MODELING ERROR IN OPTICAL ANGLE MEASUREMENTS FOR POSITION AND ORIENTATION ESTIMATION IN SENSOR NETWORKS

A thesis submitted by

Gabrielle D. Vukasin

In partial fulfillment of the requirements for the degree of Master of Science

In

Mechanical Engineering

Tufts University

August 2016

Advisor: Dr. Jason Rife
Advisor: Dr. Chris Rogers
Cross-Department Committee Member: Dr. Ethan Danahy
Abstract

In this dissertation, I describe two investigations involving the use of angle-of-arrival (AoA) measurements to enable decentralized localization within a robot team. The first investigation developed a new method for relative-orientation estimation by modifying an existing relative-position algorithm called alternating-normals iterative method (ANIM). The second investigation was an experimental characterization of ANIM performance using camera sensors. This was the first time that ANIM has been tested in hardware, as the method has previously been implemented only in simulation. The major outcome of the first investigation is that with orientation estimation, the position error produced by the modified algorithm increased by a factor of two. The major outcome of the second investigation is the justification of an error model for the camera sensor as a Gaussian distribution of noise centered on the original AoA measurement with a standard deviation of 0.4° ± 0.1°.
Acknowledgments

First, I would like to thank the National Science Foundation (grant No. 1100452) and the Massachusetts Space Grant Consortium for supporting this work.

Second, I would like to thank my committee members. I cannot thank my advisors, Professors Jason Rife and Chris Rogers, enough for all of the guidance, opportunities, and remarkable teaching they have provided me over the last two years. I would like to extend my gratitude to Professor Rife for his tireless help, constant positive attitude, and time he spent supporting my work. He has helped shape the scientist that I am today through the amount of time and effort he put forth critiquing my presentations, papers, and ideas. I would like to thank my other advisor Professor Chris Rogers for his extensive skills in LabVIEW, his eagerness to fit me into the tightest-packed schedule I have ever seen, and support to chase my passions. The opportunities he provided for me to learn robotics, BotSpeak for EV3, the cymbal robot, and the soft robotic gripper, to name a few, were crucial for my education and general happiness at Tufts. I would like to thank Professor Ethan Danahy for serving on my committee and for his hardware advice. I would also like to thank him for his advice on how to broaden my computer science knowledge and for his general cheerful attitude around the Center for Engineering Education and Outreach (CEEO). Additionally, I would like to extend my gratitude to Professor Pratap Misra for his critiques of my work, his teaching, and his enthusiasm for GPS.
I would like to thank Tufts University and the Mechanical Engineering Department in particular for the awesome facilities, opportunities, and great professors. Without these I would not have learned and succeeded as much as I have during the last two years.

I would also like to thank my fellow mechanical engineering graduate students for collaborating on problem sets and projects, supporting each other through late night study sessions, critiquing my dissertation work, and having fun outside of school. In particular, I would like to thank Scott Parker for going out of his way to give me advice, papers, critiques on my work, etc. To the rest of the ASAR and CRISP lab groups, I would also like to extend my gratitude for listening to my presentations and critiquing my work. Additionally, I would like to thank the members of the CEEO for support and aid in matters ranging from coding in unfamiliar languages to getting visas to China.

Last but not least, I would like to thank my family for all of their love and support, especially when I needed an extra boost of confidence during frantic late night calls.
# List of Tables

3.1 Node Positions Relative to Node 1 ........................................ 28
3.2 Node Orientations Relative to Node 1 .................................... 29
3.3 Output Error Metrics for 1% Measurement Noise ................. 31
3.4 Output Error Metrics for 5% Measurement Noise ................ 32
4.1 Camera Specifications .......................................................... 44
4.2 Single Camera Trials ............................................................ 53
4.3 Single Camera Trials Standard Deviations ......................... 54
4.4 Sources of Error ................................................................. 60
4.5 Node Information ............................................................... 66
4.6 Experimental Results ......................................................... 68
4.7 Corrected Experimental Results .......................................... 77
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example of Three Aircraft in a Common Coordinate System</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Three Nodes Define a Plane</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Rotational ANIM Processing by Each Node</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Three Nodes with Unknown Orientations</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>Geometry of Nodes After Orientation Algorithm</td>
<td>25</td>
</tr>
<tr>
<td>3.4</td>
<td>Results for 20 Monte Carlo Trials for (a) 1% and (b) 5% Error</td>
<td>30</td>
</tr>
<tr>
<td>3.5</td>
<td>Convergence for Sample Set of Data</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Input and Output Error Relationship in ANIM Algorithm</td>
<td>36</td>
</tr>
<tr>
<td>4.2</td>
<td>Image of a Node</td>
<td>38</td>
</tr>
<tr>
<td>4.3</td>
<td>Calculating Unit Pointing Vector from Centroid Position</td>
<td>40</td>
</tr>
<tr>
<td>4.4</td>
<td>Pinhole Camera Two-Dimensional Image Relation</td>
<td>41</td>
</tr>
<tr>
<td>4.5</td>
<td>Focal Length Determination Setup</td>
<td>43</td>
</tr>
<tr>
<td>4.6</td>
<td>Camera Calibrations</td>
<td>45</td>
</tr>
<tr>
<td>4.7</td>
<td>Image Processing Steps</td>
<td>46</td>
</tr>
<tr>
<td>4.8</td>
<td>Different LED Patterns</td>
<td>47</td>
</tr>
<tr>
<td>4.9</td>
<td>Effect of Image Processing</td>
<td>48</td>
</tr>
<tr>
<td>4.10</td>
<td>Precision Measurement Experimental Setup</td>
<td>51</td>
</tr>
<tr>
<td>Figure Number</td>
<td>Figure Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.11</td>
<td>Image Used to Align Camera</td>
<td>52</td>
</tr>
<tr>
<td>4.12</td>
<td>Tilt and Pan Directions of a Node</td>
<td>56</td>
</tr>
<tr>
<td>4.13</td>
<td>Tilt Error</td>
<td>57</td>
</tr>
<tr>
<td>4.14</td>
<td>Angular Distance Between LED and Camera Centroid</td>
<td>59</td>
</tr>
<tr>
<td>4.15</td>
<td>The Inverse Method to Determine Experimental Sensor Error</td>
<td>61</td>
</tr>
<tr>
<td>4.16</td>
<td>Measurement Pairs in a Three Node System</td>
<td>64</td>
</tr>
<tr>
<td>4.17</td>
<td>Schematic of Three-Node Experiment</td>
<td>65</td>
</tr>
<tr>
<td>4.18</td>
<td>Experimental Position Estimates</td>
<td>67</td>
</tr>
<tr>
<td>4.19</td>
<td>Zoom of Experimental Position Estimates</td>
<td>68</td>
</tr>
<tr>
<td>4.20</td>
<td>Simulation Position Estimates for Case with Sensor Measurement $\sigma = 0.5^\circ$</td>
<td>70</td>
</tr>
<tr>
<td>4.21</td>
<td>Simulation Position Estimate Distributions</td>
<td>71</td>
</tr>
<tr>
<td>4.22</td>
<td>Simulation Mean Error</td>
<td>72</td>
</tr>
<tr>
<td>4.23</td>
<td>Simulation DRMS Error</td>
<td>73</td>
</tr>
<tr>
<td>4.24</td>
<td>Simulation Normalized DRMS Error</td>
<td>74</td>
</tr>
<tr>
<td>4.25</td>
<td>Corrected Experimental Position Estimates of Node 3</td>
<td>78</td>
</tr>
<tr>
<td>4.26</td>
<td>Corrected Normalized DRMS Error</td>
<td>79</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Introduction

Two studies are presented in this dissertation. The first study extends an established decentralized localization algorithm that estimates not only relative position but also relative orientation of a group of nodes. The second is a method to estimate an input sensor error model to represent optical (camera) angle-of-arrival (AoA) measurements for this algorithm, based on experimental data.

1.2 Background on Orientation Sensing and Estimation

In order to understand why a new distributed orientation algorithm is needed, it is first useful to provide an overview of existing related technologies. Many methods of orientation estimation exist. The most obvious solution to a single robot with an unknown orientation is to use sensors such as gravimeters, magnetometers, gyroscopes, accelerometers, etc. Gravimeters measure
the direction of the local gravity field of the Earth to estimate orientation. A robust solution that could work on Earth or in space is desired, thus my sensor network localization and orientation estimation method is of interest to the distributed estimation community. Similarly, magnetometers measure the local magnetic field of the Earth and can be used to estimate attitude. However, they do not work in space or near large magnetic fields. Gyroscopes and accelerometers can be used to estimate orientation by integrating angular velocity. Current work improving attitude estimation includes Cradssidis et al. who have developed an algorithm to estimate attitude using the rate integration of three-axis gyroscopes [1]. However, gyroscopes drift over time and produce bad estimates of orientation over large timescales. Accelerometers are sensitive to small vibrations, which can increase error in orientation estimation if the time-scale of readings is too short.

Methods combining gravimeters, magnetometers, accelerometers, and gyroscopes increase the possible applications in which orientation can be estimated. Hardware combinations result in attitude and heading reference systems (AHRS) and inertial measurement units (IMU). This hardware uses filtering to estimate orientation. For example, He et al. combined gravimeter and accelerometer information in an extended Kalman filter to estimate orientation [2]. Attitude can also be estimated by three orthogonal global positioning system (GPS) antennae. GPS antennae and inertial sensors have been combined through different types of filters to produced good orientation estimations over time [3]. The prominent methods using inertials and/or GPS to estimate attitude are summarized in [4, 5, 6]. However, all of these methods fail when any one of the sensors fail. For example, when the Earth’s gravity
or magnetic field are not present, over time when integration produces large errors, or in GPS denied environments.

Without inertial sensors and GPS, a robot receives no information from Earth’s gravity or magnetic field and thus has an unknown orientation in space. This is commonly referred to as the “lost in space” problem, “… in which no information regarding the attitude of the spacecraft is available” and orientation of the spacecraft is desired [7]. It was solved with cameras by converting an image of stars to vector measurements of those stars and fitting these vector measurements to a known map of the stars [7, 8]. Improvements to these solutions are continuing to be developed. For example, Delabie et al. developed a quality check to remove unreliable star measurements that improved attitude estimation [9]. However, a known map of objects (in this case stars) is required to estimate orientation with these methods, so this can only be done in space or in a location with a known three-dimensional map.

Three-dimensional orientation estimation algorithms that can leverage existing sensing and communication systems are desirable. It is also desirable that the estimation method be decentralized, so that its performance is robust to the loss of any individual nodes. Prior work in distributed orientation estimation using AoA measurements has been focused on two-dimensional orientation estimation. For example, one distributed localization and orientation estimation method uses a probabilistic approach, but includes multiple beacons in its group of nodes [10]. Niculescu and Nath developed an algorithm that hops from node to node estimating orientation and position using AoA measurements [11]. They too require a number of ground beacons, which are nodes with known location. Both of these methods are limited in that they
are intended to address a two-dimensional localization problem. Decentralized estimation of three-dimensional relative orientation and position remains an open problem, one for which novel solutions are required.

1.3 Background on Sensor Error Modeling for ANIM

The particular approach to collaborative navigation explored here is based on a distributed estimation algorithm dubbed the Alternating-Normals Iterative Method (ANIM), which invokes geometric constraints to determine relative positions of collaborating vehicles from noisy AoA measurements. The ANIM algorithm was first presented in [12]. Thus far, ANIM has only been assessed in simulation. Experiments are needed to determine for which applications ANIM would be most beneficial. For example, one application of ANIM could be a backup navigation system that is needed for a group of rotorcraft to fly in formation over crowds to provide cell service to the increased number of cellphones in the area [13]. Cell towers are only equipped to handle a bit above their normal load of users. Therefore, crowds from concerts, protests, disasters, etc. pose an issue for users because of the limited bandwidth of the cell towers. Temporary cell towers transported by vans may provide service at the edges of a crowd. However, for large crowds or in cities where it is tough to maneuver vehicles, this is still not enough to provide ample cell service to all users in the crowd. Recently, the Federal Aviation Administration (FAA) has come out with new guidelines on flying over crowds [14]. Before these new regulations, the FAA did not allow rotorcraft to fly over crowds. Additionally, the technology for the small cell broadcasters to increase the bandwidth of cell
service has already been developed [15]. These small cell broadcasters can be mounted to the rotorcraft to provide cell service wherever they fly. Because the sky over a crowd is open, it is an ideal place for the cell beacons. Because the small cell broadcasters are mounted on rotorcraft, the constellation may be altered in real-time to suit the changing needs of the crowd. The only problem is that the FAA requires the rotorcraft to have a less than 1% chance of injuring someone [14]. Thus, each system that a rotorcraft relies on to fly in the intended manner must have a backup. These rotorcraft nominally rely on GPS for navigation, thus, a backup navigation system is needed when GPS fails. Would ANIM and AoA hardware be suitable as a backup navigation system for this group of rotorcraft? Because experiments using ANIM have not been performed previously, this question cannot be answered. This is the reason this experimental work is important: to prove for what applications ANIM may be used.

In simulation, the input of ANIM simulations are AoA measurements with an arbitrary Gaussian error distribution. The output is a constellation of position vectors that describe the relative positions of the nodes in the system. Position errors can be determined from the ground truth. The more erroneous the input AoA measurements, the more erroneous the position vectors will be. Thus, ANIM can be thought of as a function of input sensor error with the result of output position error. The relationship between the input sensor error and the output position error must be scrutinized to understand the usefulness of the ANIM algorithm. Understanding how the inputs affect the outputs will inform for what practical applications ANIM may be used. Simulations and experiments must test the algorithm to determine its useful-
ness. The simulations need reasonable sensor error inputs in order to assess the outputs with meaning. These sensor error inputs may be determined from past work or from experimentation. The types of sensors that can be used to take AoA measurements include antennae, microphones, diffraction gratings, photosensors, cameras, etc. For instance, two antennae can be used to take AoA measurements to any source of electromagnetic radiation using the time difference of arrival of the crest of a radiation wave between the two antennae. An array of antennae can be used by comparing either received signal strength or phase of electromagnetic radiation between elements. Similarly, an array of microphones can be used to take AoA measurements of acoustic waves by comparing the voltage responses of the microphones [16]. Photosensors such as photodiodes or cameras can be used to find the AoA to light sources. In this work, I am interested in optical AoA sensors (cameras) because they are a cost effective way to take experimental AoA measurements. Optical AoA sensors could be used in the application of rotorcraft providing cell service to crowds, so cameras are a useful type of AoA receiver.

In similar distributed localization and orientation estimation methods that use AoA measurements, errors are described by Gaussian noise models [10, 11, 17]. However, one noise model cannot describe all types of received AoA measurements because their errors depend on the hardware used in the AoA transmitters and receivers as well as the environments in which the measurements are taken. For example, Pederson et al. determined that a Gaussian with a standard deviation of 6° described the probability distribution function of the azimuth dispersion (AoA of radio signal) of an array of radio antennae [18]. Whereas a directional radio antenna rotating in 360° is modeled as hav-
ing a Laplacian probability distribution function for its angular measure error model with a standard deviation of 25.5° [19]. These distributions are very different, and thus, error models must be measured for each type of sensor used in collecting AoA measurements.

1.4 Contributions

This section describes the two primary contributions of this dissertation.

Contribution 1: Adapted a distributed position-estimation algorithm (ANIM) to also perform orientation estimation.

The original ANIM algorithm relies on each node having a known attitude. Direct attitude sensing may be difficult in outer space (where gravity is weak) or in urban areas (where the Earth’s magnetic field is distorted). A two-step optimization method was introduced to estimate relative orientation using only AoA measurements obtained from signals broadcast among team-members to enable collaboration within teams of robots or vehicles under these harsh conditions.

Contribution 2: Developed a tool to predict ANIM error levels for arbitrary team geometries by using inverse methods to infer an approximate sensor error model where ground truth is not accurate.

I developed a method to predict ANIM error levels for arbitrary team geometries by using inverse methods to infer an approximate sensor error model. Such methods are paramount to predict the performance of the ANIM estimation algorithm, since position error performance depends strongly on the
geometric formation of the collaborators. Predicting the magnitude of estimation input errors is difficult because the input errors associated with sensing (camera AoA measurements) cannot be determined since the ground truth of the AoA measurements is inaccurate or nonexistent. After attempting to model these systematic effects where ground truth is not accurate, this work transitions to the hypothesis that systematic errors more closely resemble random errors when analyzed in aggregate, as in the output position error for ANIM. As such, this work develops an inverse solution, using experiments to characterize the position-errors obtained from ANIM and indirectly inferring an approximate input error distribution by tuning a simulation to match experimental results. Finally, this work uses the inverse method to estimate a Gaussian input error distribution, where one sigma equals 0.4°, for camera AoA measurements.

1.5 Overview of Dissertation

This dissertation begins with a more in-depth motivation for the previously established ANIM algorithm. I will then describe the ANIM algorithm. In the next chapter, the orientation estimation addition to the original localization algorithm, Rotational ANIM, is introduced and its performance is assessed through simulation. The last chapter of this work describes an experimental method to estimate the error in optical AoA measurements using an inverse method. Inverting the relationship between the input sensor error and output position error of the ANIM algorithm, this method compares the position estimates from ANIM simulations to those from experiments to esti-
mate the sensor error of the experiments. I will then conclude this work with remarks and future work.
Chapter 2

Original ANIM Method

2.1 ANIM Motivation

The Alternating-Normals Iterative Method (ANIM) is a decentralized algorithm for members of a multi-vehicle team to estimate their relative position (and now orientation [20]) based on angle-of-arrival (AoA) measurements, which are assumed to be acquired from communication signals transmitted within the team.

One potential application of this capability is in civil aviation, as a backup for global positioning system (GPS). GPS is increasingly critical to the operation of the air transport system. GPS is already used for en route navigation and non-precision approach [21], and the FAA has mandated that nearly all conventional aircraft equip by 2020 with Automatic Dependent Surveillance - Broadcast (ADS-B), a technology that will communicate an aircrafts GPS position to other aircraft in the vicinity, effectively providing radar-like

---

This section is a reproduction of the “Introduction” section of [20] with modification.
surveillance to the flight deck [22]. Developments in the Ground Based Augmentation System (GBAS) are poised to enable automated aircraft landing with GPS in the near future [23]. In the longer term, it is envisioned that the next-generation air traffic control system will implement 4D trajectories (position and time) that take an aircraft from the departure gate all the way to the arrival gate using GPS [24, 25, 26]. Taken together these GPS-based capabilities lay the groundwork for a massive increase in the capacity of the national airspace without sacrificing safety. Despite its potential, this vision suffers a critical limitation: vulnerability to the loss of GPS [27]. Recent observations of unintentional GPS jamming at Newark International by so-called personal privacy devices underscore this risk [28].

Several alternatives to GPS are being considered in aviation, with a particular emphasis on enhancement of legacy systems. Despite being decommissioned in the US in 2010, Loran is emerging as an important potential backup navigation system [29, 30]. Researchers have also considered enhancing VOR and DME systems to increase positioning accuracy for airborne users [31, 32]. Another recently proposed alternative would invert ADS-B, using the ADS-B communication signal for multilateration during a GPS failure [33]. This section builds on the last concept, by generalizing the notion that an existing communication capability might be leveraged as a navigation alternative during a GPS outage.

I envision a concept of operations in which aircraft would acquire mutual bearing measurements by inferring AoA from peer-to-peer communication signals. Distributed estimation would be used to define aircraft position and orientation relative to a common coordinate system. The set of three air-
craft in Figure 2.1 is an example. In the diagram, the red arrows indicate the heading of aircraft and the green arrows represent bearing vectors (which are an equivalent representation of AoA data). By measuring bearing angles, broadcasting them, and processing the collection of received measurements, it is possible to define a common coordinate system that describes the relative positions of all vehicles (subject to an unknown scale factor) even when the headings of the vehicles are arbitrary or unknown. If desired, sparsely located ground beacons might be introduced as anchors, allowing the common relative coordinate system to be related to Earth-fixed coordinates.

The result would be a combined communication, navigation, and surveillance (CNS) system, through which each aircraft would estimate its own location, estimate the location of nearby aircraft, and deliver communication data at the same time [34]. The signals used to acquire AoA measurements might be ADS-B messages or other networked communication signals. Using communication signals to obtain AoA measurements would enhance available communication bandwidth by avoiding the need to dedicate additional spectrum to navigation.
The proposed concept of operations has particular advantages for an emerging world in which unmanned aerial systems (UAS) have become commonplace. Systems involving multiple UAS have a wide range of applications including surveillance, imaging, package delivery, and providing cell service to large crowds, to name a few [35]. One issue with UAS applications is that autonomous aircraft may fly at low altitudes, where ground-based navigation beacons (e.g. VOR/DME) provide poor coverage [36, 37]. A collaborative navigation capability could overcome this deficit, by providing high accuracy
positioning wherever aircraft (manned or unmanned) operate in density.

2.2 Description of ANIM

This dissertation describes the development of a decentralized position estimation algorithm that computes relative position and orientation of vehicles by combining noisy AoA measurements acquired and broadcast in a network of collaborating vehicles. I envision one application of this method to be used as backup to GPS. The novelty of this algorithm is that it allows aircraft to estimate their position and orientation relative to each other without requiring the vehicles to sense their own orientation relative to the Earth.

This section describes the original ANIM algorithm [12], which processes AoA measurements to estimate node position. The term node refers to the transceiver unit on each vehicle. It is assumed that this transceiver unit broadcasts a unique identifier signal and obtains AoA measurements from similar such signals broadcast by collaborating vehicles.

ANIM estimates the relative positions of nodes in a sensor network from initially noisy AoA measurements. ANIM is a decentralized strategy for AoA-based relative positioning for a network of $n$ nodes. It is a decentralized strategy in the sense that no single collaborator needs access to all measurements. Each node conducts processing opportunistically, using its own AoA measurements and any broadcast AoA measurements it happens to receive.

The original ANIM algorithm obtains positions from AoA measurements.

---

This section is a reproduction of the “Overview of Original ANIM Algorithm” section of [20] with modification.
converted to unit pointing vectors in a global coordinate system, for example North-East-Down (NED), under the assumption that the aircraft-based coordinates could be related to world-fixed coordinates using an AHRS. In this dissertation, I will generalize the original ANIM algorithm so that it functions even if AHRS data are not available. This is the key difference between the method proposed in this dissertation (see next chapter) and the original method [12].

To provide context for the generalized ANIM algorithm, the remainder of this section will review the details of the original algorithm. Assuming AoA measurements can be rotated to a global coordinate system, they can be expressed as pointing vectors in that coordinate system.

The simplest case in which ANIM can be used is in a network of three nodes, as represented in Figure 2.2. Each side of the figure shows three nodes labeled 1, 2, and 3. On the left in Figure 2.2a, the relative position vectors between each pair are labeled $\mathbf{p}_{12}$, $\mathbf{p}_{23}$, and $\mathbf{p}_{31}$. Here the trailing subscripts refer to the measuring (receiving) node first and the detected (broadcasting) node second. The constraint that all three vectors between nodes should be coplanar can be used to reduce measurement noise. Moreover, the triangle geometry can be reconstructed (to a scale factor) using only angle measurements. If the scale factor can be determined by complementary information (as will be discussed later), then this triangle provides relative positioning information.

As the number of nodes is increased, noise mitigation is enhanced. The sensor noise can be characterized in terms of a positioning error $\epsilon_{ij}$. The unit pointing vector estimates $\hat{\mathbf{u}}_{ij}$ can be written in terms of $\epsilon_{ij}$ and the true position vector $\mathbf{p}_{ij}$. 
The hat is used to indicate a noisy estimate.

As mentioned above, ANIM provides a processing benefit in the presence of noise, when the $\hat{u}_{ij}$ are not coplanar. ANIM effectively filters the measurements to ensure geometric consistency. The filtering is performed in two steps. First, a vector $\hat{n}_{ijk}$ is computed as an estimate of the normal vector to each plane containing three nodes. Second, the normal vectors are used to improve the estimated unit pointing vectors $\hat{u}_{ij}$. Both steps are performed by each node, incorporating all edge data available to it (both from direct measurements and broadcast data).

To implement the first step (estimating the normal vector to each set of three nodes), a least-squares method is used. To do this, all the measurements between nodes in a triplet are compiled as row vectors of a matrix $A_{ijk}$. Typically, for the triplet $i, j, k$, this matrix could include as many as six
measurements: \( \hat{u}_{ij} \), \( \hat{u}_{ji} \), \( \hat{u}_{jk} \), \( \hat{u}_{kj} \), \( \hat{u}_{ki} \), and \( \hat{u}_{ik} \). For the case in which all six measurements are available, \( A_{ijk} \) would be:

\[
A_{ijk} = [\hat{u}_{ij} \; \hat{u}_{ji} \; \hat{u}_{jk} \; \hat{u}_{kj} \; \hat{u}_{ki} \; \hat{u}_{ik}]^T.
\] (2.2)

I would like to find the estimate of \( \hat{n}_{ijk} \) that is the most orthogonal to all of the unit pointing vectors in the plane. Another way to say this is that I want the \( \hat{n}_{ijk} \) that minimizes its dot product with each of the unit pointing vectors, because the following is ideally true

\[
u_{ij} \cdot n_{ijk} = 0.
\] (2.3)

Considering all of the measurements around a triangle, and minimizing the residual errors of (2.3) applied to all edges around a triangle, I can obtain an estimate of the normal vector as follows

\[
\hat{n}_{ijk} = \arg\min_n(n^T A_{ijk} A_{ijk}^T n).
\] (2.4)

To solve for \( \hat{n}_{ijk} \) in (2.4), I use least-squares minimization, implemented efficiently using a singular value decomposition (SVD) \([38, 39]\).

Now that I have found \( \hat{n}_{ijk} \) for each triangle observed by a node, the second step of ANIM is to update each unit pointing vector estimate. To do this, a matrix \( B \) is assembled of all of the normal vectors from triangles containing a particular pointing vector. Consider the update to the pointing vector from node \( i \) to node \( j \). For node \( i \) to update \( \hat{u}_{ij} \), it considers all triangles
containing nodes $i$, $j$ and any third node $k$. The nodes $k \in \{k_1, k_2, k_3, ..., k_S\}$ include all $S$ nodes observed by node $i$ (other than node $j$). Assembling these normal vectors as rows of a matrix gives

$$\mathbf{B}_{ij} = [\hat{n}_{ijk_1} \hat{n}_{ijk_2} \hat{n}_{ijk_3} ... \hat{n}_{ijk_S}]^T. \quad (2.5)$$

Based on the identity (2.3), I want to find the $\hat{\mathbf{u}}_{ij}$ that minimizes its dot product with each of the normal vectors in $\mathbf{B}_{ij}$ (2.6).

$$\hat{\mathbf{u}}_{ij} = \arg\min_{\mathbf{u}} (\mathbf{u}^T \mathbf{B}_{ij} \mathbf{B}_{ij}^T \mathbf{u}) \quad (2.6)$$

Mirroring the first step, I use SVD in the same way as before to get the optimal value of $\hat{\mathbf{u}}_{ij}$. This step is repeated for all of the unit pointing vectors initially known by node $i$. After this update step, the new values of the estimated $\hat{\mathbf{u}}_{ij}$ are rebroadcast, so that they are shared with collaborators. All collaborators can then solve (2.4) and (2.6) again using the rebroadcast $\hat{\mathbf{u}}_{ij}$. The process repeats to convergence. A proof of convergence is provided in [40].

After iterating to convergence, all nodes share a common set of pointing vectors, with inconsistencies removed. In other words, all converged pointing vectors are constrained so that any triangle connecting three nodes has coplanar edges (the $p_{ij}$'s in Figure 2.2a). The next step is to estimate distances between nodes. This step can be accomplished by solving simultaneously, at each node, the summation of vectors around all complete triangles. This is to say that
\( \mathbf{p}_{ij} + \mathbf{p}_{jk} + \mathbf{p}_{ki} = \mathbf{0}. \) (2.7)

Therefore, if the distance between any node pair is labeled \( d_{ij} \), I can write

\[
d_{ij} \mathbf{\hat{u}}_{ij} + d_{jk} \mathbf{\hat{u}}_{jk} + d_{ki} \mathbf{\hat{u}}_{ki} = \mathbf{0}. \quad (2.8)
\]

for any triplet of coplanar pointing vectors.

Solving (2.8) simultaneously for all triangles seen by a node gives the unknown distances subject to a scale factor \( C \) since the set of equations (2.8) for all triangles is homogeneous. To reflect this scale-factor, which applies to the entire network of triangles, it is helpful to rewrite (2.8) in terms of the normalized distances \( d_{ij}' = \frac{d_{ij}}{C} \).

\[
Cd_{ij}' \mathbf{\hat{u}}_{ij} + Cd_{jk}' \mathbf{\hat{u}}_{jk} + Cd_{ki}' \mathbf{\hat{u}}_{ki} = \mathbf{0} \quad (2.9)
\]

In theory, the scale factor \( C \) can be determined if at least one distance measurement is available. Alternatively, if the positions of two nodes are known (e.g. if the network is broad enough to span two or more ground stations at surveyed locations), \( C \) can be determined. The final result is that each node has an estimate of the relative position of the rest of the nodes using AoA data for a single time step.
Chapter 3

Rotational ANIM

In this section I introduce Rotational ANIM, a modification of the original ANIM algorithm that enables nodes to estimate their orientation relative to each other, even when global orientation is unknown. The flow chart in Figure 3.1 describes the steps required to evaluate Rotational ANIM. As shown in the figure, the first step after each node acquires AoA measurements is to broadcast them to other nodes. Each node then individually estimates the relative orientation of the other nodes. Each node can only estimate the relative orientation of other nodes of which it has taken a measurement and from which it has received broadcast measurements. After relative orientation is determined, the rest of ANIM can continue as described in the previous section.

My process for estimating relative orientation involves the rotation matrices between the rigid frames associated with each pair of nodes. I estimate

This chapter is a reproduction of the “Estimating Orientation with Rotational ANIM” section of [20] with modification.
the rotation matrices by first aligning two pointing vectors that we know should be collinear, for example $\hat{u}_{ij}$ and $\hat{u}_{ji}$. Subsequently, I rigidly rotate the set of pointing vectors at one node (node $j$) about the axis defined by the collinear pointing vectors to align the remaining measurements.

In order to understand the process that aligns the remaining vectors at each node, it is useful to consider a simple example involving the orientation
between two nodes (nodes $i$ and $j$) that each observe a common third node $k$. This scenario is represented in Figure 3.2. The red arrows represent AoA measurements taken by nodes $i$ and $j$. In graphical terms, the vector $\hat{u}_{ik}$ should be rotated about the axis between $i$ and $j$ until it falls into the same plane as $\hat{u}_{jk}$.

The following is a quantitative description of the steps executed by node $i$ to estimate the relative orientation between nodes $i$ and $j$. Use the labels $A$ and $B$ to indicate the initial coordinate systems for the measurements obtained from nodes $i$ and $j$, respectively. Initial coordinate systems are representative of body-fixed coordinates for each node. This means that the AoA measurements, or unit pointing vectors, $\hat{u}_{ij}$ and $\hat{u}_{ik}$ taken by node $i$ are are initially expressed in coordinate frame $A$ and the unit pointing vectors, $\hat{u}_{ji}$ and $\hat{u}_{jk}$, taken by node $j$ are are initially expressed in coordinate frame $B$.

The first step in estimating orientation is for node $i$ to rotate $\hat{u}_{ji}$ so that it is collinear and opposite in direction to $\hat{u}_{ij}$. The idea is to define intermediate bases $A'$ and $B'$ where the third vector in the basis is in the $\hat{u}_{ij}$ or $\hat{u}_{ji}$ direction, respectively. To do this, it is sufficient to construct the rotation matrix
\[ A^r A' = [r_{a1} \ r_{a2} \ \hat{u}_{ij}]^T \]  

(3.1)

and the matrix

\[ B^r B = [r_{b1} \ r_{b2} \ \hat{u}_{ji}]^T. \]  

(3.2)

Here the vectors \( r_{a1} \) and \( r_{a2} \) are arbitrary vectors normal to each other and \( \hat{u}_{ij} \), sequenced to obey the right-hand rule. Similarly, \( r_{b1} \) and \( r_{b2} \) are arbitrary vectors normal to each other and \( \hat{u}_{ji} \), ordered again to obey the right hand rule. Since the third axes are assumed equivalent for each of these bases, all that remains is to find the simple rotation angle about this common axis that fully aligns the bases.

In order to find the simple rotation angle, the second step of my rotation algorithm uses planes containing common measurements. Where nodes \( i \) and \( j \) see many other nodes \( k \) in common, it will not be possible to make \( \hat{u}_{ik} \) and \( \hat{u}_{jk} \) coplanar for all \( k \), because of measurement noise. For this reason, I frame the process of finding the simple rotation angle as an optimization problem. The optimization problem is described by the following cost function \( J_{ij} \).

\[ J_{ij} = -\sum_{k \in K} (A')^T \hat{u}_{ik} \cdot A^r R(\theta)^B \cdot (B')^T \hat{u}_{jk} \]  

(3.3)

where \( K \) is the set of all nodes observed by both nodes \( i \) and \( j \). Each vector is expressed in a coordinate system indicated by a leading superscript. The rotation matrix \( A^r R^B \) relates the two coordinate systems \( A' \) and \( B' \). The
simple rotation about the $ij$ axis is labeled $\theta$.

The simple rotation angle $\theta$ that minimizes the above cost function is a good estimate of the true rotation. To understand this, consider each term of the summation. Each term corresponds to one triangle, as illustrated in Figure 3.2. For any term of (3.3), the two measurements to $k$ can be split into components parallel and perpendicular to the common axis between nodes $i$ and $j$.

\[
(A')\hat{u}_{ik} = (A')\hat{u}_{ik,\parallel} + (A')\hat{u}_{ik,\perp}
\]  \hspace{1cm} (3.4)

\[
(B')\hat{u}_{jk} = (B')\hat{u}_{jk,\parallel} + (B')\hat{u}_{jk,\perp}
\]  \hspace{1cm} (3.5)

In the above equations, the subscript $\parallel$ implies the parallel component of the vector (i.e. the third basis vector in $A'$ or $B'$) and the subscript $\perp$ implies the perpendicular component of the vector (i.e. the first two basis vectors in $A'$ or $B'$). Figure 3.3 depicts the components of $(B')\hat{u}_{jk}$ and $(A')\hat{u}_{ik}$ that are parallel (blue) and perpendicular (red) to the common axis between nodes $i$ and $j$. 
The meaning of the cost function (3.3) becomes more clear after substituting equations (3.4) and (3.5). After substitution I obtain

\[
J_{ij} = - \sum_{k \in K} \left[ (A^') \hat{u}_{ik,\|} \cdot (A^B) \hat{u}_{ik,\perp} \right] \cdot \\
A' R(\theta)^{B'} \cdot (A^') \hat{u}_{jk,\|} + (A^B) \hat{u}_{jk,\perp} \right].
\] (3.6)

where \( A' R^{B'}(\theta) \) can be expanded as

\[
A' R(\theta)^{B'} = \begin{bmatrix}
\cos \theta & \sin \theta & 0 \\
-\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}.
\] (3.7)

After simplifying \( J_{ij} \), the result is

\[
J_{ij} = - \sum_{k \in K} \left[ ((A^') \hat{u}_{ik,\|} \cdot (B^B) \hat{u}_{jk,\|}) + \\
((A') \hat{u}_{ik,\perp} \cdot A' R(\theta)^{B'} \cdot (B^B) \hat{u}_{jk,\perp}).
\] (3.8)
Finally, I can simplify $J_{ij}$ even further to

$$J_{ij} = \text{constant} + f(\theta).$$  \hfill (3.9)

I define $f(\theta)$ as

$$f(\theta) = -\sum_{k \in \mathcal{K}} \left( ||(A')\hat{u}_{ik,\perp}|| \cdot ||(B')\hat{u}_{jk,\perp}|| \cdot \cos(\phi_k - \theta) \right)$$ \hfill (3.10)

where $\phi_k$ is the angle between $(A')\hat{u}_{ik,\perp}$ and $(B')\hat{u}_{jk,\perp}$. In the absence of noise, all of the $\phi_k$ are equal, and the expression is minimized when $\theta$ is set equal to $\phi_k$. In the more general case, the $\phi_k$ are noisy, and the estimate $\hat{\theta}$ is chosen to get the best alignment possible, meaning the alignment that minimizes the $f(\theta)$ term.

Understanding that minimizing (3.10) obtains a good estimate of the simple rotation angle $\theta$, I note that the solution that minimizes (3.10) can be obtained by analytic means. The argument that minimizes (3.10) is $\hat{\theta}$:

$$\hat{\theta} = \arctan \left( \frac{\beta}{\alpha} \right)$$ \hfill (3.11)

where $\alpha$ and $\beta$ are defined by

$$\alpha = \sum_{k \in \mathcal{K}} (A'\hat{u}_{ik,\perp} \cdot B'\hat{u}_{jk,\perp})$$ \hfill (3.12)
\[
\beta = \sum_{k \in K} || \hat{A}' \hat{u}_{ik,\perp} \times \hat{B}' \hat{u}_{jk,\perp} ||.
\] (3.13)

To be precise, there are two solutions to the arctan function on the range \([0, 2\pi]\). One of these solutions minimizes the cost function and the other maximizes it. In practice, the minimizing solution can be identified by a simple comparison of \(J_{ij}\) for both cases.

With an estimate of \(\theta\), the relative orientation between nodes \(i\) and \(j\) can be expressed by a single rotation matrix \(A^R B\).

\[
A^R B = A^R A' A' R(\theta) B' B' R B
\] (3.14)

This matrix can now be used to convert the measurements from node \(j\) (in coordinate system \(B\)) into a common coordinate system used by node \(i\) (coordinate system \(A\)). This process can be repeated to map all available measurement sets into the common coordinate system, so that the ANIM algorithm (as described in the previous section) can be applied. The result is that ANIM can be applied in the absence of global-orientation measurements (e.g. without requiring an AHRS).

### 3.1 Simulations of Rotational ANIM

Rotational ANIM is a powerful extension of the original ANIM algorithm in that it does not require global orientation to be known; however, because Rotational ANIM estimates a greater number of degrees of freedom, its accuracy may be slightly lower than that of conventional ANIM. This section quantifies
the change in positioning accuracy that occurs when using Rotational ANIM (with AoA measurements only) as compared to an ideal case when using the original ANIM algorithm (with AoA measurements and perfect knowledge of orientation).

In these simulations, I modeled a system of six nodes distributed in Euclidean space. Table 3.1 describes the true position of nodes 2 through 6 relative to node 1. I assumed each node communicated with all five other nodes. Also, each node sensed all five other nodes (for a total of 30 AoA measurements at a single time step).

Each node was assigned an orientation relative to node 1. Table 3.2 lists orientation angle values (roll, pitch, and yaw of nodes 2 through 6 relative to node 1). For Rotational ANIM processing, these orientation angles were estimated from AoA measurements. For the original ANIM processing, these orientation angles were assumed known.

To evaluate performance given random measurement noise \( \epsilon_{ij} \), a Monte Carlo simulation was performed. The simulations of the two algorithms (original ANIM and Rotational ANIM) used the same set of \( \epsilon_{ij} \) values. For each case, results were compiled over 20 Monte Carlo trials. Error was simulated as proportional in magnitude to the distance between nodes. In other words,
Table 3.2: Node Orientations Relative to Node 1

<table>
<thead>
<tr>
<th>Node</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roll (°)</td>
<td>16.2</td>
<td>15.1</td>
<td>-9.8</td>
<td>3.9</td>
<td>-14.2</td>
</tr>
<tr>
<td>Pitch (°)</td>
<td>13.9</td>
<td>0.0</td>
<td>-0.8</td>
<td>13.7</td>
<td>8.9</td>
</tr>
<tr>
<td>Yaw (°)</td>
<td>18.8</td>
<td>14.0</td>
<td>3.6</td>
<td>-10.4</td>
<td>-9.8</td>
</tr>
</tbody>
</table>

Each Monte Carlo simulation obtained unit pointing vector measurements by adding a random perturbation to the true pointing vectors $u_{ij}$. Two noise levels were considered: 1% and a 5% error (one sigma). These noise levels correspond approximately to an angular error of 0.57° and 2.86°, respectively.

The results of the 20 Monte Carlo trials are expressed in Figure 3.4. The figure shows results for each of the two noise levels (1% in Figure 3.4a, 5% in Figure 3.4b). In the figure, the true location of each node is shown as an open blue circle. All locations are relative to node 1, so without loss of generality, node 1 is illustrated at the origin. The positions estimated by Rotational ANIM processing (without knowledge of relative orientation) are shown as green dots. The positions estimated by the original ANIM (with perfect orientation knowledge) are shown as red dots.
CHAPTER 3. ROTATIONAL ANIM

Figure 3.4: Results for 20 Monte Carlo Trials for (a) 1% and (b) 5% Error
Table 3.3: Output Error Metrics for 1% Measurement Noise

<table>
<thead>
<tr>
<th></th>
<th>Node</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ORG/ROT</td>
<td>$\Sigma_{ij}$ or $\bar{\Sigma}_{ij}$</td>
<td>Units</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>ORG</td>
<td>$\Sigma_{ij}$</td>
<td>m</td>
<td>2.2</td>
<td>4.6</td>
<td>6.8</td>
<td>3.3</td>
</tr>
<tr>
<td>ROT</td>
<td>$\Sigma_{ij}$</td>
<td>m</td>
<td>2.9</td>
<td>6.2</td>
<td>11</td>
<td>5.7</td>
</tr>
<tr>
<td>ORG</td>
<td>$\bar{\Sigma}_{ij}$</td>
<td>–</td>
<td>.42</td>
<td>.85</td>
<td>1.2</td>
<td>.35</td>
</tr>
<tr>
<td>ROT</td>
<td>$\bar{\Sigma}_{ij}$</td>
<td>–</td>
<td>.72</td>
<td>1.3</td>
<td>2.9</td>
<td>1.1</td>
</tr>
</tbody>
</table>

In order to quantify error, I used the following metric that describes 3-D RMS error

$$\Sigma_{ij} = \sqrt{\sigma_{xx} + \sigma_{yy} + \sigma_{zz}}$$  \hspace{1cm} (3.15)

where $\sigma_{xx}$, $\sigma_{yy}$, and $\sigma_{zz}$ are the diagonal elements of the covariance matrix of the $x$, $y$, and $z$ position error. Because measurement errors are angular, estimated position errors are a strong function of the distance between nodes. For this reason, it is useful to scale the three-dimensional RMS error by the internode distance $d_{ij}$ to define a relative error metric

$$\bar{\Sigma}_{ij} = \frac{\sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 + \sigma_{zz}^2}}{d_{ij}}.$$  \hspace{1cm} (3.16)

Tables 3.3 and 3.4 list the position error and normalized position error for both algorithms and both noise levels (abbreviating the original ANIM as ORG and Rotational ANIM as ROT).

There are two important ideas to draw from comparisons of the data. First, there appears to be a strong dependence of the position error on geome-
try. Second, there is a clear reduction in positioning accuracy introduced when orientation must be estimated simultaneously. By visual inspection, there is roughly a factor of two increase in the three-dimensional RMS error for the case in which orientation must be estimated (using Rotational ANIM).

As a final consideration, since ANIM algorithms are iterative, it is important to address convergence. The original ANIM algorithm has been proven to converge, with errors decreasing monotonically in each subsequent iteration [12]. Typically the original ANIM algorithm converges well within about 10 iterations. By contrast, the Rotational ANIM algorithm is not guaranteed to converge. In fact, empirical evidence indicates that the algorithm converges poorly when the relative angles between nodes are large (e.g. larger than 20°). For the relative orientations described in Table 3.2, reliable convergence was observed; however, errors did not converge monotonically. An example is shown in Figure 3.5.

The figure illustrates convergence in terms of the smallest singular value associated with each ANIM optimization step, solving either equation (2.4) for the normal, as shown in blue, or equation (2.6) for the pointing vector, as
shown in red. Ideally the minimum singular value in each case would be zero (indicating perfect orthogonality of normal and pointing vectors). Singular values are shown as a function of iteration for each of 20 Monte Carlo trials. Only the results for Rotational ANIM are shown.

![Graph showing convergence properties](image)

Figure 3.5: Convergence for Sample Set of Data

The illustration shows reasonable convergence properties for the scenario studied in this dissertation. In fact, the singular values associated with estimating the pointing vector converge monotonically toward zero. The behavior of the singular values for normal-vector estimation, by contrast, diverged initially in several cases before converging again toward zero. It would appear that the divergence is introduced by the rotational correction step (since such
divergence is not observed for the original ANIM method). The divergence is a clear problem if the relative orientation angles are too large, as subsequent iterations are not guaranteed to transition from divergence to convergence in those cases. More work is needed to find a variant of Rotational ANIM for which convergence can be guaranteed.
Chapter 4

ANIM in Experiment

4.1 Introduction

All navigation systems are subject to error. In order to understand whether or not a navigation system is sufficient for a particular application, it is necessary to characterize that error. This chapter describes a method that is used to characterize measurement error for a particular type of angle-of-arrival (AoA) sensing: optical sensing using a conventional camera. In practice, the camera can be replaced with a transceiver or any other type of sensing-transmission equipment that could be used to take AoA measurements.

In the simulations of Rotational and Original ANIM in Section 3.1, I chose noise levels of the input AoA measurements to be 1% and 5% (one sigma). These percentages convert the true unit pointing vectors to noisy unit pointing vectors, as described in Section 3.1, by adding a random perturbation to the

---

This chapter is the extension of the “Hardware Acquisition of Bearing Measurements” section of [20]
true pointing vectors with the chosen one sigma. The corresponding angular error levels are 0.57° and 2.86°, respectively. The normal distribution with one sigmas of 0.57° and 2.86° were assumed to be viable noise models for the input AoA measurements. The purpose of this chapter is to explore if these noise models accurately describe the error in an optical sensing system.

Looking at Figure 4.1, a simple relationship exists between the input sensor error and the output position error.

![Figure 4.1: Input and Output Error Relationship in ANIM Algorithm](image)

I invert this relationship in order to determine sensor error of my optical system. The reason for this is not obvious. It would be straightforward to experimentally characterize the input error, so that simulations can be used to predict the output error for different spatial configurations of the collaborators. However, it is hard to determine the angular error of such an optical system by solely taking optical measurements with one sensor because of low accuracy or nonexistent ground truth of the AoA measurements and the inability to combine different sources of error into a single error model. Significant systematic errors are prevalent in applications of ANIM with camera AoA measurements. I would like to mimic these systematic errors. These errors are due to characteristics in the environment such as lighting, error in orientation of sensor, etc. in which the optical AoA measurements are taken. Therefore, I have devised a method to invert the relationship between input sensor error and
output position error that uses a nonlinear algorithm, ANIM, to determine input sensor error. This method compares the output position error of physical experiments, which have unknown input sensor error, to those of simulations having known input sensor error. Through this comparison, I can determine the unknown input sensor error of the experiments. It is important to determine an experimental input sensor error so that it can be used as a validated input when working on further iterations of ANIM.

This chapter begins with a description of the equipment I used in single and multi-camera experiments. I then describe the theory behind three-dimensional optical AoA measurements stemming from information from a two-dimensional image. Section 4.3 expands upon the difficulty of determining the angular optical measurement error. Next, the “inverse method” is introduced as an alternate method to determine the error in the optical AoA measurements. An experiment is conducted using camera AoA measurements, which have unknown sensor error, to estimate the position of three nodes. This experiment is then simulated with different input sensor errors. Finally, this chapter concludes with the comparison of the experimental and simulation position estimates to determine the experimental sensor error.
4.2 Attributes of Camera AoA Measurements

4.2.1 Node Hardware

The system I have tested is an autonomous, decentralized, and homogeneous group of nodes that have the potential to collaborate to complete a task. In experiments, in which I will describe in depth in the rest of this chapter, I used a few sensor-microprocessor pairs to serve as nodes. Figure 4.2 is an example of such a sensor-microprocessor pair. This was a reasonable substitute for testing a decentralized group of nodes collaborating to complete a task because theoretically each sensor-microprocessor pair could be packaged into a UAV and serve as the transceiver.

There are four main components to each node: a transmitter, a receiver, a computer, and communication.

A red LED acts as the indicator or transmitter for each node. Any node
can create angle-of-arrival measurements by identifying the direction of the light arriving from the LED sources on other nodes. To keep the node’s LED and camera close (within 2.5 cm), I put the LED on top, centered over the lens. A PLA three-dimensional-printed part was designed to hold the LED to the top of the camera (see drawing in Appendix A).

Each receiver was one of three types of USB cameras [41, 42, 43]. The camera acted as a receiver because it captured the intensity and location of the LED in the camera’s image plane.

A myRIO microprocessor [44] acted as the computer of each node. It had the capability of controlling onboard vision processing using an FPGA. The image processing code was written in LabVIEW.

The communication subsystem was WiFi hardware that communicated using Transmission Control Protocol (TCP). Each node was a TCP server as well as a TCP client.

4.2.2 Theory of Camera Angle-of-Arrival Measurements

Angle-of-arrival measurements are unit pointing vectors originating at the measuring node and pointing towards the measured node in the body-fixed coordinate system of the measuring node. I have used cameras to infer the AoA measurements by and to all of the nodes in experiment. Generating unit pointing vectors begins with an image taken by the camera of another node. The image is thresholded so that only the LED of the other node will show up in the processed image. The steps involved in the image processing are described in next section. From the processed image, the centroid of the image of the LED is obtained. The centroid of the LED is expressed in \((x, y)\)
pixel coordinates, and is represented by the red pixel in Figure 4.3.

![Figure 4.3: Calculating Unit Pointing Vector from Centroid Position](image)

The camera-fixed basis can be described by the orthonormal unit vectors $c_x$, $c_y$, and $c_z$. As seen in Figure 4.3, $x$ and $y$ are in the $c_x$ and $c_y$ directions. The focal length, denoted $f$, is the distance from the image plane to the camera focal center (in the $c_z$ direction). I calibrated the distance $f$ in units of pixels, described in Section 4.2.3. To convert the $(x, y)$ pixel location of each centroid to a unit pointing vector $\hat{u}$, I used

$$
\hat{u} = \frac{x}{u_o} c_x + \frac{y}{u_o} c_y + \frac{f}{u_o} c_z. 
$$

(4.1)

The scaling parameter $u_o$ is the normalizing factor to convert the vector $u$ into the unit pointing vector $\hat{u}$, which is calculated by
\[ u_o = \sqrt{x^2 + y^2 + f^2}. \] (4.2)

In order to calculate the \( f \) in pixels, pinhole camera model theory can be used because these USB cameras can be approximated as pinhole cameras [45]. Using the two-dimensional pinhole camera model in Figure 4.4, I get a simple relation between \( x_r \) (or \( y_r \)) and \( x_i \) (or \( y_i \)), where \( x_r \) is the length of the real object and \( x_i \) is the length of the object in the image.

![Figure 4.4: Pinhole Camera Two-Dimensional Image Relation](image)

Equation (4.3) is the mathematic relation of the similar triangles created by the pinhole camera model in Figure 4.4.

\[
\begin{bmatrix}
  x_i \\
  y_i
\end{bmatrix} = -\frac{f}{z_r} \begin{bmatrix}
  x_r \\
  y_r
\end{bmatrix}
\] (4.3)

The negative sign in the equation comes from the fact that \( x_i \) is the inverse-mirror image of \( x_r \).

To calibrate focal length \( f \), in pixels, one approach is to place a camera at a known distance to an image plane and measure the distance of the plane
from the camera focal point \((z_r)\) and the horizontal displacement of a point on the plane from the optical axis \((x_r)\). If the location of that point on the image is \(x_i\), then

\[
 f = x_i \frac{z_r}{x_r}. \tag{4.4}
\]

The negative sign from (4.3) is ignored, because relative orientation of the image and the real image plane is unimportant because the software on the myRIO flips the image. In the next section, this approach is used to calculate \(f\) for each of the three types of USB cameras used in experiment.

### 4.2.3 Camera Focal Length

In my experiments, I used three different types of cameras. The focal length of each one was determined in pixels using (4.4). To determine \(f\), I placed each camera at a fixed distance, \(z_r\), from a ruler positioned horizontally in the image plane, as seen in Figure 4.5. I then chose 25-30 points along the ruler to serve as the horizontal displacement of points on the plane from the optical axis \((x_r)\). Looking at the image of the ruler, I found the \(x\) coordinate of each of the 25-30 points, which were the corresponding \(x_i\)’s.
Using (4.4), $f$ was determined for each point along the ruler and then averaged to get a final value of $f$ for each camera. I ran this procedure 3-5 times for each camera, moving the camera and replacing it between trials to minimize systematic error. The resulting focal lengths and standard deviations of the focal lengths are displayed in Table 4.1.

For each camera, the standard deviation of the focal length was less than 4% of the respective focal length, which is good because it validates the mean focal length.

In Figure 4.6, $x_i$ versus $x_r$ for all 25-30 points along the ruler is plotted for each camera. The points are fitted with first, second, and third degree
Table 4.1: Camera Specifications

<table>
<thead>
<tr>
<th>Camera Type</th>
<th>Image Dimension</th>
<th>$f_{pixels}$ (pixels)</th>
<th>$f_{pixels ; std}$ (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera A Logitech QuickCam Pro 9000 [41]</td>
<td>640 x 480</td>
<td>530</td>
<td>8.3</td>
</tr>
<tr>
<td>Camera B Genius WideCam F100 [42]</td>
<td>640 x 480</td>
<td>305</td>
<td>4.7</td>
</tr>
<tr>
<td>Camera C AUSDOM WebCam AW310 [43]</td>
<td>1280 x 720</td>
<td>988</td>
<td>38</td>
</tr>
</tbody>
</table>

polynomials, depicted by the red, light blue, and dark blue lines respectively. For each of the cameras, the linear term does not change much as the degree of the fitted polynomial increases. Therefore, it is appropriate to approximate the fit as linear. The slope of the first degree polynomial is $\frac{f}{z}$ from (4.4). Since the linear fit was appropriate for this data, it suggests that I did not need to calibrate the cameras for the effect of radial distortion, which is important to do if needed. Therefore, no corrections were made to adjust for radial distortion in later experiments.
Figure 4.6: Camera Calibrations
4.2.4 Image Processing

To find the position of the LED in an image, the centroid of the LED was located using a simple progression of image processing steps. This progression is represented by the schematic in Figure 4.7.

![Figure 4.7: Image Processing Steps](image)

The image processing for each node was done onboard each myRIO and was written in LabVIEW 2015. The first step of the image processing was to extract the red plane from the RGB image taken by the USB camera. I chose to extract the red plane from the original RGB image because it resulted in the most symmetric pattern, such as in Figure 4.8a. Figure 4.9b is an example of an extracted red plane of the RGB image in Figure 4.9a. To reduce noise from the overhead fluorescent lights in the lab where the experiments were taken as well as from the sunlight, using the green or blue plane would have been a good idea. However, the intensity pattern of the LED in the green and blue planes often looked like that in Figure 4.8c, which is not a symmetric pattern. With the time constraints of this project, there was not enough time to sample various LEDs to find one that produced the best intensity pattern in the blue or green planes with the sensitivities of the USB cameras available to me. Figure 4.8a is an example of what I consider a good intensity pattern of an LED for purposes of finding the centroid of the pattern. Figure 4.8b is an example...
of an acceptable intensity pattern because it is symmetric, even though the true centroid of the LED has been lost. Symmetry in the LED pattern of an image is important because it results in a worse estimate of the centroid in later processing steps.

The next processing steps were to convert the red plane to grayscale and threshold the intensity values by a certain window. In grayscale, each pixel has an associated intensity value between 0 and 255, 0 being black and 255 being white. The thresholding window was chosen in each lighting situation, according to what produced a better LED pattern. Typically the minimum value of the thresholding window was 252 and the maximum value was 255. Thresholding creates a binary image from the grayscale image where every pixel in the grayscale image with a value outside of the window is assigned a 0 and every pixel with a value inside the window is assigned a 1. The result is a binary image that highlights the areas of the image with values in the thresholding window only. Figure 4.9c is an example of the binary image of the original image in Figure 4.9a.
The final step was to perform blob analysis on the resulting binary image. This was done using the “Particle Analysis VI” in LabVIEW [46], which performs a searching algorithm that assigns each white pixel in the image to a blob using a connectivity condition of any of the eight surrounding pixels. The centroid of each blob, \((x_c, y_c)\), was determined using the center of mass equation where each of the M pixels has a unit mass.
This center of mass method is validated by Stone et al. who determined that the median and Gaussian methods are the most effective to find the centroid of an image of a star [47]. Finding the centroid of a star in an image is similar to that of an LED, so it is valid to use the center of mass method. Other image processing methods were attempted, but the best estimates of LED centroid location came from the method described.

4.3 Camera Error Model

Before detailing the inverse method described in Section 4.4, the difficulty of measuring error in camera AoA measurements must be explained. The purpose of this analysis is to try to estimate an error model that mimics the sources of error in a system of optical camera AoA transceiver. There are many sources of error that could be accounted for in this error model. The most important sources of error to characterize are the ones with the greatest affect on the precision and accuracy of the centroid measurement of the LED in the processed image. I hypothesized that these sources included non-artifact precision errors, camera artifact errors, orientation errors, and a systematic LED-lens offset. Non-artifact precision errors are errors that affect the measurement of the centroid of the LED in an image, but that still result

\[ x_c = \frac{1}{M} \sum_{i=1}^{M} x_i \]  

\[ y_c = \frac{1}{M} \sum_{i=1}^{M} y_i. \]
in measuring the correct object in the image. Camera artifact errors are errors that cause the wrong blob to be measured in an image. Orientation errors are errors in the camera AoA measurement due to the orientation of the sensor. The systematic LED-lens offset refers to the systematic offset caused by the distance between the indicator and the true camera center, which is the center of the camera lens. I will go into further detail about these sources in the rest of this section.

4.3.1 Non-Artifact, Precision Errors

In this section I will describe how I modeled random noise with a standard deviation of 0.05°. I hypothesized that lighting, the distance between the observing and observed nodes, and the position of the LED light on the camera’s CMOS active pixel sensor would affect the precision of the centroid measurement. Lighting is a probable source of error in the centroid measurement because under different light conditions, the processed image, like those in Figure 4.8, became either over saturated or the LED pattern was unsymmetric. Oversaturated images produced measurements of incorrectly identified objects (i.e. not the LED). Unsymmetric patterns of the LED produced less precise centroid measurements. I hypothesized that the distance between one node’s camera and the LED of another node could affect the measurement more as the distance increased because the number of pixels representing the LED decreases as this distance increases. Lastly, I deduced that a large source of error in the centroid measurement could result from the mechanism the camera used to convert the intensity of light into a digital reading. This mechanism is called a complementary metal-oxide semiconductor (CMOS) ac-
tive pixel sensor. A CMOS active pixel sensor is an array of tiny photo sensors that convert the intensity of light into a voltage, which is then converted to a digital value using transistors [48]. I predicted that errors could occur due to where on the array of photo sensors the light from the LED was captured.

To assess the error due to lighting, sensor-indicator distance, and the camera’s CMOS active pixel sensor, I performed a series of LED centroid measurements under different conditions. These trials were taken in the CRISP lab with a setup that consisted of one node placed on the floor with an LED (representing another node) placed at the same height from the ground. A reference line was placed on the ground to align the camera. Figure 4.10 is an example of such a setup.

![Figure 4.10: Precision Measurement Experimental Setup](image)
In each of the 7 trials, I chose a lighting condition and a position for the LED. The lighting conditions consisted of a combination of adjustable overhead lighting (fluorescent) and sunlight from large windows facing the North-East. The position of the LED was fixed for each trial. The position of the LED changed between trials to compare different distances between the sensor and LED and to compare where the LED light hit the CMOS active pixel array by changing the angle of the LED. Figure 4.10 shows the setup of trials 5 through 7 with constant angle and varying distance. The distance increased as the LED was moved from Position 1 to Position 3 in trials 5 to 7. Figure 4.11 is an example of the image from the camera during the experimentation.

Figure 4.11: Image Used to Align Camera

For each run in each trial, the camera was realigned with the reference line. To align the camera, I looked at the image from the camera (i.e. Figure 4.11), which has a red line superimposed down the center of the image, and aligned the red line with the reference line by eye.

The results of the 7 trials are contained in Table 4.2.
Table 4.2: Single Camera Trials

<table>
<thead>
<tr>
<th>Trial</th>
<th>Lighting</th>
<th># Points</th>
<th>Angle (°)</th>
<th>Distance (cm)</th>
<th>$std_x$ (pixels)</th>
<th>$std_y$ (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>afternoon, overheads on</td>
<td>40</td>
<td>4.9</td>
<td>59.7</td>
<td>0.24</td>
<td>0.28</td>
</tr>
<tr>
<td>2</td>
<td>night, overheads on</td>
<td>15</td>
<td>4.9</td>
<td>59.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>night, overheads on</td>
<td>50</td>
<td>4.9</td>
<td>59.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>sunny afternoon, overheads off</td>
<td>50</td>
<td>9.7</td>
<td>59.7</td>
<td>0.38</td>
<td>0.32</td>
</tr>
<tr>
<td>5</td>
<td>night, overheads on</td>
<td>50</td>
<td>18.8</td>
<td>59.7</td>
<td>0.40</td>
<td>0.31</td>
</tr>
<tr>
<td>6</td>
<td>night, overheads dimmed</td>
<td>50</td>
<td>18.9</td>
<td>110.5</td>
<td>0.46</td>
<td>0.15</td>
</tr>
<tr>
<td>7</td>
<td>night, overheads dimmed</td>
<td>50</td>
<td>19.0</td>
<td>181.6</td>
<td>0.48</td>
<td>0.37</td>
</tr>
</tbody>
</table>

From the data above, I would bound the standard deviation of a centroid by 0.5 pixels, thereby bounding the precision of the centroid measurement by 0.5 pixels. Since an angular rather than a pixel precision value was desired, I must find the corresponding standard deviations in degrees. I calculated the $x$ and $y$ standard deviation of each trial in degrees from pixels using the following method, where the mean centroid value of $(x_0, y_0)$ has the corresponding standard deviations $(std_x, std_y)$. $std_x$ bounds $x_0$ by $x_1$ and $x_2$ as follows

$$x_1 = x_0 - std_x$$

$$x_2 = x_0 + std_x.$$  \hspace{1cm} (4.7)

$$x_{2} = x_{0} + std_{x}.$$ \hspace{1cm} (4.8)

The units are in pixels. To convert this to an angular bound, the focal length, $f$, is used to bound $x_0$ by $\theta_1$ and $\theta_2$ using the following equations:
Table 4.3: Single Camera Trials Standard Deviations

<table>
<thead>
<tr>
<th>Trial</th>
<th>(\text{std}_x) (pixels)</th>
<th>(\text{std}_y) (pixels)</th>
<th>Angular (\text{std}_x) (°)</th>
<th>Angular (\text{std}_y) (°)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-3</td>
<td>0.24</td>
<td>0.28</td>
<td>0.026</td>
<td>0.030</td>
</tr>
<tr>
<td>4</td>
<td>0.38</td>
<td>0.32</td>
<td>0.040</td>
<td>0.035</td>
</tr>
<tr>
<td>5</td>
<td>0.40</td>
<td>0.31</td>
<td>0.039</td>
<td>0.034</td>
</tr>
<tr>
<td>6</td>
<td>0.46</td>
<td>0.15</td>
<td>0.045</td>
<td>0.016</td>
</tr>
<tr>
<td>7</td>
<td>0.48</td>
<td>0.37</td>
<td>0.046</td>
<td>0.040</td>
</tr>
</tbody>
</table>

\[
\theta_{x1} = \arctan\left(\frac{f}{x_0}\right) - \arctan\left(\frac{f}{x_1}\right) \tag{4.9}
\]
\[
\theta_{x2} = \arctan\left(\frac{f}{x_2}\right) + \arctan\left(\frac{f}{x_0}\right). \tag{4.10}
\]

The same procedure follows for \(y_0\). Because of my small values of \(\text{std}_x\) and \(\text{std}_y\), the \(\arctan\) function can be considered to be approximately linear over this range. Therefore, the angular \(\text{std}_x\) is equivalent to \(\theta_{x1}\) and \(\theta_{x2}\). Table 4.3 contains the standard deviations from the 7 trials. I have combined the data from the first three trials because they have different lighting conditions, but the same angle and distance values.

Looking at the results in Table 4.3, there were a few important trends. While holding the angle of the LED constant and increasing the distance between the LED and sensor (trials 5 through 7), the angular standard deviation generally increased. I expected this, because I hypothesized that uncertainty should increase with distance due to the decreasing size of the LED in the image. In comparing trials 1-3, 4, and 5-7, there was a slight trend of increasing error with increasing distance from center of the CMOS active pixel array (as...
the angle increases). Due to the limited number of points, however, this might have been a negligible trend.

Each trial was conducted in a different environment. I labeled these experiments as precision measurement experiments because ground truth of the LED was not accurately known. This is because there is not a separate method to determine the AoA measurements to serve as the ground truth. The ground truth of the AoA measurements would illuminate the accuracy of the camera AoA measurement method described in Section 4.2.2. Since such a ground truth method does not exist, only precision of the camera AoA measurement from the changes in lighting, position on the CMOS, and distance from the LED can be estimated.

In conclusion, I estimated that the precision of a centroid measurement in a certain environment was $0.05^\circ$.

### 4.3.2 Camera Artifact Errors

Camera artifact errors are errors due to measuring the centroid of the wrong blob in the image. Causes of this phenomena include other concentrated light sources in the field of view of a node. It was very difficult to estimate the distribution of errors produced by such a phenomena because it was highly dependent on the environment. This challenge of modeling the artifact-induced error distribution is one of the main reasons why an inverse method was required to model camera AoA measurement error.
4.3.3 Orientation Uncertainty

The orientation of the measuring node was potentially a large cause of correlated error across all the AoA measurements acquired from a single camera image. In taking AoA measurements with cameras, each node was assumed to have a global orientation in the room. Since my experiments involved placing nodes on the floor of a lab, I estimated the tilt and pan (see Figure 4.12) of each node relative to an arbitrary global orientation in the lab.

![Figure 4.12: Tilt and Pan Directions of a Node](image)

Initially, I had assumed the floor was relatively level and by placing a node on the floor, I assumed its tilt was within a tenth of a degree of $0^\circ$. However, after looking at the three camera experiments (see Section 4.4.4), I noted some places on the floor were not quite level. Therefore, I investigated the tilt error using a smartphone accelerometer to get a gist of the levelness of the floor where I took all of my data for this project. The results are shown in Figure 4.13.
Figure 4.13 suggests the levelness of the floor followed a Gaussian distribution centered at 0°. The standard deviation of this distribution was 0.8°, which was a significant orientation error previously assuming 0° in tilt.

The second orientation error source was due to placement of the camera in the pan direction. The alignment process I described earlier in this section consisted of aligning the centerline of an image, Figure 4.11, from a sensing node with a centerline on the floor of the lab. I could align the camera within a few 10ths of a degree with the centerline. However, I did not have a way to measure the uncertainty in placing the centerline on the floor. This may introduce systematic bias into the system.

Ideally, I would estimate the orientation of each node directly using Rotational ANIM. This would eliminate this class of error. However, the exper-
imental setup that I used is two-dimensional and Rotational ANIM requires three-dimensional geometries of nodes. Therefore, I must add orientation error into my model. Overall, I would assess this source of error to produce an offset in orientation for each node of 0.8° in the tilt direction and 0.1° in the pan direction.

4.3.4 Systematic LED-Lens Offset

I assessed that the systematic LED-lens offset to be a distance of $R \sin \theta$, where $R$ is the distance between the transmitting and receiving nodes, and $\theta$ is the angular measure between the center of the lens and the LED. There was a systematic offset in every AoA measurement taken due to the transceiver setup I have chosen. This was because the observing node measures the AoA to the transmitter (LED) of the observed node, which was offset from the center of the observed node’s camera by a distance. Figure 4.14 shows the relationship between the observed node’s transmitter (LED) and the observing node’s receiver.
The blue lines represent the distances between the receiver of the observing node and the transmitter of the observed node and between the receiver of the observing node and the receiver of the transmitting node. The important distance was the one between the receivers of the two nodes because the center of the camera’s lens (receiver) was where the AoA measurements originate. The offset distance, $x$, was 2.70 cm, 2.54 cm, and 2.54 cm in cameras A, B, and C respectively. $\theta$ is the angular offset corresponding to the offset distance. The error produced by distance was an angular offset. However, it was an offset that was dependent on $R$, the distance between nodes. Thus, it was hard to quantify for an arbitrary geometry of nodes. I quantified this as a persistent bias of $R \sin \theta$. 

Figure 4.14: Angular Distance Between LED and Camera Centroid
Table 4.4: Sources of Error

<table>
<thead>
<tr>
<th>Potential Source of Error</th>
<th>Type of Error</th>
<th>Quantity of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision Error, Neglecting Artifacts</td>
<td>Normal Random Distribution</td>
<td>0.05°</td>
</tr>
<tr>
<td>Camera Artifact Errors</td>
<td>Anomalous Errors</td>
<td>??</td>
</tr>
<tr>
<td>Orientation Uncertainty</td>
<td>Orientation Offset</td>
<td>0.8° tilt, 0.1° pan (correlated for each camera)</td>
</tr>
<tr>
<td>Systematic LED-Lens Offset</td>
<td>Persistent Bias</td>
<td>$R \sin \theta$</td>
</tr>
</tbody>
</table>

4.3.5 Issue with Error Estimation

Throughout this section, I have described possible sources of error and methods of their estimation. The goal of this project was to obtain a method to estimate the error in optical AoA measurements that was representative of error in applications that would use this sensor. Therefore, it would be prudent to combine these different sources of error to find a resulting error distribution. However, there was an issue with combining these errors. These sources manifested as errors in AoA measurements differently. Table 4.4 displays these sources.

The precision error can be attributed to random noise with a normal distribution of error with a standard deviation of 0.05°. The camera artifact errors were difficult to model because they are highly sensitive to environmental conditions. The orientation uncertainty could be expressed by an angular offset in the pan and tilt directions, which are correlated in measurements from the same camera. The systematic LED-lens offset was a persistent systematic error that was distance dependent. The ways in which these sources of error
combine was unclear. Additionally, accurate ground truth was not available, which lead to the inability to measure the accuracy of the centroid measurements, only the precision. In conclusion, it was very hard to come up with a single distribution to describe the error in camera AoA measurements by estimating all possible sources of error. Therefore, another method was necessary to estimate such a distribution that mimics the error of camera AoA measurements in a physical system.

4.4 Inverse Method

4.4.1 Introduction

![Diagram of the Inverse Method](image)

Figure 4.15: The Inverse Method to Determine Experimental Sensor Error

Now that I have proven that it was very difficult to model the sensor error of camera AoA measurements by estimating all possible sources of error, I attempted a second approach. This approach began with an experiment that used camera AoA measurements with unknown sensor error to estimate position using the ANIM algorithm. Figure 4.15 is a schematic of this method,
beginning at “1.” Once a position estimate (in blue) has been made, a comparison to simulation follows. The simulations began at “2” in Figure 4.15 with a small distribution of sensor error (in red) and output a position estimate (in blue). If the simulated position error distribution equaled the experimental position error distribution, then I would estimate the unknown sensor error as that of the simulation. However, if the simulation position error was less than the experimental position error, I would continue to increase the simulation sensor error until the experimental and simulation position errors were equivalent. Once they were equivalent, my estimated experimental sensor error distribution would be that of the last simulation performed.

4.4.2 Two-Dimensional ANIM

In the following three-node experiment and simulations, I have assessed error in groups of nodes that were planar. ANIM, as described in Section 2.2, computes the relative position for groups of nodes containing greater than three nodes. When using ANIM with a group of three nodes, there is not enough information to estimate position in the out of plane direction (z-direction). However, by modifying the second step of ANIM, I can still estimate the positions of a group of three nodes in two dimensions.

Each node begins with the six measurements, $\hat{u}$’s, between the three nodes. These measurements are assembled into a matrix

$$A_{ijk} = [\hat{u}_{ij} \hat{u}_{ji} \hat{u}_{jk} \hat{u}_{kj} \hat{u}_{ki} \hat{u}_{ik}]^T.$$  \hspace{1cm} (4.11)

Referring back to the two steps in Section 2.2, the normal vector to the plane
formed by the three nodes is desired. I solve for this normal vector, $\hat{\mathbf{n}}_{ijk}$, by minimizing the dot product of itself with each of the unit pointing vectors in matrix $A_{ijk}$, which is equivalent to

$$\hat{\mathbf{n}}_{ijk} = \text{argmin}_n (\mathbf{n}^T A_{ijk} A_{ijk}^T \mathbf{n}).$$

(4.12)

$\hat{\mathbf{n}}_{ijk}$ is found using least-squares minimization, implemented efficiently by a singular value decomposition [38, 39].

The modification of the original ANIM algorithm begins during the second step of the algorithm. The purpose of the second step of ANIM is to reduce the error in the AoA measurements, $\hat{\mathbf{u}}$’s, using all of the $\hat{\mathbf{n}}_{ijk}$ available. Since there is only one group of three nodes, there is only one $\hat{\mathbf{n}}_{ijk}$ available. To edit the measurements, I subtract the component in the $\hat{\mathbf{n}}_{ijk}$ direction from each of the measurements

$$\hat{\mathbf{u}}_{new} = \hat{\mathbf{u}} - (\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}_{ijk}) \hat{\mathbf{n}}_{ijk}.$$  

(4.13)

This component exists in each $\hat{\mathbf{u}}$ although $\hat{\mathbf{n}}_{ijk}$ is the best fit normal vector to the plane containing the three nodes. Finally, the magnitudes of the measurement pairs are averaged (i.e. $\hat{\mathbf{u}}_{12}$ and $\hat{\mathbf{u}}_{21}$, $\hat{\mathbf{u}}_{13}$ and $\hat{\mathbf{u}}_{31}$, and $\hat{\mathbf{u}}_{23}$ and $\hat{\mathbf{u}}_{32}$). The measurement pairs that are averaged together are highlighted in Figure 4.16.
The rest of the algorithm continues as the original after (2.6) in Section 2.2.

### 4.4.3 Error Metric: Distance Root Mean Square Error

Before describing the three-node experiment and simulations, it is important to describe the error metric I used to compare the two. I have chosen to use Distance Root Mean Square (DRMS) error as my error metric [49]. I chose DRMS error because the AoA measurements are in two dimensions and a single error metric for position error is desired. DRMS error is calculated using the formula

\[
DRMS = \sqrt{\sigma_x^2 + \sigma_y^2} \quad (4.14)
\]

where \(\sigma_x\) and \(\sigma_y\) are calculated from the formula
where $dx$ is the difference between the true position and the estimated position in the x-direction and $R$ is the number of runs with the same parameters in the experiment or simulation. The purpose of using DRMS error as an error metric is to represent precision in two dimensions as a single scalar for comparison purposes. $\sigma_z$ has been left out of Equation 4.14 because ANIM is not given enough information to estimate the z-coordinate.

4.4.4 Three-Node Experiment

![Schematic of Three-Node Experiment](image)

Figure 4.17: Schematic of Three-Node Experiment
Following the first step in the inverse method from Section 4.4 ("1"), I conducted an experiment to measure the position error in a group of three nodes. These nodes were the same as described in Section 4.2.1. They were placed on the floor in the CRISP lab space in the $45^\circ - 45^\circ - 90^\circ$ configuration in Figure 4.17. The blue arrows represent the headings of each of the nodes. The length of each leg was 200.4 cm and that of the hypotenuse was 283.5 cm. Table 4.5 contains the camera type, pixel array size, and heading (yaw relative to the reference frame in Figure 4.17) of each node.

<table>
<thead>
<tr>
<th>Camera Type</th>
<th>Pixel Array Size</th>
<th>Yaw</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 1</td>
<td>B</td>
<td>640 x 480</td>
</tr>
<tr>
<td>Node 2</td>
<td>C</td>
<td>1280 x 720</td>
</tr>
<tr>
<td>Node 3</td>
<td>A</td>
<td>640 x 480</td>
</tr>
</tbody>
</table>

The heading of the nodes was accomplished by measuring and marking a reference (heading) line with a protractor on the floor for each node. This line was important to realign each node in between runs. The alignment process was the same as that used in the precision measurements in Section 4.3.1, in which a centerline superimposed on the image from each node’s camera was aligned with the reference line on the floor. Five separate trials were run with slight variation in lighting, resulting in 6 AoA measurements each.

The AoA measurements were processed by ANIM as if by Node 1. The separate sensor measurement to determine the scale factor came from measuring one of the short lengths of the triangle with a tape measure. The position estimates from Node 1’s point of view are displayed in Figure 4.18.
The blue circles represent the true position of the nodes as measured with a measuring tape. The red dots represent the five estimated positions of Nodes 2 and 3 by Node 1. At a closer look, the distributions of experimental values around the true positions were different between Node 2 and Node 3, as seen in Figure 4.19.
The position estimates (red) seem to be evenly distributed about the true position of Node 2, whereas the position estimates seem to be systematically offset from the true position of Node 3. To understand the magnitude of the error of the estimates, the mean error and DRMS are important to consider. The mean position error and DRMS of the position estimates are displayed in Table 4.6.

<table>
<thead>
<tr>
<th>Node</th>
<th>Mean X Position Error (cm)</th>
<th>Mean Y Position Error (cm)</th>
<th>DRMS (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.92</td>
<td>-0.23</td>
<td>0.065</td>
</tr>
<tr>
<td>3</td>
<td>5.17</td>
<td>0.91</td>
<td>4.48</td>
</tr>
</tbody>
</table>

Clearly there is larger error in Node 3’s position than Node 2’s from these metrics. The position error in Node 3 is largely in the $x$-direction. For a further analysis of these experimental results, simulations must be consulted.
4.4.5 Three-Node Simulations

Following the second step the inverse method from Section 4.4 ("2"), I modeled the system of three nodes from the experiment with simulated AoA measurements. The orientation and location of the nodes was that of Figure 4.17 with the same headings. I assumed each node could communicate with the other two and that each node could sense the others. A Monte Carlo analysis was performed where the inputs were noisy AoA measurements. These measurements were simulated by adding a random perturbation to the true pointing vectors. The random perturbation was from a normal distribution with increasing values of one sigma. 20 Monte Carlo trials were performed at each level of one sigma. Without loss of generality, ANIM was performed on the measurements from the point of view of Node 1. The output of each Monte Carlo trial was a position estimate of Node 2 and 3. The scale factor was determined by an assumed separate sensor measurement of one of the legs of the triangle. An example of 20 Monte Carlo trials with input noise 0.5° (one sigma) is illustrated in Figure 4.20.
The blue circles represent the true position of Nodes 2 and 3 relative to Node 1 and each red dot represents the position estimate of Node 2 or 3 in each of the 20 trials. At a closer look, Figure 4.21 shows the spread of the 20 position estimates of Nodes 2 and 3.
Both distributions seem to have a bias in estimated position in the negative $y$-direction, but are relatively well distributed about the true positions. This is in contrast to the position estimate distributions of the five experimental trials in Figure 4.19. Node 2 had relatively well distributed position estimates, while Node 3 had a clear bias. I will revisit possible causes to the experimental bias in the position estimate of only Node 3 shortly.

For each level of input angular error (one sigma), the mean error of the 20 trials in the $x$ and $y$ directions are displayed in Figure 4.22.
The blue symbols represent the mean position errors for Node 2 and the green symbols represent those for Node 3. The mean errors in the $x$-direction are represented by x’s, whereas those in the $y$-direction are represented by circles. Across all input angular error, the mean position error for both nodes in both directions seems to be spread well about 0. This suggests that the 20 Monte Carlo trials were a good representation of random error. Additionally, there is a trend of increasing mean position error with increasing input angular error, which is expected from results of previous simulations (Section 3.1).

The resulting DRMS error for each group of 20 Monte Carlo trials is plotted in Figure 4.23.
The DRMS errors of the position estimates of Node 2 are represented by blue circles and those of Node 3 are represented by green circles. The behavior of Nodes 2 and 3 are very similar as the angular error increases. To compare the output DRMS errors of the simulations to those of the experiments, I normalized the DRMS errors by the DRMS error from the experiment for the respective nodes using the equation

\[
DRMS_{\text{norm}} = \frac{DRMS_{\text{sim}}}{DRMS_{\text{exp}}}
\]  

where \(DRMS_{\exp,2}\) is 0.92 cm and \(DRMS_{\exp,3}\) is 5.17 cm. Figure 4.24 displays these normalized DRMS errors for both nodes.
Again, the blue symbols represent the normalized DRMS errors for Node 2 and the green symbols represent those for Node 3. The black line denotes a normalized DRMS error of 1, which is where the experimental DRMS error equals the simulation DRMS error. Therefore, this line crosses the distribution of points at the simulation input angular error that produces the same DRMS error as the experiment for Node 2 and Node 3 respectively. The resulting input angular error is $0.25^\circ$ for Node 2 and $1.25^\circ$ for Node 3. Again, this highlights a possible systematic bias in the position estimation of Node 3 relative to Node 2.

The estimate of the DRMS error lies in the interval

$$0.92 \text{ cm} \leq \sigma_{\text{input}} \leq 5.17 \text{ cm}.$$  \hspace{1cm} (4.17)

Because I have only five experimental data points, it is important to
look at the confidence interval of the simulation DRMS error. This assumes that the simulation DRMS error is the true DRMS error in the experiments. The metric DRMS can be treated like \( \hat{\sigma} \) projected along some direction on the \( xy \)-plane. Thus, the 95% confidence interval of \( \hat{\sigma} \) may be used for the experimental DRMS. The estimated variance is described by

\[
\hat{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \hat{\mu})^2. \tag{4.18}
\]

I define \( \sigma \) as the DRMS error from simulation and \( \hat{\sigma} \) as the DRMS error from experiment. The ratio of experimental DRMS error over simulation DRMS error is

\[
\frac{\hat{\sigma}^2}{\sigma^2} = \frac{1}{N-1} \sum_{i=1}^{N} \frac{(x_i - \hat{\mu})^2}{\sigma^2}. \tag{4.19}
\]

Redistributing the factor of \((N - 1)\) to the left side of the equation yields

\[
\hat{\sigma}^2 (N - 1) = \sum_{i=1}^{N} \frac{(x_i - \hat{\mu})^2}{\sigma^2}. \tag{4.20}
\]

The distribution of the right side of (4.20) is the Chi-Squared cumulative distribution function. Therefore, for a 95% confidence interval, the upper bound, \( \Delta_{ub} \), and lower bound, \( \Delta_{lb} \), can be described by \( \chi^2(0.975, N - 1) \) and \( \chi^2(0.025, N - 1) \) respectively, where \( N \) is 5. I can bound the left side of (4.20) by these bounds as follows.

\[
\Delta_{lb} \leq \frac{\hat{\sigma}^2}{\sigma^2} (N - 1) \leq \Delta_{ub} \tag{4.21}
\]
After some manipulation the ratio of experimental DRMS error to simulation error is bound

\[ \sqrt{\Delta_{ub}(N - 1)} \leq \frac{\hat{\sigma}}{\sigma} \leq \sqrt{\Delta_{ub}(N - 1)}. \] (4.22)

The 95% confidence interval for the ratio of simulation over experimental DRMS error is

\[ 0.6 \leq \frac{\sigma}{\hat{\sigma}} \leq 2.87 \] (4.23)

and thus the interval for the simulation DRMS error is

\[ 0.6\hat{\sigma} \leq \sigma \leq 2.87\hat{\sigma}. \] (4.24)

Taking the interval from (4.17), it can be transformed into a compatible interval with (4.24), where \( \sigma_{\text{input}} \) is the simulation DRMS error, \( \sigma \), and \( \hat{\sigma} \) is the mean of 0.92 cm and 5.17 cm or 3.05 cm. The following interval is (4.17) with each side of the interval multiplied by \( \frac{\hat{\sigma}}{3.05\text{ cm}} \).

\[ 0.3\hat{\sigma} \leq \sigma \leq 1.69\hat{\sigma}. \] (4.25)

Interval (4.25) does not fit inside of the 95% confidence interval (4.24) and in fact seems to be offset from the interval (4.24), which again suggests that a systematic error is present in the measurements of either Node 2 or 3.

Because much of this data suggests that a systematic error is present in
Table 4.7: Corrected Experimental Results

<table>
<thead>
<tr>
<th>Node</th>
<th>Mean X Position Error (cm)</th>
<th>Mean Y Position Error (cm)</th>
<th>DRMS Error (cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-0.24</td>
<td>-0.19</td>
<td>1.36</td>
</tr>
<tr>
<td>3</td>
<td>-0.77</td>
<td>1.81</td>
<td>2.13</td>
</tr>
</tbody>
</table>

the experimental data, an analysis of possible causes of error is important. One possible cause of this systematic bias in position estimation of Node 3 is an orientation error in the pan direction of 1.7°. This results in a shift of the centroids of Nodes 1 and 2 by 18 or 19 pixels in the negative $x$-direction in Node 3’s image. Adjusting for this bias in orientation, the means and DRMS errors of the position estimates for Nodes 2 and 3 are displayed in Figure 4.7. Comparing these values to those in Table 4.6, it is apparent that this orientation bias has an affect on the large error in the position estimate of Node 3. Another systematic bias may be present, however, after correcting for orientation error, the experimental distribution of position estimates of the two nodes look more similar to each other, as occurs in simulation. Figure 4.25 illustrates this fact.
If the simulated DRMS errors are renormalized by the new values of the DRMS errors for Nodes 2 and 3, 1.36 cm and 2.13 cm respectively, there will be a new estimate for the input angular error. Looking at Figure 4.26, it seems this new estimate is 0.25° for Node 2 and 0.50° for Node 3. These estimates are much closer together and suggest that the largest source of input error was the orientation of the nodes. Because correcting for orientation has such a dramatic effect on the measurements, again, suggests a strong motivation for Rotational ANIM.

Another source of the systematic bias could be an error in the ground truth position of Node 3 of 1 cm in the $y$-direction. The precision of measuring ground truth of each node was about 0.5 cm considering the measurement was done with a measuring tape. Therefore, it is highly likely that the uncertainty in the ground truth could contribute to the systematic bias in the position.
In conclusion, I have used the inverse method to infer a sensor error model for camera AoA measurements. This model is a Gaussian distribution with one sigma equal to $0.4^\circ$ if orientation errors are neglected. This model can be used in future simulations of different geometries of nodes. More broadly, this method can be used to determine sensor error for different kinds of sensors whose error is not straightforward to estimate.
Chapter 5

Conclusions and Future Work

5.1 Conclusions

This dissertation presented two significant contributions.

First, I have presented a decentralized algorithm that estimates relative position and orientation of vehicles by combining noisy angle-of-arrival (AoA) measurements. Simulations were used to compare the positioning accuracy with and without the availability of complementary orientation measurements (e.g. from an AHRS). When orientation information is not available, the position error of the algorithm increases by approximately a factor of two.

Second, I developed a method to predict ANIM error levels for arbitrary team geometries by using an inverse method to infer an approximate sensor error model for optical AoA measurements. I used experiments to characterize the position-errors obtained from ANIM and indirectly inferred an approximate input error distribution for camera AoA measurements by tuning a simulation to match the experimental results. The model is a Gaussian distri-
bution of random error of 0.4° (one sigma). The reason for such an approach was because it is hard to determine an input error model for ANIM of an optical system by solely taking optical measurements with one sensor because of low accuracy or nonexistent ground truth of the AoA measurements and the inability to combine different sources of error into a single error model. Systematic errors were present in the camera AoA measurements that depended strongly on environmental effects. One effect I found to have the greatest impact was the uncertainty in the orientation of the sensor. The uncertainty in the sensor orientation doubled the estimated output position-error. Thus, adding the orientation estimation piece to ANIM will be beneficial for systems using ANIM for localization. More broadly, this inverse method can be used in the future to characterize the error of other types of sensors that are difficult to estimate on their own.

5.2 Future Work

ANIM does not use information from previous time steps to estimate relative position of the current time step. Therefore, in the future, inclusion of information from the previous time step using Kalman Filters, for example, would be important to explore. A second area to explore is the performance of ANIM without measurements between all nodes. Adding radii of communication for each node could accomplish this in simulation.

There are two areas that could be improved in Rotational ANIM. The first step of Rotational ANIM assumes that \( \mathbf{u}_{ij} \) is aligned with \( -\mathbf{u}_{ji} \). However, this is not quite true because both measurements are noisy. Future work would
relax the assumption that $\hat{u}_{ij} = -\hat{u}_{ji}$. The second issue is that convergence is not guaranteed for Rotational ANIM as it is for the original version [40]. Thus, the future of Rotational ANIM will need to include 100% convergence.

In the future, the three node experiment could be expanded to include more nodes. Importantly, the existing experimental setup should be extended into three-dimensions to obtain a true evaluation of the three-dimensional capabilities of the ANIM and Rotational ANIM algorithms. To do this, each node will have to have multiple cameras to ensure measurements between all nodes can be taken. Once the performance of ANIM without all measurements is assessed in simulation, experiments in which some nodes cannot measure or communicate with some of the others nodes can begin. The ultimate proof-of-concept for ANIM would be to make the whole sensor network mobile by attaching the transceivers to quadcopters or mobile robots.
Appendices
Appendix A

LED Holder
Bibliography


[34] S. Thompson, D. Spencer, and J. Andrews. An assessment of the communications, navigation, surveillance (CNS) capabilities needed to support the future air traffic management system.


