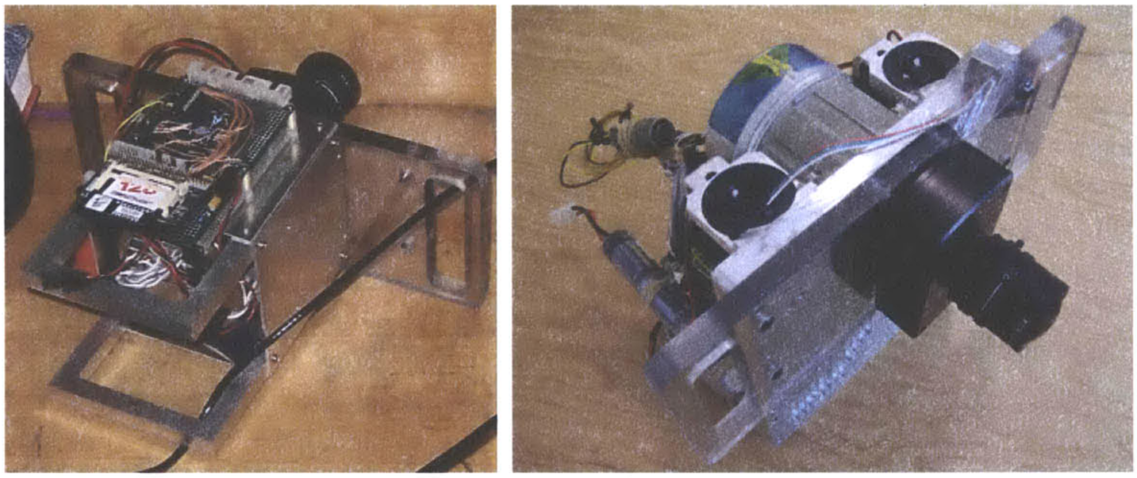


Figure 6-1: Experimental hardware developed at Draper Laboratory for inertial-visual sensing. Left—New model. Right—Old model (not used).

Specifications:

- **Camera**, Lumenera LU125M, monochrome, $2/3$" CCD sensor, maximum resolution $1280 \times 1024\ pixels$
- **Lens**, VITEK VTL-1634, focal length 1.6–$3.4\ mm$, maximum aperture F1.4, field-of-view $180°$
- **IMU**, Litton LN200 (modeled by LN2 in Table 3.1), fiber optic gyroscopes, silicon accelerometers

6.2 Data Sets

Our results are based on four data sets. Table 6.1 summarizes the basic parameters that describe each data set, and Figure 6-2 shows some examples of image data. The remaining figures in this chapter are plots of results.

In the simulated scene, Factory7, the camera follows an open-loop path through an industrial building. The motion includes some variation in speed and rotation, as if the camera were carried by a person treading carefully. Ground truth for this scene comes in the form of a parametric camera path produced by the simulator.

The gantry scenes were produced in a large dry water tank with an overhead robot. This setup bears resemblance to a related experiment by Roumeliotis et. al. [45]. The images show random objects sprinkled on the bottom of the tank, while the camera follows a closed-loop path within a small region. In these scenes, only translational motion was intended, although the body undergoes some rotation. The robot provides ground truth for the body path to within a few millimeters.

Title	**Realism**	**Camera**	**Image Size**	**Duration**	**Frame Rate**	**IMU Rate**
			pixels	*seconds*	*Hz*	*Hz*
Factory7	simulated	B.4	480×480	60	10	400
GantryB	real	B.5	460×460	52	12.6	400
GantryC	real	B.5	460×460	36	12.6	400
GantryF	real	B.5	460×460	44	12.6	400

Table 6.1: Parameters for each experiment. Camera models are described in Appendix B.

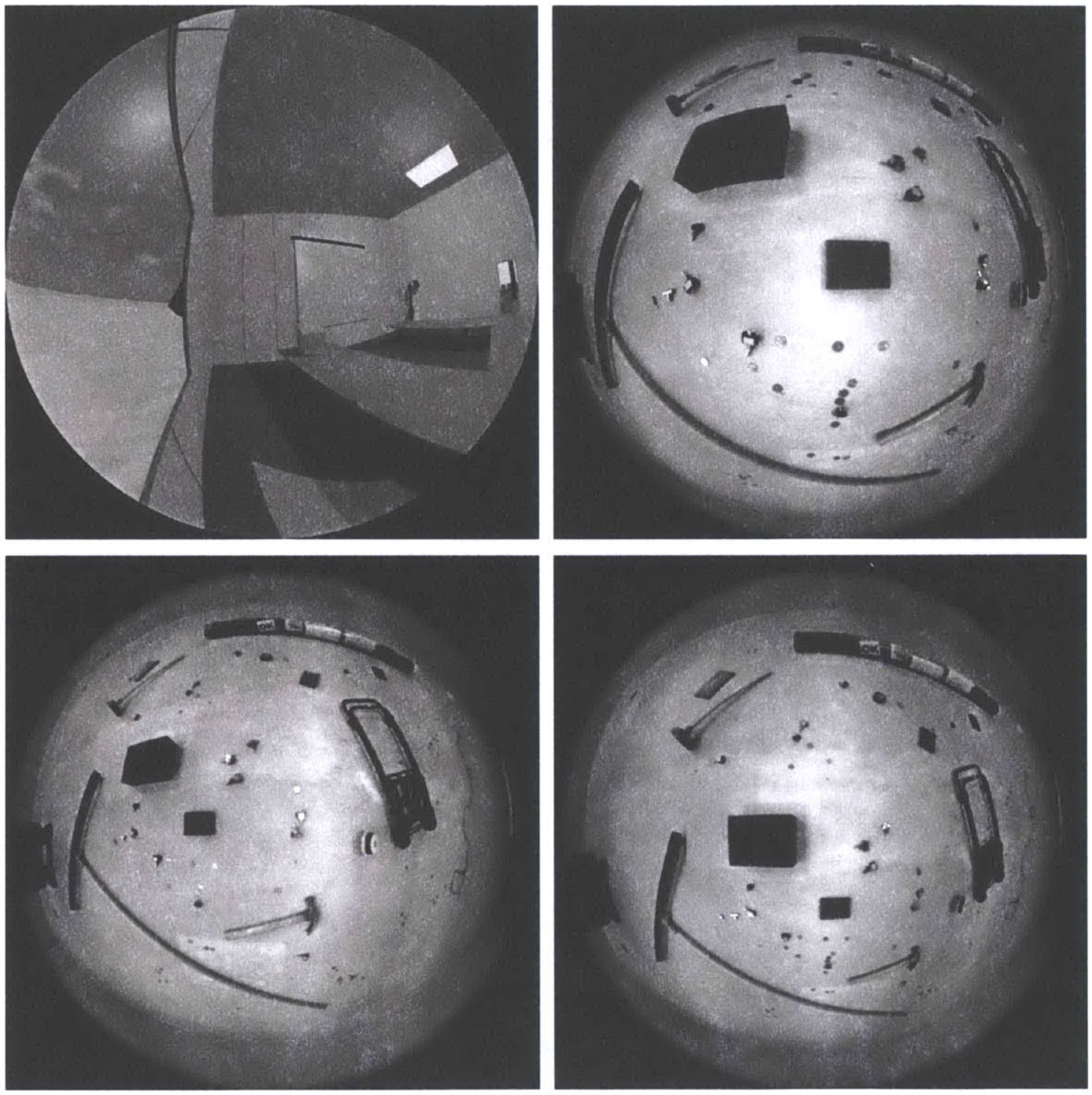


Figure 6-2: Representative images from each scene: Factory7, GantryB, GantryC, and GantryF.

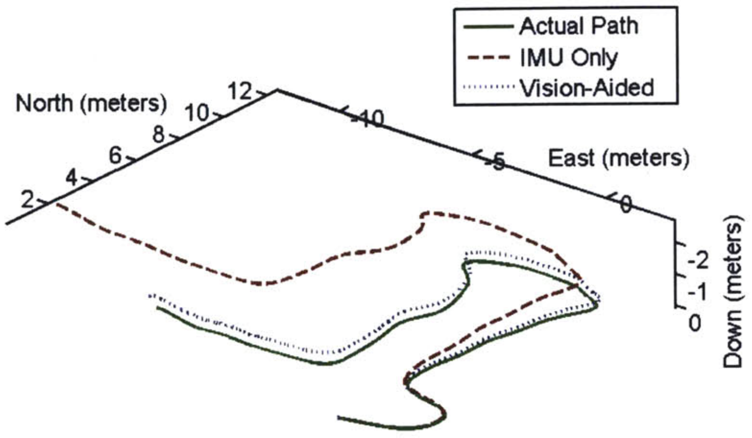


Figure 6-3: Factory7—full motion plot.

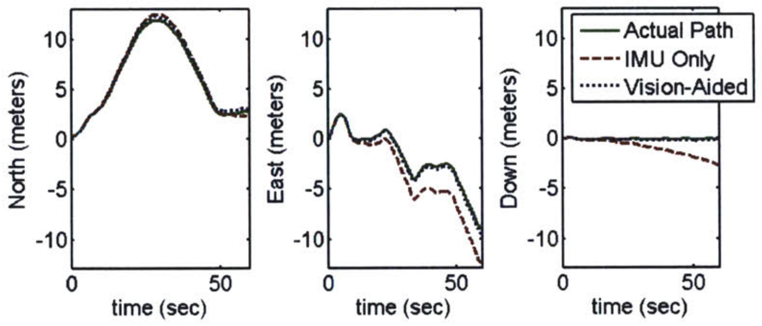


Figure 6-4: Factory7—plot of each dimension.

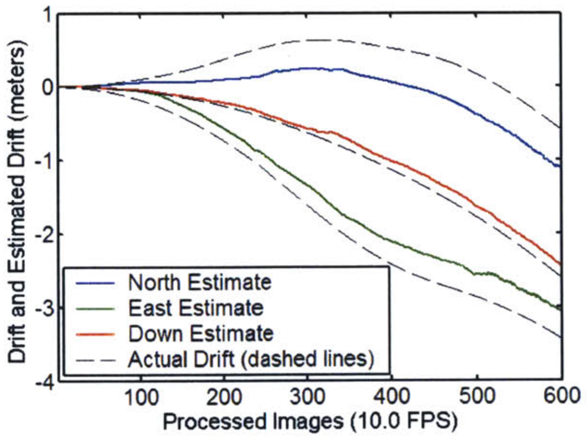


Figure 6-5: Factory7—plot of the error estimation process.

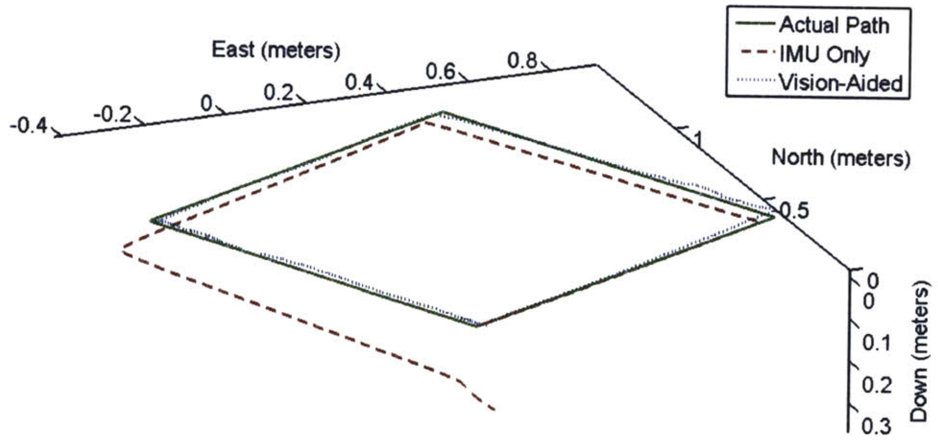


Figure 6-6: GantryB—full motion plot.

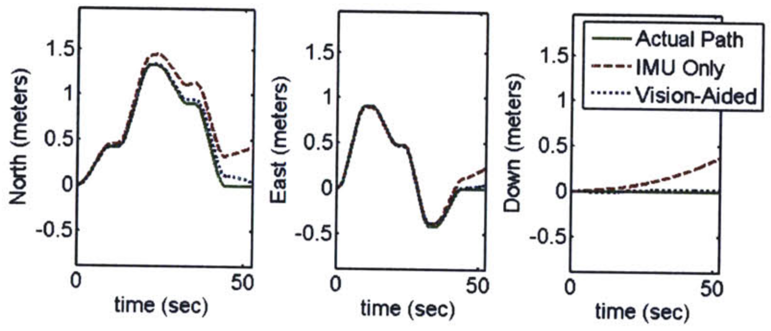


Figure 6-7: GantryB—plot of each dimension.

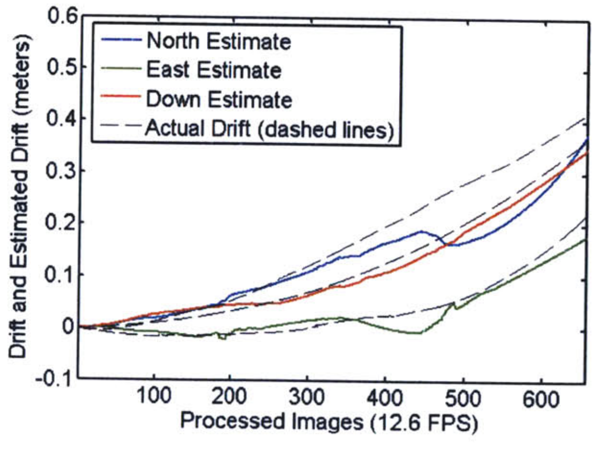


Figure 6-8: GantryB—plot of the error estimation process.

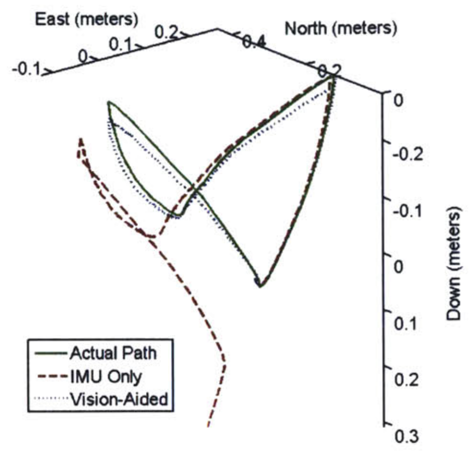


Figure 6-9: GantryC—full motion plot.

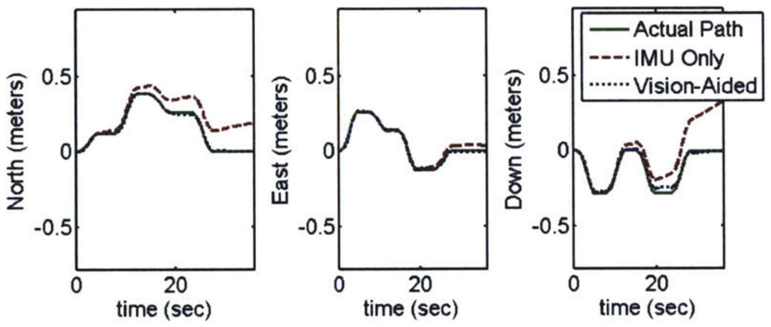


Figure 6-10: GantryC—plot of each dimension.

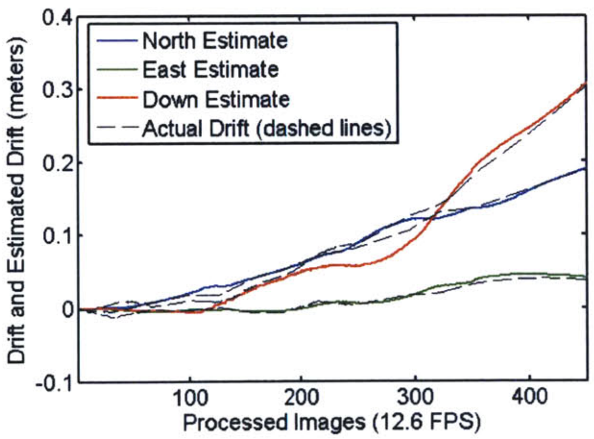


Figure 6-11: GantryC—plot of the error estimation process.

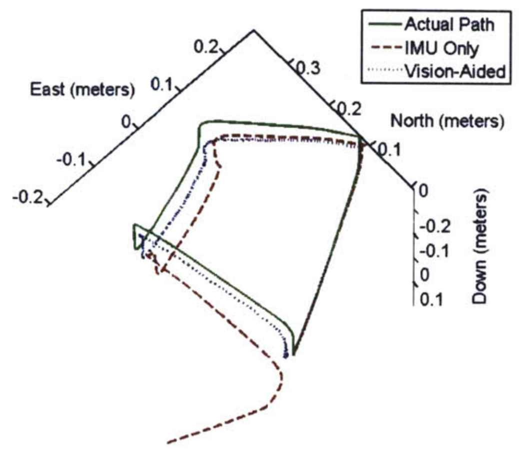


Figure 6-12: GantryF—full motion plot.

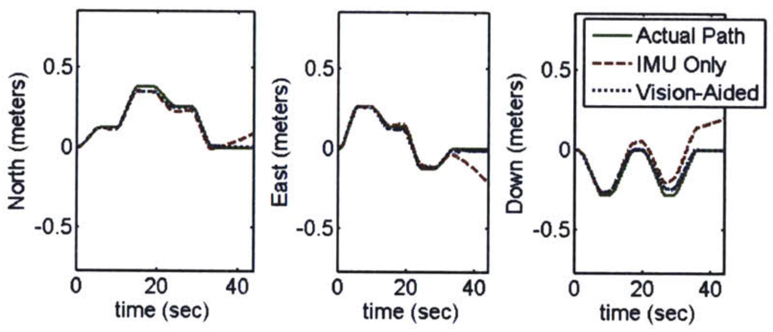


Figure 6-13: GantryF—plot of each dimension.

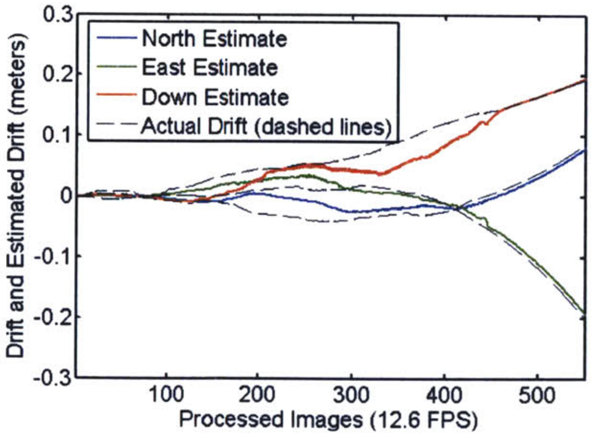


Figure 6-14: GantryF—plot of the error estimation process.

Chapter 7

Analysis

Each experiment presented in the previous chapter is an instance of a random process. The gantry data was collected with a single camera, lens, and IMU. And, the simulated scene provides one additional instance with a different set of hardware. There are four experiments in all.

The amount of data that has been collected is insufficient to determine the value of sensor tradeoffs, or to infer statistical characteristics of the filter. However, there are some aspects of the system that can be checked. In the following sections, we will validate our use of epipolar geometry, explore the overall performance of the filter, and address remaining issues.

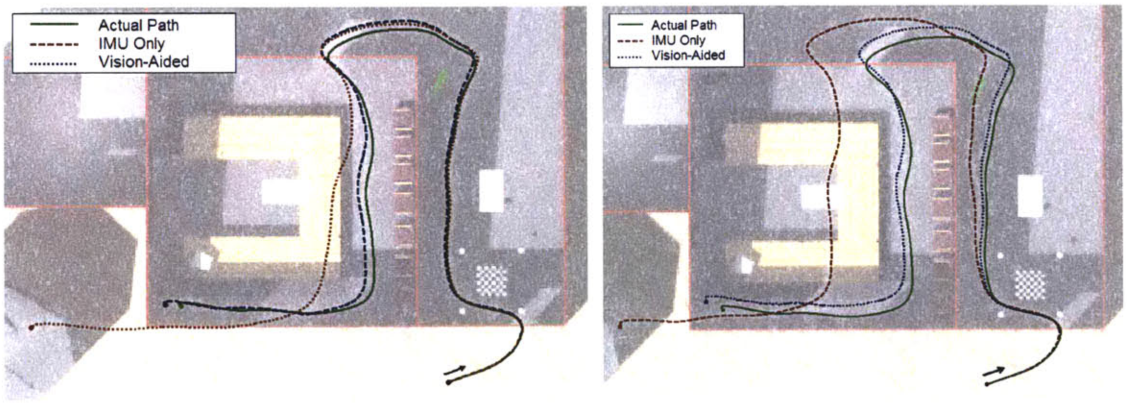


Figure 7-1: Factory7—Overhead plot showing two instances of the same scene. The inertial-visual simulator is capable of creating more instances, although this has been left to future work.

7.1 Validation of Epipolar Geometry

The geometry of the epipolar constraint should be consistent with the visual measurements in each data set. This test can be difficult to visualize with general motion. Thankfully, one data set, GantryB, begins with simple linear motion. In this case, the camera and the observed features together create epipolar constraint-plane normals that are perpendicular to the body path, as seen in Figure 7-2. The camera points downward (upward in the diagram), and the normals are formed by the right-hand-rule.

The epipolar constraint filter relies on rich body motion within a rich visual environment. Images do not provide epipolar constraints unless the body translates and features are visible. When the body motion is simple, as in the GantryB experiment, the constraints provide less information than when the motion is complex. In general, the results from the previous chapter show that visual richness is beneficial to the filter's performance.

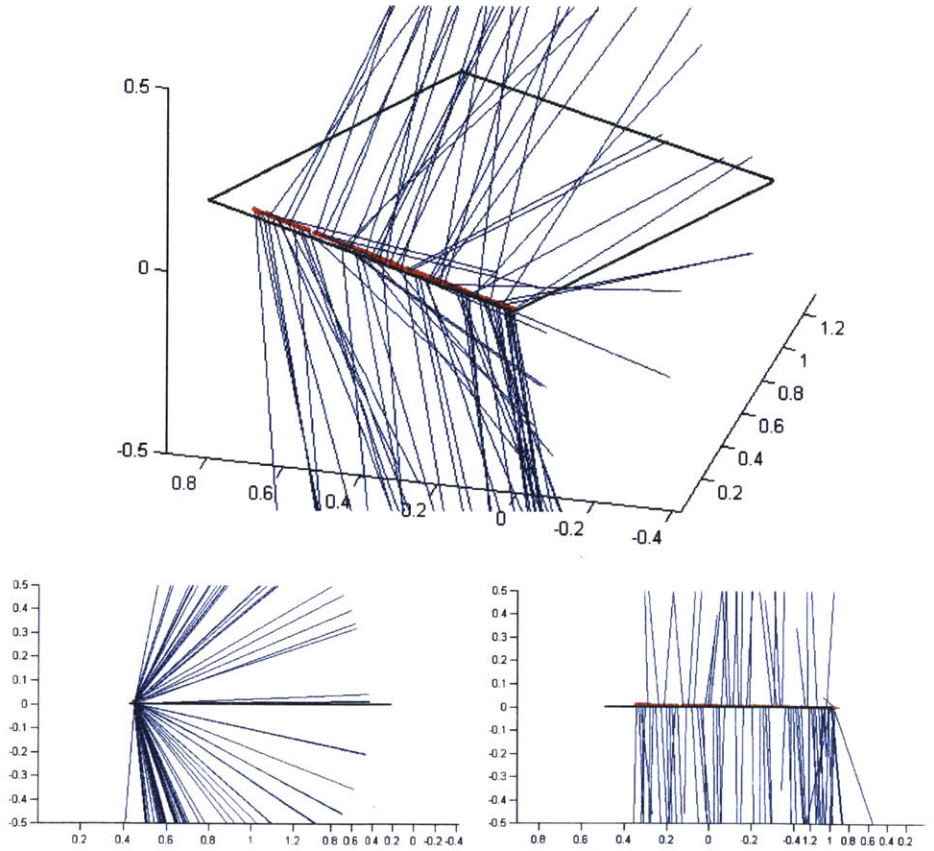


Figure 7-2: Randomly selected constraint-plane normals during the first few seconds of the GantryB experiment.

7.2 Overall Performance

Our primary goal has been to reduce inertial drift using visual information. Table 7.1 summarizes the performance of the proposed filter in terms of the magnitude of the endpoint error $\|\hat{\boldsymbol{x}}_T - \boldsymbol{x}_T\|$. Each experiment ran for a different length of time, so the total error may not be the best performance metric. For this reason, we also include the percent improvement owed to visual information. We would like to emphasize that the actual error reductions are substantial.

Title	$\sigma_{angular}$	IMU Only	Vision-Aided	Covariance Improvement	Actual Improvement
	radians	*meters*	*meters*		
Factory7	0.02	4.35	0.665	87%	85%
GantryB	0.04	0.61	0.062	97%	90%
GantryC	0.04	0.36	0.006	99%	98%
GantryF	0.04	0.30	0.012	99%	96%

Table 7.1: Summary of position error results for each of four data sets.

A covariance matrix represents the filter's internal model of its performance, and visual information should reduce the size of the covariance. For a metric of covariance size, we use the square-root of the matrix norm $\sqrt{\|\boldsymbol{\Lambda}_{1:3,1:3}\|}$. Referring to Table 7.1 again, the internal model predicts significant covariance improvements for every experiment. However, the data implies that the internal model is overconfident, because the actual improvements are less than the expected improvements.

The factory scene shows less improvement than the gantry scenes because there were fewer features in the simulated environment and the features were tracked for shorter periods of time.

The actual errors should be inliers of the normal error distribution model. In other words, the output of the filter should satisfy this inequality:

$$\sqrt{(\hat{\boldsymbol{x}}_T - \boldsymbol{x}_T)^T \Lambda_{1:3,1:3}^{-1} (\hat{\boldsymbol{x}}_T - \boldsymbol{x}_T)} < 2.5 \qquad (7.1)$$

where the number 2.5 defines a 98.8% confidence ellipsoid. This test is satisfied for all of our experiments. So, the covariance may be overconfident, but given that covariance, the error is still reasonable.

7.2.1 Relationship to SLAM

In Section 2.3.1 of Chapter 2, we discussed constraint-based localization as an alternative to SLAM. The essential principle was to treat the first observation of a feature as a reference for all subsequent observations, resulting in feature-relative localization. In a simplified one-dimensional experiment, the performance of the filter was degraded by unmodeled noise, but the overall position error remained bounded.

Our inertial-visual filter is conceptually similar to the filter in Chapter 2. Each measurement acts as a constraint defined relative to an initial observation. The measurement noise is lumped onto the current feature position. The computational requirements of mapping are eliminated by making a compromise in the noise model. And, the resulting filter is mildly overconfident, probably due to the unmodeled errors in the initial feature observations. Further analysis of the similarities between the epipolar constraint filter and constraint-based localization or SLAM are left to future work.

7.3 Remaining Issues

Unfortunately, the proposed filter exhibits chatter, as shown in Figure 7-3. This high-frequency noise appears intermittently as a velocity error that fights against the visual updates. It could be a result of Kalman learning rates that do not match the information content in the measurements, but the ultimate cause has not been determined. During our short experiments, the chatter remained small compared to the error being estimated. However, it could become a significant problem during long excursions.

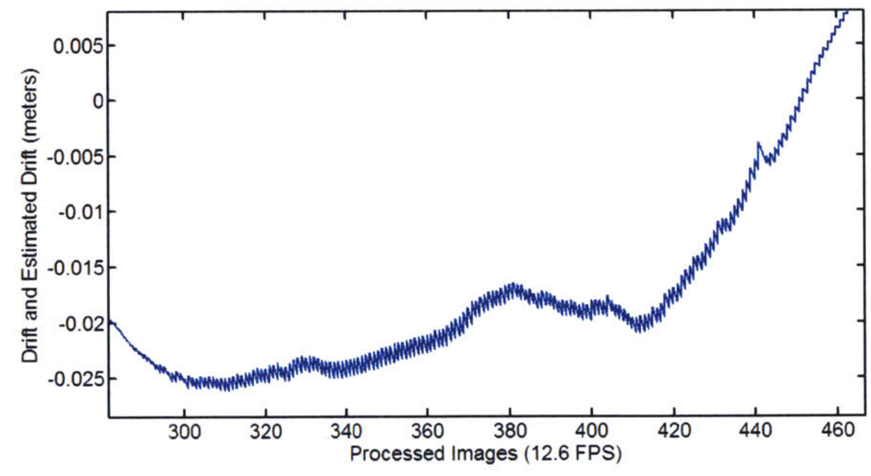


Figure 7-3: Chatter phenomenon.

In the gantry experiments, the performance of the IMU was better than anticipated. This is not a problem, but it can be explained by our data collection method. We turned on the sensor, allowed it to warm up for about 10 minutes, collected gantry motion data, and then recorded data while the gantry stood still. An estimate of the turn-on bias from the last data set was then subtracted from the previous data sets. Since the IMU was powered-down for only a fraction of a second between experiments, the turn-on bias probably did not vary. Therefore, the inertial data presented here was essentially bias-corrected before being integrated. This performance enhancement applies to all of the gantry experiments, with or without vision.

Chapter 8

Conclusions

We have proposed a new method for fusing visual and inertial data. The main problem with inertial data is unbounded drift, so our goal was to reduce or eliminate drift through stochastic visual constraints on the body position. In contrast to previous approaches, we sought a suboptimal solution that did not involve feature mapping. Our new general method is called *constraint-based localization*, and the new filter is called the *epipolar constraint filter*.

The proposed system is summarized by the block diagram in Figure 2-5. On the vision side, images from a calibrated camera are rotation-compensated, followed by feature detection and tracking. Ray-vector observations corresponding to feature directions are then passed to the filter. On the inertial side, the data comes from both the gyroscopes and the accelerometers, and we assume that the gyroscopes are ideal. The body acceleration is integrated in the world frame, taking into account a detailed model of apparent gravity, and a preliminary body position estimate is passed to the filter. Within the filter, a model of inertial uncertainty is also integrated. By defining a new epipolar residual with an assumed Gaussian error distribution, we are able to formulate a Bayes Least Squares (BLS) estimate of the inertial error states. Each visual feature provides one constraint, and multiple features have the potential to fully constrain the IMU error. The measurements reduce the size of a relative covariance matrix, and error updates are fed back into the inertial integration process through a normalized state transfer.

Although we have not proved any characteristics of the proposed method, this work contains several theoretical contributions. First, we offer a unified notation for inertial-visual estimation, presented in Tables 1.1 and 1.2. Our method is general with respect to the body motion, the IMU type, and any number of single-focus cameras connected by active linkages. Each camera can have a different image projection. And, the method is defined for any user-selectable number of visual features, including zero.

Experiments were performed on simulated and real data, and the results were promising. Localization errors were reduced by an order-of-magnitude during real experiments lasting about one minute.

8.1 Future Work

This work could be extended in several directions. In the near-term, we would like to determine how system parameters affect overall performance by conducting simulated experiments. Some important parameters are:

- Inertial bias, scale, and random walk.
- Camera projection type.
- Camera calibration accuracy.
- Image resolution.
- Field-of-view.
- Frame rate.
- Reprojection type.
- Richness of the visual environment.
- Corner strength metric.
- Tracking method.
- Length of an excursion.

The proposed system could be described as almost real-time, but it would be fully real-time with the help of custom hardware. Rotation compensation and corner detection each take about one *second* to complete on a desktop PC. These functions are simple, but they must be performed for every pixel in every image. Feature tracking is already fast, but it would also be a useful and marketable hardware function. All three of these functions lend themselves to an FPGA/SDRAM architecture.

In our implementation, the initial observation of a feature always plays the role of $\boldsymbol{z}[t_a]$. But, sometimes a later observation of the feature is more reliable than the initial observation. For instance, if a single GPS update became available, or an unexpected visual event is detected, then the features could be re-initialized at the next time step. A study of feature acquisition, tracking duration, and tracking accuracy could lead to deeper integration of our algorithm with GPS and other sensors.

Theoretical extensions to the filter might include state history smoothing (in the backward Kalman filter sense), linearized orientation updates to enable longer excursions, advanced treatment of outliers, and the straightforward application of depth estimation (SFM). An investigation of the measurement variance $\tilde{\sigma}$ could also provide insight into the performance of the filter.

Standardized data sets are useful for comparing alternative algorithms. To facilitate the future exchange of ideas, we have posted several sets of inertial-visual data on the web at `http://www.mit.edu/~ddiel/DataSets`.

Appendix A

System Parameter Tradeoffs

This appendix provides information related to the design of a visual-inertial navigation system. Such a system might include laser rangefinders, barometers, GPS receivers, and other sensors that are not discussed here. However, our aim is to evaluate a limited number of tradeoffs between IMU and camera parameters only. Specifically, we evaluate three IMUs, three image resolutions, and three areas of visual coverage.

A.1 Experiments

Since the process of scene generation is very slow, we did not want to create 27 independent scenes to test all possible parameter combinations. Instead, we chose to create 9 scenes with carefully-chosen parameters, based on Taguchi's method of orthogonal experiments [42].

For all experiments, the camera followed an identical one-minute path through a simulated factory. The three parameters in Table A.1 were varied while other system parameters were held constant. A single random seed was used to generate all of the inertial data sets so that the body path errors could be compared across experiments. The gyro noise was artificially set to zero, because the imaging algorithm was not designed to handle large orientation errors.

	IMU Type	**Resolution**	**Image Area**
	none	$\frac{pixels}{steradian}$	*steradians*
Option 1	LN2	29000	2π
Option 2	IMU1	23000	1.8π
Option 3	IMU2	17000	1.6π

Table A.1: System parameters and the options used in the array experiment. The IMU types are described in detail in Chapter 3. Note that the full visual sphere occupies 4π *steradians*.

ID	IMU Type	Image Resolution	Image Area	IMU Only	Vision-Aided
				meters	*meters*
A	Option 1	Option 1	Option 1	6.36 | 7.09	0.872 | 0.477
B	Option 1	Option 2	Option 2	6.36 | 7.09	0.969 | 0.821
C	Option 1	Option 3	Option 3	6.36 | 7.09	1.07 | 1.12
D	Option 2	Option 1	Option 2	36.7 | 47.6	4.71 | 6.57
E	Option 2	Option 2	Option 3	36.7 | 47.6	5.25 | 8.20
F	Option 2	Option 3	Option 1	36.7 | 47.6	2.91 | 9.79
G	Option 3	Option 1	Option 3	445 | 730	26.1 | 164
H	Option 3	Option 2	Option 1	445 | 730	51.7 | 63.3
I	Option 3	Option 3	Option 2	445 | 730	45.3 | 126

Table A.2: Results of the orthogonal array experiment. Expected and measured endpoint errors are listed next to each other in the divided columns.

A.2 Analysis

The results in Table A.2 come from a relatively small number of experiments, so any apparent trends should be observed with caution. In general, one should expect system performance to vary with sensor specifications. This is certainly true with respect to the IMU, as shown in Figure A-1. The IMU is the leading factor contributing to the overall performance.

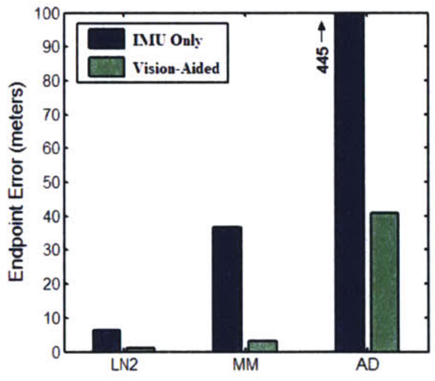


Figure A-1: Expected error for each IMU type, calculated as the square-root of the covariance matrix norm $\sqrt{\|\mathbf{\Lambda}_{1:3,1:3}\|}$ at the end of each one-minute experiment.

Intuitively, image resolution and image area must affect performance, but the amount of influence is uncertain. Considering Figure A-2, one can see that a reduction in image area negatively affects the percentage of error removed from the body path. However, the effect of image resolution is not so clear. Higher resolution is usually a benefit, but the data implies that there is an ideal resolution somewhere between Option 1 and Option 2.

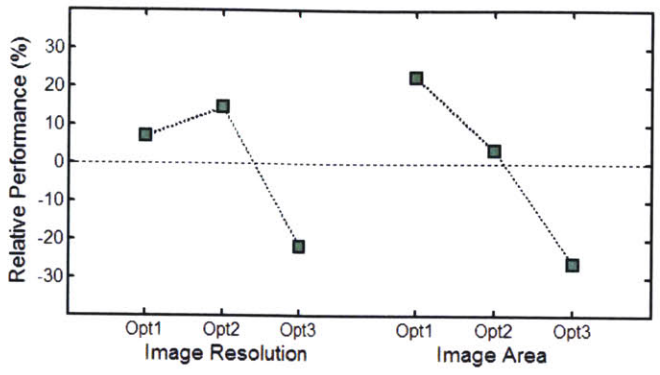


Figure A-2: Performance variation as a function of each imaging parameter. Each data point represents the mean of three experiments, and the horizontal line represents the mean of all nine experiments.

There are two plausible explanations for the resolution data. One likely explanation is that our experiments were insufficient to capture the effect of the parameter. Each data point is calculated as the mean of only three experiments, so the mean could be incorrect. Another explanation says that the data does represent the effect of the parameter–that there is an ideal resolution. High resolution images can make feature tracking difficult. For instance, a feature that moves 4 pixels in a low-resolution image could move 7 or 8 pixels in a high-resolution image. If the feature search radius is fixed at 5 pixels, then the feature will be successfully tracked in the first case, but not in the second. So, there may be an ideal resolution, given computational limitations, but our results are inconclusive.

Finally, these simulations confirm that our algorithm breaks down over time, and that the duration of usefulness depends on the sensor quality. Within a finite time, the filter becomes overconfident and the visual updates become sub-optimal. Then, as the error grows to the scale of the body path itself, the assembly of the epipolar constraint fails, and our method of outlier removal also fails. In particular, the last few seconds of experiments **G** and **I** could be described as wandering away in the wrong direction. Once the transient navigation capability is lost, the body states can only be recovered by introducing information from other sensors (ie. GPS).

Appendix B

Projection Reference

Images are formed by the projection of light rays onto a photosensitive surface. To visualize a general camera, imagine a unit sphere centered at a focus of projection[1]. A *forward projection* maps rays in the world to the image space, and an *inverse projection* maps pixels in the image space to rays in the world.

A few of the most common projection types will be presented here. Some of them correspond to real optical elements, while others are purely theoretical. For each projection type, we will provide visual-aides in two forms. First, we will demonstrate mapping to a rectangular *image space* by plotting latitude-longitude lines in the image at 10° intervals. Then, we will demonstrate mapping to the *ray space* by plotting red dots on a unit sphere, where each dot represents about 100 pixels. Visualizations appear in Figures B-1 to B-5 at the end of this appendix.

Symbolic notation for the following sections can be found in Table 1.1. A forward-right-down frame is associated with each projection such that $\vec{c} = [1\ 0\ 0]^{\mathrm{T}}$ corresponds to the optical axis of a camera. The image space $\boldsymbol{u} \in [-1\ \ 1]$ is sampled at discrete points, or pixels, using array coordinates $i \in [1 i_{max}]$ and $j \in [1 j_{max}]$.

B.1 Gall Isographic

The Gall isographic projection covers the full visual sphere. Along any given parallel or meridian, the mapping is equidistant, with minimal local distortion, except at the poles. In general, objects maintain a similar appearance as they move throughout most of the image space. The forward projection is given by

$$\boldsymbol{u} = \frac{1}{\pi}\begin{bmatrix} 2\arcsin\left(c_3\right) \\ \operatorname{arctan2}\left(\frac{c_2}{c_1}\right) \end{bmatrix} \tag{B.1}$$

[1]Also called a *Gaussian image*.

where $\boldsymbol{u}$ is stretched to fill an image array with $\frac{j_{max}}{i_{max}} = \sqrt{2}$. The inverse projection is given by

$$\vec{c} = \begin{bmatrix} \cos(\pi u_2)\cos\left(\frac{\pi}{2}u_1\right) \\ \sin(\pi u_2)\cos\left(\frac{\pi}{2}u_1\right) \\ \sin\left(\frac{\pi}{2}u_1\right) \end{bmatrix} \tag{B.2}$$

This projection was derived from the Matlab Mapping Toolbox documentation [43].

B.2 Apianus

Similar to the previous entry, the Apianus projection is also a full spherical projection. Distortion remains low over the central part of the image, and becomes moderate at the edges. Distances are preserved along parallels. The forward projection is given by

$$\boldsymbol{u} = \frac{1}{\pi}\begin{bmatrix} 2\arcsin(c_3) \\ \operatorname{arctan2}\left(\frac{c_2}{c_1}\right)\sqrt{1-\left(\frac{2}{\pi}\arcsin(c_3)\right)^2} \end{bmatrix} \tag{B.3}$$

where $\boldsymbol{u}$ is stretched to fill an image array with $\frac{j_{max}}{i_{max}} = 2$. The inverse projection is given by

$$\vec{c} = \begin{bmatrix} \cos\left(\frac{\pi u_2}{\sqrt{1-u_1^2}}\right)\cos\left(\frac{\pi}{2}u_1\right) \\ \sin\left(\frac{\pi u_2}{\sqrt{1-u_1^2}}\right)\cos\left(\frac{\pi}{2}u_1\right) \\ \sin\left(\frac{\pi}{2}u_1\right) \end{bmatrix} \tag{B.4}$$

This projection was derived from the Matlab Mapping Toolbox documentation [43].

B.3 Standard Perspective

Most lenses approximately fit the standard perspective model. Unfortunately, this model can only cover part of the image sphere, and distortion becomes untenable as the field-of-view approaches π radians[2]. This projection is characterized by a single focal parameter $\rho = \cot(\alpha_{max})$, where α_{max} is half of the horizontal field-of-view in radians. The forward projection is given by

$$\boldsymbol{u} = \frac{\rho}{c_1}\begin{bmatrix} c_3 \\ c_2 \end{bmatrix} \tag{B.5}$$

[2]One way cover the full sphere is to project the image onto six faces of a cube, where each face follows the standard perspective model.

where $\boldsymbol{u}$ is isotropically scaled to fill the horizontal dimension of an image array. The inverse projection is given by

$$\vec{c} = \frac{1}{\sqrt{u_1^2 + u_2^2 + \rho^2}} \begin{bmatrix} \rho \\ u_2 \\ u_1 \end{bmatrix} \tag{B.6}$$

B.4 Equidistant Fisheye

The term *fisheye* implies a wide-field-of-view. For simplicity, the equidistant fisheye model presented here covers π radians, or a frontal hemisphere, although it can be extended to the full sphere. This projection maintains equal angular spacing along any radius[3]. The forward projection is given by

$$\boldsymbol{u} = \frac{2 \arccos(c_1)}{\pi \sqrt{1 - c_1^2}} \begin{bmatrix} c_3 \\ c_2 \end{bmatrix} \tag{B.7}$$

where $\boldsymbol{u}$ is stretched to fill a square image array. The inverse projection is given by

$$\vec{c} = \begin{bmatrix} \cos\left(\frac{\pi \|\boldsymbol{u}\|}{2}\right) \\ \frac{u_2}{\|\boldsymbol{u}\|} \sin\left(\frac{\pi \|\boldsymbol{u}\|}{2}\right) \\ \frac{u_1}{\|\boldsymbol{u}\|} \sin\left(\frac{\pi \|\boldsymbol{u}\|}{2}\right) \end{bmatrix} \tag{B.8}$$

This projection was derived from POV-Ray public source code.

B.5 Radial Division Model

The radial division model comes from a synthesis of recently proposed projections. It represents a variety of wide-field-of-view optics, including the fisheye lens used to collect data for this paper. The following equations cover only the frontal hemisphere, but the model can be extended to the full sphere. The forward projection is given by

$$\boldsymbol{u} = \frac{r}{\sqrt{1 - c_1^2}} \begin{bmatrix} c_3 \\ c_2 \end{bmatrix} \tag{B.9}$$

with

$$r = \frac{\rho_1 \arccos(c_1) + \rho_2 \arccos(c_1)^2}{1 + \rho_3 \arccos(c_1) + \rho_4 \arccos(c_1)^2}$$

[3]It is the natural logarithmic mapping from $SO(2)$ to $\mathbb{R}^2$.

where $\boldsymbol{u}$ is stretched to fill a square image array. The inverse projection is given by

$$\vec{\boldsymbol{c}} = \begin{bmatrix} \cos(\alpha) \\ \frac{u_2}{\|\boldsymbol{u}\|}\sin(\alpha) \\ \frac{u_1}{\|\boldsymbol{u}\|}\sin(\alpha) \end{bmatrix} \tag{B.10}$$

with

$$\alpha = \begin{cases} \frac{\rho_3\|\boldsymbol{u}\|-\rho_1+\sqrt{(\rho_3\|\boldsymbol{u}\|-\rho_1)^2+4\left(\rho_2\|\boldsymbol{u}\|-\rho_4\|\boldsymbol{u}\|^2\right)}}{2(\rho_2-\rho_4\|\boldsymbol{u}\|)} & \text{when} \quad \rho_2-\rho_4\|\boldsymbol{u}\| \neq 0 \\ \frac{\|\boldsymbol{u}\|}{\rho_1-\rho_3\|\boldsymbol{u}\|} & \text{when} \quad \rho_2-\rho_4\|\boldsymbol{u}\| = 0 \end{cases}$$

Notice that the equidistant fisheye projection is a special case of the radial division model with $\boldsymbol{\rho} = \left[\frac{2}{\pi}\ 0\ 0\ 0\right]$.

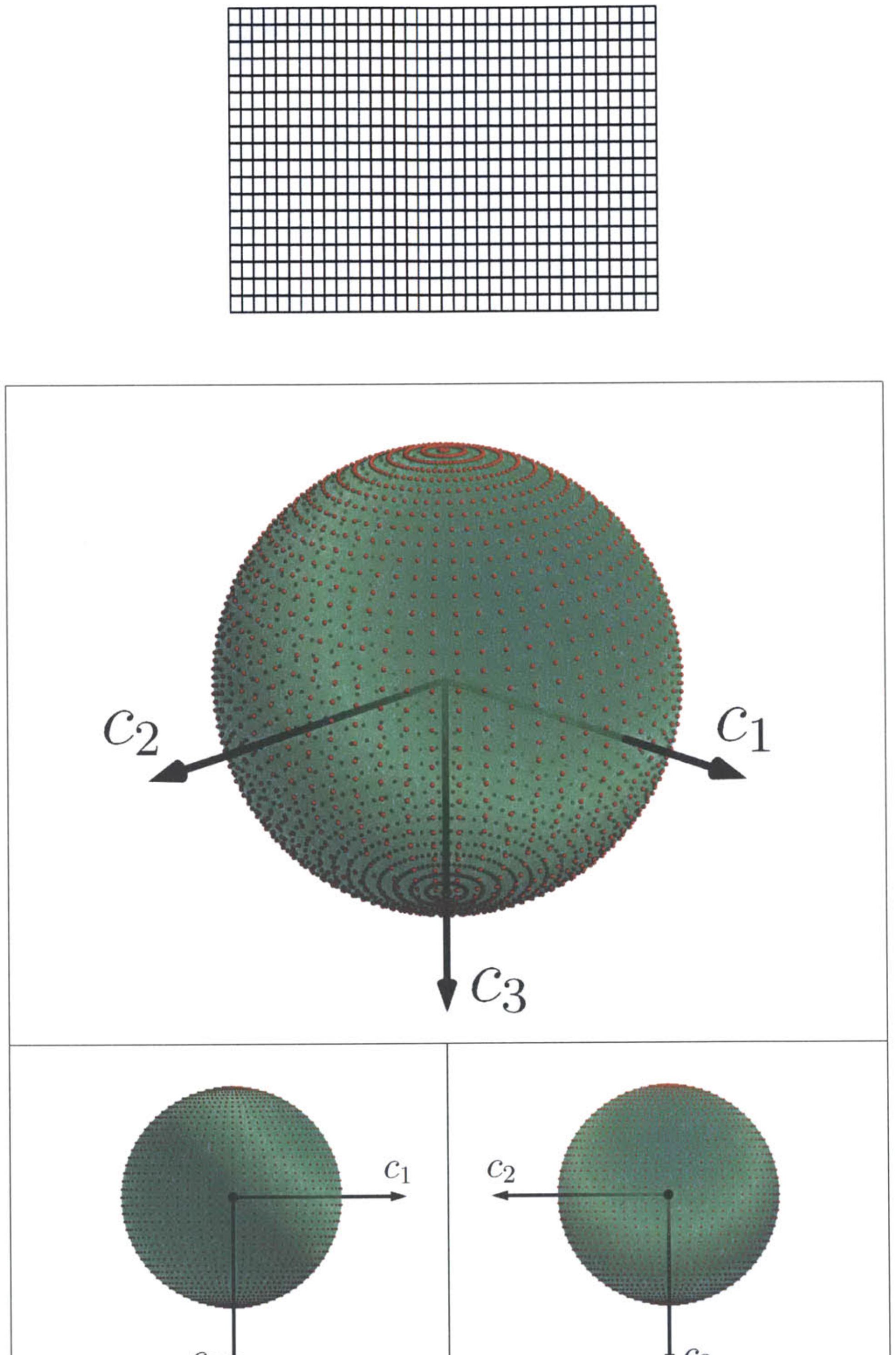
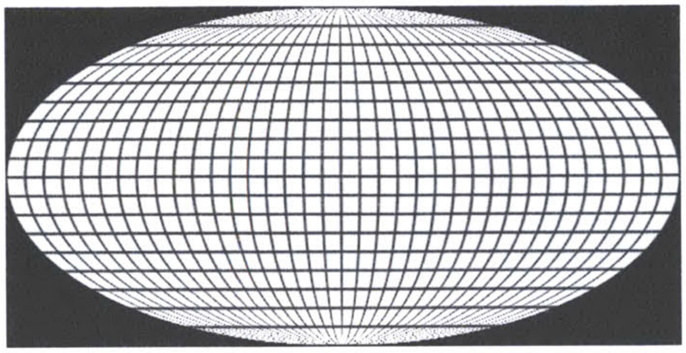


Figure B-1: Demonstration of the Gall isographic projection.



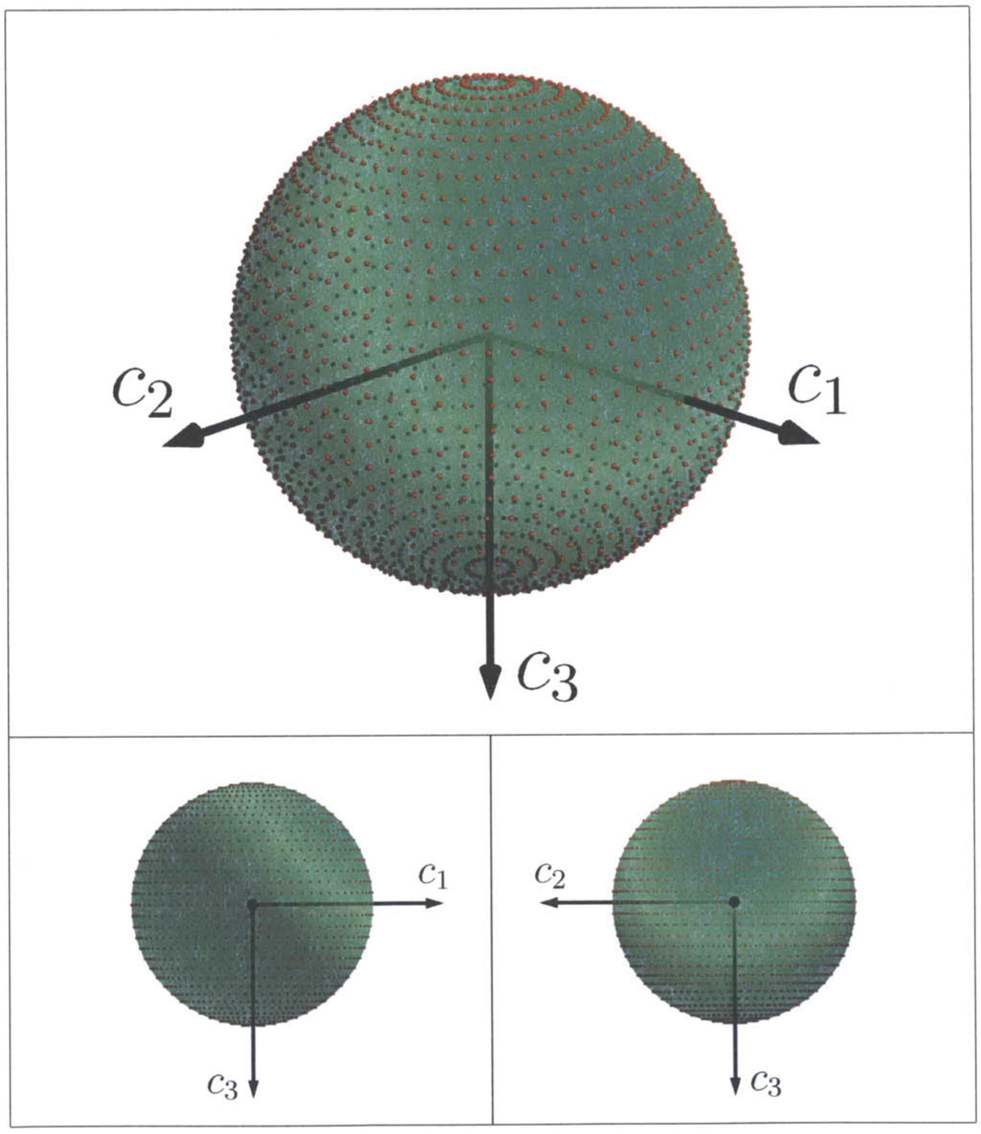


Figure B-2: Demonstration of the Apianus projection.

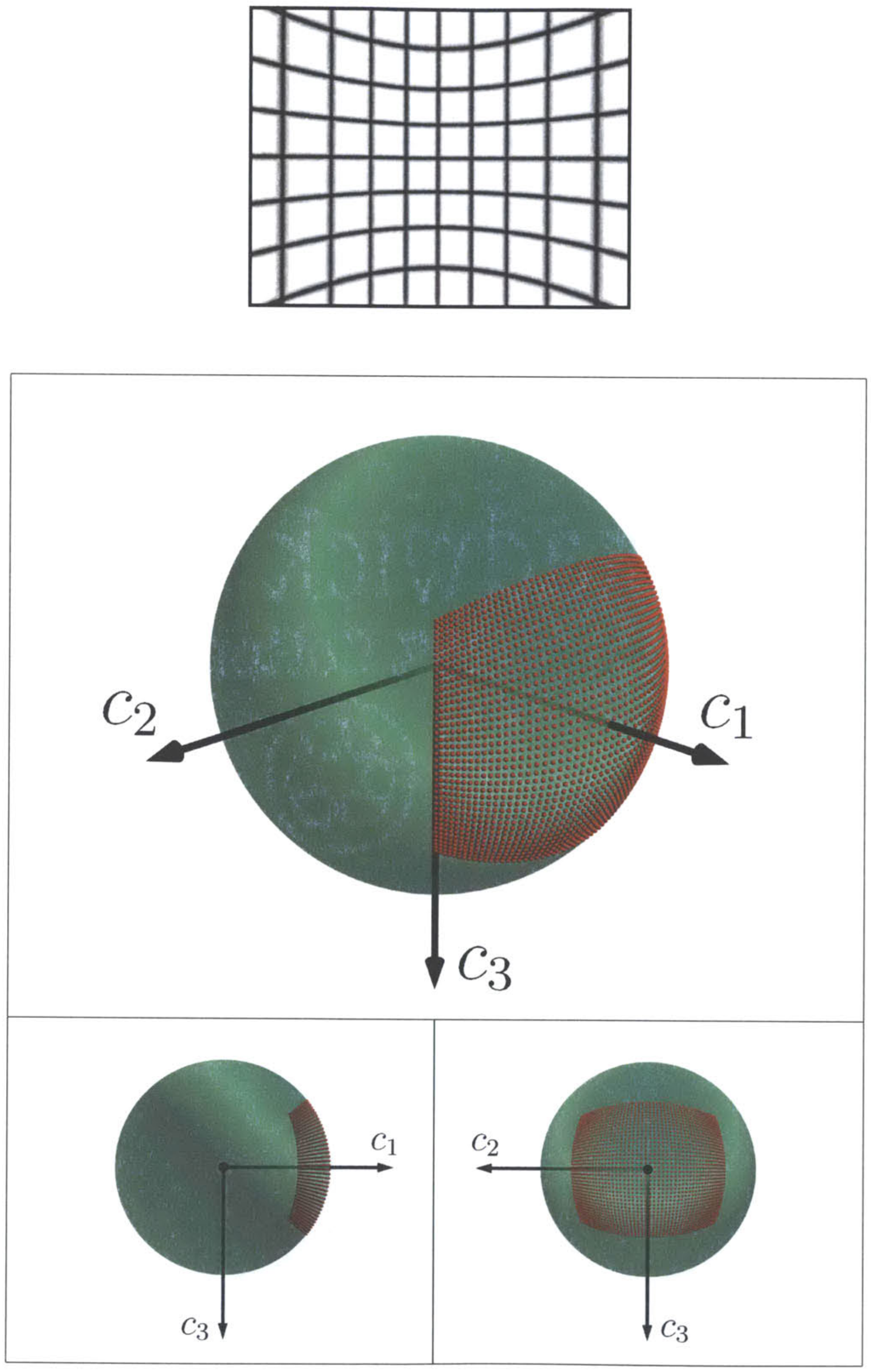
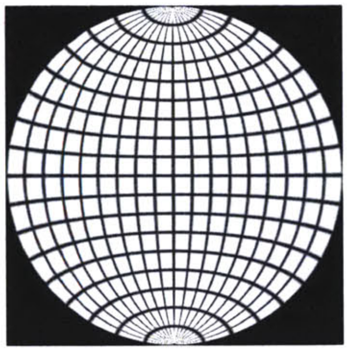


Figure B-3: Demonstration of the standard perspective projection.



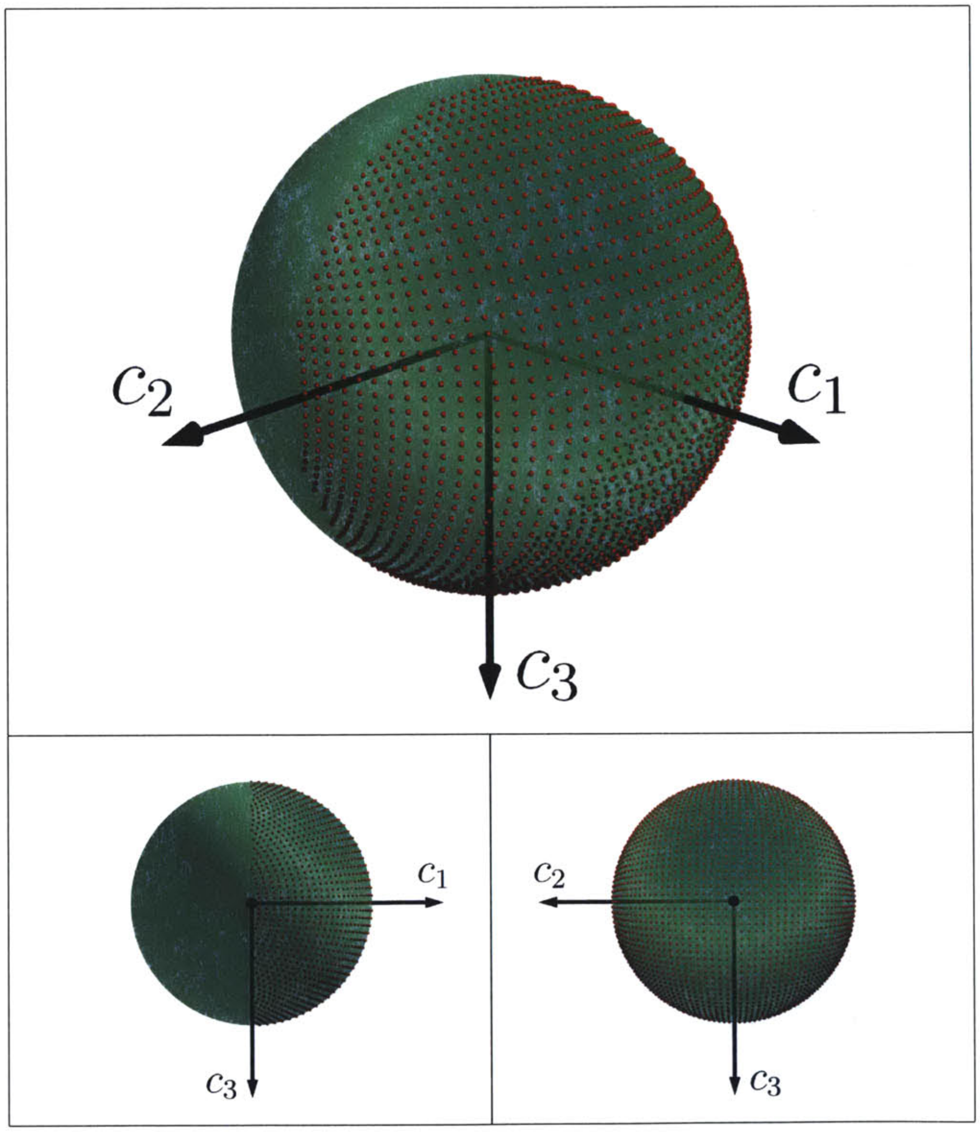


Figure B-4: Demonstration of the equidistant fisheye projection.

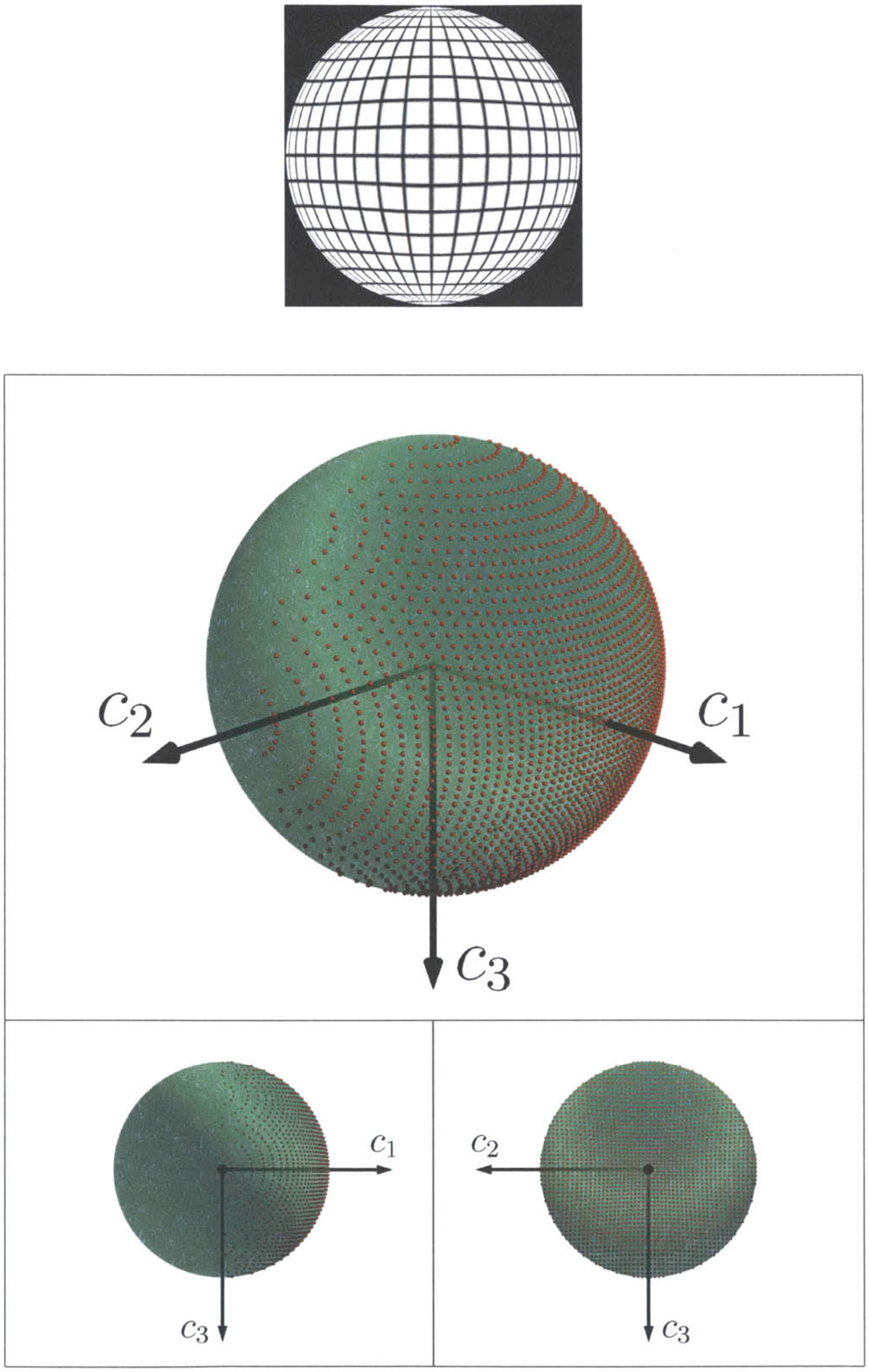


Figure B-5: Demonstration of the radial division model for projection.

Appendix C

Rotation Reference

There are many ways to represent rotation, and many possible conversions between representations [23][48]. Confusion over representation can be very difficult to resolve if a document does not provide clarification. The purpose of this Appendix is *not* to present all possible forms and conversions, but instead to present a minimum set of self-consistent representations.

Primitives must be defined before rotation can have meaning. First, we assume that two reference frames exist, where each frame is defined as a right-handed orthogonal triad. The frames represent local coordinate systems attached to each of two bodies. One frame is attached to a non-rotating body called the "world," and the other frame is attached to a free-floating body. The notation used in this Appendix can be found in Table C.1.

C.1 Orientation Representations

The process of adopting conventions usually involves some combination of rational persuasion and arm-twisting. Why use right-handed frames instead of left-handed ones? One could invent technical arguments, but in reality right-handed people simply outnumber left-handed people, and for most applications that is the determining factor [1].

To reduce the number of acceptable orientation representations, we must choose conventions. Our process begins by selecting only right-handed frames. Next, we choose four forms of notation: matrix, quaternion, Euler, and axis-angle. The matrix form has two versions, and we choose the one that transforms points in the body frame to the world frame. The quaternion form can be written in scalar-vector or vector-scalar notation. We select the scalar-vector notation so that $\vec{q} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ means "no rotation." For the Euler

[1]Farmer Philo T. Farnsworth invented the scan-line process used in the first television by analogy to the way he plowed his fields—East to West, then North to South. Many years later came the invention of the *z-buffer*, an array of positive numbers representing virtual depth into a computer screen. These two historical factors have made the right-down-forward convention popular for researchers in computer graphics and vision.

Symbol	Form	Size
$\vec{q}$	Quaternion	4-by-1
v	Axis-Angle	3-by-1
ϕ	Euler	3-by-1
$\boldsymbol{R}$	Matrix	3-by-3

Table C.1: Special notation for Appendix C only.

form, there are 12 reasonable axis sequences [2]: 121, 123, 131, 132, 212, 213, 231, 232, 312, 313, 321, 323. We choose the 123 axis sequence because it works intuitively with the forward-right-down camera frame convention. The outermost axis pans right, the middle axis tilts up, and the innermost axis rotates clockwise.

Without the previous choices, conversion between representations would be extremely burdensome. Only considering the matrix, quaternion, Euler, and axis-angle forms, there are $(34-1)34 = 1122$ possible conversions. But, by limiting the acceptable versions of each form, the number of conversions has been reduced to $(4-1)4 = 12$. This is a more tractable number. Therefore, we will present each of the four well-defined representations, and all twelve of the conversions between them.

C.1.1 Matrix Form

Matrices have the power to transform vectors from one frame to another. A rotation matrix can be constructed column-by-column from three body frame axes viewed in the world frame:

$$\boldsymbol{R} = \begin{bmatrix} | & | & | \\ \vec{e}_1 & \vec{e}_2 & \vec{e}_3 \\ | & | & | \end{bmatrix} \tag{C.1}$$

where $\vec{e}_1$, $\vec{e}_2$, and $\vec{e}_3$ represent the first, second, and third body frame axes. This matrix would rotate a point in the body frame into the world frame, and its inverse would rotate a point in the world frame into the body frame.

Conversion: Euler → Matrix

For shorthand, we use $\mathrm{c}_n = \cos(\phi_n)$ and $\mathrm{s}_n = \sin(\phi_n)$.

$$\boldsymbol{R}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_1 & -\mathrm{s}_1 \\ 0 & \mathrm{s}_1 & \mathrm{c}_1 \end{bmatrix}, \quad \boldsymbol{R}_2 = \begin{bmatrix} \mathrm{c}_2 & 0 & \mathrm{s}_2 \\ 0 & 1 & 0 \\ -\mathrm{s}_2 & 0 & \mathrm{c}_2 \end{bmatrix}, \quad \boldsymbol{R}_3 = \begin{bmatrix} \mathrm{c}_3 & -\mathrm{s}_3 & 0 \\ \mathrm{s}_3 & \mathrm{c}_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

[2]The axis sequences are also known as gimbal configurations.

$$R = R_3 R_2 R_1 = \begin{bmatrix} c_3\,c_2 & c_3\,s_2\,s_1 - s_3\,c_1 & s_3\,s_1 + c_3\,s_2\,c_1 \\ s_3\,c_2 & c_3\,c_1 + s_3\,s_2\,s_1 & s_3\,s_2\,c_1 - c_3\,s_1 \\ -s_2 & c_2\,s_1 & c_2\,c_1 \end{bmatrix} \tag{C.2}$$

Conversion: Quaternion → Matrix

$$R = \begin{bmatrix} (q_1^2 + q_2^2 - q_3^2 - q_4^2) & 2\,(q_2 q_3 - q_1 q_4) & 2\,(q_2 q_4 + q_1 q_3) \\ 2\,(q_2 q_3 + q_1 q_4) & (q_1^2 - q_2^2 + q_3^2 - q_4^2) & 2\,(q_3 q_4 - q_1 q_2) \\ 2\,(q_2 q_4 - q_1 q_3) & 2\,(q_3 q_4 + q_1 q_2) & (q_1^2 - q_2^2 - q_3^2 + q_4^2) \end{bmatrix} \tag{C.3}$$

Conversion: Axis-Angle → Matrix

$$R = I + V \sin(\|v\|) + V^2 (1 - \cos(\|v\|)) \tag{C.4}$$

$$V = \frac{1}{\|v\|} \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

where the angle $\|v\|$ is shifted to the range $[0, 2\pi)$ by adding or subtracting increments of 2π as needed. This formula is credited to a Frenchman by the name of Olinde Rodrigues[3].

C.1.2 Quaternion Form

A unit quaternion is a compact, hypercomplex, singularity-free representation of body orientation. Sir William Rowan Hamilton invented quaternions in the nineteenth century [20], and Ken Shoemake brought them to the attention of the computer graphics community [47]. A quaternion can be written as the sum of a scalar and three orthogonal imaginary numbers:

$$\vec{q} \equiv q_1 + q_2 \boldsymbol{i} + q_3 \boldsymbol{j} + q_4 \boldsymbol{k} \tag{C.5}$$

Dropping the imaginary notation, the same quaternion can be expressed as a vector of real numbers:

$$\vec{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^{\mathrm{T}} \tag{C.6}$$

Quaternions have several interesting and useful properties. A quaternion conjugate represents an inverse rotation $\vec{q}^{-1} = \vec{q}^*$. If a quaternion is written in homogenous matrix notation

$$\vec{q} = \begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \\ q_2 & q_1 & -q_4 & q_3 \\ q_3 & q_4 & q_1 & -q_2 \\ q_4 & -q_3 & q_2 & q_1 \end{bmatrix} \tag{C.7}$$

[3]There are many incorrect references to Rodrigues' formula in mathematical literature. For more information, see [54].

then consecutive rotations can be calculated by matrix multiplication. Other properties of quaternions can be found in textbooks on dynamics and kinematics.

Conversion: Euler $\longrightarrow$ Quaternion

$$\vec{q}_1 = \cos\left(\frac{\phi_1}{2}\right) + \sin\left(\frac{\phi_1}{2}\right)\boldsymbol{i}$$
$$\vec{q}_2 = \cos\left(\frac{\phi_2}{2}\right) + \sin\left(\frac{\phi_2}{2}\right)\boldsymbol{j}$$
$$\vec{q}_3 = \cos\left(\frac{\phi_3}{2}\right) + \sin\left(\frac{\phi_3}{2}\right)\boldsymbol{k}$$

$$\vec{q} = \vec{q}_3\vec{q}_2\vec{q}_1 = \begin{bmatrix} \cos\left(\frac{\phi_3}{2}\right)\cos\left(\frac{\phi_2}{2}\right)\cos\left(\frac{\phi_1}{2}\right) + \sin\left(\frac{\phi_3}{2}\right)\sin\left(\frac{\phi_2}{2}\right)\sin\left(\frac{\phi_1}{2}\right) \\ \cos\left(\frac{\phi_3}{2}\right)\cos\left(\frac{\phi_2}{2}\right)\sin\left(\frac{\phi_1}{2}\right) - \sin\left(\frac{\phi_3}{2}\right)\sin\left(\frac{\phi_2}{2}\right)\cos\left(\frac{\phi_1}{2}\right) \\ \cos\left(\frac{\phi_3}{2}\right)\sin\left(\frac{\phi_2}{2}\right)\cos\left(\frac{\phi_1}{2}\right) + \sin\left(\frac{\phi_3}{2}\right)\cos\left(\frac{\phi_2}{2}\right)\sin\left(\frac{\phi_1}{2}\right) \\ \sin\left(\frac{\phi_3}{2}\right)\cos\left(\frac{\phi_2}{2}\right)\cos\left(\frac{\phi_1}{2}\right) - \cos\left(\frac{\phi_3}{2}\right)\sin\left(\frac{\phi_2}{2}\right)\sin\left(\frac{\phi_1}{2}\right) \end{bmatrix} \tag{C.8}$$

Conversion: Axis-Angle $\longrightarrow$ Quaternion

$$\vec{q} = \begin{bmatrix} \cos\left(\frac{\|\boldsymbol{v}\|}{2}\right) \\ \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}\sin\left(\frac{\|\boldsymbol{v}\|}{2}\right) \end{bmatrix} \tag{C.9}$$

where the angle $\|\boldsymbol{v}\|$ is shifted to the range $[0, 2\pi)$ by adding or subtracting increments of 2π, and the reference orientation $\vec{q} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$ covers the singular case $\|\boldsymbol{v}\| = 0$.

Conversion: Matrix $\longrightarrow$ Quaternion

Direct conversion from matrix form to quaternion form requires an algorithm. Although implementations of the algorithm are widely available [47], it is simple enough to convert to an intermediate Euler form, followed by conversion to quaternion form.

C.1.3 Euler Form

Euler rotations are applied about multiple axes in sequential order. To find our Euler representation, one would ask, "Beginning with the body frame aligned with the world frame, how can I sequentially rotate the body about the world's forward, right, and down axes to arrive at the current orientation?" Although the Euler representation is physically grounded, its formulation is non-intuitive, and it suffers from singularities. Specifically, there are multiple numerical values that represent the same orientation.

Conversion: Matrix → Euler

$$\phi = \begin{bmatrix} \arctan\left(\dfrac{R_{32}}{R_{33}}\right) \\ \arcsin\left(-R_{31}\right) \\ \arctan\left(\dfrac{R_{21}}{R_{11}}\right) \end{bmatrix} \tag{C.10}$$

Conversion: Quaternion → Euler

$$\phi = \begin{bmatrix} \arctan\left(\dfrac{2\left(q_3q_4 - q_1q_2\right)}{q_1^2 - q_2^2 - q_3^2 + q_4^2}\right) \\ \arcsin\left(-2\left(q_2q_4 + q_1q_3\right)\right) \\ \arctan\left(\dfrac{2\left(q_2q_3 - q_1q_4\right)}{q_1^2 + q_2^2 - q_3^2 - q_4^2}\right) \end{bmatrix} \tag{C.11}$$

Conversion: Axis-Angle → Euler

Convert first to quaternion form, and then convert to Euler form.

C.1.4 Axis-Angle Form

As the name implies, the axis-angle form represents an angular rotation about a single axis. The direction of $\boldsymbol{v}$ determines the axis, and the magnitude determines the angle in radians.

Conversion: Quaternion → Axis-Angle

$$\boldsymbol{v} = \frac{\arccos\left(q_1\right)}{\sqrt{1 - q_1^2}} \begin{bmatrix} q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{C.12}$$

Conversion: Matrix → Axis-Angle

$$\boldsymbol{v} = \frac{\arccos\left(\frac{1}{2}\left(R_{11} + R_{22} + R_{33} - 1\right)\right)}{\sqrt{\left(R_{32} - R_{23}\right)^2 + \left(R_{13} - R_{31}\right)^2 + \left(R_{21} - R_{12}\right)^2}} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} \tag{C.13}$$

Conversion: Euler → Axis-Angle

Convert first to quaternion form, and then convert to axis-angle form.

C.2 Body Orientation from Angular Velocity

A typical 3-axis gyroscope measures angular velocity ω in its own body frame. One basic problem in inertial navigation is to find the body orientation from a history of angular velocity measurements. After careful inspection, we discover that ω is not a gradient of any potential. In other words, integrating it element-by-element yields no meaningful representation. Instead, we need to store a summary of past information using one of the previously discussed forms. Experience has shown that the quaternion form is best for this purpose.

The following derivation makes use of vector and hypercomplex notation as needed. To integrate angular velocity, we first consider a change in orientation over a small amount of time:

$$\mathrm{d}\vec{q} \equiv \begin{bmatrix} \cos\left(\frac{\|\omega\|\Delta t}{2}\right) \\ \frac{\omega}{\|\omega\|}\sin\left(\frac{\|\omega\|\Delta t}{2}\right) \end{bmatrix} \tag{C.14}$$

From the definition of the derivative, we have

$$\dot{q}\,(t) \equiv \lim_{\Delta t \to 0} \frac{\vec{q}\,(t+\Delta t) - \vec{q}\,(t)}{\Delta t} \tag{C.15}$$

Then, dropping the explicit dependence on time, and noting that rotations accumulate by multiplication.

$$\begin{aligned} \dot{q} &= \lim_{\Delta t \to 0} \frac{\vec{q}\mathrm{d}\vec{q} - \vec{q}}{\Delta t} \\ &= \lim_{\Delta t \to 0} \frac{\vec{q}}{\Delta t} \begin{bmatrix} \cos\left(\frac{\|\omega\|\Delta t}{2}\right) - 1 \\ \frac{\omega}{\|\omega\|}\sin\left(\frac{\|\omega\|\Delta t}{2}\right) \end{bmatrix} \\ &= \frac{1}{2}\vec{q}\begin{bmatrix} 0 \\ \omega \end{bmatrix} \end{aligned} \tag{C.16}$$

The quaternion rate may take on any magnitude perpendicular to the quaternion itself, $q\dot{q} = 0$. Finally, to find the body orientation, one would simply integrate the last form of Equation C.16.

Bibliography

[1] G. Alenyà, E. Martnez, and C. Torras. Fusing visual and inertial sensing to recover robot egomotion. *Journal of Robotic Systems*, 21:23–32, 2004.

[2] Adnan Ansar and Kostas Daniilidis. Linear pose etimation from points or lines. In *European Converence on Computer Vision*, pages 282–296, 2002.

[3] Matthew Antone and Seth Teller. Scalable extrinsic calibration of omni-directional image networks. *International Journal of Computer Vision*, 49(2–3):143–174, September–October 2002.

[4] Matthew E. Antone. *Robust Camera Pose Recovery Using Stochastic Geometry*. PhD thesis, Massachusetts Institute of Technology, February 2001.

[5] H. Bakstein and T. Pajdla. Panoramic mosaicing with a 180° field of view lens. In *Third Workshop on Omnidirectional Vision*, pages 60–67, June 2002.

[6] J. L. Barron, D. J. Fleet, S. S. Beauchemin, and T. A. Burkitt. Performance of optical flow techniques. *Computer Vision and Pattern Recognition*, pages 236–242, 1992.

[7] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas framework for scalable mapping. In *International Conference on Robotics and Automation*, volume 2, pages 1899–1906, September 2003.

[8] M. Bosse, R. Rikoski, J. Leonard, and S. Teller. Vanishing points and 3D lines from omnidirectional video. In *International Conference on Image Processing*, volume 3, pages 513–516, June 2002.

[9] Tye M. Brady, Clemens E. Tillier, Robert A. Brown, Antonio R. Jimenez, and Anthony S. Kourepenis. The inertial stellar compass: A new direction in spacecraft attitude determination. In *16th Annual AIAA/USU Conference on Small Satellites*, Logan, Utah, August 2002.

[10] A. Chiuso, P. Favaro, Hailin Jin, and S. Soatto. Structure from motion causally integrated over time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):523–535, April 2002.

[11] A. Chiuso, H. Jin, P. Favaro, and S. Soatto. 3-D motion and structure from 2-D motion causally integrated over time: Implementation. In *Sixth European Conference on Computer Vision*, pages 734–750, 2000.

[12] A. Chiuso and G. Picci. A wide-sense estimation theory on the unit sphere. In *IEEE 37th Conference on Decision and Control*, pages 3743–9754, Tampa, Florida, December 1998.

[13] Andrew J. Davison. *Mobile Robot Navigation Using Active Vision.* PhD thesis, University of Oxford, October 1999.

[14] Andrew J. Davison. SLAM with a single camera. In *SLAM/CML Workshop at International Conference on Robotics and Automation*, 2002.

[15] Andrew J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *International Conference on Computer Vision*, Nice, France, October 2003.

[16] James Donna. Personal conversations at the Charles Stark Draper Laboratory, 2004.

[17] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley-Interscience, 2 edition, 2000.

[18] EUROCONTROL and IfEN. *WGS84 Implementation Manual*, 2.4 edition, February 1998.

[19] Ryan Eustice, Oscar Pizarro, and Hanumant Singh. Visually augmented navigation in an unstructured environment using a delayed state history. *International Conference on Robotics and Automation*, April 2004.

[20] W. R. Hamilton. *The Mathematical Papers of Sir William Rowan Hamilton.* Cambridge University Press, Cambridge, England, 1967.

[21] C. Harris and M. Stephens. A combined corner and edge detector. *Fourth Alvey Vision Conference*, pages 147–151, 1988.

[22] B. Horn and B. Schunk. Determining optical flow. *Artificial Intelligence*, 1981.

[23] P.C. Hughes. *Spacecraft Attitude Dynamics.* Wiley, NY, 1986.

[24] Andreas Huster. *Relative Position Sensing by Fusing Monocular Vision and Inertial Rate Sensors.* PhD thesis, Stanford University, July 2003.

[25] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. *SPIE AeroSense Symposium*, April 1997.

[26] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82, Series D:35–45, 1960.

[27] A. D. King. Inertial navigation—forty years of evolution. *General Electric Company Review*, 13(3), 1998.

[28] John J. Leonard, Richard J. Rikoski, Paul M. Newman, and Michael Bosse. Mapping partially observable features from multiple uncertain vantage points. *International Journal of Robotics Research*, 21:943–975, 2002.

[29] J. P. Lewis. Fast normalized cross-correlation. *Vision Interface*, 1995.

[30] Litton Guidance and Control Systems. *Product Description of the LN-200 Family*, September 1996. Document No. 208961.

[31] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[32] Chien-Ping Lu, Gregory D. Hager, and Eric Mjolsness. Fast and globally convergent pose estimation from video images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):610–622, 2002.

[33] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *7th International Joint Conference on Artificial Intelligence*, 1981.

[34] Yi Ma, Jana Košecká, and Shankar Sastry. Motion recovery from image sequences: Discrete viewpoint vs. differential viewpoint. In *European Conference on Computer Vision*, Freiburg, Germany, 1998.

[35] Yi Ma, R. Vidal, S. Hsu, and S. Sastry. Optimal motion estimation from multiple images by normalized epipolar constraint. *Communications in Information and Systems*, 1(1):51–74, January 2001.

[36] Ameesh Makadia and Kostas Daniilidis. Correspondenceless ego-motion estimation using an IMU. *To appear in IEEE International Conference on Robotics and Automation*, April 2005.

[37] M. Montemerlo and S. Thrun. Simultaneous localization and mapping with unknown data association using FastSLAM. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1985–1991, September 2003.

[38] Jan Neumann, Cornelia Fermüller, and Yiannis Aloimonos. Eye design in the plenoptic space of light rays. In *Ninth IEEE International Conference on Computer Vision*, pages 1160–1167, October 2003.

[39] Jan Neumann, Cornelia Fermüller, and Yiannis Aloimonos. Polydioptric camera design and 3D motion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 294–301, June 2003.

[40] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[41] W. Pasman, S. Zlatanova, S. Persa, and J. Caarls. Alternatives for optical tracking. Technical report, Delft University of Technology, UbiCom Program, May 2001.

[42] M.S. Phadke. *Quality Engineering Using Robust Design*. Prentice Hall, NJ, November 1989.

[43] MATLAB. *Mapping Toolbox User's Guide*. The MathWorks, Inc., 2 edition, 2004.

[44] G. Qian, R. Chellappa, and Q. Zheng. Robust structure from motion estimation using inertial data. *Journal of the Optical Society of America*, 18:2982–2997, 2001.

[45] S. I. Roumeliotis, A. E. Johnson, and J. F. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 4326–4333, May 2002.

[46] Jianbo Shi and Carlo Tomasi. Good features to track. *IEEE Conference on Computer Vision and Pattern Recognition*, June 1994.

[47] Ken Shoemake. Quaternion calculus for animation. *Math for SIGGRAPH*, 1991.

[48] Malcolm D. Shuster. A survey of attitude representations. *The Journal of the Astronautical Sciences*, 41(4):439–517, October–December 1993.

[49] Dennis Strelow and Sanjiv Singh. Optimal motion estimation from visual and inertial measurements. In *Workshop on Applications of Computer Vision*, Orlando, December 2002.

[50] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning and Autonomous Robots (joint issue)*, 31(5):1–25, 1998.

[51] Carlo Tomasi and Takeo Kanade. Shape and motion without depth. Technical Report CMU-CS-90-128, Carnegie Mellon University, May 1990.

[52] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[53] U.S. National Imagery and Mapping Agency. *Department of Defense World Geodetic System*, 3 edition, January 2000. TR8350.2.

[54] Eric W. Weisstein. Eric Weisstein's World of Mathematics.

9534 [illegible]