
# Automated Aerial Refueling Position Estimation Using a Scanning LiDAR

THESIS

Joseph A. Curro II, Second Lieutenant, USAF

AFIT/GE/ENG/12-11

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright-Patterson Air Force Base, Ohio**

AFIT/GE/ENG/12-11

# Automated Aerial Refueling Position Estimation Using a Scanning LiDAR

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Joseph A. Curro II, B.S.E.E.

Second Lieutenant, USAF

March 2012

AFIT/GE/ENG/12-11

# Automated Aerial Refueling Position Estimation Using a Scanning LiDAR

Joseph A. Curro II, B.S.E.E.
Second Lieutenant, USAF

Approved:

| | |
|---|---|
| Dr. John F. Raquet, (Chairman) | date |
| Maj Kenneth A. Fisher PhD (Member) | date |
| Dr. Gilbert L. Peterson, (Member) | date |

AFIT/GE/ENG/12-11


## *Abstract*

This research examines the application of using a scanning Light Detection and Ranging (LiDAR) to perform Automated Aerial Refueling(AAR). Current attempts at AAR use Global Positioning System(GPS) and vision aided methods. This research thrust examines a method using a LiDAR in order to complement these existing methods. Specifically, this thesis presents two algorithms to determine the relative position between the tanker and receiver aircraft. These two algorithms require a model of the tanker aircraft and the relative attitude between the aircraft. The first algorithm fits the measurements to the model of the aircraft using a modified Iterative Closest Point (ICP) algorithm. This algorithm leverages the speed of a k-Dimensional Tree (k-D Tree) to quickly determine closest points between measurements and the tanker model. The second algorithm uses the model to predict LiDAR scans and compare them to actual measurements while perturbing the estimated location of the tanker. This algorithm requires a fast LiDAR simulator to quickly produce simulated scans from many position the plane was perturbed to. Each algorithm was tested with simulated LiDAR data before real data became available from test flights. Eight test flights involving a KC-135 and a Learjet as the surrogate receiver were conducted. The Learjet was outfitted with a sensor suite that included a LiDAR. Each aircraft was also outfitted with extra sensors to determine a truth trajectory for each flight. The data collected from this test flight was used to determine the accuracy of the two algorithms with real LiDAR data. After correcting for modeling errors the accuracy of each algorithm has a Mean Radial Spherical Error of about 40cm. This accuracy is well within bounds to aid either the current GPS or vision methods of AAR.

*Acknowledgements*

First and foremost, I owe a large debt of gratitude to everybody. Specifically I would like to thank everyone at the ANT center and AFRL that made the test flights possible. I especially want to thank Tom for manning the Trimble Total Station while I held the flag board for the boresighting as well as listening to any rambling I might have had about this thesis. I also want to thank Tim Penn for listening to my ideas about the thesis and getting the LaTeX template to work. I also want to acknowledge all those people out there that work on free open source software such as Ogre, Mogre, Notepad++, and all the other great free software because without software like that many theses including my own would not be possible. Finally I want to thank by thesis advisor Dr. John Raquet.

Joseph A. Curro II

*Table of Contents*

# *List of Figures*

# *List of Tables*

## *List of Abbreviations*

| Abbreviation | | Page |
|---|---|---|
| EGI | Embedded GPS Inertial Navigation Systems . . . . . | 40 |
| LEP | Last Estimated Pose . . . . . | 50 |
| LTP | Last Truth Pose . . . . . | 50 |

# Automated Aerial Refueling Position Estimation Using a Scanning LiDAR

## I. Introduction

This thesis focuses on using a scanning Light Detection and Ranging (LiDAR) to determine a relative position solution for two aircraft performing aerial refueling. The relative position solution enables the two aircraft to control themselves in order to maintain a relative position to allow Autonomous Aerial Refueling (AAR). This effort is motivated by the introduction of Unmanned Aerial Vehicles (UAVs) into the United States Air Force (USAF) that require the capability for aerial refueling.

The future UAVs will be built for the purpose of combat and not reconnaissance, such as the current generation of UAVs that includes the RQ-4 Global Hawk and MQ-9 Reaper [5] [4]. This new generation of Unmanned Combat Aerial Vehicles (UCAVs) such as the Navy X-47B Pegasus, are being designed with AAR as a capability of the final version [26]. These planes require AAR as a tactic for mission success. Current manned combat vehicles all use aerial refueling as a utility to extend range, payload, and loiter time. The USAF is moving forward with UCAVs as a part of the future of warfighting [18]. The new UCAVs will be sought as replacements to the current combat aircraft, but in order for this replacement UCAVs must take on all the responsibilities of manned combat aircraft including aerial refueling [17].

Current solutions to the AAR problem use the Global Positioning System (GPS) [11] and predictive rendering vision for relative position [31]. The predictive rendering vision solution uses virtual reality image estimates of the tanker, based on a model, to compare to actual camera images [31]. This method has limitations in accuracy but is robust to electronic interference. The GPS solution, while very accurate, can have problems with satellite acquisition when the tanker aircraft blocks the view of the sky for the receiver aircraft [15]. Also GPS signals are very low power that are

subject to electronic interference. AFRL is specifically looking into different sensors to use an alternative to situations when GPS is unavailable [30]. One such sensor is a LiDAR.

LiDAR has typically been used in navigation to determine the surrounding environment [23]. The LiDAR provides information about the unpredictable environment, such as object tracking and plane detection [25], to aid in navigation. Typical flight navigation however, is in a predictable environment without many other objects in close proximity. This is enforced by standards that require flights to be deconflicted before takeoff. This allows navigation without a high accuracy position estimate. In these cases, a LiDAR can be used to navigate by scanning the ground [28]. A notable exception to these cases is the aerial refueling situation where the environment will contain at least one other plane in close proximity. A LiDAR sensor now can be useful for flight navigation to determine the location of the other plane. This navigation attempt is very different from previous environmental navigation techniques, because the environment is much more predictable since the other plane can be known beforehand. This allows the LiDAR to search specifically for one object instead of searching for any number of unknown objects. Because of this assumption, navigation with a LiDAR in flight has many advantages over normal LiDAR navigation, and can achieve a more accurate position solution.

### *1.1 Problem Statement*

This thesis focuses on using the LiDAR to determine an accurate relative position solution between a receiver aircraft and a tanker for which a model is available. The attitude of each aircraft is provided by high quality Inertial Navigation Systems (INS). The aircraft are assumed to start in close proximity within range of the LiDAR with an accurate estimate of the initial relative position. This thesis analyzes two different algorithms to determine relative position. One method attempts to fit the LiDAR measurement to a known model of the tanker aircraft, while the second

predicts the measurements and searches for a position where the prediction closely resembles the real measurements.

## *1.2 Overview*

This thesis will be organized as follows. Chapter II will introduce the mathematical notation used in this thesis and include the relevant coordinate frames and the conversions between them. This chapter will also provide a brief background on LiDAR and graphics rendering. Also included is an overview of the necessary algorithms and techniques required by the overall relative position algorithms.

Chapter III details the two algorithms used to determine the relative position solution with a LiDAR, and a method to boresight the LiDAR. Each algorithm is described mathematically and the major techniques used in each algorithm are detailed and explained. The first algorithm explains the modifications to the Iterative Closest Point (ICP) algorithm in order to use the model of the aircraft. The second algorithm details the LiDAR simulator that is used to compute predicted measurements from the LiDAR using the aircraft model. Finally the mathematics and procedure to boresight the LiDAR are presented. An error analysis of the boresight procedure is also be explained and compared to an actual boresight completed for this thesis.

Chapter IV analyzes the algorithms and their characteristics. The parameters used in each algorithm are detailed. The characteristics of each algorithm are discussed including their strengths and weaknesses. The test flights conducted to record data used in the analysis are described, and in the results from test flights are computed using each algorithm in order to evaluate performance and possible places for error and methods to mitigate the error. A final position solution accuracy is shown given the test flight data.

Finally Chapter V concludes the thesis with an overview of the results of both the algorithms used for the relative position solution. This section also suggests areas for future work on how to improve both algorithms in terms of accuracy and integrity.

# II. Mathematical Background and Previous Research

This chapter describes the mathematical background and notation required to develop the algorithms in Chapter III. First a standard mathematical notation is developed to use throughout the document. Next the necessary reference frames are introduced and described in detail. Next a brief overview of LiDAR and graphical rendering is presented. Finally an overview of the necessary algorithms and techniques required by the overall relative position algorithms are discussed.

## *2.1 Mathematical Notation*

The following mathematical notation is used:

**Scalars:** Scalars are represented by a lowercase or uppercase letter in italics (e.g. $a, B$)

**Vectors:** Vectors are represented by a lowercase boldface letter (e.g., $\mathbf{a}$). A vector in a specific reference frame is denoted as $\mathbf{a}^B$, where vector $\mathbf{a}$ is in the $B$-frame. The transpose of a vector is represented by a vector with a superscript $T$ (e.g., $\mathbf{x}^T$). Unless a vector has a transpose, it is assumed to be vertical i.e. the number of columns is one. A list of vectors is represented by a vector with an subscript letter which represents the index of the element in the list (e.g., $\mathbf{a}_x$) where $x$ is the index of elements in $\mathbf{a}$. The total number of elements in the list is denoted as an uppercase N with a subscript of the list name (e.g., $N_a$). To index a vector the square brackets are used (e.g., $\mathbf{a}[0]$) where the index starts a zero.

**Matrices:** Matrices are represented by a uppercase boldface letter (e.g., $\mathbf{A}$). Matrices consist of scalar values in two dimensions referenced as $\mathbf{A}_{ij}$, where $i$ is the row index and $j$ is the column index. The identity matrix is denoted as $\mathbf{I}$.

**Direction Cosine Matrices:** Direction Cosine Matrices (DCM) that convert from the $a-frame$ to the $b-frame$ are denoted by $\mathbf{C}_a^b$

**Quaternion:** Quaternions are denoted a vector where the first value of the vector is the scalar quaternion value and the second through fourth is the vector value

for example $\mathbf{q}(w, x, y, z)$. Quaternions convert from $a-frame$ to $b-frame$ are denoted by $\mathbf{q}_a^b$. The identity quaternion is denoted as $\mathbf{q}^I = [1, 0, 0, 0]$

### *2.2 Reference Frames*

Navigation reference frames are important to express positions and orientation with respect to different orientations and with reference to different objects. For this document the following reference frames are defined:

**Earth-Centered Earth-Fixed ($e-frame$):** The Earth-Centered Earth-Fixed (ECEF) frame is an orthonormal basis in three dimensions with origin at the center of mass of the Earth. The ECEF frame is rigidly attached to the earth and moves and rotates with the earth. The $x$ axis points out toward the Greenwich meridian, and the $y$ axis points toward the 90 degree longitude line. This makes the $x-y$ plane lie in the equatorial plane of the Earth. The $z$ axis points toward the north pole. The ECEF frame is the frame the Global Positioning System (GPS) returns location in.

**Navigation Frame ($n'-frame$):** The navigation frame is an orthonormal basis in three dimensions, with origin located at a predetermined point on the vehicle. The $x, y, z$ axes of the navigation frame point North East Down NED respectively.

**Body Frame ($b-frame$):** the body frame is an orthonormal basis in three dimensions attached to a vehicle. The origin is located at a predetermined point on the vehicle co-located with the navigation frame. The $x, y, z$ axes of the body frame point out the nose, right wing, and bottom of an aircraft respectively. To denote different vehicles the name of the vehicle is added as a subscript to the frame. For example the tanker body frame is denoted as $b_{tanker}-frame$.

**Model Frame ($m-frame$):** the model frame is an orthonormal basis in three dimensions attached to a virtual representation of a vehicle. The origin of the model frame may not be in the same location as the body frame or navigation

frame of the vehicle it represents. The $x, y, z$ axes of the model frame point in the same directions as the body frame of that vehicle. As with the body frame, to denote different vehicles the name of the vehicle is added as a subscript to the frame. For example the tanker model frame is denoted as $m_{tanker}-frame$.

**LiDAR frame ($l-frame$):** the LiDAR frame is an orthonormal frame in three dimensions attached to the sensor origin of the LiDAR. The sensor origin is the point with zero range. The $x, y, z$ axes point in the forward, left and up directions of the LiDAR respectively.

**LiDAR measurement frame ($l'-frame$):** the LiDAR measurement frame is a spherical coordinate frame in range azimuth elevation $(r, \theta, \phi)$. The LiDAR measurement frame is the spherical representation of the LiDAR frame. Range is the slope distance to the measured point. Azimuth is the angle the slope line projected on the $x, y$ plane makes with the $x$ axis. Positive angles move in a counterclockwise motion and negative angles move in a clockwise motion. Elevation is the angle between the slope line and the line projected onto the $x, y$ plane. Positive elevation angles have a positive $z$ values. For simplicity angles greater than 180 will be treated as negative angles. Converting this frame to the $l\text{-}frame$ uses equation 2.1

$$\begin{bmatrix} x & y & z \end{bmatrix} = \begin{bmatrix} r\cos(\phi)\cos(\theta) & r\cos(\phi)\sin(\theta) & r\sin(\phi) \end{bmatrix} \tag{2.1}$$

**Virtual Camera Frame:** The virtual camera frame is an orthonormal frame used for cameras in virtual reality. In this frame the camera looks in the negative $z$ axis with $y$ axis as the up direction and $x$ axis as the right direction.

**World Coordinate Frame:** The world coordinate frame is an orthonormal frame used in virtual reality to place all objects contained in a scene. Objects in the scene include the camera, lights, and physical objects.

**Eye Coordinates Frame:** The eye coordinate frame is an orthonormal frame based on the world coordinate frame except the world coordinates frame is translated

and rotated so the world coordinates line up with the virtual camera frame from which the scene is viewed from.

**Pixel Coordinates:** Pixel coordinates are a 2D frame used to describe the layout of pixels on a computer screen. The upper left corner of the screen is the origin with positive $x$ axis moving from left to right across the screen and positive $y$ axis moving from top to the bottom of the screen.

### *2.3 Coordinate Transformation*

Coordinate transformations describe the relationship between the orientation of two different frames. A coordinate transform also converts a vector in one frame to another frame. This document has three different methods to express a coordinate transform, DCMs, Euler angles and quaternions.

DCMs consist of a 3x3 matrix of values. The values in the matrix relate to the dot product of each unit vector in one frame, with each unit vector in the second frame creating nine values [29]. The matrix allows for converting vectors in one frame to another through multiplication. For example, $\mathbf{v}^B = \mathbf{C}_A^B \mathbf{v}^A$ converts vector $\mathbf{v}$ from the $A-frame$ to the $B-frame$. Two DCMs can be combined to cut out the middle frame for example $\mathbf{C}_A^C = \mathbf{C}_B^C \mathbf{C}_A^B$ creates a conversion from the $A-frame$ to the $C-frame$ using the conversion from the $A-frame$ to the $B-frame$ and the conversion from the $B-frame$ to the $C-frame$. Unlike Euler angles there is no singularity in DCMs.

Euler angles consist of three values that represent three different single axis rotations applied one after the other. Euler angles are commonly used to transform the body frame of an aircraft from the navigation frame as an angle in yaw($\phi$), pitch($\theta$), roll($\psi$). Euler angles have singularities at certain rotations, in this example a singularity exists when the pitch angle is $\pm 90$ [29]. For navigation purposes the common order of rotations from the $n'-frame$ to the $b-frame$ is yaw pitch then roll. To

convert the Euler angles of such an order to a DCM the following equation is used:

$$\mathbf{A}_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.2}$$

$$\mathbf{A}_Y = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.3}$$

$$\mathbf{A}_X = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$\mathbf{C}_{n'}^{b} = \mathbf{A}_Z \mathbf{A}_Y \mathbf{A}_X \tag{2.5}$$

Quaternions consist of four scalar values divided into one three dimensional vector and one scalar. The scalar value is represented by a $w$ while the vector components are represented as $(x, y, z)$. A simplistic way to imagine a quaternion rotation is to think of it as a rotation vector where the quaternion vector value gives the axis of rotation, and the quaternion scalar decides the degree of rotation. Quaternions are often grouped as one vector with four values. The magnitude of a quaternion vector must equal one for the quaternion to be normal $\|[w, x, y, z]\| = 1$ [16]. Quaternions that are not normal do not provide correct transformations between coordinate frames. A quaternion can be converted to a DCM and then used to transform vectors. The conversion to a DCM is shown in Equation 2.6. For notation purposes, $\mathbf{R}(\mathbf{q})$ denotes the DCM of quaternion $\mathbf{q}$. Quaternions can convert vectors from one frame to another after conversion to a DCM [27]. Quaternions are used to simplify

mathematical operations and for storage simplicity.

$$\mathbf{R}(\mathbf{q}_A^B) = \mathbf{C}_A^B = \begin{bmatrix} w^2+x^2-y^2-z^2 & 2(xy-wz) & 2(xz+wy) \\ 2(xy+wz) & w^2-x^2+y^2-z^2 & 2(yz-wx) \\ 2(xz-wy) & 2(yz+wx) & w^2-x^2-y^2+z^2 \end{bmatrix} \quad (2.6)$$

Where the quaternion is $\mathbf{q}_A^B = (w, x, y, z)$

.

The coordinate transformation allows algorithms to operate in the most convenient and intuitive frame possible. Vectors can be easily transformed to any frame and the frames themselves can be transformed to any other frame.

### *2.4 LiDAR*

LiDAR is a technology used to sense objects at a distance. Mirrors direct a beam of light or laser toward a target. The sensor then detects the return of the light or laser and determines the time of flight in order to calculate the distance to the object [12]. The light or laser is pulsed in order to distinguish one beam from another [19]. The rotation of the mirrors determine the azimuth and elevation of the beam. A scanning LiDAR uses a rotating mirror to direct the beam in a pattern [13]. The pattern only depends on the methods used to steer the beam. The patterns can be in a grid of lines or even make circular passes. Another fundamental type of LiDAR uses commands from a user or control loop to drive the mirrors and direct the beam at an arbitrary azimuth and elevation. For example, the Triangulation + LiDAR (TriDAR) has six degrees of freedom to direct the beam in any direction [22]. Both types of LiDAR require returns off the scanned object in order to calculate the distance to the object. The stronger the return the more accurate the range. This makes scanning rough, coarse objects ideal while shiny, smooth objects may pose a problem [3]. A mirror-like object will direct the beam away and not return the beam to the sensor. This may cause a missed return or a false reading if the beam returns to

the sensor after striking another object, thus increasing the perceived range. Another problem can be when the laser strikes the edge of an object. When sensing the edge of an object, the scan can wrap around the object and create a false return [3]. A LiDAR normally returns measurements in spherical coordinates (range, azimuth, and elevation) which can be converted to a cartesian coordinate system for analysis.

### *2.5 LiDAR Simulation with Graphics Rendering*

LiDAR simulation is necessary to perform early analysis of algorithms that use LiDAR measurements [20]. Powell *et al.* [21] have shown LiDAR simulation with commercial ray tracing software is possible while Peinecke *et al.* [20] have simulated a LiDAR with graphics hardware acceleration. This background explores hardware acceleration as the method of choice. With the modern Graphics Processing Units (GPUs) or graphics cards, parallel floating point operations are possible using a pipeline approach. The pipeline consists of sets of dedicated hardware that perform the same operation in parallel. Newer graphics cards allow for custom programming of steps in the pipeline. This allows users to take advantage of this parallel processing for other uses besides normal rendering. The two common locations in the pipeline where custom code can be written are the vertex and fragment shader. The vertex shader carries out operations on a per vertex basis such as transforming vertexes to different frames. The fragment shader operates on a per pixel basis and uses the output of the vertex shader. Each pixel is derived from an object and contains the color of the object, as well as range to the object. Also at this point in the pipeline it is possible to transform the world coordinates of the scene into the eye coordinates frame. This frame has a fixed rotation to the $l-frame$ making conversion in the pipeline possible. A simple fixed DMC can transform the eye coordinates frame to the $l-frame$. Thus, the distance from the origin to the object is quickly computed. This allows the actual color of the object to be replaced by a scaling based on distance. After the final rendering of the object, the location of the pixel will give the azimuth and elevation, while the color will give the range. Thus, the final result can

be interpreted as a simulated LiDAR scan of the entire object over all azimuths and elevations. This end result can be leveraged to quickly simulate either a scanning or controlled LiDAR, depending on a mask applied to the picture [20].

### *2.6 Iterative Closest Point*

The Iterative Closest Point (ICP) algorithm allows for the registration of one point cloud to another [8]. ICP determines the optimal rotation and translation to apply to a point cloud in order to minimize distance to the closest point in another point cloud. The two clouds are denoted as the $\mathbf{x}_i$ point cloud and the $\mathbf{p}_i$ point cloud. The set $\mathbf{p}_i$ and $\mathbf{x}_i$ must have equal size and each element in $\mathbf{x}_i$ with index $i$ corresponds to $\mathbf{p}_i$ with the same $i$ index. The objective is to rotate and translate the $\mathbf{p}_i$ to more closely match $\mathbf{x}_i$. To accomplish this the algorithm must minimizes the equation 2.7.

$$f(\mathbf{q}, \mathbf{t}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|\mathbf{x}_i - \mathbf{R}(\mathbf{q})\mathbf{p}_i - \mathbf{t}\|^2 \tag{2.7}$$

where $\mathbf{q}$ is the quaternion to apply as a DCM to point cloud $\mathbf{p}_i$, $\mathbf{t}$ is the translation to apply to $\mathbf{p}_i$, and $N_p$ is the number of points in each set $\mathbf{x}_i$ and $\mathbf{p}_i$. The algorithm operates in iterations and each iteration brings set $\mathbf{p}_i$ closer to set $\mathbf{x}_i$. One iteration of the ICP algorithm first determines the mean of both sets by

$$\boldsymbol{\mu}_x = \frac{1}{N_x} \sum_{i=1}^{N_x} \mathbf{x}_i \text{ and } \boldsymbol{\mu}_p = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{p}_i \tag{2.8}$$

where $\boldsymbol{\mu}_x$ and $\boldsymbol{\mu}_p$ are the means of the point clouds $\mathbf{x}_i$ and $\mathbf{p}_i$ respectively. Next the algorithm determines the covariance of the sets by

$$\boldsymbol{\Sigma}_{px} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left(\mathbf{p}_i(\mathbf{x}_i)^T\right) - \boldsymbol{\mu}_x\boldsymbol{\mu}_p \tag{2.9}$$

where $\mathbf{\Sigma}_{px}$ is the covariance between point clouds $\mathbf{x}_i$ and $\mathbf{p}_i$. Next a matrix $\mathbf{Q}$ is made from $\mathbf{\Sigma}_{px}$. To simplify the notation an auxiliary matrix is created $\mathbf{A} = \mathbf{\Sigma}_{px} - \mathbf{\Sigma}_{px}^T$. Next an auxiliary vector is created $\boldsymbol{\delta} = [\mathbf{A}_{23}\mathbf{A}_{31}\mathbf{A}_{12}]^T$. Finally we use $\boldsymbol{\delta}$ to make $\mathbf{Q}$:

$$\mathbf{Q}(\mathbf{\Sigma}_{px}) = \begin{bmatrix} trace(\mathbf{\Sigma}_{px}) & \boldsymbol{\delta}^T \\ \boldsymbol{\delta} & \mathbf{\Sigma}_{px} + \mathbf{\Sigma}_{px}^T - trace(\mathbf{\Sigma}_{px})\mathbf{I} \end{bmatrix} \tag{2.10}$$

where $trace()$ is the trace of the matrix in parentheses. To obtain the optimal rotation as a quaternion, the eigenvector $\mathbf{q}_x$ corresponding to the largest eigenvalue of the matrix $\mathbf{Q}$ is selected. The optimal translation is the difference between the means of the sets corrected for the rotation as shown in Equation 2.11.

$$\mathbf{t}_x = \boldsymbol{\mu}_x - \mathbf{R}(\mathbf{q}_x)\boldsymbol{\mu}_p \tag{2.11}$$

The rotation and translation are then applied to the set $\mathbf{p}_i$. The new set $\mathbf{p}_i$ is then used to start a new iteration of the algorithm. Each time the number of points in the set $\mathbf{p}_i$ must be equal and correspond to the points in set $\mathbf{x}_i$. Correspondence can be determined by different metrics such as the closest point in set $\mathbf{p}_i$ to the point in set $\mathbf{x}_i$, or by using metadata from each point such as color [14].

Iteration continues until a threshold is reached. Different thresholds can be used to terminate the algorithm. The matrix $\mathbf{\Sigma}_{px}$ relates how close of a fit each set is to the other by covariance [8]. Converting $\mathbf{\Sigma}_{px}$ into a cross correlation matrix $\mathbf{P}_{px}$ and tanking the trace, gives a single value $\rho_{px}$ to determine closeness of fit. Thresholds of $\rho_{px}$ to indicate close fits are based on the underlying geometry of the sets and can be used to determine when the sets are adequately registered. Alternatively, the translation distance $\mathbf{t}_x$ can be used to determine when the current amount of correction is so small the new $\mathbf{t}_x$ is not worth the time taken to calculate [33]. Again the translation distance is dependent on the sets being registered. The simplest threshold is to use a max number of iterations. The max iterations can be derived empirically by observing when the sets are adequately registered for the specific cause,

or determined based on a time limit. The threshold can also be any combination of these methods. Regardless of the threshold used, the ICP algorithm determines rotations and translations to move point clouds so they are more closely registered.

The ICP algorithm is not prefect however. While convergence on a solution is guaranteed, convergence on the correct solution is not [8]. If the registration between the sets is initially far off, a local minimum could be reached where the sets are not correctly registered, but an infinite number of iterations will not move closer to the true solution. The shape of the sets can pose a problem for accurate registration. For example, a spherical object is difficult to register correctly, because all rotations about the correct center will appear similar. To solve this problem, at the final solution $\rho_{px}$ can be examined to determine if it appears to be a local minimum. If $\rho_{px}$ exceeds thresholds, the ICP algorithm must be started from different orientations in order to find the correct registration. The threshold values for $\rho_{px}$ depend on the geometry of the sets being registered.

### *2.7 k-D Tree*

A k-Dimensional (k-D) tree is a data structure used to spatially store a finite number of K dimensional variables. k-D trees are spatially sorted in order to perform certain algorithms quickly, such as the nearest neighbor search [7]. In order to be spatially sorted, a k-D tree starts with a set of variables with K dimensions. The variables are sorted along the first dimension using some metric. Next, a dividing point is chosen using some method such as the median. A node is made using the variable closest to this dividing point. Next the set is divided into two new sets corresponding to the variables above and below the dividing node. The new sets are sorted using the next dimension. The dimension will wrap back to the first dimension if no new dimension exists. This process is repeated until the number of variables in a node falls below a maximum [6]. Now the variables are spatially sorted. To search for a given node, the search would start at the root node and compare to the search value. If greater it would move to the top child node, and conversely if less move

to the lower child node. By repeating this pattern the search value can be quickly found. This method can be used to find an approximate nearest neighbor to a point in three dimensional space. This is approximate because in some cases if the two child nodes are very close the closer node in the current dimension may not contain the true nearest neighbor. In such cases both the near child and the far child nodes must be searched in order to determine a more accurate nearest neighbor. However in some cases, the threshold used to determine when the children are too close may fail and the true nearest neighbor missed. Despite this limitation this search method is much quicker than a brute force method of examining every variable one by one. Also with appropriate thresholds the incorrect nearest neighbors will be very close to the true nearest neighbor [10].

### *2.8 Reporting Errors*

Reporting error in a concise manner is vital to compare and understand the algorithms presented in this paper. A brief explanation of the error terms used in this thesis is provided.

**RMS Error** : Root Mean Square(RMS) error is a method to combine the mean and standard deviation of an error into one value. In this case the mean represents a bias from the true value. RMS error is useful for one dimensional values such as a range or distance in one axis. RMS is calculated with the following equation $RMS = \sqrt{\frac{\sum_{i=1}^{N_x} x_i^2}{N_x}}$ where $x_i$ is a set of values that represent the error from a true value, and $N_x$ is number of values.

**MRSE** : Mean Radial Spherical Error (MRSE) is a method to combine the mean and standard deviation of a position error from two sets of points into one value. This is useful to represent a 3D position error which has three means, three standard deviation, and three RMS values for each axis into one value to quickly compare to other 3D position errors. MRSE is calculated with the following equation

$MRSE = \sqrt{\frac{\sum_{i=1}^{N_p} \left(x_i^2 + y_i^2 + z_i^2\right)}{N_p}}$ where $x_i, y_i, z_i$ represent element $i$ of the error in each axis from two points in 3D space, and $N_p$ is number of points.

# III. Algorithms and Analysis

This chapter outlines the concepts to determine relative position from the LiDAR sensor measurements and how to boresight the LiDAR. The chapter is organized as follows. First the relevant assumptions for both algorithms will be outlined. Next the Model Based ICP (MBI) algorithm that uses the measurements converted into cartesian space will be outlined. Next the Position Perturbations Difference (PPD) algorithm that keeps the measurements in spherical coordinates will be outlined. Both algorithms will be explained as a step by step process in order to use the last estimate of position to determine the new position estimate. Finally a method to boresight a LiDAR is presented. The method includes the mathematics and procedure used to boresight the LiDAR. An error analysis of the boresight procedure will also be explained and compared to an actual boresight completed for this thesis.

## *3.1 Algorithm Assumptions*

Both algorithms use the same underlying assumptions in order to determine the relative position of the air vehicles. The list of assumptions is as follows:

**Inertial Measurement Units** Both of the air vehicles contain accurate navigational grade Inertial Measurement Units (IMUs), that during the time of the algorithms have an accurate measurement of the attitude of the air vehicles. The relative orientation between the two aircraft is assumed to be the orientation described by the IMUs.

**Initial Position Estimate** The initial estimate of the relative position is known within a statistical accuracy.

**Sensor Setup** The LiDAR is positioned on the receiver facing the tanker aircraft so that during refueling, the tanker is in the field of view of the LiDAR. The position and orientation of the LiDAR with respect to the receiver aircraft is known.

**Measurement Timing** The LiDAR scans are assumed to be taken in one time epoch. All measurements from the LiDAR are assumed to be taken at the start time of the scan. This assumption is accurate in low relative dynamic situations such as AAR.

**Tanker Model** An accurate model of the tanker is available as a mesh object defined by vertices and triangles. The origin of this model can be related to the actual tanker in some orientation and position.

**Relative Position** The $n' - frame$ orientation of each plane is nearly equal due to the close proximity of the aircraft to each other.

## *3.2 MBI Algorithm*

In this section the fundamental steps of the Model Based ICP (MBI) Algorithm are stated and defined. The algorithm consists of the following steps shown in Figure 3.1:

1. Build Data Structures
2. Find Closest Points on Model to Measurements
3. Determine Position with Modified ICP
4. Determine if Threshold is Reached
5. Apply Sensor Position

*3.2.1 Build Data Structures.* First the tanker model data set is defined. The model consists of two distinct sets. The first set $\mathbf{v}_i$ consists of every vertex of the model and can be thought of as a point cloud for the tanker model, where $i$ is the index of each vertex. Each vertex is represented as a set of three floating point numbers for the $x, y, z$ axes. The model also consists of a list of triangles $\mathbf{t}_i$ to develop the mesh for the model. Each triangle consists of three integer numbers that represent the index $i$ of a vertex in $\mathbf{v}_i$.

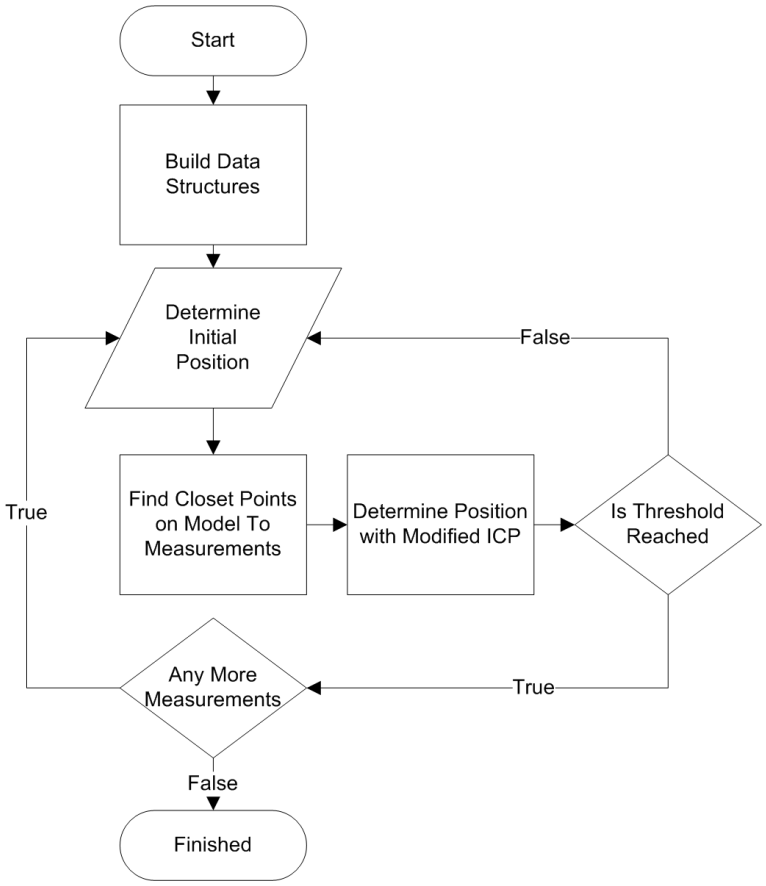


Figure 3.1: Flowchart showing the steps involved in MBI algorithm.

The second data structure is a k-D tree. The k-D tree (KDT) consists of the set of vertices $\mathbf{v}_i$ from the model. The closest vertex $\mathbf{v}_k$ in the KDT from an arbitrary point $\mathbf{x}$ is denoted as $\mathbf{v}_k = \text{KDT.NN}(\mathbf{x})$, where NN stands for the Euclidean distance nearest neighbor function. The third data structure is a key value pair dictionary, referred to as DCT, that relates each vertex to the triangles that use the vertex. For example, assume vertex $\mathbf{v}_k$ is used in three triangles $\mathbf{t}_{1-3}$. Thus for the entry of $\mathbf{v}_k$ it would return all the vertices of the three triangles. In this example, there would be nine three value vectors. To obtain the values we poll the DCT as such $DCT[\mathbf{v}_k] = \mathbf{t}_{\mathbf{v}i}$ where $i$ is the index of the vertices of the triangles. In this example the max value of $i$ is nine. Each successive three values consists of one triangle. Thus $\mathbf{t}_{\mathbf{v}1-3}$ is the three vertices of the first triangle $\mathbf{t}_1$.

*3.2.2 Find Closest Points on Model to Measurements.* The first step of the MBI algorithm is to determine the closest point on the tanker model for each of the measurements $\mathbf{x}_i^{l'}$. The KDT uses the nearest neighbor function to determine the closest vertex in the model for each measurement. However the points stored in the KDT are in the $m_{tanker}-frame$. Since there are many vertices in the KDT it is faster to transform the measurements from the $l-frame$ to the $m_{tanker}-frame$ first then query the KDT with the transformed points. Next the vertex returned from the KDT is used as a starting point to determine the closest point on the triangle list of the model.

*3.2.2.1 Orientation.* To determine the closest points to measurements the measurements must be converted from the $l'-frame$ to the $m-frame$ of the tanker. First the conversion from the $l'-frame$ to the $l-frame$ is calculated, as described in Section 2.2, to obtain measurements $\mathbf{x}_i^l$. Next the attitude measurements from the IMUs of each aircraft are combined into a relative DCM. To accomplish this, the Euler angles for the tanker and receiver are used to generate $\mathbf{C}_{n'}^{b_{tanker}}$ and $\mathbf{C}_{n'}^{b_{receiver}}$ respectively. The two DCMs are combined with $\mathbf{C}_l^{b_{receiver}}$ (obtained from extrinsic calibration described in Section 3.4) and $\mathbf{C}_{b_{tanker}}^{m_{tanker}}$ (determined during model creation) using 3.1

$$\mathbf{C}_l^{m_{tanker}} = \mathbf{C}_{b_{tanker}}^{m_{tanker}} \mathbf{C}_{n'}^{b_{tanker}} \mathbf{C}_{n'}^{b_{receiver}{}^T} \mathbf{C}_l^{b_{receiver}} \tag{3.1}$$

With this DCM each measurement from the $l-frame$ can be converted to the $m_{tanker}-frame$ as shown $\mathbf{x}_i^{m_{tanker}} = \mathbf{C}_l^{m_{tanker}} \mathbf{x}_i^l$.

*3.2.2.2 KDT and Triangle Searching.* Next the KDT is used to retrieve the closest vertex of each point $\mathbf{q}_i^{m_{tanker}} = KDT.NN(\mathbf{x}_i^{mtanker})$. However this point is only the closest vertex to each measurement point as shown in Figure 3.2. To improve accuracy the closest point on the mesh to the measurement must be calculated. To acquire the closest point on the mesh, a search of the triangles near the closest vertex is conducted. All the triangles that include the vertex returned from

the KDT are searched, because any of the triangles could contain the closest point to the measurement point. The DCT is queried to return all the triangles that use the point returned from the KDT as a vertex as shown in Figure 3.3. To determine where on the triangle the closest point is to the measurement, the following function is used $[\mathbf{y}, f] = ClosestPointTriangle(\mathbf{x}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ [9] where $\mathbf{x}$ is the measurement point, $\mathbf{y}$ is the point that lies on the triangle, and $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are the three vertices of a triangle returned from the DCT. The value $f$ flags where the point $\mathbf{y}$ is located on the triangle. This flag represents if the point is inside or on one of the edges of the triangles denoted by a one and zero respectively. This function is iterated over all the triangles that use the vertex returned from KDT.

$$[\mathbf{y}_k, f_k] = ClosestPointTriangle(\mathbf{x}_i, \mathbf{t}_{3k+1}, \mathbf{t}_{3k+2}, \mathbf{t}_{3k+3}) \tag{3.2}$$

where $\mathbf{t} = DCT[\mathbf{q}_i]$ and $k$ is in the range of $0 - (N_\mathbf{t}/3 - 1)$ where $N_\mathbf{t}$ is the number of points in $\mathbf{t}$. With this a list of the closest points on each triangle $\mathbf{y}_k$ and their locations on the triangle $f_k$ is created. Next the closest point to $\mathbf{x}_i$ in $\mathbf{y}_k$ is found with the equation $d_k = \|(\mathbf{x}_i - \mathbf{y}_k)\|$. The closest point is denoted as $\mathbf{y}_b$. However if the flag $f_b$ for the closest point $\mathbf{y}_b$ is on the edge of the triangle, then it is possible a closer point exists on a triangle not in the current list of triangles. If the closest point is on the edge, the previous algorithm must start over with the other two vertices of the triangle that contained the current closest point as shown in Figure 3.4. This gives two more lists of $\mathbf{y}_k$. These new lists are added to the previous list and a new $\mathbf{y}_b$ is calculated. This process is repeated until $\mathbf{y}_b$ does not lie on the edge of a triangle. The end result is a set of points in the set $\mathbf{p}_i^{m_{tanker}}$ which contains the closest point to the measurement point $\mathbf{x}_i^{m_{tanker}}$ after searching across triangles of the model mesh.

*3.2.3 Determine Position with modified ICP.* After the two point clouds $\mathbf{p}_i^{m_{tanker}}$ and $\mathbf{x}_i^{m_{tanker}}$ are collected, the best registration is determined. First each point cloud is converted from the $m_{tanker}-frame$ to the $b_{receiver}-frame$ so the result will be relative position with respect to the receiver aircraft. Next, to align the two

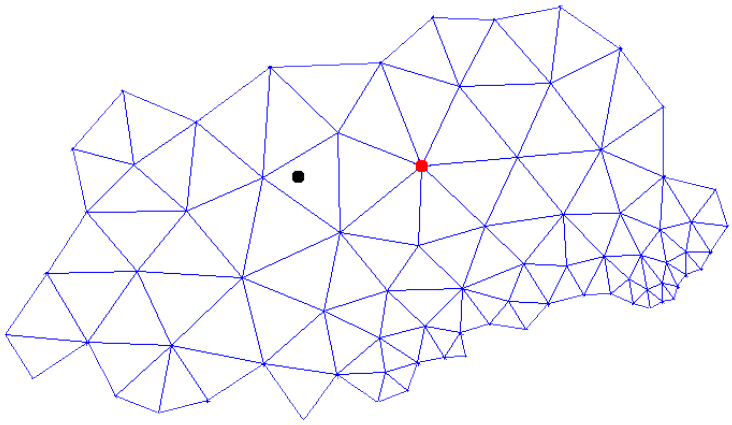

Figure 3.2: The black point represents the measurement point that the algorithm is trying to find the nearest point on the mesh for. The red point is returned from querying the k-D Tree as to the nearest neighbor to the black point.

sets, a modified ICP algorithm is used. In typical ICP a best rotation and translation is determined. However with the navigation grade IMUs, the best rotation is already known. Thus, only the best translation is required from the algorithm. To accomplish this Equation 2.9 and 2.10 are skipped and only Equations 2.8 and 2.11 are used, with the assumption that $\mathbf{R}(\mathbf{q}_x) = \mathbf{I}$. This calculated translation is added to the overall translation $\mathbf{t}_t^{m_{tanker}}$. The overall translation is used to determine the final translation from the original estimated position.

*3.2.4 Determine if Threshold is Reached.* The estimate of position incrementally converges to one solution with each iteration of the algorithm. How many iterations to calculate depends on the accuracy required, the shape of the tanker model, and the measurements from the LiDAR. The algorithm in this thesis uses a combination of two thresholds to determine the overall threshold. The first threshold sets a maximum number of iterations derived empirically from a nominal data set. The second threshold is the last incremental corrective distance, also derived empirically from a nominal data set. The dual threshold allows for an upper limit to constrain the iterations while allowing an early finish if the match is recognized quickly.

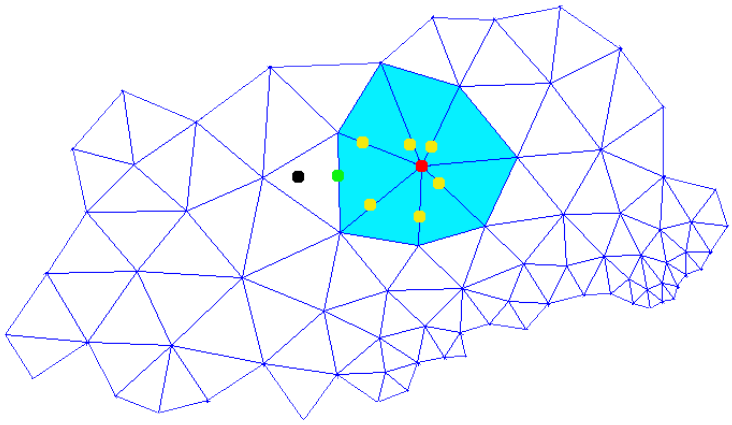

Figure 3.3: The triangles with the red point as a vertex are determined from the triangle dictionary DCT shown as the cyan triangles. The closest point on each of those triangles to the black point is determined to be the yellow points with the green point being the closest. The green point lies on an edge of the triangle thus more searching is required.

*3.2.5 Apply Sensor Position.* After the threshold is reached, the algorithm has obtained the final position estimate $\mathbf{t}_t^{mtanker}$. However the final position estimate is in the $m_{tanker}-frame$ and must be converted the $b_{receiver}-frame$ before it can be useful. For this equation 3.3 is used

$$\mathbf{t}_f^{b_{receiver}} = \mathbf{C}_{m_{tanker}}^{b_{receiver}} \mathbf{t}_t^{m_{tanker}} + \mathbf{l}_o^{b_{receiver}} \tag{3.3}$$

where $\mathbf{l}_o^{b_{receiver}}$ is the translation between the origin of the $l'-frame$ and the $b_{receiver}-frame$ in the $b_{receiver}-frame$.

## *3.3 PPD Algorithm*

In this section the fundamental steps of the Positions Perturbations Difference (PPD) algorithm are stated and defined. The algorithm consists of the following steps as shown in Figure 3.5:

1. Perturb Position
2. Predict Measurements

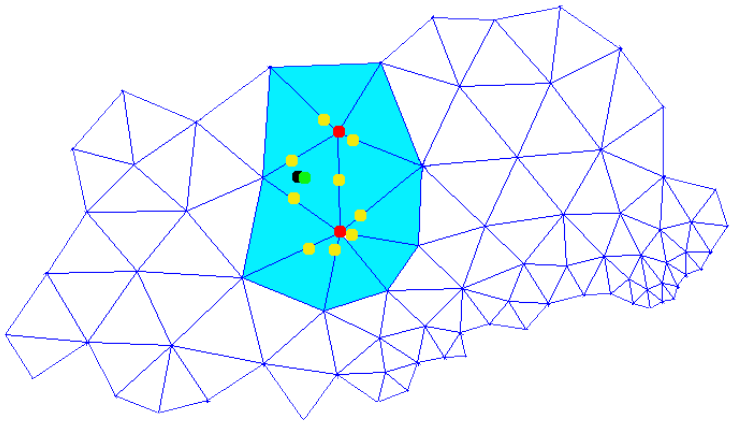

Figure 3.4: Two new red points are determined as the other vertices of the triangle that contained the previous green point. These new red vertices are used to make a new list of cyan colored triangles. Again the closest point from each of these triangles to the black point is calculated. In this case the green point is not on an edge so it is returned as the closest point on the mesh to the black point.

3. Compare Predicted and Actual Measurements

4. Determine Best Position

This algorithm, unlike the MBI algorithm, is not constrained to a specific frame. This allows the calculation to be estimated in the desired frame from the beginning thus reducing coordinate transformations.

*3.3.1 Perturb Position.* The method to perturb the position can be based on a number of different criteria depending on the situation, from gradient slopes to random guesses. This thesis implements an algorithm that searches a box volume with increasing precision to converge to a best position. There are four factors that shape the search space–the box length $b$, divisions of the box $d$, the number of times to shrink the box $c$, and how much to shrink the box $s$. From these factors a search can be conducted. First a box of length $b$ is centered at the starting point, usually the last position estimate. The box is divided $d$ times in each direction. For example, in a division of 2 there are 3 segments in each axis making 2-d slices of 9 squares for a total of 27 new boxes. Each box center is a possible candidate for the position of the tanker and each has its comparison metric calculated. The position with the best

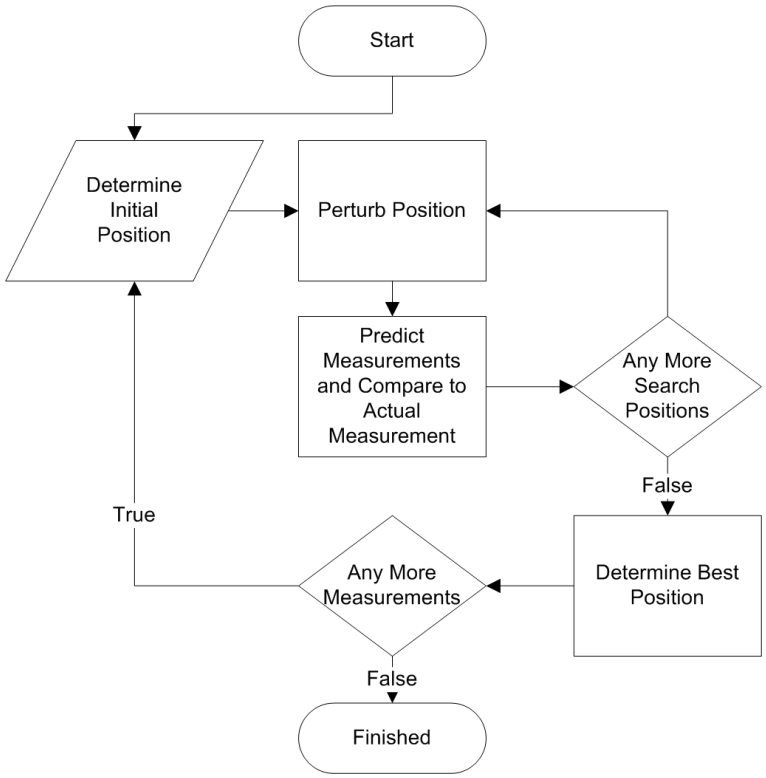


Figure 3.5: Flowchart showing the steps involved in PPD algorithm.

metric is determined as the center for a new box. This new box has length equal to the length of the boxes that the original box was divided into so $b_n = s(b/(d+1))$ where $b_n$ is the new box length and $b$ is the previous box length. The box is enlarged by the factor of $s$ to account for the optimal spot being outside this box but inside the spots searched from other boxes. This new box is once again divided and searched. This process repeats for $c$ times. The position with the lowest metric at the end of all the perturbations is declared the best position. This method has the advantage of spanning a fixed number of perturbations thus forcing a time limit for each pose.

*3.3.2 Predict Measurements.* When the tanker is perturbed, the measurements in that position are simulated. Many different methods can be used to simulate measurements. This paper uses a predictive rendering approach to quickly simulate range measurements from the tanker model. The tanker model is loaded into a graphics engine capable of using custom programmed code in the graphics pipeline. The

fragment shader is overridden with custom code to output not the color of the model, but rather a range color scheme. The function requires a maximum and minimum distance to scale the color values to. This value is in the same units as the measurements and a typical maximum value would be the maximum range of the LiDAR and a typical minimum would be zero. The function then scales the distance to the pixel being rendered using this color scale. The scale uses a rainbow scale with purple as close and red for far to increase human readability. The function then outputs this color scale instead of the usual object color. Thus the screen renders the tanker model in a color range scale as shown in Figure 3.6. With the pixel coordinates of the pixel, the camera location, and camera orientation, and camera projection matrices, the azimuth and elevation to each pixel can be determined.

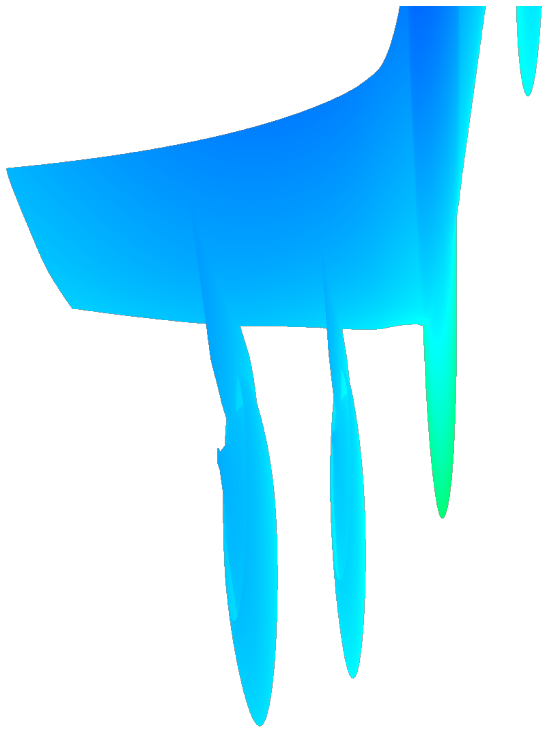

Figure 3.6: Renders of the tanker with its color representing the range from the virtual camera or in this case the virtual LiDAR. The minimum and maximum range of the LiDAR provide a scaling to which colors can be applied. In this case the nose of the fuselage is farther away and turning a green while the tail is closer and turning a dark blue.

To increase speed only the points with desired azimuths and elevations are converted to a point cloud. This algorithm predefines the valid pixels based on the azimuth and elevation range of the LiDAR that creates the LiDAR scan. This set of elevations and azimuths is used to create a mask defining the pixels to be converted into a point cloud as shown in Figure 3.7. To determine this mask first define a set of valid azimuth elevation pairs $\mathbf{v_{ea}}$ where the first and second values are the elevation and azimuth respectively, next a set of valid pixels $\mathbf{v_p}$ can be derived using equation 3.4

$$
\begin{aligned}
f_{pdx} &= \frac{N}{2}\mathbf{P}[0,0] && (3.4)\\
f_{pdy} &= \frac{M}{2}\mathbf{P}[1,1] && (3.5)\\
\mathbf{v_{p}}_i[1] &= round\left(\frac{N}{2} - \tan\left(\mathbf{v_{ea}}_i[2]\right) f_{pdx}\right) && (3.6)\\
v_{py} &= round\left(\frac{N}{2} - \tan\left(\mathbf{v_{ea}}_i[2]\right) f_{pdy}\right) && (3.7)\\
\mathbf{v_{p}}_i[2] &= round\left(\frac{M}{2} - \tan\left(\mathbf{v_{ea}}_i[1]\right)\sqrt{{f_{pdy}}^2 + \left(v_{py} - \frac{N}{2}\right)^2}\right) && (3.8)
\end{aligned}
$$

where $N$ is the number of pixels horizontally, $M$ is the number of pixels vertically, and $\mathbf{P}$ is the 4x4 projection matrix of the virtual camera. The pixels are in pixel coordinates where the leftmost, rightmost, top, and bottom pixels have values of 0,$N$,$M$, and 0 respectively. The valid pixel pairs are then used to pick off the correct pixels in the rendered picture and pair them with the corresponding elevation and azimuth as shown in Figure 3.8.

*3.3.3 Compare Simulated and Actual Measurements.* The simulated measurements must be compared to the actual measurements to derive some metric of how close the current position estimate is. This algorithm pairs measurements with the exact same elevation and azimuth, since for the given prediction of scans it is possible that no measurement exists for every elevation and azimuth. Measurements

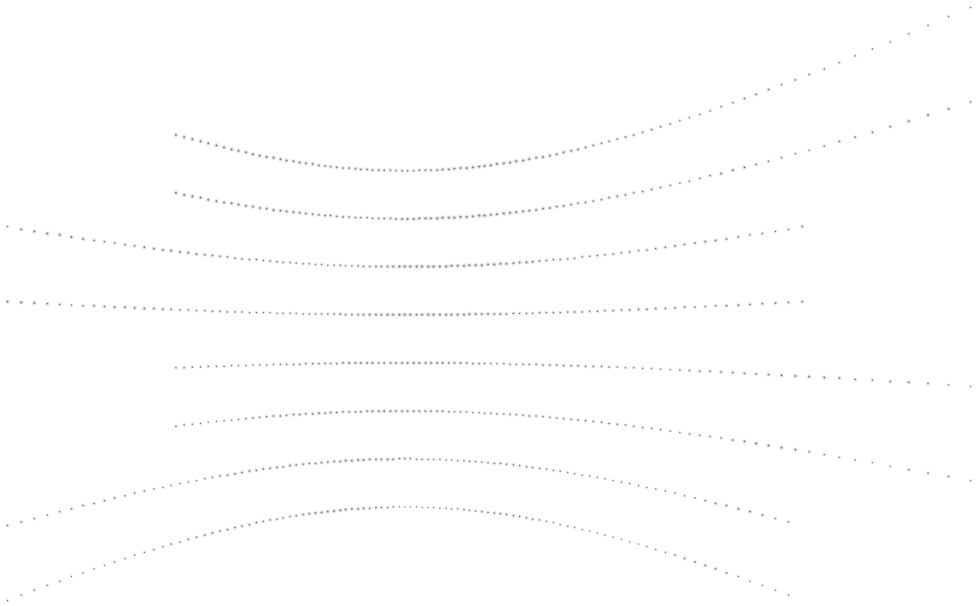

Figure 3.7: Pixel mask for each of the azimuth and elevation pairs. This mask is for the Ibeo LUX 8L. The lines of constant elevation curve as described in Section 3.4.1.3.

that are not paired are thrown out and not used in the metric. The RMS error between the ranges of paired measurements is computed. This thesis refers to this error as RMS measurement Range Error (RRE). In order to reject predictions with few matching measurements, but by chance very small RRE, the error $e_{RRE}$ is inflated by the ratio of the number of paired measurements to the number of total measurements from the LiDAR as shown $e_{SRRE} = e_{RRE}\frac{N_{measured}}{N_{paired}}$. This is called the Scaled RMS measurement Range Error (SRRE). This method promotes minimizing RRE as well as pairing measurements.

*3.3.4 Determine Best Position.* After the tanker has been perturbed to many positions and the SRRE compared for those many positions, a best position must be declared. This thesis chooses the position of the lowest SRRE. Other methods that interpolate between measurements of the lowest SRRE could be used, however

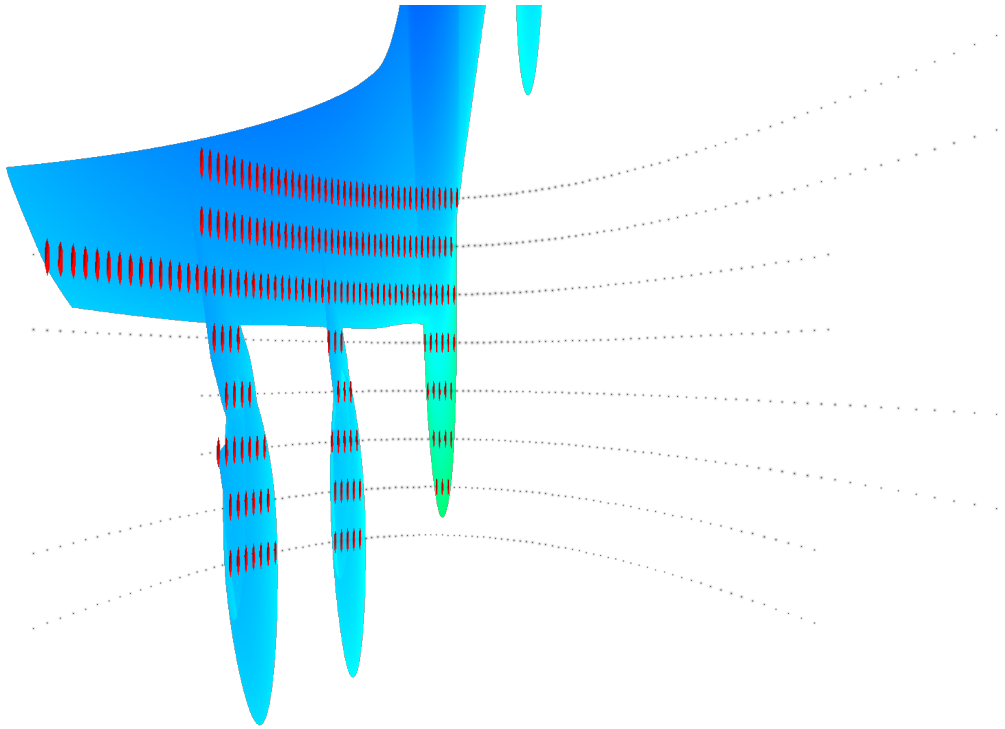

Figure 3.8: Mask applied to rendered image returning beams. In this picture each returned beam is represented by a red sphere. The pixels at each of these locations provide the range through their color and azimuth and elevation by their location in the image. Since a mask was used, the pairing between azimuth-elevation pairs and pixel coordinates can be used to quickly relate pixel coordinates to an azimuth and elevation.

this thesis chooses the simplest method to demonstrate the capability to track based on prediction.

### *3.4 LiDAR Boresight Technique*

This section outlines a method to boresight a LiDAR. This particular method was developed in order to determine the position and orientation of the LiDAR $l-frame$ in an arbitrary frame without a camera or exact range measurements from the LiDAR. First an overview of boresighting and other boresight methods are outlined. Next the mathematics and procedure to boresight the LiDAR are presented. Finally an error analysis for the boresight method is discussed and compared to a boresight completed for the test flights discussed in Section 4.1.

*3.4.1 Overview.* A boresight of a sensor determines the intrinsic sensor frame relative to another extrinsic frame. For example to boresight a rifle the intrinsic direction of fire is lined up with the extrinsic sight of the gun. To determine the intrinsic sensor frame, the sensor data must be used. In the example of the rifle, the direction of fire is determined by firing the gun and recording where the bullet strikes. To boresight a LiDAR one must determine where the beams strike external objects. This method records where beams strike a flag board. Unlike a rifle, a scanning LiDAR has many beams which can be used to determine the boresight of the LiDAR. This method determines where many beams strike a flag board in order to boresight the LiDAR.

*3.4.1.1 Previous Work.* Previous work on boresighting a LiDAR has focused on determining the relative orientation between a camera and LiDAR. Willis et. all [32] used a camera and single line scanning LiDAR to achieve a relative boresight between the camera and LiDAR. They used a zigzag wall pattern that changed color at each elbow to relate color changes to range changes. Pandey et. all [19] coordinated an omnidirectional camera and multilevel scanning LiDAR using checkerboard patterns placed around the sensors. Once again a relative position between the omnidirectional camera and LiDAR was the objective. This type of boresight is not useful for the purpose of using a stand alone LiDAR to estimate position, because neither the camera or LiDAR frame is the desired frame of reference to use. The camera and LiDAR also must be able to observe the same objects in many different orientations in order to determine an accurate relative orientation. Thus a method that references the LiDAR orientation in an arbitrary frame is required to relate the LiDAR to a more useful real world frame of reference. This method also does not rely on precise range measurements from the LiDAR thus mitigating range calibration error.

*3.4.1.2 Problem.* In order to transform the measurements from the LiDAR into other reference frames, the sensor orientation and sensor origin are re-

quired. To obtain the origin and orientation, measurements from the LiDAR must be used since diagrams or measurements of the sensor are imprecise and contain error because each sensor is different. In this method, a surveying laser system is used to precisely pinpoint surveyed points within millimeter accuracy. A graphical utility and flag board outlined in the Section 3.4.3 are used to provide a target for the surveying laser system. Only scan lines of low elevation are collected for reasons explained in Section 3.4.2.

*3.4.1.3 Scan Characteristics.* The theory to boresight the LiDAR involves describing where many beams strike know locations. First the pattern of the beams must be understood. LiDAR beams that scan with constant azimuth will trace a vertical line on a plane. Beams that scan with constant elevation however, will not trace a straight horizontal line. Beams of constant elevation draw a parabola on a plane. This occurs because a beam of constant elevation traces out a cone as it scans, and a cone intersecting a plane creates a parabola. Thus, to characterize beams of constant elevation, the elevation must be small or else the parabolic effect will curve the line. In this boresight method parabolic lines cannot be used to determine the boresight, thus only beams of low elevation will be used to determine the boresight.

*3.4.2 Mathematics to Determine Boresight.* First a point cloud of strike points is obtained in a frame other than the $l-frame$ as shown in Figure 3.9. This point cloud consists of points where beams strike a flag board at varying ranges. This point cloud is called the $\mathbf{p}_x$ where $x$ is the element index of the point cloud. The origin and orientation of the sensor are calculated using this point cloud. To determine the origin, strike points of each beam are fit to a line. Points of the same beam are denoted as $\mathbf{b}_x$ where $x$ is the element index of the point cloud. Each beam is described by a point of intersection $\mathbf{c}$ and the slope $\mathbf{s}$ of the line. The equation for the center is the average of the points of a beam.

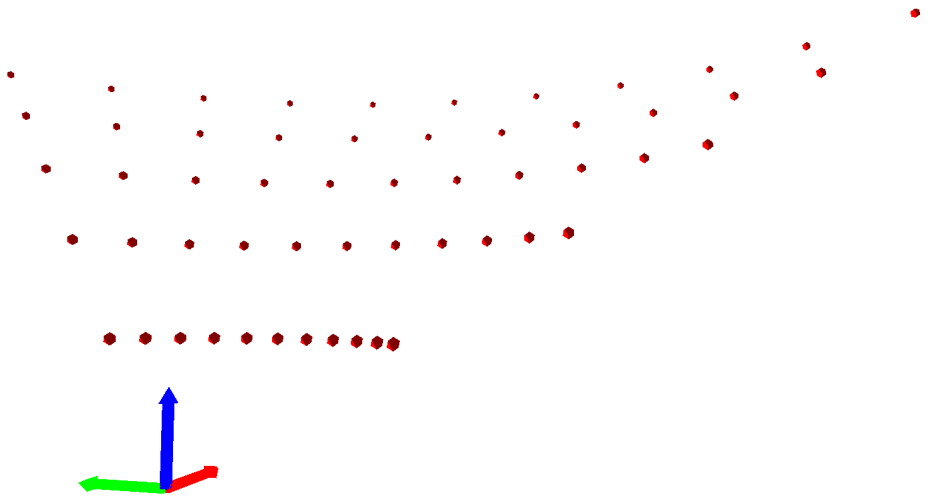

Figure 3.9: LiDAR boresight simulated scan showing beams for azimuth lines from -50 to 50 degrees in 10 degree increments. The elevation of all lines is 30 degrees to show the geometry principles but normally would not be used for boresighting. Each red sphere represents a scanned point from the LiDAR that struck the flag board at a different range. The axis is the $l-frame$ from which the measurements were taken and has been included in all pictures to provide clarity but would not be known until the end of the boresight.

$$\mathbf{c} = \frac{1}{N_b} \sum_{k=1}^{N_b} \mathbf{b}_k \tag{3.9}$$

The equation for the slope takes multiple steps [2]. First determine the covariance of the beam line data

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{b}_1 - \mathbf{c} \\ \cdots \\ \mathbf{b}_x - \mathbf{c} \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 - \mathbf{c} \\ \cdots \\ \mathbf{b}_x - \mathbf{c} \end{bmatrix}^T \tag{3.10}$$

The slope is the eigenvector with the largest eigenvalue of the covariance matrix $\mathbf{\Sigma}$. The slopes are also normalized so the length of the vector is one in order to aid in a later step. This process is repeated for each beam to determine a list of intersections $\mathbf{c}_x$ and a list of slopes $\mathbf{s}_x$, where $x$ denotes the center and slope that belong to beam

$\mathbf{b}_x$. Next to determine the sensor origin $\mathbf{o}$ the best intersection of all the beams is determined [24]. The best fit lines and sensor origin are shown in Figure 3.10.

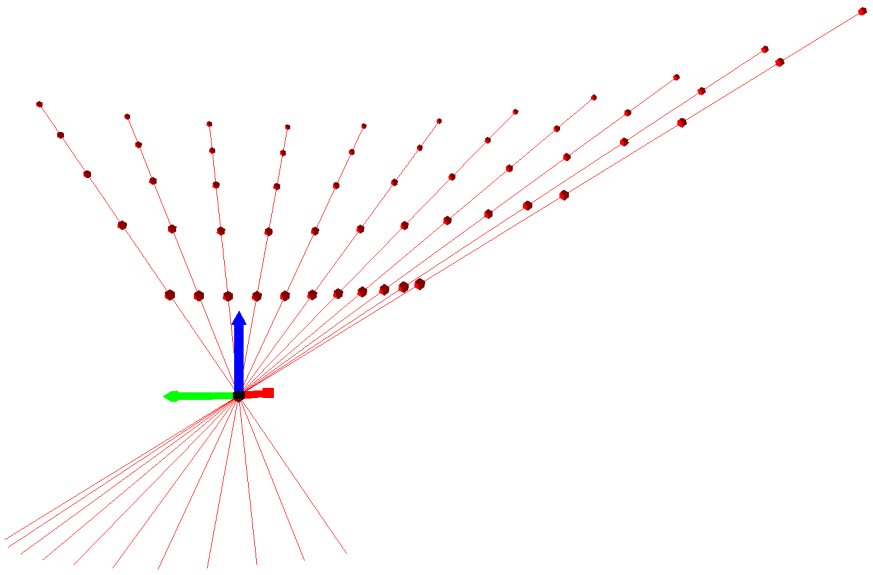

Figure 3.10: Each red line represents the best fit line to the points of that beam. Next the best fit intersection of all the points is determined and used as the origin as shown by the black sphere at the center of the axis.

$$\mathbf{B} = \sum_{k=1}^{N_s} \left(\mathbf{I}_3 - \mathbf{s}_k \mathbf{s}_k^T\right) \tag{3.11}$$

$$\mathbf{B2} = \sum_{k=1}^{N_s} \left(\mathbf{I}_3 - \mathbf{s}_k \mathbf{s}_k^T\right) \mathbf{c}_k \tag{3.12}$$

$$\mathbf{o} = \mathbf{B}^{-1}\mathbf{B2} \tag{3.13}$$

The next step is to determine the $x, y, z$ axes for the $l-frame$ of the LiDAR. To determine the $z$ axis denoted as $\mathbf{z}$, the slopes of beams with equal elevation are collected into a list $\mathbf{s}_k$. These slopes make a cone with the point of the cone at the sensor origin. The base of the cone is a plane whose normal is in the same direction as the $z$ axis of the LiDAR. To determine the normal of this plane, the slopes are

treated as points on the base of the cone and fit to a plane as shown in Figure 3.11. This is done with a linear regression of the plane equation $1 = ax + by + cz$, where $x, y, z$ are points being fitted to the plane and $a, b, c$ are the determined variables to describe the plane.

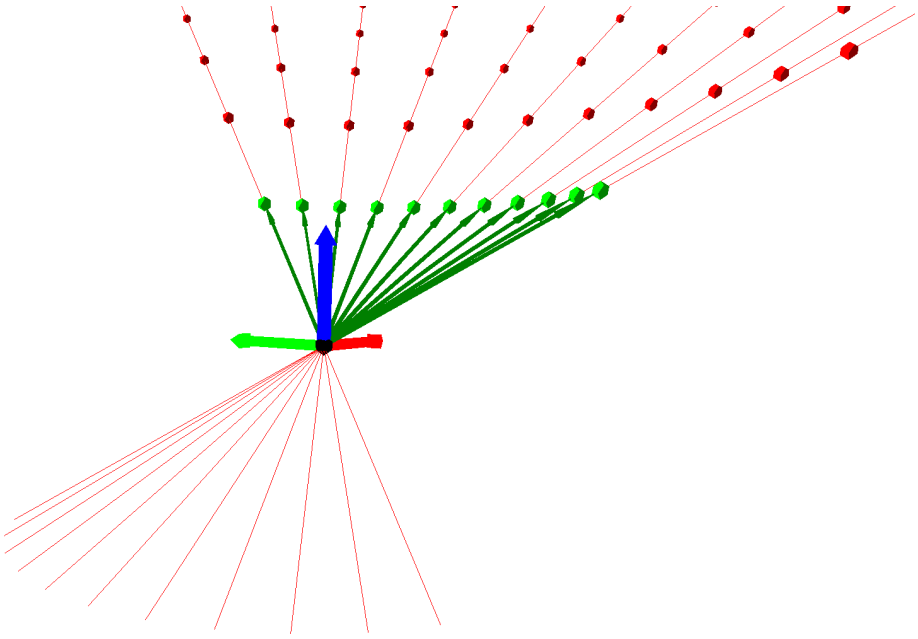

Figure 3.11: The normalized slope vectors, shown as dark green arrows, are used to determine the green set of points. These points trace out the base of a cone created by all the slope lines.

$$\mathbf{C} = \sum_{k=1}^{N_s} \mathbf{s}_k \mathbf{s}_k^T \tag{3.14}$$

$$\mathbf{C}_2 = \sum_{k=1}^{N_s} \mathbf{s}_k \tag{3.15}$$

$$\mathbf{z} = \mathbf{C}^{-1}\mathbf{C}_2 \tag{3.16}$$

The vector $\mathbf{z}$ is then inverted if it is the down direction as opposed to the up direction. The best fit plane is shown in Figure 3.12. The next axis to determine is the $x$ axis or 0 degree azimuth angle of the LiDAR. The slope of the 0 degree azimuth line can be

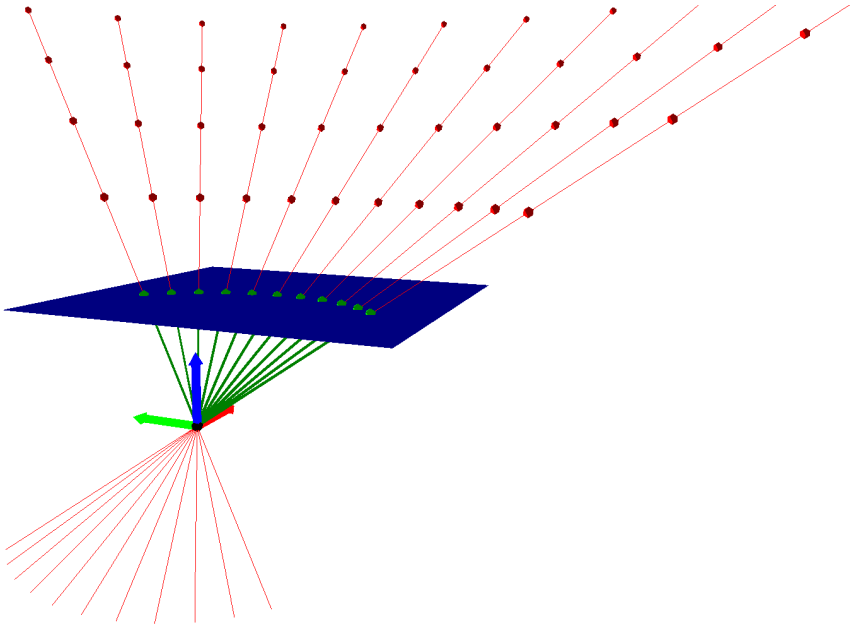

Figure 3.12: The green slope points are fit to the blue plane which is the base of the cone made by all the slope vectors. This blue plane has a normal parallel to the blue axis arrow. Thus from this the blue axis arrow has been determined.

used for this direction, but some LiDARs may not have a 0 degree azimuth. In this case the slopes of lines can be rotated about the $z$ axis by their know azimuth angle to determine an $x$ axis.

$$\mathbf{x} = \mathbf{C}_k^0 \left( \mathbf{z} \times \mathbf{s}_k \right) \tag{3.17}$$

Where $\mathbf{C}_k^0$ is an axis rotation about the $z$ axis with an angle equal to the azimuth angle of the slope line $\mathbf{s}_k$. While this vector points in the correct $x$ axis, it is not perpendicular to the $z$ axis as shown in Figure 3.13. To correct this, first the $y$ axis is computed by the cross product between the $z$ and $x$ axis by $\mathbf{y} = \mathbf{z} \times \mathbf{x}$. Then the $x$ axis is recalculated with the cross product $\mathbf{x} = \mathbf{y} \times \mathbf{z}$. Now the boresight of the LiDAR is known relative to the frame the point clouds were collected in.

*3.4.3 Data Collect Procedure.* In order to perform this mathematical calculation, points where beams strike must be determined. The method of this thesis

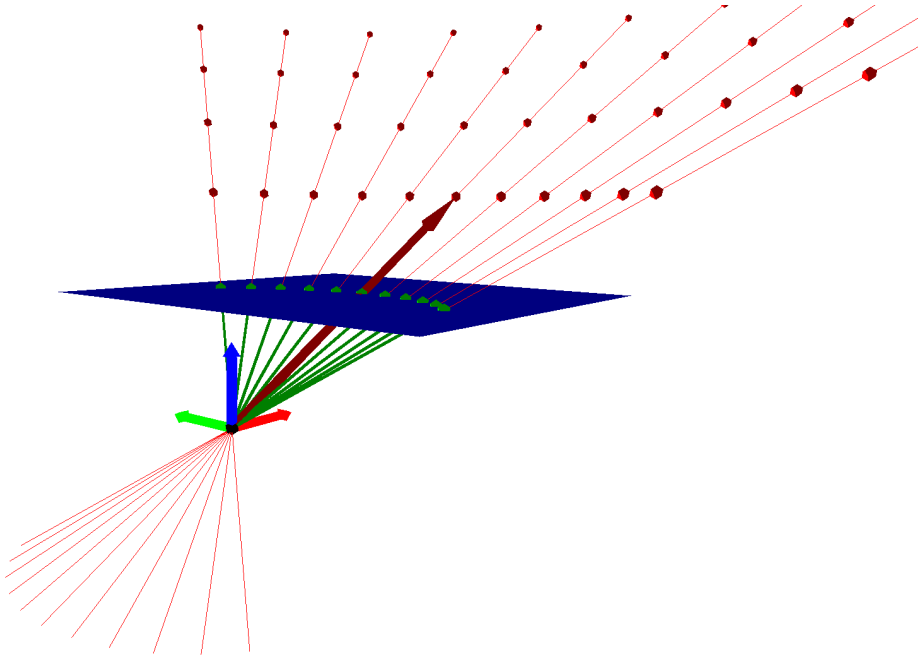

Figure 3.13: The red axis arrow is determined by rotating all the slope vectors about the blue axis by the azimuth they represent creating the dark red arrow. This axis is in the correct direction as the red axis arrow however it must be orthogonal to the blue axis arrow. Thus a series of cross products are used to determine the green axis arrow and then the final red axis arrow.

uses a flag board to determine a beam strike point, and then a surveying laser to give an accurate location for the strike point of the beam. The flag board is shaped so the board can be lined up with a horizontal scan line and then translated over to the desired vertical scan line. The flag board is "L" shaped, thus the horizontal scan line must have a low elevation so that the scan projected onto the board is almost straight. Lines of higher elevation will start to curve and make it impossible to line up with the square "L" shape. A notch in the flag board is used to line up the board with lines of constant elevation. The flag board is positioned so half of the desired horizontal elevation line runs across the notch in the board as shown by the green line in Figure 3.14. This is confirmed to be the notch of the board, and not the edge of the whole board, by examining the lines of elevation below the desired elevation. The line below the desired line must hit the entire board and extend beyond the desired line otherwise the desired line did not hit the notch. This same process is repeated

for the vertical line while keeping the horizontal line lined up with the notch. When both lines are lined up with the notches, the intersection of both lines is inside the red box of the board as shown in Figure 3.14.

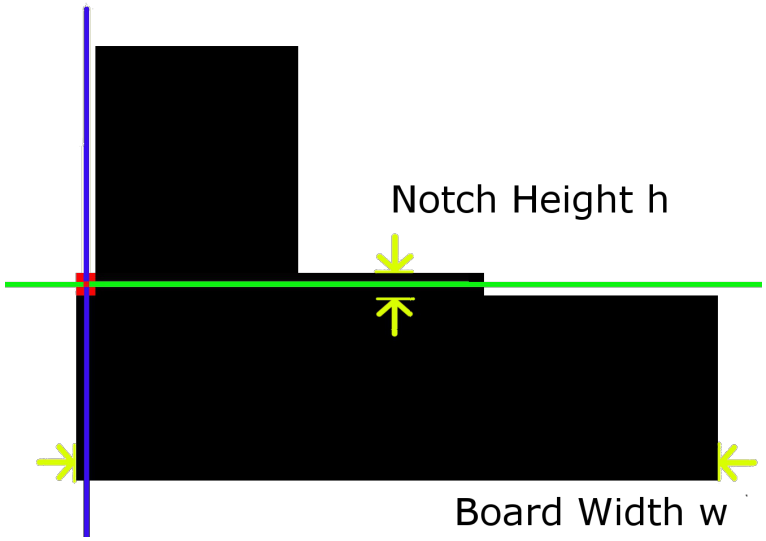


Figure 3.14: The flag board used to catch lines of equal elevation (green line) and lines of equal azimuth (blue line). The intersection in the red region is where the beams of the given elevation and azimuth strikes the board.

The center of the red box is then scanned with the surveying laser to collect a point where the beam strikes. In order to position the flag board, the distance of the scans must be know in real time with a graphical utility. The graphical utility allows a person to quickly line up the board with the desired horizontal and vertical lines. After the position of the board is confirmed with the graphical utility, another person on the surveying laser can scan the red square on the flag board and record the strike location.

*3.4.4 Error Analysis.* The error of such an approach is dependent on the size of the notch in the flag board and the flag board distance from the LiDAR. If it is assumed the flag board is positioned correctly and the strike point is in the red box the maximum angular error can be calculated. Let $r$ be the distance to the strike point on the board, $h$ be the height of the notch, and $w$ be the width of the flag board. If the surveying laser is perfect and surveys the exact center of the red box,

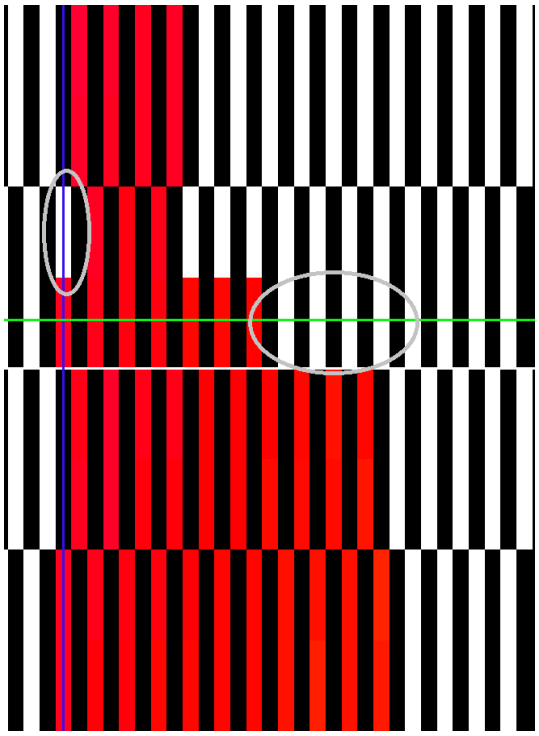

Figure 3.15: Graphical utility used to position board. Each rectangle represents a beam from the LiDAR. The color of the rectangle represents what the beam strikes. In order to keep geometry consistent, when an equivalent azimuth does not exist at other elevations, the azimuth must still be drawn but as a black rectangle. White rectangles represents laser scans that miss the flag board. Red rectangles represents laser scans that strike the flag board. The following have been added to clarify the picture but are not part of the utility. The green and blue lines represent the horizontal and vertical lines to capture respectively. The gray circles show the white rectangles that miss the board due to the notch in the board. These are the beams that are lined up by the person holding the board in order to assure the green and blue lines intersect in the correct rectangle, and thus capture the correct azimuth and elevation.

the approximate max angular error is $e_{yaw} = e_{pitch} = \tan^{-1}(h/(2r))$. In this equation $h/2$ is the max error that will still allow the strike point to have hit the board. When determining the accuracy of measured points this is replaced by the average residual distance $d_{mean}$ from fitting the points to a line. This error applies only to the yaw and pitch of the LiDAR and assumes no error in roll. Any error in roll will decrease the error in yaw and pitch or else the conditions for the strike point to be in the red box would not be met. To determine the max roll error, it is assume that there is no

error in either yaw or pitch to produce equation $e_{roll} = \tan^{-1}(h/w)$. Thus for a given flag board, a minimum distance to survey, the angular error can be bounded.

In a test of this method, six beams were used to record at least four strike points for each beam. The board was positioned by hand and the survey was performed using optics in the surveying laser. The notch size was $h = 2.56$cm with a board width of $w = 300$cm and a minimum distance of $r_{min} = 400$cm. The mean of the distance from the strike point to the best fit line for all the points was $d_{mean} = 1.33$cm. This average is greater than half the notch size but this is expected as other errors arise in human error such as holding the flag board in place and surveying the exact center of the red box. If the average error is used, and minimum strike distance is known, it is possible to calculate the mean yaw or pitch error with equation $e_{yaw} = e_{pitch} = \tan^{-1}(d_{mean}/r_{min}) = 0.1905$ degrees or 3.3 MRAD. If it is assumed all the error is contained in the roll, the worst possible roll error is $e_{roll} = \tan^{-1}(h/w) = 0.4889$ degrees or 8.5331 MRAD. Thus it is possible to accurately boresight a scanning LiDAR using a flag board and surveying laser.

# IV. Results

This chapter analyzes the results from simulations and actual test flight data using both the MBI and PPD algorithms. The test flights are described in detail to frame the nature of the results derived from the test flights. Simulated results were created to describe expected results with the flight configuration. The simulation results include analysis of properties of the algorithms such as convergence areas and modifying input parameters. The test flight data was used to generate estimated relative position and compared to the true relative position of the two aircraft.

## *4.1 Test Flight Description*

A series of test flights were conducted by the Air Force Research Laboratories (AFRL) at Forbes Air Field near Topeka Kansas. The flights included a Boeing KC-135 Stratotanker as the tanker aircraft and a Calspan Learjet as the substitute for an actual receiver aircraft. A total of eight test flights were conducted to collect measurement data. The LiDAR was not the only sensor being tested during the test flights, thus some test flights received no useable LiDAR data. When useable LiDAR data was received, the tanker and receiver flew racetrack patterns typical of actual refueling procedures.

*4.1.1 Learjet Modification.* The Learjet was modified for the test in order to evaluate the selected sensors and provide a truth system of measurements for post analysis of the test flights. The nose cone of the Learjet was modified to allow the sensor suite to see outside the nose cone as shown in Figure 4.1. The sensors included in the sensor suite were an Ibeo LUX 8L Laser Scanner, a Proscillica 1660C camera, and a NovAtel SPAN-SE Receiver as shown in Figure 4.2. The sensor suite was installed on a mounting plate in the nose cone of the Learjet as shown in Figures 4.3 and 4.4. This mounting plate allowed the relative orientation of the sensors to be calculated in a laboratory before the actual flight test. The nominal orientation for the Ibeo LUX was a pitch of 12 degrees and a roll of 16 degrees, while the Proscillica camera was pitched 30 degrees. This aimed the LiDAR at the left wing as shown in

Figure 4.5 while the Proscillica was aimed at the center of the fuselage as shown in Figure 4.6. The sensor suite was connected to a laptop on-board the Learjet in order to collect and monitor data. This monitoring ensured the sensors were operating correctly and proper data was being recorded. If problems were observed, steps to fix the errors could be taken, such as rebooting a sensor or modifying sensor settings via the laptop. The Learjet was also outfitted with an array of GPS antennas, GPS receivers, and IMUs. In total, there were three operational GPS antennas mounted to the top of the Learjet. Each GPS antenna was connected to an LN-251 Embedded GPS Inertial Navigation Systems (EGI) and to a NovAtel OEM4 GPS receiver. The Novatel receivers acted as a truth position system for the position estimates from the EGI system. The truth as referred to by this thesis is the position solution from the Novatel GPS receivers and only the attitude data from the EGIs. A recording system separate from the sensor system recorded the raw data of the EGI and Novatel receiver. The truth then includes a location in ECEF coordinates and attitude in the form of yaw pitch and roll Euler angles from the $n'-frame$. The Learjet was also equipped with a communication system to transfer data to the tanker aircraft.

Figure 4.1: Nose cone of Learjet modified with panes of glass to allow sensors to see through nose cone.

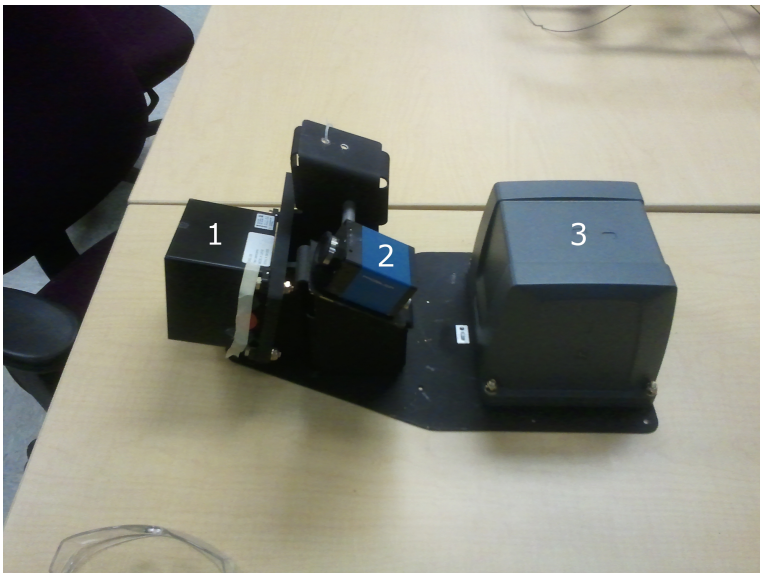


Figure 4.2: Sensor suite installed in the nose cone of the Learjet. Sensors from left to right are Ibeo LUX 8L Laser Scanner(1), Proscillica 1660C Camera (2), NovAtel SPAN-SE Receiver (3)

*4.1.2 KC-135 Modification.* The KC-135 tanker was modified for the test in order to obtain truth measurements for the test flights. The KC-135 has only one GPS antenna connected to two LN-251 EGIs and two NovAtel OEM4 GPS receivers to serve as truth for the test flights. The KC-135 was outfitted with a recording system to record the measurements from the on-board EGI and Novatel receiver. The truth from the tanker refers to the attitude data from the EGIs and the position from the Novatel receiver. The tanker was also fitted with a communication system to transfer data to the Learjet and ground station during flight. The data to the ground station allowed realtime monitoring of the test flight and conditions of the tanker. The tanker acted as a relay for communications from the Learjet and was the only aircraft to communicate with the ground station. The tanker was also outfitted with optical markers as shown in Figure 4.6. The markers allowed different vision algorithms to be applied to the images taken by the camera. The markers also served as known locations on the tanker to scan with a commercial laser surveying system.

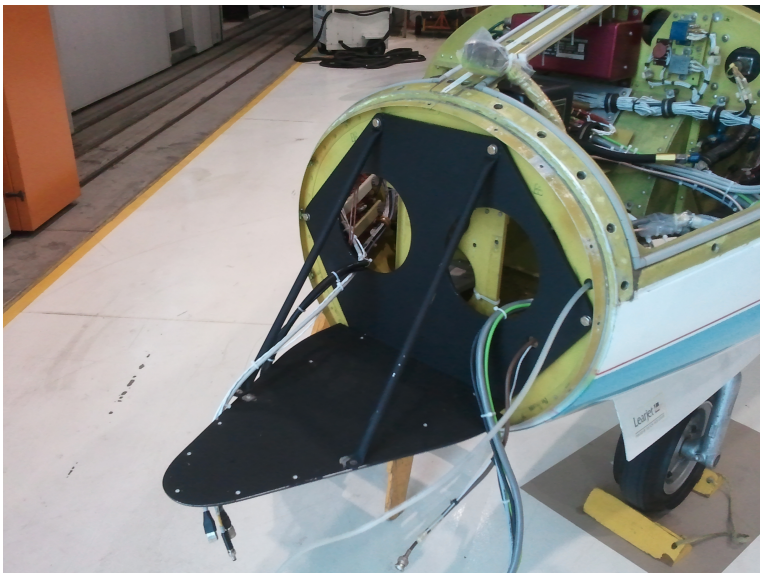

Figure 4.3: Mounting plate in nose cone of Learjet.

*4.1.3 Ibeo LUX 8L.* This section presents a brief overview of the characteristics of the Ibeo LUX 8L scanning LiDAR. The LiDAR has eight levels of elevation that are scanned four levels at a time, the top four then the bottom four. Each elevation is 0.8 degrees apart, splitting the 0 degree mark for elevations starting at 0.4 and -0.4. The azimuth scans are not evenly spread out. The positive four elevation lines are divided into the top two and bottom two elevations lines. The top two elevation lines stretch from 33.5 to -59.5 degrees in increments of 0.5 degrees, and the bottom two elevations reach from 49 to -50 in increments of 0.5 degrees. This is repeated on four elevation layers on the bottom scan. Figure 4.7 shows the scan pattern. The LiDAR quantizes measurements into bins of 0.04m. The measurements have a standard deviation of 0.1m as stated in the manual [13]. The scan frequency is 50Hz per four line scan giving a full eight line scan a frequency of 25Hz. The Ibeo LUX 8L can also return multiple ranges per elevation and azimuth as echoes. Echoes occur usually when the beams hit transparent objects such as glass or raindrops. Each range return also has an associated pulse width that describes the width of the pulse when it struck an object. These characteristics were used when simulating data from the LiDAR simulator.


Figure 4.4: Sensor suite fully mounted in the nose cone of Learjet.

*4.1.4 Lever Arms.* The tanker and receiver were equipped with many sensors, each with their own relative reference frames. In order to analyze the data from these sensors, the lever arms and orientations between the sensors must be determined with respect to a common reference frame. It is also necessary to relate these lever arms to the same frame. This thesis relates all the sensors to the frames of the aircraft they were installed on, i.e., the $b_{receiver}-frame$ and $b_{tanker}-frame$. The sensors in the tanker include the GPS antenna and EGIs. The position of the EGIs is not important since only the inertial measurements were recorded, however the orientation is important. In this case, the orientation of the EGIs is exactly in line with the $b_{tanker}-frame$. The GPS antenna position was available by referencing the diagram of the aircraft.

The Learjet lever arms were much more difficult to obtain since custom sensors were installed. The location and orientation of the custom sensors had to be done on a case by case basis, whereas the pre-installed sensors could be referenced to a diagram of the aircraft. The GPS antenna locations were obtained from a aircraft diagram in the same manner as the tanker antennas. The EGIs and sensor suite were custom installations on the Learjet, and orientations and position had to be determined during

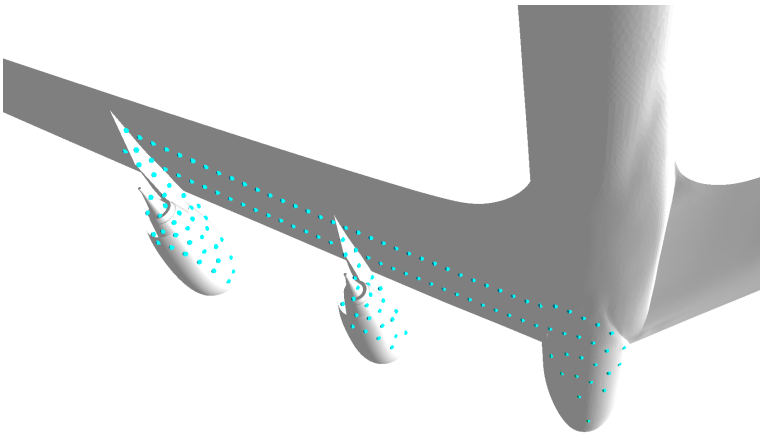

Figure 4.5: Predicted scan area on the tanker from the LiDAR. The left wing, engine pods, and fuselage are struck by LiDAR scans as shown by the blue spheres.

the flight test. The EGI calibration and boresight was completed by professionals working with the test flights. To obtain the orientation of the sensor suite a Trimble S3 surveying system was used. The position and orientation of the Proscillica camera in the Trimble S3 frame was determined using a standard camera calibration [1]. To determine orientation and position of the Ibeo LUX 8L, the procedure in Section 3.4 was used. The Trimble Surveying system was related to the $b_{receiver}-frame$ by surveying points on the aircraft outlined in the aircraft diagram. The IMU in the sensor suite was calibrated and boresighted using the same method as the EGIs. Thus a table of lever arms and orientations was created, which can be found in the Appendix in Tables A.2-A.3.

*4.1.5 Test Flight Overview.* In total, eight test flights were conducted. Of the eight test flights, only five collected useable LiDAR data. Flight three encountered sensor issues that created incorrectly time tagged data, and thus the data was not useable. During flight four, the sun was in the eyes of the pilots, which caused them to deviate from the aerial refueling pattern, causing the LiDAR returns to miss the wing in many scans. Flight five encountered adverse weather conditions that prevented simulating refueling, thus no LiDAR data was collected. Flight six operated outside the approved area for the LiDAR, resulting in nearly no useable LiDAR data. Of

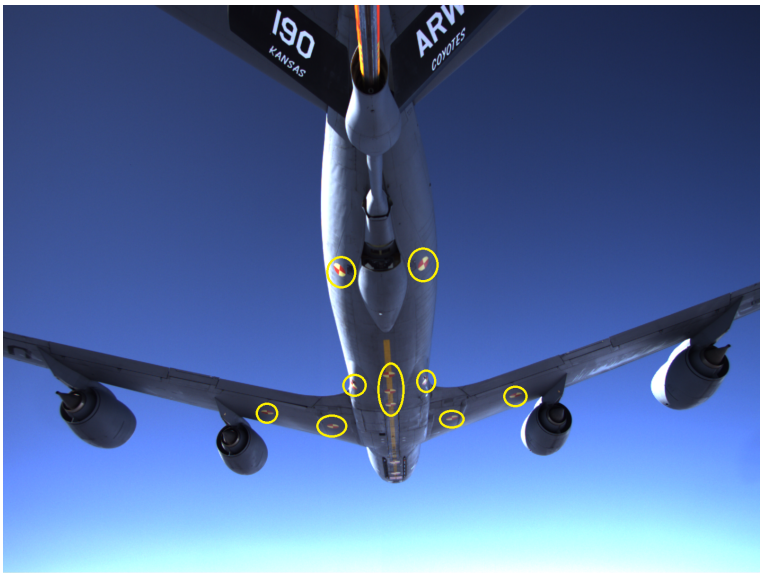


Figure 4.6: KC-135 with markers installed on the body of the aircraft. Markers have been circled with yellow. Picture taken in flight by Proscillica camera.

the remaining flights, flights one and four collected one pass each of useable LiDAR data. Flights two, seven, and eight collected five passes each of useable LiDAR data. Each pass lasted on average 7 minutes and consisted of an approach from pre-contact position, a hold of contact position, and a back off to pre-contact position. Flights two and eight were collected on sunny days, and during the flight the sun came across the field of the view of the LiDAR. When the LiDAR pointed directly at the sun, erroneous measurements were recorded. The measurements varied the range where the LiDAR was looking at the sun, thus creating a cone of spurious measurements. When the sun came all the way across the LiDAR almost at a 90 degree angle another type of erroneous measurements were recorded. In this case the glare from the sun causes spurious measurements across the range of view of the LiDAR. For a complete list of the test flights refer to Tables A.5 and A.6 in the Appendix. After the test flights were completed, simulation and analysis was completed to determine the accuracy of the position estimate algorithms.

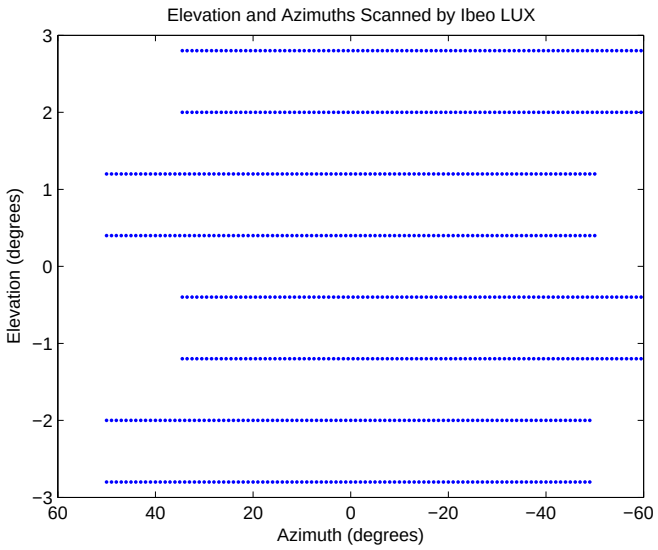


Figure 4.7: Scan pattern of the Ibeo LUX 8L. View is seen as if standing behind LiDAR.

## *4.2 Functional Analysis*

Before the algorithms could be used for position estimates, the performance of the major components of each algorithm was verified. The k-D Tree method to determine closest points on the plane model to measurements had to be verified to return results close to the true closest points. The true points were determined by an exhaustive brute force search of all triangles on the mesh. The LiDAR simulator was analyzed to determine the error between the simulated measurements and measurements created through brute force ray tracing, where ray tracing is considered the true measurement. The speedup from using these two methods over the brute force methods is required to accomplish the algorithms within hours instead of weeks.

*4.2.1 k-D Tree Analysis.* The k-D tree method to determine closest points was analyzed in two modes. One mode allowed branching into both child nodes if the separation distance was too small, and the other mode ignored the far child node and only went with the near child node, no matter the separation distance. The first method that uses both nodes is more accurate, but the time to complete the search is dependent on the search value. The speed of the second method is independent of the

Table 4.1: Normal k-D Tree with far child error statistics. Axis in $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | 0.0023 cm | 0.1144 cm | 0.1144 cm |
| **y axis** | 0.0030 cm | 0.2882 cm | 0.2883 cm |
| **z axis** | 0.0126 cm | 0.2723 cm | 0.2726 cm |
| **MRSE** | 0.4129 cm | | |

Table 4.2: k-D Tree Without Far Child Error Statistics. Axis in $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | 0.0312 cm | 6.6887 cm | 6.6887 cm |
| **y axis** | 0.0265 cm | 3.4902 cm | 3.4902 cm |
| **z axis** | 0.0177 cm | 2.5246 cm | 2.5247 cm |
| **MRSE** | 7.9558 cm | | |

search value but increases the error to the true closest point. Both methods searched across triangles in order to find the closest point on the mesh of triangles as described in Section 3.2.2.2. These results were compared to a brute force method that searched every triangle in order to determine the closest point on the plane model. One flight pass consisting of 275 different poses was used to compare the different methods. The normal k-D tree with far child method has error statistics for each axis as well as the MRSE as shown in Table 4.1 and the time taken for each method shown in Table 4.3. The k-D tree without far child method commits more error as shown in Table 4.2. However by allowing more error the algorithm speed increase of about three times compared to the standard method as shown in Table 4.4. The normal k-D tree, while slower than the no far child, is still three orders of magnitude faster compared to the brute force method. This thesis takes a mix of speed and error and uses the normal k-D tree with far child as the standard k-D tree nearest neighbor search method.

*4.2.2 LiDAR Simulator Analysis.* The LiDAR simulator was tested for accuracy and speed compared to the brute force method of iterating over every triangle

Table 4.3: Time taken for each closest method.

| k-D Tree | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **k-D Tree Normal** | 17.581ms | 4.768ms | 18.214ms |
| **k-D Tree with Far Child** | 5.138ms | 1.613ms | 5.385ms |
| **Brute Force** | 16969.399ms | 4003.207ms | 17433.528ms |

Table 4.4: k-D Tree speedup compared to the brute force method.

| k-D Tree | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **k-D Tree Normal Speedup** | 976.816 | 66.133 | 979.044 |
| **k-D Tree with Far Child Speedup** | 3349.180 | 255.525 | 3358.878 |

on the plane model to check for hits from the desired rays. The ray shooting method is considered to generate the true measurements in order to determine the accuracy of the LiDAR simulator. A base pose was chosen to evaluate the differences between the LiDAR Simulator and the ray shooting method. The LiDAR simulator at a resolution of 1680x1050 had statistics as shown in Table 4.5 and the time taken shown in Table 4.6. The simulator was also tested at a lower resolution to see the effect on range error. Resolution changes the error because each pixel represents a certain elevation and azimuth. If the pixel is not exactly the desired elevation and azimuth, error is induced from the rounding of the elevation and azimuth. Increasing the resolution reduces the roundoff error and produces more accurate results. To prove the error is resolution dependent the same run was performed with a resolution of 1360x768. The errors increased as shown in Table 4.5 and time reduced as shown in Table 4.6. Thus to improve accuracy the resolution must be increased. However increasing the resolution will require more processing and slow the simulator. The LiDAR simulator at 1680x1050 was slower than the LiDAR simulator at 1360x768 as shown in Table 4.7. As seen with the k-D tree there is a tradeoff between speed and accuracy. This thesis uses a mix of speed and error and uses the LiDAR simulator at a resolution of 1680x1050.

Table 4.5: Error between LiDAR Simulator and Ray Tracer ranges using a base pass

| Resolution | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **1680x1050** | -0.148cm | 5.774cm | 5.776cm |
| **1360x768** | -0.499cm | 11.083cm | 11.094cm |

Table 4.6: Time taken for each LiDAR simulator

| LiDAR Simulator | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **1680x1050** | 26.681ms | 2.558ms | 26.803ms |
| **1360x768** | 17.740ms | 2.206ms | 17.876ms |
| **Brute Force** | 58349.322ms | 78.003ms | 58349.374ms |

### *4.3 Simulation Analysis*

With the functional components of each algorithm verified, the parameters of each algorithm were determined. Simulated data based on the flight test trajectories was created from the LiDAR simulator and used to generate relative positions using the MBI and PPD algorithms. The test flight recorded relative position and attitudes were used but not the LiDAR data. The LiDAR data was simulated as described in Section 3.3.2, with the characteristics of the Ibeo LUX 8L described in Section 4.1.3, and the lever arms setup of the test flight. The positions were also down sampled to one position every second. This simulation also serves as a baseline performance for the relative position estimation. Other simulations to determine the convergence region of the MBI algorithm and determine the sensitivity of the PPD algorithm were also conducted. MBI is heavily dependent on the initial guess and can converge to local minimums if the guess is too far from the truth. This simulation determines how

Table 4.7: LiDAR Simulator speedup compared against the brute force method

| Resolution | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **1680x1050** | 2196.605 | 109.812 | 2199.338 |
| **1360x768** | 3309.028 | 175.458 | 3313.659 |

far the initial solution can vary before the solution converges to a local minimum. The PPD sensitivity determined how accurately the SRRE can predict the true relative position. The tanker was perturbed about a nominal position to determine how sensitive the SRRE is to small changes in position.

*4.3.1 MBI Parameters.* The two parameters to determine for the MBI algorithm, as described in Section 3.2.4, were the move threshold and the max iterations . The algorithm was run using two different modes in order to determine these parameters. The first is the Last Estimated Pose (LEP) mode, where the initial guess is the best estimate from the previous time step. The other is the Last Truth Pose (LTP) mode, where the initial guess is the truth position in the last time step. By estimating the parameters in the LTP the algorithm should be more stable and less likely to diverge. The LTP results are expected to be better than the LEP method. The LEP method is used to make sure the algorithm converges over time, because it is the method the algorithm will use to determine its actual best position. Parameters were chosen and run using the LTP method to lower the MRSE of the entire run. After the MRSE dropped below 0.5m the LTP method was used to assure the algorithm could still track the position. This aspect is very important because, due to the low dynamics of AAR, the error statistics can appear reasonable even when the algorithm is not tracking the position.

In order to determine if tracking occurred figures such as Figure 4.8 were used. This figure shows from the starting point of the position how much of the error was eliminated. Values below the 100% line reduced the starting error and moved towards the true position. If the values average below 100% then tracking is occurring. To demonstrate how error statistics can appear low while tracking is not occurring consider this test case. At one time instant the algorithm moves away from the true current position but by chance moves very close to the next position. At the next time instant the error is very low so even when the algorithm moves away from the true position the error is remains low. By only examining the error, instances

Table 4.8: Table showing the parameters used for the MBI algorithm during simulation.

| Parameter | Value |
|---|---|
| **Move Threshold** | 3.048E-06 (m) |
| **Max Iterations** | 3000 |

such as these are not visible. It requires examination of the error reduced from the starting position. Thus by trial and error using the LTP then LEP methods while observing not only position error statistics, but percentage error reduced statistics as well, the final parameters in Table 4.8 were determined. These parameters were used to examine the actual flight data. The error statistics for a simulated run of flight 1 pass 1 are shown in Table 4.9 and Figure 4.9.

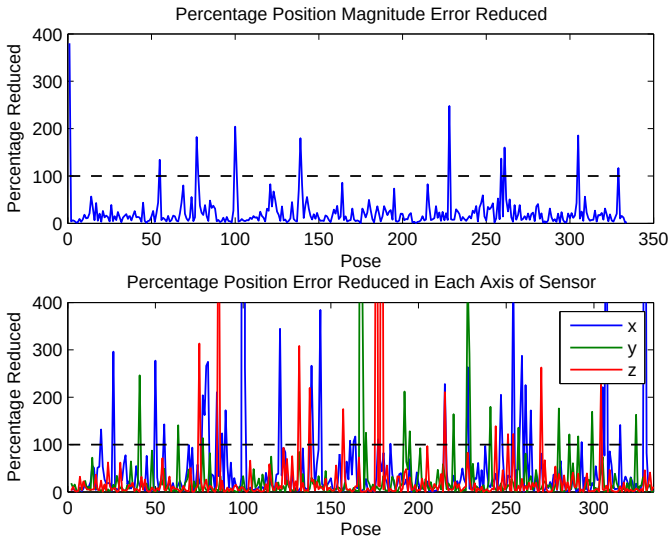


Figure 4.8: MBI Algorithm percentage of starting error reduced for flight 1 pass 1 using final parameters. Figure used to determine if algorithm is tracking positions to reduce initial error. Axis in $l-frame$.

*4.3.2 MBI Convergence.* The initial position estimate is important for convergence of the MBI algorithm and this section explores how accurate the initial position estimate must be for convergence to the global minimum to be assured. This

Table 4.9: MBI Algorithm using final parameters and simulated LiDAR measurements from Flight 1 Pass 1. Axis in $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -1.603 cm | 3.856 cm | 4.171 cm |
| **y axis** | 0.251 cm | 0.998 cm | 1.027 cm |
| **z axis** | 0.157 cm | 0.355 cm | 0.388 cm |
| **MRSE** | 4.313 cm | | |
| **LiDAR Data** | Simulated Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup | | |

is not to be confused with the question of how does the relative position or LiDAR setup effect the ability to converge to the global minimum. In this test, the relative position of the aircraft is set to the ideal contact position with no relative attitude difference. The LiDAR configuration is the same from the test flight in terms of orientation and scan pattern. The simulator from the PPD algorithm was used to simulate the LiDAR scans. The MBI algorithm was started at positions in a box pattern, with the truth position at the center of the box to determine at what regions the algorithm converges to local minimums. The first box has a side length of ten meters and was divided into 100 one meter boxes. Viewing the path the estimated position moves during the simulation leads to insight on which directions have high visibility.

It quickly became apparent that the algorithm can determine the vertical axis position of the $b_{tanker}-frame$ in a few iterations as shown in Figure 4.10. Next the left-right axis was narrowed down as the lines slowly merged into the center. The forward back direction was the last to be narrowed down. This is consistent with the error observed in simulation from Table 4.9 where the forward-back axis had the most variance. Despite the visibility issues in the forward-back, axis all of the positions within the ten meter box eventually converged to within millimeters of the correct original location in the center of the box as shown in Figure 4.12. Next a 50 meter box broken into 100 five meter boxes was used. In this case local minimums were

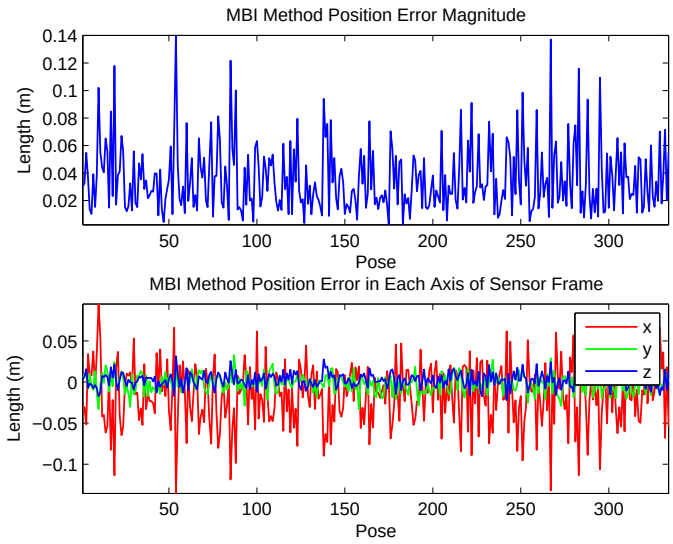


Figure 4.9: MBI error using simulated LiDAR measurements and Last Estimated Pose mode. Top plot shows the error length between the best estimated position and the truth position for the simulated data set. The bottom plot shows the error in each direction as seen through the LiDAR. Axis in $l-frame$.

observed as shown in Figure 4.11 and 4.12 . The minimums converge closely to the same vertical level but differ in the left-right and forward-backward axes. However, when the algorithm does converge on a local minimum, there are methods to determine the converged position is not the global minimum.

One such method is to evaluate $\rho_{px}$ which is described in Section 2.6. Figure 4.13 shows the correlation between $\rho_{px}$ and the position error. When the position error is large, $\rho_{px}$ drops from the value of those points with low position error. The value of $\rho_{px}$ at the global minimum can be determined empirically in simulation by starting the algorithm at the true center and observing the final $\rho_{px}$ value. This value can be compared to the values obtained from the algorithm when the starting position is not known in order to determine if the algorithm converged at a local or global minimum. Thus, even if the initial position estimate is poor, it can be determined if a local minimum was reached and if so, a more expansive search space can be used to find the true global minimum. While testing the convergence unexpected characteristics of the algorithm were discovered, and these will be described in the next section.

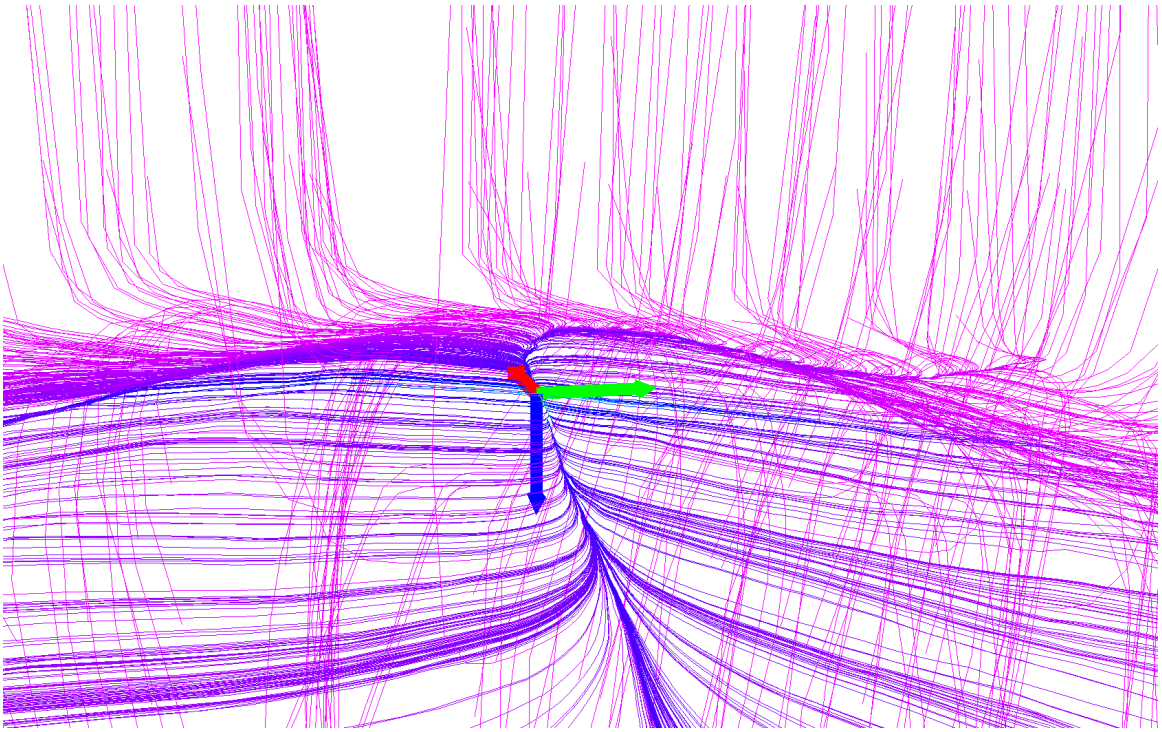

Figure 4.10: Convergence simulation run with a 10m box. The x,y,z axes correspond to the red, green, and blue arrows respectively in the $b_{tanker}-frame$. Each line represents a trail of the position as it moved towards its convergence point. The color of the line represents how many iterations have been completed to reach that position. Lines with few iterations and thus far from their convergence point are colored purple and gradually move towards red as they complete more iterations. Thus all the purple lines have completed few iterations while the blue lines are closer to finishing. Red/Orange lines are not visible because when they are close to converging the incremental movement is very small and the small lines are difficult to see.

*4.3.3 MBI Characteristics.* One characteristic of the MBI algorithm is the speed for each iteration. The closer the initial guess is to the truth, the faster the iteration was completed. The positions where the position did not converge on the global minimum took much more time compared to the ones that did converge on the global minimum. This time characteristic was also observed over the course of testing when gross measurement errors were accidentally introduced. Analysis showed nearly all of the computation time is used to determine the closest point to each measurement point, specifically querying the k-D tree. This is attributed to using the far child approach in the k-D tree. As shown in Section 4.2.1, using the far child approach

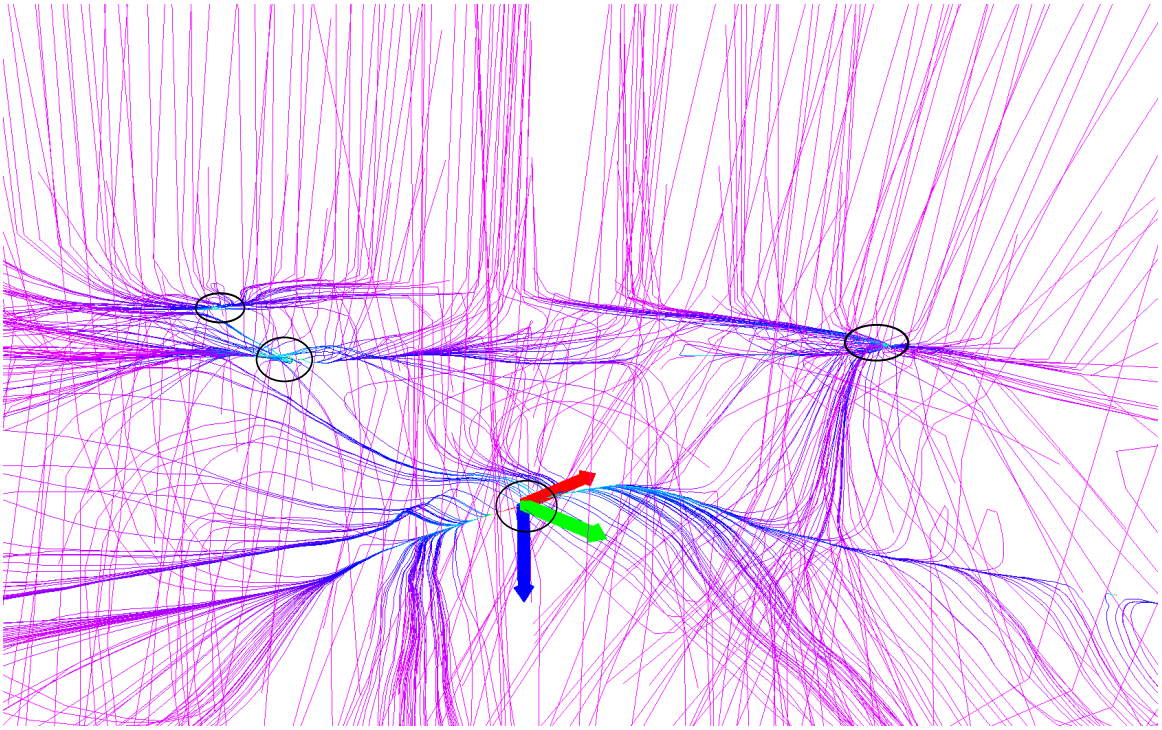

Figure 4.11: Convergence simulation run with a 50m box. The x,y,z axes correspond to the red, green, and blue arrows respectively in the $b_{tanker}-frame$. For Description of lines see Figure 4.10. Local Minimums have been circled in black. The global minimum is the circle on the coordinate axes.

is slower compared to the no far child approach but the speed is worsened in poor matches. This characteristic makes the MBI algorithm very slow at narrowing down a poor initial estimate but very quick at taking a close estimate and narrowing down the error.

Another characteristic is the problem of cyclical iterations. This occurs when the algorithm determines the next best position to be the last best position. The new best position will again point to the same position it did before, thus creating a loop. This looping is why a max number of iterations was introduced as a necessary stopping condition, because the move threshold will not decrease in this loop. The looping can move between an variable number of positions before returning to a previous position and the exact spot of each position cannot be guaranteed. Thus, any method to determine the looping must be quick and robust to handle looping for any number of positions. The problem of how to break the loop is also not trivial, since it is likely the

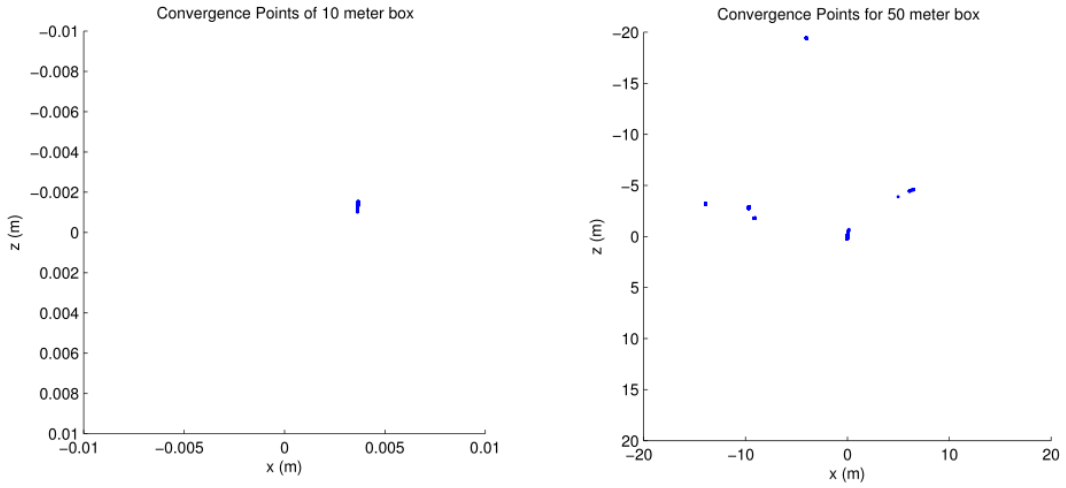


Figure 4.12: Left Plot: Convergence points for the 10m box test. This is an overhead view through the virtual camera frame looking down with -z as forward direction and x as right direction. All points converge to less than one cm of true center (0,0,0). Right Plot: Convergence points for the 50m box test. This is an overhead view looking down with -z as forward direction and x as right direction. Many points converged to true center (0,0,0) while many others can be as far as 20m off. Local minimums cause convergence to incorrect locations.

algorithm could slip back into the loop. One such method would be to use simulated annealing. Even large perturbations are still likely converge on the correct position since the MBI algorithm has a large area of convergence.

One characteristic observed in simulation is the accuracy in each direction. The final position estimate for the simulation of flight 1 pass 1 had much more variance in the forward-back axis compared to the other axes as shown in Table 4.9. This is most likely due to the shape of the scanned area of the tanker. The LiDAR was aimed at the left side of the tanker and thus receives returns from the left wing, engine pods and a small part of the fuselage. With this setup, the two faces of the left wing and engine pod sides are visible in measurements from the LiDAR. Faces perpendicular to a given axis are the best objects to obtain visibility in that direction, but will give almost no information in the other two directions. Thus, from this setup the left wing gives information to the vertical direction of the $b_{tanker}-frame$ and the engine pod sides give information on the side direction. The forward-back axis, however,

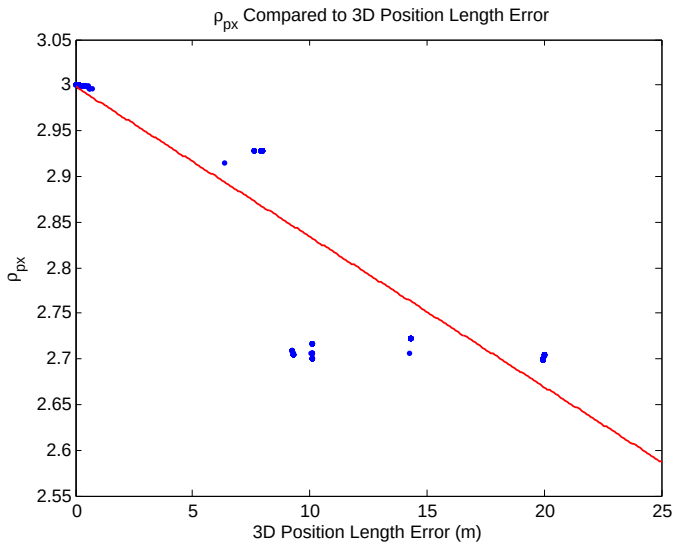


Figure 4.13: Correlation between $\rho_{px}$ and the position error. As 3D position error increases $\rho_{px}$ decreasing thus allowing large position error to be detected. The red line is a linear fit trend-line.

is the worst since there is no face perpendicular to that direction. This is expected from an aircraft as faces perpendicular to the direction of flight would induce drag for the aircraft, thus those types of faces are avoided. This type of error statistics in each direction is seen again in the PPD simulation. From this it can be determined the error is not specific to the method but the LiDAR setup. After the parameters were determined for the MBI method, the parameters for the PPD method were determined, as described in the next section.

*4.3.4 PPD Parameters.* The four parameters determined for the PPD algorithm were the box length, divisions, depth and shrink factor as described in Section 3.2.4. The LEP and LTP modes, combined with information from the position error and percentage error reduced figures were once again used in order to narrow down the thresholds in terms of reducing starting error as well as convergence. There were two main ideas when developing these parameters. The first idea uses depth to quickly shrink the final quantization error. The final quantization error is the distance between positions at the final depth. This idea searches at many depths while the

Table 4.10: Final parameters used for the PPD algorithm. This set uses depth as the vehicle to search a large volume.

| Parameter | Value | Derived Parameter | Value |
|---|---|---|---|
| **Box Length** | 2 (m) | **Final Quantization** | 3.276 mm |
| **Divisions** | 3 | **Positions Per Depth** | 65 |
| **Depth** | 7 | **Total Positions** | 455 |
| **Shrink Factor** | 1.6 | | |

number of divisions is small. This creates a small final quantization error at the last depth but still covers a large search area at the early depths. The second idea uses many divisions with a little depth. To achieve a similar quantization error, without changing the box length, the number of search positions will exponentially increase. For example, the parameters in Table 4.10 are the final parameters used, and Table 4.11 is a set of test parameters with the same final quantization and box length but only one depth. The second set of parameters has six orders of magnitude more positions to search, and thus takes much longer to complete. However, the second set of parameters is a more complete search of the box volume and mitigates parts of local minimum problem that will be described later in Section 4.3.6. This thesis uses the first set of parameters to obtain position estimates quickly with minimal final quantization error. In the simulation, the local minimum problem was not observed enough to switch to the second set of parameters. These parameters were used in simulation of flight 1 pass 1 to obtain the error statistics in Table 4.12 as shown in Figure 4.14. To determine the accuracy of searching, a sensitivity analysis was performed on the PPD method of perturbing.

*4.3.5 PPD Sensitivity.* In order to determine the expected errors of the PPD algorithm a sensitivity analysis was performed to determine how sensitive the RRE is to minor changes in position. The tanker model was given a position and orientation typical of the refueling envelope and then perturbed from that position to determine sensitivity. First, the tanker position was shifted in one axis at a time

Table 4.11: Sample set of parameters for the PPD algorithm. This set uses divisions as the vehicle to search a large volume. Since there is only one depth, the Shrink Factor is not applicable. This set was not used in final analysis due to the large number of positions to search.

| Parameter | Value | Derived Parameter | Value |
|---|---|---|---|
| **Box Length** | 2 (m) | **Final Quantization** | 3.278 mm |
| **Divisions** | 975 | **Positions Per Depth** | 929714177 |
| **Depth** | 1 | **Total Positions** | 929714177 |
| **Shrink Factor** | N/A | | |

Table 4.12: PPD Method error statistics for a simulated run of flight 1 pass 1. Axis in $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | 0.082 cm | 12.349 cm | 12.331 cm |
| **y axis** | 0.066 cm | 1.371 cm | 1.370 cm |
| **z axis** | -0.036 cm | 0.631 cm | 0.631 cm |
| **MRSE** | 12.423 cm | | |
| **LiDAR Data** | Simulated Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup | | |

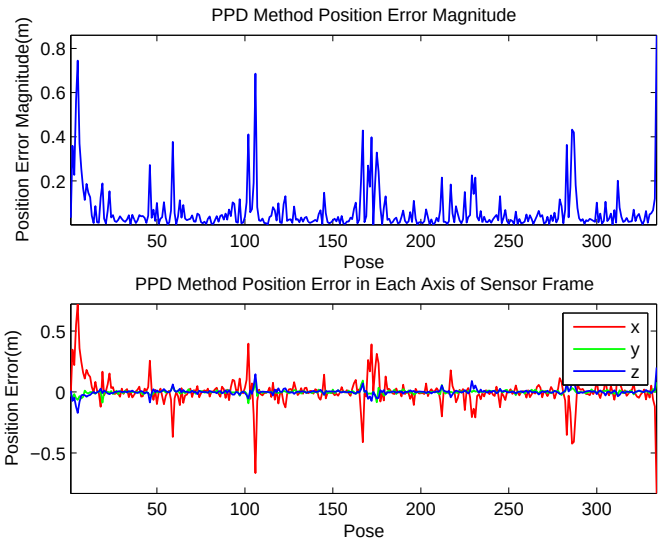


Figure 4.14: PPD error with simulated LiDAR measurements for flight 1 pass 1, axis in $l-frame$.

and the SRRE was recorded as shown in Figure 4.15. The SRRE creates a valley at the center and slopes up as it moves away from the center. The slope is a result of the measurements slowly losing pairs and increasing in RMS error. The SRRE slopes most quickly in the $z$ axis due to missing left wing measurements as the tanker moves up or down. The $x$ and $y$ axes have a smaller change in SRRE, because measurements still pair as the $x$ and $y$ axes are varied but the RMS error becomes slowly worse. The position was also varied in two axes at once as shown in Figure 4.16 and 4.17. The lack of visibility in the $x$ and $y$ axes is also apparent in Figure 4.16 where the surface is mostly flat except for the one corner. Slicing in $y$ and $z$ or the $z$ and $x$ planes as shown in Figure 4.17 reveals that the $z$ axis has much more visibility as it creates a valley in both plots. The visibility is best in the $z$ axis because of the large wing. The $y$ axis only has a small section of the engine pods to provide visibility, while the $x$ axis has no such face to provide visibility. This same characteristic was observed in the MBI algorithm in Section 4.3.2 where the vertical axis was the first to narrow down. This means the visibility issue is independent of the method used, but stems from the scan area and LiDAR setup.

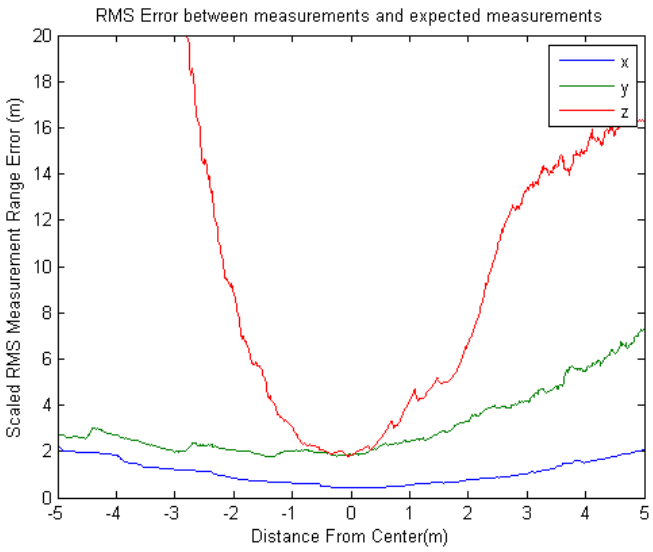


Figure 4.15: SRRE determined from varying the position of the tanker in each axis in the $b_{receiver}-frame$.

*4.3.6 PPD Characteristics.* Unlike the MBI Algorithm, the total runtime of the PPD algorithm can be directly determined before the simulation is started. Also each search position at a set depth can be done in parallel to further increase speed. This makes the PPD algorithm ideal for a time constrained problem.

A similarity with the MBI Algorithm is the local minimum problem but for different reasons. There are three different ways for the PPD Algorithm to converge on a local minimum instead of the global minimum. The first is if the global minimum is not within the search volume defined by the box length. This problem occurs if the dynamics are too great and the true position moves more than a box length away from the initial position. This can be solved by increasing the box length to a suitable size for the specific dynamics situation. The PPD method may also not converge on the global minimum even if it is within the box length.

The second way the PPD algorithm can arrive at a local minimum is if during one of the depth searches the position with the smallest SRRE does not lead to the global SRRE minimum. For example, suppose after searching the first depth there are two points with low SRRE compared to the rest of the search points but the two

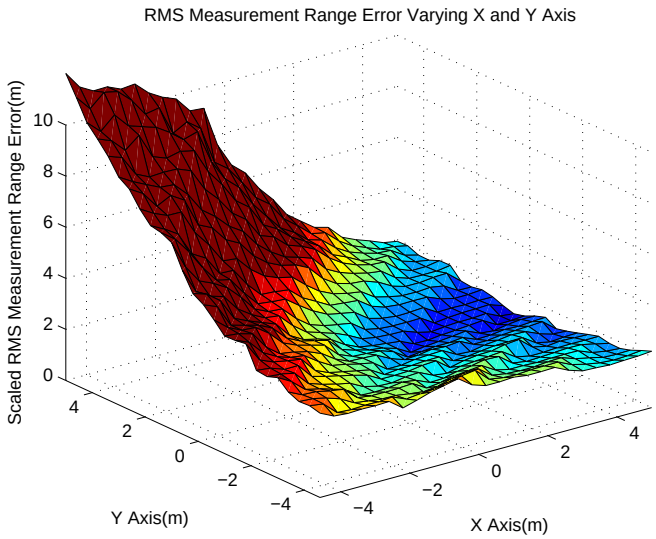


Figure 4.16: SRRE determined by varying the position of the tanker in the x and y axis of the $b_{receiver} - frame$.

search points are far apart from each other. There can be only one global SRRE minimum, and the search spaces for these two points is mutually exclusive. Thus, if the point with lower SRRE at this depth does not contain the global SRRE minimum, the algorithm will not find the global minimum.

Another problem with finding the true position is that it relies on SRRE as a metric to determine the position with smallest position distance error. This method will only find the true position if that is also the position with the smallest SRRE. Figure 4.18 shows a correlation between the position distance error and the SRRE from a simulated run of flight 1 pass 1. Thus, if the true position is not the position with the smallest SRRE, the algorithm will finish on a position that is not the true position.

## *4.4 Initial Position Estimates*

After the simulated LiDAR data position estimates were used to determine the parameters of the two algorithms the simulated LiDAR measurements was replaced by the actual LiDAR measurements from the test flight. Again, the truth position

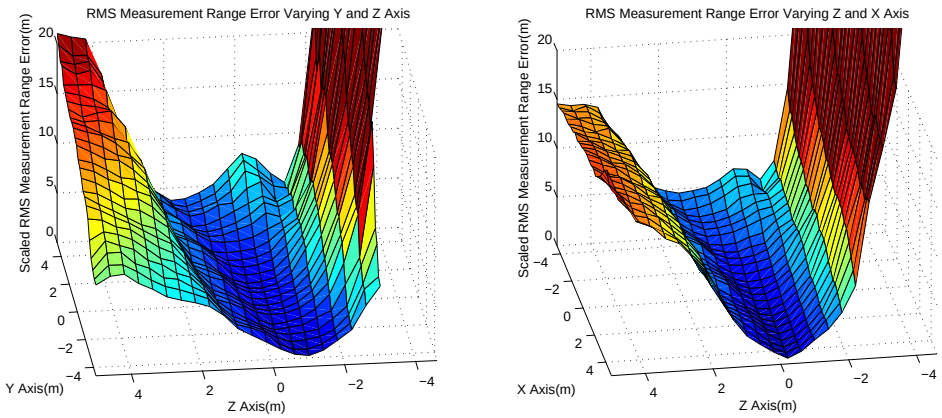


Figure 4.17: Left Figure: SRRE varying the position of the tanker in the y and z axis of the $b_{receiver}-frame$.
Right Figure: SRRE varying the position of the tanker in the z and x axis of the $b_{receiver}-frame$.
Each figure shows sensitivity to changes in the z axis, but comparatively less sensitivity to changes in the $x$ and $y$ axis.

and LiDAR scans were sampled at one position/scan for every second. The algorithms were tested in order to prove the reliability of the parameters selected from analysis and determine the current accuracy of the algorithms with real data. Flight 1 pass 1 was selected as the first data set to analyze. Using the standard parameters in Table 4.8 and the LEP as the starting point, the position error in Table 4.13 was obtained. Next the PPD algorithm was used to create position estimates for flight 1 pass 1. Using the standard parameters in Table 4.10 and the LEP as the starting point, the position error results in Table 4.14 were obtained. In both of the algorithms the MRSE error is at least six times worse than the simulation results shown in Tables 4.9 and 4.12. In order to diminish this discrepancy between the simulation and actual measurements possible sources of error were identified and mitigated.

## *4.5 Modeling Error Corrections*

Many different possible sources of error were identified as possible causes for the discrepancy between simulated and actual measurements position error. The first source of error was the tanker model accuracy in flight. The second error source was

Table 4.13: MBI Algorithm error statistics for flight 1 pass 1 with actual LiDAR data. Axis are in the $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -21.800 cm | 25.177 cm | 33.275 cm |
| **y axis** | -4.954 cm | 3.766 cm | 6.219 cm |
| **z axis** | -53.341 cm | 4.588 cm | 53.537 cm |
| **MRSE** | 63.342 cm | | |
| **LiDAR Data** | Actual Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup | | |

Table 4.14: PPD Algorithm error statistics for flight 1 pass 1 with actual LiDAR data. Axis are in the $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -10.879 cm | 35.690 cm | 37.260 cm |
| **y axis** | -4.826 cm | 7.275 cm | 8.721 cm |
| **z axis** | -55.692 cm | 7.179 cm | 56.151 cm |
| **MRSE** | 67.951 cm | | |
| **LiDAR Data** | Actual Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup | | |

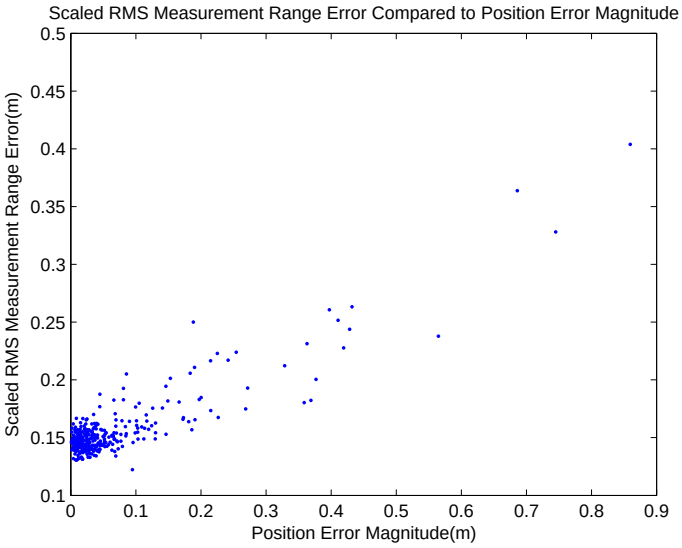


Figure 4.18: PPD algorithm showing the correlation between SRRE and position error of simulated run of flight 1 pass 1.

the LiDAR boresight error. This is the error induced by incorrectly boresighting the LiDAR to the aircraft body frame. Another source of error is bad measurements from the LiDAR due to the sun as described in Section 4.1.5. These were the sources of errors selected to reduce in order to give a more accurate estimate of the position of the tanker aircraft.

*4.5.1 Tanker Model Error.* The first error observed is the deflection of the tanker wings in flight. As planes fly the wings deflect upwards under load. This deflection is dependent on the amount of fuel in the wings and the amount of lift being provided from each wing. This deflection was not captured in the current model of the tanker. The current model was created with a laser scanning system that scanned the tanker on the ground. In order to correct this error, the model of the tanker was modified to account for the wing deflection in flight.

During the test flight the tanker had approximately the same amount of fuel on board and in the refueling position each wing provided the same amount of lift. From these assumptions a static adjustment was applied equally to both wings. In order to determine this deflection, the pictures taken from the Proscillica camera in flight

were overlayed on images rendered using the tanker model with the model located at the true position. It immediately became clear that the wings did flex upward during the flight. A correction was applied to the wings in order to flex them upward. In the $b_{tanker}-frame$, the $z$ axis of each wing was decreased(shifted upward) according to the $y$ distance from the center of the plane. The wing was only deflected if it was a certain $y$ distance from the center of the plane. In order to not accidentally deflect the tail a maximum distance in the $x$ axis was added as well. The equation to deflect the wing is as follows:

$$z_{new} = z_{old} - w_l(y - d_n)^{w_e} \tag{4.1}$$

where $y$ is the distance in the $y$ axis from the center of the aircraft, $d_n$ is the minimum $y$ axis distance from the center of the plane before the correction is applied, $w_e$ and $w_l$ are the wing deflection exponent and linear values respectively that were determined through trial and error by comparing the overlay of pictures from the camera compared to the predicted rendered image as shown in Figure 4.19. Thus the wing deflection in flight was corrected.

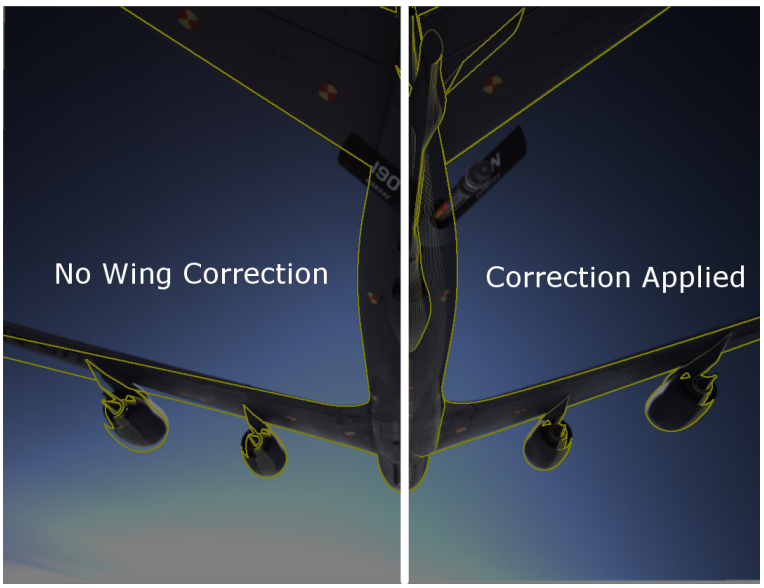


Figure 4.19: KC-135 with outline overlay of model plane overlayed on picture taken by Proscillica camera. Left side shows plane without wing correction. Right side shows overlay after wing deflection correction has been applied to the tanker model.

*4.5.2 LiDAR Boresight Correction.* The boresight of the LiDAR with respect to the aircraft body frame is not a perfect process. Including the errors from the actual method described in Section 3.4.4, there is also error in the what is considered the actual aircraft body frame. For example, the method described in Section 3.4 boresights the LiDAR with respect to a frame that can be measured. That frame then must be referenced to the plane body frame. The method to determine the plane body frame also contains errors. Thus, with these two errors added together there can be significant error introduced in the boresight of the LiDAR, because even small errors in boresight angles lead to large distance errors on the scanned target.

In order to fine tune the boresight, the actual LiDAR measurements were compared to the tanker model at the true relative position. The corrections to apply are an additional yaw, pitch, and roll applied as Euler angles to the orientation of the LiDAR. At first, the error was corrected manually by modifying the Euler angles of the LiDAR to better fit the measurements from the LiDAR to the tanker model. Next, a simulation was set up to compute SRRE at different combination of Euler angles. The different combinations of Euler angles were determined by incrementing each one by 0.05 degrees within a window of $\pm 2$ degrees from the manually determined Euler angles. To make sure the combination with the smallest SRRE was not specific to only one pose, the SRRE was determined for two poses. The SRRE from each was normalized from 0 to 1 by the smallest and largest SRRE of that pose and then added together to obtain a metric with a minimum of zero and max of two. The Euler angles that minimized this metric were chosen as the final Euler angle correction. In order to determine if the LiDAR boresight corrections and tanker model correction were applicable for all the flights and passes, a comparison of the corrected errors was completed.

*4.5.3 Error Comparison.* To determine the amount of error reduced for each modeling error correction, four different passes with all the combinations of corrections were performed on flight 1 pass 1. In order to reject poses where the

Table 4.15: Comparison of different modeling error corrections. Statistics are for the RRE of flight 1 pass 1 at the true position.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **No Correction** | 74.943 cm | 6.038 cm | 75.184 cm |
| **Tanker Model Correction** | 61.674 cm | 5.566 cm | 61.924 cm |
| **LiDAR Boresight Correction** | 56.050 cm | 5.822 cm | 56.350 cm |
| **All Corrections** | 40.329 cm | 7.706 cm | 41.055 cm |

measurements were not reliable, only poses with at least 100 measurements and 100 predicted measurements were used in the comparison of the model error corrections. A comparison of the RRE of all modeling error corrections is shown in Figure 4.20 with statistics in Table 4.15. The tanker model correction always reduced the RRE for this run compared to no corrections. The boresight correction also almost always reduced the RRE compared to no corrections. When both corrections are applied at the same time, the RRE was again almost always reduced for this run when compared to the no corrections case.

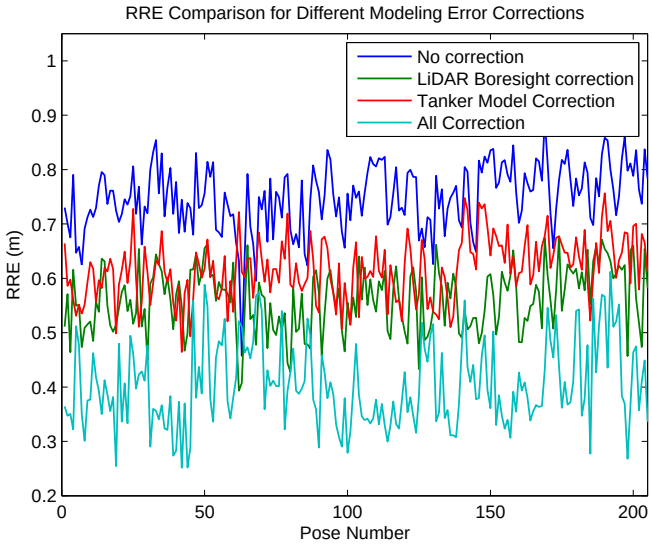


Figure 4.20: Comparison of different modeling error corrections applied. Poses with under 100 measurements or predicted measurements were not used in the comparison and were removed.

*4.5.4 LiDAR Filtering.* As described in Section 4.1.5, the sun corrupted the LiDAR measurements. In order to filter these measurements characteristics of the incorrect measurements were examined. One characteristic to examine was the pulse width reported by the Ibeo LUX 8L. The pulse width for beams created by the sun were usually smaller than the pulse width of beams from the tanker as shown in Figure 4.21. Also when the range of measurements created by the sun was past the tanker they generally grew much more sparse compared to the tanker points. Sun measurements close to the LiDAR however had about the same sparsity as the tanker measurements. However a minimum range for the tanker can be declared as the aircraft maintained a certain separation so as not to collide. Another characteristic is of the measurements from the tanker. Since the same section of the tanker was scanned in each pass, the variance of the tanker measurement point cloud was very predictable. The Ibeo LUX 8L can also return multiple beams for each azimuth and elevation, but only one echo hits the tanker, so any other echoes must be filtered. Using these characteristics a filter was created to remove the sun measurement errors as shown in Figure 4.22.

The minimums used by the filter are shown in Table 4.16. Any beam with characteristics under any of these values is removed. In terms of the echo all echoes after the two closest are rejected. Next any beams that still have echoes are examined and the one that has the lower range is rejected. This removes the echoes that were created by the nose cone glass and keeps only echoes that struck the tanker. Next if the variance in the $x$ axis of the $l-frame$ is above the max value then the radius neighbor filter is applied to remove sparse measurements. The radius neighbor filter examines each measurement and counts the neighbors within a radius. If the number of neighbors does not meet a minimum, that point is rejected. The points are removed at the end of the filter so the order of determining neighbors will not effect the outcome. This filter is applied to each scan in order to remove incorrect measurements created by the sun. An example of the filter applied to a sun-corrupted scan is shown in Figure 4.23. The filter does remove some measurements from the

tanker, but in general it removes measurements created by the sun or nose cone glass. With these corrections applied, the pose estimates were once again determined with the two methods.

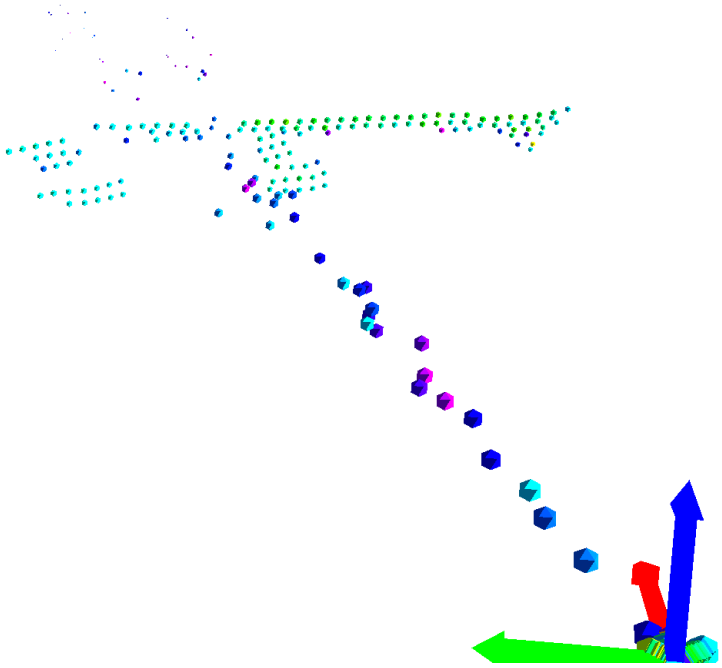

Figure 4.21: LiDAR scan with sun causing measurement errors. Measurements represented by spheres with the color showing the pulse width of the beam as reported by the Ibeo LUX 8L. Beams that strike the tanker are green in color while the beams created by the sun are blue and purple. Measurements created by the sun are also more sparse at farther ranges compared to the tanker measurements. Axis represents the $l-frame$ with the $x, y, z$ corresponding to the red, green, and blue arrows.

## *4.6 Final Position Estimates*

With the corrections of the tanker model and boresight applied, the MBI and PPD algorithms were once again used to generate a relative position estimate for flight 1 pass 1. With the corrections applied, the MRSE of both algorithms decreased compared to the original position estimates as shown in Tables 4.17 and 4.19. The decrease in MRSE was attributed mostly to a decrease in the mean because the corrections applied removed a bias from the position estimates. Also to test the filtering capabilities position estimates for flight 2 pass 1 for both algorithms were

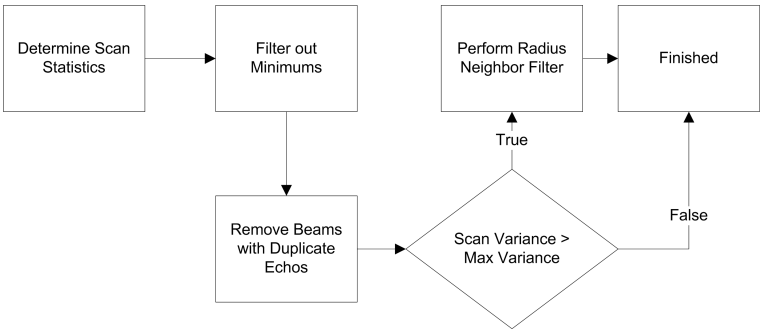


Figure 4.22: Flowchart showing steps used to filter LiDAR scans.

Table 4.16: Parameters used by the filter to remove measurement errors created by the sun.

| Minimum Parameters | Value | Neighbor Filter Parameter | Value |
|---|---|---|---|
| **Minimum Range** | 17 m | **Maximum Variance** | 550 $m^2$ |
| **Minimum Pulse Width** | 0.65 m | **Neighbor Radius** | 2 m |
| **Minimum Echo** | 2 | **Neighbors Required** | 3 |

also computed as shown in Tables 4.18 and 4.20. While the modeling corrections did reduce the MRSE of both algorithms compared to the no modeling corrections case, the MRSE of the simulated LiDAR position estimates was still about three times lower in the case of the PPD algorithm and ten times lower in the MBI algorithm. With all the identified modeling errors corrected, other reasons for the discrepancy could not be found. Instead, an analysis to determine if this position estimate error was as low as the current setup and collected measurements would allow. To start this analysis, the RRE was observed for three different positions for all of the times a position estimate was computed. The first position was the starting point of the position estimate which was the last estimated position. The second position was the final position estimate computed by the algorithm. The third position was the true position given by the truth data. The RRE of these three positions was compared for the computed position estimate of both algorithms as shown in Tables 4.21 and 4.22. In both algorithms the RRE was on average lower for the final estimated position compared to true position. Since this metric is used in the PPD algorithm to determine the final estimate, the

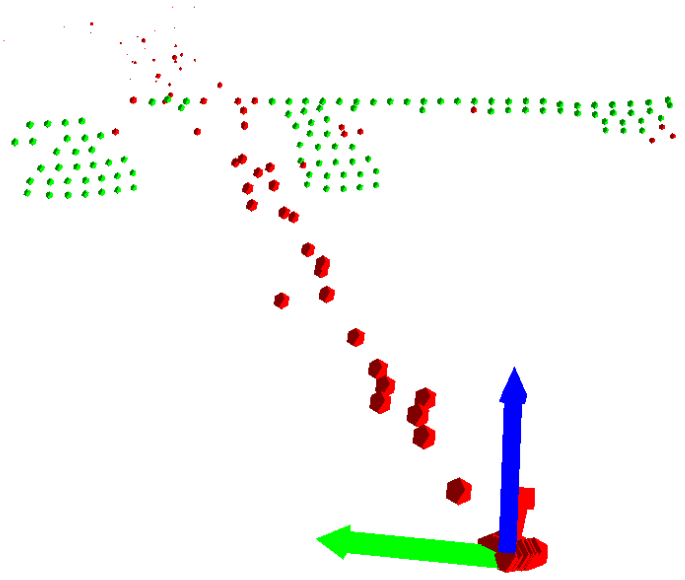

Figure 4.23: LiDAR scan with sun causing measurement errors with filter applied. Green spheres are measurements kept by the filter while the red spheres are rejected by the filter. Some points on the plane are marked as red but overall filter removes mainly measurements caused by the sun.

PPD algorithm cannot arrive at the true position with this metric. The MBI method, while not using the metric, was still able to consistently arrive at positions with lower RRE than the true position. This indicates that these algorithms have used all the available information from these measurements and any further MRSE reduction must be accomplished by correcting lever arms and orientations at the preflight stage.

The lever arms and orientations are crucial in determining the MRSE of the algorithms. This is because the algorithms in essence can only determine the position of the tanker in the $l-frame$. That position is then transformed into the $b_{receiver}-frame$ with the lever arms and orientations so it can be compared to the truth data. It is possible the algorithms are accurately determining the position in the $l-frame$, but the transformation to the $b_{receiver}-frame$ is incorrect. While the LiDAR boresight was adjusted post-flight as outlined in Section 4.5.2, no method to correct the other orientations is available, since the flight test has been completed and all equipment disassembled. It would be useful to obtain statistics on the lever arms

Table 4.17: MBI Algorithm error statistics for flight 1 pass 1 with actual LiDAR data and all model corrections. Axis are in the $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -17.365 cm | 27.883 cm | 32.813 cm |
| **y axis** | -4.463 cm | 3.482 cm | 5.658 cm |
| **z axis** | -4.590 cm | 5.058 cm | 6.825 cm |
| **MRSE** | 33.989 cm | | |
| **LiDAR Data** | Actual Flight 1 Pass 1 | | |
| **LiDAR Setup** | Test Flight Setup with All Model Corrections | | |

Table 4.18: MBI Algorithm error statistics for flight 2 pass 1 with actual LiDAR data and all model corrections. Axis are in the $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -25.123 cm | 33.062 cm | 41.486 cm |
| **y axis** | -0.449 cm | 4.605 cm | 4.620 cm |
| **z axis** | -6.681 cm | 6.405 cm | 9.248 cm |
| **MRSE** | 42.755 cm | | |
| **LiDAR Data** | Actual Flight 2 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup w/Model Corrections | | |

and orientations in order to determine statistics on the error they could contribute, but such measurements are not available because many lever arms and orientations were measured by hand with rulers and compasses. However, the relative positions used as truth have an associated 1-$\sigma$ value shown in Table 4.23 which were well below the magnitude of the errors observed in the final positions estimates from both algorithms. Despite the relative position statistics, to correct or bound the lever arm and orientations errors, new methods to measure the lever arms and orientations must be applied at the preflight stage. Another possibility to explain these errors is that they are inherent in the specific LiDAR setup used in the test flight. A different LiDAR setup may aid to reduce the MRSE by increasing visibility or providing visibility in different axes. The next section simulated a different LiDAR setup in order to evaluate this hypothesis.

Table 4.19: PPD Algorithm error statistics for flight 1 pass 1 with actual LiDAR data and all model corrections. Axis are in the $b_{receiver} - frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -8.246 cm | 35.401 cm | 36.297 cm |
| **y axis** | -2.913 cm | 6.864 cm | 7.447 cm |
| **z axis** | -7.686 cm | 4.869 cm | 9.095 cm |
| **MRSE** | 38.153 cm | | |
| **LiDAR Data** | Actual Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup w/Model Corrections | | |

Table 4.20: PPD Algorithm error statistics for flight 2 pass 1 with actual LiDAR data and all model corrections. Axis are in the $b_{receiver} - frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -11.435 cm | 33.649 cm | 35.492 cm |
| **y axis** | 0.546 cm | 8.752 cm | 8.756 cm |
| **z axis** | -8.112 cm | 7.059 cm | 10.747 cm |
| **MRSE** | 38.103 cm | | |
| **LiDAR Data** | Actual Flight 2 Pass 1 | | |
| **LiDAR Setup** | Test Flight Setup w/Model Corrections | | |

Table 4.21: MBI Algorithm RRE comparison at different positions for flight 1 pass 1.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **RRE Initial Position** | 57.350 cm | 23.736 cm | 62.055 cm |
| **RRE Final Position** | 37.616 cm | 11.277 cm | 39.265 cm |
| **RRE True Position** | 41.710 cm | 8.687 cm | 42.602 cm |

Table 4.22: PPD Algorithm RRE comparison at different positions for flight 1 pass 1.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **RRE Initial Position** | 57.287 cm | 23.459 cm | 61.891 cm |
| **RRE Final Position** | 28.591 cm | 9.491 cm | 30.120 cm |
| **RRE True Position** | 41.710 cm | 8.687 cm | 42.602 cm |

Table 4.23: Relative position truth data 1-$\sigma$ statistics for flight 1 pass 1. Statistics are only for positions where receiver and tanker were simulating refueling. Axis are in the $e-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis 1-$\sigma$** | 2.028 cm | 0.133 cm | 2.032 cm |
| **y axis 1-$\sigma$** | 4.112 cm | 0.455 cm | 4.137 cm |
| **z axis 1-$\sigma$** | 3.302 cm | 0.015 cm | 3.302 cm |
| **MRSE** | 5.670 cm | | |

## *4.7 Simulate Other LiDAR Setup*

This section determines if error in the previous simulated LiDAR position estimates can be reduced by changing the scan pattern and orientation of the LiDAR (referred to as the LiDAR setup). Specifically, the field of view of the LiDAR will be increased to observe more of the tanker, and the LiDAR will be directed to the view the tanker fuselage instead of focusing on the wing. This modification will be simulated using the same method as the previous estimates using simulated LiDAR measurement in Section 4.3. This analysis assumes reduction of the simulated position estimate from one setup to another will translate to real world error reduction if another test flight were to be conducted with the new setup.

*4.7.1 Simulate Custom LiDAR.* To simulate a new LiDAR (called "Custom LiDAR"), the scan pattern as well as the placement of the LiDAR was changed. In analysis of the test flights, the wings were often missed entirely, and quite often only engine pods appeared in the measurements. In fact, only engine pods were scanned for nearly all of flight 4. Custom LiDAR was oriented to aim at the fuselage of the tanker. The fuselage is a much larger part of the plane and is more predictable, because roll does not rotate the fuselage in the same way the wings will rotate out of view. To view the fuselage, the LiDAR was pitched 30 degrees upward and rolled 90 to the side. The elevation range of the LiDAR was also increased to scan the wings of the plane. This gives a scan that views much more area of the plane compared to

the Ibeo LUX 8L LiDAR. This increased scan area produces may more scan points to determine an accurate position. The final scan area scanned elevations from -19 to 20 degrees in increments of one degree, with azimuths of -22 to 22 degrees in increments of one degree. Custom LiDAR had the same range and standard deviation as the Ibeo LUX 8L. With the Custom LiDAR setup position estimates were generated with the MBI and PPD algorithms.

*4.7.1.1 Custom LiDAR Position Estimates.* The Custom LiDAR was used to create a simulated set of measurements for flight 1 pass 1. The simulated set was corrupted to the same standard deviation as the Ibeo LUX 8L and used as measurements to determine position estimates for the MBI and PPD algorithms. The MBI position estimate MRSE, show in Table 4.24, is smaller than the MRSE from the simulated measurements in the flight test LiDAR setup shown in Table 4.9. The more important difference is what axis the error is from. While both setups had most of the MRSE in the $x$ axis, the Custom LiDAR setup had less $y$ axis RMS error. The RMS error in the $z$ axis however became worse. The PPD algorithm position estimate shown in Table 4.25 show similar results compared to the simulated measurements with the flight test LiDAR setup shown in Table 4.12. Thus, the new setup decreased the MRSE and it supports the idea that the LiDAR setup can change the RMS error in each axis and thus the visibility. This can be used to design a LiDAR setup for a particular situation. For example the $x$ and $z$ axis may be very important to prevent planes from colliding, thus error in that channel may be prioritized over error in other channels. However, the Custom LiDAR increases the RMS error in the $z$ axis. This was unexpected as the Custom LiDAR was aimed at the fuselage to give a large surface perpendicular to the $z$ axis to increase visibility in the $z$ axis as shown in Figure 4.24. This increase in error can be attributed to the new orientation of the LiDAR. When the LiDAR was pitched upward some of the range variance was moved from the $x$ axis to the $z$ axis. Since the simulation corrupts the range measurements, this led to a worse estimate in the $z$ axis which was now more in line with the range

Table 4.24: MBI Algorithm error statistics for flight 1 pass 1 using the Custom LiDAR setup. Axis are in the $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -2.136 cm | 1.812 cm | 2.800 cm |
| **y axis** | -0.083 cm | 0.424 cm | 0.432 cm |
| **z axis** | -0.920 cm | 0.352 cm | 0.985 cm |
| **MRSE** | 2.999 cm | | |
| **LiDAR Data** | Simulated Flight 1 Pass 1 | | |
| **LiDAR Setup** | Custom LiDAR Setup | | |

Table 4.25: PPD Algorithm error statistics for flight 1 pass 1 using the Custom LiDAR setup. Axis are in the $b_{receiver}-frame$

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | 0.638 cm | 10.595 cm | 10.598 cm |
| **y axis** | -0.003 cm | 0.782 cm | 0.780 cm |
| **z axis** | 0.005 cm | 0.710 cm | 0.709 cm |
| **MRSE** | 10.650 cm | | |
| **LiDAR Data** | Simulated Flight 1 Pass 1 | | |
| **LiDAR Setup** | Custom LiDAR Setup | | |

axis of the LiDAR. However, this type of scan should provide improved visibility in the orientation of the tanker compared to the flight test LiDAR setup.

*4.7.2 Simulated Attitude Estimates.* The Custom LiDAR and flight test LiDAR setup were both used to compute attitude as well as position to determine if the Custom LiDAR provides improved attitude visibility. Previously this thesis made the assumption that attitude was provided by the EGIs and the LiDAR was only providing a position estimate. However, the MBI algorithm has an ability to determine attitude, because it was derived from ICP algorithm which can determine rotations to fit point clouds together. Thus, a modification was made to the MBI algorithm where the rotation determined from the ICP part of the algorithm was included in the position estimate to make it a pose estimate. This modification was applied in Section 3.2.3 where all steps of the ICP algorithm were completed.

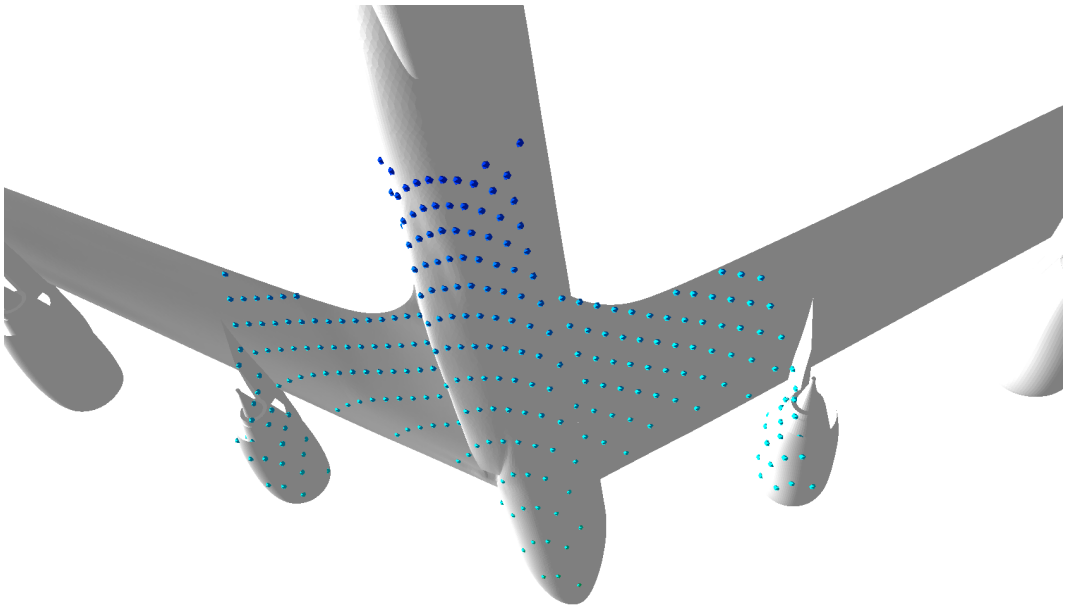

Figure 4.24: Custom LiDAR scan on the tanker. Blue spheres are measurement points returned by the Simulated LiDAR. The scan points are mostly flat but the sides of the engine pods provide vertical visibility.

The Custom LiDAR setup was used to determine pose estimates for flight 1 pass 1 including attitude as shown in Table 4.26. The Custom LiDAR setup was compared to error statistics with the Flight Test LiDAR setup with the same MBI Algorithm modification to determine attitude as shown in Table 4.27. The Custom LiDAR setup maintains a lower MRSE and lower RMS error in each Euler angle compared to the flight test LiDAR setup. In fact, the MRSE of the Custom LiDAR while determining attitude is lower than the MRSE of the Custom LiDAR when it had attitude from the EGIs. Thus, the Custom LiDAR setup provides a pose estimate with lower MRSE and lower RMS error in Euler angles compared to the flight test LiDAR setup. However the flight test LiDAR still maintained a MRSE within centimeters of the simulations that used attitude from the EGIs, thus more analysis was performed to attempt to replicate the results with actual flight test measurements, as described in the next section.

Table 4.26: MBI Algorithm error statistics for flight 1 pass 1 using the Custom LiDAR setup and determining the attitude of the tanker. Axis in $b_{reciever}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | 0.170 cm | 2.211 cm | 2.207 cm |
| **y axis** | -0.086 cm | 1.316 cm | 1.312 cm |
| **z axis** | -0.266 cm | 0.861 cm | 0.897 cm |
| **MRSE** | 2.720 cm | | |
| **Yaw** | -0.030 MRAD | 0.496 MRAD | 0.494 MRAD |
| **Pitch** | 0.224 MRAD | 0.432 MRAD | 0.484 MRAD |
| **Roll** | -0.011 MRAD | 0.574 MRAD | 0.572 MRAD |
| **LiDAR Data** | Simulated Flight 1 Pass 1 | | |
| **LiDAR Setup** | Custom LiDAR Setup | | |

Table 4.27: MBI Algorithm error statistics for simulated flight 1 pass 1 using the Flight Test LiDAR setup and determining the attitude of the tanker. Axis in $b_{reciever}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | 1.737 cm | 4.339 cm | 4.654 cm |
| **y axis** | -3.438 cm | 3.023 cm | 4.568 cm |
| **z axis** | -1.000 cm | 1.537 cm | 1.827 cm |
| **MRSE** | 6.772 cm | | |
| **Yaw** | -2.378 MRAD | 2.054 MRAD | 3.135 MRAD |
| **Pitch** | 0.758 MRAD | 1.174 MRAD | 1.393 MRAD |
| **Roll** | 0.231 MRAD | 1.006 MRAD | 1.028 MRAD |
| **LiDAR Data** | Simulated Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup w/Model Corrections | | |

## *4.8 Actual Attitude Estimates*

Since simulation results showed promise of determining attitude of the tanker actual data was used to determine pose estimates for flight 1 pass 1. The modified MBI algorithm when estimating pose for flight 1 pass 1 shown in Table 4.28, had a MRSE 10 times worse compared to the position estimates from the normal MBI algorithms shown in Table 4.17. In order to decrease the MRSE the sampling frequency of pose estimates was increased from one position/scan per second to ten positons/scans per second. This was done to give the algorithm a starting point closer to the true position, since less time had elapsed between estimates. This would not make any difference in previous position estimation algorithms, because the convergence analysis done in Section 4.3.2 revealed that starting position would not effect the solution within 5m of the true solution. However, the modified MBI algorithm now determines attitude and thus that analysis does not apply to this modified MBI algorithm. MBI algorithm MRSE was decreased by sevenfold as shown in Table 4.29 when the sampling frequency was increased to 10Hz. This shows that the modified MBI algorithm is converging on local minimums, and more accurate initial positions will mitigate the local minimum convergence. However, the modified MBI algorithm is far from tracking the Euler angles. An analysis of the ratio of Euler angle error reduced shown in Table 4.30 shows that the algorithms on average cannot provide a better estimate of attitude, rather, it only keeps the attitude close enough to not totally diverge. A new method to determine initial position had to be determined in order to provide better Euler angle estimates.

*4.8.1 PPD + MBI Algorithm.* In order to reduce the initial error for the modified MBI algorithm, and provide a better estimate of attitude, the PPD + MBI algorithm was created. The PPD + MBI algorithm uses the PPD algorithm with the last attitude estimate to provide a position estimate for the modified MBI algorithm. With this improved position estimate, the modified MBI algorithm then determines the pose of the tanker. Since the PPD part of the algorithm provides a better initial

Table 4.28: MBI Algorithm error statistics for flight 1 pass 1 using the Flight Test LiDAR setup and determining the attitude of the tanker. Axis in $b_{reciever}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | -309.307 cm | 149.732 cm | 343.316 cm |
| **y axis** | 87.553 cm | 63.152 cm | 107.767 cm |
| **z axis** | -247.303 cm | 45.163 cm | 251.353 cm |
| **MRSE** | 438.928 cm | | |
| **Yaw** | 10.286 MRAD | 17.877 MRAD | 20.548 MRAD |
| **Pitch** | 209.080 MRAD | 11.809 MRAD | 209.410 MRAD |
| **Roll** | 122.467 MRAD | 27.026 MRAD | 125.385 MRAD |
| **LiDAR Data** | Actual Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup w/Model Corrections | | |

Table 4.29: MBI Algorithm error statistics for flight 1 pass 1 and determining the attitude of the tanker. Position estimates were produced at 10Hz instead of the usual 1Hz. Axis in $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | 28.535 cm | 21.017 cm | 35.435 cm |
| **y axis** | -44.911 cm | 18.982 cm | 48.755 cm |
| **z axis** | 24.048 cm | 18.204 cm | 30.157 cm |
| **MRSE** | 67.395 cm | | |
| **Yaw** | -17.025 MRAD | 8.530 MRAD | 19.041 MRAD |
| **Pitch** | -22.141 MRAD | 11.737 MRAD | 25.057 MRAD |
| **Roll** | -24.003 MRAD | 9.919 MRAD | 25.970 MRAD |
| **LiDAR Data** | Actual Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup w/Model Corrections | | |

Table 4.30: MBI Algorithm Euler angle error reduction for pose estimates in Table 4.29. Values below 1 decreased error while values above 1 increased error e.g. 2 means error was doubled from initial estimate.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **Yaw** | 1.251 | 5.117 | 5.265 |
| **Pitch** | 1.068 | 0.601 | 1.226 |
| **Roll** | 1.060 | 0.450 | 1.151 |

Table 4.31: PPD + MBI Algorithm error statistics for flight 1 pass 1. Axis in $b_{receiver}-frame$.

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **x axis** | 11.267 cm | 22.003 cm | 24.679 cm |
| **y axis** | -37.315 cm | 19.740 cm | 42.196 cm |
| **z axis** | 12.365 cm | 15.557 cm | 19.847 cm |
| **MRSE** | 52.758 cm | | |
| **Yaw** | -13.992 MRAD | 10.192 MRAD | 17.298 MRAD |
| **Pitch** | -15.705 MRAD | 10.426 MRAD | 18.839 MRAD |
| **Roll** | -21.575 MRAD | 9.747 MRAD | 23.666 MRAD |
| **LiDAR Data** | Actual Flight 1 Pass 1 | | |
| **LiDAR Setup** | Flight Test LiDAR Setup w/Model Corrections | | |

position for the modified MBI, the sampling frequency of positions/scans was reduced back to the original 1Hz. Thus, the PPD + MBI provided estimates faster since the sampling frequency was 10 times slower. The PPD + MBI algorithm was used to calculate pose estimates for flight 1 pass 1 as shown in Table 4.31 with lower MRSE compared to the MRSE of the modified MBI with 10Hz sampling. The bigger difference between the PPD + MBI and the modified MBI at 10Hz is the improved attitude accuracy. The Euler Angles for pitch and roll had lower RMS error compared to the MBI Algorithm sampled at 10Hz. However, the algorithm did not reduce the error of the initial Euler angles estimate as much as the modified MBI at 10Hz as shown in Tables 4.32 and 4.30. In conclusion even the PPD + MBI cannot track the attitude Euler angles with the flight test LiDAR setup. However, if real data was obtained from using the Custom LiDAR setup and pose estimates determined with PPD + MBI, the Euler Angles tracking would more accurate as the Custom LiDAR setup had less RMS error in Euler angles compared to the Flight Test LiDAR setup as described in the previous section.

Table 4.32: PPD + MBI Algorithm Euler angle error reduction for pose estimates in Table 4.31

| Parameter | Mean | 1-$\sigma$ | RMS |
|---|---|---|---|
| **Yaw** | 2.321 | 6.481 | 6.871 |
| **Pitch** | 2.477 | 12.406 | 12.626 |
| **Roll** | 2.871 | 10.131 | 10.510 |

## *4.9 Summary*

In summary the actual measurements created a position estimate with larger MRSE compared to the simulated measurements. However, this appears to be the best estimate possible using the actual measurements, since both algorithms had similar error, and both algorithms drove the RRE below the error of the true position. The position estimates obtained a MRSE of about 40cm after all modeling corrections were applied. The field of view and orientation of the LiDAR can be modified to lower the over all MRSE or the RMS error in a desired axis.

# V. Conclusion

This thesis presents two methods for relative position estimation during aerial refueling using a scanning LiDAR. In this chapter, conclusions regarding the research and analysis are discussed. Also areas for future research and analysis are discussed.

## *5.1 Conclusions*

The position estimates using the two algorithms demonstrated using a scanning LiDAR approach to determine relative position during aerial refueling is possible. The MBI algorithm can accurately fit a model of the tanker aircraft to measurements from the LiDAR. While convergence of the MBI algorithm is not guaranteed, it was demonstrated for one pose that with an initialization inside a 10 meter box, the solution always converged on the true solution. The PPD algorithm can quickly search for a position that matches the measurements from the LiDAR to determine an accurate relative position. A LiDAR simulator is used to generate estimated measurements in order to calculate the SRRE between the estimated measurements and actual measurements.

Relative position estimates using simulated data had 4.3cm MRSE for the MBI algorithm and 12.4cm MRSE for the PPD algorithm. Relative position estimates using actual data had 33.9cm MRSE for the MBI algorithm and 39.1cm MRSE for the PPD algorithm after modeling corrections were applied. RMS error in the forward-back axis of the $b_{tanker}-frame$ is larger compared to other axes while using the flight test LiDAR setup. Position estimates using simulated data with the Custom LiDAR setup decreased the RMS error in the forward-back axis but increased the vertical axis RMS error compared to the position estimates using simulated data with the flight test LiDAR setup. Also, The MRSE for the Custom LiDAR setup was lowered to 2.99cm MRSE compared to the 4.3cm MRSE of the flight test LiDAR setup. This shows the LiDAR setup can shift RMS error from one axis to another or reduce the MRSE altogether.

Attitude and position estimates were simulated using a modified MBI algorithm with the Custom LiDAR setup and showed Euler angle RMS errors under 1MRAD and MRSE of 2.7cm. The flight test LiDAR calculated attitude and position estimates with the simulated data using the modified MBI algorithm as well, and achieved Euler angle RMS errors of about 3MRAD with a MRSE of 6.7cm. Attitude and position estimates on actual flight test data were poor and had an MRSE of 4.3m. The PPD + MBI algorithm however, obtained a MRSE of 52.7cm and Euler angle RMS errors under 25MRAD when attitude and position estimates were calculated from actual flight test data. This shows that position estimates can be calculated even if the attitude is not provided by the IMUs.

Based on the results, either algorithm has an accuracy of about 40cm MRSE for real flight test data after applying modeling corrections. The final errors are a result not from the algorithms, but rather from the lever arms and orientation calculations. With more precise lever arms and orientations the MRSE can be further reduced to achieve more accurate position estimates. Attitude and position estimates can be achieved with the PPD + MBI algorithm and result in MRSE of about 52cm with Euler angle RMS errors under 25MRAD.

## *5.2 Future Work*

This section outlines areas of future work for further research. Since the problem of automated aerial refueling using scanning LiDAR is a new field, there are many different areas of research for future analysis.

*5.2.1 Sensor Fusion.* In this thesis the LiDAR was the only sensor used to determine the relative position with help from the IMUs for attitude. Using the MBI method, an attitude can be generated without the IMUs, and then compared to the IMU attitude with a Kalman filter. This can be useful if the accuracy of the IMUs is questionable, or if integrating the sensors is not desirable. Another sensor to use for position estimates is a camera. The camera and LiDAR could be combined to

determine the position and attitude of the tanker. The LiDAR aids the camera well because a camera cannot determine range accurately while a LiDAR gives range as a raw measurement.

*5.2.2 Algorithm Improvement.* The algorithms introduced in this thesis had flaws that could be corrected. The MBI algorithm has a problem of looping between position estimates. These loops can degrade both position estimates and speed. The modes of the looping are not restricted to a set number of positions, and the exact locations cannot be guaranteed, thus some kind of looping detector is required to search for looping in a variable number of positions. Also the question of how to break the loop is not trivial. A new initial position or large random perturbation could return to loop once again.

The PPD algorithm has the problem of relying on SRRE to correlate to position error. A better metric that correlates more closely to position error could be used to determine the best position. Also, more comprehensive methods of searching such as gradient slopes could be used to find the position of the lowest SRRE more quickly. Also, the parameters put forth in this thesis are not optimal. They were determined empirically using simulated flight data, and they performed reasonably well with the actual flight data. If the position error and time constraints are known, optimized parameters can be calculated to meet these specifications.

*5.2.3 Speedup For Real Time Operation.* The current implementation of each algorithm cannot be done in real time. The position estimates were all generated post flight and the time to generate each position estimate was longer than the sampled time interval. On average, the MBI algorithm and PPD showed about the same time performance of about 12 seconds per position estimate. The PPD algorithm can be parallelized and the speed is determined by the parameters which makes it easier to speed up. However the MBI algorithm must be done sequentially, and the time taken to converge to a solution is dependent on the fit of the measurements to the model.

*5.2.4 LiDAR Setup.* This thesis argues that these algorithms were limited by the LiDAR setup used in the test flight. Since the simulated position estimates were much different compared to the position estimates with actual data, an analysis of how different setups change the error cannot be done in simulation until the simulated position estimates match position estimates with actual data. The current LiDAR simulator does not mimic a real LiDAR perfectly thus error will be introduced when the LiDAR simulator does not accurately simulate a real LiDAR scan. Ultimately, to verify a new LiDAR setup a new test flight or mock test flight must be conducted to assure the simulation results.

*5.2.5 Controlled LiDAR.* While this thesis used a scanning LiDAR, a more sophisticated LiDAR capable of being controlled could be used. A controlled LiDAR can be used to retrieve the most useful information from the tanker while ignoring measurements that repeat the same information. The reduction in measurements would speed up position estimates and provide only the most useful information. The controlled LiDAR can also be used to search for specific points on the tanker such as installed markers that reflect laser beams directly back to the sender. A LiDAR can distinguish these returns from other returns and be used to track the markers. If measurements can be directly related to points on a model, the accuracy of the position will greatly improve since registration is much easier.

# VI. Appendix A

This appendix lists a tables useful for further analysis of specific topics. The first table lists the final lever arms used for simulations. The second table includes all the test flights and passes to show properties about each pass including length and the useability of the LiDAR measurements.

Table A.1: Lever arms for sensors in the receiver. All lever arms in the $b_{reciever} - frame$.

| Sensor | Lever Arm (x,y,z) | Units |
|---|---|---|
| **GPS Antenna 1** | (-263.8,2.9,-61.7) | in. |
| **GPS Antenna 2** | (-291.6,2.5,-61.7) | in. |
| **GPS Antenna 3** | (-361.1,2.5,-61.7) | in. |
| **GPS Antenna 4** | (-436,5.5,-61.7) | in. |
| **EGI 1** | (-314.2,-15.6,-19.8) | in. |
| **EGI 2** | (-307.1,-17.3,-19.8) | in. |
| **EGI 3** | (-298.5,-15.6,-19.8) | in. |
| **Sensor Suite IMU** | (-116.3,0,-9.65) | in. |
| **Camera** | (-2.737,-0.088,-0.389) | m |
| **LiDAR** | (-2.637,-0.054,-0.226) | m |

Table A.2: Final sensor orientations for the sensors in the receiver aircraft. The quaternion rotation is with respect to the $b_{receiver}-frame$.

| Sensor | Quaternion (w,x,y,z) |
|---|---|
| **Sensor Suite IMU** | (1,0,0,0) |
| **Camera** | (0.3663031,0.6149861,0.6031746,0.3518449) |
| **LiDAR** | (0.1255817,-0.9864451,0.01746362,0.1041674) |
| **EGI 1** | (1,0,0,0) |
| **EGI 2** | (1,0,0,0) |
| **EGI 3** | (1,0,0,0) |

Table A.3: Lever arms for sensors in the tanker. All lever arms in the $b_{tanker}-frame$.

| Sensor | Lever Arm (x,y,z) | Units |
|---|---|---|
| **GPS Antenna 1** | (-390.0, -4.5, -302.0) | in. |
| **EGI 1** | (Not Listed) | in. |
| **EGI 2** | (Not Listed) | in. |
| **Tanker Model Origin** | (-130.0, 0.0, -302.0) | in. |

Table A.4: Final sensor orientations for the sensors in the tanker aircraft. The quaternion rotation is with respect to the $b_{tanker}-frame$.

| Sensor | Quaternion (w,x,y,z) |
|---|---|
| **EGI 1** | (1,0,0,0) |
| **EGI 2** | (1,0,0,0) |
| **Tanker Model** | (1,0,0,0) |

Table A.5: Description of test flights 1-4 and LiDAR data status.

| Pass | Time (s) | Description | LiDAR Data |
|---|---|---|---|
| **Flight 1** | | | |
| Pass 1 | 399 | Typical pass | OK |
| Pass 2 | 59 | Pass too short and plane too far away | Unusable |
| Pass 3 | 61 | Pass too short and plane too far away | Unusable |
| Pass 4 | 75 | Pass too short and plane too far away | Unusable |
| Pass 5 | 62 | Pass too short and plane too far away | Unusable |
| **Flight 2** | | | |
| Pass 1 | 448 | Sun in view caused measurement errors | Filtering Required |
| Pass 2 | 748 | Sun in view caused measurement errors | Filtering Required |
| Pass 3 | 428 | Sun in view caused measurement errors | Filtering Required |
| Pass 4 | 432 | Sun in view off to the right side thus no issues expected from measurement errors | OK |
| Pass 5 | 544 | Sun in view caused measurement errors | Filtering Required |
| **Flight 3** | | | |
| All Passes | 0 | Sensor time tag malfunction | Unusable |
| **Flight 4** | | | |
| Pass 1 | 616 | Plane flew higher than usual resulting in measurements of mostly engine pods | Questionable |

Table A.6: Description of test flights 5-8 and LiDAR data status.

| Pass | Time (s) | Description | LiDAR Data |
|---|---|---|---|
| **Flight 5** | | | |
| All Passes | 0 | Flight test focused on other objectives | No data |
| **Flight 6** | | | |
| All Passes | 0 | Flight operated outside area cleared for LiDAR use | No data |
| **Flight 7** | | | |
| Pass 1 | 320 | Typical pass | OK |
| Pass 2 | 176 | Sun in view caused measurement errors | Filtering Required |
| Pass 3 | 219 | Sun in view, minimal measurement errors | OK |
| Pass 4 | 321 | Sun in view, minimal measurement errors | OK |
| Pass 5 | 358 | Sun in view caused measurement errors | Filtering Required |
| **Flight 8** | | | |
| Pass 1 | 332 | Typical pass | OK |
| Pass 2 | 633 | Typical pass | OK |
| Pass 3 | 319 | Sun in view caused measurement errors | Filtering Required |
| Pass 4 | 327 | Typical pass | OK |
| Pass 5 | 625 | Sun in view caused measurement errors | Filtering Required |

# *Bibliography*


1. "Camera Calibration Toolbox for Matlab". Online, July 2010. URL `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

2. "Fitting a line in 3D". Online, February 2010. URL `http://stackoverflow.com/questions/2298390/fitting-a-line-in-3d`.

3. Adams, M.D. "Lidar design, use, and calibration concepts for correct environmental detection". *Robotics and Automation, IEEE Transactions on*, 16(6):753 –761, dec 2000. ISSN 1042-296X.

4. Air Combat Command, Public Affairs Office. "MQ-9 Reaper". *The Official Web Site of the U.S. Air Force*, January 2012. URL `http://www.af.mil/information/factsheets/factsheet.asp?id=6405`.

5. Air Combat Command, Public Affairs Office. "RQ-4 Global Hawk". *The Official Web Site of the U.S. Air Force*, January 2012. URL `http://www.af.mil/information/factsheets/factsheet.asp?id=13225`.

6. Bentley, Jon Louis. "Multidimensional Binary Search Trees Used for Associative Searching". Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC 27514, 1980.

7. Bentley, Jon Louis. "Multidimensional Divide-and-Conquer". Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA 15213, 1980.

8. Besl, P.J. and H.D. McKay. "A method for registration of 3-D shapes". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239 –256, feb 1992. ISSN 0162-8828.

9. Eberly, David. "Distance Between Point and Triangle in 3D". March 2008.

10. Greenspan, M. and M. Yurick. "Approximate k-d tree search for efficient ICP". *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, 442 – 448. oct. 2003.

11. Hinchman, Jacob and Daniel Schreiter. "Automated Aerial Refueling". *Presentation to Aerial Refueling Systems Advisory Group (ARSAG)*, April 2007.

12. Huang, Lili and M. Barth. "A novel multi-planar LIDAR and computer vision calibration procedure using 2D patterns for automated navigation". *Intelligent Vehicles Symposium, 2009 IEEE*, 117 –122. june 2009. ISSN 1931-0587.

13. Ibeo Automotive Systems GmbH, Merkurring 60-62 D - 22143 Hamburg. *Operating Manual ibeo LUX 2010 Laserscanner*, 2010 edition edition, 2010.

14. Joung, Ji Hoon, Kwang Ho An, Jung Won Kang, Myung Jin Chung, and Wonpil Yu. "3D environment reconstruction using modified color ICP algorithm by fusion

of a camera and a 3D laser range finder". *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 3082 –3088. oct. 2009.

15. Khanafseh, Samer A. and Boris Pervan. "Automous Airborne Refuelng of Unmanned Air Vehicles Using the Global Positioning System". *Journal of Aircraft*, 44(5):1670–1682, September-October 2007.

16. Khoder, W., B. Fassinut-Mombot, and M. Benjelloun. "Quaternion Unscented Kalman Filtering for integrated Inertial Navigation and GPS". *Information Fusion, 2008 11th International Conference on*, 1 –8. 30 2008-july 3 2008.

17. Lewis, William K. *UCAV - The Next Generation Air-Speriority Fighter*. Master's thesis, School of Advanced Airpower Studies, Air Universith Maxwell AFB Alabama, June 2002. URL `http://www.au.af.mil/au/awc/awcgate/saas/lewis.pdf`.

18. Orton, Megan. "Air Force Remains Committed to Unmanned Aircraft Systems". *The Official Web Site of the U.S. Air Force*, January 2009. URL `http://www.af.mil/news/story.asp?id=123131324`.

19. Pandey, G., J. McBride, S. Savarese, and R. Eustice. "Extrinsic Calibration of a 3D Laser Scanner and an Omnidirectional Camera". Ford Motor Company Research and Innovation Center, Dearborn, MI 48124 USA, 2010.

20. Peinecke, N., T. Lueken, and B.R. Korn. "Lidar simulation using graphics hardware acceleration". *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*, 4.D.4–1 –4.D.4–8. oct. 2008.

21. Powell, G., R. Martin, D. Marshall, and K. Markham. "Simulation of FLIR and LADAR data using graphics animation software". *Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on*, 126 –134. 2000.

22. Ruel, S., T. Luu, M. Anctil, and S. Gagnon. "Target Localization from 3D data for On-Orbit Autonomous Rendezvous x00026; Docking". *Aerospace Conference, 2008 IEEE*, 1 –11. march 2008. ISSN 1095-323X.

23. Schneider, S., M. Himmelsbach, T. Luettel, and H.-J. Wuensche. "Fusing vision and LIDAR - Synchronization, correction and occlusion reasoning". *Intelligent Vehicles Symposium (IV), 2010 IEEE*, 388 –393. june 2010. ISSN 1931-0587.

24. Slabaugh, Greg, Ron Schafer, and Mark Livingston. "Optimal Ray Intersection For Computing 3D Points From N-View Correspondences". October 2001.

25. Soloviev, A. and M.U. de Haag. "Three-Dimensional Navigation with Scanning Ladars: Concept amp; Initial Verification". *Aerospace and Electronic Systems, IEEE Transactions on*, 46(1):14 –31, jan. 2010. ISSN 0018-9251.

26. Staff, Defense Systems. "Navy to outfit an X-47B prototype with refueling gear". *Defense Systems*, November 2011. URL

http://defensesystems.com/articles/2011/11/07/agg-navy-x47b-refueling-capability.aspx?admgarea=DS.

27. Titterton, D. and J. Weston. *Strapdown Inertial Navigation Technology*. IET RADAR, SONAR, NAVIGATION AND AVIONICS. The Institution of Electrical Engineers, Michael Faraday House Six Hills way, Stevenage Herts, SG1 2AY, United Kingdom, 2nd edition edition, 2004.

28. Toth, C., D.A. Grejner-Brzezinska, and Young-Jin Lee. "Terrain-based navigation: Trajectory recovery from LiDAR data". *Position, Location and Navigation Symposium, 2008 IEEE/ION*, 760 –765. may 2008.

29. Veth, Michael J. *Fusion of Imaging and Inertial Sensors for Navigation*. Ph.D. thesis, Air Force Institute of Technology, 2950 Hobson Way WPAFB OH, 45433, September 2006.

30. Warwick, Graham. "AFRL Advances Autonomous Aerial Refueling". *Aviation Week*, June 2008.

31. Weaver, Adam D. *Using Predictive Rendering as a Vision-Aided Technique for Autonomous Aerial Refueling*. Master's thesis, Air Force Institute of Technology, 2950 Hobson Way WPAFB OH, 45433), month=March, year=2009,.

32. Willis, A.R., M.J. Zapata, and J.M. Conrad. "A linear method for calibrating LIDAR-and-camera systems". *Modeling, Analysis Simulation of Computer and Telecommunication Systems, 2009. MASCOTS '09. IEEE International Symposium on*, 1 –3. sept. 2009. ISSN 1526-7539.

33. Zinsser, T., J. Schmidt, and H. Niemann. "A refined ICP algorithm for robust 3-D correspondence estimation". *Image Processing, 2003. ICIP 2003. Proceedings. 2003 International Conference on*, volume 2, II – 695–8 vol.3. sept. 2003. ISSN 1522-4880.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704–0188*



| 1. REPORT DATE *(DD–MM–YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From — To)* |
|---|---|---|
| 22–03–2012 | Master's Thesis | Sept 2010 — Mar 2012 |

**4. TITLE AND SUBTITLE**

Automated Aerial Refueling Position Estimation
Using a Scanning LiDAR

**5a. CONTRACT NUMBER**

DACA99–99–C–9999

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Curro II, Joseph A., 2d Lt USAF

**5d. PROJECT NUMBER**

11G235

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFIT/GE/ENG/12-11

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Air Force Research Laboratory (Jacob Hinchman)
2130 8th Street
Wright-Patterson AFB, OH 45433
(937) 785-8291; Jacob.Hinchman@wpafb.af.mil

**10. SPONSOR/MONITOR'S ACRONYM(S)**

AFRL

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION / AVAILABILITY STATEMENT**

Approval for public release; distribution is unlimited.

**13. SUPPLEMENTARY NOTES**



**14. ABSTRACT**

This research examines the application of using a scanning Light Detection and Ranging(LiDAR) to perform Automated Aerial Refueling(AAR). Specifically this thesis presents two algorithms to determine the relative position between the tanker and receiver aircraft. These two algorithms require a model of the tanker aircraft and the relative attitude between the aircraft. The first algorithm fits the measurements to the model of the aircraft using a modified Iterative Closest Point (ICP) algorithm. The second algorithm uses the model to predict LiDAR scans and compare them to actual measurements while perturbing the estimated location of the tanker. Each algorithm was tested with simulated LiDAR data before real data became available from test flights. The data collected from this test flight was used to determine the accuracy of the two algorithms with real LiDAR data. After correcting for modeling errors the accuracy of each algorithm is about a Mean Radial Spherical Error of 40cm.

**15. SUBJECT TERMS**

LiDAR, Iterative Closest Point, Automated Aerial Refueling, Relative Position Estimation

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Dr. John Raquet |
| U | U | U | UU | 114 | 19b. TELEPHONE NUMBER *(include area code)* (937) 785–3636, ext 4580; John.Raquet@afit.edu |