

RICE UNIVERSITY

**Distributed Alternating Localization-Triangulation
of Camera Networks**

by

William E. Mantzel

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

APPROVED, THESIS COMMITTEE:

Richard G. Baraniuk
Professor of Electrical and Computer
Engineering

Hyeokho Choi
Faculty Fellow/Lecturer of Electrical
and Computer Engineering

Mark P. Embree
Assistant Professor of Computational
and Applied Mathematics

Houston, Texas

April, 2005

Abstract

Distributed Alternating Localization-Triangulation of Camera Networks

by

William E. Mantzel

Localization, estimating the positions and orientations of a set of camera nodes, is a critical first step in camera-based sensor network applications such as geometric estimation and scene reconstruction. We propose a distributed algorithm for camera network localization based on feature point correspondences between multiple cameras with sparse overlapping view structure. We prove convergence of the iterative piecewise-linearized algorithm using the projection onto convex sets (POCS) principle, since this algorithm corresponds to projecting onto subspaces that approximately overlap. We provide bounds on convergence rates and worst-case errors and show experimental results from actual images. Finally, we introduce a new technique to obtain initial localization estimates based on trajectory observations.

Acknowledgments

My first thanks to my advisor Dr. Richard Baraniuk for providing invaluable guidance and feedback. Next, I would like to thank Dr. Hyeokho Choi for challenging and critiquing my work. I would also like to thank Dr. Mark Embree, a surprising wealth of technical expertise, stimulating suggestions, and support. Next I would like to thank Blaine Morrow for his help in the experimentation and Chad Leahy for his help in producing high quality 3-D figures. I also acknowledge and appreciate all those around the department as well as Dr. Jan Hewitt who generously gave their time reviewing draft after draft. Finally, I would like to thank my friends and family (and my loving parents in particular) for their unending patience and support.

Contents

Abstract	ii
Acknowledgments	iii
List of Illustrations	vi
1 Introduction	1
1.1 Sensor Networks	1
1.2 Computer Vision Applications	2
1.3 Camera Localization	2
1.4 Distributed Camera Localization	5
1.5 Overview	7
2 Camera Models and Theory	8
2.1 Coordinate Systems	8
2.2 Camera Model	10
3 Distributed Localization	14
3.1 Localization and Triangulation	14
3.2 <i>DALT</i> Algorithm	17
3.3 <i>DALT</i> Extension for Omni-Directional Cameras	19

4	Convergence Analysis	22
4.1	<i>DALT</i> with Fixed Orientations (Ideal Case)	22
4.2	<i>DALT</i> with Fixed Orientations (General Case)	28
5	Practical Considerations	34
5.1	Initialization	34
5.2	Trajectory Estimation	35
5.3	Feature Point Correspondence	37
5.4	Communication/Computation Tradeoff	41
5.5	Network Robustness	43
6	Results	45
6.1	<i>DALT</i> Simulation	45
6.2	Omni-Directional Camera Experiment	49
7	Conclusions	53
A	Sufficient Conditions for Localization Uniqueness	55
	Bibliography	59

Illustrations

2.1	Coordinate System Transformation	12
3.1	Omni-directional cameras	20
5.1	Trajectory Estimation	38
6.1	Results on omni-directional dataset	51
6.2	Error plots for omni-directional dataset	52
A.1	Microcluster Topology	57

Chapter 1

Introduction

1.1 Sensor Networks

Increasingly powerful microprocessors, along with recent advances in micro-sensor technology, wireless communications, and power efficiency, provide new opportunities to sense, compute, and discover via wireless sensor networks. A *sensor network* consists of a collection of nodes scattered throughout an environment of interest. Each node comprises a power supply, microprocessor, wireless communication system, and a set of sensors.

Typically it requires less energy to compute on a quantity than transmit it to another node or out of the network. So when the power supply is limited, it is important to minimize inter-node communications (and hence energy consumption) by pushing as much signal processing into the network as possible. This drives a need for communication-efficient, distributed algorithms [1].

Because most sensor network applications estimate spatially dependent events, the sensor nodes must be able to determine where they are located, a process called localization. Many localization techniques use acoustic delays and RF intensities to estimate positions [2]. However, little work has been done to make use of cameras in distributed localization. As CCD and CMOS image/video cameras become smaller, less expensive, and more power efficient, it will become possible to include cameras on the nodes of a

sensor network, enabling localization through image information.

1.2 Computer Vision Applications

After localization, many useful tasks in computer vision are enabled. Prime examples are photogrammetry, 1-D feature point correspondence, image-based navigation, and image-based rendering. Photogrammetry involves making metric measurements using images. This is useful in aerial cartography, surveying, and estimating the 3-D geometry of a scene.

Localization effectively simplifies feature point correspondence. For example, given two cameras A and B having overlapping views knowledge of where the cameras are makes it easier to match up points. Localized camera nodes can help some nearby user perform image-based navigation by estimating the user's position and orientation based on the sensors' and user's views. Image-based rendering and virtualized reality create an image from a novel viewpoint by piecing together pixel values from nearby cameras. For example, view morphing is a technique used in movies to create the effect of a moving camera, via the use of several precisely placed static cameras. All of these applications can be implemented on an camera network, once the localization problem is solved.

1.3 Camera Localization

Given camera nodes in set \mathbb{F} at unknown locations, we would like to estimate each camera's position and orientation using the content of all their images. The most common way to approach this problem is to find a set \mathbb{P} of feature points in 3-D space that appear in

known 2-D locations in several of the images from the cameras, the so-called “feature point correspondence problem”. The camera’s measurement of a given point’s location in the image plane makes is equivalent to the estimated direction from camera to that point in 3-D space.

A natural metric between two points in an image is then the angle between the two corresponding directions. Suppose we define $\theta_{f,p}$ as the angle between the measured feature point ray and the actual ray to that point. Using the idea that we should place cameras and points in a configuration that is most consistent with their measurements, we can generally state this localization problem as:

$$\min \sum_{f \in \mathbb{F}} \sum_{p \in \mathbb{P}(f)} c(\theta_{f,p})^2, \quad (1.1)$$

where $c(\cdot)$ is a continuous strictly monotonic function with $c(0) = 0$. Note that the inner sum is only carried out over points that are visible by a camera f , denoted $\mathbb{P}(f)$, and that we do not assume that all points are visible to all cameras. Estimating this optimal configuration is known as the sparse camera localization problem.

Because the equations describing the relation between feature points and cameras’ poses are nonlinear, researchers have found various ways to circumvent this problem. Kanade et al. have used a paraperspective projection model that produces a structure in the equations so that all the parameters can be recovered through the singular value decomposition. However, the modelling error introduced through the paraperspective approximation may produce undesirable results. Also, the paraperspective factorization handles

cases where the set of cameras all have an overlapping view and share some common set of points, an assumption that is often too restrictive.

A similar problem, the structure-from-motion problem, involves using a sequence of images to estimate both the motion of the camera and the structure of the environment. However, most such approaches deal with sequences of images from a single camera, not sets of images from multiple stationary cameras. The distinction is subtle, but there are three main differences between the distributed camera network localization problem and the motion recovery problem that prevent us from applying previous work directly. First, in the motion recovery problem it is often reasonable to assume that the images are ordered in a sequence where adjacent images differ in their point of view only slightly, and that the feature points of interest are common to all images [3]. In our problem, the camera connectivity structure defined through overlapping views is usually sparse and complex. Also, the localization of a sensor network needs to be distributed so that each camera sensor makes estimates based on localized information without knowledge from the other end of the network. Finally, feature point correspondence between two images is typically a difficult problem unless the views are known to be similar. However, in the case of camera network localization, it is made considerably easier by solving correspondences between *moving* objects, a luxury the motion recovery researchers did not have.

Brand et al. have recently presented an elegant and effective solution to the sparse camera network localization problem that does not require cameras to be placed into groups [4]. Reporting millimeter level accuracy, they claim substantial improvements over state of the

art methods. However, their approach requires centralized processing on all acquired images.

1.4 Distributed Camera Localization

Though this localization optimization could be solved by any of the aforementioned techniques in a centralized framework, a sensor network works under a different constraint. A sensor network is severely limited by its energy source, which is in turn limited by its small size. Since wireless communication is the most energy-demanding operation of a sensor node, any practical sensor network application must communicate and compute in a conservative way. Hence, it is not feasible to transmit entire images across the network so that some super-node can run an iterative nonlinear optimization algorithm.

Previous researchers have used epipolar geometries to tackle distributed alignment and mosaicking, similar to the localization problem. Stein et al. have proposed a method where the relative geometry of a set of images is discovered by the use of planar motion [5]. They employ the perspective model directly and use planar motion to their advantage by computing planar homographies. This requires scene motion within a plane, though some precautions are taken to make their approach robust to outliers. This assumption is often reasonable, as for example in urban scene monitoring when the distance from the camera to ground plane is very large compared to the scale of moving objects. This technique falls into a class of techniques that uses point correspondences between pairs of cameras to recover the relative epipolar geometry between that pair and to use each relative geometry to

compute the camera network localization. Such approaches work well when the overlapping view structure of the camera network is roughly linear, i.e. when each camera has an overlapping view with at most two cameras.

In contrast, without specifying pairs of cameras or relying on stereo techniques, we propose a localization technique that is well suited for more camera networks with more complex overlapping view structures. This technique first uses a least-squares algorithm from photogrammetry to estimate feature points in 3-D space using known cameras. In turn these newly estimated points are used to help cameras estimate their position and orientation using the Direct Linear Transform (DLT) [6] with the un-approximated perspective projection model.

We call this procedure *Distributed Alternating Localization-Triangulation (DALT)* because it uses localized cameras to estimate the position of points and then uses those points to localize each camera. This approach works well for camera networks because it is distributed, widely applicable to a variety of camera network configurations, robust to network outages, and proven to converge to its optimum.

Our chief contribution in this thesis is a camera network localization algorithm that is distributed, proven to converge to a local optimum, and capable of handling a sparse overlapping view topology. Our work does not describe a complete system, but rather a component of a larger machine. This component is heavily dependent on feature correspondence and tracking. We acknowledge the previous work that has been done in this area [7, 8], but concede that there are likely necessary advancements that must be made in

effectively distributing this task across a sensor network in order for our algorithm to be most effective.

Because images alone are incapable of producing measurements of absolute scale, there must be some additional information to truly localize these cameras. In practice, this initial estimate may be provided by another localization technique, or by a magnetometer, accelerometers or GPS receivers on a few sensor nodes. In the absence of this initial information, some relative calibration information can be obtained through epipolar calibration [9] or through trajectory estimation, as discussed in Section 5.2.

1.5 Overview

The remainder of this thesis is organized as follows. In Chapter 2, we explain the necessary background and theory. In Chapter 3, we describe the *DALT* method for localizing a sensor network and prove convergence. In Chapter 4, we analyze the theoretical properties of this algorithm. In Chapter 5, we discuss practical details for an efficient implementation such as networking concerns or estimation of scale. In Chapter 6, we show simulated and real-world results for our *DALT* algorithm. Finally, we conclude with a discussion of our approach and remarks about potential extensions applications.

Chapter 2

Camera Models and Theory

This chapter gives an overview of transformations between coordinate systems so that the reader can understand the subsequent geometrical relationships presented. It also introduces the perspective or pinhole camera model and describes how the camera's intrinsic parameters relate a pixel coordinate to a direction in 3-D space. Following this overview of coordinate system transformations and camera models, we construct a bilinear set of constraints useful in localization.

2.1 Coordinate Systems

In order to understand the relation of points in 3-D space to their corresponding 2-D image coordinates, we must first understand the transformation between coordinate systems.

For the purposes of this thesis, each three-dimensional (3-D) visible point of interest $X = [X_x \ X_y \ X_z]^T \in \mathbb{R}^3$ will be called a *feature point*. Each feature point may be represented with coordinates using the usual orthonormal canonical basis $X = X_x e_1 + X_y e_2 + X_z e_3$. In this thesis, the canonical basis will represent a world-referenced coordinate system. For example, e_1 , e_2 , and e_3 could represent one-meter-long vectors along the directions of East, North, and Up with respect to some fixed origin.

Alternatively, this feature point could be expressed in a camera's coordinate system defined by orthonormal basis vectors R_x , R_y , and R_z and shift vector S . Here, the shift

vector S represents the shift of the alternate coordinate system's origin to the origin of the world-referenced coordinate system. This representation may be written as

$$X = \mathbf{R}_x(\bar{X}_x - S_x) + \mathbf{R}_y(\bar{X}_y - S_y) + \mathbf{R}_z(\bar{X}_z - S_z)$$

for some $\bar{X} = [\bar{X}_x \ \bar{X}_y \ \bar{X}_z]^T \in \mathbb{R}^3$ and fixed $S = [S_x \ S_y \ S_z]^T \in \mathbb{R}^3$. This relation can also be expressed in matrix-vector notation as:

$$X = \mathbf{R}(\bar{X} - S)$$

or the reverse transformation:

$$\bar{X} = \mathbf{R}^T X + S. \quad (2.1)$$

We could also express this relation in terms of $T = -\mathbf{R}S$, the camera coordinate system's origin expressed in the world coordinate system (the location of the camera). After substitution, this coordinate system representation becomes

$$\bar{X} = \mathbf{R}^T(X - T). \quad (2.2)$$

We define the orthogonal rotation matrix as $[\mathbf{R}_x \ \mathbf{R}_y \ \mathbf{R}_z] = \mathbf{R} \in \mathbb{R}^{3 \times 3}$ and require $\det(\mathbf{R}) = +1$ (orthogonal matrices with $\det(\mathbf{R}) = -1$ are known as *reflections*). It will be convenient to parameterize \mathbf{R} in terms of a vector $\omega \in \mathbb{R}^3$ as

$$\mathbf{R} = \begin{bmatrix} r_{1x} & r_{1y} & r_{1z} \\ r_{2x} & r_{2y} & r_{2z} \\ r_{3x} & r_{3y} & r_{3z} \end{bmatrix} = \exp \left(\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \right) \quad (2.3)$$

as discussed by Grassia [10]. The effect of multiplication by this rotation matrix is to rotate the coordinates about the axis ω by an amount $\|\omega\|$ (in radians). Note in particular that $R\omega = \omega$, so that this direction vector (expressed as the difference between two points) has the same representation in either coordinate system.

All coordinate systems are thus denoted by their basis vectors and translation vector and are referenced to the canonical world coordinate system unless otherwise stated. A coordinate system transformation referenced to any other coordinate system will be called a relative transformation.

For this thesis, we will primarily be interested in the representation of feature points in camera-based coordinate systems, where the origin is the focal point of the camera and the orthonormal basis vectors coincide with the cameras axes. In particular, R_x , R_y , and R_z will represent the Right-Down-Out basis, where the Out vector R_z coincides with the camera's optical axis and the Right and Down vectors R_x and R_y vectors are chosen appropriately in the plane normal to the optical axis, called the "image plane".

2.2 Camera Model

In this section, we discuss the relation between feature points in 3-D space and their projection in the 2-D image plane. After doing so, the relationship between the scale factor of similarity and the information from images will be clear.

Various projection models have been proposed for the sake of computational simplicity (such as the affine or orthographic model and the paraperspective model). However, the

most accurate model remains the perspective or pinhole model simply because it is the only model that requires light to travel in straight lines. In geometrical terms, this constraint requires a set of points that are collinear with the camera's focal center to project to the same point on the image plane. This model is illustrated in Fig. 2.1. As shown, the image coordinates are related by a scale factor of similarity to the point coordinates in the camera's coordinate system. This projection can be written as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\bar{X}_x}{\bar{X}_z} \\ \frac{\bar{X}_y}{\bar{X}_z} \end{bmatrix}. \quad (2.4)$$

These normalized pinhole image coordinates are related to the actual image coordinates by a continuous function $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$. This function is commonly modeled as either affine or a slightly more involved parameterized function that takes into account terms of radial distortion caused by the lens. Previous work by Tsai, Zhang, Heikkila, Silven, and Bouguet has led to powerful and practical means for computing these parameters [11–14]. Hence, in practice, f can be effectively inverted with sub-pixel accuracy. Since these intrinsic parameters vary on a small scale (less than their measurement precision) over a wide variety of environmental conditions, they can often be solved off-line before deployment. Hence, for the remainder of the thesis, we will assume that we have access to the normalized image coordinates $[u, v]^T$.

Combining transformation $\bar{X} = \mathbf{R}^T X + S$ with equation (2.4) yields the following two

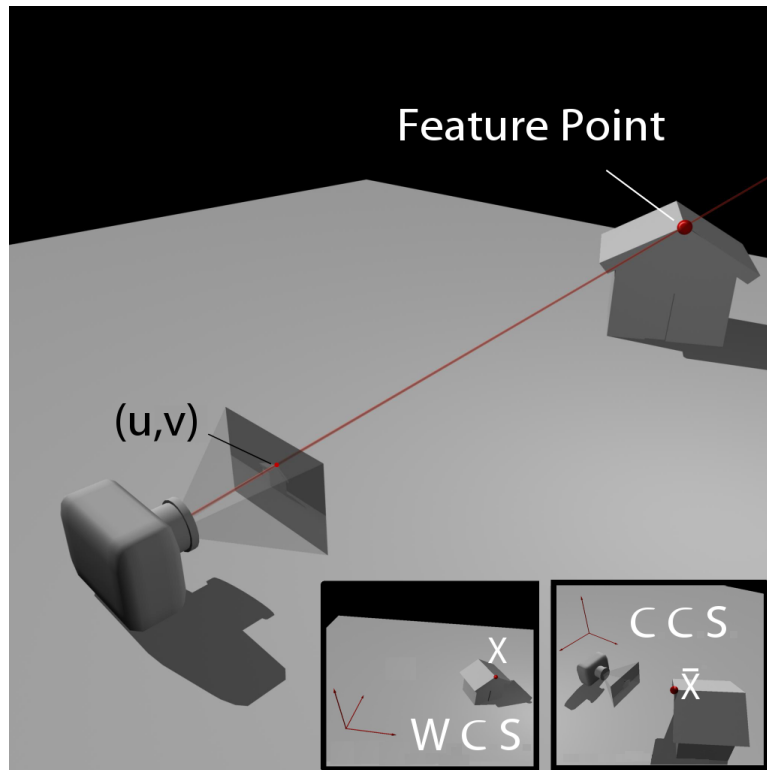


Figure 2.1 : A single point has a coordinate representation in both the World Coordinate System (WCS) and Camera Coordinate System (CCS) as X and \bar{X} . Its image coordinates are obtained by tracing the line from the point to the camera's focal center, and observing where it crosses the hypothetical image plane. After this perspective projection, the normalized image plane coordinates are given by $[u_{f,p}, v_{f,p}]$.

constraints for each camera/feature point pair:

$$u(\mathbf{R}_z^T X + S_z) = \mathbf{R}_x^T X + S_x \quad (2.5a)$$

$$v(\mathbf{R}_z^T X + S_z) = \mathbf{R}_y^T X + S_y. \quad (2.5b)$$

Alternatively, we could write this set of constraints for T as:

$$u\mathbf{R}_z^T(X - T) = \mathbf{R}_x^T(X - T) \quad (2.6a)$$

$$v\mathbf{R}_z^T(X - T) = \mathbf{R}_y^T(X - T). \quad (2.6b)$$

Later, we will use different representations depending on the type of estimation that is being performed.

With this bilinear relation between locations of points and cameras in 3-D space, image coordinates, and orientation of cameras, we are ready to approach a distributed solution to the localization problem in the next chapter.

Concerning notation, the image coordinates of feature point $p \in \mathbb{P}$ observed by camera $f \in \mathbb{F}$ will be denoted $[u_{f,p}, v_{f,p}]^T$. Note that many such $[u_{f,p}, v_{f,p}]^T$ will not be defined due to visibility constraints and occlusions. We will write $X_p \in \mathbb{R}^3$ for the p th point in the canonical world coordinate system. \mathbf{R}_f and T_f will represent camera f 's orientation and translation respectively. We will denote matrices in boldface and distinguish vectors and scalars by using upper and lowercase respectively. The exception to this rule is the vector ω . All norms are the ℓ_2 norm.

Chapter 3

Distributed Localization

Now that we understand the geometrical structure of images, we can discuss the approach towards localizing a sensor network. In this chapter, we will develop the *Distributed Alternating Localization-Triangulation (DALT)* algorithm, an iterative technique for estimating cameras' positions and orientations. Finally, we will prove convergence of this procedure and analyze convergence rates.

3.1 Localization and Triangulation

Equations (2.6a) and (2.6b) from the previous chapter are nonlinear if \mathbf{R}_f , S_f and X_p are all unknown. Such problems typically do not have closed form solutions. However, if we know \mathbf{R}_{f_i} and S_{f_i} for $n \geq 2$ cameras $\{f_1, f_2, \dots, f_n\} \subset \mathbb{F}$ that see point $p \in \mathbb{P}$, we can estimate X_p .

For each camera, equations (2.6a) and (2.6b) give

$$u(\mathbf{R}_z^T X + S_z) = \mathbf{R}_x^T X + S_x$$

$$v(\mathbf{R}_z^T X + S_z) = \mathbf{R}_y^T X + S_y,$$

which can be rewritten using the coordinate vectors as

$$u((\mathbf{R}e_3)^T X + e_3^T S) = (\mathbf{R}e_1)^T X + e_1^T S$$

$$v((\mathbf{R}e_3)^T X + e_3^T S) = (\mathbf{R}e_2)^T X + e_2^T S.$$

Accumulating these equations for each of the $n \geq 2$ cameras gives an over-determined linear system for the $X_p \in \mathbb{R}^3$

$$\begin{bmatrix} (\mathbf{R}_{f_1}(u_{f_1,p}e_3 - e_1))^T \\ (\mathbf{R}_{f_1}(v_{f_1,p}e_3 - e_2))^T \\ (\mathbf{R}_{f_2}(u_{f_2,p}e_3 - e_1))^T \\ \vdots \\ (\mathbf{R}_{f_n}(v_{f_n,p}e_3 - e_2))^T \end{bmatrix}_{2n \times 3} \begin{bmatrix} X_p \end{bmatrix}_{3 \times 1} = \begin{bmatrix} (e_1^T - u_{f_1,p}e_3^T)S_{f_1} \\ (e_2^T - v_{f_1,p}e_3^T)S_{f_1} \\ (e_1^T - u_{f_2,p}e_3^T)S_{f_2} \\ \vdots \\ (e_2^T - v_{f_n,p}e_3^T)S_{f_n} \end{bmatrix}_{2n \times 1}, \quad (3.1)$$

that we solve in the least-squares sense. This process is called *triangulating* a point.

Similarly, if we know the locations of $m \geq 6$ points $X_{p_i} \{p_1, p_2, \dots, p_m\} \subset \mathbb{P}$ that are visible to camera f , we can formulate a null space equation to estimate \mathbf{R}_f and S_f ,

$$\begin{bmatrix} X_{p_1}^T & 0_{1 \times 3} & -u_{fp_1}X_{p_1}^T & 1 & 0 & -u_{fp_1} \\ 0_{1 \times 3} & X_{p_1}^T & -v_{fp_1}X_{p_1}^T & 0 & 1 & -v_{fp_1} \\ X_{p_2}^T & 0_{1 \times 3} & -u_{fp_2}X_{p_2}^T & 1 & 0 & -u_{fp_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0_{1 \times 3} & X_{p_m}^T & -v_{fp_m}X_{p_m}^T & 0 & 1 & -v_{fp_m} \end{bmatrix}_{2m \times 12} \begin{bmatrix} \mathbf{R}_f e_1 \\ \mathbf{R}_f e_2 \\ \mathbf{R}_f e_3 \\ S_f \end{bmatrix}_{12 \times 1} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}_{2m \times 1}. \quad (3.2)$$

This ‘‘camera-from-points’’ technique follows from the direct linear transformation (DLT) [6]. This equation is a special case of the DLT equation when the linear intrinsic parameters are known.

This problem of simultaneously estimating the parameters of \mathbf{R}_f and T_f can be written

as:

$$\min_{\|Z\|=1} \|\mathbf{F}Z\|,$$

for parameter vector Z .

The solution to this system is a null space estimate of the $2m \times 12$ DLT matrix \mathbf{F} that we obtain by taking the eigenvector corresponding to the smallest eigenvalue of $\mathbf{F}^T \mathbf{F}$. The sign of the eigenvector is chosen such that the determinant of the orientation matrix is positive. The camera's orientation and translation $\hat{\mathbf{R}}$ and \hat{S} are then determined up to a positive scale factor.

However, the $\hat{\mathbf{R}}$ determined in this process may not be orthogonal. We can enforce orthogonality with the singular value decomposition (SVD) of the orientation matrix, given as $\mathbf{U}^T \hat{\mathbf{R}} \mathbf{V} = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$. The closest valid orientation (in Frobenius norm) to the set $\{\alpha \hat{\mathbf{R}} : \alpha > 0\}$ is $\mathbf{U} \mathbf{V}^T$ with corresponding translation $\frac{1}{\sigma} \hat{S}$ for $\sigma = \frac{1}{3}(\sigma_1 + \sigma_2 + \sigma_3)$.

There are alternative approaches to solving this linear system and then forcing the result onto the three-dimensional space of orthogonal matrices with positive determinant. Nister presents an efficient solution to the five-point relative pose problem that can be easily extended to this case for localization [15]. Also, given an initial estimate for position and orientation as above, one can optimize the least-squares constraints (2.6a) and (2.6b) directly on ω and T while keeping X_p fixed. Here, we use the Levenberg-Marquart algorithm for this nonlinear least-squares optimization to ensure a monotonically decreasing and thus converging cost function. [16]

Armed with these two techniques, we are now able to approach the distributed local-

Algorithm 1 *DALT*

while localization unknown **do**

if received localization broadcast of 2 or more linked cameras **then**

use these neighboring localizations to triangulate 6 or more feature points

initial self-localization from these points

end if

end while

repeat

broadcast localization estimate of self to linked cameras

use all available localizations to triangulate as many feature points X_p as possible

with fixed orientation, optimize translation T_f

with fixed translation, optimize orientation \mathbf{R}_f

until localization estimate converges

ization problem.

3.2 *DALT* Algorithm

As we saw in the previous section, by alternating between localization and triangulation on each camera node in an iterative fashion we are able to compute and propagate a globally consistent joint localization estimate across the network.

The pseudocode in Algorithm 1 describes the algorithm that runs on each node at a high level. The algorithm begins when two or more neighboring nodes with initial local-

ization estimates first broadcast their position to other neighboring cameras. These other cameras then triangulate visible points, use the points' estimates to compute their own initial positions, then broadcast the estimate to their neighbors. Through this algorithm, the triangulation of positions of new feature points lead to the localization of previously unlocalized cameras and vice versa. This algorithm is easily distributable if each camera node localizes itself using triangulated points and then collaborates with nearby camera nodes to triangulate other common points. Now we will give the details behind lines 5, 6, 10, 11, and 12.

Lines 5, 10: Triangulation. Given estimates for the translations and orientations of at least two cameras that view a common feature point, we estimate that point's 3-D position X by solving a least-squares system.

Line 6: Initial Localization. Conversely, if there exist at least 6 feature points with known 3-D positions, then we use (2.5a) and (2.5b) to estimate the twelve parameters in R and S , the orientation and translation of the camera as shown in (3.2).

Line 11: Optimizing Translation. In the case where there already exists some estimate of orientation, we estimate the translation of the camera T in a manner similar to triangulation while leaving the estimate for orientation fixed. As in triangulation, only two or more points are needed.

Line 12: Optimizing Orientation. To optimize orientation while keeping the translation estimate fixed, we use the Levenberg-Marquart algorithm to optimize each camera's orientation estimate individually [16]. Using constraints (2.6a) and (2.6b) in a least-squares

optimization ensures that the new orientation will match those constraints more closely than their previous values. The Levenberg-Marquart is a robust numerical optimization procedure guaranteed to converge, but the point of convergence may only be a local optimum.

We use the exponential map parameterization from Chapter 2 equation (2.3) to optimize the three parameters ω_x , ω_y , and ω_z directly, rather than use linear methods and later orthogonalize the rotation matrix \mathbf{R} . Though this is a nonlinear optimization, it does not present much of a computational challenge to the camera node because only three degrees of freedom are involved (resulting in a 3×3 matrix inverse). Also, each node has all the information it needs to perform this optimization locally, so no communication is necessary.

3.3 *DALT* Extension for Omni-Directional Cameras

Some camera network deployments may have nodes that are able to *actuate*. That is, the camera may be rotated around mechanically over a wide range of views in order to focus on an event that catches its attention. [17]. Still other networks may comprise omni-directional cameras whose images span a wide field of view, usually through the use of a mirror [18]. In this case, instead of measuring image coordinates on a hypothetical image plane with unit distance away, a more natural projection surface here is the unit sphere, denoted \mathbb{S}^2 . We still employ the perspective projection model; however, now the coordinates are given as $V = [V_x \ V_y \ V_z]^T$ where $\|V\| = 1$. The relation between the point camera based coordinates and the image coordinates V is now $\bar{X} = V\|\bar{X}\|$. In other words, the 3-D feature point \bar{X} lies on the line spanned by V . For this representation, we use the following

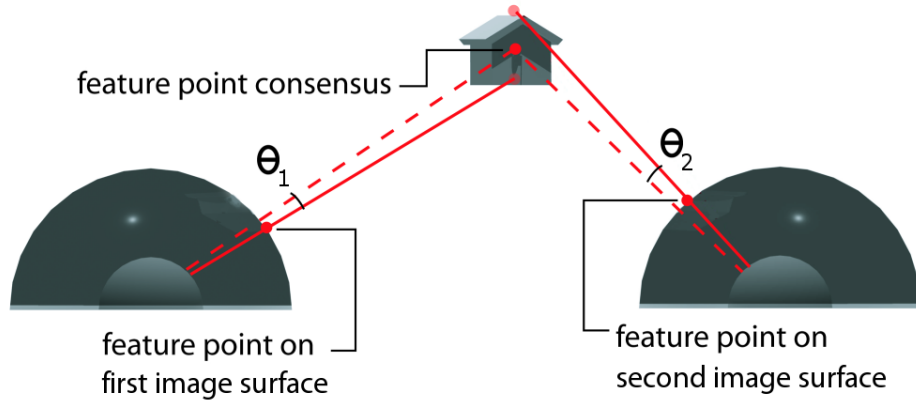


Figure 3.1 : Omni-directional projection model. The feature point shown is triangulated using a least-squares technique that attempts to find the point in 3-D space that is the most consistent with the feature directions from each node that views this point.

constraints:

$$V_x \mathbf{R}_z^T (X - T) = V_z \mathbf{R}_x^T (X - T), \quad (3.3a)$$

$$V_z \mathbf{R}_y^T (X - T) = V_y \mathbf{R}_z^T (X - T), \quad (3.3b)$$

$$V_y \mathbf{R}_x^T (X - T) = V_x \mathbf{R}_y^T (X - T). \quad (3.3c)$$

The first constraints (3.3a) and (3.3b) follow naturally from the perspective planar constraints (2.6a) and (2.6b). In fact, the perspective planar case can be thought of as a particular case of the omni-directional case when $V = [u \ v \ 1]^T$ and the unit norm constraint is removed. The last constraint (3.3c) is linearly dependent on the first two and is required to enforce symmetry between all axes.

We note that minimizing the squared error in these constraints is equivalent to minimizing $\|V \times \bar{X}\|$. This constraint is then a natural candidate for the cost function (1.1) because of its direct relation to the angle between the feature direction and the actual direction.

More precisely, we are minimizing

$$\epsilon = \sum_{f \in \mathbb{F}} \sum_{p \in \mathbb{P}(f)} \|V_{f,p} \times \bar{X}_{f,p}\|^2 \quad (3.4)$$

$$= \sum_{f \in \mathbb{F}} \sum_{p \in \mathbb{P}(f)} \sin(\theta_{f,p})^2 \|\bar{X}_{f,p}\|^2 \quad (3.5)$$

$$= \sum_{f \in \mathbb{F}} \sum_{p \in \mathbb{P}(f)} \sin(\theta_{f,p})^2 \|\mathbf{R}_f^T(X_p - T_f)\|^2 \quad (3.6)$$

$$= \sum_{f \in \mathbb{F}} \sum_{p \in \mathbb{P}(f)} \sin(\theta_{f,p})^2 \|(X_p - T_f)\|^2, \quad (3.7)$$

where $\theta_{f,p}$ is the angle between the line drawn from camera f to point p , and the feature direction vector V for that camera/point pair. The process of triangulation for omnidirectional cameras is shown in Figure 3.1.

This error metric has the benefit of being intuitive and sensible, while being computationally advantageous because it is linear in both X_p and T_f . The only drawback to this metric is its dependence on $\|X_p - T_f\|$ so that the minimization will favor solutions where the feature points are closer to the cameras. Fortunately as we will later see, this “gravitational” effect that tries to pull the cameras and points to a single place can be effectively mitigated in most cases. As a simple illustration, note that for fixed camera positions, if a point is well aligned with many camera nodes, moving a point slightly closer to a camera may cause the alignment angular errors to increase drastically. This increase will prevent the gravitational effect from significantly affecting the estimate. In other words, the penalty incurred in increasing θ often prohibits the solution from significantly decreasing $\|X_p - T_f\|$. Fortunately, we can usually terminate the algorithm before this gravitational effect has a substantially negative impact.

Chapter 4

Convergence Analysis

In this section we analyze the performance of the *DALT* algorithm. We begin by specifying and analyzing a centralized version of the algorithm and then show that *DALT* is an equivalent distributed algorithm. Subsequently we will analyze the centralized algorithm and through its convergence, prove convergence of the distributed algorithm.

4.1 *DALT* with Fixed Orientations (Ideal Case)

Because the global set of equations describing the orientation, translation, and scene geometry is nonlinear, it is difficult to describe the centralized global solution. Furthermore, the orthogonalization of the \mathbf{R} matrix at each iteration of the DLT makes it difficult to analyze the error propagation in rotation. In this section, we will analyze this procedure in the case when orientation is fixed and does not change through iterations. We use Projection Onto Convex Sets (POCS) to show how the local solutions of the iterative *DALT* algorithm relate to the global optimum. For purposes of analysis, we assume that one camera is defined to be the origin. In practice, one may remove this constraint, though the localization will be free to “drift” unless certain anchor points are specified (via GPS for instance).

In this case, and when orientation is fixed, the constraints from equations (2.6a) and

(2.6b) become the linear equation.

$$\mathbf{A}T = \mathbf{B}X \quad (4.1)$$

or equivalently,

$$\begin{bmatrix} -\mathbf{A} & \mathbf{B} \end{bmatrix} \begin{bmatrix} T \\ X \end{bmatrix} = 0 \quad (4.2)$$

where $X = [X_1^T, X_2^T, \dots, X_P^T]^T$ and $T = [T_1^T, T_2^T, \dots, T_F^T]^T$.

We will consider the ideal case first, where there exists a non-trivial exact solution to (4.1) and (4.2) thanks to perfect accuracy in our feature point and orientation estimates. We then extend our results to the general case when the feature points and orientation estimates have errors in them.

Theorem 1 *For fully connected camera networks with two or more cameras, (4.2) has a unique solution up to a scale factor so that $\dim(\text{null}([- \mathbf{A} \ \mathbf{B}])) = 1$. This means that any valid solution to (4.2) is a scalar multiple of some non-trivial base solution.*

Proof: See appendix for proof and more relaxed conditions for uniqueness of localization for more general topologies. There it is shown that fully connected topologies of 3 or more camera nodes (such that each pair of cameras has an overlapping view) form a microcluster for which this condition holds. This condition also holds for 2 cameras with overlapping views. □

Corollary 2 *Matrices A and B have full column rank.*

Proof: By the previous theorem which states that $[-\mathbf{A} \quad \mathbf{B}]$ has a one-dimensional null space, there exist true localization T' and point cloud X' such that in general $[-\mathbf{A} \quad \mathbf{B}]\begin{bmatrix} T \\ X \end{bmatrix} = 0 \implies \begin{bmatrix} T \\ X \end{bmatrix} = \beta \begin{bmatrix} T' \\ X' \end{bmatrix}$ for some $\beta \in \mathbb{R}$. Without loss of generality let $\|\mathbf{A}T'\| = \|\mathbf{B}X'\| = 1$. It cannot be the case that \mathbf{A} and \mathbf{B} are both rank deficient since $\dim(\text{null}(\mathbf{A}^\perp) + \dim(\text{null}(\mathbf{B}^\perp)) \leq \dim(\text{null}([-\mathbf{A} \quad \mathbf{B}]^\perp)) = 1$. Assume for the sake of a contradiction that only one of \mathbf{A} or \mathbf{B} has a one dimensional null space. If $\mathbf{A}T' = 0$ for nonzero T' , then X' must be zero. Likewise, if $\mathbf{B}X' = 0$ for nonzero X' , then T' must be zero. If both X' and T' are nonzero, then we have a contradiction. \square

Corollary 3 $\text{Range}(\mathbf{A}) \cap \text{Range}(\mathbf{B}) = \{\alpha_1 \mathbf{A}T' : \alpha_1 \in \mathbb{R}\} = \{\alpha_2 \mathbf{B}X' : \alpha_2 \in \mathbb{R}\}$.

Proof: Theorem 1 and Corollary 2 show that the subspace $\text{Range}(\mathbf{A}) \cap \text{Range}(\mathbf{B})$ has dimension one. Given that $\mathbf{A}T' = \mathbf{B}X'$, this subspace is simply all scalar multiples of these vectors. \square

If an initial estimate for the positions of all the cameras T_0 was known, then we could compute a least-squares estimate for X as:

$$X_1 = (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \mathbf{A}T_0 = \mathbf{B}^\dagger \mathbf{A}T_0, \quad (4.3)$$

where \mathbf{B}^\dagger is the pseudoinverse of \mathbf{B} . Note that if $\mathbf{A}T_0 \in \text{Range}(\mathbf{A}) \cap \text{Range}(\mathbf{B})$ initially, then this least-squares solution would produce a solution $[T_0|X_1]^T = \alpha [T'|X']^T$ of (4.2) with scalar α where $[T'|X']^T$ is the true solution. If not, we could estimate T_1 from X_1 in a similar manner.

Estimating T from X and X from T iteratively produces a technique for iteratively localizing when the orientations are known. This type of technique is known as *Alternating least-squares* [19]. The n -th iteration of this technique can be formulated as

$$AT_n = \prod_{i=1}^n (AA^\dagger BB^\dagger) AT_0 = C^n AT_0 \quad (4.4)$$

where $C = AA^\dagger BB^\dagger$.

We now show that this centralized operation behaves exactly as the distributed algorithm does. Subsequently, we will analyze the centralized algorithm (called *CALT*) and through its properties, infer those of *DALT*.

Now because each constraint (row) of A only involves one camera T_f and each constraint of B only involves one point X_p , they each have a block diagonal structure after appropriate permutation of their constraints (rows) as illustrated here:

lation operations. Hence, the centralized algorithm (*CALT*) is equivalent to the operations performed by the distributed algorithm (*DALT*).

Theorem 4 *When (4.2) has an exact unique solution up to a scale factor, CALT converges to the optimum solution up to a scale factor.*

$$T_n \rightarrow \beta T'$$

$$X_n \rightarrow \beta X'$$

Proof: Recall that the n th iteration of this *DALT* algorithm can be formulated as:

$$\mathbf{A}T_n = \prod_{i=1}^n (\mathbf{A}\mathbf{A}^\dagger \mathbf{B}\mathbf{B}^\dagger) \mathbf{A}T_0 = \mathbf{C}^n \mathbf{A}T_0 \quad (4.5)$$

The transformations $\mathbf{A}\mathbf{A}^\dagger$ and $\mathbf{B}\mathbf{B}^\dagger$ are orthogonal projections onto the subspaces $\text{Range}(\mathbf{A})$ and $\text{Range}(\mathbf{B})$ respectively. Since subspaces are convex, the theory of Projection onto Convex Sets (POCS) tells us that successive projections onto two convex sets whose intersection is nonempty will converge to an element in that intersection [20]. By Corollary 3 this intersection $\text{Range}(\mathbf{A}) \cap \text{Range}(\mathbf{B})$ comprises sets $\{\alpha_1 \mathbf{A}T' : \alpha_1 \in \mathbb{R}\} = \{\alpha_2 \mathbf{B}X' : \alpha_2 \in \mathbb{R}\}$. Thus, $\mathbf{A}T_n \rightarrow \beta \mathbf{A}T'$ for some $\beta \in \mathbb{R}$. It follows that $\mathbf{A}^\dagger \mathbf{A}T_n = T_n \rightarrow \beta \mathbf{A}^\dagger \mathbf{A}T' = \beta T'$, and by a similar line of reasoning, $X_n \rightarrow \beta X'$, where T' and X' are the solutions of (4.1). Furthermore, we can decompose our initial estimate T_0 into $\beta T' + T_\epsilon$ where $\mathbf{A}T_\epsilon \in (\mathbf{A}T')^\perp$, the space orthogonal to $\mathbf{A}T'$ so that $(\mathbf{A}T')^T (\mathbf{A}T_\epsilon) = 0$. One can see that $\beta \mathbf{C}^n \mathbf{A}T' = \beta \mathbf{A}T'$ for all n since $\mathbf{A}T' \in \text{Range}(\mathbf{A}) \cap \text{Range}(\mathbf{B})$. Also, $\mathbf{C}^n \mathbf{A}T_\epsilon \in (\mathbf{A}T)^\perp$ for all n . Since $\|\mathbf{C}x\|_2 < \gamma \|x\|_2$ when $x \in (\mathbf{A}T)^\perp$ for some $\gamma < 1$,

it follows that $C^n AT_e$ converges to zero, so that by linearity, T_n converges to $\beta T'$. This tells us that if we know orientation and our feature points perfectly, the *DALT* algorithm under fixed orientation will converge onto the subspace spanned by the vector $[X'|T']^T$. Hence, our algorithm will converge to some scalar multiple β of the actual camera localization configuration regardless of the initial estimates for X and T . This β could take on any value, even zero in some pathological cases (for example, when every node's initial position is orthogonal to its optimal position). However, β will better reflect the true scale if a good initial estimate is made. \square

4.2 *DALT* with Fixed Orientations (General Case)

In practice, there will be error in any estimate of orientation and feature points, so that the subspaces have a trivial common element $\text{Range}(\mathbf{A}) \cap \text{Range}(\mathbf{B}) = \{0\}$. POCS tells us that *DALT* will cause the estimate of the position of all feature points and cameras to converge to the origin (the effect previously mentioned as “gravity”. In spite of this disheartening result, we are able to show that successive projections lead to a weaker form of convergence to the least-squares optimum. First we will characterize this optimum.

We seek to minimize $\|\mathbf{AT} - \mathbf{BX}\|^2$. However, we must also impose a norm constraint on X and T so that this criterion is not trivially satisfied with X and T at the origin. Requiring $\|\mathbf{AT}\| = \|\mathbf{BX}\| = 1$ is a natural choice since we are minimizing $\|\mathbf{AT} - \mathbf{BX}\|^2$. Also, $\|\mathbf{AT}\|$ and $\|\mathbf{BX}\|$ define norms on X and T since \mathbf{A} and \mathbf{B} have full column rank.

Note that we are unable to constrain $\|T\|$ and $\|X\|$ directly since we don't know the relation of these magnitudes, only that $\|AT\|$ should be roughly $\|BX\|$. This minimum least-squares can easily be cast as a maximum inner product optimization since

$$\begin{aligned} \min_{\substack{\|AT\|=1 \\ \|BX\|=1}} \|AT - BX\|^2 &= \min_{\substack{\|AT\|=1 \\ \|BX\|=1}} \|AT\|^2 - 2(AT)^T(BX) + \|BX\|^2 \\ &= \min_{\substack{\|AT\|=1 \\ \|BX\|=1}} 1 - 2(AT)^T(BX) + 1 \\ &= 2(1 - \max_{\substack{\|AT\|=1 \\ \|BX\|=1}} (AT)^T(BX)). \end{aligned}$$

Fortunately, this maximum inner product is related to a quantity called the first *principle angle* between subspaces. In particular, the cosine of the first principle angle is defined as the maximum inner product between any unit vector from one subspace and any unit vector from another [21]. We leverage the related properties to analyze these minimizers and construct a direct method for solving this system.

To this end, let $A = Q_A R_A$ and $B = Q_B R_B$ be Q-R factorizations of matrices A and B such that $Q_A \in \mathbb{R}^{M \times 3P}$, $Q_A^T Q_A = I$, and R_A is upper triangular. Likewise, $Q_B \in \mathbb{R}^{M \times 3F}$, $Q_B^T Q_B = I$ and R_B is upper triangular. Let $q = \min(3P, 3F)$ and let the SVD of $Q_A^T Q_B$ be given as $Y^T (Q_A^T Q_B) Z = \text{diag}(\sigma_1, \dots, \sigma_q) = \Sigma$.

Theorem 5 (*Centralized ALT algorithm*) *The minimizers of*

$$\min_{\substack{\|AT\|=1 \\ \|BX\|=1}} \|AT - BX\|^2$$

are given as

$$\hat{T} = A^\dagger Q_A Y_1$$

$$\hat{X} = \mathbf{B}^\dagger \mathbf{Q}_B Z_1.$$

Proof: This SVD can be characterized as a maximization [21]

$$\begin{aligned} \sigma_1 &= Y_1^T \mathbf{Q}_A^T \mathbf{Q}_B Z_1 \\ &= \max_{\|Y\|=1} \max_{\|Z\|=1} Y^T (\mathbf{Q}_A^T \mathbf{Q}_B) Z \\ &= \max_{\|\mathbf{Q}_A Y\|=1} \max_{\|\mathbf{Q}_B Z\|=1} (\mathbf{Q}_A Y)^T (\mathbf{Q}_B Z) \\ &= \max_{\|\mathbf{A}T\|=1} \max_{\|\mathbf{B}X\|=1} (\mathbf{A}T)^T (\mathbf{B}X) \end{aligned} \quad (4.6)$$

$$= (\mathbf{A}\hat{T})^T (\mathbf{B}\hat{X}) \quad (4.7)$$

where equation (4.6) can be formulated by letting $Y = \mathbf{R}_A x$, $Z = \mathbf{R}_B t$ for invertible \mathbf{R}_A , \mathbf{R}_B (since \mathbf{A} and \mathbf{B} are full rank). Therefore,

$$\mathbf{A}\hat{T} = \mathbf{Q}_A Y_1$$

$$\mathbf{B}\hat{X} = \mathbf{Q}_B Z_1$$

$$\hat{T} = \mathbf{A}^\dagger \mathbf{Q}_A Y_1$$

$$\hat{X} = \mathbf{B}^\dagger \mathbf{Q}_B Z_1.$$

□

Theorem 6 *The CALT algorithm (and hence the equivalent DALT algorithm) converges to its optimal value up to a rate of exponential decay. More explicitly:*

$$\begin{aligned} \frac{1}{\sigma_1^{2n}} T_n &\rightarrow \beta \hat{T} \\ \frac{1}{\sigma_1^{2n-1}} X_n &\rightarrow \beta \hat{X} \end{aligned}$$

as $n \rightarrow \infty$.

Proof: It is readily seen that

$$\begin{aligned} \mathbf{A}\mathbf{A}^\dagger &= \mathbf{Q}_A \mathbf{R}_A (\mathbf{R}_A^T \mathbf{Q}_A^T \mathbf{Q}_A \mathbf{R}_A)^{-1} \mathbf{R}_A^T \mathbf{Q}_A^T = \mathbf{Q}_A \mathbf{Q}_A^T \\ \mathbf{B}\mathbf{B}^\dagger &= \mathbf{Q}_B \mathbf{R}_B (\mathbf{R}_B^T \mathbf{Q}_B^T \mathbf{Q}_B \mathbf{R}_B)^{-1} \mathbf{R}_B^T \mathbf{Q}_B^T = \mathbf{Q}_B \mathbf{Q}_B^T. \end{aligned}$$

Using these Q-R decompositions with the SVD $\mathbf{Y}^T (\mathbf{Q}_A^T \mathbf{Q}_B) \mathbf{Z} = \Sigma$ in (4.5) gives

$$\begin{aligned} \mathbf{A}T_n &= \mathbf{C}^n \mathbf{A}T_0 \\ &= \mathbf{Q}_A ((\mathbf{Q}_A^T \mathbf{Q}_B)(\mathbf{Q}_A^T \mathbf{Q}_B)^T)^n \mathbf{R}_A T_0 \\ &= \mathbf{Q}_A \mathbf{Y} \Sigma^{2n} \mathbf{Y}^T \mathbf{R}_A T_0. \end{aligned} \tag{4.8}$$

If $\sigma_1 = 1$, then the first principle angle is 0 and there exists an exact solution to $\mathbf{A}T = \mathbf{B}X$ as in the previous section. Otherwise, when $1 > \sigma_1 > \sigma_2 > \dots > \sigma_n$, we know that Σ^{2n} (and hence T_n) converge to zero. However, $\frac{1}{\sigma_1^{2n}} \Sigma^{2n}$ converges to $e_1 e_1^T$ as $n \rightarrow \infty$ so that

$$\begin{aligned} \frac{1}{\sigma_1^{2n}} \mathbf{A}T_n &\rightarrow \mathbf{Q}_A \mathbf{Y}_1 \mathbf{Y}_1^T \mathbf{Q}_A^T \mathbf{Q}_A \mathbf{R}_A T_0 \\ &= \mathbf{A}\hat{T} [(\mathbf{A}\hat{T})^T (\mathbf{A}T_0)] \\ &= \beta \mathbf{A}\hat{T} \end{aligned} \tag{4.9}$$

where β can be thought of as the result of an inner product that tries to preserve the original estimate of scale inherent in T_0 .

So far, we have shown that under fixed orientations, the estimate of the camera translation vector T converges to its least-squares optimum up to a rate of exponential decay. A

similar line of reasoning would show that the feature point position vector X converges to its optimum up to a rate of exponential decay as well. \square

To analyze convergence rates, note that the camera position initialization vector T_0 can be decomposed into $\beta\hat{T} + T_\epsilon$ where $\mathbf{A}T_\epsilon \in (\mathbf{A}\hat{T})^\perp$. Since $\mathbf{C}^n \mathbf{A}\hat{T} = \sigma_1^{2n} \mathbf{A}\hat{T}$ and $\|\mathbf{C}^n \mathbf{A}T_\epsilon\| \leq \sigma_2^{2n} \|\mathbf{A}T_\epsilon\|$, the relative error due to the T_ϵ term dies off as $O\left(\left(\frac{\sigma_2}{\sigma_1}\right)^{2n}\right)$ in the worst case.

Incidentally, these singular values σ_1 and σ_2 of $\mathbf{Q}_A^T \mathbf{Q}_B$ are the cosines of the principal angles θ_1 and θ_2 between subspaces $\text{Range}(\mathbf{A})$ and $\text{Range}(\mathbf{B})$ [21].

Fortunately, in many cases $\sigma_1 \approx 1$ while $\sigma_2 \ll 1$. For example, a 10-camera network viewing a common set of 50 points with feature point error of 5 pixels on a 320×240 display and an error in orientation of 5 degrees, we have $\sigma_1 = 0.9978$ and $\sigma_2 = 0.4997$. In this case, after only 5 *DALT* iterations the error term would drop to $\sigma_2^{10} \approx 0.001$ of its original value while the scale factor is only attenuated by a factor of $\sigma_1^{10} \approx 0.98$. This reassures us that the decay effect due to the first principle angle will be minor and that we can reduce most of the localization error without significantly affecting the initial scale factor estimate.

In cases where the tracking or orientation accuracy is low, the gravitational effects induced by a low value of σ_1 may be non-negligible. In other cases, the system may be ill-conditioned, likely due to an insufficient diversity of feature points or a poorly connected node, causing σ_2 to be closer to 1.

In these cases, the initial scale factor can be prevented from shrinking to zero by imposing a position constraint on a single camera or point with respect to another camera, point, or origin. These constraints can be included among the other least-squares constraints in the localization process and can be weighted appropriately so that it is not too overbearing on the rest of the *DALT* algorithm.

Chapter 5

Practical Considerations

Up to this point, we have presented the *DALT* technique for localization from a theoretical perspective. However, to implement this algorithm effectively, there are several details that should be considered. In this chapter, we first discuss several ways to obtain an initialization estimate including a novel technique called trajectory estimation. Next, we discuss how scene motion can aid the challenging correspondence problem via the epipolar constraint. Next, we explore the tradeoff between computation and communication in the context of a specific part of the algorithm. Finally, we discuss the robustness of the algorithm to link outages or the insertion of new nodes into the network.

5.1 Initialization

Since *DALT* is an iterative algorithm, there needs to be some initial estimate for the localization parameters. Initial estimates for the *DALT* iterations can be provided in several ways. First, in some cases a few “anchor” camera nodes could be able to estimate their positions via global positioning system (GPS) receivers. Second, the absolute orientation of each camera node could be inexpensively estimated with an accelerometer and magnetometer. In the absence of such sensors, the relative orientation between pairs of cameras can be extracted directly from the essential matrices determined from pairwise point corre-

spondences [22].

As we saw above in Chapter 4, this algorithm will converge regardless of initial information. However, the seven degrees of freedom discussed in Appendix A will more closely resemble their actual values if good initial estimates are available.

5.2 Trajectory Estimation

Without GPS information from two or more locations, the metric scale of the camera network is undetermined. This scale factor could be estimated by manually surveying certain feature points. For practical purposes, however, this manual intervention can be made much simpler by simply throwing an object. We now show how trajectory estimation can lead to recovery of the unknown scale factor as well as two out of three of the degrees of freedom needed to transform the orientation of the local camera network's coordinate system to the world-referenced coordinate system, all through the effects of gravity.

Surprisingly, the perspective projection of a parabolic trajectory contains enough information to estimate the coordinate system transformation between the camera and the trajectory. Consider the parabolic motion in the trajectory's coordinate system

$$X_{\Gamma}(t) = \begin{bmatrix} V_x(t - t_0) \\ \frac{1}{2}g(t - t_0)^2 \\ 0 \end{bmatrix}$$

where t_0 and V_x are the time of apex (when the projectile reaches its peak) and horizontal velocity of this trajectory, respectively. Since this motion is planar, we can estimate a

coordinate system transformation between the camera and the trajectory via a planar homography. If two cameras view the same trajectory, then we could compute their relative coordinate system transformations as the composition of this pair of transformations. Furthermore, this transformation is not just determined up to a scale factor. Because the physical constant of gravity is known to be roughly $9.8 \frac{m}{s^2}$ we can now determine the scale factor of similarity between these two cameras. Also, since the direction of gravity is known, we can estimate the orientation of the cameras with respect to the world coordinate system up to one degree of freedom (rotation about the gravity axis).

When t_0 and V_x are not known, they can be estimated using any two-dimensional non-linear optimization such as the Levenberg-Marquardt method [16]. Here, if the reprojection is performed after the homography is computed for a given t_0 and V_x , then we could use the reprojection error as the error metric to refine the two parameter estimates.

Using two off-the-shelf digital cameras with resolutions of 320×240 pixels and with different frame rates of 20 and 15 frames/second, we were able to evaluate this method with real data. Figure 5.1 depicts almost two seconds of video of a thrown tennis ball and the resulting estimated coordinate system transformation between the camera and the trajectory. Note that no feature point correspondence was used here. In fact the two cameras had different frame rates. Instead of transmitting sets of feature points, the camera node only transmits 6 scalars describing the coordinate system transformation and two scalars for the time of apex and horizontal velocity. The receiving camera can use the time of apex to determine whether or not they viewed a common trajectory. In practice, it is possible to

estimate this value to within 0.1 seconds (2 to 3 frames).

In several experiments, this method has been consistent to within 16 cm on a 4 meter baseline (4 % error) when the corresponding reprojection error was approximately 2 pixels. This would be considered poor performance for a final localization estimate, but it provides a reasonable starting point for the *DALT* algorithm to improve on and often provides the only means to obtain the scale factor and world coordinate system estimates. Simulation results have shown that the localization is accurately determined in the ideal case.

Figure 5.1 shows two photomosaics of a time sequence of a single tennis ball trajectory viewed by each of the cameras. After tracking the tennis ball in each frame, the relative coordinate system transformation is computed. As an illustration, the resulting estimated coordinate systems are reprojected and superimposed on each mosaic. Because the scale factor of similarity is also recovered, we can show the three axes with unit meter length.

The practical uses of trajectory estimation immediately follow. One can conveniently estimate the scale factor of similarity of the network and discover two out of three degrees of freedom of the orientation of the camera network with respect to the world coordinate system, all with the toss of an object.

5.3 Feature Point Correspondence

Many classic computer vision techniques are based on matched feature points. However, there are many obstacles that make correspondence a difficult problem. Typically, feature points are extracted from a pair of images using corner and edge detection algorithms.



Figure 5.1 : Trajectory Estimation algorithm to localize several cameras by estimating local transforms between their own coordinate system and that of the trajectory. Shown in (a) and (b) are the mosaiced video streams from cameras a and b respectively with the estimated coordinate system Γ superimposed. Since Trajectory Estimation also solves the scale factor of similarity, we show one meter long arrows in 3-D space

These points are matched using some sort of parameterized transformation, such as an affine transformation [8]. However, these features may not contrast as well in other images which could possibly cause a mismatch in the feature correspondence. Also, for a smooth curved surface, the point of occlusion (edge) from one vantage point could be different from the point of occlusion of another vantage point. These problems only become amplified when there is a large baseline between the cameras.

The use of scene motion for a fixed set of cameras makes the distributed camera network correspondence problem manageable [7]. By tracking moving objects, a camera is able to extract one feature point per object from each frame. If the object appears sufficiently small, then this feature point can be taken to be the object's centroid. This camera can

then broadcast these feature points to nearby cameras. If these cameras have been time synchronized, each feature point can be immediately matched with a simultaneous feature point in a nearby camera (after possibly interpolating to ensure that the feature points occur at the same sampling time). In this way, if the cameras were tracking the same object, then each time frame gives a new matched feature point. Otherwise, none of the points will match.

The question is no longer which feature points from the cameras are a match, but whether the cameras were looking at the same instance of motion in the first place. The epipolar constraint, discussed by Faugeras et al. [23], gives some assistance in solving this problem. For any normalized image coordinates $M_p = [u_{i,p}, v_{i,p}, 1]^T$ of camera i , and $M'_p = [u_{j,p}, v_{j,p}, 1]^T$ of camera j corresponding to the same point p observed in cameras i and j satisfy:

$$M_p^T \mathbf{E} M'_p = 0 \quad (5.1)$$

where \mathbf{E} is called the *essential matrix* and is specific to cameras i and j .

This essential matrix can be computed by the eight point algorithm [22] or any of a number of other ways [23].

After computing the best-fit essential matrix for this set of point pairs, $\hat{\mathbf{E}}$, the pair of point sets is determined to be a match if the following quantity is below a certain threshold:

$$\sum_p (M_p^T \hat{\mathbf{E}} M'_p)^2. \quad (5.2)$$

However, the threshold should depend on the variation of the points because small or trivial

motions (i.e., linear motion) would trivially satisfy this constraint.

Of course, this approach only addresses the problem when the *DALT* algorithm is allowed to pause until some motion is observed before continuing. To ensure that the constraint is not trivially met, the essential matrix is determined with a set of well conditioned matched points that yield no singularity. For example, linear motion with constant velocity is rejected. Also, the smallest moving entities are used so that the approximation error in using the object's centroid is minimized.

A similar approach was taken by Azarbayejani and Pentland [7], though they required their system to compute a simplistic linear intrinsic model since the more powerful calibration techniques had not yet been developed [12]. Likewise, Stein et al. used a similar technique while using an uncalibrated homography matrix instead of an essential matrix [5].

In some cases, the correspondence may produce outliers with high probability. Though the least-squares epipolar constraint has a measure of robustness built into it, other techniques may not. In this case, these bad point matches may be detected using their intermediate residual error values while running the *DALT* algorithm. For example, this residual error for omnidirectional images is $\|V_{f,p} \times \bar{X}_{f,p}\|$. The camera nodes could determine a global threshold before deployment, or they may be able to determine one in an ad-hoc manner after deployment by computing statistics of these residual values. Scherber and Papadopoulos recently presented one such method for distributed linear computation (e.g., taking an average) [24].

Regardless of the chosen method of correspondence, it is important to choose a diverse

set of points so that the *DALT* computation isn't ill conditioned. This could happen for instance if all of the points are co-linear, or if all of the points are taken from a small region in space, or if there are an insufficient number of points.

5.4 Communication/Computation Tradeoff

A fundamental theme of the sensor network paradigm is the importance of distributed local computation and in-network processing rather than sending all information to a central processor. Clearly, these power savings are not present when the network is small enough or well connected enough that it is just as easy to communicate to a base station node as neighboring camera nodes. Rather, the power efficiency of distributed computation begins to present itself as the network grows and becomes more complex.

Consider the localization of a 300-node network of where each node requires an average of 10 hops to reach the base station. Suppose the average number of feature points observed per camera node is 100. Sending these 2-D feature points to the base station would require 2000 scalars to be transmitted on average. If we were to instead use the *DALT* algorithm, this number would be reduced to $200 + 6n$ where n is the number of iterations of the algorithm. As the average number of hops to base station grows, this gap will only widen. It is then clear that for sufficiently large and complex networks, we must use a distributed approach.

We wish to perform as many of the computations locally as possible in order to aggressively minimize the communication requirements. For example, line 10 of the *DALT*

algorithm in Chapter 3 involves a triangulation operation that could be computed by a single camera node that would then broadcast the result to other nodes. Instead, each camera node is required to compute this result so that this broadcast is unnecessary.

To get a concrete idea of the energy savings this redundant computation provides over the otherwise necessary communication in this particular scenario, consider the *MoteIV telos* platform [25]. This sensor network architecture utilizes the Texas Instruments MSP430 processor drawing 3mW of power and running at 8MHz. Its Chipcon communication module draws 50mW of power and transfers data at a rate of 250kbps.

The least-squares computation necessary to solve this triangulation will take $\frac{2}{3}n^3 + n^2m + nm + 2n^2$ multiply-accumulates for m constraints and n unknowns. In this case, there are a pair of constraints for every camera contributing to this point's triangulation, and three unknowns for the point's position. Supposing that 5 cameras viewed this point, this operation requires under 200 multiply-accumulates for this point. On this 8MHz processor this entire computation would terminate in less than 20 μ s, and consume about 60 nJ of energy per point per iteration on each processor or **300 nJ** total. In contrast, sending each 3-D point as three 16 bit scalars would require 250 μ s of the communication system leading to more than **10 μ J** per point per iteration as shown in Table 5.4. As a conservative estimate, we assume that the camera node only needs to broadcast this vector once, instead of multiple times to each neighboring camera.

This rough computation gives an idea of the order of magnitude difference between computation and communication. Note that this factor does not depend on the number of

Table 5.1 : Energy comparison between Method A and Method B. In Method A, a specific camera node triangulates the positions of feature points and then broadcasts this result to its neighbors. In Method B, every camera node does this computation, but does not broadcast the result.

Method A (μJ)	Method B (nJ)
10	300

feature points because both energies would scale up linearly. In addition to saving energy, this particular method also simplifies the system because the designation of which nodes will perform specific triangulations is no longer necessary.

5.5 Network Robustness

In addition to the energy saving benefits of the distributed algorithm, it is also advantageous for a camera network to distribute the localization algorithm so that network outages and node insertions and deletions do not have a catastrophic effect on the algorithm.

If a node is inserted, then not only can it be localized immediately, but also its information can immediately contribute to the improved localization of other cameras in the network rather than requiring coordination from some root node. In fact, the *DALT* algorithm does not require a comprehensive routing scheme at all because all communication is done between nearby cameras.

In addition to node insertions, the algorithm is capable of handling node deletions. Sometimes in a sensor network, nodes will become unresponsive due to a dead battery,

electronic malfunction, radio wave interference, or physical theft. In the case that a set of nodes fail or become unresponsive, a given microcluster (a type of network topology discussed in Appendix A) may break into several smaller microclusters. In this case, each microcluster becomes an autonomous unit capable of localizing itself without intervention from the rest of the network. If a set of nodes become available that cause several microclusters to amalgamate into a single microcluster, the iterative *DALT* algorithm will bring their localization estimates into a common coordinate system.

Chapter 6

Results

Here we will discuss the simulations and experiments that were performed to evaluate these techniques. First, we present some simulated results of our algorithm in order to illustrate some of its properties and tradeoffs. Then, we demonstrate our algorithm on high-resolution omnidirectional images with large baseline distances.

6.1 *DALT* Simulation

In order to evaluate *DALT* while taking into account that the solution could be off by a coordinate system and scale, we use Root-Mean-Square (RMS) error between two properly registered sets of points as our error metric. First, both sets of points are translated so their centers of mass are at the origin. Then they are scaled so that their mean squared norm is 1. Finally, they are rotationally aligned using the solution to the Procrustes problem [4]. Δ is then defined as the RMS error between the two resulting registered point clouds, also known as the cRMSd metric [26]. Once a camera network localization is performed, it can be evaluated based on the Δ error between their computed positions and their actual positions.

Rather than just showing results in Δ for various input parameters, we would like to draw comparisons with other techniques. One such technique proposed by Brand et al.

involves an eigenvalue optimization to localize large-scale sparse camera networks [4].

Though this spectral method is not a distributed approach, we have a common assumption in that we have some fixed estimate for orientation that is optimized separately from our main algorithm. Also, both main algorithms use directional information in a least-squares framework to estimate the locations of cameras using an eigenvalue/singular value approach. In this section we shall draw a rough comparison between the two algorithms through simulation.

The simulation was performed as follows. A set of p points are randomly chosen from inside the unit sphere to represent the feature points. A set of n cameras are oriented randomly using the exponential matrix parameterization and then positioned randomly, provided their optical axis intersects the origin. The cameras track normalized image coordinates that are corrupted by white Gaussian noise with variance σ_u^2 . Likewise, σ_ω^2 represents the variance of errors in the orientation vector ω .

The *DALT* algorithm is evaluated by its centralized equivalent as given above in Chapter 4. For the spectral method, a set of epipoles are extracted from pairs of cameras with overlapping views. Then, both the spectral solution and the *DALT* solution are obtained through an eigenvalue optimization. Both the spectral method and *DALT* are each evaluated on their Δ error, Δ_S and Δ_D respectively. Also, the two techniques are compared using dB as $10 \log(\Delta_D/\Delta_S)$. A positive dB represents an advantage of the spectral method, a negative dB represents an advantage of the *DALT* algorithm. Because of the multiple parameters involved that determine the performance, we have shown a few results in Table 6.1

Table 6.1 : *DALT* performance on fully connected cameras

σ_ω	σ_u	n	p	dB	Δ_D
0.010	0.010	20	50	+1.9	5.1e-2
0.010	0.003	20	50	-0.2	1.9e-2
0.003	0.003	20	50	-0.8	7.8e-3
0.001	0.001	20	50	-1.3	2.3e-3
0.003	0.003	20	100	-3.2	7.2e-3
0.003	0.003	30	100	-0.2	6.1e-3
0.003	0.003	10	100	-3.1	7.3e-3
0.003	0.003	10	200	-2.7	7.0e-3

rather graphing them. Each row represents the average result of fifty trials for each algorithm. In most cases, the *DALT* algorithm performs comparably or perhaps slightly better than the spectral method, though its performance drops somewhat for large well connected networks.

For the next simulation, we imposed artificial visibility constraints on the cameras so that each camera overlaps with at most m of its neighbors. For given cameras, $c, c' \in \{1, 2, \dots, n\}$, these two cameras only share common visible points if $|c - c'| \leq m/2$. Each camera views at least a set of six points specific to that camera called its primary points. In addition, each camera also views the primary points belonging to the other cameras it

Table 6.2 : *DALT* simulated performance on sparse camera networks. The simulation is run for n cameras whose view overlaps with m other cameras. The error in feature points and initial orientation is σ_u and σ_ω . The resulting performance measured in the Δ error metric against ground truth is listed in column Δ_D . The performance relative to the spectral method is listed in dB under column dB.

σ_ω	σ_u	n	m	dB	Δ_D
0.003	0.003	20	12	-1.8	1.0e-2
0.001	0.003	20	12	-2.1	8.9e-3
0.003	0.003	20	8	-1.5	1.3e-2
0.003	0.003	20	6	-1.3	1.6e-2
0.001	0.001	20	6	-3.6	3.4e-3
0.010	0.010	20	4	+3.7	2.5e-1
0.003	0.003	10	4	-4.4	1.3e-2
0.003	0.003	50	4	-6.9	1.4e-1

overlaps with.

The results from this sparse network simulation shown in Table 6.2 seem to suggest that the accuracy in orientation and feature point tracking play a much more critical role when there is a low amount of connectivity in the camera network. Also, these results suggest that *DALT* improves upon the spectral method as the network grows and becomes more sparse, and as the error in orientation and feature points drops.

6.2 Omni-Directional Camera Experiment

To evaluate this algorithm on real omni-directional image data, we used images from MIT’s online omni-directional image database. Of these 566 omni-directional nodes, the Green Court dataset consists of 30 nodes over a region of roughly 80m by 115m. Of these 30, there are two microclusters comprised of 26 and 4 nodes. We present results for this set of 26 nodes.

For this set of nodes, we compared against two sources: the initial GPS readings, and the processed position information using the method discussed in [27]. The reported accuracy of these sources is 2.5 meters and 5cm respectively.

We manually tracked roughly 250 points across 26 nodes with an average estimated pixel error of 5 pixels with the exception of roughly 20 mismatched points. These mismatched points were later identified by their high residual error and removed from the dataset. Each node has up to 53 of these points in its visibility set and each pair of linked cameras has at least 10 common points.

Using the available intrinsic calibration and orientation information, we geo-referenced these feature direction vectors into a common coordinate system, then we ran the centralized equivalent of the distributed *DALT* algorithm, immediately giving the asymptotic result.

When evaluating our results using the Δ RMS error metric as shown in Figures 6.1 and 6.2, we note that our 3-D localization differs from the initial GPS dataset by 2.98m

(or 1.87m of error in the 2-D ground plane). All nodes coincided with their GPS measured counterparts to within 7m (or 5m in the 2-D ground plane). For reference, the post-processed data differs from the initial GPS dataset by 4.28m (or 3.63m of error in the 2-D ground plane). Apart from the seven degrees of freedom in the Δ error metric used in the rigid-body registration step (which reduces the apparent RMS error slightly), our result is obtained independently from the GPS estimates, which are reportedly accurate to within 2.5m. Our conclusion is then that the results are at least as accurate as either of the two other sources.

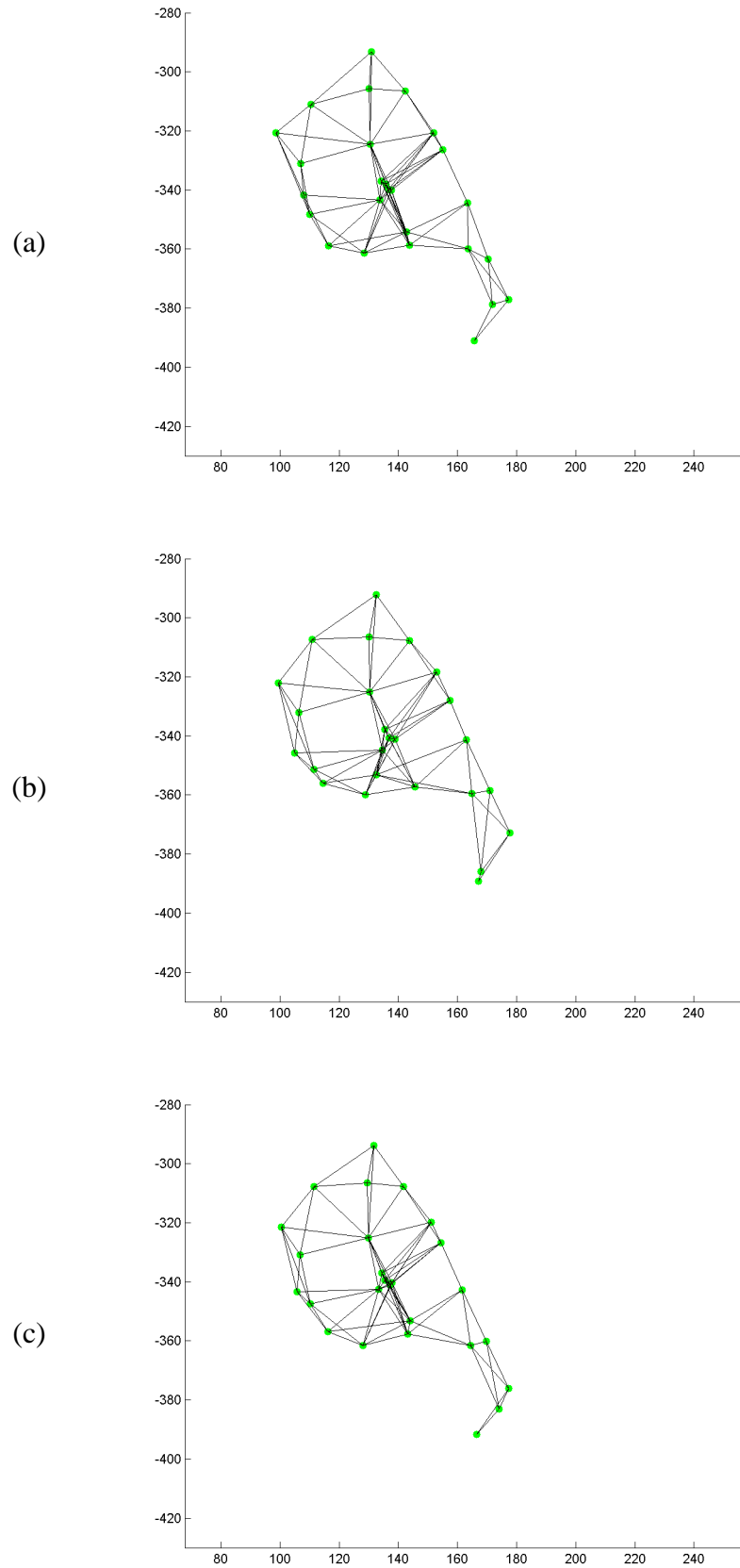
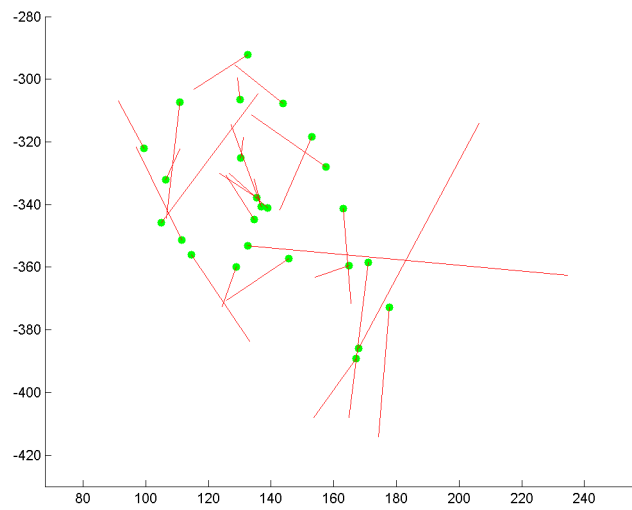
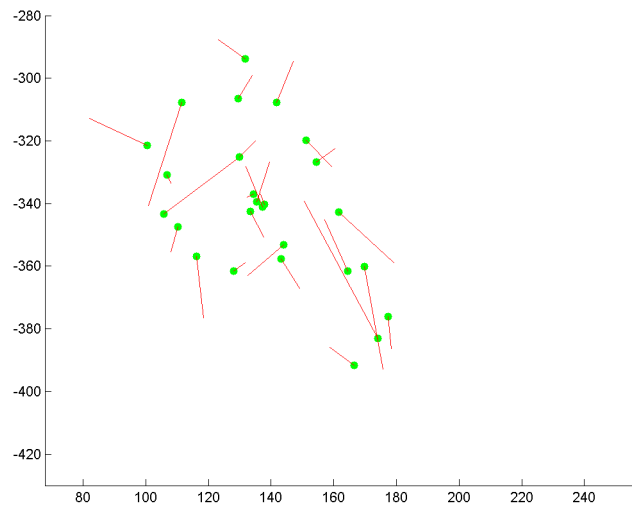


Figure 6.1 : Localization of omni-directional set. Initial and post-processed results from the method in [27] are shown in (a) and (b). The results from the *DALT* algorithm is shown in (c).



(a)



(b)

Figure 6.2 : Localization errors for omni-directional set. Both the post-processed result from [27], and the result from the *DALT* algorithm are shown in (a) and (b) referenced to the initial GPS data. In both plots, the localization estimates were first registered to the GPS data before comparison. The error lines are drawn in the direction of the GPS reading. Their length represents $10\times$ the discrepancy with GPS.

Chapter 7

Conclusions

In this thesis, we have established a framework for effectively distributing a particular type of least-squares minimization useful in camera network localization. We have given a proof of convergence and explored graph theoretic conditions under which this point of convergence is unique. this type of particular type of Alternating Least-Squares [19]

We have also explored some of the practical challenges of a sensor network such as energy minimization, and motivated the need for distributed algorithms to meet this challenge. In the case where no prior pose information was available, we proposed a practical means of obtaining initial localization through trajectory estimation.

DALT not only effectively handles the kinds of sparse camera networks that few localization techniques are suited for, but it also demonstrates comparable performance to the Spectral Method, a centralized sparse camera network localization algorithm [4]. *DALT* has the additional property that it can work either as a centralized eigenvalue post-processing localization optimization or as a distributed algorithm where the computation is split across each processor on the network.

The localization of a camera network enables many exciting applications such as scene representation and geometrical extraction. However, it is not yet clear how well these applications will cope with the challenges of a sensor network. The exchange of a few

images between nodes will perhaps be acceptable, but there will likely be a significant need for distributed compression, especially on video data. Investigation in this area as well as advancement towards an implementation of an actual wireless ad-hoc camera network will likely prove compelling areas for future research and development.

Appendix A

Sufficient Conditions for Localization Uniqueness

Images alone are not capable of giving us absolute metric information, nor do they tell us what coordinate system we are in. For example, a miniaturized upside down version of Mount Rushmore shot at the appropriate places would look the same as the real thing for similar camera poses.

It is then easy to show that the solution to (4.1) is not unique. In particular, given any \mathbf{R}_f , T_f , and X_p satisfying the constraints (2.6a), (2.6b) the substitutions

$$\begin{aligned}\mathbf{R}_f &\leftarrow \mathbf{R}_f \mathbf{R}_*^T \\ T_f &\leftarrow \alpha_*(T_f + \mathbf{R}_f T_*) \\ X_p &\leftarrow \alpha_* \mathbf{R}_*(X_p - T_*)\end{aligned}$$

will also satisfy the constraints. Here T_* , \mathbf{R}_* , and α_* are an arbitrary translation, rotation, and scaling, respectively. Thus, using only image information at each camera, any localization will be given relative to a local coordinate system (defined by T_* and \mathbf{R}_*) and scaling (defined by α_*).

In the particular case of a pair of cameras, this relative localization is known as *weak calibration*. We wish to extend this concept to so-called *sparse camera networks* where the fields of view of the cameras do not all coincide. To this end, we introduce some concepts to describe camera network topologies that can be uniquely localized up to a scale factor.

Definition 7 Two cameras f_i and f_j are linked (denoted $f_i \leftrightarrow f_j$) if they each view the same set of six or more non-coplanar feature points.

Faugeras et al. have shown that if two calibrated cameras share six or more common feature points, then their relative orientation is determined, and their relative position is determined up to a scalar [9].

Definition 8 A sparse camera network is one in which at least one pair of the cameras is not linked.

Definition 9 A linked triangle is a set of three cameras such that all three pairs of cameras within the triangle are linked.

Theorem 10 Consider a linked triangle Δ with the properties that (i) the position and orientation of one of the cameras is known (call it f_1) and (ii) the distance from that camera to one of the other cameras (call it f_2) is known. Then the positions and orientations of all three cameras f_1, f_2, f_3 can be uniquely computed.

Proof: The orientations of f_2 and f_3 can be determined through their relative orientations from f_1 . The position of f_2 can be determined because the distance and direction from f_1 are known. The position of f_3 can be determined through triangulation of intersecting epipolar rays from the known positions f_1 and f_2 . □

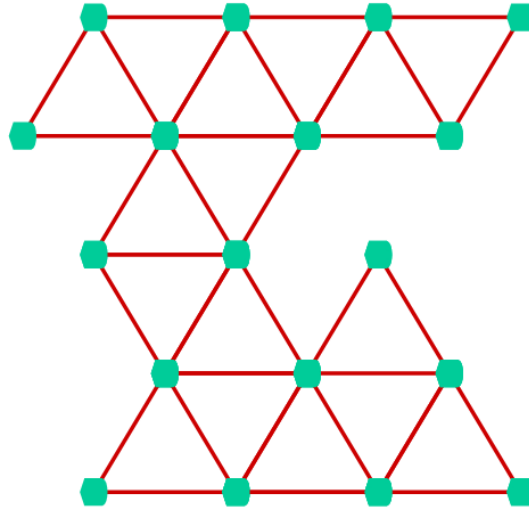


Figure A.1 : Microcluster illustration for linked cameras. Each camera node within this microcluster can potentially determine their relative position and orientation up to a single scale factor.

Definition 11 Two cameras f_i f_j are triple-wise connected if there exist linked triangles $\Delta_1, \Delta_2, \dots, \Delta_n$ such that triangles Δ_r and Δ_{r+1} have two cameras in common, and $f_i \in \Delta_1$ and $f_j \in \Delta_n$.

Definition 12 A microcluster is a set of cameras M with the special property that if $f_i \in M$ then $f_j \in M$ if and only if f_i is triple-wise connected to f_j .

An example of a microcluster is illustrated in Figure A.1.

Theorem 13 All cameras within a given microcluster can be uniquely relatively localized in some coordinate system up to one global scale factor so that the minimum seven degrees of freedom are achieved.

Proof: In the microcluster M consider a linked triangle $\Delta_1 = \{f_1, f_2, f_3\}$. Define the coordinate system to be a scaled version of camera f_1 's coordinate system such that f_1

and f_2 are unit length apart (the fixed scale factor). Then, by Theorem 10, the positions and orientations of all cameras in Δ_1 are determined. For any other camera f_j there exist linked triangles Δ_1 through Δ_n such that Δ_i and Δ_{i+1} have two cameras in common and $f_j \in \Delta_n$. By Theorem 10, if the positions and orientations of the cameras in Δ_i are known, then so are those in Δ_{i+1} . Since the positions and orientations of the cameras of Δ_1 have been determined, it follows through induction that all cameras in $\Delta_1 \cup \dots \cup \Delta_n$ can be localized. In particular, the position and orientation of f_j are determined. \square

Note that each microcluster will be localized in terms of a potentially different local coordinate system.

Bibliography

- [1] G. J. Pottie and W. J. Kaiser, “Wireless integrated network sensors,” *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.
- [2] J. Heidemann N. Bulusu, D. Estrin, “Gps-less low cost outdoor localization for very small devices,” *IEEE Pers. Commun.*, vol. 7, no. 5, pp. 28–34, 2000.
- [3] C. Poelman and T. Kanade, “A paraperspective factorization method for shape and motion recovery,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 3, pp. 206–218, 1997.
- [4] S. Teller M. Brand, M. Antone, “Spectral solution of large-scale extrinsic camera calibration as a graph embedding problem,” in *Europ. Conf. on Comput. Vision (ECCV)*, 2004.
- [5] L. Lee, R. Romano, and G. Stein, “Monitoring activities from multiple video streams: Establishing a common coordinate frame,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 758–767, 2000.
- [6] Y. Abdel-Aziz and H. Karara, “Direct linear transformation into object space coordinates in close-range photogrammetry,” in *Proc. Symp. on Close-Range Photogram.*, 1971, pp. 1–18.
- [7] A. Azarbajehani and A. Pentland, “Camera self-calibration from one point correspon-

- dence,” Tech. Rep., MIT Media Lab, 1995.
- [8] C. Lee, D. B. Cooper, and D. I. Keren, “Computing correspondence based on regions and invariants without feature extraction and segmentation,” in *CVPR*, 1993.
- [9] Q. Luong and O. Faugeras, “Camera calibration, scene motion, and structure recovery from point correspondences and fundamental matrices,” *Int. J. Comput. Vision*, vol. 22, no. 3, pp. 261–289, 1997.
- [10] F. Sebastin Grassia, “Practical parameterization of rotations using the exponential map,” *J. Graph. Tools*, vol. 3, no. 3, pp. 29–48, 1998.
- [11] R. Y. Tsai, “A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses,” *IEEE Trans. Robot. Automat.*, vol. 3, no. 4, pp. 323–344, 1987.
- [12] J. Bouguet, “Camera calibration toolbox for matlab,” http://www.vision.caltech.edu/bouguetj/calib_doc/.
- [13] J. Heikkila and O. Silven, “A four-step camera calibration procedure with implicit image correction,” in *CVPR*, 1997, p. 1106, IEEE Computer Society.
- [14] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 11, pp. 1330–1334, 2000.

- [15] D. Nister, “An efficient solution to the five-point relative pose problem,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 6, pp. 756–777, May 2004.
- [16] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge Univ. Press, 1988.
- [17] S. N Sinha and M. Pollefeys, “Towards calibration a pan-tilt-zoom camera network,” in *OMNIVIS*, 2004.
- [18] G. Kogut, M. Blackburn, and H.R. Everett, “Using video sensor networks to command and control unmanned ground vehicles,” in *AUVSI*, 2003.
- [19] J. De Leeuw, F. W. Young, and Y. Takane, “Additive structure in qualitative data: An alternating least squares method with optimal scaling features,” *Psychometrika*, vol. 41, no. 4, pp. 471–, 1976.
- [20] S. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*, Oxford University Press, 1998.
- [21] G. Golub and C. Van Loan, *Matrix Computations*, John Hopkins University Press, 1996.
- [22] H. Longuet-Higgins, “A computer algorithm for reconstructing a scene from two projections,” *Nature*, vol. 293, pp. 133–135, 1981.

- [23] O. Faugeras and S. Maybank, “Motion from point matches: multiplicity of solutions,” *Int. J. Comput. Vision*, vol. 4, no. 3, pp. 225–246, 1990.
- [24] D. S. Scherber and H. C. Papadopoulos, “Locally constructed algorithms for distributed computations in ad-hoc networks,” in *IPSN. 2004*, pp. 11–19, ACM Press.
- [25] J. Polastre, G. Tolle, and J. Hui, “Low power mesh networking with telos and ieee 802.15.4,” in *SenSys. 2004*, pp. 319–319, ACM Press.
- [26] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-d point sets,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 5, pp. 698–700, 1987.
- [27] S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master, “Calibrated, registered images of an extended urban area,” in *CVPR*, 2001.