Technical Document 3252, Rev. 1
January 2012

Algorithm for Triangulating Visual Landmarks and Determining Their Error Covariance

Justin Gorgen
Lee Lemay

SSC Pacific
San Diego, CA 92152-5001

Technical Document 3252, Rev. 1
January 2012

Algorithm for Triangulating Visual Landmarks and Determining Their Error Covariance

Justin Gorgen
Lee Lemay

SSC Pacific
San Diego, CA 92152-5001

SSC Pacific
San Diego, California 92152-5001

J. J. Beel, CAPT, USN
Commanding Officer

C. A. Keeney
Executive Director

ADMINISTRATIVE INFORMATION

This report was prepared for the Office of Naval Research by the Navigation Systems R&D (Code 5234) of the SoS & Platform Integration Division (Code 52300), SSC Pacific.

Released by
K.S. Simonsen, Head
Navigation Systems R&D Branch

Under authority of
A. R. Dean, Head
SoS & Platform Integration Division

SB

CONTENTS

1. INTRODUCTION 1

2. TRIANGULATION IN A EUCLIDEAN FRAME 2

2.1 SOLVING FOR DEPTH 3
2.2 THE EFFECT OF LINEAR LEAST SQUARES ON THREE-DIMENSIONAL TRIANGULATION 4

3. ADAPTING TO A PINHOLE CAMERA MODEL AND NAVIGATION FILTER 5

3.1 PINHOLE CAMERA AND CAMERA CALIBRATION MODEL 6
3.2 FROM THE CALIBRATED IMAGE TO THE BODY FRAME 8
3.3 EULER ANGLES AND DIRECTION COSINE MATRIX 9
3.4 CLOSED-FORM SOLUTION FOR THE DIRECTION RAYS, CAMERA POSITION VECTORS, AND LANDMARK DEPTH 9

4. ESTIMATING THE QUALITY OF THE SOLUTION 10

4.1 DERIVING THE POSITION JACOBIAN 11
4.2 DEFINING THE INPUT COVARIANCE MATRIX 15

5. ESTIMATING PROJECTION ERROR 16

6. EVALUATION OF PERFORMANCE 19

6.1 TRIANGULATION SIMULATION SET-UP 19
6.2 TRIANGULATION SIMULATION RESULTS 21
6.3 PROJECTION COVARIANCE ESTIMATION 25

7. CONCLUSION 26

Figures

1. The variables needed for the triangulation of an unknown point X 2
2. The landmark, X, exists at the intersection of rays extended from each observation point (T1 and T2) towards the landmark 2
3. The linear least squares algorithm finds the λ1, λ2 such that the vectors λ1d1, λ2d2, and T form a triangle 3
4. The linear least squares algorithm will find a λ1 and λ2 that minimize the magnitude of the expression |λ1 d1 – λ2 d2 – T| 4
5. The 3-D reconstructions from back-projecting each view may be slightly different. The final estimate of the point will be the average of the two back-projections 4
6. The transformations needed to reach the navigation frame from the camera frame. 5
7. The pinhole camera model projects 3-D points on to a 2-D plane (figure from Equation [2]) 6
8. The vector x from Equation (11) points toward the feature point in the camera frame of reference 8
9. The camera frame is translated and rotated away from the body frame 8
10. The shape and contents of the input covariance matrix. Regions 1 and 3 contain the NED, phi, theta, psi covariances for the first and second images, respectively. Regions 2 and 4 contain the U,V covariances for images 1 and 2 15
11. The Monte Carlo simulation set-up 19
12. The spread of the triangulation solution values in the north/east plane. The x-axis in each sub-chart represents East coordinates, in meters; the y-axis represents north coordinates, in meters. The measured covariance, represented by dashed black ellipses, is the covariance calculated from the samples. The expected covariance, represented by solid red ellipses, is the distribution around the true point location calculated by the covariance estimation algorithm. For small values of Euler angle noise variance (1 degree or less), the predicted distribution is accurate enough that the expected and measured covariance ellipses overlap in the figure 23
13. The distribution of errors versus the expected distribution of errors. The blue lines represent observed distributions while the black lines represent expected distributions. The observed and expected distributions match up well for all cases except for large attitude errors combined with small camera position errors (Figures g and j) 24
14. The results of the Monte Carlo simulation of UV projection error 25
15. The distribution of Mahalanobis distances for the Monte Carlo simulation of UV projection errors 26

Tables

1. Variables used in deriving triangulation and covariance equations 6
2. Mean values used for Monte Carlo simulation 20
3. Test Conditions 21
4. Comparison of random noise from different grades of gyroscopes 22

1. INTRODUCTION

The work described in this report has to do with the problem of vision or camera-based navigation. In these problems, it is often necessary to map the location of some feature or object, which is subsequently used when calculating the position of a camera-based navigation system user. Mapping the location of this feature relative to some known location is easy; one simply needs to know a direction and distance from the known point to the feature point. However, it is not always practical or possible to measure the distance to an object (e.g., in robotics, finding the location of a visual landmark that a robot can see, but not reach with a laser rangefinder or mechanical arm). All is not lost, however. It is possible to map the location of the visual landmark if the robot can measure the angle towards it from two other known positions. The distance to the landmark can be deduced from the angles.

Triangulation, which this technical report discusses, is the act of mapping a feature using information about the direction towards a landmark from two or more known locations. This report focuses on a method of solving for the location of a visual landmark—a feature point appearing in two or more images—using a linear least squares algorithm. Although it does not perform as optimally in terms minimizing position error as non-linear, iterative triangulation algorithms[1], the linear least squares approach is fast, fixed-cost, and suitable for machine computation. Furthermore, because the least squares algorithm has a closed-form solution, it simple to propagate uncertainties in camera position and orientation to uncertainty in the three-dimensional (3-D) location of the feature point.

The approach described in this report was developed to triangulate visual landmarks identified in images from two locations taken with a calibrated camera. This report assumes the locations of the camera are extracted from a navigation system, which gives the camera position in the north-east-down (NED) local coordinate frame and camera orientation in the so-called "National Aeronautics and Space Administration (NASA) Aerospace 3-2-1 Euler angles"[2]. The result is a closed-form equation that maps 16 variables (roll, pitch, yaw, NED position coordinates for each camera location; and pixel coordinates for each image) to three variables, the x, y, and z coordinates of the feature point.

This report also presents a method for estimating the error in the triangulation solution as well as an evaluation of the performance of the error estimation. To estimate the error of the triangulation algorithm, the triangulation equation is linearized about the given inputs using a first-order Taylor expansion. The triangulation equation must be linearized because the values used in the linear-least squares problem are non-linear functions of the camera orientation. The linearization allows the error covariance of the 16 input variables to be mapped to the error covariance of the 3-D point location. To examine the performance of the error estimation algorithm, we run Monte Carlo simulations with noisy camera positions and Euler angles and compare the distribution of the outputs with the expected distribution. The result is an estimate of the visual landmark's location in three dimensions with an accurate description of the uncertainty in the position. With the triangulation and error-estimation algorithms, it is possible to accurately locate visual landmarks and describe the uncertainty of the landmark's position coordinates.

2. TRIANGULATION IN A EUCLIDEAN FRAME

We will first demonstrate a method for solving the triangulation problem with vector algebra in a general Euclidean reference frame, and then show how these vectors can be extracted from camera and navigation-filter output. The triangulation algorithm first uses position and direction information to determine the distance to the feature point from each known location, and then assigns a location to the feature point by adding these distances to each of the observation locations.

Phrasing the triangulation problem in terms of vectors is straightforward. To locate an unknown point from two known observation points, four pieces of information are needed: the location of each observation point, and a direction from each observation point to the unknown point.

This information can be described with four column vectors: $\mathbf{T}_1$, $\mathbf{T}_2$, $\in R^3$, which describe the location of each observation point in the navigation frame, and $\mathbf{d_1}$, $\mathbf{d_2} \in R^3$, which are vectors that point from the observation point to the landmark. These quantities are illustrated in Figure 1.

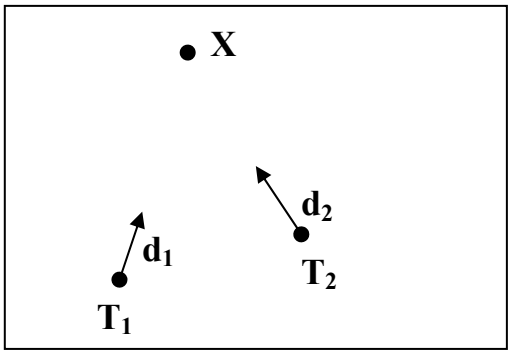

Figure 1. The variables needed for the triangulation of an unknown point X.

The unknown point, $\mathbf{X}$, lies at a distance λ_1 along the ray defined by extending the vector $\mathbf{d_1}$ from the point $\mathbf{T_1}$. The same is true for some distance λ_2 along the ray defined by $\mathbf{d_2}$ and $\mathbf{T_2}$. The unknown point is located at intersection of these two line-of-sight vectors, as shown in Figure 2.

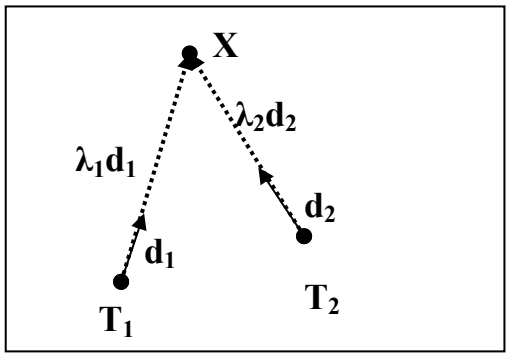

Figure 2. The landmark, X, exists at the intersection of rays extended from each observation point (T_1 and T_2) towards the landmark.

This geometric construction leads to the equality:

$$\mathbf{X} = \lambda_1 \mathbf{d}_1 + \mathbf{T}_1 = \lambda_2 \mathbf{d}_2 + \mathbf{T}_2 . \tag{1}$$

Here, the terms $\lambda_1\mathbf{d_1}+\mathbf{T_1}$ and $\lambda_2\mathbf{d_2}+\mathbf{T_2}$ are the line-of-sight vectors, representing a ray drawn from each observation location to the feature point. The unknown point, $\mathbf{X}$, can be found by first solving for the depth scalars λ_1, λ_2, and then plugging in to Equation (1) to solve for $\mathbf{X}$.

2.1 SOLVING FOR DEPTH

Rearranging Equation (1) gives an equation that can easily solve for depth:

$$\lambda_1\mathbf{d}_1 - \lambda_2\mathbf{d}_2 = \mathbf{T}_2 - \mathbf{T}_1 . \tag{2}$$

For convenience, $\mathbf{T} \equiv \mathbf{T_2} - \mathbf{T_1}$. Using this definition, Equation (2) above can be written as the following vector equation:

$$\begin{bmatrix}\mathbf{d}_1 & -\mathbf{d}_2\end{bmatrix}\begin{bmatrix}\lambda_1 \\ \lambda_2\end{bmatrix} = \mathbf{T} . \tag{3}$$

Solving this using linear least squares will yield the depths λ_1, λ_2 that minimize the distance between $\lambda_1\mathbf{d_1}$-$\lambda_2\mathbf{d_2}$, and $\mathbf{T}$. In other words, this attempts to form a triangle with sides $\lambda_1\mathbf{d_1}$, $\lambda_2\mathbf{d_2}$, and $\mathbf{T}$, as shown in Figure 3.

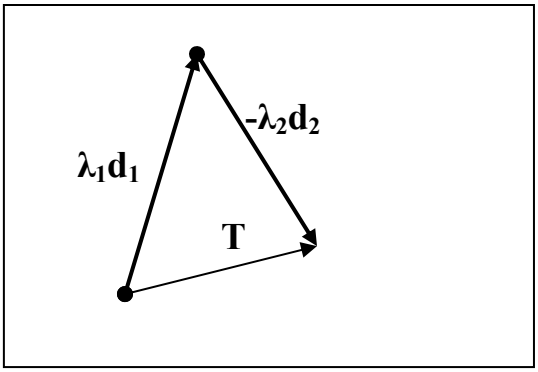

Figure 3. The linear least squares algorithm finds the $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$ such that the vectors $\boldsymbol{\lambda}_1\mathbf{d}_1$, $\boldsymbol{\lambda}_2\mathbf{d}_2$, and T form a triangle.

Solving Equation (3) using linear least squares leads to the expression:

$$\begin{bmatrix}\lambda_1 \\ \lambda_2\end{bmatrix} = \frac{\begin{bmatrix}\|\mathbf{d}_2\|^2 & \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1 \\ \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1 & \|\mathbf{d}_1\|^2\end{bmatrix}\begin{bmatrix}\mathbf{d}_1^{\mathrm{T}}\mathbf{T} \\ -\mathbf{d}_2^{\mathrm{T}}\mathbf{T}\end{bmatrix}}{\|\mathbf{d}_1\|^2\|\mathbf{d}_2\|^2 - \|\mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1\|^2} . \tag{4}$$

With this closed-form solution for the depths, the location of the unknown point $\mathbf{X}$ can be easily found by back projecting each depth from each observation point. Rearranging Equation (1) to illustrate the effect of both depths on the solution leads to

$$\mathbf{X} = \frac{\lambda_1\mathbf{d}_1 + \mathbf{T}_1}{2} + \frac{\lambda_2\mathbf{d}_2 + \mathbf{T}_2}{2} . \tag{5}$$

In two-dimensions, this solution is identical to solving for **X** using only one observation direction and one depth. The next section discusses the purpose and geometric meaning of the average of the two solutions.

2.2 THE EFFECT OF LINEAR LEAST SQUARES ON THREE-DIMENSIONAL TRIANGULATION

In two dimensions, the rays from each observation always intersect if they are not parallel. In three dimensions, due to noise, the rays usually not intersect. However, as long as the two rays are not parallel, the least squares algorithm described above will give a solution that can be used as an estimate of the feature point's location. In what follows, we describe the geometric meaning of this estimate.

The solution generated by the linear least squares algorithm is the point where the lines almost intersect. To be more precise, the position estimate **X** from Equation (5) gives the mid-point of the shortest line-segment that connects the two line-of-sight vectors, $\lambda_1\mathbf{d_1}+\mathbf{T_1}$ and $\lambda_2\mathbf{d_2}+\mathbf{T_2}$. This is best illustrated by demonstrating which norm is minimized by the least squares estimate for depth in Equation (4).

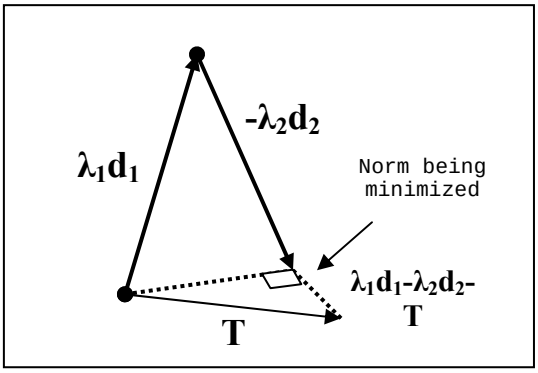

Figure 4. The linear least squares algorithm will find a $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ that minimize the magnitude of the expression $|\boldsymbol{\lambda}_1 d_1 - \boldsymbol{\lambda}_2 d_2 - T|$.

As seen in Figure 4, the least squares method calculates the depths that minimize the error between the vector difference, $\lambda_1\,\mathbf{d_1} - \lambda_2\,\mathbf{d_2}$, and the estimate of camera translation, **T**. This is equivalent to calculating the projection of **T** onto the plane defined by $\mathbf{d_1}$ and $\mathbf{d_2}$. When these depths are independently used in reconstruction from each view, the two reconstructions are inconsistent, as shown in Figure 5.

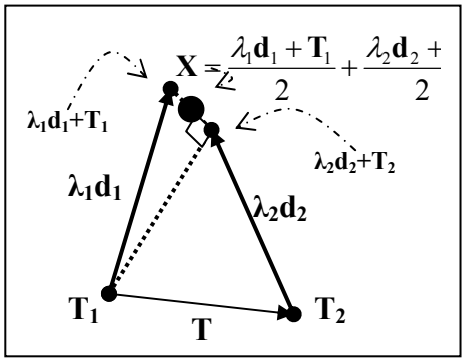

Figure 5. The 3-D reconstructions from back-projecting each view may be slightly different. The final estimate of the point will be the average of the two back-projections.

Averaging the results of the two back-projections produces a solution for the unknown position. We will use this solution as our final estimate of the feature point's position. This algorithm can now be used in general to triangulate a 3-D location. Adapting this algorithm for use with cameras and navigation filters will require models for camera behavior and extracting Euclidean camera location and orientation from navigation filter outputs.

3. ADAPTING TO A PINHOLE CAMERA MODEL AND NAVIGATION FILTER

As we have seen, the only information needed to triangulate the location of an unknown point is the line-of-sight vectors toward the feature point from two known locations. This information can be extracted from the output of a navigation filter. The location of each observation point is the position output of the navigation filter at the time an image was taken, corrected for lever-arm effects if the camera is not co-located with the navigation system. Determining the line-of-sight vector in the navigation frame is more complicated because it requires knowledge of the attitude of the observer, lever-arm effects, and camera calibration. Three transformations are required for mapping a feature's image coordinates to a 3-D line-of-sight vector in the navigation frame:

1. A transformation to convert a feature's *u,v* pixel coordinates to a 3-D line-of-sight vector in the camera's frame of reference.
2. A transformation to map this 3-D line-of-sight vector from the camera's frame of reference to the navigating body's frame of reference.
3. A transformation to map the 3-D line-of-sight vector from the body frame to the navigation frame. After this transformation, the line-of-sight vector points from the observation point towards the feature point's location in the navigation frame.

This process is summarized in Figure 6.

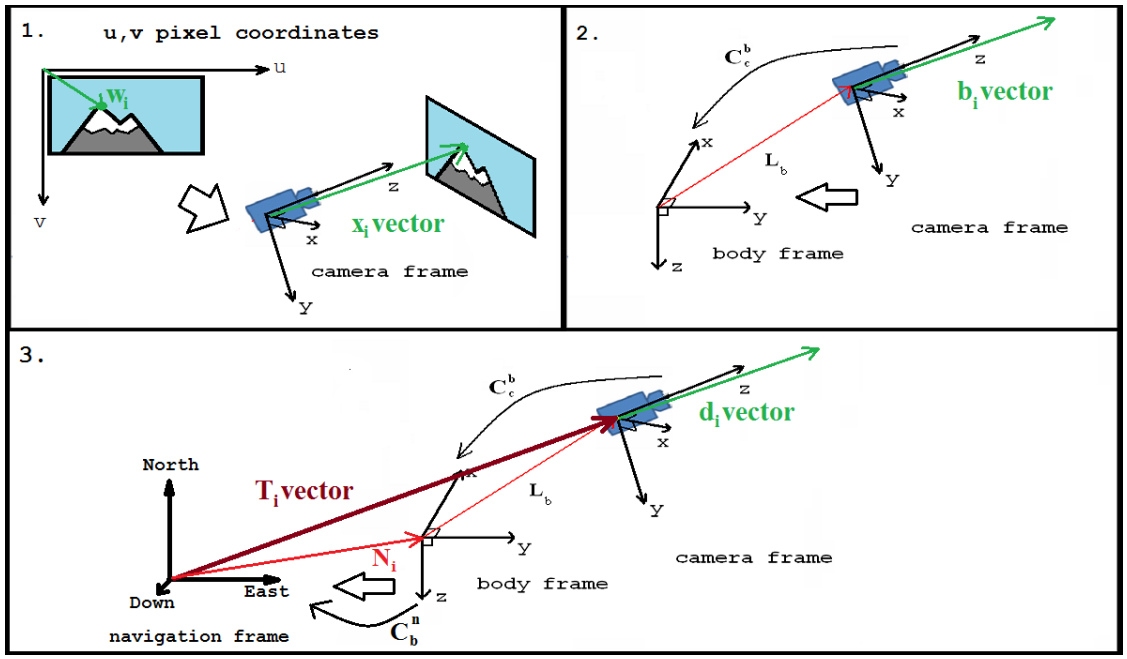

Figure 6. The transformations needed to reach the navigation frame from the camera frame.

While describing the transformation from pixel coordinates to vectors in the navigation coordinates, several placeholder variables and measured matrices will be used. For reference, these variables are summarized in Table 1.

Table 1. Variables used in deriving triangulation and covariance equations.

w	Vector representing the u,v pixel coordinates of a feature point
x	Vector representing the line-of-sight vector in the camera frame.
b	Vector representing the line-of-sight vector in the body frame.
d	Vector representing the line-of-sight vector in the navigation frame
$\mathbf{C}_c^b$	Rotation matrix used to rotate the vector **x** from the camera frame to the body frame
$\mathbf{C}_b^n$	Rotation matrix used to rotate the vector **b** from the body frame to the navigation frame
N	Vector representing the position output by the navigation filter
L	Vector representing the lever arm from the user to the camera center, in the body's frame of reference
T	Vector representing the location of the camera center in the navigation frame

3.1 PINHOLE CAMERA AND CAMERA CALIBRATION MODEL

The pinhole camera model is the classic model used to map a feature's 3-D coordinates to 2-D pixel coordinates[3]. Its inverse can therefore be used to convert a feature's pixel coordinates to a vector that is consistent with the Euclidian coordinates of the navigation frame. As shown in Figure 7, the pinhole camera model maps 3-D points $[x, y, z]^T$ in the camera's frame of reference to 2-D coordinates on an image.

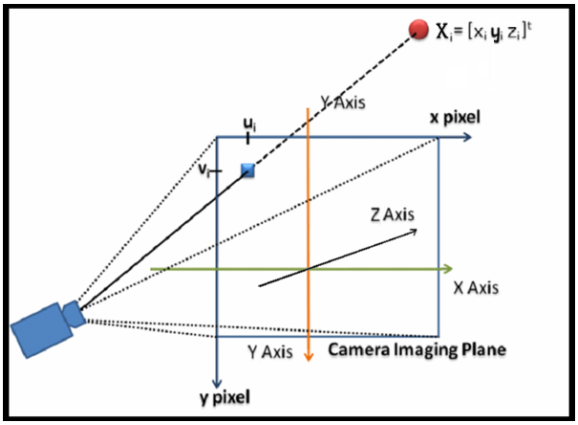

Figure 7. The pinhole camera model projects 3-D points onto a 2-D plane (figure from Equation [2]).

To use pixel coordinates for triangulation, the calibration matrix of the camera that took the image must be known. The camera calibration matrix is used when modeling the projection of a scene onto an image sensor. This projection can be modeled by:

$$\mathbf{K} = \begin{bmatrix} f_x & skew & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} \lambda u \\ \lambda v \\ \lambda \end{bmatrix} = \mathbf{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{\lambda} \mathbf{K} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{8}$$

Here, u,v are the pixel coordinates of a feature point, **K** is the camera calibration matrix, $[x,y,z]^T$ are the coordinates of the feature point in the camera's frame of reference, and λ is the depth scalar in the same units as the navigation frame.

From Equation (8), we can recover a vector **x,** which points in the direction of the feature point in the camera's frame of reference. This vector can also be described as the calibrated image of the feature point.

$$\mathbf{w} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{9}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \lambda \, \mathbf{K}^{-1} \mathbf{w}, \tag{10}$$

$$\mathbf{x} = \mathbf{K}^{-1} \mathbf{w}. \tag{11}$$

The vector **x** is a vector in the camera's frame that points towards a feature point, meeting it at a depth λ. This is shown in Figure 8.

Because of the camera calibration matrix, this λ is consistent with the depth that is solved for in the linear least squares problem above. Thus, an image coordinate can be mapped from the $[u\ v\ 1]^T$ coordinate system to a vector **x** in the camera frame that is now related to the navigation frame by a simple rigid body transformation. This **x** can be then transformed to the navigation frame to find the vector **d** that is needed for triangulation.

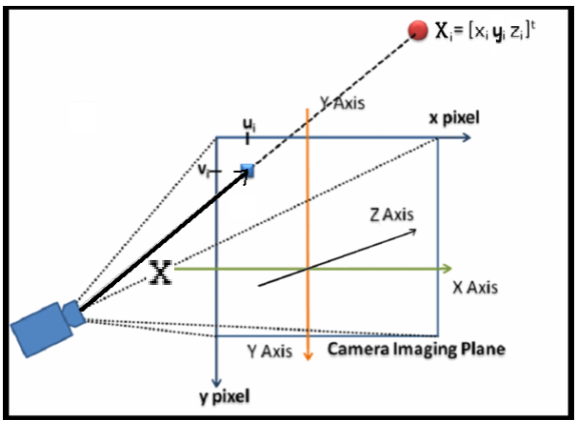

Figure 8. The vector x from Equation (11) points toward the feature point in the camera frame of reference.

3.2 FROM THE CALIBRATED IMAGE TO THE BODY FRAME

The calibrated image vector **x** now needs to be aligned to the navigation frame in order to triangulate the feature point. The navigation filter's estimate of attitude can be used to align the camera frame with the navigation frame. However, the attitude reported by the filter is the orientation of some rigid body frame with respect the navigation frame, and this frame is not usually the same as the camera frame. Therefore, the calibrated image vector from the previous step has to be transformed to the user's body frame before being transformed to the navigation frame.

The transformation from camera frame to body frame is a simple rigid body transformation, i.e., it consists of a translation and rotation. Here we will call the translation vector **L**, and the rotation matrix $\mathbf{C}_c^b$, and their effect is shown in Figure 9.

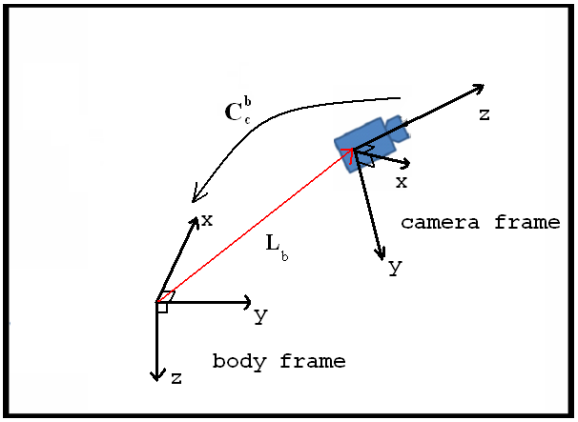

Figure 9. The camera frame is translated and rotated away from the body frame.

The vector **L** represents the lever arm between the camera center and the point whose location is output by the navigation filter. The vector **L** is defined in the body frame and thus its effect on the camera's location in the navigation frame can easily be calculated. The lever is used to extract the camera's location from the location of the navigation body. For the purposes of this paper, it is assumed that the $\mathbf{C}_c^b$ matrix and lever arm **L** are error-free and known. With this information, the vector **x** from the previous step can be transformed to a vector in the user's body frame.

Converting from camera frame to body frame can be accomplished using the following formula:

$$\mathbf{b}=\mathbf{C}_c^b\mathbf{x}\,. \tag{12}$$

The calibrated image vector is now expressed in the body frame. The lever arm **L** will be used later to define the observation position.

3.3 EULER ANGLES AND DIRECTION COSINE MATRIX

The last step before triangulation can be completed is transforming the vector **b** from the body frame to the navigation frame to get the line-of-sight vector $\lambda_i\mathbf{d_i}+\mathbf{T_i}$. This is accomplished using information from the output of the navigation filter. This report assumes that the navigation filter outputs orientation in Euler angles in the NASA 3-2-1 aerospace sequence yaw (ψ), pitch (θ), roll (Φ)[4].

The transformation from body frame to navigation frame is accomplished using a 3 x 3 rotation matrix, here called $\mathbf{C}_b^n$. This rotation matrix is a function of the Euler angles as shown below, where *c* and *s* are used as shorthand for *cosine* and *sine*:

$$\mathbf{C}_b^n=\begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix}. \tag{13}$$

With a simple matrix multiplication, vectors in the body frame are transformed to the navigation frame.

$$\mathbf{d}=\mathbf{C}_b^n\mathbf{b}\,. \tag{14}$$

The $\mathbf{C}_b^n$ matrix is also used to solve for the position of the camera at the time the image was taken. The navigation filter outputs a position solution as a 3-vector, and the position of the camera is simply the output of the navigation filter, $\mathbf{N_i}$, plus the rotated body-frame components of the lever arm **L**. Therefore, we can write:

$$\mathbf{T_i}=\mathbf{N_i}+\mathbf{C}_{b_i}^n\mathbf{L}\,. \tag{15}$$

We now have all the information needed to compute the depth of the feature point, and thus, the 3-D location of the feature point.

3.4 CLOSED-FORM SOLUTION FOR THE DIRECTION RAYS, CAMERA POSITION VECTORS, AND LANDMARK DEPTH

Combining all of the above steps into a single equation for the direction vectors and location vectors, we get

$$\mathbf{d_i}=\mathbf{C}_{b_i}^n\mathbf{C}_c^b\mathbf{K}^{-1}\mathbf{w}\,. \tag{16}$$

The values from Equations (11), (15), and (16) can be substituted into Equation (4) to calculate an estimate for the depths λ_1, λ_2. One important observation is that the magnitude of the direction vectors $\mathbf{d_1}$ and $\mathbf{d_2}$ only depend on $\mathbf{K}$ and the image coordinates $\mathbf{w_1}$ and $\mathbf{w_2}$, i.e., the magnitude of $\mathbf{d_i}$ equals the magnitude of $\mathbf{x_i}$:

$$\|\mathbf{d_i}\| = \|\mathbf{x_i}\|. \tag{17}$$

Thus, Equation (4) can be simplified into the following, which will simplify deriving the equations for estimating the error of this triangulation algorithm:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \frac{\begin{bmatrix} \|\mathbf{x}_2\|^2 & \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1 \\ \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1 & \|\mathbf{x}_1\|^2 \end{bmatrix} \begin{bmatrix} \mathbf{d}_1^{\mathrm{T}}\mathbf{T} \\ -\mathbf{d}_2^{\mathrm{T}}\mathbf{T} \end{bmatrix}}{\|\mathbf{x}_1\|^2 \|\mathbf{x}_2\|^2 - \|\mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1\|^2}. \tag{18}$$

The partial derivatives of this equation contains fewer terms because the $\mathbf{x_i}$ terms do not depend on $\mathbf{C}_c^b$ or $\mathbf{C}_b^n$. It is therefore easier to take the Jacobian of this expression to estimate the error covariance of the solution.

4. ESTIMATING THE QUALITY OF THE SOLUTION

One way to characterize the quality of the triangulation solution is to calculate a covariance of the estimation errors as a function of pixel noise, camera position error, and camera attitude error. Since the relationship between $\mathbf{X}$ error, pixel error, and camera attitude error is non-linear, this will require a linearization of the triangulation equation. If the estimate of the landmark positions, $\mathbf{X}$, generated by the triangulation method outlined above is to be used for navigation, tracking, or surveying, we will need to know if our estimate of $\mathbf{X}$ is accurate and be able to quantify that accuracy. The classic way of estimating the error covariance for a variable described by a non-linear function like the triangulation algorithm is through a first-order Taylor series expansion.

The first-order Taylor series expansion treats each x, y, and z component of the position estimate $\mathbf{X}$ as a continuous, differentiable function of several input variables. The input variables will depend on the individual application. A first-order Taylor series expansion allows for an estimate of how errors in each of those input variables contribute to errors in the x, y, and z positions of the position estimate. As mentioned in Section 3.3, it is assumed the camera calibration matrix $\mathbf{K}$, the camera to body transform $\mathbf{C}_c^b$, and the lever-arm $\mathbf{L}$ are known and error-free. Thus, this section assumes that there will be errors in the camera positions, camera attitudes, and the pixel coordinates of the landmark in each image. The camera position and attitude will come from a navigation filter that outputs a position solution in NED coordinates and orientation in 3-2-1 NASA-aerospace roll-pitch-yaw Euler angles. Therefore, each observation of the landmark will have eight sources of error (errors in north, east, down, roll, pitch, yaw, u, and v coordinates). Thus, the landmark position $\mathbf{X}$ is estimated from two observations that each has eight sources of error. This results in $\mathbf{X}$ being described as a function of the following 16 variables:

$$\mathbf{X} = f(N_1, E_1, D_1, \phi_1, \theta_1, \psi_1, u_1, v_1 N_2, E_2, D_2, \phi_2, \theta_2, \psi_2, u_2, v_2) = f(z).$$

Taking the partial derivatives of these functions forms a Jacobian matrix, and multiplying this Jacobian with the prior estimates of error in each of the 16 variables produces a reliable estimate of the error of the triangulation solution.

The Jacobian of **X**, which will be referred to as $\mathbf{J}$, is a 3x16 matrix with the form:

$$\mathbf{J} \equiv \left[\frac{\delta \mathbf{X}}{\delta N_1} \; \frac{\delta \mathbf{X}}{\delta E_1} \; \frac{\delta \mathbf{X}}{\delta D_1} \; \frac{\delta \mathbf{X}}{\delta \phi_1} \; \frac{\delta \mathbf{X}}{\delta \theta_1} \; \frac{\delta \mathbf{X}}{\delta \psi_1} \; \frac{\delta \mathbf{X}}{\delta u_1} \; \frac{\delta \mathbf{X}}{\delta v_1} \; \frac{\delta \mathbf{X}}{\delta N_2} \; \frac{\delta \mathbf{X}}{\delta E_2} \; \frac{\delta \mathbf{X}}{\delta D_2} \; \frac{\delta \mathbf{X}}{\delta \phi_2} \; \frac{\delta \mathbf{X}}{\delta \theta_2} \; \frac{\delta \mathbf{X}}{\delta \psi_2} \; \frac{\delta \mathbf{X}}{\delta u_2} \; \frac{\delta \mathbf{X}}{\delta v_2} \right]. \tag{19}$$

$$\because \delta \mathbf{X} = \mathbf{J} \delta z$$

The covariance estimation will be performed by the standard covariance transformation equation[5]:

$$\begin{aligned} \mathbf{P_X} &= \mathrm{E}\left\{ \left(\mathbf{X} - \overline{\mathbf{X}}\right)\left(\mathbf{X} - \overline{\mathbf{X}}\right)^{\mathbf{T}} \right\} = \mathrm{E}\left\{ \delta \mathbf{X} \delta \mathbf{X}^{\mathbf{T}} \right\} \\ &= \mathrm{E}\left\{ \mathbf{J} \delta z \delta z^{\mathbf{T}} \mathbf{J}^{\mathbf{T}} \right\} \\ &= \mathbf{J} \; \mathrm{E}\left\{ \delta z \delta z^{\mathbf{T}} \right\} \mathbf{J}^{\mathbf{T}} \\ &= \mathbf{J} \; \mathbf{\Omega} \; \mathbf{J}^{\mathbf{T}}, \end{aligned} \tag{20}$$

where $\mathrm{E}\{\}$ is the expectation operator, $\mathbf{P_X}$ is the covariance of the point's location, and $\mathbf{\Omega}$ is the 16x16 covariance matrix of the 16 input variables that represent the camera's positions and the visual landmark's pixel coordinates.

4.1 DERIVING THE POSITION JACOBIAN

Calculating $\mathbf{J}$ can be difficult, and to simplify the calculations we make the following observations. From the position solution Equation (5), we see that the position solution is mainly a function of six variables: λ, **d** and **T** for each of the two viewpoints. We will take advantage of this and use the chain rule to simplify the derivation of the Jacobian. Furthermore, symmetry in equation (5) also simplifies the derivation of the Jacobian. That is, the term $\lambda_{\mathbf{i}} \mathbf{d_i} + \mathbf{T_i}$ appears twice. This allows us to express the Jacobian as the sum of two matrices:

$$\mathbf{J} = \frac{\mathbf{J}_1 + \mathbf{J}_2}{2}. \tag{21}$$

Here, $\mathbf{J}_1$ and $\mathbf{J}_2$ represent the Jacobians of the terms $\mathbf{r_1} = \lambda_1 \mathbf{d}_1 + \mathbf{T}_1$ and $\mathbf{r}_2 = \lambda_2 \mathbf{d}_2 + \mathbf{T}_2$, respectively. Each Jacobian $\mathbf{J}_i$ contains the partial derivatives of $\mathbf{r_i}$ with respect to the 16 input variables:

$$\mathbf{J_i} = \left[\frac{\delta \mathbf{r_i}}{\delta N_1} \; \frac{\delta \mathbf{r_i}}{\delta E_1} \; \frac{\delta \mathbf{r_i}}{\delta D_1} \; \frac{\delta \mathbf{r_i}}{\delta \phi_1} \; \frac{\delta \mathbf{r_i}}{\delta \theta_1} \; \frac{\delta \mathbf{r_i}}{\delta \psi_1} \; \frac{\delta \mathbf{r_i}}{\delta u_1} \; \frac{\delta \mathbf{r_i}}{\delta v_1} \; \frac{\delta \mathbf{r_i}}{\delta N_2} \; \frac{\delta \mathbf{r_i}}{\delta E_2} \; \frac{\delta \mathbf{r_i}}{\delta D_2} \; \frac{\delta \mathbf{r_i}}{\delta \phi_2} \; \frac{\delta \mathbf{r_i}}{\delta \theta_2} \; \frac{\delta \mathbf{r_i}}{\delta \psi_2} \; \frac{\delta \mathbf{r_i}}{\delta u_2} \; \frac{\delta \mathbf{r_i}}{\delta v_2} \right].$$

The Jacobian of the expression $\lambda_i \mathbf{d_i} + \mathbf{T_i}$ is found by taking the partial derivatives with respect to the 16 input scalars in *z*. The partial derivative $\mathbf{r_i}$ with respect to any scalar variable *a* is:

$$\frac{\delta \mathbf{r_i}}{\delta \mathrm{a}} = \frac{\delta \lambda_i}{\delta \mathrm{a}} \mathbf{d_i} + \lambda_i \frac{\delta \mathbf{d_i}}{\delta \mathrm{a}} + \frac{\delta \mathbf{T_i}}{\delta \mathrm{a}} \tag{22}$$

Next, the partial derivatives of $\mathbf{d_i}$, $\mathbf{T_i}$,and λ_i are determined. Recall from Equation (16) that the vector $\mathbf{d_i}$ is function of several change of bases operations on an image's u,v coordinates. The partial derivatives of the depth vector $\mathbf{d_i}$ are therefore,

$$\begin{aligned} \frac{\delta \mathbf{d_i}}{\delta \mathrm{a}} = & \frac{\delta \mathbf{C}_{b_i}^{n}}{\delta \mathrm{a}} \mathbf{C}_c^b \mathbf{K}^{-1} \mathbf{w_i} \\ & + \mathbf{C}_{b_i}^{n} \left(\frac{\delta \mathbf{C}_c^b}{\delta \mathrm{a}} \mathbf{K}^{-1} \mathbf{w_i} \right. \\ & \left. + \mathbf{C}_c^b \left(\frac{\delta \mathbf{K}^{-1}}{\delta \mathrm{a}} \mathbf{w_i} + \mathbf{K}^{-1} \frac{\delta \mathbf{w_i}}{\delta \mathrm{a}} \right) \right). \end{aligned} \tag{23}$$

For this report, the camera-to-body transformation matrix $\mathbf{C}_b^c$ and the calibration matrix $\mathbf{K}$ are assumed to be known and invariant, so Equation (23) simplifies to

$$\begin{aligned} \frac{\delta \mathbf{d_i}}{\delta \mathrm{a}} = & \frac{\delta \mathbf{C}_{b_i}^{n}}{\delta \mathrm{a}} \mathbf{C}_c^b \mathbf{K}^{-1} \mathbf{w_i} \\ & + \mathbf{C}_{b_i}^{n} \mathbf{C}_c^b \mathbf{K}^{-1} \frac{\delta \mathbf{w_i}}{\delta \mathrm{a}}. \end{aligned} \tag{24}$$

With the partial derivatives of $\mathbf{d_i}$ defined, the next terms of Equation (22) to define are the partial derivatives of $\mathbf{T_i}$. From Equation (15), we can see that $\mathbf{T_i}$ depends on $\mathbf{C}_b^n$, $\mathbf{L}$, and $\mathbf{N_i}$. The partial derivatives of $\mathbf{T_i}$ are therefore,

$$\frac{\delta \mathbf{T_i}}{\delta \mathrm{a}} = \frac{\delta \mathbf{N_i}}{\delta \mathrm{a}} + \frac{\delta \mathbf{C}_{b_i}^{n}}{\delta \mathrm{a}} \mathbf{L} + \mathbf{C}_{b_i}^{n} \frac{\delta \mathbf{L}}{\delta \mathrm{a}}. \tag{25}$$

The lever-arm $\mathbf{L}$ is assumed known and invariant, so Equation (25) simplifies to

$$\frac{\delta \mathbf{T_i}}{\delta \mathrm{a}} = \frac{\delta \mathbf{N_i}}{\delta \mathrm{a}} + \frac{\delta \mathbf{C}_{b_i}^{n}}{\delta \mathrm{a}} \mathbf{L}. \tag{26}$$

The partial derivatives of the depths, λ_i, are calculated in a similar fashion to the partial derivatives for the terms of $\mathbf{d_i}$ and $\mathbf{T_i}$. However, Equation (18) is more complex than the one used for the position equation. To simplify the process of taking the partial derivatives of this equation, we will take advantage of the division rule of derivatives. That is, we will separate the expression in

Equation (18) into two parts, the numerator and denominator and combine the partial derivatives of these parts:

$$\frac{\delta\begin{bmatrix}\lambda_1\\ \lambda_2\end{bmatrix}}{\delta \mathrm{a}} = \frac{\delta\left(\frac{\begin{bmatrix}\|\mathbf{x}_2\|^2 & \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1\\ \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1 & \|\mathbf{x}_1\|^2\end{bmatrix}\begin{bmatrix}\mathbf{d}_1^{\mathrm{T}}\mathbf{T}\\ -\mathbf{d}_2^{\mathrm{T}}\mathbf{T}\end{bmatrix}}{\|\mathbf{x}_1\|^2\|\mathbf{x}_2\|^2 - \left\|\mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1\right\|^2}\right)}{\delta \mathrm{a}} = \frac{\mathrm{low}\dfrac{\delta\,\mathrm{high}}{\delta \mathrm{a}} - \mathrm{high}\dfrac{\delta\,\mathrm{low}}{\delta \mathrm{a}}}{(\mathrm{low})^2}. \tag{27}$$

$$\mathrm{high} = \begin{bmatrix}\|\mathbf{x}_2\|^2 & \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1\\ \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1 & \|\mathbf{x}_1\|^2\end{bmatrix}\begin{bmatrix}\mathbf{d}_1^{\mathrm{T}}\mathbf{T}\\ -\mathbf{d}_2^{\mathrm{T}}\mathbf{T}\end{bmatrix}$$
$$\mathrm{low} = \|\mathbf{x}_1\|^2\|\mathbf{x}_2\|^2 - \left\|\mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1\right\|^2. \tag{28}$$

We need the partial derivatives of the placeholder variables *high* and *low* to compute the partial derivatives of the depths. These partial derivatives are

$$\begin{aligned}\frac{\delta\,\mathrm{low}}{\delta \mathrm{a}} = {} & 2\bullet|\mathbf{x}_2|^2\bullet\left(\frac{\delta\mathbf{x}_1}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{x}_1\\ & +2\bullet|\mathbf{x}_1|^2\bullet\left(\frac{\delta\mathbf{x}_2}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{x}_2\\ & -2\bullet\left(\left(\frac{\delta\mathbf{d}_2}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{d}_1+\left(\frac{\delta\mathbf{d}_1}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{d}_2\right)\left(\mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1\right)\end{aligned} \tag{29}$$

$$\begin{aligned}\frac{\delta\,\mathrm{high}}{\delta \mathrm{a}} = {} & \begin{bmatrix} 2\bullet\left(\frac{\delta\mathbf{x}_2}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{x}_2 & \left(\left(\frac{\delta\mathbf{d}_2}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{d}_1+\left(\frac{\delta\mathbf{d}_1}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{d}_2\right)\\ \left(\left(\frac{\delta\mathbf{d}_2}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{d}_1+\left(\frac{\delta\mathbf{d}_1}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{d}_2\right) & 2\bullet\left(\frac{\delta\mathbf{x}_1}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{x}_1\end{bmatrix}\begin{bmatrix}\mathbf{d}_1^{\mathrm{T}}\mathbf{T}\\ -\mathbf{d}_2^{\mathrm{T}}\mathbf{T}\end{bmatrix}\\ & +\begin{bmatrix}\|\mathbf{x}_2\|^2 & \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1\\ \mathbf{d}_2^{\mathrm{T}}\mathbf{d}_1 & \|\mathbf{x}_1\|^2\end{bmatrix}\begin{bmatrix}\left(\frac{\delta\mathbf{d}_1}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{T}+\mathbf{d}_1^{\mathrm{T}}\frac{\delta\mathbf{T}}{\delta \mathrm{a}}\\ -\left(\frac{\delta\mathbf{d}_2}{\delta \mathrm{a}}\right)^{\mathbf{T}}\mathbf{T}-\mathbf{d}_2^{\mathrm{T}}\frac{\delta\mathbf{T}}{\delta \mathrm{a}}\end{bmatrix}.\end{aligned} \tag{30}$$

The terms $\mathbf{x_1}$ and $\mathbf{x_2}$ in Equations (27) through (30) are the calibrated image vectors from Equation (11). The partial derivatives of $\mathbf{x_i}$ are

$$\frac{\delta \mathbf{x_i}}{\delta \mathrm{a}} = \mathbf{K}^{-1} \frac{\delta \mathbf{w_i}}{\delta \mathrm{a}}. \tag{31}$$

With these derivatives of the intermediate variables $\mathbf{d_i}$, $\mathbf{T_i}$,and $\lambda_\mathbf{i}$ defined, only the partial derivatives of $\mathbf{w_i}$, $\mathbf{N_i}$, and $\mathbf{C}_{b_i}^n$ need to be found. The variables $\mathbf{w_i}$, $\mathbf{N_i}$, and $\mathbf{C}_{b_i}^n$ are functions of the input variables only. Their partial derivatives are

$$\frac{\delta \mathbf{w_i}}{\delta \mathrm{a}} = \begin{cases} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, & \mathrm{a} = \mathrm{u_i} \\ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, & \mathrm{a} = \mathrm{v_i} \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

$$\frac{\delta \mathbf{N_i}}{\delta \mathrm{a}} = \begin{cases} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, & \mathrm{a} = N_\mathrm{i} \\ \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, & \mathrm{a} = E_\mathrm{i} \\ \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, & \mathrm{a} = D_\mathrm{i} \\ 0, & \text{otherwise} \end{cases} \tag{33}$$

$$\frac{\delta \mathbf{C}_{b_i}^n}{\delta \mathrm{a}} = \begin{cases} \begin{bmatrix} 0 & c_\phi s_\theta c_\psi + s_\phi s_\psi & -s_\phi s_\theta c_\psi + c_\phi s_\psi \\ 0 & c_\phi s_\theta s_\psi - s_\phi c_\psi & -s_\phi s_\theta c_\psi - c_\phi c_\psi \\ 0 & c_\phi c_\theta & -s_\phi c_\theta \end{bmatrix}, & \mathrm{a} = \phi_\mathrm{i} \\ \begin{bmatrix} -s_\theta c_\psi & s_\phi c_\theta c_\psi & c_\phi c_\theta c_\psi \\ -s_\theta s_\psi & s_\phi c_\theta s_\psi & c_\phi c_\theta s_\psi \\ -c_\theta & -s_\phi s_\theta & -c_\phi s_\theta \end{bmatrix}, & \mathrm{a} = \theta_\mathrm{i} \\ \begin{bmatrix} -c_\theta s_\psi & -s_\phi s_\theta s_\psi - c_\phi c_\psi & -c_\phi s_\theta s_\psi + s_\phi c_\psi \\ c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ 0 & 0 & 0 \end{bmatrix}, & \mathrm{a} = \psi_\mathrm{i} \\ 0, & \text{otherwise} \end{cases} . \tag{34}$$

With these partial derivatives defined, it is now possible to construct the Jacobian matrices $\mathbf{J}_1$ and $\mathbf{J}_2$, and use them in Equation (21) to determine $\mathbf{J}$. The Jacobian $\mathbf{J}$ is now ready to be used to calculate the covariance of the feature point's location using Equation (20). However, before that calculation can be performed, the input covariance matrix, $\mathbf{\Omega}$, must be defined.

4.2 DEFINING THE INPUT COVARIANCE MATRIX

The input covariance matrix contains the variances and co-variances of the 16 input variables. It is used to calculate a feature point's 3x3 position covariance matrix after triangulation.

The input covariance matrix (Figure 10) is a 16x16 positive-definite symmetric matrix. The 16 input variables are: the camera position for each observation (in NED coordinates), the camera orientation for each observation (in roll-pitch-yaw Euler angles), and the feature position in each image (in u and v pixel coordinates).

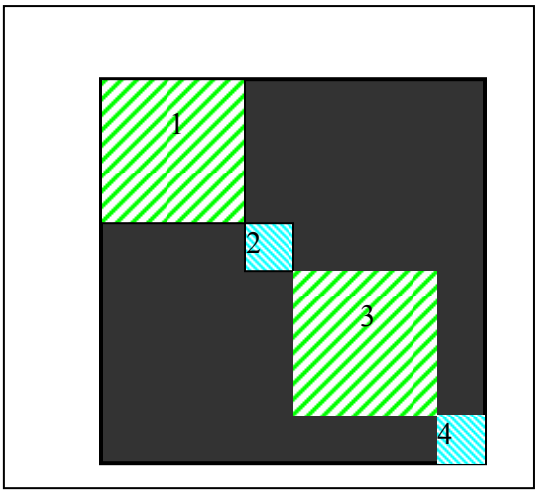

Figure 10. The shape and contents of the input covariance matrix. Regions 1 and 3 contain the NED, phi, theta, psi covariances for the first and second images, respectively. Regions 2 and 4 contain the u,v covariances for images 1 and 2.

In Figure 10, the 16x16 input covariance matrix is divided into four regions. Regions 1 and 3 are the covariances of the outputs of the navigation filter. Regions 2 and 4 are the covariances of the feature location, in pixels. Region 1 is a 6x6 covariance matrix containing entries from a navigation system's estimate of error:

$$\Omega_{1:6,1:6} = \mathrm{E}\left(\begin{bmatrix} (\delta N_1)^2 & \delta E_1\delta N_1 & \delta D_1\delta N_1 & \delta\phi_1\delta N_1 & \delta\theta_1\delta N_1 & \delta\psi_1\delta N_1 \\ \delta E_1\delta N_1 & (\delta E_1)^2 & \delta D_1\delta E_1 & \delta\phi_1\delta E_1 & \delta\theta_1\delta E_1 & \delta\psi_1\delta E_1 \\ \delta D_1\delta N_1 & \delta D_1\delta E_1 & (\delta D_1)^2 & \delta\phi_1\delta D_1 & \delta\theta_1\delta D_1 & \delta\psi_1\delta D_1 \\ \delta\phi_1\delta N_1 & \delta\phi_1\delta E_1 & \delta\phi_1\delta D_1 & (\delta\phi_1)^2 & \delta\theta_1\delta\phi_1 & \delta\psi_1\delta\phi_1 \\ \delta\theta_1\delta N_1 & \delta\theta_1\delta E_1 & \delta\theta_1\delta D_1 & \delta\theta_1\delta\phi_1 & (\delta\theta_1)^2 & \delta\psi_1\delta\theta_1 \\ \delta\psi_1\delta N_1 & \delta\psi_1\delta E_1 & \delta\psi_1\delta D_1 & \delta\psi_1\delta\phi_1 & \delta\psi_1\delta\theta_1 & (\delta\psi_1)^2 \end{bmatrix}\right),$$

where E is the expectation operator. Similarly region 3 is defined as

$$\Omega_{9:14,\ 9:14} = \mathrm{E}\left(\begin{bmatrix} (\delta N_2)^2 & \delta E_2\delta N_2 & \delta D_2\delta N_2 & \delta\phi_2\delta N_2 & \delta\theta_2\delta N_2 & \delta\psi_2\delta N_2 \\ \delta E_2\delta N_2 & (\delta E_2)^2 & \delta D_2\delta E_2 & \delta\phi_2\delta E_2 & \delta\theta_2\delta E_2 & \delta\psi_2\delta E_2 \\ \delta D_2\delta N_2 & \delta D_2\delta E_2 & (\delta D_2)^2 & \delta\phi_2\delta D_2 & \delta\theta_2\delta D_2 & \delta\psi_2\delta D_2 \\ \delta\phi_2\delta N_2 & \delta\phi_2\delta E_2 & \delta\phi_2\delta D_2 & (\delta\phi_2)^2 & \delta\theta_2\delta\phi_2 & \delta\psi_2\delta\phi_2 \\ \delta\theta_2\delta N_2 & \delta\theta_2\delta E_2 & \delta\theta_2\delta D_2 & \delta\theta_2\delta\phi_2 & (\delta\theta_2)^2 & \delta\psi_2\delta\theta_2 \\ \delta\psi_2\delta N_2 & \delta\psi_2\delta E_2 & \delta\psi_2\delta D_2 & \delta\psi_2\delta\phi_2 & \delta\psi_2\delta\theta_2 & (\delta\psi_2)^2 \end{bmatrix}\right).$$

The expected values for each of these variables are obtained from the covariance estimate that is output by the navigation filter.

Regions 2 and 4 in Figure 10 contain the uncertainty in the feature point's u,v coordinates for each picture. These values are characteristic of the feature point detection algorithm.

Region 2 is defined as

$$\Omega_{7:8,7:8} = \mathrm{E}\left(\begin{bmatrix} (\delta u_1)^2 & \delta u_1\delta v_1 \\ \delta u_1\delta v_1 & (\delta v_1)^2 \end{bmatrix}\right).$$

Similarly, Region 4 is defined as

$$\Omega_{15:16,15:16} = \mathrm{E}\left(\begin{bmatrix} (\delta u_2)^2 & \delta u_2\delta v_2 \\ \delta u_2\delta v_2 & (\delta v_2)^2 \end{bmatrix}\right).$$

With these regions of $\mathbf{\Omega}$ defined, they must be populated with actual values before Equation (20) can be used to calculate $\mathbf{P_x}$. The values for each entry in regions 1 and 3 are output from the navigation filter, and regions 2 and 4 are defined by the choice of feature point detection algorithm. Once $\mathbf{\Omega}$ is complete, then $\mathbf{J}$ and $\mathbf{\Omega}$ can be used in Equation (20) to solve for $\mathbf{P_x}$. With this $\mathbf{P_x}$ and $\mathbf{X}$ from Equation (5), we now have an estimate for the position of the feature point and an estimate of the uncertainty of the position.

5. ESTIMATING PROJECTION ERROR

In addition to estimating a landmark's coordinates from a sequence of images, we can estimate where in an image a landmark should appear given its coordinates and an estimate of the camera's location. That is, using the pinhole camera model, we can predict the pixel coordinates of a visual landmark. More formally, given a landmark's position in the navigation frame as well as estimates of the camera's position and attitude, we can estimate u_i and v_i, the pixel coordinates of a feature point, and $\mathbf{P}_{uv}$, the error covariance of the estimation of the pixel coordinates. Combined with estimates of the error in the camera position and landmark location, this pixel error estimate can be used for feature matching, feature-point integrity algorithms, or tightly coupled visual navigation.

We will first derive a closed-form equation for estimating the pixel coordinates of a feature point, and then use the Jacobian, $\mathbf{J}_{uv}$,of this closed-form equation to estimate the error covariance matrix. To estimate a visual-landmark's pixel coordinates, one simply rearranges the triangulation equations from Equation (1) to obtain

$$\mathbf{d_i} = \frac{\mathbf{1}}{\boldsymbol{\lambda}_\mathbf{i}}(\mathbf{X} - \mathbf{T}). \tag{35}$$

Substituting the results from Equations (16) and (17) obtains

$$\mathbf{C}_b^n \mathbf{C}_c^b \mathbf{K}^{-1} \mathbf{w_i} = \frac{\mathbf{1}}{\boldsymbol{\lambda}_\mathbf{i}}(\mathbf{X} - \mathbf{T}). \tag{36}$$

Rearranging Equation (36) to solve for u and v results in

$$\mathbf{w_i} = \begin{bmatrix} u_\mathbf{i} \\ v_\mathbf{i} \\ 1 \end{bmatrix} = \frac{\mathbf{1}}{\boldsymbol{\lambda}_\mathbf{i}} \mathbf{K} \left(\mathbf{C}_c^b\right)^\mathbf{T} \left(\mathbf{C}_b^n\right)^\mathbf{T} (\mathbf{X} - \mathbf{T}). \tag{37}$$

Here, $\boldsymbol{\lambda}_\mathbf{i}$ is the predicted depth of the landmark in the scene. It is obtained through a similar equation:

$$\boldsymbol{\lambda}_\mathbf{i} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{K} \left(\mathbf{C}_c^b\right)^\mathbf{T} \left(\mathbf{C}_b^n\right)^\mathbf{T} (\mathbf{X_i} - \mathbf{T}). \tag{38}$$

We now have a simple set of equations for estimating the pixel coordinates of a feature point. These equations can also be used to define an estimate of the error of the projection.

The method used to estimate the error is the same as in Section 4—the first-order Taylor expansion method of linearization. To simplify the partial derivatives of the projection equation described in Equation (37), we first note that both the equations for $u_\mathbf{i}$, $v_\mathbf{i}$, and $\boldsymbol{\lambda}_\mathbf{i}$ contain the expression $\mathbf{K} \left(\mathbf{C}_c^b\right)^\mathbf{T} \left(\mathbf{C}_b^n\right)^\mathbf{T} (\mathbf{X} - \mathbf{T})$. We will call this expression $\mathbf{h}$, as it represents the homogenous coordinates of the feature point on the imaging plane[6]:

$$\mathbf{h} = \mathbf{K} \left(\mathbf{C}_c^b\right)^\mathbf{T} \left(\mathbf{C}_b^n\right)^\mathbf{T} (\mathbf{X} - \mathbf{T}). \tag{39}$$

With the homogenous coordinates defined, we can express the pixel coordinates and depth of a feature point as a function of $\mathbf{h}$:

$$\begin{bmatrix} u_\mathbf{i} \\ v_\mathbf{i} \\ 1 \end{bmatrix} = \frac{\mathbf{1}}{\lambda_\mathbf{i}} \mathbf{h_i} \tag{40}$$

$$\lambda_\mathbf{i} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \quad \mathbf{h_i}. \tag{41}$$

It is easy to take the partial derivatives of these expressions to use in the calculation of the Jacobian matrix:

$$\frac{\delta \mathbf{w_i}}{\delta \mathrm{a}} = \frac{\lambda_\mathbf{i} \frac{\delta \mathbf{h_i}}{\delta \mathrm{a}} - \frac{\delta \lambda_\mathbf{i}}{\delta \mathrm{a}} \mathbf{h_i}}{(\lambda_\mathbf{i})^2} \tag{42}$$

$$\frac{\delta \lambda_\mathbf{i}}{\delta \mathrm{a}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \frac{\delta \mathbf{h_i}}{\delta \mathrm{a}}. \tag{43}$$

With the partial derivatives of the pixel coordinates and depth derived in terms of the homogenous image coordinates, we now need to calculate the partial derivatives of the homogenous image coordinates:

$$\begin{aligned} \frac{\delta \mathbf{h_i}}{\delta \mathrm{a}} = {} & \frac{\delta \mathbf{K}}{\delta \mathrm{a}} \left(\mathbf{C}_c^b\right)^\mathbf{T} \left(\mathbf{C}_b^n\right)^\mathbf{T} \left(\mathbf{X_i} - \mathbf{T}\right) \\ & + \mathbf{K} \left(\frac{\delta \left(\mathbf{C}_c^b\right)^\mathbf{T}}{\delta \mathrm{a}} \left(\mathbf{C}_b^n\right)^\mathbf{T} \left(\mathbf{X_i} - \mathbf{T}\right) \right. \\ & \quad + \left(\mathbf{C}_c^b\right)^\mathbf{T} \left(\frac{\delta \left(\mathbf{C}_b^n\right)^\mathbf{T}}{\delta \mathrm{a}} \left(\mathbf{X_i} - \mathbf{T}\right) \right. \\ & \quad \left. \left. + \left(\mathbf{C}_b^n\right)^\mathbf{T} \left(\frac{\delta \mathbf{X_i}}{\delta \mathrm{a}} - \frac{\delta \mathbf{T}}{\delta \mathrm{a}} \right) \right) \right). \end{aligned} \tag{44}$$

The partial derivatives of $\mathbf{K}$, $\mathbf{C}_c^b$, $\mathbf{C}_b^n$ and $\mathbf{T}$ are the same as the ones derived in Section 4.1.

Under the assumptions of a constant lever arm, camera calibration, and $\mathbf{C}_c^b$, the pixel coordinates of a feature points are a function of nine variables: the world coordinates of the feature point(three degrees of freedom), and the position and orientation of the camera (six degrees of freedom). With a NED coordinate frame and 3-2-1 NASA aerospace Euler angles, the Jacobian matrix $\mathbf{J}_{uv}$ can be defined as follows:

$$\mathbf{J_w} = \begin{bmatrix} \frac{\delta \mathbf{w_i}}{\delta N_f} & \frac{\delta \mathbf{w_i}}{\delta E_f} & \frac{\delta \mathbf{w_i}}{\delta D_f} & \frac{\delta \mathbf{w_i}}{\delta \phi_f} & \frac{\delta \mathbf{w_i}}{\delta \theta_f} & \frac{\delta \mathbf{w_i}}{\delta \psi_f} & \frac{\delta \mathbf{w_i}}{\delta N_\mathbf{i}} & \frac{\delta \mathbf{w_i}}{\delta E_\mathbf{i}} & \frac{\delta \mathbf{w_i}}{\delta D_\mathbf{i}} \end{bmatrix} \tag{45}$$

$$\mathbf{J}_{uv} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{J_w}.$$

Here, the subscript *f* indicates variables from the navigation filter's current position and attitude estimates and the subscript **i** represents the world coordinates of the **i**th feature point. Note that $\mathbf{w_i} = \begin{bmatrix} u_\mathbf{i} & v_\mathbf{i} & 1 \end{bmatrix}^\mathbf{T}$ is a 3-vector, with a constant third entry, and therefore the third row of $\mathbf{J_w}$ is removed by multiplying $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ by $\mathbf{J_w}$.

With this Jacobian defined, the covariance of the projection can be calculated using the standard method:

$$\mathbf{P}_{uv} = \mathbf{J}_{uv}\mathbf{P}_{in}\left(\mathbf{J}_{uv}\right)^{\mathrm{T}}. \tag{46}$$

Here, $\mathbf{P}_{uv}$ is the predicted covariance matrix for the uv pixel errors, and $\mathbf{P}_{in}$ is the 9x9 covariance matrix of the nine input variables (camera position, camera orientation, and landmark location). The performance of this covariance estimator will be discussed in Section **Error! Reference source not found.**.

6. EVALUATION OF PERFORMANCE

With algorithms to estimate errors in landmark triangulation or landmark projection defined, we must now test how accurately the algorithms estimate the magnitude and direction of errors. To evaluate the performance of the triangulation error covariance estimate, a Monte Carlo simulation was run on the triangulation algorithm. The algorithm was given inputs of known variance. The distribution of the outputs of the triangulation algorithm was compared to the distribution described by the output of the covariance estimation algorithm. The variances of the inputs were changed to illustrate the effect of camera position noise and camera orientation noise separately.

6.1 TRIANGULATION SIMULATION SET-UP

The Monte Carlo simulation was run assuming a system with two cameras placed side-by-side and triangulating a feature point. The experiment set-up is shown in Figure 11.

The two cameras were facing west, with a feature point located approximately 50 meters to the west. The cameras were spaced 10 meters apart in the north direction, and the feature point's projection was calculated using the same camera calibration matrix for each camera. The exact parameters are listed in Table 2.

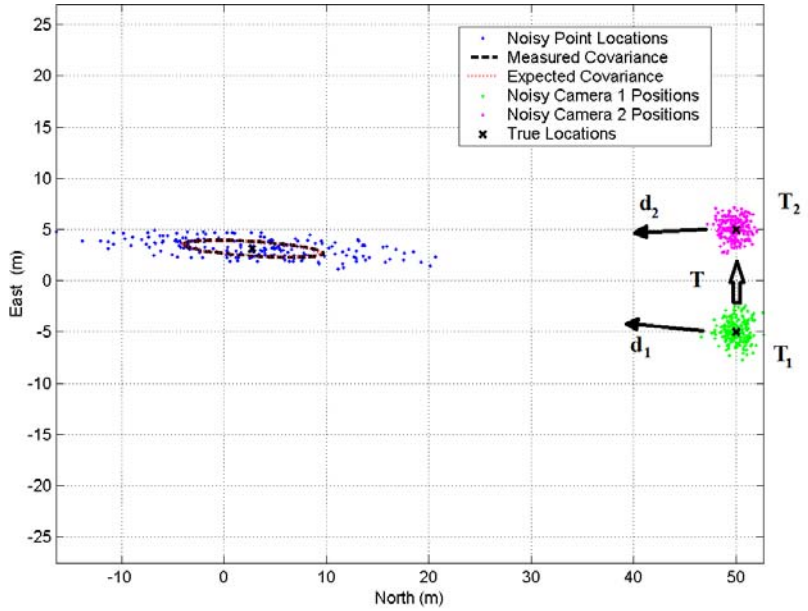

Figure 11. The Monte Carlo simulation set-up.

Table 2. Mean values used for Monte Carlo simulation.

Feature Point Location $\mathbf{X}$ (NED)	$\begin{bmatrix} 3.14 \\ 2.718 \\ -1.414 \end{bmatrix}$ meters
$\mathbf{C}_c^b$	$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ (dimensionless)
$\mathbf{K}$	$\begin{bmatrix} 2136.9 & 0 & 475.1 \\ 0 & 2133.2 & 560.3 \\ 0 & 0 & 1 \end{bmatrix}$ (dimensionless)
$\mathbf{C}_{b_1}^n, \mathbf{C}_{b_2}^n$	$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (dimensionless)
$\mathbf{T}_1$	$\begin{bmatrix} -5 \\ 50 \\ 0 \end{bmatrix}$ (meters)
$\mathbf{T}_2$	$\begin{bmatrix} 5 \\ 50 \\ 0 \end{bmatrix}$ (meters)

The simulation was run under 12 different configurations with varying magnitudes of position, attitude, and pixel error. Each run consisted of 100,000 trials. The noise magnitudes were set up as indicated inTable 3. The standard deviations were the same for camera positions and orientations.

The performance of the error covariance estimation was evaluated by calculating the Mahalanobis distance of the error of each trial in each run. The Mahalanobis distance is calculated by comparing the observed error, $\hat{\mathbf{X}} - \overline{\mathbf{X}}$, with the expected error as predicted by the covariance matrix. The Mahalanobis distance, m, is calculated using the following formula:

$$m^2 = \left(\hat{\mathbf{X}} - \overline{\mathbf{X}}\right)^{\mathrm{T}} \left(\hat{\mathbf{P}}_{\mathbf{x}}\right)^{-1} \left(\hat{\mathbf{X}} - \overline{\mathbf{X}}\right). \tag{47}$$

Here, $\hat{\mathbf{P}}_{\mathbf{x}}$ is the 3x3 covariance matrix calculated using Equation (20), $\hat{\mathbf{X}}$ is the output of the triangulation algorithm given the noisy inputs, and $\overline{\mathbf{X}}$ is the true landmark location.

The Mahalanobis distance is a dimensionless distance that follows a chi-square distribution with the same number of degrees of freedom as the covariance matrix[7]. Thus, the performance of a three-dimensional position estimate was evaluated by comparing the distribution of the Mahalanobis distances, m, of the errors with the expected 3-degree-of-freedom χ^2-distribution.

Table 3. Test Conditions.

Run	Variables	Standard Deviation
A.	Roll,pitch,yaw	0.01 degrees
	N,E,D	1 meter
B.	Roll,pitch,yaw	0.01 degrees
	N,E,D	5 meters
C.	Roll,pitch,yaw	0.01 degrees
	N,E,D	10 meters
D.	Roll,pitch,yaw	1 degree
	N,E,D	1 meter
E.	Roll,pitch,yaw	1 degree
	N,E,D	5 meters
F.	Roll,pitch,yaw	1 degrees
	N,E,D	10 meters
G.	Roll,pitch,yaw	5 degrees
	N,E,D	1 meter
H.	Roll,pitch,yaw	5 degrees
	N,E,D	5 meters
I.	Roll,pitch,yaw	5 degrees
	N,E,D	10 meters
J.	Roll,pitch,yaw	10 degrees
	N,E,D	1 meter
K.	Roll,pitch,yaw	10 degrees
	N,E,D	5 meters
L.	Roll,pitch,yaw	10 degrees
	N,E,D	10 meters

6.2 TRIANGULATION SIMULATION RESULTS

The results from the Monte Carlo simulation are shown in Figure 12. The results from each of the twelve runs are arranged to show how the performance varies as a function of position noise magnitude and angle noise magnitude.

Figure 12 contains a top-down view of the distribution of points. The largest uncertainty is in the direction perpendicular to the camera plane. This is because the errors in the locations and orientations of the two cameras were uncorrelated, and thus the uncertainty in the parallax between the two views of the landmark caused the depths to be miscalculated. Nevertheless, because the covariance of the input errors was known, the distribution of the errors in the triangulation solution was accurately predicted.

For smaller values of Euler angle noise, the error distribution is accurately predicted by the algorithm presented in this report. This is expected, as the camera triangulation equation is a linear function of the camera position but a nonlinear (trigonometric) function of the camera orientation's Euler angles. Thus, the larger the error in angle, the less accurate the Taylor expansion will be. Furthermore, at larger errors for angles the position error becomes biased, and points tend to be triangulated at a location closer than the truth value. These biases, however, appear at large values of angle noise standard deviation (10 degrees), which is much larger than the gyro noise from tactical-grade inertial measurement units. For example, if triangulation for a visual landmark is performed on pictures taken 10 seconds apart, Euler angle standard deviations are rarely more than 1 degree. Tactical-grade inertial measurement units (IMUs) exhibit gyro noise that is usually on the order of 0.03-0.1 $^\circ/\sqrt{\text{hr}}$ or 1.8-6 $^\circ/hr/\sqrt{\text{Hz}}$ [8]. A navigation system with 0.1 $^\circ/\sqrt{\text{hr}}$ Gaussian gyro noise, sampling the gyro at 100 Hz, would see uncorrelated gyro rate errors of about 0.0167 $^\circ/\text{sec}$ per sample. Summed over 10 secs (1,000 samples), these gyro rate errors would result in Euler angle errors with standard deviations of about 0.00528 degree per axis.

shows similar figures for other grades of IMUs.

Because of the small magnitude of random Euler angle errors from even inexpensive gyroscopes, the presented algorithm for triangulation is suitable for triangulation from navigation systems. For triangulation with camera attitudes that do not have quality attitude measurements, the presented method accurately estimates the triangulation error for angle noise with a standard deviation smaller than 5 degrees. Figure 13 illustrates the accuracy of the triangulation error-estimation in a different fashion. Here, the distribution of Mahalanobis distances is compared to the χ^2-distribution for three degrees of freedom.

As shown in Figure 13, for small values of angle variance the error estimate is accurate, and the expected error distribution follows the actual error distribution. However, as the standard deviation of the Euler angle noise approaches 5 degrees for roll, pitch, and yaw, the error estimate begins to degrade. As the position noise increases for a fixed amount of angle noise, the position noise has a larger weight on the final error than the angle noise and the estimate regains its accuracy.

Table 4. Comparison of random noise from different grades of gyroscopes[9].

	Silicon-Vibratory MEMs	Tactical-grade IFOGs	Aviation-Grade Spinning Mass
Random gyro rate noise	$1^\circ/\sqrt{\text{hr}}$	$0.1^\circ/\sqrt{\text{hr}}$	$0.002^\circ/\sqrt{\text{hr}}$
Random gyro rate noise per sample at100 Hz	$0.167^\circ/\text{sec}$	$0.0167\ ^\circ/\text{sec}$	$3.3\text{x}10^{-4}\,^\circ/\text{sec}$
Random Euler angle error standard deviation after 10 seconds (1,000 samples at 100 Hz)	0.0528°	0.00528°	$1.0\text{x}10^{-5}$°
Random Euler angle error standard deviation after 90 seconds (9,000 samples at 100 Hz)	0.1584°	0.01584°	$3.0\text{x}10^{-5}$°

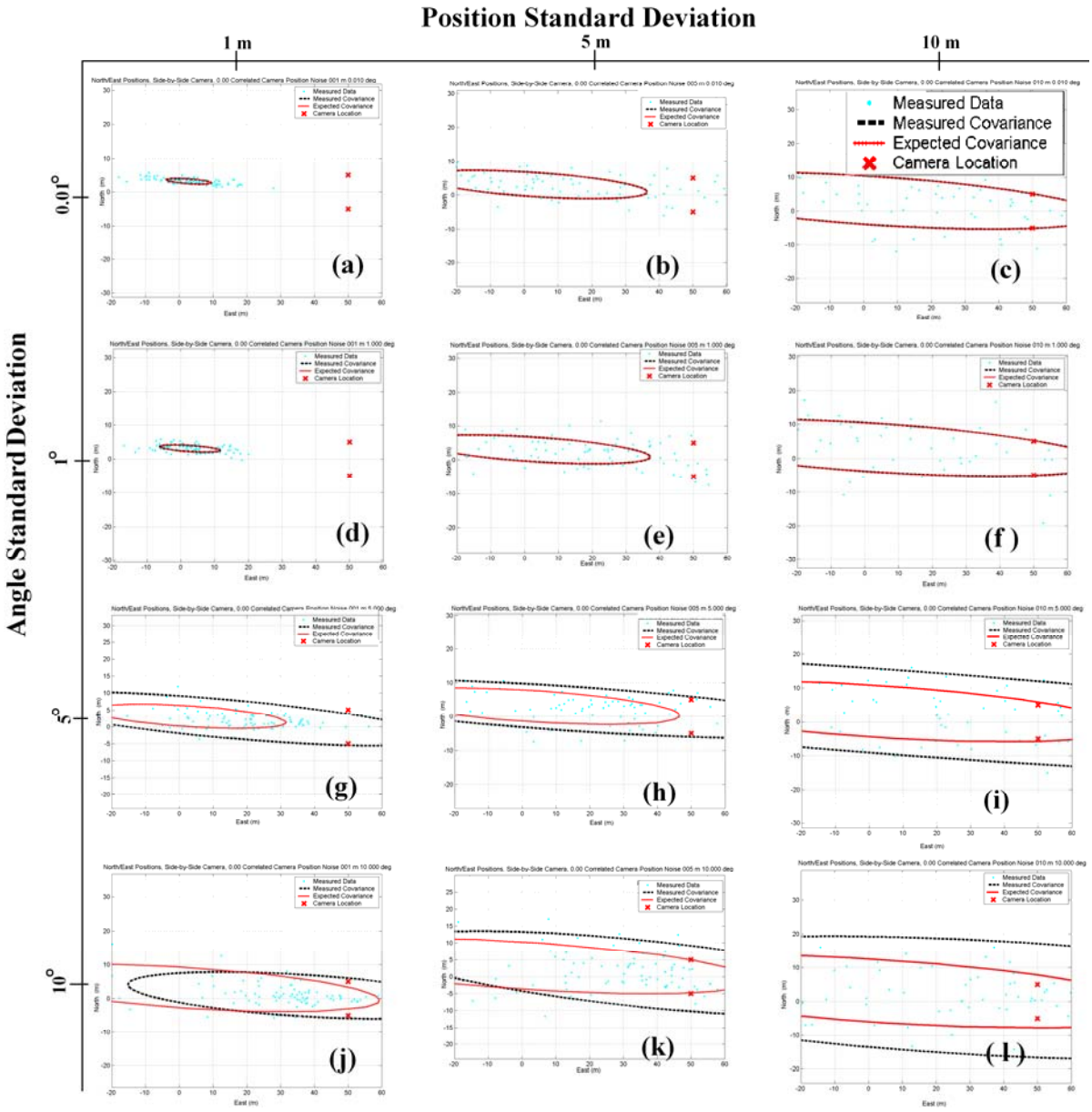

Figure 12. The spread of the triangulation solution values in the North/East plane. The x-axis in each sub-chart represents east coordinates, in meters; the y-axis represents north coordinates, in meters. The measured covariance, represented by dashed black ellipses, is the covariance calculated from the samples. The expected covariance, represented by solid red ellipses, is the distribution around the true point location calculated by the covariance estimation algorithm. For small values of Euler angle noise variance (1 degree or less), the predicted distribution is accurate enough that the expected and measured covariance ellipses overlap in the figure.

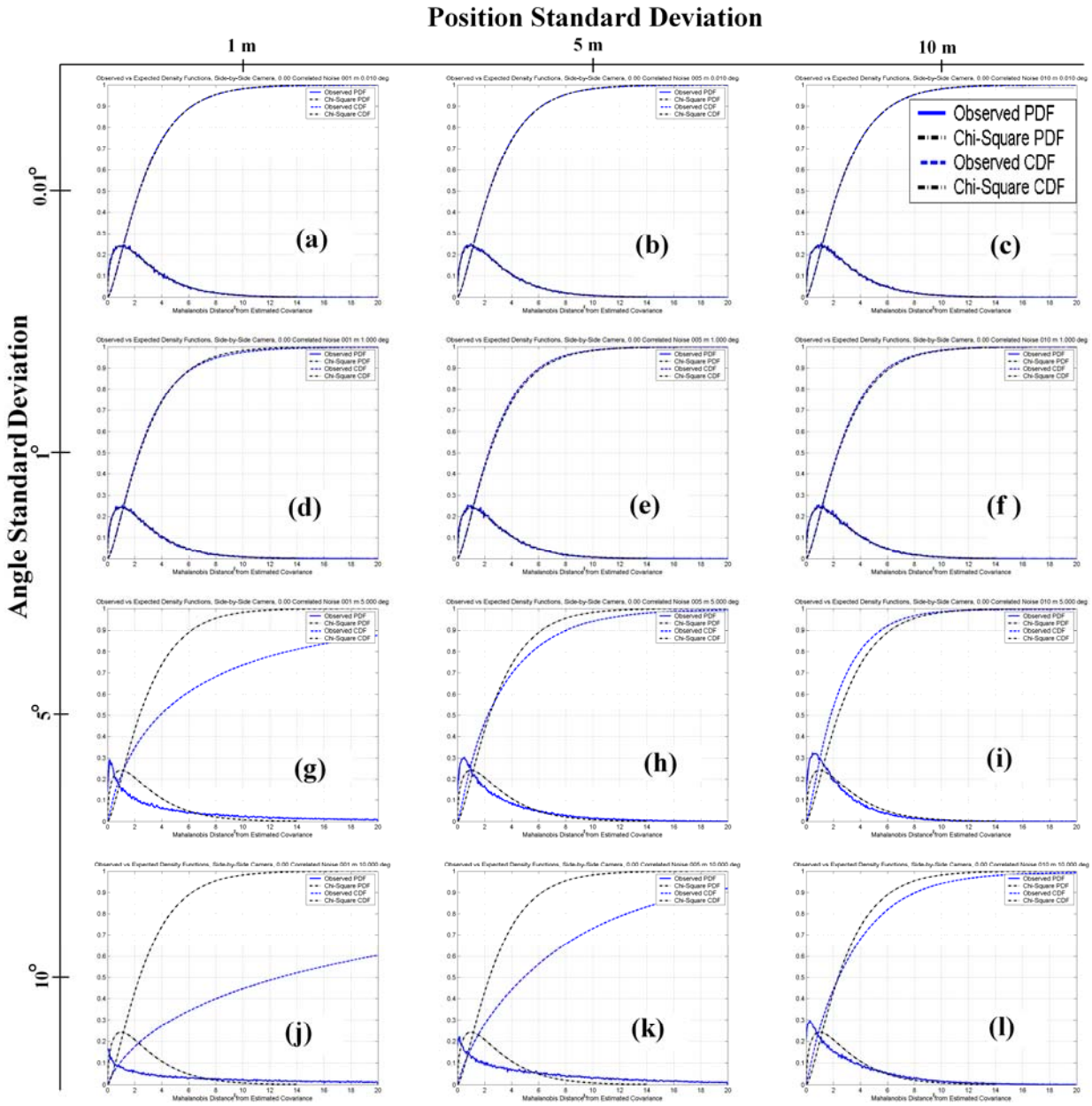

Figure 13. The distribution of errors versus the expected distribution of errors. The blue lines represent observed distributions while the black lines represent expected distributions. The observed and expected distributions match up well for all cases except for large attitude errors combined with small camera position errors (Figures g and j).

6.3 PROJECTION COVARIANCE ESTIMATION

The projection covariance estimator derived in Section 5 was verified using a Monte Carlo simulation. In this simulation, a point with a noisy location was projected onto a camera with noisy position and attitude. The results of this simulation are shown in Figure 14.

Figure 14 shows that the observed distribution of uv coordinates follows the expected distribution. That is, the shape of the error distribution on the uv-plane is accurately predicted.

To evaluate the performance of the error estimate, the Mahalanobis distance is calculated using the following equation:

$$m = \sqrt{\left(\begin{bmatrix}\hat{u}\\ \hat{v}\end{bmatrix} - \begin{bmatrix}\bar{u}\\ \bar{v}\end{bmatrix}\right)^{\mathbf{T}} \left(\hat{\mathbf{P}}_{uv}\right)^{-1} \left(\begin{bmatrix}\hat{u}\\ \hat{v}\end{bmatrix} - \begin{bmatrix}\bar{u}\\ \bar{v}\end{bmatrix}\right)}. \tag{48}$$

Here, $\hat{\mathbf{P}}_{uv}$ is the covariance matrix calculated using Equation (46), $\hat{u}$ and $\hat{v}$ are from the output of the projection estimation Equation (37) given the noisy inputs, and $\bar{u}$ and $\bar{v}$ are the true pixel coordinates.

The Mahalanobis distances for the u and v errors follow a 2-dof χ^2-distribution, as shown in Figure 15.

The distribution of the Mahalanobis distances of the u,v coordinate errors follows the expected Chi-square distribution. That is, the magnitude of error is accurately predicted.

The simulation results indicate that the projection covariance estimator is accurate when characteristics of the input noise are known.

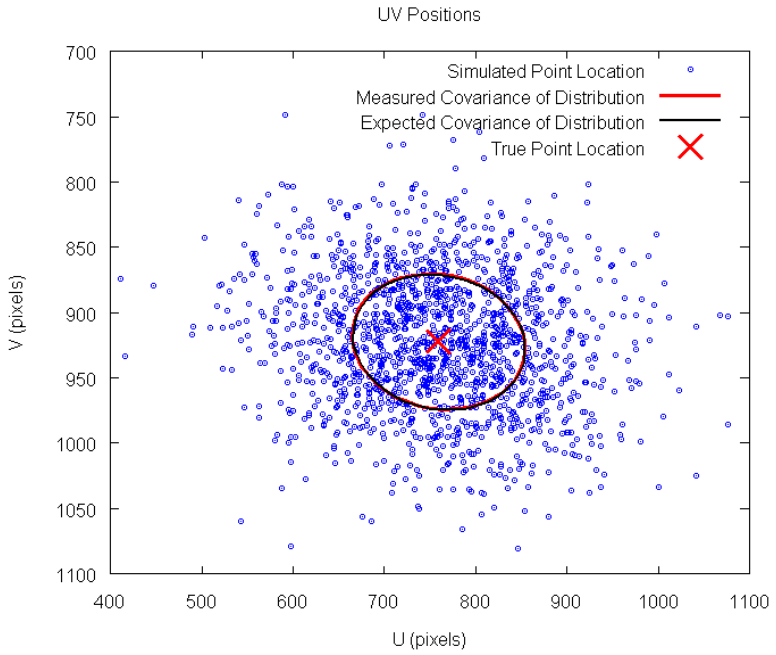

Figure 14. The results of the Monte Carlo simulation of UV projection error.

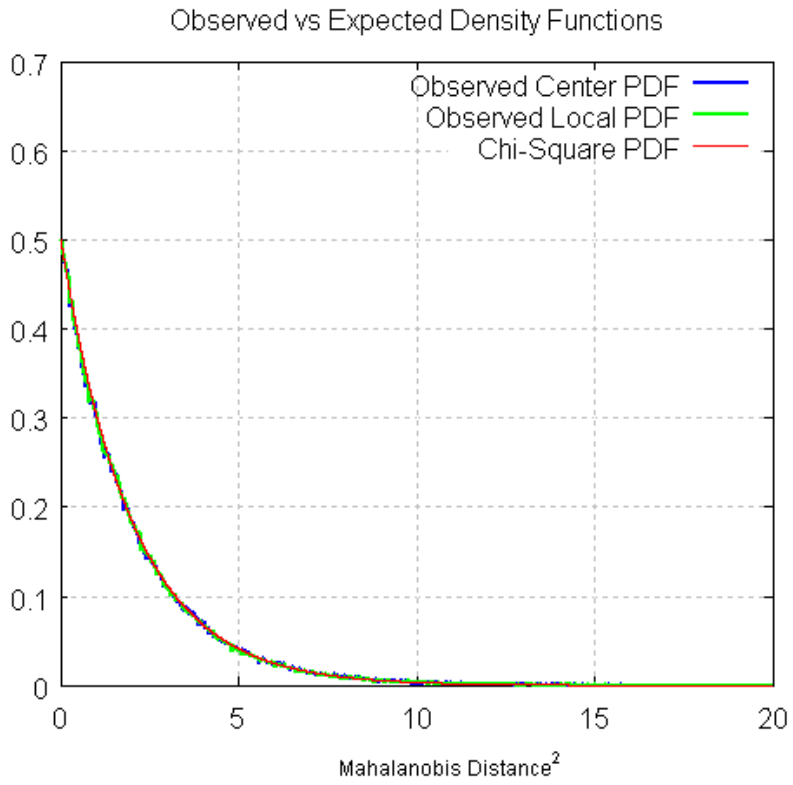

Figure 15. The distribution of Mahalanobis distances for the Monte Carlo simulation of UV projection errors.

The distribution of the Mahalanobis distances of the uv coordinate errors follows the expected Chi-square distribution. That is, the magnitude of error is accurately predicted.

The simulation results indicate that the projection covariance estimator is accurate when characteristics of the input noise are known.

7. CONCLUSION

This report presented an algorithm for triangulating the location of a visual landmark from two camera observations. A method for estimating the accuracy (covariance) of the triangulation solution was also presented. For the most part, the covariance estimation algorithm described in this paper provides an accurate estimate of the triangulation error. However, it is important to note that this algorithm is limited to applications where Euler angle noise is smaller than 5 degrees. This is because the main decider of the accuracy of the error estimate is the magnitude of the Euler angle noise. As the magnitude of Euler angle noise approaches 5 degrees for each axis, the linear approximations made by the presented algorithm no longer accurately predict errors. Thus, for Euler angle noise greater than 5 degrees, the triangulation results and error estimates become un-reliable. However, most navigation systems have an Euler angle noise standard deviation much smaller than 5 degrees (on the order of half a degree for inexpensive microelectromechanical systems (MEMS) or magnetometers)[10], so this non-linearity is a non-issue. Therefore, in systems where the camera's attitude is determined using inexpensive MEMS gyros or better, the linear triangulation method and its associated covariance estimator remain useful for mapping visual landmarks and estimating the position errors.

[1] Yi Ma, Stefano Soatto, Jana Kosecka, S. Shankar Sastry, *An Invitation to 3-D Vision*, Springer-Verlag, New York, 2010, p. 130.

[2] Lee Lemay, Chen-Chi Chu, Demoz Gebre-Egziabher, Rohan Ramlall, "Precise Input and Output Error Characterization for Loosely Integrated INS./GPS/Camera Navigation System" *Proceedings of the 2011 International Technical Meeting of The Institute of Navigation*, San Diego, CA, January 2010, pp. 880-894

[3] Ma, Soatta, Kosecka, Sastry, p. 55.

[4] Baruh, H., *Analytical Dynamics*, New York, NY: McGraw-Hill, 1999

[5] Robert Grover Brown, Patrick Y.C Hwang, *Introduction To Random Signals and Applied Kalman Filtering*, 2nd Edition, John Wiley & Sons, New York, 1992, p. 61.

[6] Richard Hartley, Andrew Zisserman, *Multiple View Geometry in Computer Vision*, 2nd edition, Cambridge University Press, 2010. p. 3

[7] Prasanta Chandra Mahalanobis , "On the generalized distance in statistics" Proceedings of the National Institute of Sciences of India, Calcutta, India, January 1936, pp. 49-55.

[8] P. D. Groves, *Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems*, Artech House, Boston, London, 2008, p. 117.

[9] Groves, p. 116.

[10] Groves, p. 117.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-01-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
January 2012	Final	

4. TITLE AND SUBTITLE

Algorithm for Triangulating Visual Landmarks and Determining Their Covariance

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS

Justin Gorgen
Lee LeMay

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

SSC Pacific, 5622 Hull Street, San Diego, CA 92152–5001

8. PERFORMING ORGANIZATION REPORT NUMBER

TD 3252

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

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217-5660

10. SPONSOR/MONITOR'S ACRONYM(S)

ONR

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

12. DISTRIBUTION/AVAILABILITY STATEMENT

Authorized for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

We present a least squares method for triangulating a landmark location given the bearing towards it from two known locations. For the three-dimentional case, this method is shown to be equivalent to finding the mid-point method of triangulation. This method is then extended to triangulating three-dimensional coordinates for viual landmark from two calibrated images of the landmark taken from known camera locations. Triangulation error and error covariance is estimated by propagating camera position, attitude, and pixel registration errors through a first-order Taylor expansion of the closed-form solution. The performance of the covariance estimator is evaluated and found to be a reliable estimate except in the presencse of large camera attitude errors. The error estimate is accurate for camera Euler angle errors with standard deviations of less than 5 degrees.

15. SUBJECT TERMS

triangulation error covariance least-squares method first-order Taylor expansion

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Justin Gorgen
U	U	U	U	40	19B. TELEPHONE NUMBER *(Include area code)* (619) 553-5234

INITIAL DISTRIBUTION

84300	Library	(2)
85300	Archive/Stock	(1)
85300	S. Baxley	(1)
56480	J. Gorgen	(18)

Defense Technical Information Center
Fort Belvoir, VA 22060–6218 (1)

SSC San Diego Liaison Office
C/O PEO-SCS
Arlington, VA 22202–4804 (1)

Center for Naval Analyses
Alexandria, VA 22311–1850 (1)

Approved for public release; distribution is unlimited.

SSC Pacific
San Diego, CA 92152-5001