TAKING MAN OUT OF THE LOOP: METHODS TO REDUCE HUMAN INVOLVEMENT IN SEARCH AND SURVEILLANCE APPLICATIONS

A Dissertation

by

KEVIN MICHAEL BRINK

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2010

Major Subject: Electrical Engineering

TAKING MAN OUT OF THE LOOP: METHODS TO REDUCE HUMAN INVOLVEMENT IN SEARCH AND SURVEILLANCE APPLICATIONS

A Dissertation

by

KEVIN MICHAEL BRINK

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2010

Major Subject: Electrical Engineering

ABSTRACT

Taking Man Out of the Loop:

Methods to Reduce Human Involvement in Search and Surveillance Applications.

(December 2010)

Kevin Michael Brink, B.A., Mathematics, University of San Diego;

M.S., Mathematics, Texas A&M University

Co–Chairs of Advisory Committee: Dr. Shankar Bhattacharyya
Dr. John E. Hurtado

There has always been a desire to apply technology to human endeavors to increase a person's capabilities and reduce the numbers or skill level required of the people involved, or replace the people altogether. Three fundamental areas are investigated where technology can enable the reduction or removal of humans in complex tasks.

The first area of research is the rapid calibration of multiple camera systems when cameras share an overlapping field of view allowing for 3D computer vision applications. A simple method for the rapid calibration of such systems is introduced. The second area of research is the autonomous exploration of hallways or other urban-canyon environments in the absence of a global positions system (GPS) using only an inertial motion unit (IMU) and a monocular camera. Desired paths that generate accurate vehicle state estimates for simple ground vehicles are identified and the benefits of integrated estimation and control are investigated. It is demonstrated that considering estimation accuracy is essential to produce efficient guidance and control. The Schmidt-Kalman filter is applied to the vision-aided inertial navigation system in a novel manner, reducing the state vector size significantly. The final area of research is a decentralized swarm based approach to source localization using a high fidelity environment model to directly provide vehicle updates. The approach is an extension of a standard quadratic model that provides linear updates. The new approach leverages information from the higher-order terms of the environment model showing dramatic improvement over the standard method.

To my parents, my sister and Nina

ACKNOWLEDGMENTS

I would like to thank Dr. Hurtado for his help, patience and the nearly endless opportunities provided to me. I would also like to thank the rest of the committee, especially Dr. Bhattacharyya, for their help and guidance.

I would like to thank my friends at Sandia National Laboratories, especially Dr. Jeffery Carlson and Dr. David Wilson for their help, support, and friendship during my internships and after. Thanks goes to Dr. David Jeffcoat and Dr. Timothy Klausiutis at the AFRL for their efforts on my behalf and the opportunity they made available to me. And thanks to Dr. Andrey Soloviev and Dr. Adam Rutkowski for all the help and encouragement the last couple years.

I would like to thank all my friends who have stayed with me and the new ones I have met over the last five years. I would like to thank my family for their love and support. Finally, I would like to thank my wife, Nina, for the patience and grace she has shown me during this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

The goal of this research is to develop or advance technology that minimizes the human input required for search and surveillance applications. As such, this work is conducted in a broad setting, but focuses on three specific applications:

Well-calibrated multi-camera systems are used in many applications, such as security and human-robot interaction, to provide three-dimensional representations of changes within the shared camera views (3D computer vision) [20], [27], [28]. The calibration process is often a difficult and tedious task requiring complicated optimizations or knowledge of the true position of multiple calibration target points within the field of view. Simple optimization algorithms often require the use of large calibration objects or one has to accurately measure the world position of numerous feature points, which may require image correspondences have to be identified by hand. For simple targets where little or nothing is known of the target location, the optimizations tend to be much more complicated.

A simple, effective method is introduced which calibrates a system of cameras with no knowledge of the true calibration target positions and is compatible with a steepest decent optimization. This method is not intended to directly compete with standard camera calibration methods, but instead allows rapid deployment of camera systems in non-traditional settings and provides an easy method to use for those interested in applications of 3D computer vision and not the study of camera calibration and optimization specifically. The proposed method also allows for deployment with just two cameras, as well as calibration of camera systems where some individual

The journal model is *IEEE Transactions on Automatic Control.*

pairs of cameras do not share an overlapping field of view.

The second area of research, autonomous exploration of unknown environments, has become a hot-topic problem over the last decade. Specifically, there is a lot of interest in global positioning system (GPS) denied environments of hallways or alleyways using an unmanned ground vehicle (UGV) [12], [22], [33], [42]. The primary interest of this portion of research lies in the integration of estimation and control to provide efficient, accurate guidance. It has been well documented that additional accelerations, like S-turns, improve observability of the system and the accuracy of the state estimates. The extent these maneuvers benefit the system is investigated with an overarching goal to develop capabilities for autonomous optimization or selection of paths based on current state estimates.

The setting is an unknown hallway with unknown features visible to the camera. As a UGV progresses down the hallway, the unknown feature states are estimated and used to aid the inertial navigation system (INS) egostate (position, velocity, attitude) estimates in the absence of GPS. Unlike traditional simultaneous localization and mapping (SLAM) applications, specifically bearings-only, also known as vision SLAM (vSLAM), the filter used does not keep Euclidian coordinate estimates for the unknown features, but instead estimates initial range and bearing for each feature. The approach is referred to as the unit sphere observation model [43]. In the monocular camera case, bearing values are measured with a camera and the range to a feature is estimated using a synthetic stereo approach.

Using the unit sphere approach with a complementary form EKF allows for easy assimilation of additional sensors to the system and there is also the added benefit that a vehicle motion model is not required. Not requiring a motion model makes this framework applicable to any vehicle type, especially small and micro vehicles, whose motion models can to be unreliable. The unit sphere is a relatively unused approach to vSLAM and one consequence of the filter design is the initial bearing state estimates for a feature are the errors of direct measurements (the camera measures the bearings, which is directly fed into the filter, so the initial covariance value is the measurement uncertainty). This provides interesting possibilities for filter size reductions through the use of Schmidt-Kalman filter methods [41].

A thorough examination of the estimation performance as a function of path parameters is provided using sinusoidal and sawtooth paths with varied amplitudes and spatial periods in the open-loop setting. Closed-loop guidance is applied to the same prescribed paths and the introduction of turns in the hallway is also addressed. The advantages of integrated estimation and control is demonstrated, showing that "low cost" nominal paths can require more total control than some "higher cost" nominal paths in the closed-loop setting. An observability analysis of the six degree of freedom (6DoF) vSLAM problem is performed to provide insight into filter over-confidence and potentially path selection. Finally, drastic reductions in filter state size is achieved using the Schmidt-Kalman filter formulation, reducing the number of states per feature from three to just one.

The final area of research is the decentralized, swarm approach to source localization. This approach allows for low cost, unsophisticated individual robotic agents to perform a difficult task. Agents share position and sensor measurement information to develop polynomial environment models which are used to determine desired agent updates [19]. The standard quadratic environment model, linear update method is improved upon by allowing more accurate, arbitrary dimensional polynomial environment models. A Lagrangian expansion technique is used [1], which leverages the higher order model terms, to solve for agent position updates. Agent's updates are no longer simply in the direction of the model gradient, but are instead in the direction of the estimated minimum of the higher-order polynomial.

Significant improvement over standard quadratic methods is demonstrated using a fourth order environment model. Demonstrations of the higher order method are performed on a cubic function, the Rosenbrock Banana function and Matlab's Peaks function. The ability to locate iso-contours is also demonstrated. The method is developed in a generalized fashion and can be applied to 2D, 3D, or arbitrary dimensional spaces, drastically increasing the array of environments and uses for decentralized search applications.

CHAPTER II

SIMPLE EXTRINSIC SELF-CALIBRATION OF MULTI-CAMERA SYSTEMS

A. Introduction

The goal of camera self-calibration is to identify the extrinsic camera parameters (global position and orientation) of a camera system without knowledge of the world position of the calibration target points. Calibrated multi-camera systems have a wide range of potential uses as they allow for the 3D localization of changes occurring in shared image spaces (3D computer vision). However, due to the tedious nature of many calibration procedures, multi-camera systems are generally limited to fixed location, long-term applications. By removing the requirement for position knowledge of the calibration target points, there is potential for rapid deployment and therefore a broad expansion of plausible environments and applications for 3D computer vision.

There has be a lot of work in the related area of 3D computer vision [20], [27], [28]. A well-calibrated camera system is capable of identifying and displaying the volume elements, called voxels, currently experiencing motion, thus providing a 3D rendering of motion within a space. However, the focus of this research will be the calibration and not the uses of these camera systems. Significant research in camera calibration has been performed in the photogrammetry and computer vision communities. For multiple camera systems, especially those with wide baselines, methods involving 3D [13], [15] and 2D [10], [47], [52] targets for extrinsic calibration are not desirable. Although very accurate, the target objects are subject to observability and scalability issues, so other methods have been created to avoid these problems.

Methods using 1D targets that take advantage of the known distance between collinear points, or the rotation of collinear points about a fixed point have recently been introduced [26], [31], [54], [53]. These methods look very promising, and although more computationally complex, there is no scale ambiguity and the targets result in a relatively simple correspondence problem.

Point targets, i.e. 0D objects, have long been used [15], [30] but have always been susceptible to noise and scaling issues because the distance between targets is unknown. In the past, the point correspondence problem has relied on RANSAC or similar computationally complex methods. Lately, however, simple objects such as a penlight or laser pointer have been used to generate correspondences [17], [44] using synchronized video. With the correspondence issue greatly simplified, 0D methods have become more desirable once again.

The goal here is to develop a simple, practical self-calibration routine that is easily implementable using basic optimization methods applications in 3D computer vision. The focus is on developing a suitable optimization cost function for unknown, 0D target points and demonstrating its accuracy in simulation.

A very simple two step process is proposed where the intrinsic calibration (focal length, lens model, etc.) is performed beforehand, either in the lab or in the field. Target features are then recorded throughout the desired space and extrinsic camera parameters (position and orientation) are optimized, using the cost function, to best fit the collected target point data.

B. Camera Model and 3D Reprojection

The pinhole camera model and a least squares method for solving the world location of a feature point are presented. Three dimensional reprojection of corresponding points on two images from calibrated cameras is the primary component required for the proposed calibration method. The basic mathematical underpinnings for the pinhole model and 3D reprojections are provided here.

1. Pinhole Camera Model

A key element of camera work is the mapping of three dimensional world coordinates onto the two dimensional pixel coordinates. Generally, this mapping is nonlinear and rather complicated; however, it can be greatly simplified by making assumptions regarding the camera lens. For the cost function, (2.16), to be developed requires a ideal pinhole camera model assumption be made. A basic overview of the pinhole camera model is provided herein and some required nomenclature is presented in Table I. For a more thorough treatment, the reader is directed to [15]. See Fig. 1 for a graphical illustration of the camera mapping.

Table I. Camera Nomenclature

Notation	Definition
$\mathbf{P}_w = [X_w, Y_w, Z_w]^T$	Target feature location in the world coordinate frame
$\mathbf{P}_c = [X_c, Y_c, Z_c]^T$	Target feature location in the camera coordinate frame
$\mathbf{P}_s = [X_s, Y_s]^T$	Target feature location in the sensor coordinate frame
$\mathbf{P}_{pix} = [U, V]^T$	Target feature location in the pixel coordinate frame
$\mathbf{R}_j$	Orthonormal rotation matrix between world and camera coordinate frames
f	Focal length
i	Calibration target index
j	Camera index
$\mathbf{P}_j$	World coordinates of the jth camera

Mapping a 3D world coordinate to the associated pixel coordinate is a three step

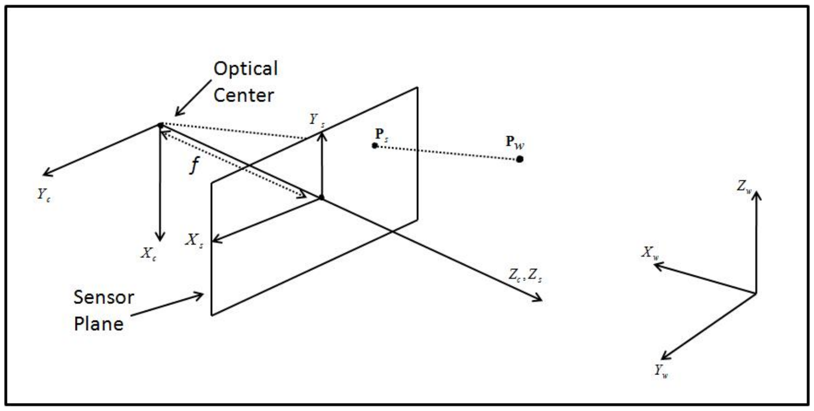


Fig. 1. Coordinates of a 2D imaging system.

process, see Fig. 2. First, one maps the global location to the camera frame, then to the sensor frame (the 2D location on the camera sensor), and finally to the pixel frame proving the corresponding pixel values.

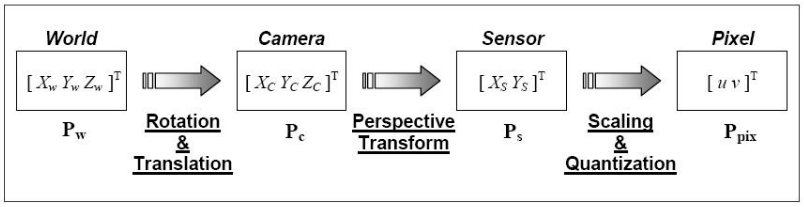


Fig. 2. Mapping from global position to pixel values (Henderson, 2006).

A standard transformation is used to take a point in world coordinate frame and map it to the camera coordinate frame of the jth camera.

$$\mathbf{P}_c^j = \mathbf{R}_j(\mathbf{P}_w - \mathbf{P}_j). \tag{2.1}$$

This text follows the conventions seen in camera model literature, so $\mathbf{P}$ represents a vector and $\mathbf{R}$ represents the rotation matrix.

A point defined in the camera coordinate frame, $\mathbf{P}_c^j$, can be mapped to a point on the sensor plane of the camera, $\mathbf{P}_s^j$, using similar triangle relations associated the pinhole camera model. Fig. 3 graphically illustrates the relationships.

$$\frac{X_s}{f} = \frac{X_c}{Z_c} \tag{2.2}$$

$$\frac{Y_s}{f} = \frac{Y_c}{Z_c} \tag{2.3}$$

The pixel value associated with the point $\mathbf{P}_s$ is dependent on the size of the sensor array as well as the number of pixels. The position on the sensor array is mapped to the pixel pair $[U, V]^T$ as

$$U = \left[\frac{N_w}{w} X_s\right] \tag{2.4}$$

$$V = \left[\frac{N_h}{h} Y_s\right] \tag{2.5}$$

where $[\cdot]$ denotes the nearest integer value. The values N_w and N_h are the number of pixels in the X_s and Y_s directions of the sensor plane respectively, where w is the width of the sensor, and h is the height. Assuming square pixels, and using (2.2) and (2.3), the pixel values can be rewritten as

$$U = \left[S \frac{X_c}{Z_c}\right] \tag{2.6}$$

$$V = \left[S \frac{Y_c}{Z_c}\right] \tag{2.7}$$

where $S = f\frac{N_w}{w} = f\frac{N_h}{h}$.

2. 3D Reprojection

The 3D reprojection of corresponding pixel values is not as simple as mapping a 3D point to pixel values because it is attempting to generate 3D information from 2D data. However, each pixel maps back to the sensor plane, and each location on the sensor plane is associated with the ray originating at the focal point and emanating

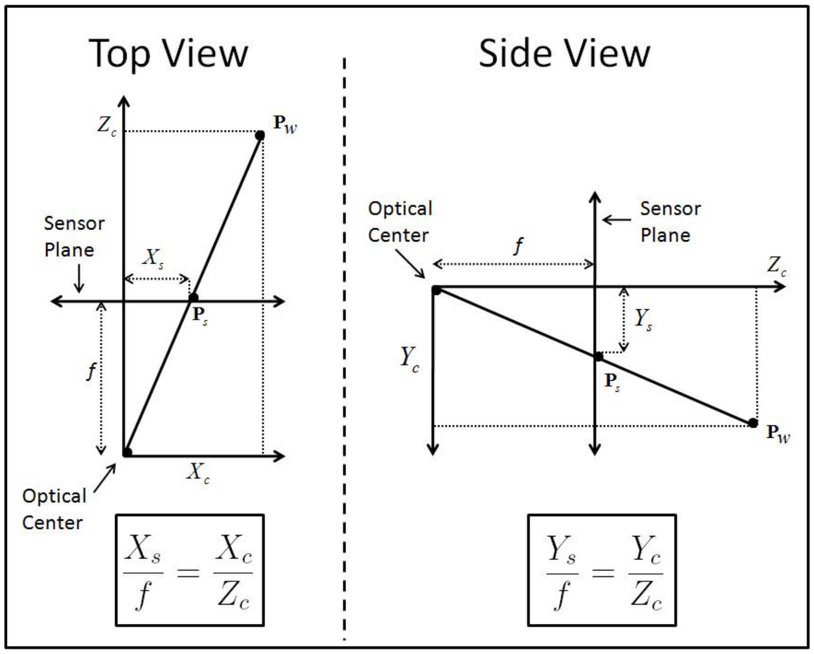


Fig. 3. Image plane relations.

out through the image plane. Any point on that ray will be mapped to the specific point on the sensor plane and thus the specific pixel values. Assuming two cameras see the same point in space, the associated rays (one from each camera) should intersect at that point in space. If the position and orientation of the two cameras are known, the pixel values can be used to generate rays and reproject the 3D location to their intersection. This section details the development of that 3D reprojection mapping.

Assuming two ideal pinhole cameras with known extrinsic parameters, this text follows the development seen in [17]. Starting with the standard equation (2.1), the rotation matrix of the jth camera is inverted.

$$\begin{bmatrix} X_i^W - X_j \\ Y_i^W - Y_j \\ Z_i^W - Z_j \end{bmatrix} = \mathbf{R}_j^{-1} \begin{bmatrix} X_C^{ij} \\ Y_C^{ij} \\ Z_C^{ij} \end{bmatrix} \tag{2.8}$$

Rewriting the previous equation in terms of scaled pixel position, $\frac{U_j^i}{S_j}$ and $\frac{V_j^i}{S_j}$,

$$\begin{bmatrix} X_i^W - X_j \\ Y_i^W - Y_j \\ Z_i^W - Z_j \end{bmatrix} = \mathbf{R}_j^{-1} \begin{bmatrix} Z_C^{ij}\frac{U_j^i}{S_j} \\ Z_C^{ij}\frac{V_j^i}{S_j} \\ Z_C^{ij} \end{bmatrix} \tag{2.9}$$

Expanding the above relation in terms of elements of $\mathbf{R}_j$, denoted r_{lm} for the lth row and mth column, achieves the following results by dropping the j notation and noting that $\mathbf{R}^{-1} = \mathbf{R}^T$.

$$\frac{X_i^W - X_j}{Z_i^W - Z_j} = \frac{Z_C^{ij}(r_{11}\frac{U_j^i}{S_j} + r_{12}\frac{V_j^i}{S_j} + r_{13})}{r_{31}\frac{U_j^i}{S_j} + r_{32}\frac{V_j^i}{S_j} + r_{33}} \equiv A_j^i \tag{2.10}$$

$$\frac{X_i^W - X_j}{Z_i^W - Z_j} = \frac{Z_C^{ij}(r_{21}\frac{U_j^i}{S_j} + r_{22}\frac{V_j^i}{S_j} + r_{23})}{r_{31}\frac{U_j^i}{S_j} + r_{32}\frac{V_j^i}{S_j} + r_{33}} \equiv B_j^i \tag{2.11}$$

The process is repeated for the kth camera to arrive at A_k^i and B_k^i. Rearranging equations (2.10) and (2.11) results in a matrix equation that facilitates a least squares solution to the projected world coordinates of the ith target.

$$\begin{bmatrix} 1 & 0 & -A_j^i \\ 0 & 1 & -B_j^i \\ 1 & 0 & -A_k^i \\ 0 & 1 & -B_k^i \end{bmatrix} \begin{bmatrix} X_i^W \\ Y_i^W \\ Z_i^W \end{bmatrix} = \begin{bmatrix} X_j - A_j^i Z_j \\ Y_j - B_j^i Z_j \\ X_k - A_k^i Z_k \\ Y_k - B_k^i Z_k \end{bmatrix} \tag{2.12}$$

The previous equation is in the form of $\mathbf{Ax} = \mathbf{b}$ with known 4×3 matrix $\mathbf{A}$ and 4×1 vector $\mathbf{b}$ and unknown 3×1 vector $\mathbf{x}$. Eq. (2.12) can now be solved providing a least squares approximation, $\tilde{\mathbf{x}} = \tilde{\mathbf{P}}_{jk}^i$. Note, that for prefect reprojection the vectors extending from each image plane would intersect. This is not generally the case, so the least squares solution locates the midpoint between the vectors at there closest point, as seen in Fig. 4.

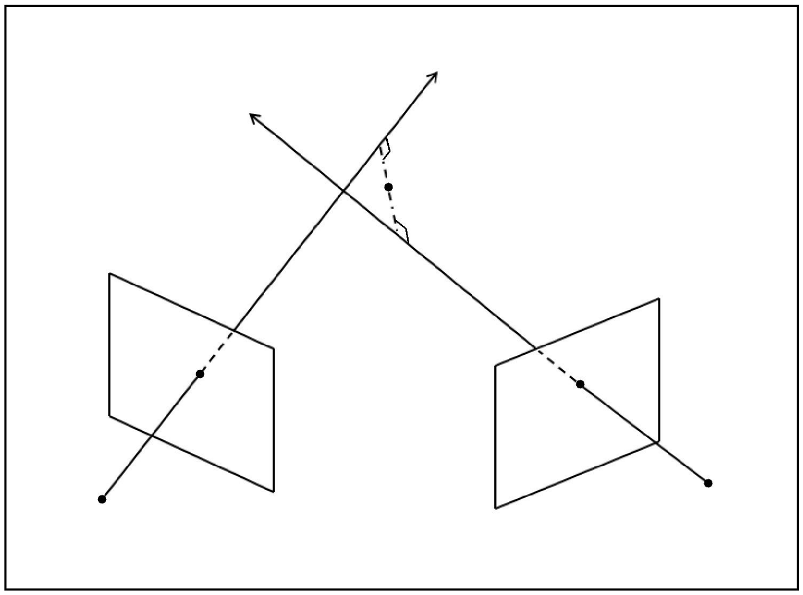


Fig. 4. 3D Reprojection.

C. Intrinsic Calibrations

In reality, cameras are not perfectly represented by the pinhole model. Even if the focal length and sensor size is known, there is generally lens distortion that causes target points to map to unexpected pixel values. A user must characterize this distortion in order to map true pixel values seen by a camera to the ideal pinhole pixel value required to perform the extrinsic calibration. Caltech's Matlab Camera Toolbox, [2], offers a five-parameter model for camera distortion with three radial distortion parameters (b_1, b_2, and b_3), and two tangential distortion parameters (t_1 and t_2).

If the distortion parameters are known, the distorted pixel values, U_d and V_d, can be mapped to the undistorted or corrected pixels U_u and V_u.

$$\begin{bmatrix} U_u \\ V_u \end{bmatrix} = [1 + b_1 r^2 + b_2 r^4 + b_3 r^6] \begin{bmatrix} U_d \\ V_d \end{bmatrix} + D_{tan} \tag{2.13}$$

where $r^2 = U_d^2 + V_d^2$ and

$$D_{tan} = \begin{bmatrix} 2t_1 U_d V_d + t_2(r^2 + 2U_d^2) \\ t_1(r^2 + 2V_d^2) + 2t_2 U_d V_d \end{bmatrix} \tag{2.14}$$

The required intrinsic parameters can be identified with [2], allowing the user to rectify images or individual pixel values resulting in undistorted data. An example is shown in Fig. 5.

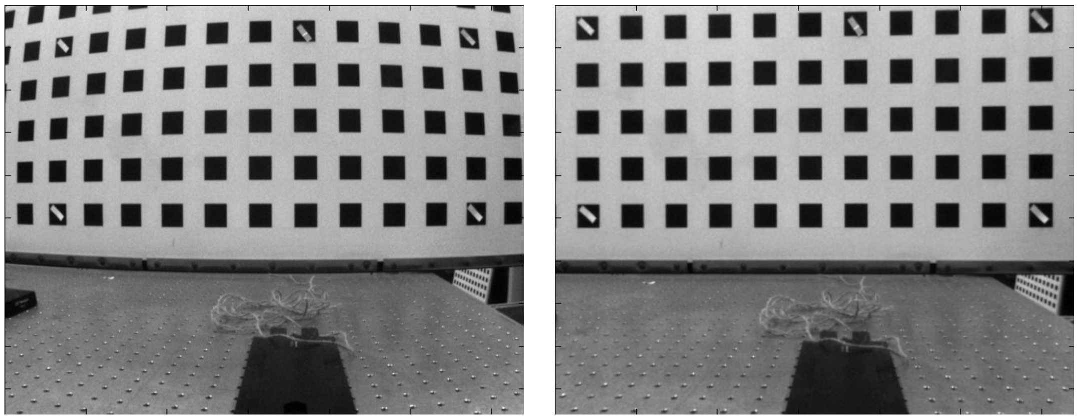


Fig. 5. Left: Distorted image. Right: Rectified image.

D. Extrinsic Calibrations

The goal here is to develop a self-calibration method that identifies the unknown extrinsic camera parameters and does so without knowledge of the true world positions of the calibration targets. To do so, the “fitness” of the 3D reprojections are evaluated. For the ith target and the jth and kth cameras, the 3D reprojection is evaluated via the error function, $\mathbf{e} = \mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}$. From (2.12) the error function is formally written as

$$\mathbf{e}_i^{jk} = \begin{bmatrix} 1 & 0 & -A_j^i \\ 0 & 1 & -B_j^i \\ 1 & 0 & -A_k^i \\ 0 & 1 & -B_k^i \end{bmatrix} \begin{bmatrix} \tilde{X}_i^{jk} \\ \tilde{Y}_i^{jk} \\ \tilde{Z}_i^{jk} \end{bmatrix} - \begin{bmatrix} X_j - A_j^i Z_j \\ Y_j - B_j^i Z_j \\ X_k - A_k^i Z_k \\ Y_k - B_k^i Z_k \end{bmatrix} \tag{2.15}$$

If the camera parameters are known perfectly, and neglecting pixel noise, then $\mathbf{e}_i^{jk}$ would equal zero. Although $\mathbf{e}_i^{jk}$ lacks a physical meaning, intuitively, the smaller its value the better the camera parameters fit the pixel data. Therefore the following is to be minimized

$$J = \sum_i \sum_{j \neq k} [\mathbf{e}_i^{jk}]^T [\mathbf{e}_i^{jk}] \tag{2.16}$$

Eq.2.16 will be refered to as the “reprojection fitness” (RF) cost function which is a direct adaptation of the “inverse” cost function seen in [17]. The inverse cost function attempts to minimize the variance of the 3D reprojections whereas the RF cost function minimizes the numerical error associated with the least squares solutions. The assumption is made that, given the correct calibration of the system, each pair of cameras will return the same world location of a target. Also $\mathbf{e}_i^{jk} = 0$ if the ith target isn't visible to either the jth or kth cameras.

This approach is desirable for a number of reasons. Unlike many calibration techniques with unknown feature positions, there is no need to optimize over the N target world positions. This reduces the dimensionality of the classic calibration problem by $3N$ parameters. Additionally, unlike most calibration methods, including [17], the RF cost function only requires two cameras and is less computationally complex than the inverse function.

The RF cost function is also versatile in that for larger camera systems, specific camera pairs do not need to share a common field of view. As a consequence of only requiring two cameras, there is no limitation to only optimize over features seen by three or more cameras. This increases (drastically in the three camera case) the physical space where calibration points can be collected, leading to a broader sample set of target points available for calibration.

E. Implementation

Given the RF cost function (2.16), some practical implementation details are provided here. Some of these details are not required for numerical simulation but they will be needed for the implementation of a real system. It is assumed that intrinsic camera parameters are already calibrated, which can be done beforehand in a lab or out in the field.

1. Extrinsic Calibration Target

Robust point correspondences are required for the RF cost function because outliers can drastically affect the solutions. This can be achieved using RANSAC or other correspondence algorithms, but the preferred method is to take advantage of video data and use a single, 0D target such as a penlight [44]. In an indoor setting a glow in the dark target can often work well. With color cameras there are many additional possible solutions because a uniquely colored object could be used then located in the image.

Given a single 0D target and time-synched cameras, the user can easily retrieve corresponding pixel values for the target at multiple locations. The centroided pixel values associated with the target object should be taken to provide a sub-pixel accuracy. For automated image processing to collect pixel correspondences, the type of camera, lens, and environment need to be considered when selecting a target object.

In the case of manual pixel identification, the target object is of much less importance, but a larger number of target locations is recommended to compensate for the decreased accuracy, which is generally no longer sub-pixel and sometimes biased by the user.

2. Optimization Initial Parameter Guesses

The rotation matrix, $\mathbf{R}_j$, has many characterizations. For ease of use a 3-2-1 Euler angle set is chosen [40], as seen in (2.17). That is, given the Euler angle set $[\alpha, \beta, \gamma]$, the camera is first rotated about the z-axis by γ, then the y-axis by β, and finally the x-axis by α. Note that the work is performed in a right-handed coordinate system and a positive rotation is in the counter-clockwise direction, Fig. 6. Any Euler angle set will work, in fact any other parametrization of $\mathbf{R}_j$ will work; however, the 3-2-1 set allows the user to make very accurate initial guesses for the rotation parameters which increases the likelihood of correctly identifying the global minima of the RF cost function.

$$\mathbf{R}_j = \begin{bmatrix} \cos\beta\cos\alpha & \cos\beta\sin\alpha & -\sin\beta \\ \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha & \sin\gamma\sin\beta\sin\alpha - \cos\gamma\cos\alpha & \sin\gamma\sin\beta \\ \sin\gamma\cos\beta\cos\alpha - \sin\gamma\sin\alpha & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha & \cos\gamma\sin\beta \end{bmatrix} \tag{2.17}$$

Once pixel data has been collected and rectified for a reasonable number of target points, and initial guesses are made for the camera parameters and the RF cost function can then be applied.

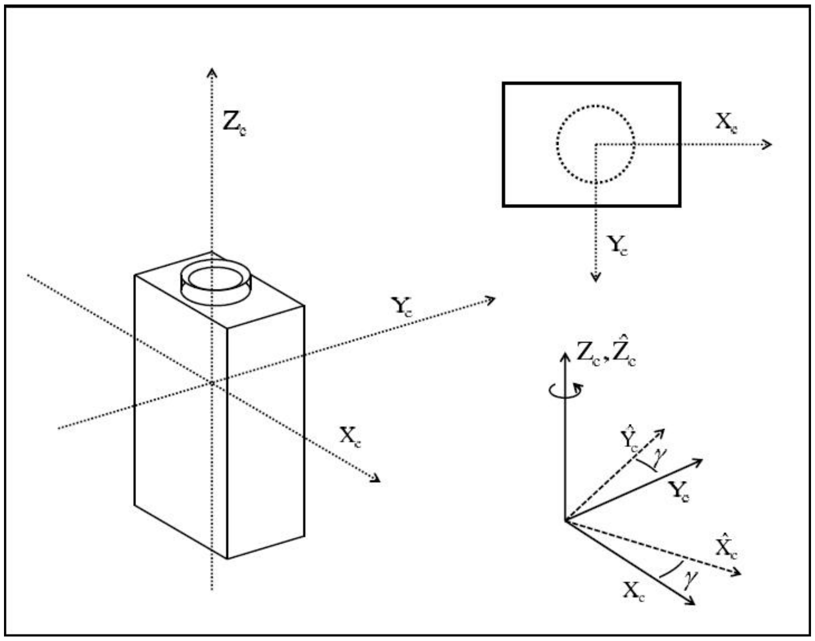


Fig. 6. Camera frame description, the upper right camera is looking into the page.

F. Numerical Simulation

Numerical simulations are performed on potential camera arrangements to evaluate the RF cost function performance. The RF cost function becomes ill-posed for certain camera orientations and/or target sets, but is demonstrated to perform quite well in many useful settings. A simple gradient descent optimization is used to demonstrate the RF cost function accuracy and its functionality. For the purposes of this optimization, the Euler angles are converted to Modified Rodriguez Parameters, (MRPs) [40]. This is done because MRPs have an extended linear range which improves the conditioning of the cost function and allows larger updates; the 3-2-1 Euler angles are much more user friendly in terms of estimating parameters.

The camera positions are not included in the optimization because a gradient search is ill-suited to handle the scaling differences between camera position and orientation parameters. Assuming the camera positions have been measured or are otherwise known to a reasonable degree, simulations show that rather large position errors are tolerated quite well. Conceptually, and often practically, there is the option of using the cameras to define the world coordinate frame. In this case the exact location of the first camera is known because it is chosen. The user can also define the space so one axis extends from the first camera through the second camera (or define a specific offset); thus, the user can arbitrarily assign two of the second camera's location components, only considering measurement error along a single axis. For the third camera, the user is able to assign one component arbitrarily, this defines a plane between all three cameras, the other two location components must be measured. For any additional cameras, errors must be considered in all three position components.

Errors in the camera position estimates (measurements) are generated in simulation as uniform random variables with values in the range ± 6 inches for each required axis. For initial guesses of camera orientation parameters, the truth plus normal errors with means of 10 degrees for each parameter are used. Standard black and white cameras, with 640×480 resolution on a 1/2" sensor and a focal length of 2.8 mm, are modeled for the simulations.

1. A Small Room With Three Cameras

Very often the 3D calibration of camera systems is used to facilitate the 3D rendering of movement in a confined area. This may be for security purposes, human robot interaction, or any other number of applications. To demonstrate the effectiveness of the cost function, the placement of three cameras around a 20ft by 20ft room is simulated. This could also represent three cameras set on tripods in an outdoor setting. A representation of the simulated camera setup is shown in Fig. 7. True camera parameters are listed in Table II. For the simulations, 150 targets points are generated as a set of 3D uniform random variables that run from floor to ceiling, 0ft to 8ft, and wall to wall, to within two feet of the walls. The true camera parameters are used to generate the true pixel positions with random measurement noise added. The noise is normal with a mean of one pixel, which is also quite large.

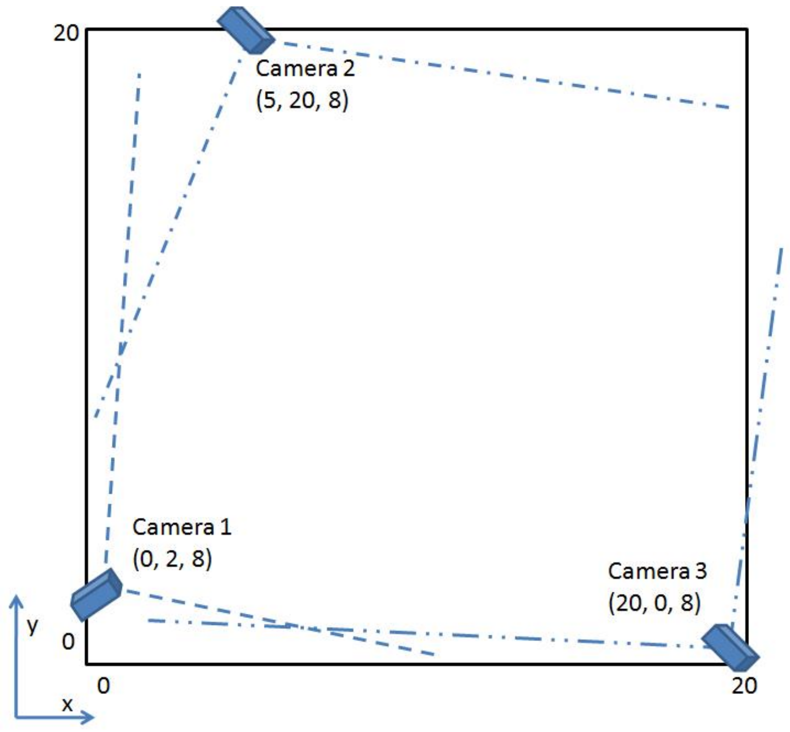


Fig. 7. Depiction of camera configuration used in simulations.

The simulated camera views of the target points can be seen in Fig. 8, where the boxed region represents the cameras' fields of view. Each 'x' represents a specific camera's view of a target point. The points outside of the boxed region represent target points not seen by the corresponding camera; however, if the other cameras have the target in view, it is still used in the optimization.

The initial guesses for camera parameter values for this example can be seen in Table III. Those initial values are used to generate a 3D reprojection, Fig. 9, where the 3D reprojections are not representative of the true system with estimated 3D

Table II. True Extrinsic Camera Parameters

Camera	X_j	Y_j	Z_j	α_j	β_j	γ_j
1	0.0	2.0	8.0	-125.0	0.0	-55.0
2	5.0	20.0	8.0	-115.0	0.0	-145.0
3	20.0	0.0	8.0	-120.0	0.0	30.0

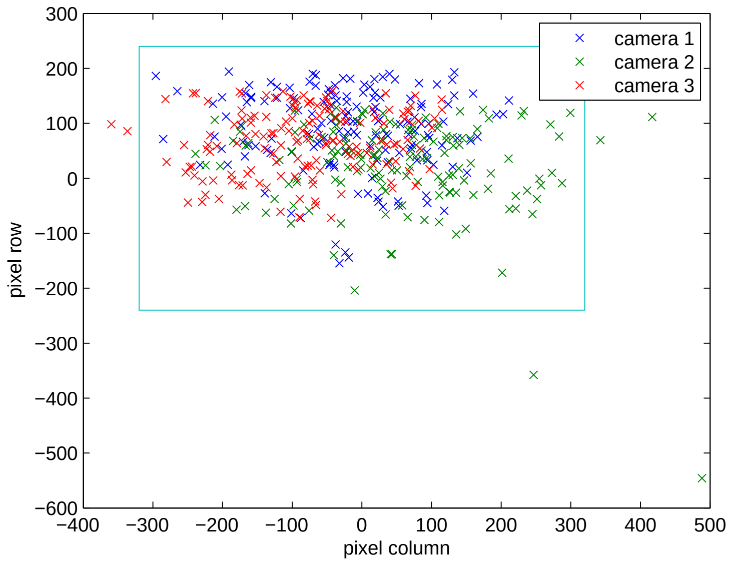


Fig. 8. Camera views of target points.

locations spread well beyond the 20ft×20ft grid. Little to no correspondence between the 3D reprojections of the same target points from different camera pairs (ideally each camera pair reprojects the same target point back to the same 3D location).

After minimizing the RF cost function, (2.16), with respect to the camera orientations, the correspondences between the 3D reprojections of the target points by each camera pair are clearly seen, Fig. 10. The reprojections are well within the 20ft×20ft grid, and each camera pair is producing very similar reprojections. Note that some 'x' values are by themselves, particularly at the edges, this is because only

Table III. Initial Guess of Extrinsic Camera Parameters

Camera	X_j	Y_j	Z_j	α_j	β_j	γ_j
1	0.0	2.0	8.0	-115.2084	4.7041	-54.5367
2	5.4345	20.0	8.0	-104.7321	-2.1340	-149.7814
3	20.3516	0.3247	8.0	-121.9071	-5.6609	37.3890

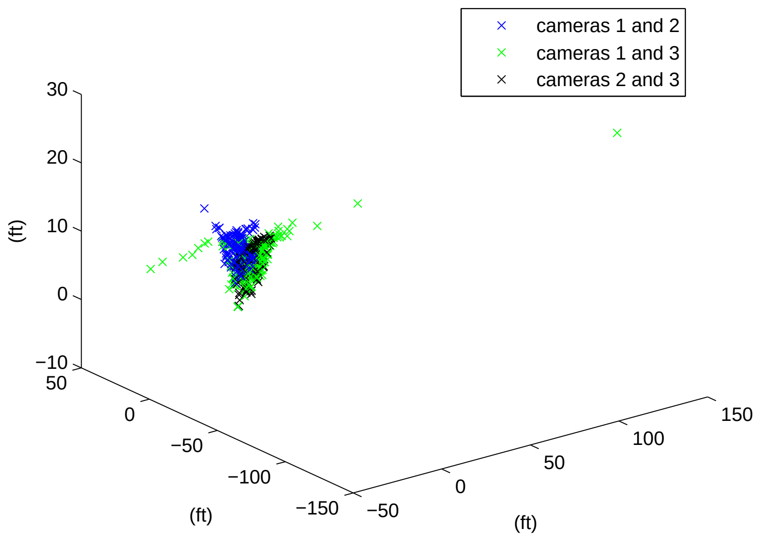


Fig. 9. 3D reprojection of target points using initial guesses for camera parameters.

two cameras see the target point, so only one 3D reprojection can be produced. The final parameter estimates can be seen in Table IV.

The resulting parameter estimates are within 2.0 degrees of the truth on all angles, which would be a decidedly good optimization when considering the noise involved. Note that since the camera positions are fixed, and wrong, some of the error in the angles may actually be corrective, allowing the reprojection points to come closer to the truth than if the angles perfectly matched the true orientation values.

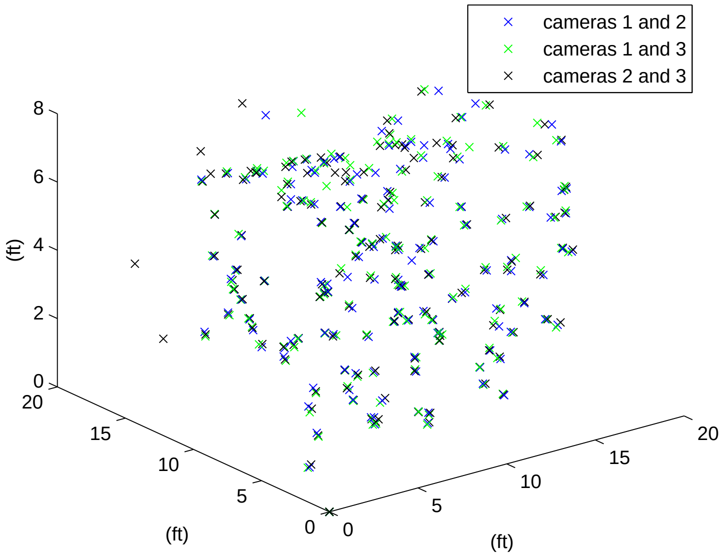


Fig. 10. 3D reprojection of target points using optimized camera parameters.

Table IV. Three Camera Self-calibration Results and Associated Error

Camera	α_j	β_j	γ_j	$\Delta\alpha_j$	$\Delta\beta_j$	$\Delta\gamma_j$
1	-124.7312	-0.1537	-55.9048	0.2688	-0.1537	-0.9048
2	-114.8297	-0.4086	-143.1626	0.1703	-0.4086	1.8374
3	-119.7610	0.3051	30.7038	-0.2390	0.3051	0.70386

While it is important in many applications to reproject points back to the true 3D position, it may be even more important for a given 3D point in space that all the camera pair reprojections are consistent. The largest discrepancy between reprojections and the variance of reprojections are both good measures of consistency. The constancy of the reprojections is addressed with a set of test targets, not used in calibration, which range from 4ft to 16ft in the x and y directions placed at one foot intervals. Along the z-axis, targets are placed every foot between 1ft and 6ft.

Using the previously solved parameters, the 3D reprojections of each test target is very consistent, seen in Fig. 11, and the maximum discrepancy between reprojections for each target features is rather small with the vast majority under six inches, seen in Fig. 12. The average error falls in the range of four inches or less. The majority of the larger errors are test points near the ceiling or towards the outside edges, as expected, and some are due to unusually large pixel noise for a particular point. Finally the reprojection variance can be seen in Fig. 13. Overall these figures show consistent 3D reprojection results for each camera pair and suggest the results could be used for 3D computer vision applications.

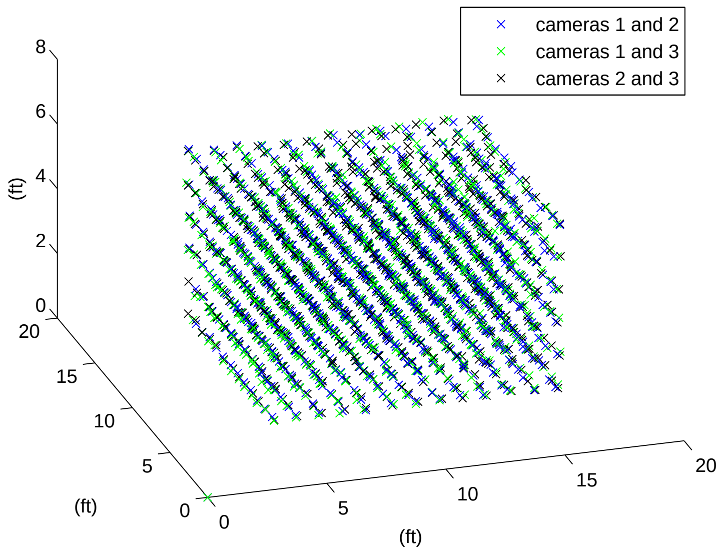


Fig. 11. 3D reprojections of test targets using optimized camera parameters.

If the camera positions are known quite well, to within three inches (quite feasible for an indoor space), and a 0.25 mean error in pixel measurements is achieved, the results are improved and the smaller maximum discrepancies can be seen in Fig. 14. In this case, the room could be very finely partitioned for high resolution 3D computer vision applications.

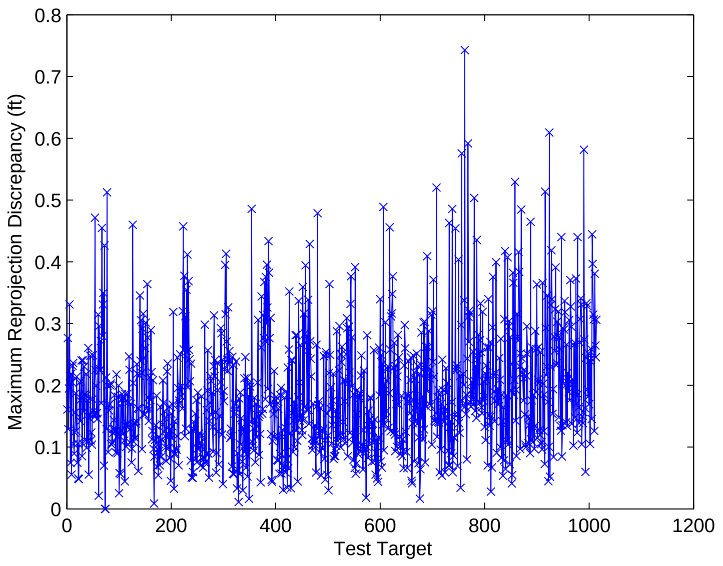


Fig. 12. Maximum discrepancy between 3D reprojections of test targets.

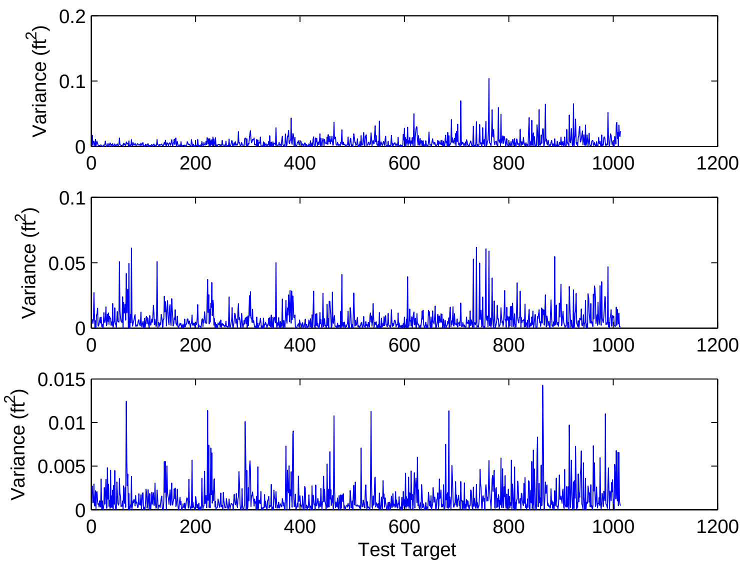


Fig. 13. Variance of 3D reprojections of test targets.

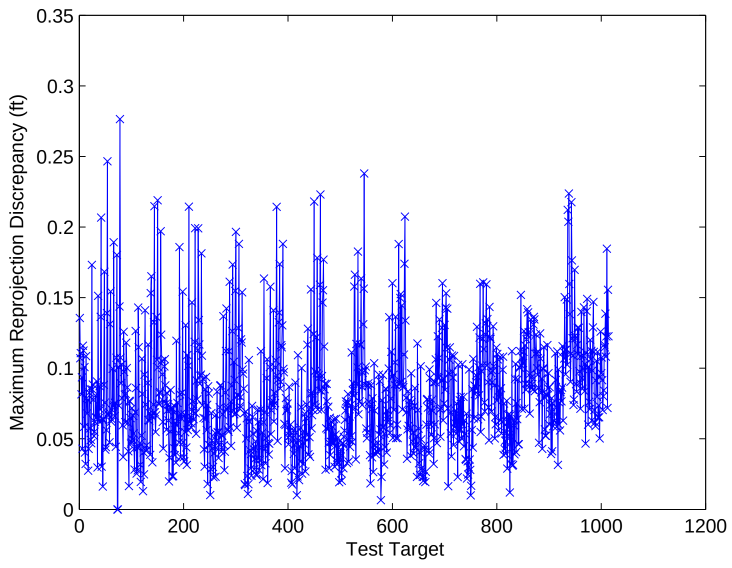


Fig. 14. Maximum discrepancy between 3D reprojections of test targets for improved camera position estimates.

The previous results presented were from a single test run, so several more simulations are needed to characterize the performance. The presented configuration was used and the process was repeated 100 times. This included generating new target points, new noisy pixel data, and new initial guess for camera parameters (including the error to the assigned camera positions). Out of 100 runs, 76 converge to results very similar to what is seen in the example, while 24 runs did not result in usable parameter sets.

The five worst results from the 24 simulations which did not converge to the truth were selected and 100 additional simulations were run on each. Using the same values as the non-converging run, only the initial parameter guesses are perturbed. Starting with the original, non-convergent initial guesses, a normal random variable of mean zero and standard deviation of 10 degrees is added to each camera angle for each of the additional 100 runs. For all five cases, the additional 100 runs achieved convergence between 64 and 78 times using the newly perturbed initial parameter guesses.

Converging about 75% of the time does not seem impressive. However, this implies that out of 100 feature point distributions and initial guesses of camera parameters, the first attempt was convergent in 75% of cases. The feature distributions that did not converge with the original initial guesses where found to be convergent after perturbations of the original initial guesses. In other words, the system is in general convergent for a set of feature points, even if the initial optimization attempt is not.

2. A Small Room With Two Cameras

Using the same setup as the three camera example, shown in Fig. 7, the third camera is dropped from the system and the same optimization procedure is preformed again.

The main drawback with calibrating just two cameras is the lack of a defined coordinate system. Because the target points are not assigned a global position, it is always possible to transform the system by simply rotating about the vector passing through the two cameras. Assuming the calibration targets are also transformed by this rotation the camera views would not change. Since the target positions are not known, the cameras may in fact find a solution which is accurate up to the described rotation transformation. The spatial relations between true 3D values and 3D reprojections will be accurate, but the 3D reprojections will be shifted by the transformation with respect to the global frame.

Given a unit vector $\hat{\mathbf{e}}$, the rotation matrix about $\hat{\mathbf{e}}$ by and angle of ϕ can be defined as

$$\mathbf{C}(\phi) = \mathbf{I} + (1 - \cos(\phi))[\times\hat{\mathbf{e}}][\times\hat{\mathbf{e}}] - \sin(\phi)[\times\hat{\mathbf{e}}] \tag{2.18}$$

where $[\times\hat{\mathbf{e}}]$ is the skew-symetric matrix of $\hat{\mathbf{e}}$ [40]. The vector of rotation, $\hat{\mathbf{e}}$, associated with the two camera example is the normalized vector pointing from camera 1 to camera 2.

$$\hat{\mathbf{e}} = \frac{\mathbf{x_2} - \mathbf{x_1}}{|\mathbf{x_2} - \mathbf{x_1}|} \tag{2.19}$$

In order to rotate the 3D reprojections about the appropriate vector, they must first be written with respect to $\mathbf{x_1}$ as the origin (otherwise points are not rotating about the vector connecting $\mathbf{x_1}$ to $\mathbf{x_2}$, but rather the same vector direction except emanating from the origin). $\mathbf{C}(\phi)$ is applied to the translated 3D positions and then the rotated points are translated back with respect to the true origin. The points $\mathbf{x_1}$ and $\mathbf{x_2}$, and in fact everything on the vector connecting them, will be invariant to the transformation as they lie on the vector of rotation, but every point lying off that vector will be mapped to a new location. The value of ϕ for any given optimization will be arbitrary based on initial errors of the system and must be adjusted through trial and error until correspondence is obtained (i.e. test points near the floor are level, or some other metric is achieved).

Fig. 15 shows the true location of test points, the initial 3D reprojection, and the transformed 3D reprojections. While the initial 3D reprojection captures the essence of the system, the coordinate frame is clearly incorrect and can be seen to have rotated about $\hat{\mathbf{e}}$. However, the transformation procedure rotates the reprojections back about $\hat{\mathbf{e}}$ and brings the reprojected points into close proximity of the true values. 3D reprojection discrepancies and variance cannot be produced because there is only one reprojection for just two cameras, but the close alignment with the true test point positions suggests accurate 3D renderings is possible for a two cameras system.

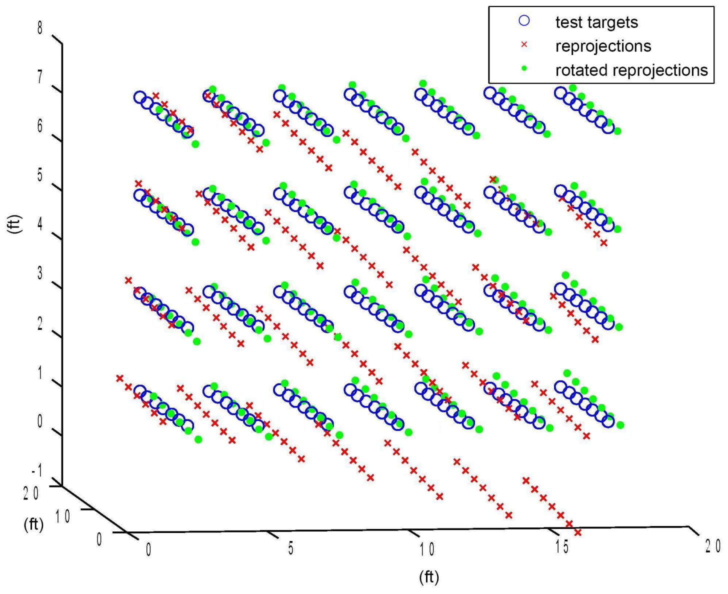


Fig. 15. Test targets, blue. Original 3D reprojection of test targets containing rotation error, red. Corrected 3D reprojection of test targets, green.

G. Comparison to Variance Based Inverse Cost Function

The reprojection fitness cost function and the work in this chapter is an extension of work done in [17], where "forward" and "inverse" cost functions were developed and tested. Specifically the inverse cost function that minimized the square of the variance of the 3D reprojections was adapted. This section compares the inverse cost function with the RF cost function.

The inverse cost function was successful in 62 of 100 simulations using the three camera setup, where the RF cost function was successful in 76 of the 100 runs. Similar to the RF cost function results, additional perturbations of the nonconvergent initial guesses generally resulted in convergent estimates for the inverse cost function, but for a smaller fraction of runs. Most notably, the inverse cost function suffers from less consistent final estimates and larger reprojection discrepancies due to the comparatively shallow nature of the cost function minimum.

For each camera, three MRPs are used to define the rotation matrix for optimization purposes. Using a random set of calibration points and sampling both cost functions, seven of nine MRPs are set to the true values and the other two are varied around the true value. Local cost function mappings are generated. Displaying the cost function evaluation as a function of error size in the two varied parameters (one from camera one, the other from camera two), surface and contour plots can be seen in Figs. 16 and 17.

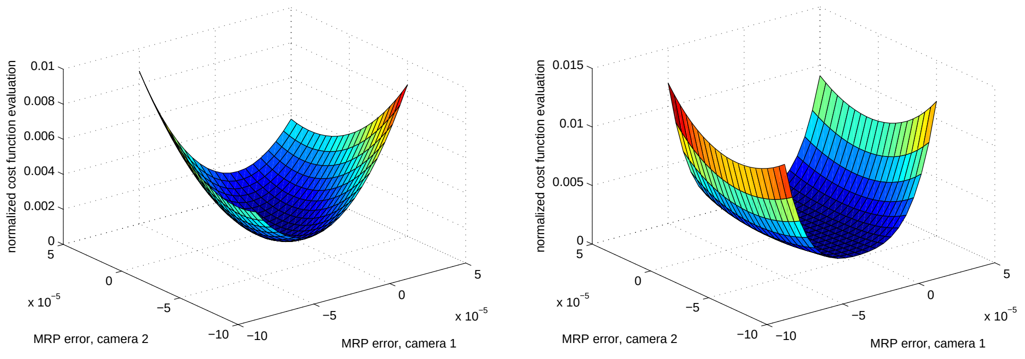


Fig. 16. Left: Normalized RF cost function for two varied camera parameters. Right: Normalized inverse cost function for two varied camera parameters.

Clearly shown, the RF cost function is much sharper, especially in the MRP parameter for camera two, than the inverse cost function. This is consistent with the increased convergence rate of the RF cost function and the tighter grouping of final estimates over several runs. This added sharpness not only generates more consistent results, but faster convergence as well. The results shown are representative of additional pairings of the nine total parameters showing that the inverse function is less sensitive to changes in certain parameters. Thus, the RF cost function is better suited to simple gradient based optimizations.

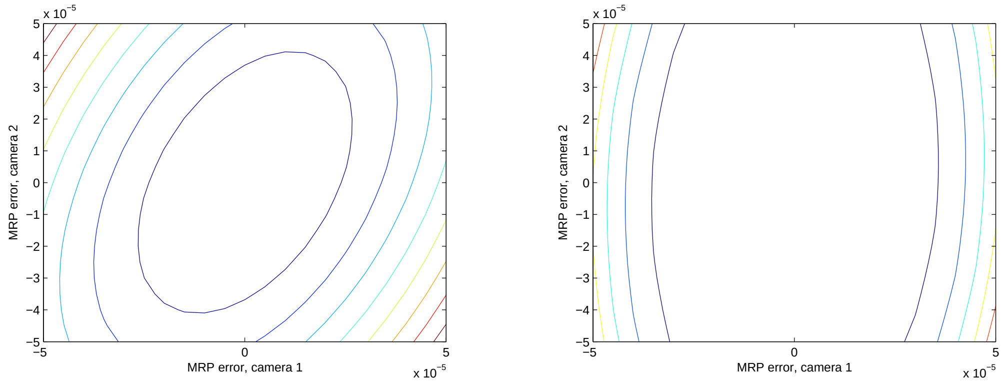


Fig. 17. Left: RF cost function contour lines for two varied camera parameters. Right: Inverse cost function contour lines for two varied camera parameters.

H. Summary

A simple, straightforward method for the extrinsic calibration of multi-camera systems has been presented. The method shown allows for the rapid deployment of cameras, quick collection of calibration target data, and accurate estimation of camera orientations with just basic camera and optimization knowledge. The RF cost function was shown to be an improvement from the inverse cost function approach.

Overall the RF cost function is superior to the inverse cost function in several facets; it only requires two cameras total, for additional cameras it only requires that two see a given target point, and the cost function is better suited to gradient based approaches. A two-camera system calibration was demonstrated while the three-camera system results showed strong calibration accuracy even for crude position measurements and noisy pixel measurements using a low resolution camera.

CHAPTER III

VISION-AIDED INERTIAL NAVIGATION

A. Introduction

The benefits of tightly integrating inertial navigation sensors with global positioning systems (GPS) are well known [4]. The use of an inertial measurement unit (IMU) along with GPS is effective because the sensors complement one another; the IMU provides a high frequency feedback, whereas the GPS provides a slower more stable benchmark signal. This allows a Kalman filter, or something similar, to process the IMU data at a high rate with intermittent corrective updates based on GPS data resulting in a high precision inertial navigation systems (INS).

Currently, GPS/IMU systems are the standard method for accurate INS capabilities. However, GPS signals are not always available and this leaves the INS vulnerable. And because even the best IMUs experience gyro drift and accelerometer biases, an INS running with an IMU alone will experience unstable egostate (position, velocity and attitude) estimates. For any sort of autopilot the vehicle will drift well off its intended course during any prolonged GPS absence. Often a human operator can control the vehicle until GPS is again available, but in the case of an unmanned air or ground vehicle (UAV/UGV) that is not the case.

Due to the vulnerability of INS to GPS outages, there is significant interest in the aerospace and defence communities to develop autonomous systems that can navigate effectively in the absence of GPS [3], [12], [45]. GPS may be unavailable due to location such as in an "urban canyon" environment of city streets or inside buildings. Or, it may be denied due to malicious jamming or spoofing of the signal where falsified information is transmitted to disrupt or fool the GPS [9], [50].

Although navigation systems have been developed that use just two GPS satellites [8] when four or more are required for a standard GPS system, an INS will be more robust and more widely applicable if it can function without any GPS signal at all. This counteracts the jamming/spoofing threat while also providing guidance in environments were signals are simply not available. Several approaches have been developed including the use of laser scanners [22], stereo vision [42], [45], known features, odometry and monocular cameras [33].

This research focuses on navigating within unknown corridors, such as a hallway or other "urban canyon"-like environment, using a vision-aided inertial navigation approach with a single monocular camera. It is assumed that there is no known map of the environment and the location of features seen by the camera is unknown. In this case the geometry of features can not be used in a GPS-like (globally defined) manner to solve for camera and vehicle position and orientation.

The single monocular camera approach was chosen because a single camera is scalable in ways stereo pairs and laser range finders are not. Range determination from a stereo pair of cameras is limited by the baseline separation between the cameras, and on certain vehicles, a large separation may not be possible. Furthermore, laser range finders have limited range; those with larger ranges typically have higher power requirements and higher mass, that may be a limiting factor in some applications. Moreover, if a bearings-only INS estimator is able to provide suitable solutions then the results would only be improved upon if the vehicle platform is capable of supporting additional sensors.

Vision-aided inertial navigation is effectively a simultaneous localization and mapping (SLAM) application where a range measuring device is not used. The term vision SLAM (vSLAM) is used to reference the camera as the only additional sensor aiding the IMU. The primary concern of this research will be the vehicle egostate estimates and not the feature map itself and at times the standard capability of SLAM and vSLAM to recognize previously seen features may not be required, thus making the application a visual odometry method.

Inspiration is taken from the success of outdoor vSLAM applications that either operate with just their INS and cameras when GPS is not available [25], or operate without GPS altogether [5], [29]. Like other vSLAM applications, a complementary form EKF is used [4], [5], [24]; However, in this research a unit sphere based observation model is used [43]. The filter keeps estimates of each feature's position in spherical coordinates as opposed to typical Euclidean coordinates in most filters. The unit-sphere based observation model leverages geometric constraints, as opposed to the traditional SLAM and vSLAM methods of comparing estimated and measured 3D position (SLAM) or estimated and measured pixel values (vSLAM). The unit-sphere approach was developed for easy assimilation of multiple sensors, referred to as multi-sensor fusion, and although only a single camera is used in this research, in related efforts there is a desire for a generic and flexible filter structure.

The vision-aided inertial navigation research to be presented herein concerns itself with identifying and understanding which vehicle paths lead to better estimates, the integration of estimation and control, and the reduction of filer size required for such INS systems. This research is conducted in simulation, where results will be applied to hardware implementations occurring in related efforts.

It is well established that vehicle motion in a camera-aided system is paramount to producing accurate estimation. The observability of the system is improved due to additional accelerations such as S-turn maneuvers, and improved observability leads to more accurate estimates. This area of research takes a specific scenario of a UGV with a forward facing monocular camera moving down a hallway with previously unknown feature points disbursed along the walls. Specific path types, sinusoidal and sawtooth, are simulated and estimation accuracy results are presented. Simulations are conducted with open-loop control using prescribed paths to look at path effacy. Simulations are also done with closed-loop control where egostate estimates are used by the guidance system attempting to follow a desired path.

After extensive simulation of paths with varied characteristics, clear preferences are shown for a specific scenario. However, if the scenario is changed, either due to feature availability or equipment changes, the resulting performance of a given path may also vary. The near-term goal of this research is to identify trajectories for a given scenario that, on average, produce accurate estimation at an acceptable control cost. The long-term goal is to develop methods to properly select effective trajectories in real-time regardless of the scenario.

It will be shown in future chapters that the covariance, a top candidate for the real-time judging of estimator performance, is unreliable. Simulations demonstrate that in the closed-loop case, acceleration monitoring can provide modest guidelines for path selection. In the same vain, an observability analysis of the system is performed to give insight into desired trajectories and the difficulties associated with vSLAM systems.

Finally, a Schmidt-Kalman filter is used to reduce the number of states required in the filter. A reduction from three states per feature to just one is achieved. Performance comparisons are given for both the open-loop and closed-loop cases.

B. The Complementary Form Extended Kalman Filter

A complementary form EKF is used to estimate the errors in vehicle egostates states and feature states, nomenclature can bee found in Table V. The EKF state vector used is

$$\mathbf{x} = [\delta\mathbf{r}^T,\, \delta\mathbf{v}^T,\, \delta\boldsymbol{\alpha}^T,\, \boldsymbol{\gamma}^T,\, \boldsymbol{\beta}^T,\, \cdots\, \rho_p^1,\, \theta_p^1,\, \phi_p^1,\, \cdots]^T \tag{3.1}$$

In this formulation, the vehicle egostates are propagated using the INS that is generated using IMU measurements. When bearing measurements become available from the camera, the Kalman filter estimates the errors between the true egostates and the current estimates, the system is represented in Fig. 18. The discrete representation of the system propagation based on the IMU measurements is

$$\mathbf{x}_{k+1}^- = \mathbf{F}\mathbf{x}_k^+ \tag{3.2}$$

$$\mathbf{P}_{k+1}^- = \mathbf{F}\mathbf{P}_k^+\mathbf{F}^T + \mathbf{Q} \tag{3.3}$$

The first 15×15 elements of $\mathbf{F}$, written in 3×3 submatrix form, are

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & \mathbf{I}\Delta t & 0 & 0 & 0 \\ 0 & \mathbf{I} & -[\times\mathbf{f}]\Delta t & 0 & \mathbf{C}_N^b\Delta t \\ 0 & 0 & \mathbf{I} & \mathbf{C}_N^b\Delta t & 0 \\ 0 & 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{I} \end{bmatrix} \tag{3.4}$$

The rest of $\mathbf{F}$ applies the identity operation to all feature states as features are assumed stationary. The INS is used in place of the vehicle model, so there is no need to propagate vehicle states based on control commands in the prediction phase. In (3.2) no noise term is added because the INS is a function of measurements and the noise is captured in the covariance matrix, $\mathbf{P}$. Because of the fast INS update rate, it is used “as is” until the next camera measurements become available. In the case no camera measurement is ready the estimated state values are set to the INS prediction, $\mathbf{x}_{k+1}^+ = \mathbf{x}_{k+1}^-$.

Table V. Complementary EKF Nomenclature

Notation	Definition
$\mathbf{x}_k$	Filter states at time k
$\delta\mathbf{r}$	Vehicle position errors
$\delta\mathbf{v}$	Vehicle velocity errors
$\delta\boldsymbol{\alpha}$	Vehicle attitude errors
$\boldsymbol{\gamma}$	Gyro drift
$\boldsymbol{\beta}$	Accelerometer bias
ρ^1	Estimate of initial range to a feature
θ^1, ϕ^1	Estimates of initial bearings to a feature
$\mathbf{F}$	State transition matrix
$\mathbf{P}$	State covariance matrix
$\mathbf{Q}$	Process noise matrix
$\mathbf{C}_N^b$	Rotation matrix between the vehicle and navigation frames
$[\times\mathbf{f}]$	Skew-symmetric matrix of the measured specific forces
$[\cdot]^-$	Predicted value
$[\cdot]^+$	Corrected (updated) value
$\mathbf{H}$	Observation matrix
$\mathbf{K}$	Kalman gain matrix
$\mathbf{R}$	Measurement covariance matrix
Δt	Time step between measurements

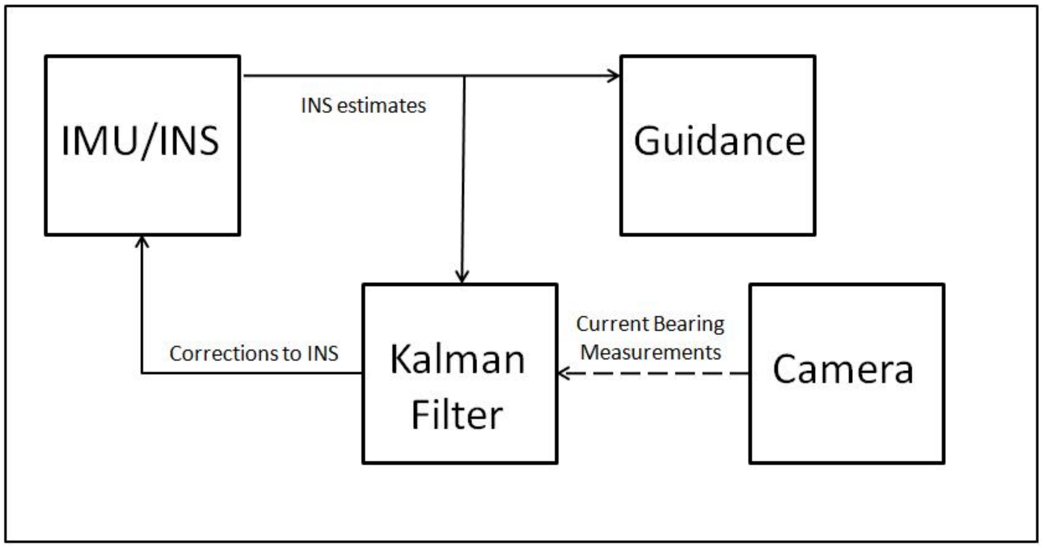


Fig. 18. Complementary form Kalman filter design of vision-aided INS systems. The dashed line represents the slower update rate of camera measurements.

When the camera has captured an image and bearing measurements to features are extracted, they are used to estimate the errors in the INS. This allows the INS to provide guidance information at the IMU update rate and receive corrective image-based updates at the rate the image data is being processed. The update equations are

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{K}(\mathbf{y} - \mathbf{H}\mathbf{x}_k^-) \tag{3.5}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_k^- \tag{3.6}$$

where

$$\mathbf{K} = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1} \tag{3.7}$$

and $\mathbf{H}$ is the observation matrix, which is discussed in the following section.

C. Unit Sphere Observation Model

Typical vSLAM applications use euclidian coordinate estimates of feature point locations to generate anticipated pixel coordinates for each feature based on the estimated vehicle states. Those anticipated pixel values are then compared to the measured pixel values, as seen by the camera. The difference is then used in the extended Kalman filter (EKF) to update the filter states. Although, not the focus of this research, an alternative approach to the SLAM problem is used. The filter takes advantage of geometric constraints on the unit sphere relating the initial bearing and range to a feature, the change in position and orientation of the vehicle, and the latest bearing measurement to a feature. Instead of directly comparing measured and anticipated pixels as in most vSLAM applications, the filter measures how well the constraint equations are met to generate the gain matrix **K**, as seen in (3.7), and update filters states accordingly.

A unit-circle frame is represented in Fig. 19, which associates with one of the two constraints associated with the unit-sphere, (3.8). A full workup of the constraint equation and corresponding observation model can be seen in [43], only the essentials are provided here using nomenclature found in Table VI.

In the six degree of freedom (6DoF) case the two constraint equations for a single feature are

$$(\mathbf{e}^2)^T \cdot \Delta\mathbf{C}_N^b \cdot \mathbf{B} \cdot \Delta\mathbf{R} = (\mathbf{e}^1)^T \cdot \mathbf{B}^T \cdot \Delta\mathbf{C}_N^b \cdot \mathbf{e}^2 \cdot \rho^1 \tag{3.8}$$

$$(\mathbf{e}^2)^T \cdot \Delta\mathbf{C}_N^b \cdot \mathbf{D} \cdot \Delta\mathbf{R} = (\mathbf{e}^2)^T \cdot \Delta\mathbf{C}_N^b \cdot \mathbf{e}_\perp^1 \cdot \rho^1 \tag{3.9}$$

where

$$\mathbf{e}^j = \begin{bmatrix} \cos(\phi^j) \cdot \cos(\theta^j) \\ \sin(\phi^j) \cdot \cos(\theta^j) \\ \sin(\theta^j) \end{bmatrix} \tag{3.10}$$

Table VI. Unit Sphere Constraint Nomenclature

Notation	Definition
$\Delta\mathbf{R}$	Change in vehicle position
$\Delta\mathbf{C}_N^b$	Change in vehicle orientation
$(\mathbf{e}^j)$	Unit normal in the direction of the feature at the jth time instant
$(\mathbf{e}_\perp^j)$	Unit normal perpendicular to $(\mathbf{e}^j)$
ρ^1	Estimate of the initial range to a feature
θ^1, ϕ^1	Estimates of initial bearing values
θ^j, ϕ^j	Bearing measurement at the jth time instant

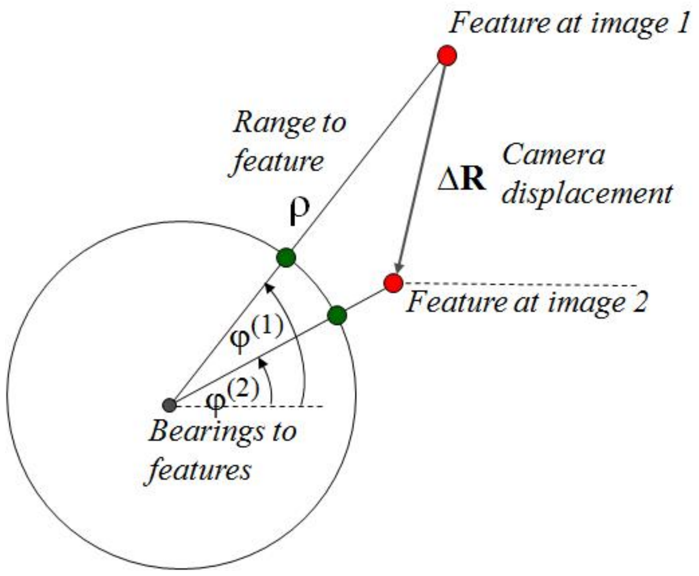


Fig. 19. Unit circle representation of translational only motion (Soloviev et al., 2009).

$$\mathbf{B} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.11}$$

$$\mathbf{D} = \begin{bmatrix} 0 & 0 & -\cos(\phi^1) \\ 0 & 0 & -\sin(\phi^1) \\ \cos(\phi^1) & \sin(\phi^1) & 0 \end{bmatrix} \tag{3.12}$$

The motion constraints are used to generate motion observables by substituting egostate estimates into the equations, then subtracting the right side from the left side in both (3.8) and (3.9). By writing each component (position, velocity, altitude, bearings, range, etc.) as the truth plus an error, the equations can be reorganized and contain constant, linear and quadratic error components. The constant term will be zero because it will be the constraint equation exactly and the linear error terms will be associated with position, attitude, initial bearings and initial range. Finally to support the complementary form EKF formulation the system is linearized by dropping the quadratic error terms that results in equation (9) from [43], rewritten in 3.13.

$$\boldsymbol{\eta}_p^{(n)} = \mathbf{H}_{p,\Delta\mathbf{R}}\delta\mathbf{r} + \mathbf{H}_{p,\Delta\boldsymbol{\alpha}}\delta\boldsymbol{\alpha} + \mathbf{H}_{p,\mathbf{f}} \begin{bmatrix} \delta\phi_p^1 \\ \delta\theta_p^1 \\ \delta\rho_p^1 \end{bmatrix} + \mathbf{H}_{p,\boldsymbol{\varepsilon}} \begin{bmatrix} \delta\phi_p^n \\ \delta\theta_p^n \end{bmatrix} \tag{3.13}$$

The reader should note in (3.8) and (3.9) that the constraints scale with the size of vehicle motion and distance to features. This scaling will cause a bias where the filter tends to underestimate distance traveled and the distance to features in non-observable or weakly observable settings. To avoid this scalability the user could simply divide both sides of each equation by the ρ term and normalize the constraint equation (so the system is unitless in distance). However, dividing the constraints by

ρ causes the partial derivative with respect to ρ to become highly nonlinear. While normalizing the constraints does improve estimator accuracy, in weakly observable cases the estimates are still not suitable. For paths that generate better observability the un-normalized filter avoids the scale biasing (there is enough pertinent information in the filter to prevent scaling) and estimator results are more consistent than the more nonlinear, normalized constraints. For this reason the non-normalized constraints, (3.8) and (3.9), are used throughout.

This covers the basic estimator development. There is still the issue of a synthetic stereo approach to generate an initial range estimate for a feature, however this is covered in [43], and will not be revisited except to say care must be taken when generating the initial range estimate. If the range values are calculated and used without regard for the correlation with the uncertainties in your vehicle states, then the range estimates will effectively match the INS errors providing no benefit to the estimator.

D. Simulation: Hallway With Turns

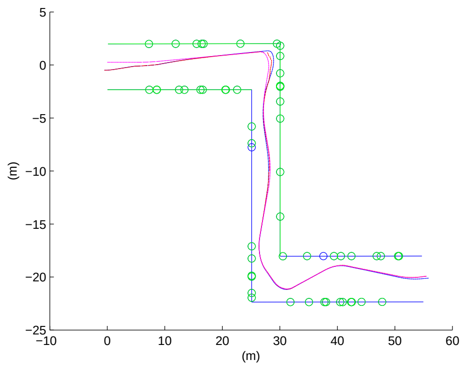
The estimator is simulated in a generic hallway environment that is three meters wide and three meters tall. There are random feature points distributed along the walls. Using a forward facing camera, the vehicle is asked to track a specified path using a simple closed-loop guidance law, to be discussed in Chapter IV. The vehicle runs solely on IMU signals to start while generating initial guess for feature distances. Once features are initialized into the filter, estimation and correction of egostates occur as previously described.

An example with turns can be seen in Fig. 20 along with the position estimation error and 3-sigma values. For normal distributions the standard deviation, generally denoted sigma, σ, is the square root of the variance of the distribution. Statistically speaking, approximately 99.7% of random samples from the normal distribution will fall in the range $\mu \pm 3\sigma$, where μ is the mean value of the normal distribution. Kalman filters make assumptions that the states have a normal-like distribution, the estimator covariance determines the 3-sigma boundary the vast majority of errors should lie in. Fig. 20 shows the position error is within the 3-sigma bounds and the improvement due to hallway turns is especially evident at the first turn. The UGV has a final position error of 0.36 meters.

Paths used in the simulation are combinations of the sawtooth and sinusoidal paths. Straight line trajectories are used with rounded corners, minimum radius of two meters, allowing the vehicle to travel at a constant speed of two meters per second for simulation. It is clear in Fig. 20 the UGV is able to keep accurate egostate estimates. If the vehicle did not have the camera onboard, the IMU drifts and biases would cause the estimates to diverge. In Fig. 21 the vehicle operates without a camera. In this simulation, open-loop control is used so the vehicle tracks the desired path exactly. Clearly seen, with the IMU alone the filter is unable to keep an accurate estimate of vehicle egostates and the simulation finishes with a 10.1 meter error in its position estimate.

E. Simulation: Visual Odometry Filters

In SLAM or vSLAM the EKF is able to initialize a feature, lose it from view, return to and recognize (reassociate) the feature. This is an important trademark because it allows loop closure that boosts estimate accuracy in feature and vehicle egostates [5]. This capability requires a feature map be maintained and when a feature is revisited it must be verified to be a previously identified feature, which can be a computationally



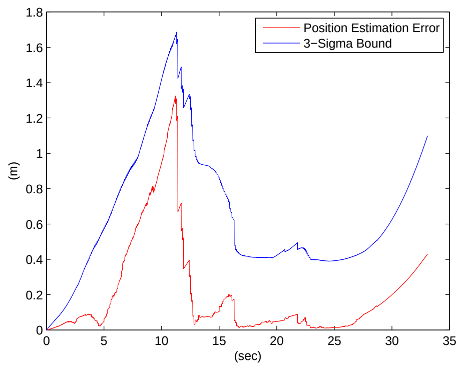


Fig. 20. Left: Modified sinusoidal path in a hallway with turns. Magenta lines are the desired vehicle position, red lines are the estimated vehicle position, and blue lines are the true vehicle position. Right: Position error and 3-sigma values vs. time.

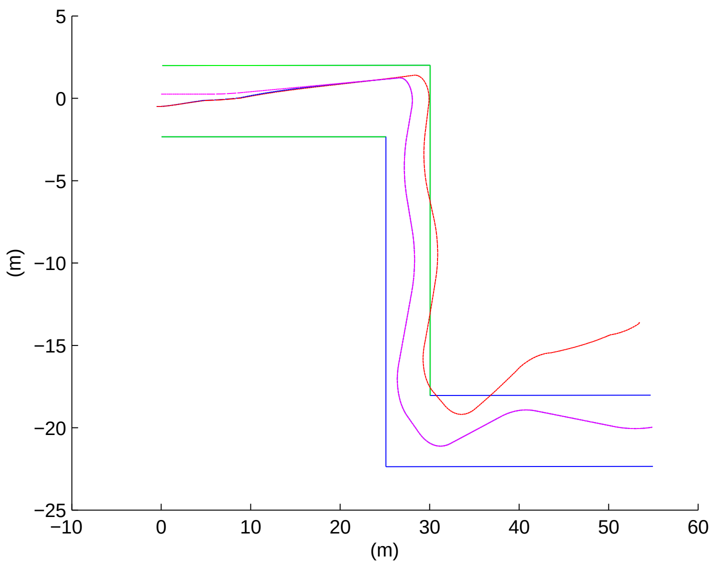


Fig. 21. Modified sinusoidal path in a hallway with turns. IMU only, no camera. Magenta lines are the desired vehicle position, red lines are the estimated vehicle position, and blue lines are the true vehicle position.

complex task. If instead a visual odometry approach is taken and revisited features are treated as new features there is no longer a need to recognize and reassociate features. This drastically simplifies the hardware implementation of this problem requiring only the use of a simple feature tracker, but will generally reduce accuracy of the system because there is pertinent information going unused.

The same hallway with turns simulation can be seen in Fig. 22, where a visual odometry filter is used. The UGV simulation finishes with a position error of less than 0.22 meters that is better than the associative filter, although the average error over the course of the run is larger. Again, you can clearly see the hallway turns dramatically improve estimation errors, although not to the same extent, likely due to the inability to reassociate previously seen feature points and the error introduced when reinitializing them. Over several run with differing features the average visual odometry filter simulations resulted in errors approximately 1.5 times that of the full vSLAM filter.

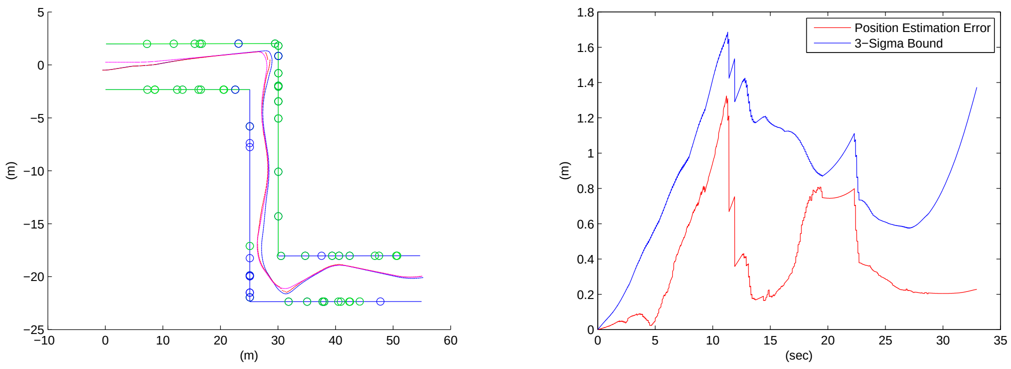


Fig. 22. Left: Modified sinusoidal path in a hallway with turns for a visual odometry filter. Magenta lines are the desired vehicle position, red lines are the estimated vehicle position, and blue lines are the true vehicle position. Right: Position error and 3-sigma values vs. time.

F. Summary

Autonomous navigation in GPS denied environments is becoming an increasingly desired capability. The complementary form Kalman filter was introduced along with the unit sphere observation model allowing a non-GPS based INS to keep egostate estimates in unknown settings. In the following chapters, only an IMU and camera will be considered although the unit sphere approach easily accommodates additional sensor packages including GPS when available. Not only does this filter setup provide the flexibility to navigate in most imaginable settings, it also acts as a safety margin for navigation when GPS is expected. The complementary form of the filter does not require a vehicle model, making this approach applicable to UAVs/UGVs of any size. The filter was demonstrated in simulation for both the vSLAM and visual odometry approaches.

CHAPTER IV

IDENTIFYING DESIRED VEHICLE PATHS

A. Introduction

Simulations of a UGV navigating a long straight hallway or corridor are conducted. The hallway is three meters wide and three meters tall, and there are features randomly distributed on the walls. The position of each feature is initially unknown to the vehicle. As the UGV moves down the hallway it keeps estimates of its egostates. Features seen by the camera are incorporated into the filter to aid egostate estimation using the observation model introduced in Chapter III. The hallway scenario is used to gauge the effects of path, feature visibility and availability on egostate estimates.

Two path types are generated for the UGV to follow, sinusoidal and sawtooth, described by their amplitude and spatial period. Example paths can be seen in Fig. 23. The performance of the estimator is initially examined using open-loop guidance, when the vehicle follows a desired path exactly. While not physically possible to implement, results indicate which motion types provide good egostate estimates without the complications of feedback control. Later closed-loop guidance using the estimated vehicle states is simulated and shown to be consistent with open-loop results.

B. Simulation Settings

For sinusoidal paths, the vehicle maintains a constant speed of two meters per second along the path and the vehicle orientation is in the direction of travel. Sawtooth paths move forward at two meters per second and take a tenth of a second to stop and half a second to accelerate from zero to two meters per second at each path apex, the vehicle turns at a rate of 60 degrees per second (note these values were altered and it showed negligible impact on the estimator performance). A pinhole camera model is used with a 40 by 30 degree field of view and 640 by 480 resolution. The angular measurement errors have a standard deviation of 0.23 degrees. The distance at which features become visible is simulation can also be set by the user.

The IMU bias and drift are modeled using first-order Gauss-Markov processes characterized by a gyro drift stability of 200 degrees per hour and an accelerometer bias of two milli-gravity. The UGV begins with zero uncertainty in initial position (there is no global signal, so any error here is irrelevant), an initial velocity estimation error with a standard deviation of 0.02 meters per second and an initial vehicle orientation error with a standard deviation of 0.1 degrees.

Because the UGV has no prior knowledge of feature locations, a synthetic stereo technique is used to initialize range and angular error estimates and covariance [43]. As the UGV comes across new features it continues to use the synthetic stereo to initialize new features as it progresses down the hallway. A depiction of the UGV in the hallway can be seen in Fig. 24.

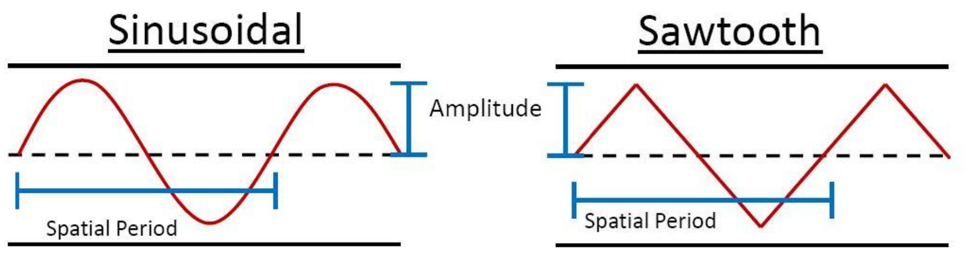


Fig. 23. Sinusoidal and sawtooth path descriptions.

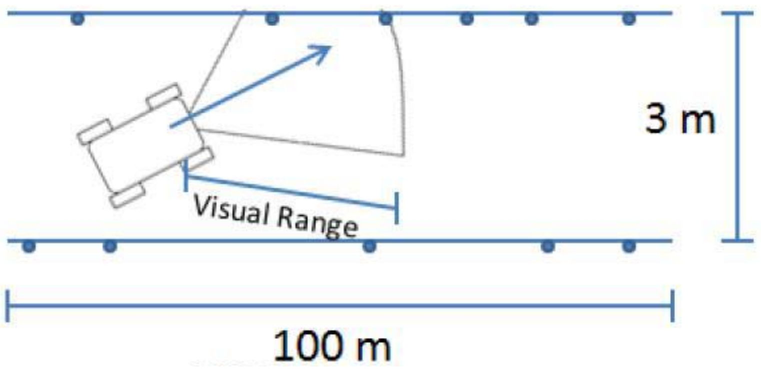


Fig. 24. Depiction of a UGV moving down a hallway.

C. Desired Control Signatures

Sinusoidal and sawtooth paths should accommodate almost all UGVs, and are fully characterized by their amplitude and spatial period parameters (aside from phase shifts, that is not currently considered). The goal here is to identify which parameter sets provide high estimation accuracy in the average case. Feature reassociation is perfect in these simulations, meaning that if a feature leaves the field of view and later returns to the field of view, it will be recognized as the same feature and not as a new feature. Thus, turning sharply could be beneficial since the UGV repeatedly revisits established features as it moves down the hallway in typical SLAM fashion.

1. Prescribed Paths

The first example is for 80 total features, 40 on each wall. The features are visible from up to 25 meters away. The UGV navigates the same hallway for each pairing of amplitude and spatial period in Table VII and the final position errors at 75 meters are recorded. Results of single runs for each parameter pairing is shown in Figs. 25

Table VII. Sample Amplitude and Spatial Period Values

Index	Amplitude (m)	Spatial Period (m)
1	0.1139	1.1391
2	0.1709	1.7086
3	0.2563	2.5629
4	0.3844	3.8443
5	0.5767	5.7665
6	0.8650	8.6498
7	1.2975	12.9746
8	-	19.4620
9	-	29.1929
10	-	43.7894

and 26 and there is a clear correlation between path parameters and performance. These plots are representative of the average results with differing initial biases and features distributions.

There is a large region of path parameters with final position errors of 0.75 meters (1%) or less (everything located below the dark blue, 0.75 meter contour in Fig. 26). Some regions even have errors as small as 0.1%. Not surprisingly the parameter sets that do well are combinations of larger amplitudes and shorter spatial periods, which are the paths requiring the most acceleration. The associated sawtooth results can be seen in Figs. 27 and 28. It should be noted that the sawtooth performs almost as well, but begins to have trouble for the larger amplitudes, where the sinusoid excels, due to the lengthy straight line sections associated with those paths.

It is of interest that the error contours of the sinusoidal path correspond rather closely to the maximum acceleration experienced by the vehicle. However, if the visual

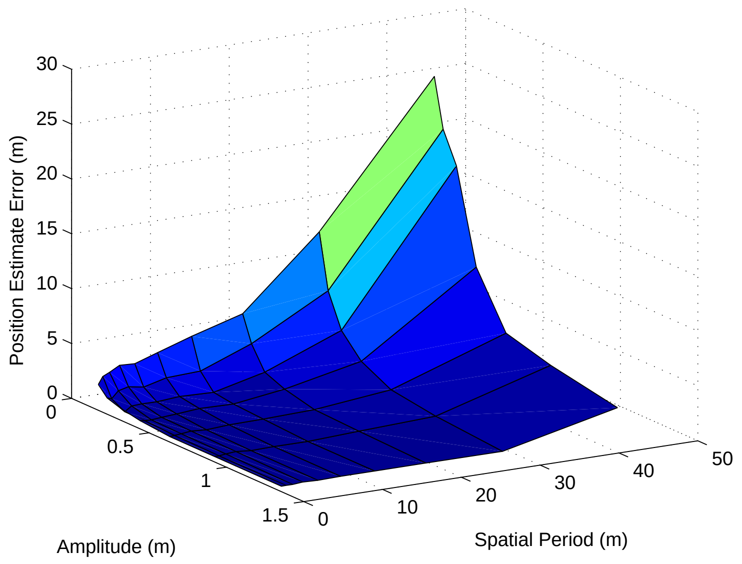


Fig. 25. Sinusoidal path parameters vs. estimation error using open-loop guidance, 80 features, and 25 meter sight distance.

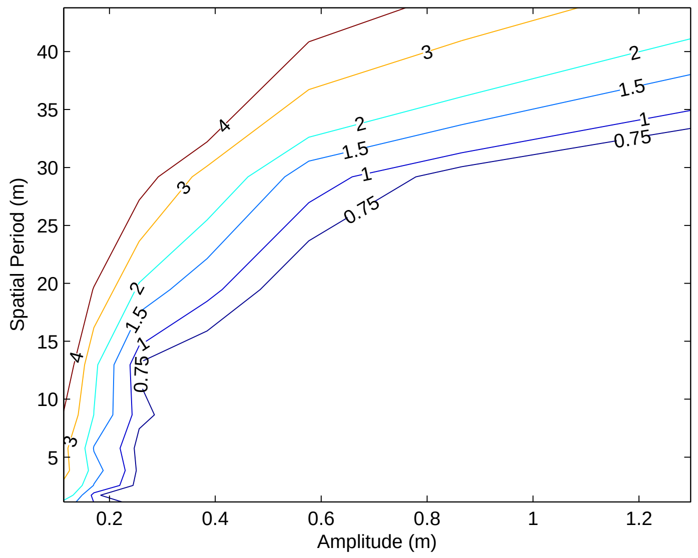


Fig. 26. Sinusoidal path error contours using open-loop guidance, 80 features, and 25 meter sight distance.

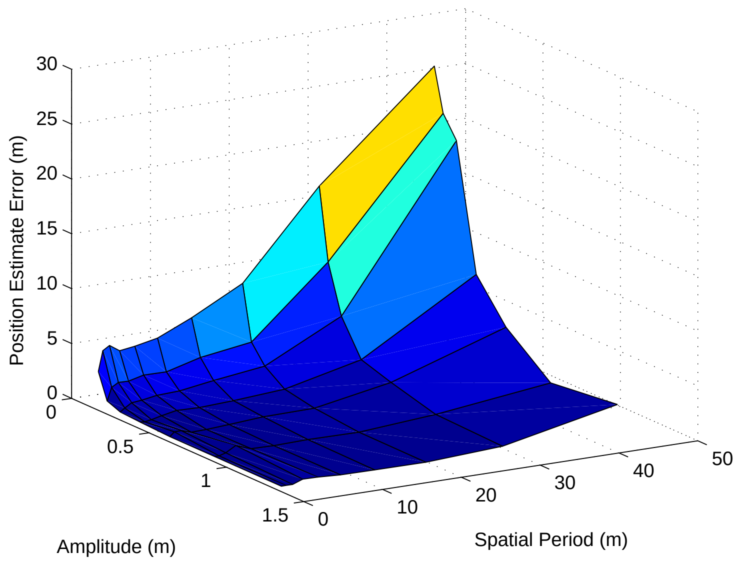


Fig. 27. Sawtooth path parameters vs. estimation error using open-loop guidance, 80 features, and 25 meter sight distance.

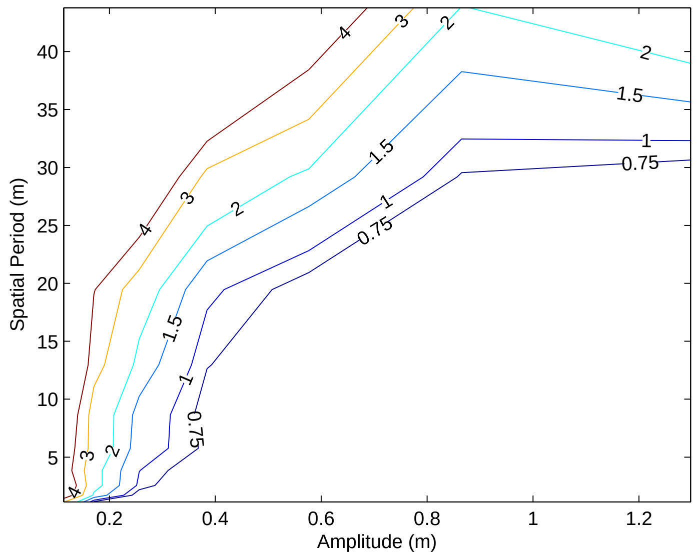


Fig. 28. Sawtooth path error contours using open-loop guidance, 80 features, and 25 meter sight distance.

range or the number of features is decreased, the error contours begin to resemble the vehicle's maximum angular orientation with respect to the hallway walls, Fig. 29. This suggests that turning sharply towards the wall is important when the feature count is low becasue it provides a longer time history with individual features, leveraging more of the available information.

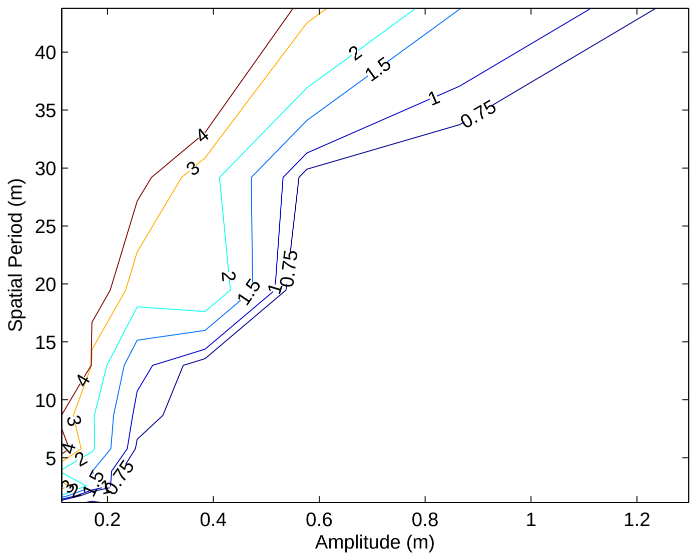


Fig. 29. Sinusoidal path parameters vs. estimation error using open-loop guidance, 10 features, and 25 meter sight distance.

Simulations show the low visual ranges negatively affect the estimator, and as visual range is increased performance improves quickly until the vehicle is able to see features at 20 meters, where additional sight distance shows little benefit. Interestingly the estimator achieves a larger path parameter set with under 1% error with only 10 features on the walls as opposed to the 80 feature case when the sight distance is 25 meters. However, the best results are no longer in the 0.1% range as in the 80 feature case.

Additional features can degrade estimator performance if their initialization introduces more error into the filter than the features can mitigate during the time they are seen. For the larger spatial periods where there is little angular change of the vehicle along its path features are generally seen from a distance and grouped near one another in the image. In this case an additional features provides little additional information because they are practically overlapping other features. Again, this is evident in Fig. 29, where the 1% or less error region is actually larger for the 10 feature case that it is for the 80 feature case shown in Fig. 26.

If however, a more energetic path is used the features are more often seen while close to the vehicle (due to the sharper angle with respect to the hallway), effectively spreading them out in the image plane and the additional features may now provide additional, unique information that results in more accurate estimates. This is also demonstrated in 10 feature case where the smallest estimation errors are on the order of 0.5% as opposed to 0.1% in the 80 feature case. Using just 10 features results in a larger "acceptable" region, but it lacks the required information to attain the most accurate estimates.

More feature points is intuitively better, but it is also computationally expensive. Additional features will be beneficial if they provide unique information to the filter, and the more energetic the path the more features can be used. This suggests a user can be selective when choosing to add a feature into the filter. When potential features are stacked near one another, especially for lower energy paths, the user should only incorporate a couple into the filter. In this case fewer features will not only reduce the computational loads, but actually improve egostate estimation making this an ideal rule to follow. For the more energetic paths, additional features beyond the 80 used, quickly stop improving estimation accuracy and simply require more computations for the same or even degraded results.

2. Closed-loop: Proportional-difference Control Feedback

The previous simulations used prescribed vehicle positions at each time step and control signatures that provided good egostate estimates were identified. Now closed-loop control of the vehicle, using the estimated state rather than the true state, is simulated to examine the system performance in closed-loop. A very simple proportional-difference (PD) control method is applied using a damped oscillator model and nomenclature can be seen in Table VIII.

Table VIII. Closed-loop Control Nomenclature

Notation	Definition
$\mathbf{r}_n$	Nominal (desired) position
$\tilde{\mathbf{r}}$	Estimated position
$\mathbf{a}_n$	Nominal acceleration
$\tilde{\mathbf{e}}$	Estimated position error
$\dot{\tilde{\mathbf{e}}}$	Estimated position error rate
$\tilde{\mathbf{a}}$	Corrective acceleration term
c, k	Control gains
dt	Time step
$\mathbf{a}$	Total acceleration

The nominal acceleration required to follow a desired path at the ith step, $\mathbf{a}_n^i$, is

$$\mathbf{a}_n^i = (\mathbf{r}_n^{i+1} - 2\mathbf{r}_n^i + \mathbf{r}_n^{i-1})/dt^2 \tag{4.1}$$

The estimated position and velocity errors over the last few time steps are defined using the estimated position values, $\tilde{\mathbf{r}}$.

$$\tilde{\mathbf{e}}^i = \tilde{\mathbf{r}}_n^{i-1} - \mathbf{r}_n^{i-1} \tag{4.2}$$

$$\dot{\tilde{\mathbf{e}}}^i = (\tilde{\mathbf{e}}^i - \tilde{\mathbf{e}}^{i-1})/dt \tag{4.3}$$

Using the damped oscillator model, the corrective acceleration term is,

$$\tilde{\mathbf{a}}^i = -c\dot{\tilde{\mathbf{e}}}^i - k\tilde{\mathbf{e}}^i \tag{4.4}$$

In this case letting $c = 2\sqrt{k}$ provides a critically damped system (when position estimates are equal to the truth). The guidance now uses the total acceleration,

$$\mathbf{a}^i = \mathbf{a}_n^i + \tilde{\mathbf{a}}^i \tag{4.5}$$

This is clearly not a robust controller, but will demonstrate the use of estimated egostates in a guidance scheme. If the estimator error becomes too large, there is nothing limiting the accelerations and the system could diverge drastically. However, the path signatures that provided accurate estimates in open-loop also perform well in closed-loop for a several gain values. Examples in this section will use a gain value of $k = 15$.

An example sinusoidal path with an amplitude of 0.87 meters and a spatial period of 12.97 meters is shown in Figs. 30. The vehicle begins displaced 0.9 meters from the desired path and is commanded to track back appropriately. The estimator performs quite well and over the first 20 meters the UGV estimated position converges with the desired path, keeping excellent egostate estimation accuracy throughout. If a larger spatial period of 43.79 meters is used the estimate begins to diverge as seen in Fig 31. The estimated position tracks the desired path but the estimate is no longer accurate and the true vehicle position begins to deviate from desired location. This is expected as similar errors were seen in the estimates for the open-loop case. Similar results are obtained for the sawtooth path type.

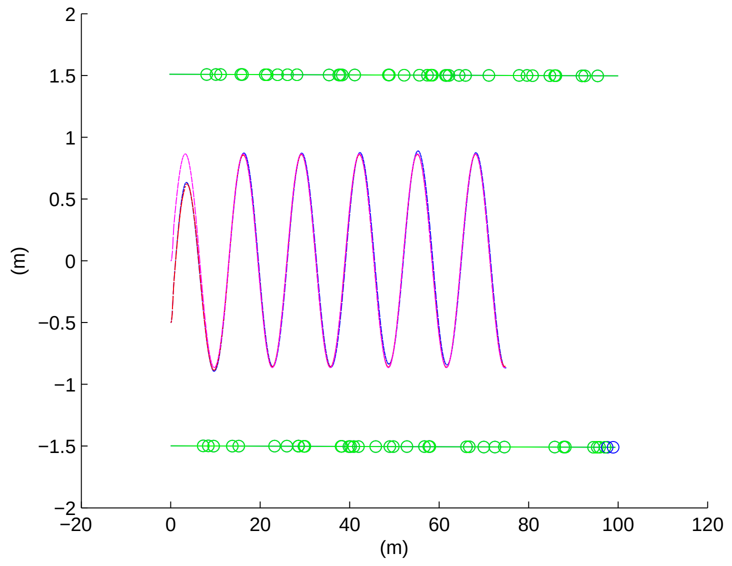


Fig. 30. Desired path tracking using closed-loop guidance for a 12.97 meter spatial period and 0.87 meter amplitude path. Magenta lines are the desired vehicle position, red lines are the estimated vehicle position, and blue lines are the true vehicle position.

The simulations from the previous section are repeated for the closed-loop case with the control pairings from Table VII. The surface and contour maps of final position error as a function of spatial period and amplitude for the sinusoidal case can be seen in Figs. 32 and 33. These examples clearly outperform the open-loop cases for the same trajectories, Figs. 25 and 26, due to the additional accelerations generated in the guidance law. However; the acceptable (1%) range, remains virtually unchanged (small errors in position estimates translates to small additional acceleration terms).

Similar results are obtained for the sawtooth path-type. However, the simulated vehicle is no longer running straight trajectories. A more robust simulation is needed

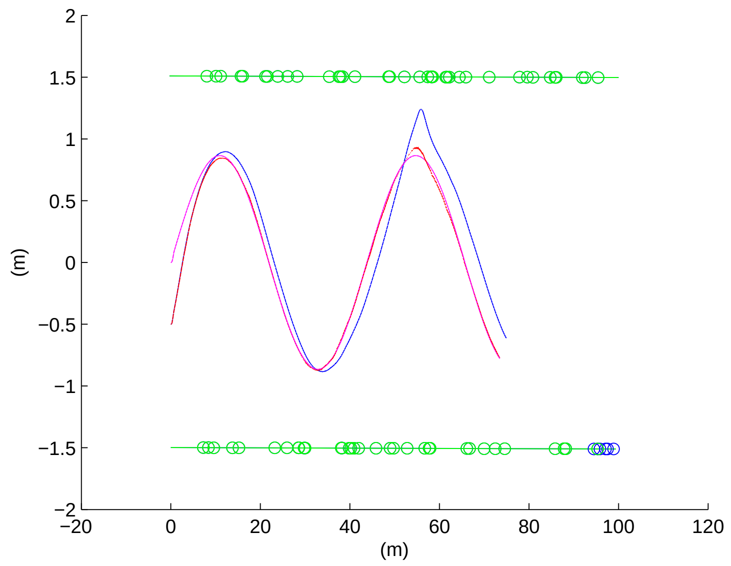


Fig. 31. Desired path tracking using closed-loop guidance for a 43.79 meter spatial period and 0.87 meter amplitude path. Magenta lines are the desired vehicle position, red lines are the estimated vehicle position, and blue lines are the true vehicle position.

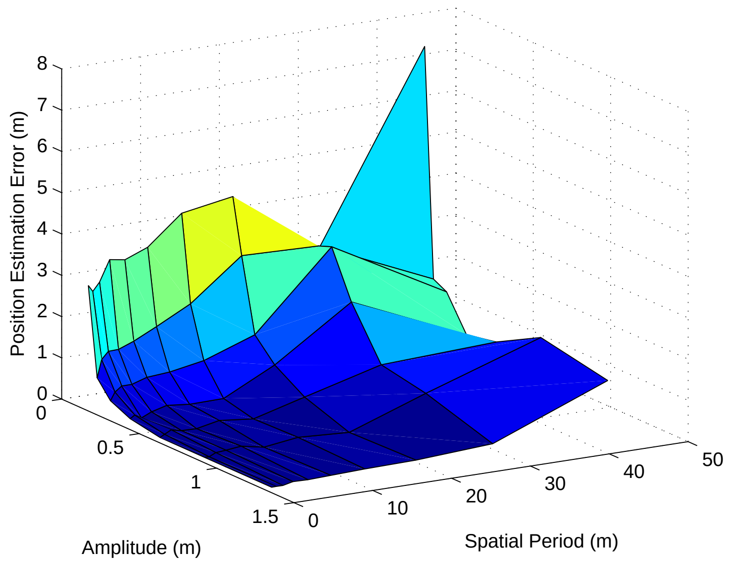


Fig. 32. Sinusoidal path parameters vs. estimation error using closed-loop guidance, 80 features, and 25 meter sight distance.

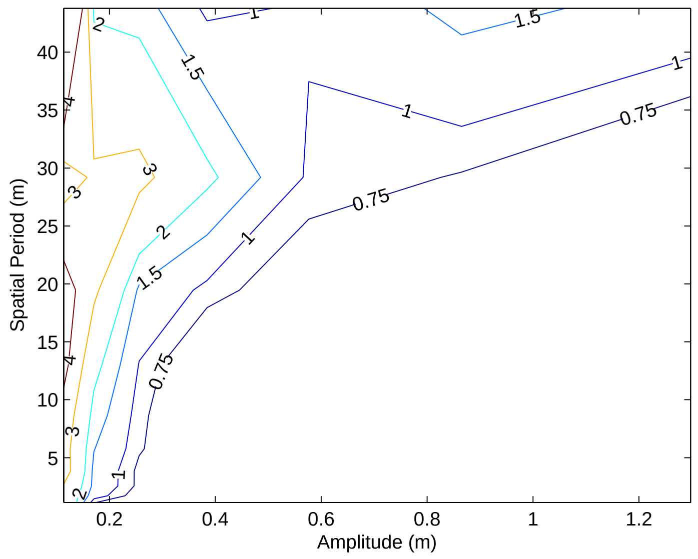


Fig. 33. Sinusoidal path estimation error contours using closed-loop guidance, 80 features, and 25 meter sight distance.

where corrects are made to straight line courses once estimated position errors reach a certain size in order to be more consistent with sawtooth path types. Again, additional control is used and it outperforms the open-loop simulation for the same desired nominal trajectory.

3. Benefits of Integrated Estimation and Control

The PD controller from the previous section is used to demonstrate the importance of taking estimation into account when choosing a path. Generally in the closed-loop setting, the more nominal acceleration a path requires the larger the total acceleration will be. However, due to more inconsistent estimation results the corrective acceleration terms for lower nominal acceleration paths are sometimes rather large. At some point, the additional corrective acceleration terms will outweigh the initial cost benefits of the "lower cost" nominal path. In these cases a "higher cost" nominal path should be used.

With no specific vehicle model, the total acceleration for a path is used as a measure of the control cost. The true control cost would certainly be vehicle dependent and require knowledge of the platform for simulation. The total acceleration used is a simple and practical substitute assuming larger or more accelerations will incur a larger control cost.

The acceleration contours for the closed-loop case can be seen in Fig. 34. As the spatial period is increased the total acceleration initially decreases drastically before leveling out and eventually beginning to increase. The increase in acceleration is entirely due to the corrective acceleration terms as the nominal accelerations continue to decrease as the spatial period increases. Fig. 34 clearly shows that approximately 15 meter spatial periods are the most efficient for this particular simulation in terms of total acceleration used. The acceleration demands more than doubles by increasing the spatial period from 15 meters to 40 meters yet the estimation errors actually increases, shown in Fig. 33.

The evaluation index of total acceleration multiplied by the estimation error with even weights on each component is applied to the closed-loop simulations. This cost function rewards accurate estimates with minimal accelerations; while simplistic it captures the essence of the trade off between control cost and estimation accuracy in a crude manner. It becomes clear for this particular measure that the use of excessively costly nominal paths is unwarranted as their is little if any estimation accuracy gain compared to the significant acceleration increase. Additionally, using low cost nominal paths also becomes undesirable, notably due to the poor estimation, but also due to the increased control required. Fig. 35 displays the associated contour lines of the ad hock evaluation function.

To emphasize the connection between estimation and control, a single varied

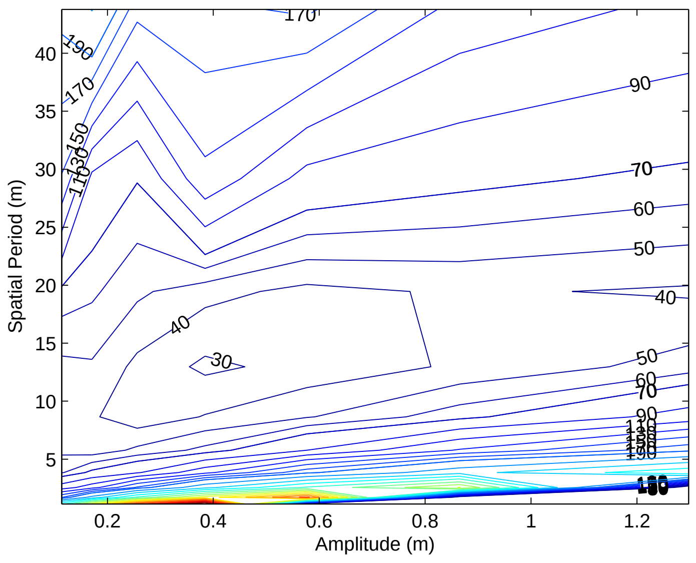


Fig. 34. Total acceleration contours as a function of path parameters for sinusoidal paths using closed-loop guidance, 80 features, and 25 meter sight distance.

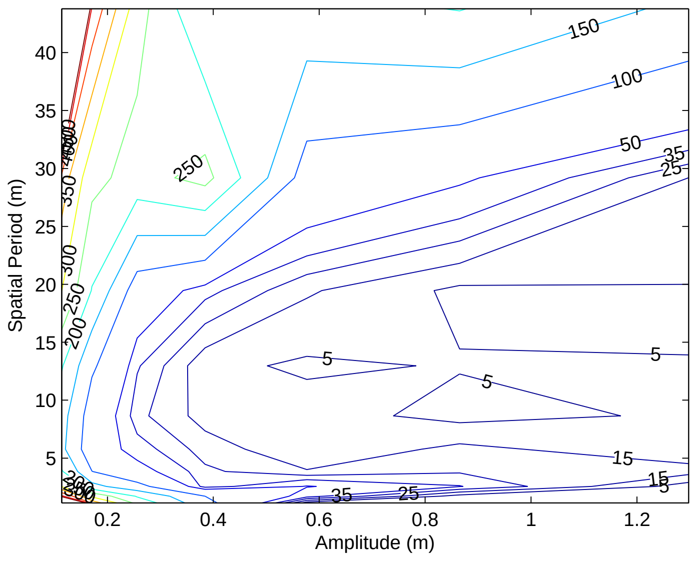


Fig. 35. Total acceleration multiplied by estimation error contours as a function of path parameters for sinusoidal paths using closed-loop guidance, 80 features, and 25 meter sight distance.

parameter is used with a fixed amplitude of 0.87 meters. Fig. 36 shows the estimation error as a function of spatial period for sinusoidal paths with 0.87 meter amplitude. Fig. 37 shows the total accelerations for each path, demonstrating the less accurate estimates of larger spatial period paths can result additional, undesired accelerations. It becomes clear that for this particular configuration, it is ideal from a estimation and control perspective to use spatial periods in the range of 10 to 20 meters. Any longer, and the estimates degrade (and costs rise slightly), any shorter and the control cost increases rapidly with little or no estimation benefit. Again, using the simple index of total acceleration multiplied by the position estimation error, the desired spatial period is easily identified and Fig. 38 shows the minima at approximately a 13 meter spatial period.

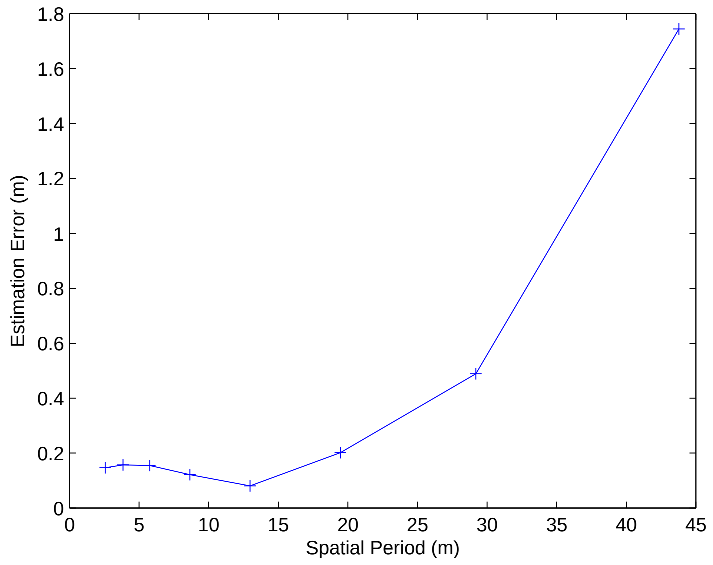


Fig. 36. Position error vs. spatial period for sinusoidal paths with 0.87 meter amplitude using closed-loop guidance, 80 features, and 25 meter sight distance.

These simulations were done with a simple damped oscillator model to provide

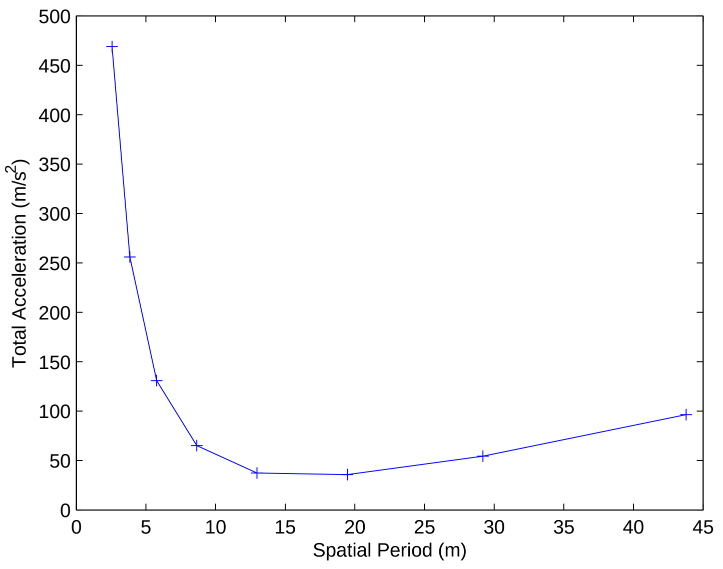


Fig. 37. Total acceleration vs. spatial period for sinusoidal paths with 0.87 meter amplitude using closed-loop guidance, 80 features, and 25 meter sight distance.

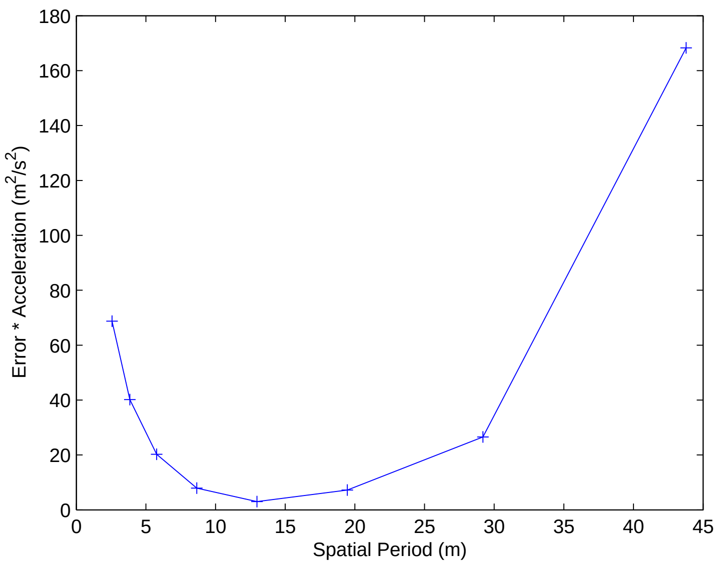


Fig. 38. Total acceleration multiplied by position error vs. spatial period for sinusoidal paths with 0.87 meter amplitude using closed-loop guidance, 80 features, and 25 meter sight distance.

the guidance law with no vehicle model. The use of more sophisticated guidance laws should reduce the phenomenon of increased control cost due to poor estimates; however, the examples demonstrate that poor path choices will undoubtable reduce the effectiveness of the guidance, generating erroneous controls due to the less than stellar accuracy of the egostate estimates. There is a need for estimation to be considered in path selection, not just to meet estimation accuracy requirements, but to reduce control cost as well.

D. Summary

A hallway simulation has been developed for UGVs using camera-aided INS navigation. A unit sphere based complementary EKF was implemented in simulations to estimate vehicle egostates as the UGV traversed the hallway using just an IMU and the bearing measurements of unknown feature points captured with a monocular camera.

It is clear that nearly straight trajectories are not a viable option for accurate estimation. Since the system must be excited in some manner, two very simple path types were simulated. The sinusoidal and sawtooth paths were shown to provide the system with enough data to accurately estimate vehicle egostates. This was accomplished without the need for a vehicle motion model making results, and the estimator, applicable to most UGV types. It was also demonstrated that a relatively small number of features are required for accurate egostate estimation.

A simple closed-loop guidance law was developed and demonstrated the egostate estimates were sufficiently accurate to be used in guidance laws. A clear benefit to path selection with estimation accuracy as a criterion was shown. Cases where demonstrated where higher initial cost trajectories were shown to be favorable to lower initial cost trajectories because of additional accelerations generated in the guidance law due to poor estimation accuracy and consistency.

CHAPTER V

TOWARDS INTEGRATED ESTIMATION AND CONTROL

A. Introduction

Given specific vehicles, environments, guidance laws, sensors and estimators; there is clearly a “most desirable” path choice for the proper balance of control cost versus estimation accuracy. While what that balance is is highly debatable and how to capture this relationship in an effective manner is anything but simple, the work in this chapter is focused on possible methods and obstacles of integrating estimation and control.

Recall, the goal behind integrated estimation and control is to generate more efficient controls by accounting for estimation accuracy when selecting a trajectory to follow. The error and acceleration contours for position error and acceleration, seen in Figs. 33 and 34 in Chapter IV, clearly demonstrated that accurate estimation can lead to more efficient control commands from a given guidance law. The simplistic cost function of total acceleration multiplied by the estimation error, seen in Fig. 35, also captured this relationship and could be used to select path parameters after an exhaustive search in simulation or a controlled setting. Unfortunately, exhaustive searches are not generally implementable in the field and there is no truthing to provide estimate error values (or the estimator would not be needed), so alternative path selections must be developed in order to be applicable in new, unknown environments.

A proactive path selection approach is investigated with the intent of using current filter states to aid in path selection. Here an N-step covariance based path selection method is considered and shown to be undesirable currently, due to estimator overconfidence. A 6DoF observability analysis is performed and gives incite into the overconfidence of the filter and other potential path selection criterion.

Finally, a reactive path selection method involving acceleration monitoring is investigated and shown to have some promise. The time history of accelerations is used to determine if more or less nominal control input should be used. It is shown to accurately identify the desired path parameters from earlier sections.

B. N-step Covariance Optimization

To maximize the accuracy of the egostate estimates the most obvious criterion to optimize is the expected covariance of those states. To do so, the future covariance must be predicted for a potential path given current filter states. An N-step ahead path optimization that minimize the expected value of a joint estimation and guidance cost function (involving the state estimates and their covariance as well as control costs) is developed in [48]. They make small perturbations to a nominal path in order to identify feature locations and avoid collisions with minimal additional control effort. However; their system has the benefit of GPS signals, which are used for navigation, and only a small number of features to track in order to avoid collision. In this case the path perturbations are designed primarily to affect the estimates and covariance for the features, not the egostates themselves. Due to the GPS signal the egostate estimates are quite good, although perturbations improve observability for the vehicle [38], the largest uncertainties lie in the feature states and those covariances will be the most affected by the additional guidance. Vision SLAM egostate estimates are highly sensitive to changes in feature state estimates due to large covariance values (because there is no GPS signal to rely on, the features are vastly more important). In the vSLAM case, changes made to the path will affect the feature state estimates, but it will also highly influence the egostate estimates, complicating the matter.

The idea of using estimated covariance as a selection criterion is desirable and is investigated here. Initially the covariance values are monitored for accuracy, Fig. 39 shows the final position error and 3-sigma bound at 75 meters for 0.87 meter amplitude sinusoidal paths over several spatial periods. Similar, but slightly less accurate results are seen for the sawtooth path in Fig. 40. Both figures clearly suffer from filter overconfidence for larger spatial period paths, were the position error is actually larger than the 3-sigma bounds. Overconfidence is known to plague EKF implementations of vSLAM, [29]. Currently covariance does not appear to be a reliable indicator of performance and it seems highly unlikely covariance based path selection, based on estimated future covariance, would be viable.

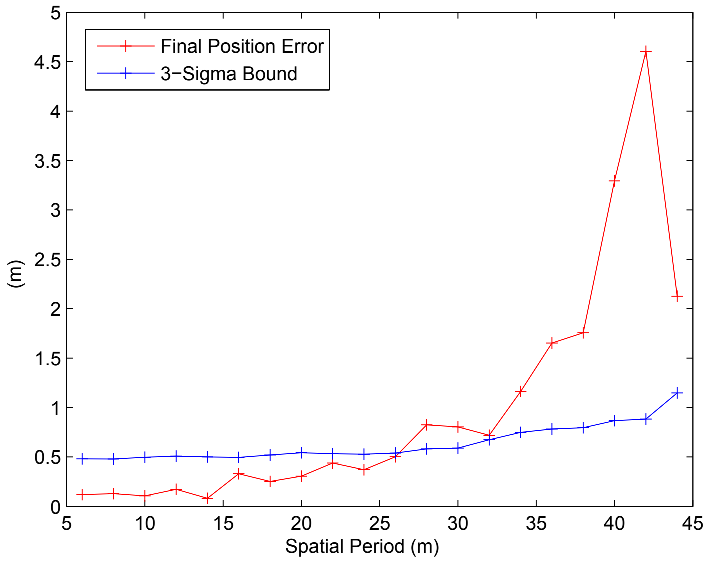


Fig. 39. Position error and 3-sigma value vs. spatial period for sinusoidal paths with 0.87 meter amplitude using closed-loop guidance, 80 features, and 25 meter sight distance.

To verify this in simulation, a "designer" path was generated by allowing the filter to propagate 250 steps into the future for several potential paths, then the path

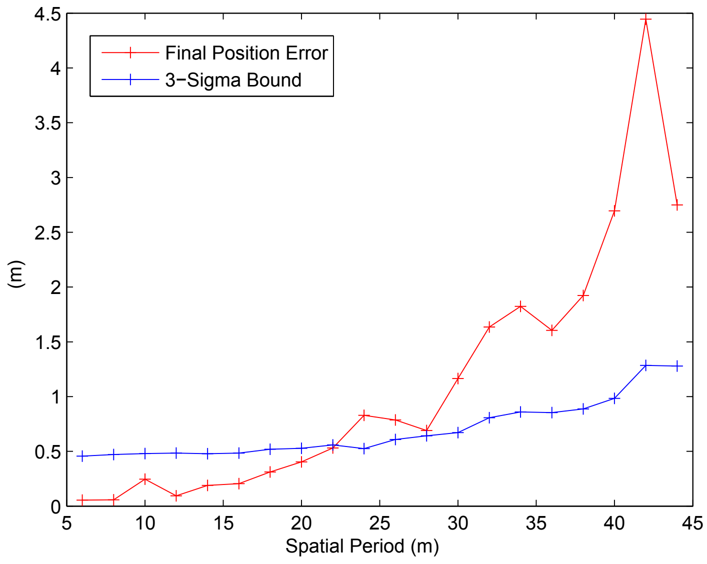


Fig. 40. Position error and 3-sigma value vs. spatial period for sawtooth paths with 0.87 meter amplitude using closed-loop guidance, 80 features, and 25 meter sight distance.

with the smallest position covariance per meter of forward travel was selected. After the UGV moved 250 steps down the preferred path, the selection process was repeated for another set of potential paths extending out an additional 250 steps. Because this was done in simulation, the filter did not have to estimate future covariance for each path, but was instead allowed to use the true covariance results due to each path in the evaluation. Even so, the designer paths generated did not improve estimator performance. Path selection succeeded in generating more confident estimates, but not necessarily a more accurate ones.

Overconfident filters are common in EKFs, especial in the case of significant linearization error or consistently low observability. The complementary form filter removes the nonlinear motion model, but the constraint equations contain nonlinear terms (similar to the nonlinear mapping from world coordinates to the pixel values in standard SLAM and vSLAM). The most significant linearizations are due to errors in range estimates. The range based linearization errors are inherit in the system, but they and other states can be more accurately estimated and with valid covariance if they are made more observable.

C. 6DoF Observability Analysis of vSLAM Systems

Observability analysis is an excellent way to gain incite into potential pitfalls for a given estimator. By identifying the expected observable states as functions of vehicle motion the designer can better understand the source of estimation error and find methods to mitigate its occurrence. The observability analysis of vSLAM has been well treated, however the majority of these analysis are limited to the much simpler 2D planner case [18], [37], [46]. Six degree of freedom (6DoF) SLAM observability analysis has also been treated, but generally includes range measurement along with the bearing measurements to aid the INS and the results are not applicable to vSLAM applications. The same procedure applied to range and bearing SLAM in [5], [24] is applied to 6DoF vSLAM observers here and results are compared.

The filter carries 15 egostate estimates including six states to estimate the gyro drifts and accelerometer biases. Drift and bias are not technically required to run the filter. They are used because estimating the drift and bias mitigates errors in the INS position and attitude solutions that are used in the constraint equation and make the observation linearizations more accurate. Because they are not truly needed, and for simplicity, they are not include in the observability analysis. Nomenclature for this section can be seen in Table IX.

Without further delving into the unit sphere development [43], it suffices to know the reduced form of the transition matrix, **F**, and linearized observation matrix

Table IX. Observability Analysis Nomenclature

Notation	Definition
$\mathbf{F}$	Transition matrix
$\mathbf{H}$	Observation matrix
$\mathbf{H}_{\delta\mathbf{r}}$	Position error observation submatrix
$\mathbf{H}_{\delta\boldsymbol{\alpha}}$	Attitude error observation submatrix
$\mathbf{H}_{\delta\boldsymbol{p}}$	Feature error observation submatrix
$[\times\mathbf{f}]$	Skew-symmetric specific force matrix
i	Feature indexing
j	Time indexing

$\mathbf{H} = [\mathbf{H}_1^T \cdots \mathbf{H}_N^T]^T$ for N visible targets. The $\mathbf{F}$ matrix from Chapter III is modified by removing the drift and bias components and written in the continuous time framework. At time j, $\mathbf{F}$ can be can be written as

$$\mathbf{F}^j = \begin{bmatrix} 0 & \mathbf{I} & 0 & 0 & \cdots & 0 \\ 0 & 0 & -[\times\mathbf{f}^j] & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \tag{5.1}$$

and $\mathbf{H}_i^j$ is

$$\mathbf{H}_i^j = [\mathbf{H}_{\delta\mathbf{r},i}^j \; 0 \; \mathbf{H}_{\delta\boldsymbol{\alpha},i}^j \; \cdots \; \mathbf{H}_{\delta\boldsymbol{p},i}^j \; \cdots\,] \tag{5.2}$$

where $\mathbf{H}_{\delta\mathbf{r},i}^j$, $\mathbf{H}_{\delta\boldsymbol{\alpha},i}^j$ and $\mathbf{H}_{\delta\boldsymbol{\rho},i}^j$ are the 2×3 observation submatrices.

1. Instantaneous Observability

The instantaneous observability of an n state system is the collection of feature states that are observable due to sensor measurements from a single time segment. The instantaneous observability matrix at time j can be computed as

$$\mathbf{O}^j = [(\mathbf{H}^j)^T,\ (\mathbf{H}^j\mathbf{F}^j)^T,\ (\mathbf{H}^j(\mathbf{F}^j)^2)^T \cdots (\mathbf{H}^j(\mathbf{F}^j)^{n-1})^T]^T \tag{5.3}$$

where the number of observable states is equal to the rank of $\mathbf{O}^j$. For this system $(\mathbf{F}^j)^m = 0$ for $m > 2$ and the instantaneous observability matrix for a single feature can be written as

$$\mathbf{O}^j = \begin{bmatrix} \mathbf{H}^j_{\delta\mathbf{r},1} & 0 & \mathbf{H}^j_{\delta\boldsymbol{\alpha},1} & \mathbf{H}^j_{\delta\boldsymbol{\rho},1} \\ 0 & \mathbf{H}^j_{\delta\mathbf{r},1} & 0 & 0 \\ 0 & 0 & -\mathbf{H}^j_{\delta\mathbf{r},1}[\times\mathbf{f}^j] & 0 \end{bmatrix} \tag{5.4}$$

The matrix $\mathbf{O}^j$ is 6×12 and clearly has rank six or less, meaning that at most six of the desired 12 filter states can be observed from a single time period for a single feature. Adding a second feature generates the following instantaneous observability matrix

$$\mathbf{O}^j = \begin{bmatrix} \mathbf{H}^j_{\delta\mathbf{r},1} & 0 & \mathbf{H}^j_{\delta\boldsymbol{\alpha},1} & \mathbf{H}^j_{\delta\boldsymbol{\rho},1} & 0 \\ \mathbf{H}^j_{\delta\mathbf{r},2} & 0 & \mathbf{H}^j_{\delta\boldsymbol{\alpha},2} & 0 & \mathbf{H}^j_{\delta\boldsymbol{\rho},2} \\ 0 & \mathbf{H}^j_{\delta\mathbf{r},1} & 0 & 0 & 0 \\ 0 & \mathbf{H}^j_{\delta\mathbf{r},2} & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{H}^j_{\delta\mathbf{r},1}[\times\mathbf{f}^j] & 0 & 0 \\ 0 & 0 & -\mathbf{H}^j_{\delta\mathbf{r},2}[\times\mathbf{f}^j] & 0 & 0 \end{bmatrix} \tag{5.5}$$

Now $\mathbf{O}^j$ is 12×15 but will have at most nine observable states (the skew-symmetric terms only provides two independent rows). The addition of more features will not help, increasing observable states by at most two while increasing the filter size by three states. The observability for a single time instance will be rank deficient by $N+4$ for $N \geq 2$ features.

The addition of a range measurement would lead to a 9×12 and a 12×15 matrix and eight and 11 observable states respectively [5]. For range and bearings SLAM the trend continues, and given appropriate vehicle motion the instantaneous observability will remain rank four deficient regardless of the number of features. In vSLAM, with no range sensor, requires at least two time steps to generate range and orientation values. Therefore, the estimator will always be severely rank deficient in terms of instantaneous observability, a far cry from range and bearings SLAM.

2. Total Observability

The total observability matrix (TOM) is a measure of the observable states from the first measurement time period to the jth measurement time period and can be represented as

$$\mathbf{O}^j_{TOM} = \begin{bmatrix} \mathbf{O}^1 \\ \mathbf{O}^2 e^{\mathbf{F}^1\Delta^1} \\ \vdots \\ \mathbf{O}^j e^{\mathbf{F}^{j-1}\Delta^{j-1}\ldots\mathbf{F}^1\Delta^1} \end{bmatrix} \tag{5.6}$$

where Δ^i is the time between measurements $i-1$ and i. For the single target case, $\mathbf{O}_{TOM}$ can be generated for two time periods using a Taylor series expansion of $e^{\mathbf{F}^1\Delta^1}$.

$$e^{\mathbf{F}^1\Delta^1} \approx \mathbf{I} + \mathbf{F}^1\Delta^1 \tag{5.7}$$

resulting in

$$\mathbf{O}_{TOM} = \begin{bmatrix} \mathbf{H}^j_{\delta\mathbf{r},1} & 0 & \mathbf{H}^j_{\delta\boldsymbol{\alpha},1} & \mathbf{H}^j_{\delta\boldsymbol{\rho},1} \\ 0 & \mathbf{H}^j_{\delta\mathbf{r},1} & 0 & 0 \\ 0 & 0 & -\mathbf{H}^j_{\delta\mathbf{r},1}[\times \mathbf{f}^j] & 0 \\ \mathbf{H}^{j+1}_{\delta\mathbf{r},1} & -\mathbf{H}^j_{\delta\mathbf{r},1}[\times \mathbf{f}_j]\Delta^j & \mathbf{H}^{j+1}_{\delta\boldsymbol{\alpha},1} & \mathbf{H}^{j+1}_{\delta\boldsymbol{\rho},1} \\ 0 & \mathbf{H}^{j+1}_{\delta\mathbf{r},1} & -\mathbf{H}^j_{\delta\mathbf{r},1}[\times \mathbf{f}^j]\Delta^j & 0 \\ 0 & 0 & -\mathbf{H}^{j+1}_{\delta\mathbf{r},1}[\times \mathbf{f}^{j+1}] & 0 \end{bmatrix} \quad (5.8)$$

The matrix will be rank nine when $\mathbf{f}^j \times \mathbf{f}^{j+1} \neq 0$ and noting $\mathbf{H}_{\delta\mathbf{r},1}$ and $\mathbf{H}_{\delta\boldsymbol{\alpha},1}$ vary with time provided there is some vehicle translation. This leaves the estimator three states short of full observability, the best result possible since there is no global reference.

In order to study the total observability rank of the system over time, a much more convenient format is available. If $null(\mathbf{O}^j) \subset null(\mathbf{F}^j)\ \forall j$ then $rank(\mathbf{O}^j_{TOM}) = rank(\mathbf{O}^j_{SOM})$, [24], where

$$\mathbf{O}_{SOM} = [(\mathbf{O}^1)^T(\mathbf{O}^2)^T \ldots (\mathbf{O}^j)^T]^T. \quad (5.9)$$

An equivalent requirement is the basis vectors of $\mathbf{O}^j$ must span the basis vectors of $\mathbf{F}^j$. For the two feature case it can be assumed that $[(\mathbf{H}^j_{\delta\mathbf{r},1})^T(\mathbf{H}^j_{\delta\mathbf{r},2})^T]^T$ to be rank three. The sixth through ninth rows of $\mathbf{O}^j$ span the rows of $\mathbf{F}^j$ associated with the identity matrix and the 10th through 15th rows of $\mathbf{O}^j$ will span the rows of $\mathbf{F}^j$ containing $[\times\mathbf{f}^j]$, allowing the use of $\mathbf{O}_{SOM}$ to analyze system observability rank for two or more features.

For two time steps and two features $\mathbf{O}_{SOM}$ can be written as

$$\mathbf{O}_{SOM} = \begin{bmatrix} \mathbf{H}^{j}_{\delta\mathbf{r},1} & 0 & \mathbf{H}^{j}_{\delta\boldsymbol{\alpha},1} & \mathbf{H}^{j}_{\delta\boldsymbol{\rho},1} & 0 \\ \mathbf{H}^{j+1}_{\delta\mathbf{r},1} & 0 & \mathbf{H}^{j+1}_{\delta\boldsymbol{\alpha},1} & \mathbf{H}^{j+1}_{\delta\boldsymbol{\rho},1} & 0 \\ \mathbf{H}^{j}_{\delta\mathbf{r},2} & 0 & \mathbf{H}^{j}_{\delta\boldsymbol{\alpha},2} & 0 & \mathbf{H}^{j}_{\delta\boldsymbol{\rho},2} \\ \mathbf{H}^{j+1}_{\delta\mathbf{r},2} & 0 & \mathbf{H}^{j+1}_{\delta\boldsymbol{\alpha},2} & 0 & \mathbf{H}^{j+1}_{\delta\boldsymbol{\rho},2} \\ 0 & \mathbf{H}^{j}_{\delta\mathbf{r},1} & 0 & 0 & 0 \\ 0 & \mathbf{H}^{j+1}_{\delta\mathbf{r},1} & 0 & 0 & 0 \\ 0 & \mathbf{H}^{j}_{\delta\mathbf{r},2} & 0 & 0 & 0 \\ 0 & \mathbf{H}^{j+1}_{\delta\mathbf{r},2} & 0 & 0 & 0 \\ 0 & 0 & -\mathbf{H}^{j}_{\delta\mathbf{r},1}[\times \mathbf{f}^{j}] & 0 & 0 \\ 0 & 0 & -\mathbf{H}^{j+1}_{\delta\mathbf{r},1}[\times \mathbf{f}^{j+1}] & 0 & 0 \\ 0 & 0 & -\mathbf{H}^{j}_{\delta\mathbf{r},2}[\times \mathbf{f}^{j}] & 0 & 0 \\ 0 & 0 & -\mathbf{H}^{j+1}_{\delta\mathbf{r},2}[\times \mathbf{f}^{j+1}] & 0 & 0 \end{bmatrix} \tag{5.10}$$

The first eight rows can contain at most six independent rows, the next eight can contain three more, and the last eight hold the final the independent rows assuming $\mathbf{f}^{j} \times \mathbf{f}^{j+1} \neq 0$, for a total of at most 12 independent rows. Although they are in different coordinate frames than typical SLAM, $\mathbf{H}^{j}_{\delta\mathbf{r},i}$ and $-\mathbf{H}^{j}_{\delta\boldsymbol{\rho},i}$ are still equivalent up to a coordinate transform, representing errors associated with the relative position of the UGV and the feature, and this generates the rank three deficiency. For two time steps the filter should generate full observability, up to relative position, of the system.

These results mirror that of the range and bearings SLAM problem; however, in vSLAM the range estimates are a effectively the result of synthetic stereo. Analytically the vSLAM system may be fully observable; however, with a forward facing camera the stereo baseline used to generate the range estimate is often far too small with respect to the bearing measurement errors to obtain practical observability of the system. High update rates on the camera are valuable as a method of averaging down measurement error, but not valuable in regard to information content from one image to the next. Unlike range and bearings SLAM, vSLAM will gain little to no practical observability of feature range over short time intervals. This implies that observability of velocity, and therefore position, may be particularly weak in the direction the camera faces, lending to the overconfidence of the filter for nearly straight paths.

The observability in the direction of travel would be greatly improved if the camera was turned to the side. However, the observability in the direction the camera is now facing is severely reduced. Camera angles of 10, 20, and 45 degrees where simulated in the same manner as as the zero degree camera offset in Chapter IV, and the new camera configuration did not improve performance. In fact, the larger the angular offset the worse the results. Observability analysis suggests even though the observability in the direction of travel is improved now that motion is somewhat orthogonal to the camera orientation, the lack of motion in the direction the camera now faces makes the system highly subject to IMU biases. Because the observability is weak in the direction the camera faces, the reduced motion in that direction further degrades estimation accuracy.

Because the concern is always that biases will accumulate in the direction the camera faces, a forward facing camera may be the best option. With a forward facing camera, the minimal orthogonal motions, and associated biases, becomes highly observable. Again, this leaves the direction of travel less observable, but this configuration also generates the largest possible signals in the direction the camera faces. This helps identify the final IMU bias and appears to be the best compromise.

To better identify specifically which states are and are not observable the observability Grammian can be applied, which is constructed from either the instantaneous or total observability matrices as

$$\mathbf{Q} = \mathbf{O}^T\mathbf{O} \tag{5.11}$$

The eigenvectors associated with the zero eigenvalues of $\mathbf{Q}$ are the unobservable subspace of the filter states. Eigenvectors with large eigenvalues are strongly observable. Based on the analytical form of $\mathbf{Q} = {\mathbf{O}_{TOM}}^T\mathbf{O}_{TOM}$, the resulting zero eigenvalues associate with linear combinations of errors in the vehicle positions and feature states. There is no manner in which the vehicle can translate or rotate, regardless of features or camera angles, to provide full observability of the vehicle position. This is due the lack of a global reference, the system will be observable up to the local frame as expected. However, the velocities and attitude states are in the observable space so with proper vehicle motion those states are observable. With full observability of velocities and attitudes, drift in the vehicle position estimates can be mitigated, but not eliminated, leading to long term accuracy of the egostate estimation.

3. Observability Recap

The unobservable subspace associated with the instantaneous observability Grammian will be of rank six or higher. For N features vSLAM will be $N + 4$ states short of full observability for $N \geq 2$. In range and bearings SLAM, only a single subspace associated with velocity and/or attitude will be unobservable in the instantaneous case and [5] analytically identified that subspace as a function of the current vehicle motion. For vSLAM there will be $N + 1$ unobservable subspaces that include velocity and/or attitude, making similar analysis of the unobservable states significantly more complicated. A more comprehensive study is warranted in the hope of producing a corollary result for vSLAM, one that may be used in path planning. When considering the total observability over multiple camera measurements, vSLAM gains full local observability. However, as discussed before, the range observability is a function of camera baseline, and often remains minimally observable due to bearings sensor error.

D. Path Selection via Acceleration Monitoring

The concept is for the vehicle to vary its path and monitor the total acceleration (control) commands issued by the guidance laws in order to identify the preferred path. Unfortunately the vehicle does not know the optimal total acceleration values, so the relative changes in acceleration as a function of path is monitored. Taking inspiration from the results seen in Figs. 36 through 38 of Chapter IV, it appears that in the sinusoidal case, the optimal estimation results coincide nicely with the spatial period that produced the minimal total accelerations. A smaller spatial period may generate better estimation results, but it was shown there would be only minor improvement at the expensive of a significant acceleration increase. Longer spatial periods quickly incur larger estimation errors.

To demonstrate the merits of this approach, the UGV errors on the side of estimation accuracy and begins moving with a relatively small spatial period, and slowly increase the period, Fig. 41. The concept is to reduce nominal accelerations until the total acceleration commands produced begin to level off. The results from Chapter IV are based on the total accelerations for the life of the path and not the acceleration at any given point so a running average will be taken.

The resulting acceleration profile for the path in Fig. 41 can be seen in Fig. 42. The blue lines are the instantious acceleration commands as functions of forward progress, and are clearly too noisy to be an effective indicator or performance. The red line is the running average acceleration over the previous half period (i.e. averaged results will contain data from the current and previous spatial period portion of the run, the window size is designed to average the entire previous half period), and is very smooth. The plot shows dramatic reductions in acceleration commands over the first 10 meters, with a smaller reduction between 10 and 20 meters of forward progress. After the 20 meter mark, little to no reduction is seen in the acceleration commands.

At 20 meters of forward progress down the hallway the simulation is using a 15 meter spatial period path, that appears to be the ideal spatial period in terms of acceleration requirements, it also matches the earlier results seen in Fig. 34. This demonstrates that the acceleration required for a single half period is representative of the average acceleration for the particular spatial period over the course of a 75 meter run. This is again verified in Fig. 43 where the discrete average acceleration per half period is shown as a function of the spatial period and closely matches the profiles from Chapter IV.

If a vehicle carefully monitors the its acceleration profile there is the potential to exploit the information to determine if more or less nominal control should be applied. This requires a very rapid assessment in a hallway environment, but would easily apply to UAVs, especially high-fliers, who have long flight times in which to identify and use the appropriate flight profile. This approach is crude, but is more desirable than simply setting path parameters without any additional information. In the previous example it was clear in the first 10 meters of hallway that the control cost could be reduced to less than half the original value without drastically affecting estimate quality.

The same simulation is run but with 7.5 meters of sight distance. The results are shown in Fig. 44 where the ideal acceleration commands were reached at a much

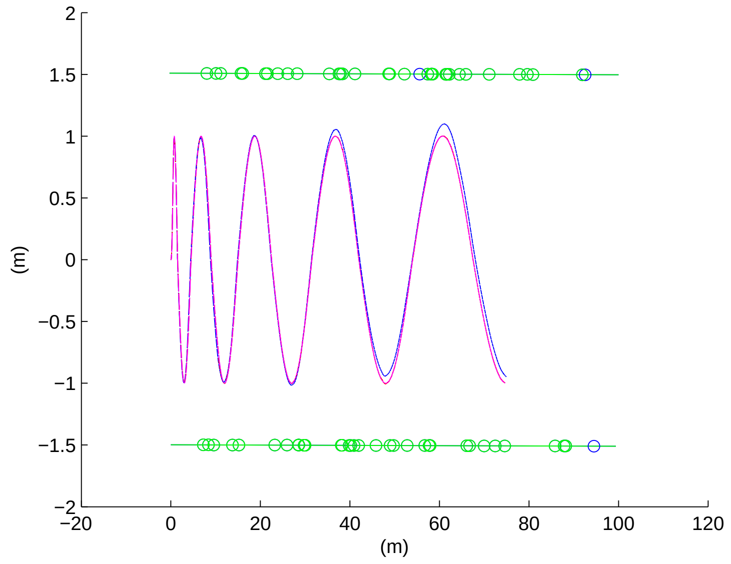


Fig. 41. Sinusoidal path with one meter amplitude, using closed-loop guidance, 25 meter sight distance, and increased spatial period every half cycle. Magenta lines are the desired vehicle position, red lines are the estimated vehicle position, and blue lines are the true vehicle position.

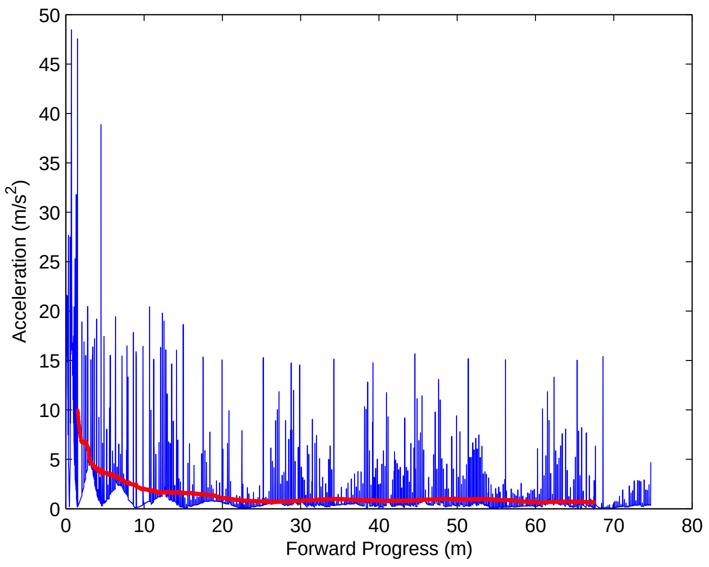


Fig. 42. Acceleration values as function of forward progress in blue, smoothed acceleration profile in red, for a sinusoidal path using closed-loop guidance, one meter amplitude, 25 meter sight distance, and increased spatial period every half cycle.

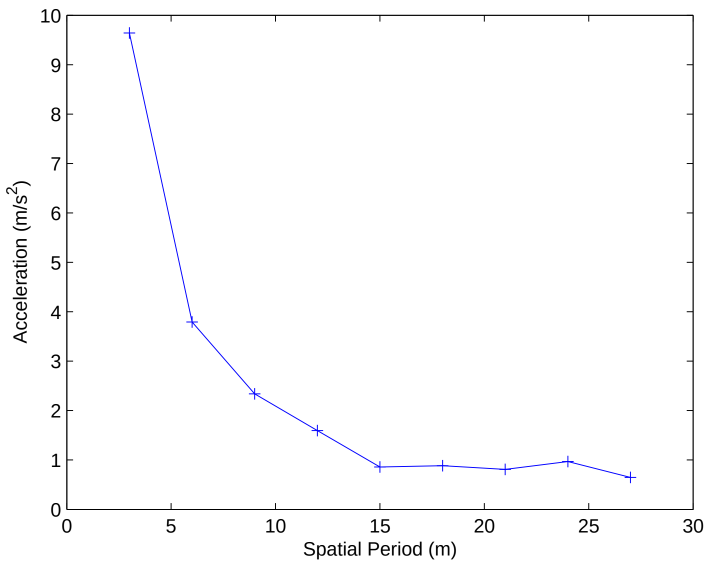


Fig. 43. Average acceleration of each half period as a function of spatial period for sinusoidal path using closed-loop guidance, one meter amplitude, 25 meter sight distance, and increased spatial period every half cycle.

earlier point. The ideal values appears to be near 10 meters of forward progress that corresponds to a nine meter spatial period. Similarly, by adjusting the path amplitude the desired spatial period is also affected, Fig 45, although less dramatically. This demonstrates that the acceleration commands are sensitive to the differing environment and sensor capabilities.

In an unknown or changing environment, monitoring the acceleration itself may prevent the use of too little or too much acceleration while maintaining solid estimator accuracy. Methodical alterations of nominal acceleration could be performed to identify a safe, yet cost effective path. Or, intermittent spot checks could be performed to see if increasing or decreasing nominal acceleration rates is warranted.

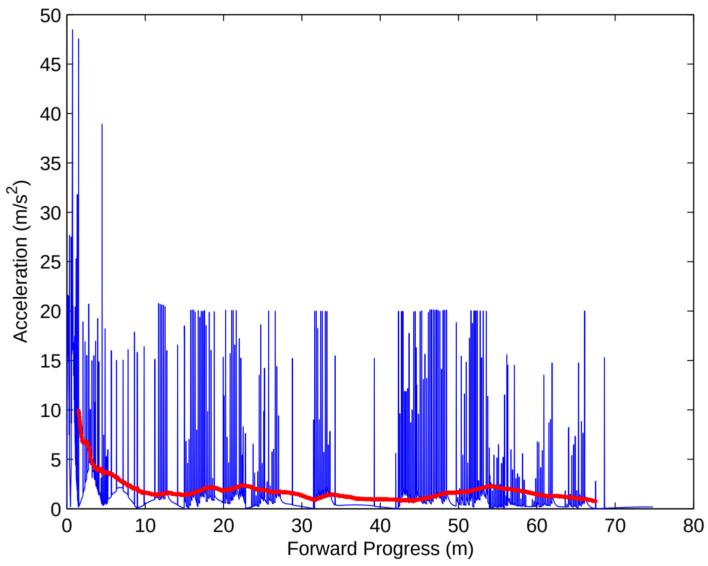


Fig. 44. Acceleration values as function of forward progress in blue, smoothed acceleration profile in red for a sinusoidal path using closed-loop guidance, one meter amplitude, 7.5 meter sight distance, and increased spatial period every half cycle.

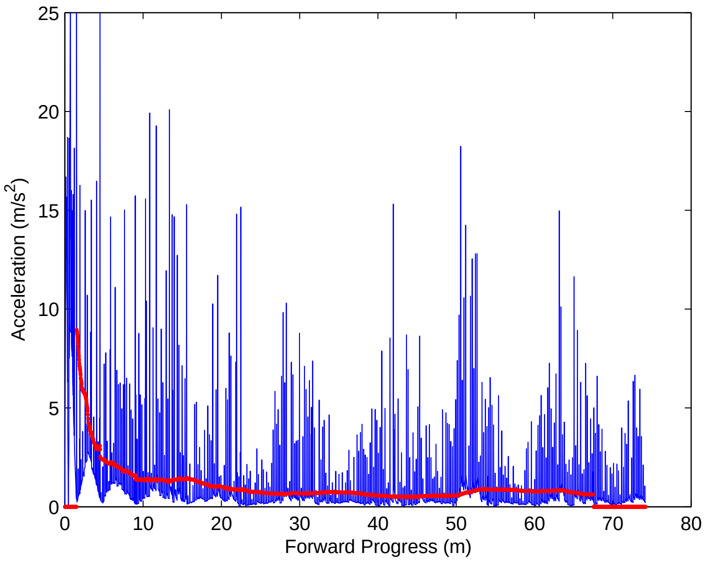


Fig. 45. Acceleration values as function of forward progress in blue, smoothed acceleration profile in red for a sinusoidal path using closed-loop guidance, 0.5 meter amplitude, 25 meter sight distance, and increased spatial period every half cycle.

E. Summary

Currently covariance based path optimization appears to be a misguided approach. Therefore, a predetermined path approach would be advisable. The choice of which prescribed path (a moderately integrated approach) may be obtainable by monitoring the corrective acceleration terms as assigned by the guidance law. For a truly integrated estimation and control approach either methods for more reliable covariance estimates or alternative path selection criterion must be investigated.

Based on the observability analysis, the vSLAM approach is significantly less observable over a single measurement time period than standard SLAM, because three states are carried per feature, and only two measurements are generated. The total observability results match for two or more time instances with appropriate vehicle excitation and the vehicle has full observability up to the local frame. The difference lies in relying on artificial stereo to act as the range sensor as opposed to an actual range measurement, this suggests that even though the vSLAM system analytically gains the required observable states over a second time epoch, many of those states are not practically observable. Additionally, the range estimates comes from multiple measurement times, and as such is subject to multiple linearization points, leading to an artificial inflation of the observability matrix.

A more in depth investigation into the specific unobservable states could yield methods for path optimization/selection by either improving observability or mitigating the filter overconfidence [18] and allowing covariance to be used. Another possible approach is to move to a higher-order or particle filter approach to better handle the nonlinearities in the system, especially those due to errors in the range estimates [29].

Acceleration monitoring could be a viable method to identify an appropriate spatial period for a given amplitude path. The method is responsive to environment changes, such as feature availability, sensor quality and path amplitude, and provides suitable results. As indicated in earlier sections the amplitude of sinusoidal paths should be chosen to be large, because it generally improves estimates and has little effect on control cost for most spatial periods. The corollary approach, acceleration monitoring to determine desired amplitude for a given spatial period, is not recommended due to the smaller discrepancies in performance.

CHAPTER VI

FILTER ORDER REDUCTION

A. Introduction

The unique observation structure of the unit sphere based filter can be exploited in order to reduce the filter state vector size. Due to the use of measurements to directly initialize feature error states a Schmidt-Kalman filter can be successfully implemented. Simulations validate the results comparing them to the full vSLAM implementation used in earlier chapters. The Schmidt-Kalman filter was introduced to account for, but avoid estimating, biases or other model uncertainties, especially those with poor observability, in large systems when simply ignoring those uncertainties lead to unacceptable estimation accuracy. This method also reduced the computational load on the estimator because less states are being propagated and updated in the filter [41].

Because the bearing error covariances are directly initialized as the measurement covariance (mapped back to the initial reference frame) and those states experience very small updates over the life of the filter, it suggests the filter may not need to estimate the states. However, simply dropping the states out of the estimator completely and using the measurements “as is” with no regard of measurement error results in unacceptable estimator performance. Implementing the Schmidt-Kalman filter avoids estimating the initial bearings states and simply takes into account the uncertainty in the sensor measurement. This approach will not work in a standard SLAM or vSLAM application because the states maintained for each feature are generally the Euclidean coordinates of the feature in world frame. Clearly, one would be unable to remove any of those states from a filter, making this application of a

Schmidt-Kalman filter to this vSLAM application rather unique.

B. Schmidt-Kalman Filter Development

When estimating using an Extended Kalman Filter (EKF) there may be particular uncertainties that when ignored can cause excessive errors or overconfidence in the estimator. However, the inclusion of these uncertainties as filter states may be undesirable due to their unobservable nature or the dramatic increase to the computational complexity of the filter. The Schmidt-Kalman filter, also known as the "Consider" Kalman filter, allows the filter to account for the influence of those uncertainties without the need to estimate their values [34], [51].

The development in [4] is used and starts with the standard Kalman Filter gain and update equations.

$$\mathbf{K} = (\mathbf{P}^{-}\mathbf{H}^{T})\boldsymbol{\alpha}^{-1} \tag{6.1}$$

$$\boldsymbol{\alpha} = (\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{T}) \tag{6.2}$$

$$\bar{\mathbf{x}}^{+} = \bar{\mathbf{x}}^{-} + \mathbf{K}(\mathbf{z} - \mathbf{H}\bar{\mathbf{x}}^{-}) \tag{6.3}$$

$$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P} \tag{6.4}$$

Consider filters partition the filter states into states to be estimated and those to be considered, $\bar{\mathbf{x}} = [\mathbf{x}|\mathbf{y}]^{T}$. This alters the process and measurement models as well, most notably $\mathbf{z} = [\mathbf{H}_x|\mathbf{H}_y][\mathbf{x}|\mathbf{y}]^{T}$ where $\mathbf{H}_x$ and $\mathbf{H}_y$ are the portions of the original $\mathbf{H}$ matrix associated with states now being estimated and those just being considered. The covariance matrix $\mathbf{P}$ is also redefined in terms of $\mathbf{P}_x$, covariance of states to be estimated, $\mathbf{P}_{xy}$ and $\mathbf{P}_{yx}$, cross covariance of the states to be estimated and those to be considered, and $\mathbf{P}_y$, the covariances of states to be considered.

The new gain and update equations for the Schmidt-Kalman filter are as follows

$$\mathbf{K}_x = (\mathbf{P}_x^- \mathbf{H}_x^T + \mathbf{P}_{xy}^- \mathbf{H}_y^T)\boldsymbol{\alpha}_x^{-1} \tag{6.5}$$

$$\begin{aligned}\boldsymbol{\alpha}_x =&(\mathbf{H}_x \mathbf{P}_x^- \mathbf{H}_x^T + \mathbf{H}_x \mathbf{P}_{xy}^- \mathbf{H}_y^T + ...\\ &\mathbf{H}_y \mathbf{P}_{yx}^- \mathbf{H}_x^T + \mathbf{H}_y \mathbf{P}_y^- \mathbf{H}_y^T + \mathbf{R})\end{aligned} \tag{6.6}$$

$$\mathbf{x}^+ = \mathbf{x}^- + \mathbf{K}_x(\mathbf{z} - \mathbf{H}_x \mathbf{x}-) \tag{6.7}$$

$$\mathbf{y}^+ = \mathbf{y}^- \tag{6.8}$$

$$\mathbf{P}_x = (\mathbf{I} - \mathbf{K}_x \mathbf{H}_x)\mathbf{P}_x^- - \mathbf{K}_x \mathbf{H}_y \mathbf{P}_{xy}^- \tag{6.9}$$

$$\mathbf{P}_{xy} = (\mathbf{I} - \mathbf{K}_x \mathbf{H}_x)\mathbf{P}_{xy}^- \mathbf{K}_x \mathbf{H}_y \mathbf{P}_y^- \tag{6.10}$$

$$\mathbf{P}_{yx} = \mathbf{P}_{xy} \tag{6.11}$$

$$\mathbf{P}_y = \mathbf{P}_y \tag{6.12}$$

C. Schmidt-Kalman Filter Applied to the Unit Sphere Constraint Equations

In the Schmidt-Kalman filter framework the filter includes the uncertainties of the initial bearings measurements into the state update equations without having to update the bearings themselves. The inverted matrix $\boldsymbol{\alpha}$ will still be size $m \times m$ for m measurements, however the resulting gain matrix, $\mathbf{K}$, will reduce in size significantly based on the number of former states now only being considered. For an original n state filter, if r states are kept and s are considered, $\mathbf{K}$ will reduce from an $n \times m$ matrix to an $r \times m$ matrix. For unit sphere, 6DoF vSLAM two out of three feature states can be considered, decreasing the size of the state vector and gain matrix by nearly 2/3rds. Further savings may also be found in calculation redundancies in the Schmidt-Kalman filter update laws.

In the general case, the $\mathbf{P}_y$ matrix is diagonal because $\mathbf{y}$ is often model parameter uncertainties which are independent. The $\mathbf{P}_{xy}$ and $\mathbf{P}_{yx}$ matrices begin as zero matrices, then build up a covariance between considered states and the estimated states over time. In the unit sphere case each new features initial bearings and range estimate are mapped back to the original vehicle location and orientation. This mapping inherently builds covariance between the new feature states and the existing states. Therefore the filter will have a full $\mathbf{P}_y$ matrix that will not be updated and a non-zero initial $\mathbf{P}_{xy}$ and $\mathbf{P}_{yx}$ matrices that are updated.

There will be available data going unused in the implementation of the Schmidt-Kalman filter, but the computational savings may be worth the trade off in accuracy. A thorough investigation into computational savings would require careful optimization of both the full and Schmidt-Kalman filters, and is not performed here. In this case only the estimation accuracy of a vSLAM application is investigated.

Using the same simulation as Chapter IV the Schmidt-Kalman filter is applied with an open-loop controller. Bearing measurements are still updated while the range estimate is being initialized, and once the feature is fully initialized into the filter the bearings states are dropped out of the state vector and are simply considered from that point on.

Comparing the Schmidt-Kalman results in Figs. 46 and 47 to the open-loop results of Chapter IV seen in Figs. 25 and 26, the Schmidt-Kalman filter does not perform as well as the full implementation. The acceptable control region (1% or smaller error) is reduced to about half the size, but the filter does achieve more than a 50% reduction in state vector size.

The closed-loop simulation results for the Schmidt-Kalman filter are shown in Figs. 48 through 51. Most notably, the 1% or smaller error region is dramatically increased and closely matches the full EKF case, Figs. 32 and 33. It is of interest that the 1% or less control region increases between the open and closed-loop Schmidt-

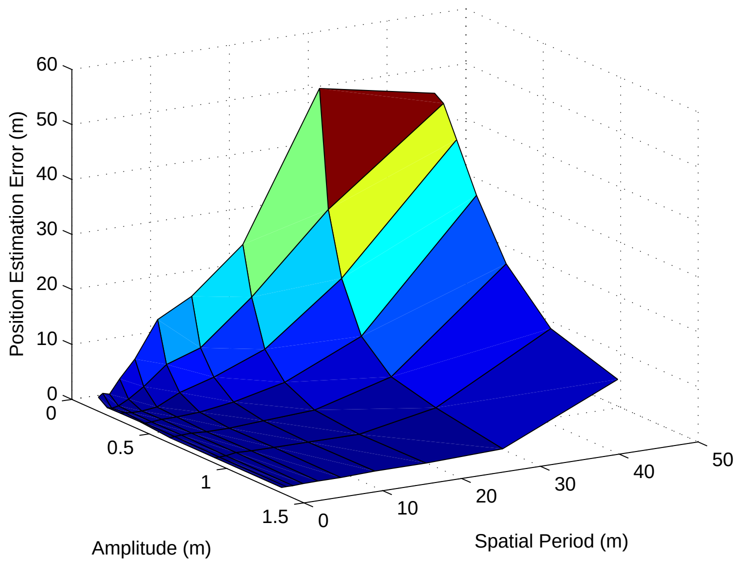


Fig. 46. Schmidt-Kalman filter: Sinusoidal path parameters vs. estimation error using open-loop guidance, 80 features, and 25 meter sight distance.

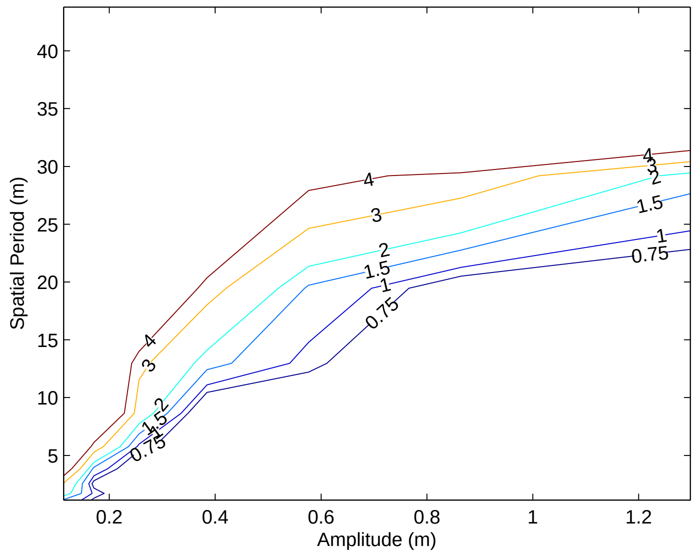


Fig. 47. Schmidt-Kalman filter: Sinusoidal path error contours using open-loop guidance, 80 features, and 25 meter sight distance.

Kalman filter case while the full EKF implementation showed almost no change from open to closed-loop. Also, for closed-loop control, the total accelerations required is, on average, double for the Schmidt-Kalman filter compared with the full EKF implementation.

It is suspected the additional accelerations generated by the closed-loop control was required for the Schmidt-Kalman filter to provide sufficient information to update filter states. With a fixed covariance for bearing error, after a feature has been seen for a period of time the discrepancies in the constraint equations must be larger for the reduced order case than the full implementation before updates to states are made. Otherwise the errors in the constraint equation could be accounted for by the bearing uncertainty and no action would be taken by the filter. The additional accelerations revealed errors in the constraint equations that could not be accounted for by the bearing uncertainty, allowing the filter to updated vehicle states more effectively and producing better egostate estimate accuracy.

D. Summary

The filter state reduction of a vSLAM application is a novel result due to the unit sphere observation model. While not validating the choice of a Schmidt-Kalman filter over the standard EKF formulation, the Schmidt-Kalman filter was shown to have similar accuracy in close-loop simulation at the expense of additional control cost. Depending on the vehicle and computers being used there may in fact be advantages as nearly a 2/3rds reduction in filter size was achieved.

While untested, there is the potential to use the unit sphere constraints not just for vSLAM but for SLAM applications as well. Effectively reversing the constraint equations used here and using current range measurements with current bearings at

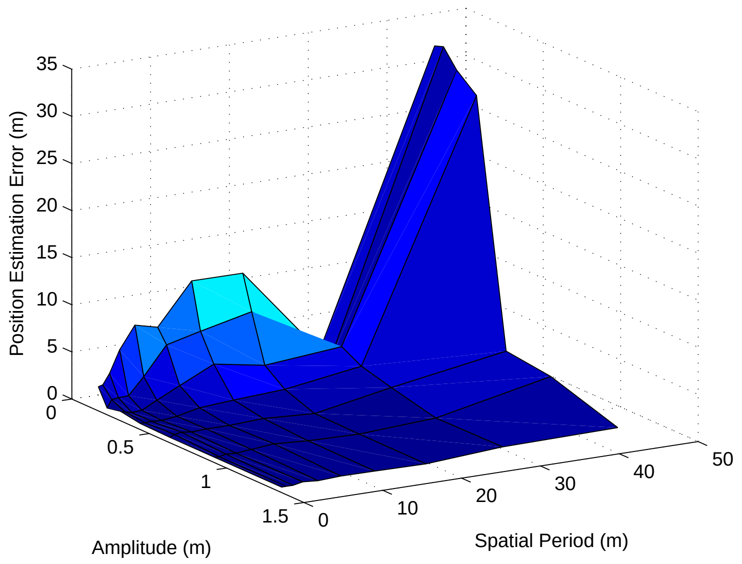


Fig. 48. Schmidt-Kalman filter: Sinusoidal path parameters vs. estimation error using closed-loop guidance, 80 features, and 25 meter sight distance.

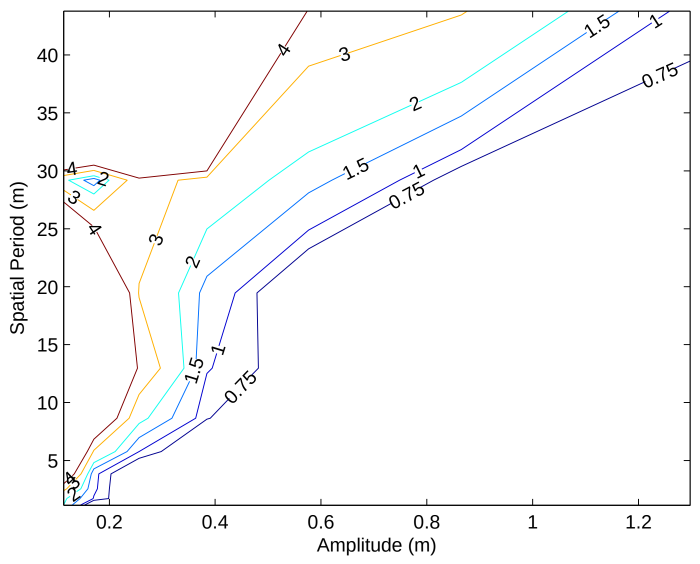


Fig. 49. Schmidt-Kalman filter: Sinusoidal path error contours using clsoed-loop guidance, 80 features, and 25 meter sight distance.

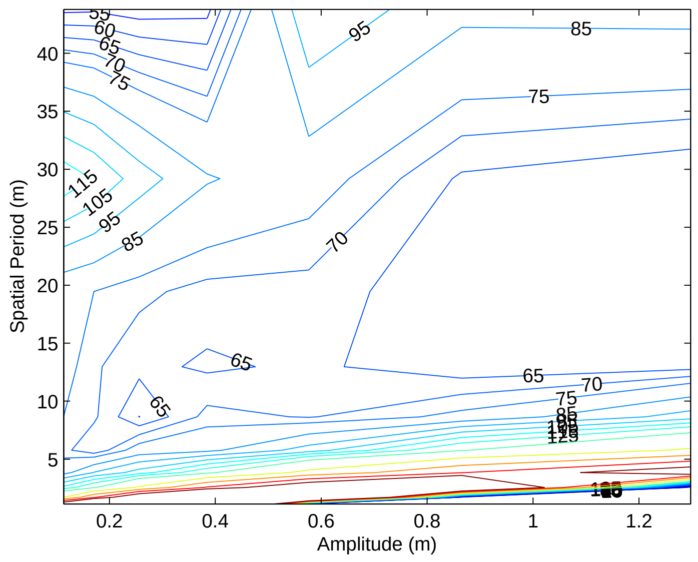


Fig. 50. Schmidt-Kalman filter: Sinusoidal path parameters vs. total acceleration contours using closed-loop guidance, 80 features, and 25 meter sight distance.

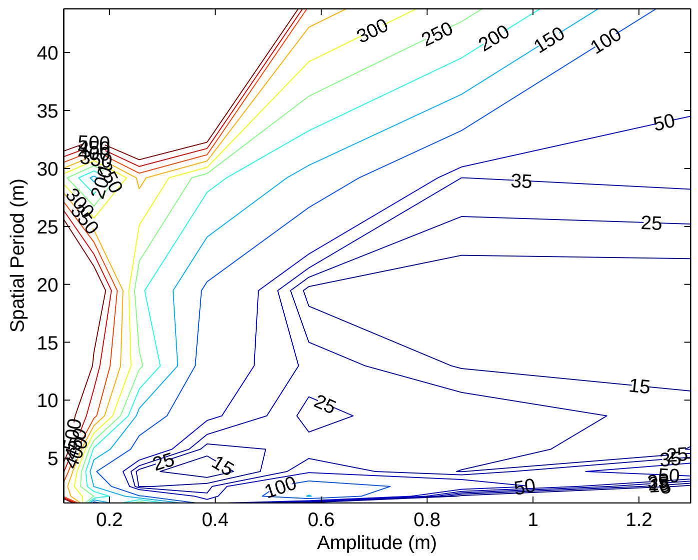


Fig. 51. Schmidt-Kalman filter: Sinusoidal path parameters vs. total acceleration multiplied by estimation error contours using closed-loop guidance, 80 features, and 25 meter sight distance.

each time update the filter would only need to keep initial bearings as filter states. This may not be a efficient utilization of resources compared to standard SLAM methods because it only produces two measurement equations per feature, and not three. However, because range measurements are taken at each update, only two states, the initial bearings, are are needed. The unit sphere approach with range would then allow the Schmidt-Kalman filter to be applied to the initial bearings states, removing feature states from the estimator entirely, resulting in a 15 state SLAM application.

CHAPTER VII

A HIGHER-ORDER METHOD FOR COOPERATIVE, DECENTRALIZED SOURCE LOCALIZATION

A. Introduction

Cooperative agent-based methods are used to generate solutions to many problem types. The solution methods are referred to as cooperative because they are carried out by a team of agents where the sharing of information is paramount. Although the types of problems vary widely, a common feature of these methods is the sharing of measurement information among team members. Commonly, the autonomous agents, robotic or otherwise, use the shared information to create a mathematical model and identify possible solutions to difficult problems. Based on these test solutions the agents update their position in the solution space, resample, and re-evaluate those possible solutions until an acceptable solution is attained. In many instances, the agents work in a decentralized manner; that is, after information is exchanged, the agents individually create models, identify possible solutions, update, etc.

An application of the above general procedure is the cooperative localization of optima or iso-contours of unknown environmental fields, such as odor plumes [21], [49], acoustic sources [6], or (underwater) chemical plumes or heat sources [23]. In these scenarios a source emits a measurable scalar field and a team of robotic agents measure the scalar field. Subsequent to sharing their position measurements and scalar information among themselves, the agents individually determine their position update, which they believe will take them closer to the unknown source location, [14], [36]. The robotic agent controls are categorically represented as decentralized feedback controls.

There has been a lot of work done using gradient based approaches for source localization, often with a single vehicle, [21], [39], and the results show that multiple sensors on a single vehicle can work but are often inadequate because of the close proximity of the onboard sensors. It has also been demonstrated that multiple vehicle teams that share and integrate information can increase the efficiency and robustness of the system [11], [16]. Using multiple agents with single sensors effectively extends the distance of sensors coverage, and because each agent can move independently, the multi-agent system is exploited by allowing each agent to update based on its quadratic approximation to the environment.

In earlier work, a decentralized control method for swarms of agents using a quadratic model approximation was developed to identify possible source locations [19]. The quadratic approximation approach is attractive because it leads to a linear update equation for the agent positions. In this current study, the quadratic model is replaced by higher-order approximations to help identify possible source locations. This creates a nonlinear expression for the agent position updates, which is solved using a Lagrangian expansion technique [1]. The motivation for investigating a higher-order methods is to attain improved performance in the cooperative source localization problem without a significant increase in computational complexity.

B. Review of the Quadratic Model Approach

For simplicity, consider a group of robotic agents that operate in the plane. Their task is to localize a source that emits a measurable scalar field, $F(x, y)$. Assume that the source appears as a minimum of the field, so from an optimization perspective, the function $F(x, y)$ becomes an objective function to be minimized. This assumption is in keeping with many applications; for example, chemical plumes, and light and acoustic beacons satisfy this property [7], [35]. It is assumed that the agents can evaluate the field via a sensor (which amounts to a function evaluation) and can communicate their positions (or locations) and sensor measurements to neighboring agents.

To simplify the discussion, it is assumed that the agents are in a region where the Hessian matrix of the objective function remains positive definite and the objective function decreases with a unit step along the search direction. Several useful modifications to the eventual update algorithm are presented in [19] for the case that these conditions are not satisfied. The control law for the ith agent begins with a local quadratic approximation to the true field.

$$\begin{aligned} F(x,y) \approx a_0 + a_1(x-x_*) + a_2(y-y_*) + \frac{1}{2}a_3(x-x_*)^2 \\ + a_4(x-x_*)(y-y_*) + \frac{1}{2}a_5(y-y_*)^2 \end{aligned} \tag{7.1}$$

In this equation, $F(x,y)$ represents the true field, the pair (x_*, y_*) represents the location of the ith agent, and a_j are unknown coefficients that define the quadratic approximation. This equation can be written in a convenient matrix form.

$$F(\mathbf{q}) \approx a_0 + \mathbf{g}^T\mathbf{q} + \frac{1}{2}\mathbf{q}^T\mathbf{H}\mathbf{q} \tag{7.2}$$

Here, $\mathbf{q} = [x-x_*,\ \ y-y_*]^T$, $\mathbf{g} = [a_1,\ \ a_2]^T$, and $\mathbf{H} = [a_3,\ \ a_4;\ \ a_4,\ \ a_5]$. The unknown coefficients a_j appear linearly and therefore they can be easily determined by fitting the quadratic model to the sensor measurements of neighboring agents. The gathered set of equations can be written in one matrix form, such as $\mathbf{Da} = \mathbf{f}$, where $\mathbf{D}$ is a $P \times 6$ matrix and p is the number of measurements; $\mathbf{a}$ is a 6×1 column arrangement of the unknown coefficients a_j; and $\mathbf{f}$ is a $p \times 1$ column arrangement of corresponding sensor readings. A least square solution is applied if $p > 6$.

Once the coefficients defining the quadratic model are computed, the position update for the ith agent is determined from equating the gradient of the field approximation to zero. This gives an update equation for the ith agent that depends on the model.

$$[x_{*,k+1}, \;\; y_{*,k+1}]^T = [x_{*,k}, \;\; y_{*,k}]^T - \mathbf{H}_k^{-1}\mathbf{g}_k \tag{7.3}$$

The position update marks the presumed source location, which is the minimum of the quadratic model that best fits the data supplied by the neighboring agents. The index k denotes the current position and information whereas $k+1$ denotes the new position.

Notationally, equation (7.3) can be written as $\boldsymbol{\delta}_{k+1} = \boldsymbol{\delta}_k - \mathbf{H}_k^{-1}\mathbf{g}_k$. Notice that $\mathbf{H}_k$ is the Hessian matrix of the local quadratic approximation associated with the current location of the ith agent, whereas $\mathbf{g}_k$ is the corresponding gradient vector. This update equation is the basis of the quadratic method of decentralized control for a swarm of agents performing source localization: each agent uses neighboring agent information to determine the coefficients a_j and then updates its position accordingly. The method is seen as a variation on the theme of quasi-Newton methods for function minimization, or perhaps a Newton-type method [32] that approximates both the Hessian and gradient by finite differences.

Once the agents have updated their positions, they can resample and reapply equation (7.1) for each agent. Many implementation details of this method, such as dealing with non-positive definite Hessians and constrained updates, are omitted here but are addressed in [19]. The steps to implementing the Quadratic Model Approach are given below:

1. Each agent evaluates the field with its sensor and communicates their current positions and sensor measurements to neighboring robots. The communication

can be done wirelessly.

2. Each agent accepts the shared information and fits the neighboring agent data to a quadratic model in matrix form, $\mathbf{Da} = \mathbf{f}$.

$$\mathbf{D} = \begin{bmatrix} 1 & (x_1 - x_i) & (y_1 - y_i) & \cdots & \frac{1}{2}(y_1 - y_i)^2 \\ 1 & (x_2 - x_i) & (y_2 - y_i) & \cdots & \frac{1}{2}(y_2 - y_i)^2 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & (x_p - x_i) & (y_p - y_i) & \ddots & \frac{1}{2}(y_p - y_i)^2 \end{bmatrix}$$

$$\mathbf{a} = [\ a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5\ ]^T$$

$$\mathbf{f} = [\ F(x_1, y_1) \quad F(x_2, y_2) \quad \ldots \quad F(x_p, y_p)\ ]^T$$

The index p denotes there are p participating agents, i.e., the ith agent and $p - 1$ neighboring agents. The various agent locations help compose the matrix $\mathbf{D}$, whereas the corresponding sensor readings help compose the vector $\mathbf{f}$. The unknown coefficients a_j are determined from this equation using the principle of least squares, which an agent's micro controller can perform using QR factorization or any other robust method.

3. Once the coefficients of $\mathbf{a}$ are determined, the gradient vector $\mathbf{g} = [a_1,\ a_2]^T$, and Hessian matrix $\mathbf{H} = [a_3,\ a_4;\ a_4,\ a_5]$, are formed and the agent can update its position according to equation (7.3), which is restated here.

$$[x_{*,k+1},\ y_{*,k+1}]^T = [x_{*,k},\ y_{*,k}]^T - \mathbf{H}_k^{-1}\mathbf{g}_k \tag{7.4}$$

C. A Higher-order Extension

The quadratic approximation can be extended in a straightforward way, starting with a higher-degree polynomial representation of the scalar field for the ith agent.

$$\begin{aligned} F(x,y) \approx a_0 + a_1(x-x_*) + a_2(y-y_*) + a_3(x-x_*)^2 \\ + a_4(x-x_*)(y-y_*) + a_5(y-y_*)^2 \\ + a_6(x-x_*)^3 + a_7(x-x_*)^2(y-y_*) + \ldots \end{aligned} \tag{7.5}$$

As before, the pair (x_*, y_*) represents the location of the ith agent and a_j are unknown coefficients that define the polynomial approximation. The order of the polynomial representation is given as n. The process for developing an update equation for the ith agent follows the steps that were performed for the quadratic method. A group of neighboring agents communicate their sensor readings and locations to the ith agent, and because the unknown coefficients a_j appear linearly, they can be determined by fitting the higher-order model to the neighboring true sensor measurements. Again, the set of equations can be written as a matrix equation $\mathbf{Da} = \mathbf{f}$. Where $\mathbf{D}$ is a $p \times M(n)$ matrix, p is the number of neighboring measurements, and $M(n)$ is the number of unknown coefficients; $\mathbf{a}$ is a $M(n) \times 1$ column arrangement of the unknown a_j coefficients and $\mathbf{f}$ is a $p \times 1$ column arrangement of corresponding sensor readings. The number of unknown coefficients depends on the order of the polynomial representation.

$$M(n) = \frac{1}{2}(n-1)(n+4) + 3 \tag{7.6}$$

It should be noted that $M(n)$ is the minimum number of data points the ith agent requires to create its nth order model. This does not necessarily require $M(n)$ agents, but rather $M(n)$ viable data points, some of which could possibly be maintained in memory assuming recent measurements are still accurate.

As before, once the coefficients that define the polynomial model are computed, the position update for the ith agent is determined from equating the gradient of the field approximation to zero. It is convenient to write the gradient using indicial notation, wherein an index that is repeated within a term is summed over its range.

$$0 = \boldsymbol{\phi}_i + \boldsymbol{\phi}_{ij}\mathbf{q}_j + \boldsymbol{\phi}_{ijk}\mathbf{q}_j\mathbf{q}_k + \boldsymbol{\phi}_{ijkl}\mathbf{q}_j\mathbf{q}_k\mathbf{q}_l + \ldots \tag{7.7}$$

This equation deserves some comments. The computed coefficients a_j are contained in the parameters $\boldsymbol{\phi}$, which can be considered as higher-order tensors. The first-order tensor $\boldsymbol{\phi}_i$ contains the elements of the gradient vector $\mathbf{g}$ from before: specifically, it contains the coefficients from the linear part of equation (7.5); the second-order tensor $\boldsymbol{\phi}_{ij}$ contains the elements of the Hessian, $\mathbf{H}$ from before: specifically, it contains the coefficients from the quadratic part of equation (7.5). The higher-order $\boldsymbol{\phi}$'s contain the appropriate remaining a_j coefficients. Furthermore, $\mathbf{q}_j$ means the jth element of $\mathbf{q}$, which is constructed as before, $\mathbf{q} = [x - x_*, \;\; y - y_*]^T$. Finally, if the original polynomial representation given in equation (7.5) is order n, then this gradient equation has polynomial order equal to $n - 1$.

The position update for the ith agent is constructed from the elements of $\mathbf{q}$ that satisfy (7.7). Eq. (7.7) is a vector equation (i.e., $\mathbf{q}$ is 2×1 column arrangement) and there are no closed form expressions for the general polynomial form. Consequently, to determine the elements of $\mathbf{q}$, a Lagrangian expansion technique is used [1].

To begin, a ℓth-order expansion of $\mathbf{q}_j$ is defined in terms of an artificial scalar parameter λ.

$$\mathbf{q}_j = \boldsymbol{\varepsilon}_{0j} + \lambda\boldsymbol{\varepsilon}_{1j} + \lambda^2\boldsymbol{\varepsilon}_{2j} + \ldots + \lambda^\ell\boldsymbol{\varepsilon}_{\ell j} \tag{7.8}$$

Here, like $\mathbf{q}$, each $\boldsymbol{\varepsilon}$ term is a 2×1 vector.

Moreover, the system is perturbed by introducing powers of the same artificial parameter λ into equation (7.7) as scalar multiples of the $\boldsymbol{\phi}$'s.

$$0 = \boldsymbol{\phi}_i + \boldsymbol{\phi}_{ij}\mathbf{q}_j + \lambda\boldsymbol{\phi}_{ijk}\mathbf{q}_j\mathbf{q}_k + \lambda^2\boldsymbol{\phi}_{ijkl}\mathbf{q}_j\mathbf{q}_k\mathbf{q}_l + \dots \tag{7.9}$$

Now equation (7.9) can be written using the assumed form of $\mathbf{q}$, and the subsequent equation can be arranged by gathering coefficients based on the associated order of the artificial parameter λ.

$$\begin{aligned} 0 = (\boldsymbol{\phi}_i + \boldsymbol{\phi}_{ij}\boldsymbol{\varepsilon}_{0j}) &+ \lambda(\boldsymbol{\phi}_{ij}\boldsymbol{\varepsilon}_{1j} + \boldsymbol{\phi}_{ijk}\boldsymbol{\varepsilon}_{0j}\boldsymbol{\varepsilon}_{0k}) \\ &+ \lambda^2(\boldsymbol{\phi}_{ij}\boldsymbol{\varepsilon}_{2j} + 2\boldsymbol{\phi}_{ijk}\boldsymbol{\varepsilon}_{0j}\boldsymbol{\varepsilon}_{1k} + \boldsymbol{\phi}_{ijkl}\boldsymbol{\varepsilon}_{0j}\boldsymbol{\varepsilon}_{0k}\boldsymbol{\varepsilon}_{0l}) \\ &+ \lambda^3(\boldsymbol{\phi}_{ij}\boldsymbol{\varepsilon}_{3j} + \dots \end{aligned} \tag{7.10}$$

The unknown $\boldsymbol{\varepsilon}$ vectors can be determined in a recursive manner by requiring that each parenthetical term of (7.10) independently vanish.

$$\boldsymbol{\varepsilon}_{0m} = -\boldsymbol{\phi}_{im}^{-1}\boldsymbol{\phi}_i \tag{7.11}$$

$$\boldsymbol{\varepsilon}_{1m} = -\boldsymbol{\phi}_{im}^{-1}\boldsymbol{\phi}_{ijk}\boldsymbol{\varepsilon}_{0j}\boldsymbol{\varepsilon}_{0k} \tag{7.12}$$

$$\boldsymbol{\varepsilon}_{2m} = -\boldsymbol{\phi}_{im}^{-1}(2\boldsymbol{\phi}_{ijk}\boldsymbol{\varepsilon}_{0j}\boldsymbol{\varepsilon}_{1k} + \boldsymbol{\phi}_{ijkl}\boldsymbol{\varepsilon}_{0j}\boldsymbol{\varepsilon}_{0k}\boldsymbol{\varepsilon}_{0l}) \tag{7.13}$$

$$\vdots$$

Because the interest lies in solving for $\mathbf{q}$, which is assumed to be of order ℓ, only the coefficients associated with λ's of order ℓ or less are considered. This results in an over-determined system and only the first $\ell + 1$ parentheticals must be solved; the other higher-order terms will go unused. This truncation can lead to some approximation error in the update; however the added information provided by this approach has been found beneficial over the quadratic-based approach; this will be demonstrated in the following section.

Once the $\boldsymbol{\varepsilon}$ vectors are determined, the vector $\mathbf{q}$ can be formed by setting the artificial parameter λ equal to one, which brings equations (7.7) and (7.9) into agreement, as required. This gives the position update for the ith agent.

$$\mathbf{q} = \boldsymbol{\varepsilon}_0 + \boldsymbol{\varepsilon}_1 + \boldsymbol{\varepsilon}_2 + \ldots + \boldsymbol{\varepsilon}_\ell \tag{7.14}$$

or

$$\begin{aligned}[x_{*,k+1}, \;\; y_{*,k+1}]^T = {}& [x_{*,k}, \;\; y_{*,k}]^T \\ & + \gamma(\boldsymbol{\varepsilon}_0 + \boldsymbol{\varepsilon}_1 + \boldsymbol{\varepsilon}_2 + \ldots + \boldsymbol{\varepsilon}_\ell)\end{aligned} \tag{7.15}$$

Where γ is a step size parameter, $0 < \gamma \leq 1$. For $\gamma = 1$ the position update marks the presumed source location, which is the minimum as approximated by the Lagrangian expansion of the polynomial model that best fits the data supplied by the neighboring agents. The index k denotes the current position and information, and $k+1$ denotes the new position.

The quadratic update term, $\boldsymbol{\varepsilon}_0$, of (7.14) is solved in (7.11). Because $\boldsymbol{\varepsilon}_i$, for $i \neq 0$, is dependent on the higher-order tensors, shown in (7.12) and on, they are referred to as higher-order update terms. This implies the solution update is not as simple as a steepest decent method (of a truncated higher-order model), and does in fact take additional information from the higher-order terms in the model.

1. Iso-contour Localization

If instead of identifying minima of a field source the interest is finding iso-contours, the same approximation and update procedure can be used. The agents still generate the nth order environment model $F(x, y)$, as seen in Section C. To locate the contour $F(x, y) = c$, for some constant c, sensor positions with updated with respect to the associated $(2n)$th order function $G(x, y)$ where

$$G(x, y) = (F(x, y) - c)^2 \tag{7.16}$$

In order to update with respect to $G(x, y)$, the same update laws as before are used with no need to alter the update parameter ℓ of equation (7.14) unless desired. All that is required is to relate the coefficients of $G(x, y)$ to the coefficients, a_j, of $F(x, y)$ and update as previously described.

2. Technical and Implementation Notes

The Hessian, $\boldsymbol{\phi}_{im}$, was assumed to be invertible. In the case that it is non-positive definite, an alternate Hessian based on the absolute values of the eigenvalues of $\boldsymbol{\phi}_{im}$ can be used instead to determine an update. Also, in practice, the updates should be constrained to create smoother paths. This can be done by adjusting the value of γ in (7.14) as needed. For mobile robotic platforms, additional sensors/methods must also be employed to avoid collision. Again, these and other implementation concerns are thoroughly addressed in [19] for the quadratic method and are applicable in the higher-order method as well.

In [11], pressure measurements are used to model a gradient map to locate an acoustic signal. While real vehicles were not used, hand placed sensors verified that their quadratic model equivalent algorithm worked and provided appropriate position updates. In [6], miniature robots were equipped with temperature gauges and used the quadratic model update law to locate a single source of dry ice. This implementation was based on that work, and returned very similar results to the numerical simulations, in [19]. It is believed the straight forward implementation of the higher-order update laws onto any autonomous platform with appropriate sensors should also achieve results similar to the numerical simulations in the following section.

D. Numerical Simulation Results

In this section some numerical simulation results are presented to gauge the efficacy of the higher-order method. Specifically, all numerical results were achieved with a fourth-order method, and the update vector $\mathbf{q} = \boldsymbol{\varepsilon}_0 + \boldsymbol{\varepsilon}_1 + \boldsymbol{\varepsilon}_2$. The results are compared and contrasted against the quadratic method results discussed in [19].

1. A Cubic Environment

First a cubic environment is considered.

$$F(x, y) = x^2 - 0.2xy + y^2 - 0.3x^3 - 0.1y^3 \tag{7.17}$$

This function has a minimum at the origin. Results from the quadratic and fourth-order methods are shown in Fig. 52, upper left and upper right. A total of 25 agents are used and agent movement is indicated from 'x' to 'o'. In these plots, the quadratic method used 20 position updates with a maximum update length of 0.1 units, whereas the fourth-order method used 25 position updates with the same maximum update length. Both methods are able to sufficiently locate the minimum, and in each case the minimum could be located with fewer steps but the trajectories would contain sharp corners. The purpose of this example is to show that the higher-order method works as well as the quadratic method for relatively simple environment models.

In practice, however, it is may not desirable nor practical to have each of the robotic agents approach a minima. The use of supporting robotic agents, ones which help provide a rich set of data points, may be desired. Many more complicated control schemes may be used to control such support agents but for simplicity support agents are stationary in this note.

In Fig. 52, bottom left, the quadratic method search agents are greatly influenced by the support agents. In this case it biases the model to the right of the true minima as agents stationed to the left of zero on the x-axis measure a larger scalar value than those to the right. This causes the mobile search agents to falsely believe the minima is located away from the origin. Certain configurations of search versus support agents as well as total number of agents may yield better results, but a resulting bias will remain unless communication is severely limited (a technique referred to as "local communication," where the agents only use information provided by other agents within a specified distance) or all agents are allowed to converge together. More advanced mobile support agent models were also used and displayed similar results.

Note in Fig. 52, bottom right, the fourth-order method is not negatively influenced by the supporting agents and locates the minima. Also, the support agents, while acting in a more realistic manner, help maintain a more numerically stable approximation model with respect to the least squares procedure.

2. A Cubic Environment Iso-contour Localization

If the interest is locating the iso-contour line, $c = 20$, the update law is altered as seen in the previous section. Fig. 53, left, shows the results when $F(x, y)$ is a quadratic approximation and the associated $G(x, y)$ is a fourth-order polynomial. Because the environment model is quadratic, it does not adequately model the test environment. The result is agents are near, but off the true contour. Fig. 53, right, shows the results when $F(x, y)$ is a fourth-order approximation (and therefore very accurate) and the associated $G(x, y)$ is a eighth order polynomial. It is clear that the higher-order approach accurately locates the desired iso-contour.

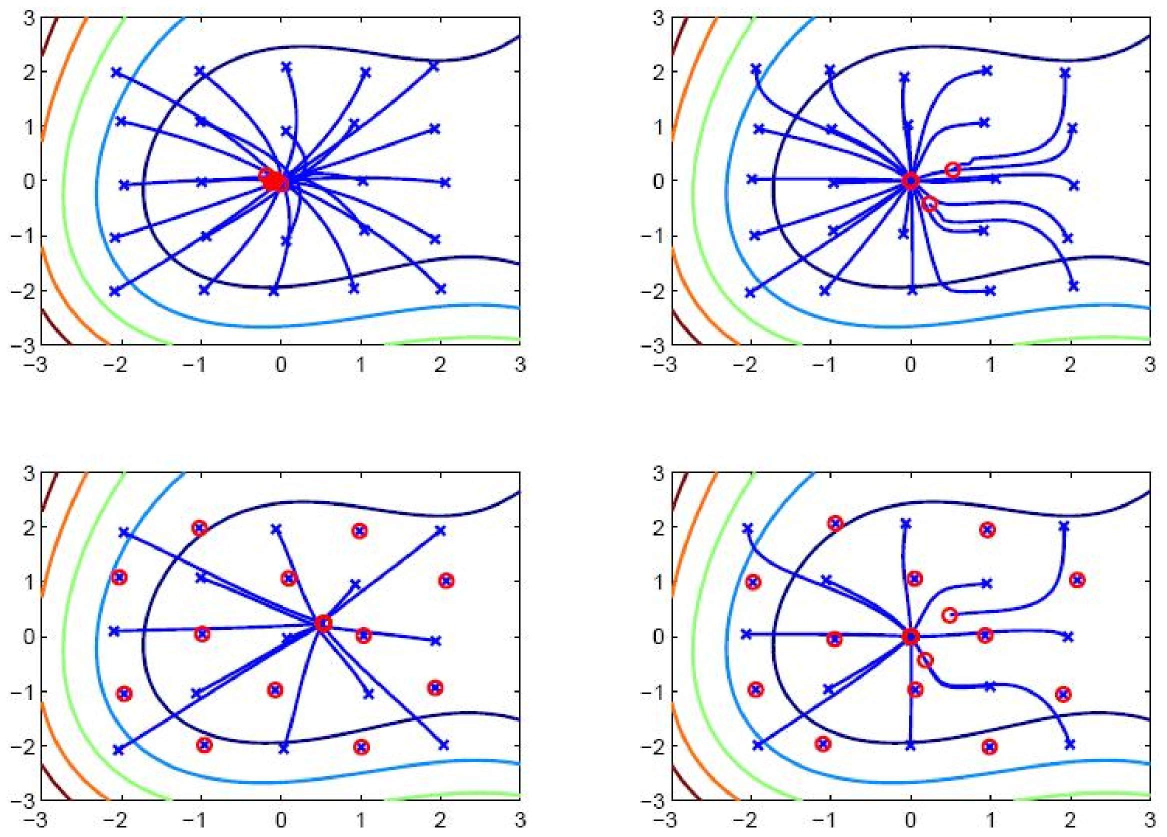


Fig. 52. Cubic test function: Upper left: Quadratic method. Upper right: Fourth-order method. Bottom left: Quadratic method with stationary support agents. Bottom right: Fourth-order method with stationary support agents.

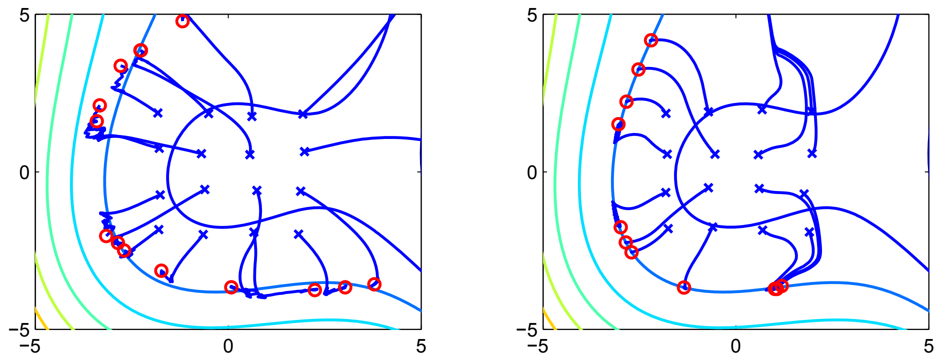


Fig. 53. Iso-contour localization: Left: Quadratic method. Right: Fourth-order method.

3. Rosenbrock Function

The Rosenbrock function presents a challenge. This popular non-convex function is often used to test the capability of optimization methods and has a global minimum located at $(1, 1)$.

$$F(x, y) = 100(y - x^2)^2 + (1 - x)^2 \tag{7.18}$$

To begin, twenty-five agents are distributed around the origin. Results from a quadratic method and a fourth-order are shown in Fig. 54. The upper left plot shows the quadratic model results where the quadratic model leads the agents to the origin, after which, the agents cease to update. Due to the restrictive nature of the quadratic model, the steep gradient of the valley sides dominate the approximation and the more subtle gradient leading to the minima is missed. Once the robots have converged they no longer have a rich set of data points and become more or less stationary. Results for the fourth-order model are shown in the upper right plot. Here, the agents move along the steep gradient to the valley, and then move along the valley to the global minimum at $(1, 1)$. At the final time shown, sixteen of the twenty-five agents have located the minimum. Three of the agents are actually led away from the plotting area because their constructed fourth-order models indicated other possible minima. (Recall that this approach is decentralized, which means that each robotic agent decides its own update based on the environment model and location associated with that individual.)

The lower plots of Fig. 54 show results for the quadratic method and a fourth-order method with stationary support agents similar to those seen in the cubic example. Unfortunately, the quadratic method still leads the agents more or less to the origin, and even with a more diversified data set, once they mobile agents approach a position near $(0.5, 0.5)$ they too cease to update. The stationary support agents were used with the fourth-order method, the results are shown in the lower right plot and are a slight improvement over the no support agent scenarios.

Plots pertaining to the quadratic method use twenty position updates (more updates did not improve performance) at a maximum update length of 0.1 units. Plots pertaning to the fourth-order method also use a maximum update length of 0.1 units. For this maximum update length, four of the twenty-five agents converge on the minimum in twenty position updates with no support agents, and an additional twelve converge by fifty position updates. For the fourth-order example with support agents, three of the thirteen mobile agents converge on the minimum in twenty position updates, an additional four converge by fifty position updates, and an additional two converge by seventy updates.

Although this example considers a fourth-order method on a fourth-order function, there is numerical error introduced when the agents create their environment models, thus there is not a perfect match. More significantly, as noted earlier, the Lagrangian expansion technique will provide some truncation error, it is encouraging to see the fourth-order method work so well for the Rosenbrock function. Moreover, this example suggests that the higher-order method may be useful in other numerical optimization tasks in that the method applies to higher-dimensional problems.

4. Peaks Function

Next the MATLAB "peaks" function is considered. This multi-modal function was used in evaluating the performance of the quadratic method in [19] and the key figures are reproduced herein. Fig. 55 shows a collection of thirty-six agents cooperatively localizing the global minimum of the function. The agents successfully collect around the minimum after nearly fifty position update steps. Fig. 56 shows the same collection of thirty-six agents cooperatively localizing the minimums of the function using

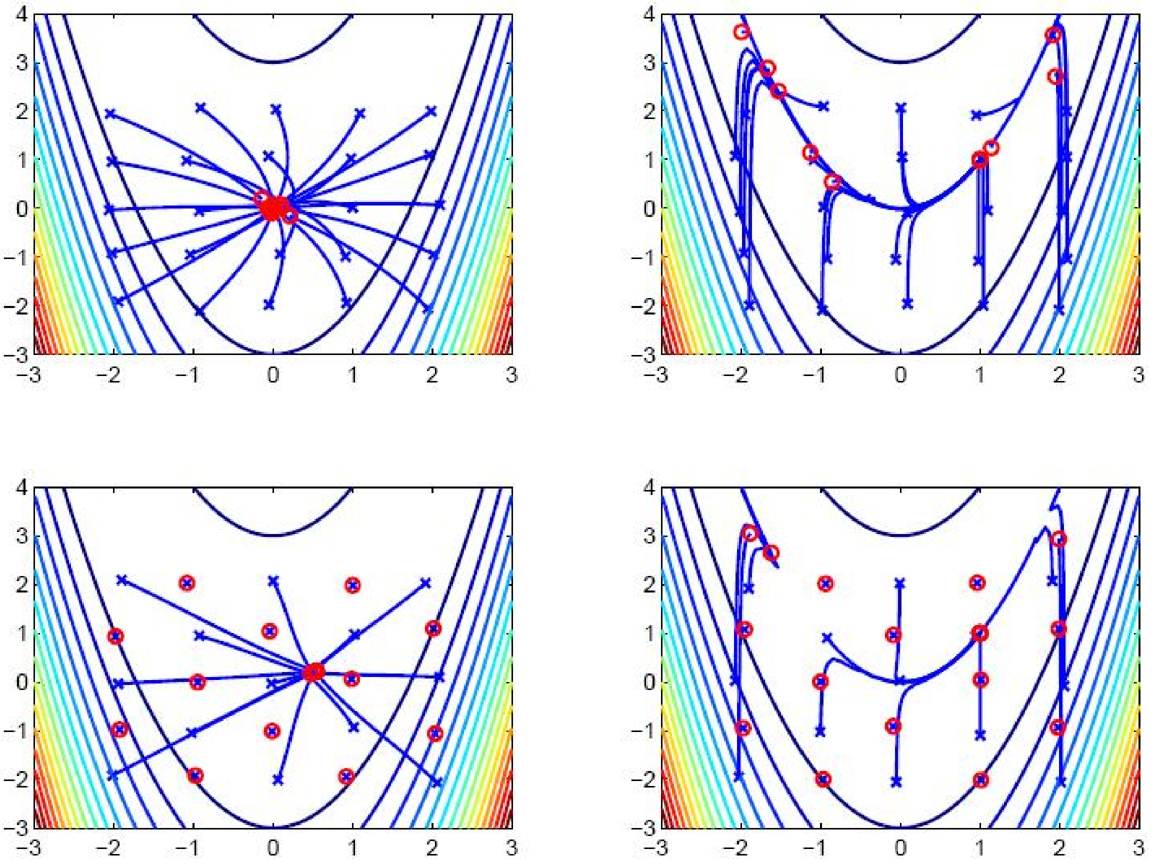


Fig. 54. Rosenbrock test function: Upper left: Quadratic method. Upper right: Fourth-order method. Lower left: Quadratic method with stationary support agents. Lower right: Fourth-order method with stationary support agents.

local communication, where communication is restricted to agents within a two unit radius. These simulations demonstrate that the quadratic method is capable of locating global and local minimums by tuning the communication radius of the agents. However, this can be problematic since there is no reason to believe a user will have advanced knowledge of the spatial resolution required for accurate environment mappings.

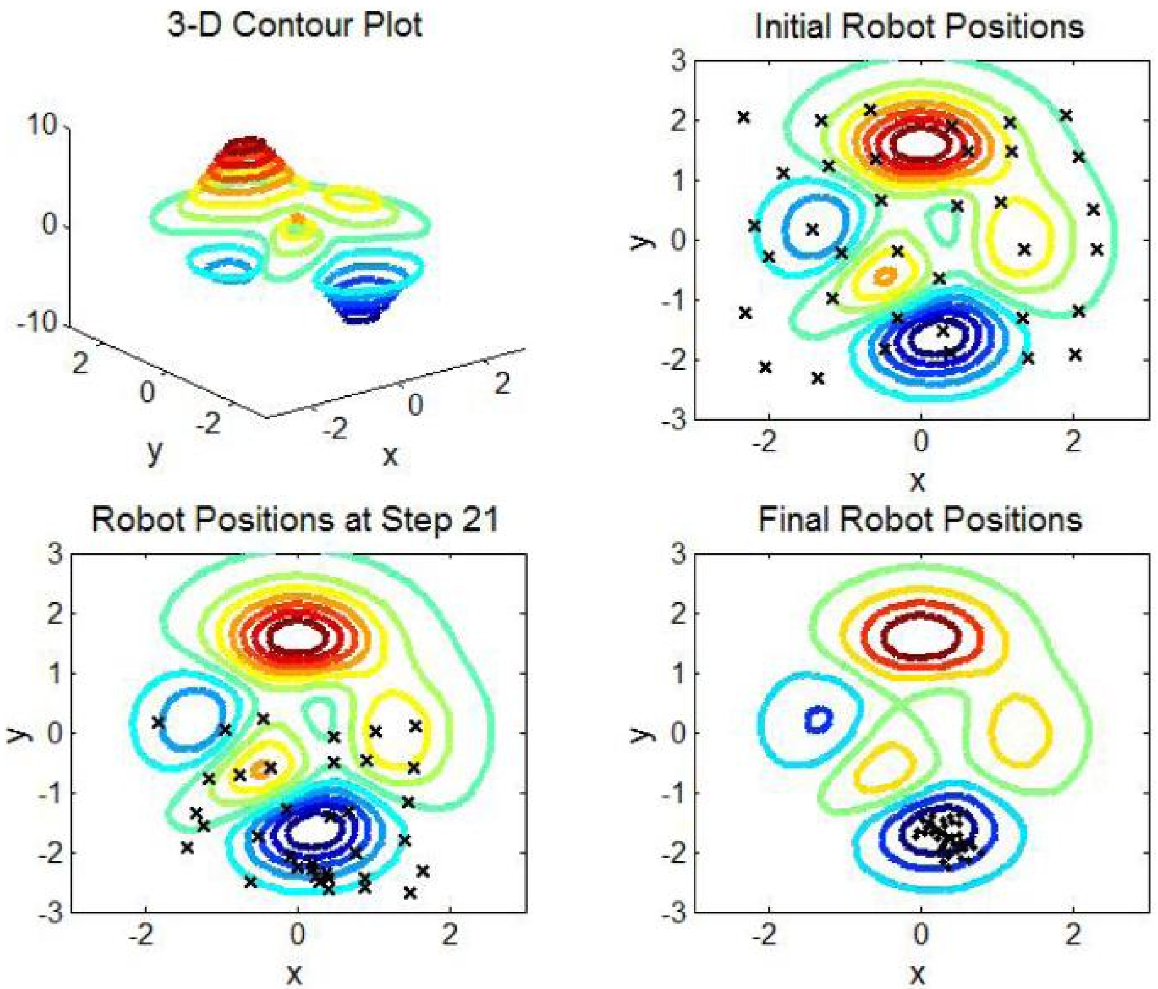


Fig. 55. Peaks test function: Quadratic method with global communication (Hurtado et al., 2004).

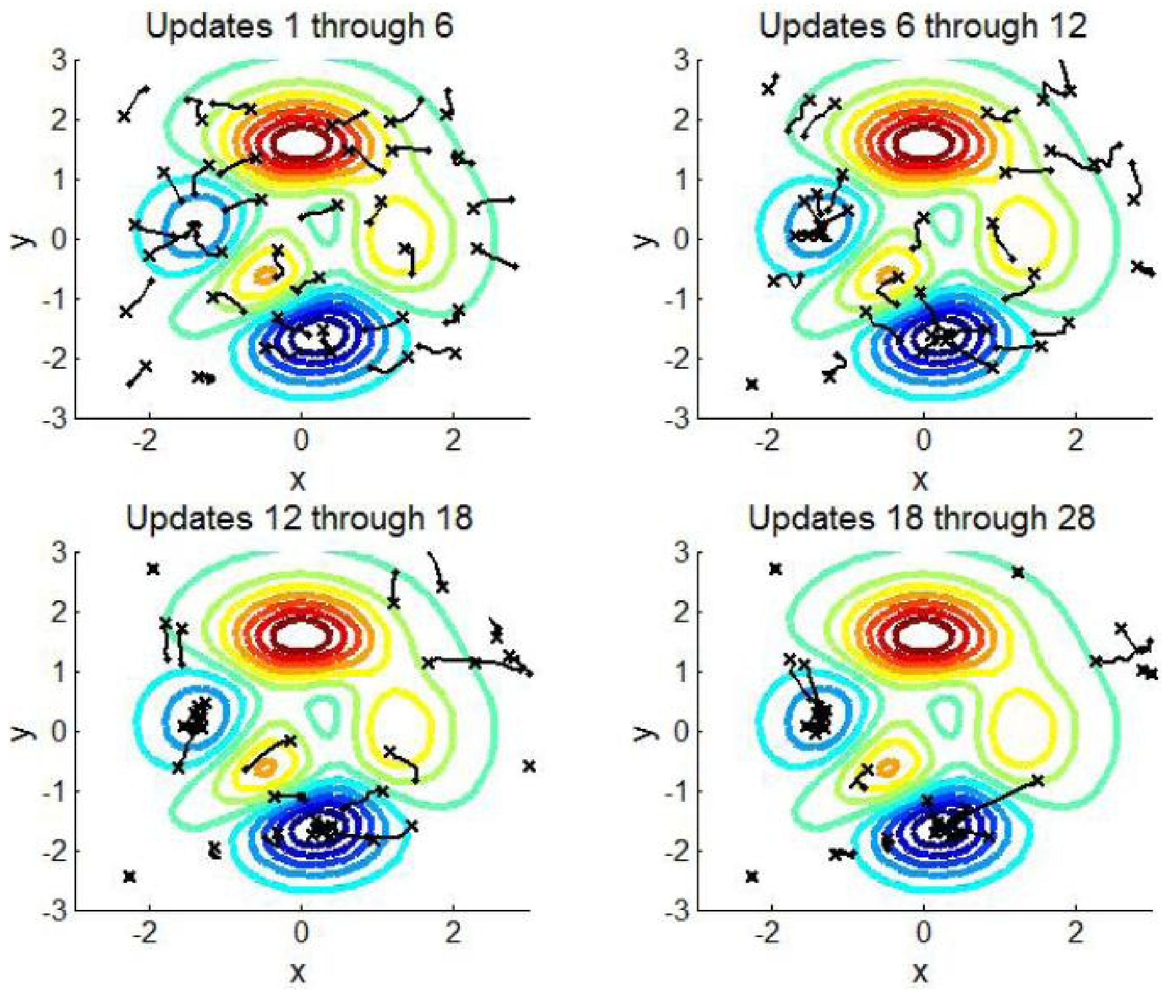


Fig. 56. Peaks test function: Quadratic method with local communication (Hurtado et al., 2004).

The performance of the fourth-order method with twenty-five agents is shown in Fig. 57. The plots show that the fourth-order method is able to find and gather around the global and local minimums, without any special communication features. The results are comparable to the quadratic method using the special local communication feature. Note that altering the communication radius for the quadratic method will affect the outcome and can easily cause the agents to miss the local minimum (larger communication radius) or strand an agent (smaller communication radius) which is too far removed from other agents; There is no such concern using the higher-order method and we can simply use global communication.

5. Robustness to Measurement and Position Error

The two predominate errors associated with implementation of these search algorithms are measurement errors due to the scaler sensors and position estimate errors. The robustness of the higher-order method is demonstrated by comparing the fourth-order model performance using the peaks test function with no error, with 5% error in position readings, with 3% error in measurement of the scalar function, and with errors in both position and measurement. A 5% error refers to error with normal distribution and a mean of 5% of the value range. For position, the range is between -3 and 3, so the error value is normal with mean of 0.3 (5% error is added in both x and y directions). For sensor measurements, the peaks function ranges between -7 and 8, so the error value is normal with mean of 0.5 units.

The higher-order method preforms admirably in each situation, see Fig 58. The plots show only the initial position, ‘x’, and the final agent positions, ‘o’. Note that several agents appear in the same final position in the zero error case, upper left. Clearly the performance is reduced when one or both errors are introduced, but the results are still rather accurate. The introduced errors of each robotic agent change

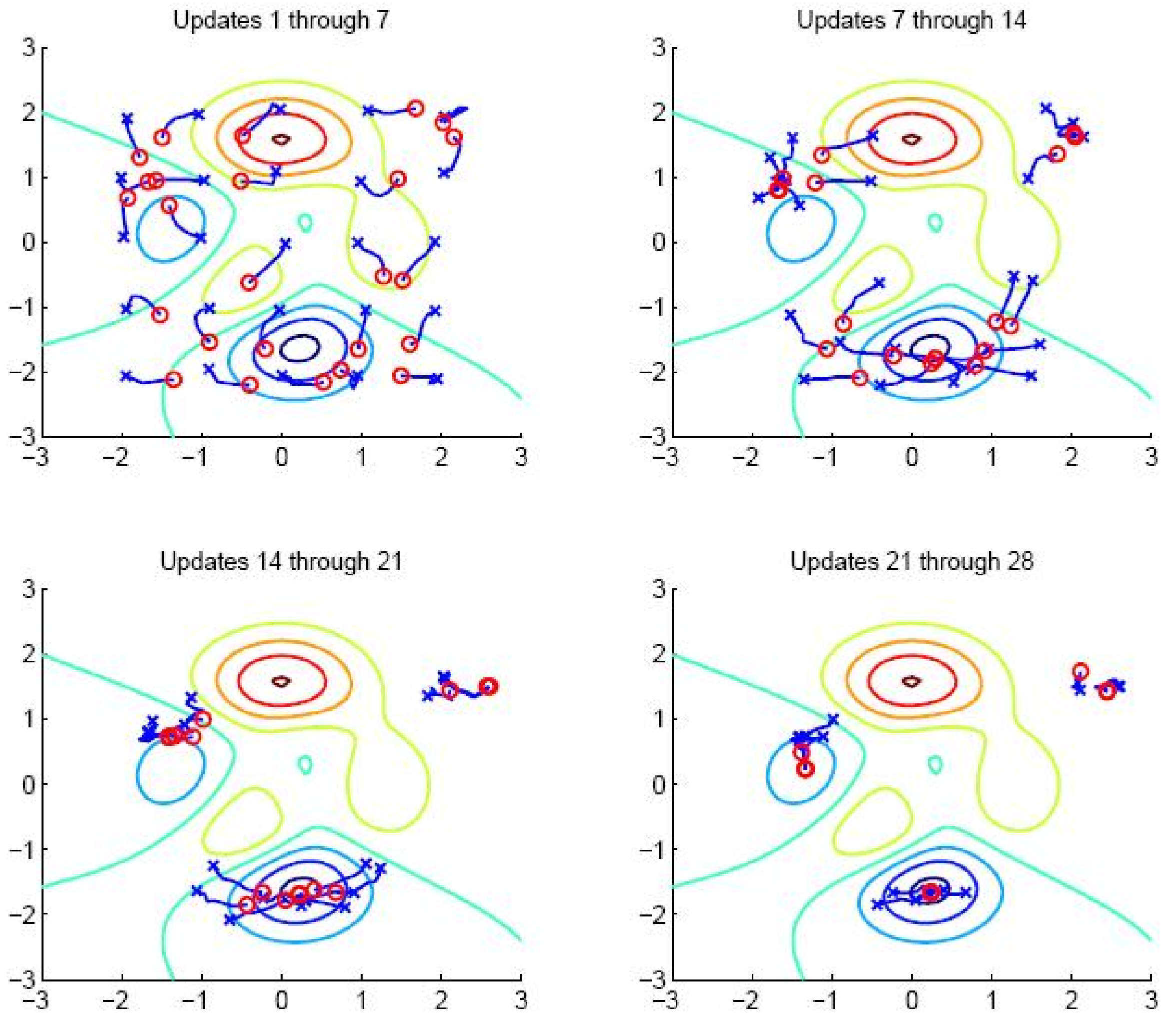


Fig. 57. Peaks test function: Fourth-order method.

randomly from one sensor reading to the next, same for the position error, creating the choppy update paths seen. If, instead, agents were to take longer sensor readings and average over time, they can reduce noise, and presumably improve performance even in a turbulent setting [49]. When each agent was assigned unique, constant biases for measurements and position estimates throughout the simulation very similar results were achieved.

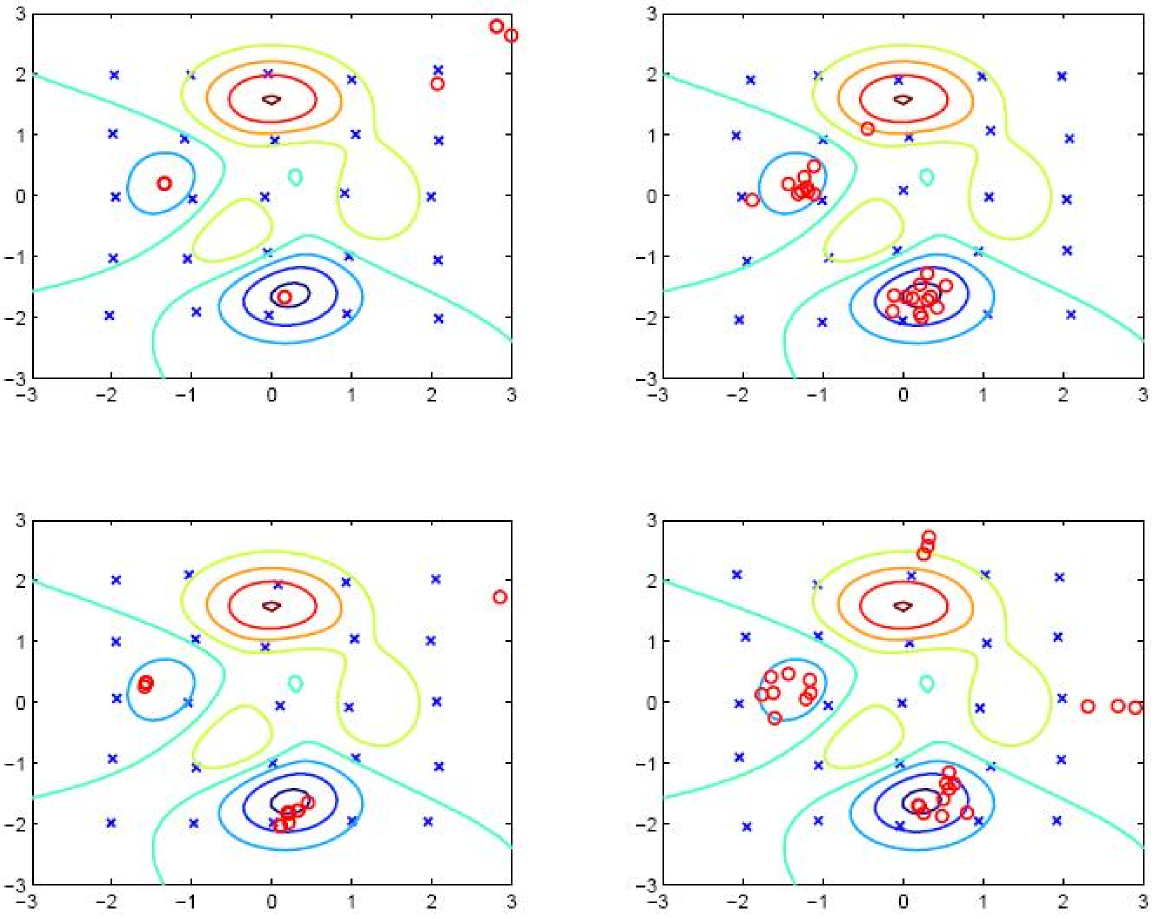


Fig. 58. Peaks test function: Fourth-order method, starting agent positions marked with 'o', final agent positions marked with 'x'. Upper left: No errors. Upper right: 5% position error. Lower left: 3% measurement errors. Lower right: 5% measurement and 3% position error.

6. Effects of Higher-order Update Terms

Generally the performance increase between the quadratic and a higher-order method is attributed to the increase in model order. Given a more accurate model, a linear update can be solved by truncating the model and following the quadratic method often delivering better results than simply starting with the quadratic model. However, it should be noted that the fourth-order method examples shown herein used the first three update terms from (7.14), the latter two update terms based on the higher-order tensors in the approximation.

A portion of the Rosenbrock test function is shown in Fig. 59 with update laws using one, two, and three update terms. While all three simulations result in agents converging at the minimum, both the middle and lower plot provide a smoother path and do so with fewer updates than the upper plot which used only the quadratic update term. The second and third update terms run along iso-contours of the function, combining these update terms with the quadratic term generate more efficient trajectories and demonstrates the value in solving for the higher-order update terms.

E. Summary

The main strength of the higher-order method is that it generates a more accurate environment model. This more accurate model provides the possibility to locate multiple minima. In [19], it was shown that lower order models can identify multiple minima by restricting communication distances between neighboring agents, but this is likely not a desired practice and is not needed in the the higher-order approach. Moreover, the higher-order approach is able to act on more subtle features of the environment, as demonstrated with the Rosenbrock function. The higher-order method was shown to provided accurate results in scenarios the quadratic method would not

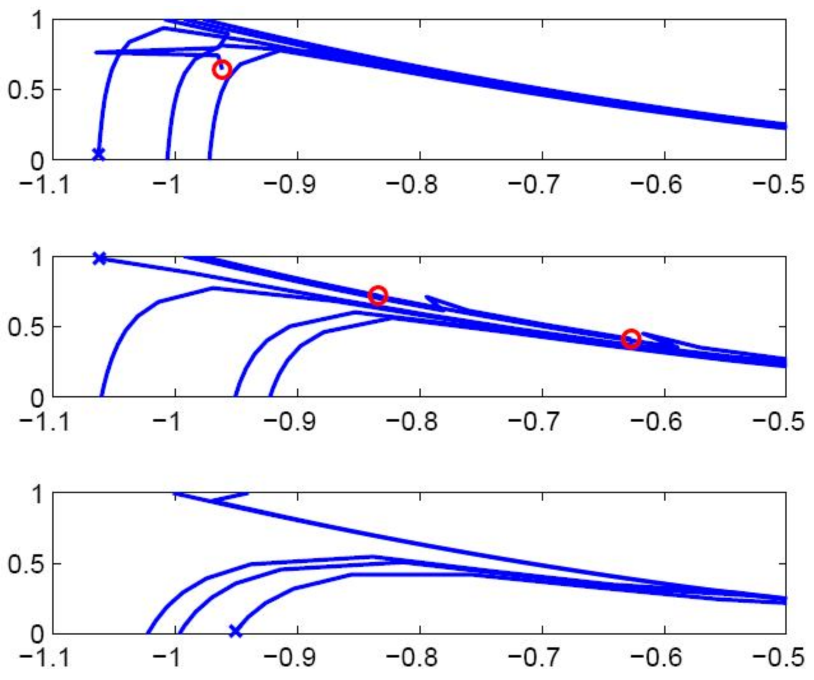


Fig. 59. Rosenbrock test function: Fourth-order method. Upper: First update term. Middle: First two updates terms. Bottom: All three update terms.

because of the quadratic model's tendency to mask subtle features or its propensity to be biased by agents located away from the minima. Because of the additional freedom in the environment model the higher-order method does not encounter these problems.

One perceived disadvantage of the higher-order approach is the requirement of more data points. A quadratic method for agents acting on a two-dimensional surface requires six independent measurements, whereas the fourth-order method requires fifteen. To sidestep this issue, a system that has a limited number of agents could use old data points in relatively static environments; [21], [49] and others use a single robotic agent and rely heavily on old data. The higher order method also has the advantage that it can utilize lower-order models when old data is perceived to be unreliable. And, unlike single agent systems this method may prove more valuable with time-variant sources, as models can be created using only current or very recent if multiple agents are being employed.

Overall the higher-order method is more robust and has a wider range of possible use. In cases where a quadratic model works well the fourth-order model also preformed well, while the quadratic model failed in several examples where the higher-order method succeeded. The higher-order method is a strong improvement over previous quadratic-based localization methods.

CHAPTER VIII

CONCLUSIONS AND FUTURE WORK

A. Simple Extrinsic Self-calibration of Multi-camera Systems

1. Conclusions

The problem of multi-camera system calibration was addressed. The focus for this research was to develop a simple method for calibration of such systems that only requires a basic understanding of cameras, image processing, and optimization. The ability to use video to obtain image correspondences of a single point target moved throughout the shared fields of view of the cameras has made 0D methods highly desirable. The corresponding pixel values are easily attained through video; however, there is no information regarding the true, global position of the feature points. Previously, in [17], a novel approach was used that reprojected the estimated 3D location of the feature point as a function of camera parameters and pixel values for each camera pairing. The evaluation function being minimized was the variance of 3D reprojections for each of the target points.

Instead of minimizing the variance, the RF cost function, developed in Chapter II, minimizes the numerical fitting error when solving for the 3D reprojections. This approach was shown to be beneficial in several ways. First, the process is less computationally complex, requiring less calculations in the cost evaluation. Second, the approach is more general; it only requires two cameras (not three). Not only does the RF evaluation only require two cameras, but when more cameras are used only two cameras need to see a feature before it is included in the optimization. This can drastically increase the amount of physical space target features can be collected in and results in a more uniform solution. Third, for the same feature points, the RF cost function is sharper than its predecessor with a more clearly defined minimum making it better suited for simple optimization methods.

Overall, the new RF cost function was shown to be a more efficient evaluation index, applicable to more applications and provide better results for comparable applications.

2. Future Work

Data has been collected and the RF cost function has been used to calibrate camera systems in a lab setting. Collaborations are anticipated with groups who use 3D computer vision and have three dimensional video motion detection (3DVMD) software available. The goal is to compare results in the field to more standard calibration results, however the true test will be the 3D renderings provided by the 3DVMD software using RF evaluation based calibration results.

B. Vision-aided Inertial Navigation

1. Conclusions

A vision-aided inertial navigation system was introduced using a complementary form EKF and the unit sphere observation model. Primarily results were presented using a vSLAM system where a feature map was generated and the vehicle recognized previously seen features. The visual odometry approach was also demonstrated. For a UGV in a hallway environment it was shown that basic paths, sinusoids and sawtooths, provided the IMU and forward facing camera sufficient information to maintain accurate egostate estimates for a large array of path amplitudes and spatial periods. It was also shown that position estimates were accurate and consistent enough for many of those paths to be used in the guidance law.

Examples in Chapter IV demonstrated the need to consider estimation accuracy in the control cost when path planning, integrated estimation and control. It was shown that a poor choice in nominal path can lead to poor egostate estimates, which in turn cause inefficient control outputs from the guidance law. Methods to recognize which paths are accurate and efficient were investigated, and it was clear that the covariance of the current estimator is not a reliable index and that overconfidence is prevalent in the filter for less active paths.

A 6DoF observability analysis was performed and showed that the instantaneous observability of the system is very weak, as expected, but that over multiple measurements the system gains full local observability. Further investigation shows that the increased observability is in some part due to vehicle translation relative to the features. Because the motion between measurements is small, it takes several measurements (and therefore more time) to develop practical observability, particularly in the direction the camera is facing.

It was demonstrated in Chapter V that monitoring the changes in total acceleration as the desired path is adjusted can indicate when more or less nominal control should be used. This is promising, although it is a reactive method.

Finally, the Schmidt-Kalman filter was applied to the vSLAM system to achieve nearly a 2/3rds reduction in filter size. Estimation results were degraded in the open-loop case, where position estimate errors basically doubled. For closed-loop control, the estimation results were equivalent for the full and reduced EKF at the expense of doubling the control effort in the Schmidt-Kalman filter case. Depending on the system in use, especially one with generally erratic motion (flapping vehicles, small UAVs), it may be beneficial to require less computational power at the expense of control effort.

2. Future Work

The hardware implementation of such a vSLAM system is the main focus of future work. Once a hardware testbed is in place, a higher fidelity simulation environment will be developed that incorporates more realistic hallway settings, realistic path generation using a vehicle model for accurate control cost, and camera resolution and IMU capabilities that match the vehicle/hardware in use.

Continued investigations into integrated estimation and control are needed. Specifically, path selection methods are being investigated that rely on observability analysis and not covariance. This could potentially allow the identification of recently unobservable states and providing updated controls to counteract that. Camera orientation is being more thoroughly investigated, as well as the use of multiple cameras to generate improved observability in the direction of travel. This work will rely heavily on the higher fidelity hallway environment.

C. A Higher-order Method for Cooperative, Decentralized Source Localization

1. Conclusions

Based on a standard quadratic model approach to source localization, a higher-order polynomial approach was introduced. Previously, agents generated a quadratic model estimate of their environments, then used a gradient method (linear updates) to update their position towards the assumed minimum. The higher-order approach allows agents to generate high(er) fidelity environment models to be used when solving for their desired position update. The higher-order terms generate nonlinear equations; however, a Lagrangian expansion technique [1] is used which avoids the use of a nonlinear solver.

Without significantly increasing the computational complexity of solving for the position updates, the higher-order method was shown to outperform the quadratic approach in several examples. The higher-order terms were shown to contribute to smoother paths, specifically for the Rosenbrock function. Overall, the higher-order method was shown to be a significant improvement over its predecessor.

2. Future Work

Hardware testing which compares the quadratic and higher-order methods would be ideal, but requires access to several vehicles. Additional study of which environment types benefit the greatest due to the improved model fidelity should be conducted. And optimization applications of nonphysical systems should be considered due to the generic architecture of the higher-order method.

The topics in this dissertation are methods that enable autonomous navigation and cooperative search in indoor environments. One application that ties the methods together is one that allows the implementation of the higher-order source localization approach on agents that use vision-aided inertial navigation. The scenario could be chemical weapons stored inside a large warehouse or a container ship where GPS is often blocked or suffering from severe multipath issues. In this hypothetical scenario, a swarm of micro-sized UAVs would be sent to cooperatively locate and confirm the source location, and then report the accurate weapons location without being noticed. Similar scenarios could include chemical leaks inside industrial buildings, heat sources in disaster areas, or locating gunfire (sound) sources in GPS jammed/spoofed settings.

REFERENCES

[1] R. Bellman, *Perturbation Techniques in Mathematics, Physics, and Engineering.* New York: Holt, Rinehart and Winston, Inc., 1964.

[2] J. Bouguet, "Camera Calibration Toolbox for Matlab," Accessed Nov. 2007, http://vision.caltech.edu/bouguetj/calib_doc.

[3] K. Brink, J. Hurtado, A. Soloviev, A. Rutkowski, and T. Klausutis, "Integrated Estimation and Control Approach for Vision Aided Inertial Navigation," *2010 AIAA Infotech@Aerospace Conference*, Atlanta, GA, Apr. 2010.

[4] R. Brown and P. Hwang, *Introduction to Random Signals and Applied Kalman Filtering,* 3rd ed. New York: John Wiley and Sons, Inc., 1997.

[5] M. Bryson and S. Sukkarieh, "Observability Analysis and Active Control for Airborne SLAM," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 44, no. 1, pp. 261-280, Jan. 2008.

[6] R. H. Byrne, D. R. Adkins, S. E Eskridge, J. J. Harrington, E. J. Heller, and J. E. Hurtado, "Miniature Mobile Robots for Plume Tracking and Source Localization," *Journal of Micromechatronics*, vol. 1, no. 3, pp. 253-261, 2002.

[7] R. H. Byrne, S. E. Eskridge, J. E. Hurtado, and E. L. Savage, "Algorithms and Analysis for Underwater Vehicle Plume Tracing," *Sandia Report: SAND2003-2643*, 2003.

[8] Y. Cui and S. S. Ge, "Autonomous Vehicle Positioning With GPS in Urban Canyon Environments," *IEEE Transactions on Robotics and Automation,* vol. 19, no. 1, pp. 15-25, Feb. 2003.

[9] G. Chen and M. Harigae, “IMM Based Interference/Jamming and Spoofing Detection in a DGPS-aided Inertial System,” in *Proceedings of the AIAA 10th International Space Planes and Hypersonic Systems and Technologies Conference*, Kyoto, Japan, April 24-27, 2001.

[10] I. Chen and S. Wang “An Efficient Approach for Dynamic Calibration of Multiple Cameras,” *IEEE Transactions on Automation Science and Engineering*, vol. 6, pp. 187-194, Jan. 2009.

[11] J. L. Dohner, G. R. Eisler, B. J Driessen, and J. E. Hurtado, “Cooperative Control of Vehicle Swarms for Acoustic Target Localization by Energy Flows,” *Journal of Dynamic Systems, Measurement, and Control*, vol. 126, no. 4, pp. 891-895, 2004.

[12] M. G. Farley, “An Alternate Approach to GPS Denied Navigation Based on Monocular SLAM Techniques,” in *Proceedings of the 2008 National Technical Meeting of the Institute of Navigation*, San Diego, California, January 25-28, 2008.

[13] O. Faugeras, *Three-dimensional Computer Vision: A Geometric Viewpoint.* Cambridge, MA: MIT Press, 1993.

[14] S. Haihang, G. Muhe, H. Kezhong, “An Integrated GPS/CEPS Position Estimation System for Outdoor Mobile Robots,” *IEEE International Conference on Intelligent Processing Systems*, Beijing, China, vol. 2, pp. 1282-1286, Oct. 1997.

[15] R. Heartley and A. Zisserman, *Multiple View Geometry.* Cambridge, MA: Cambridge University Press, 2000.

[16] A. T. Hayes, A. Martinoli, and R. M. Goodman, “Distributed Odor Source

Localization," *IEEE Sensors, Special Issue on Artificial Olfaction*, vol. 2, no.3, pp. 260-271, 2002.

[17] B. G. Henderson, "A Novel Method for Extrinsic Self Calibration of Wide-baseline Three-dimentional Computer Vision Systems." M.S. thesis, Electrical Engineering, University of New Mexico, Albuquerque, NM, 2006.

[18] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "Analysis and Improvement of the Consistency of Extended Kalman Filter-based SLAM," *Proceedings of the IEEE International Conference on Robotics and Automation*, Pasadena, CA, pp. 473-479, May 2008.

[19] J. E. Hurtado, R. D. Robinett III, C. R. Dohrmann, and S. Y. Goldsmith, "Decentralized Control for a Swarm of Vehicles Performing Source Localization," *Journal of Intelligent and Robotic Systems*, vol. 41, pp. 1-18, 2004.

[20] D. R. Hush and E. E. Schnetzer, "2-D Video Motion Detection (VMD): Theory and Operations Manual, *Internal Sandia Report: SAND88-0888*, 1988.

[21] H. Ishida, K. Suetsugu, T. Nakamoto, and T. Moriizumi, "Study of Autonomous Mobile Sensing System for Localization of Odor Source Using Gas Sensors and Anemometric Sensors," *Journal of Sensors and Actuators*, vol. 45, pp. 153-157, 1994.

[22] M. Joerger, "Autonomous Ground Vehicle Navigation Using Integrated GPS and Laser-scanner Measurements," *IEEE/ION Position, Location and Navigation Symposium*, vol. 1-3, pp. 988-997, 2008.

[23] S. Kalantar and U. Zimmer, "Optima Localization by Vehicle Formations Imitating the Nelder-Mead Simplex Algorithm," *Journal of Autonomous Robots*,

vol. 27, pp. 239-260, 2009.

[24] J. Kim and S. Sukkarieh, "Improving the Real-time Efficiency of Inertial SLAM and Understanding Its Observability," in *Proceedings of 2004 International Conference on Intelligent Robots and Systems,* Sendai, Japan, Sept. 2004.

[25] J. Kim and S. Sukkarieh, "6DoF SLAM Aided GNSS/INS Navigation in GNSS Denied and Unknown Environments," *Journal of Global Positioning Systems,* vol. 4, no. 1-2, pp. 120-128, 2005.

[26] Y. Kojima, T. Fujii, and M. Tanimoto, "New Multiple-camera Calibration Method for a Large Number of Cameras," in *Proceedings of the SPIE*, vol. 5665, pp. 156-163, 2004.

[27] J. Krger, B. Nickolay, P. Heyer and G. Seliger, "Image Based 3D Surveillance for Flexible Man-robot-cooperation," *CIRP Annals: Manufacturing Technology*, vol. 54, no. 1, pp. 19-22, 2005.

[28] V. I. Kurtz, "A Visual Artificial Intelligent Surveillance System to Protect Against the Insider Threat, *INMM Conference Proceedings*, 1989.

[29] J. Langelaan and S. Rock, "Navigation of Small UAVs Operating in Forests," *AIAA Guidance, Navigation and Control Conference*, Providence, Rhode Island, Aug. 2004.

[30] Y. Ma, S. Soatto, J. Koseck, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models.* New York: Springer, 2003.

[31] M. Machacek, M. Sauter, and T. Rösgen, "Two-step Calibration of a Stereo Camera System for Measurements in Large Volumes," *Journal of Measurement Science and Technology*, vol. 14, no. 9, pp. 1631-1639, 2003.

[32] R. Mifflin, "A Superlinearly Convergent Algorithm for Minimization Without Evaluating Derivatives," *Math Programming*, vol. 9, pp. 100-117, 1975.

[33] X. Nguyen, B. You, and S. Oh, "A Simple Framework for Indoor Monocular SLAM," *International Journal of Control, Automation, and Systems*, vol. 6, no. 1, pp. 62-75, Feb. 2008.

[34] R. Y. Novoselov, S. M. Herman, S. M. Gadaleta, and A. B. Poore, "Mitigating the Effects of Residual Biases with Schmidt-Kalman Filtering," *IEEE 7th International Conference on Information Fusion*, vol. 1-2, pp. 358-365, 2005.

[35] S. Pang and J. A. Farrell, "Chemical Plume Source Localization," *IEEE Transactions: Systems, Man, and Cybernetics*, vol. 36, no. 5, pp. 1068-1080, 2006.

[36] S. Panzieri, F. Pascucci, and G. Ulivi, "An Outdoor Navigation System Using GPS and Inertial Platform," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 2, pp. 134-142, 2006.

[37] L. Perera, A. Melkumyan, and E. Nettleton, "On the Linera and Nonlinear Observability Analysis of the SLAM Problem," in *Proceedings of IEEE International Conference on Mechatronics, 2009*, pp. 2061-2068, May 2009.

[38] I. Rhee, M. F. Abdel-Hafez, and J.L. Speyer, "Observability of an Integrated GPS/INS During Maneuvers," *IEEE Transactions on Aerospace and Electronic Systems,* vol. 40, no.2, pp. 526-535, Apr. 2004.

[39] G. Sandini, G. Lucarini, and M. Varoli, "Gradient Driven Selforganizing System," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, pp. 429-432, 1993.

[40] H. Schaub and J. Junkins, *Analytical Mechanics of Space Systems.* Reston, VA: AIAA Inc., 2003.

[41] S. F. Schmidt, “Applications of State-Space Methods to Navigation Problems,” *Advances in Control Systems*, vol. 3, pp. 293-340, 1966.

[42] F. Bayoud and J. Skaloud, “Vision-aided Inertial Navigation System for Robotic Mobile Mapping,” *Journal of Applied Geodesy,* vol. 2, pp. 3952, 2008.

[43] A. Soloviev, J. Touma, T. Klausutis, M. Miller, A. Rutkowski and K. Fontaine. “Integrated Multi-aperture Sensor and Navigation Fusion,” in *Proceedings of ION GNSS-2009,* Savannah, Georgia, Sept. 2009.

[44] T. Svoboda, D. Martinec, and T. Pajdla, “A Convenient Multicamera self-Calibration for Virtual Environments” *Presence: Teleoperators and Virtual Environments archive*, vol. 14, no. 4, pp. 407-422, Aug. 2005.

[45] M. J. Veth, “Fusion of Imaging and Inertial Sensors for Navigation. Ph.D. dissertation, Electrical Engineering, Air Force Institute of Technology, Dayton, OH, 2006.

[46] T. Vidal-Calleja, M. Bryson, S. Sukkarieh, A. Sanfeliu, and J. Andrade-Cetto, “On the Observability of Bearings-only SLAM,” in *Proceedings of 2007 IEEE International Conference on Robotics and Automation*, pp. 4114-4119, Apr. 2007.

[47] F. Wang, “A Simple and Analytical Procedure for Calibrating Extrinsic Camera Parameters,” *IEEE Transactions on Robotics and Automation*, vol. 20, pp. 121-124, Feb. 2004.

[48] Y. Watanabe, “Stochastically Optimized Monocular Vision-based Navigation and Guidance.” Ph.D. dissertation, Aerospace Engineering, Georgia Institute

of Technology, Atlanta, GA, Apr. 2008.

[49] L. Wei, J. Farrell, and R. Carde, "Tracking of Fluid-advected Odor Plumes: Strategies Inspired by Insect Orientation to Pheromone," *Journal of Adaptive Behavior*, vol. 9, pp. 142-170, 2001.

[50] N. A. White, P. S. Maybeck, and S. L. DeVilbiss, "Detection of Interference/Jamming and Spoofing in DGPS-aided Inertial System, *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, no. 4, pp. 12081217, 1998.

[51] D. Woodbury and J. L. Junkins, "On the Consider Kalman Filter," 2010 AIAA Guidance, Navigation, and Control Conference, Toronto, Ontario, Canada, Aug. 9-12, 2010.

[52] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330-1334, 2000.

[53] Z. Zhang, "Camera Calibration with One-dimensional Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, pp. 892-899, 2004.

[54] Z. Zhao, and Y. Liu, "New Multi-camera Calibration Algorithm Based on 1D Objects," *Journal of Zhejiang University Science A*, vol. 9, no. 6, pp. 799-806, 2008.

VITA

Kevin Michael Brink earned a B.A. degree in mathematics at the University of San Diego in May 2005. He received an M.S. degree in mathematics and a Ph.D. in Electrical Engineering from Texas A&M University in May 2007 and December 2010. His research interests are in the areas of vision-based control, vision-based navigation, estimation, and decentralized control. He can be reached at kmbrink@gmail.com or Department of Aerospace Engineering, H.R. Bright Building, Room 706, 3141 TAMU, College Station, Texas 77843-3141.

The typist for this dissertation was Kevin Michael Brink.