

202000-30-T

AD-A206 588

Technical Report—Study Service Contract

GADS ALGORITHM DEVELOPMENT UNDER ENGINEERING ANALYSIS TASK

J.W. SHERMAN

Image Processing Systems Division

MARCH 1989

(Period September 1988—December 1988)

Department of the Air Force
Armament Division
Eglin AFB, FL 32542-5320
F08635-87-C-0074

DTIC
ELECTE
S 12 APR 1989 D
E



This document has been approved
for public release and sale in
distribution is unlimited.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

ADA206588

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a REPORT SECURITY CLASSIFICATION Unclassified			1b RESTRICTIVE MARKINGS None		
2a SECURITY CLASSIFICATION AUTHORITY			3 DISTRIBUTION/AVAILABILITY OF REPORT Unlimited		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE					
4 PERFORMING ORGANIZATION REPORT NUMBER(S) 202000-30-T			5 MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Environmental Research Institute of Michigan		6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION Department of the Air Force Armament Division		
6c. ADDRESS (City, State, and ZIP Code) P.O. Box 8618 Ann Arbor, MI 48107-8618			7b. ADDRESS (City, State, and ZIP Code) AD/KRT Eglin AFB, FL 32542-5320		
8a. NAME OF FUNDING /SPONSORING ORGANIZATION Department of the Air Force Armament Division		8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F08635-87-C-0074		
8c. ADDRESS (City, State, and ZIP Code) AD/KRT Eglin AFB, FL 32542-5320			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.
					WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) GADS Algorithm Development Under Engineering Analysis Task (U)					
12. PERSONAL AUTHOR(S) James W. Sherman					
13a. TYPE OF REPORT Technical		13b. TIME COVERED FROM 9/88 TO 12/88		14. DATE OF REPORT (Year, Month, Day) 1989 March	
15. PAGE COUNT 18					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP			
19 ABSTRACT (Continue on reverse if necessary and identify by block number)					
→ Under the Engineering Analysis Task of Contract F08635-87-C-0074, two development efforts pertaining to GADS (Graphic Attitude Determination System) algorithms were pursued. These efforts resulted in two upgrades to the GADS algorithms. The first upgrade resulted from a direct linear algebra solution for the model drawing equations for a body of revolution. The second upgrade resulted from developing an error analysis that is driven by the model being displayed to determine an estimated covariance for the errors in the position and attitude of the model being fit to the displayed image. (K) C					
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION Unclassified		
22a NAME OF RESPONSIBLE INDIVIDUAL James W. Sherman			22b TELEPHONE (Include Area Code) (313) 994-1200 x2829		22c OFFICE SYMBOL

TABLE OF CONTENTS

1.0	INTRODUCTION	1
2.0	IMPLEMENTATION OF ERIM ALGORITHM 3 FOR MODEL DRAWING ...	3
2.1	LINEAR ALGEBRA SOLUTION FOR TANGENT POINT	3
2.2	REVISED DISPLAY ALGORITHM	7
3.0	GENERATING A THEORETICALLY BASED ERROR COVARIANCE MATRIX	11
3.1	DERIVATION OF COVARIANCE MATRIX	15
3.2	IMPLEMENTATION OF COVARIANCE COMPUTATION ALGORITHM	16

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



1.0 INTRODUCTION

Under the Engineering Analysis Task of Contract F08635-87-C-0074, two development efforts pertaining to GADS (Graphic Attitude Determination System) algorithms were pursued. These efforts resulted in two upgrades to the GADS algorithms. The first upgrade resulted from a direct linear algebra solution for the model drawing equations for a body of revolution. The second upgrade resulted from developing an error analysis that is driven by the model being displayed to determine an estimated covariance for the errors in the position and attitude of the model being fit to the displayed image.

2.0 IMPLEMENTATION OF ERIM ALGORITHM 3 FOR MODEL DRAWING

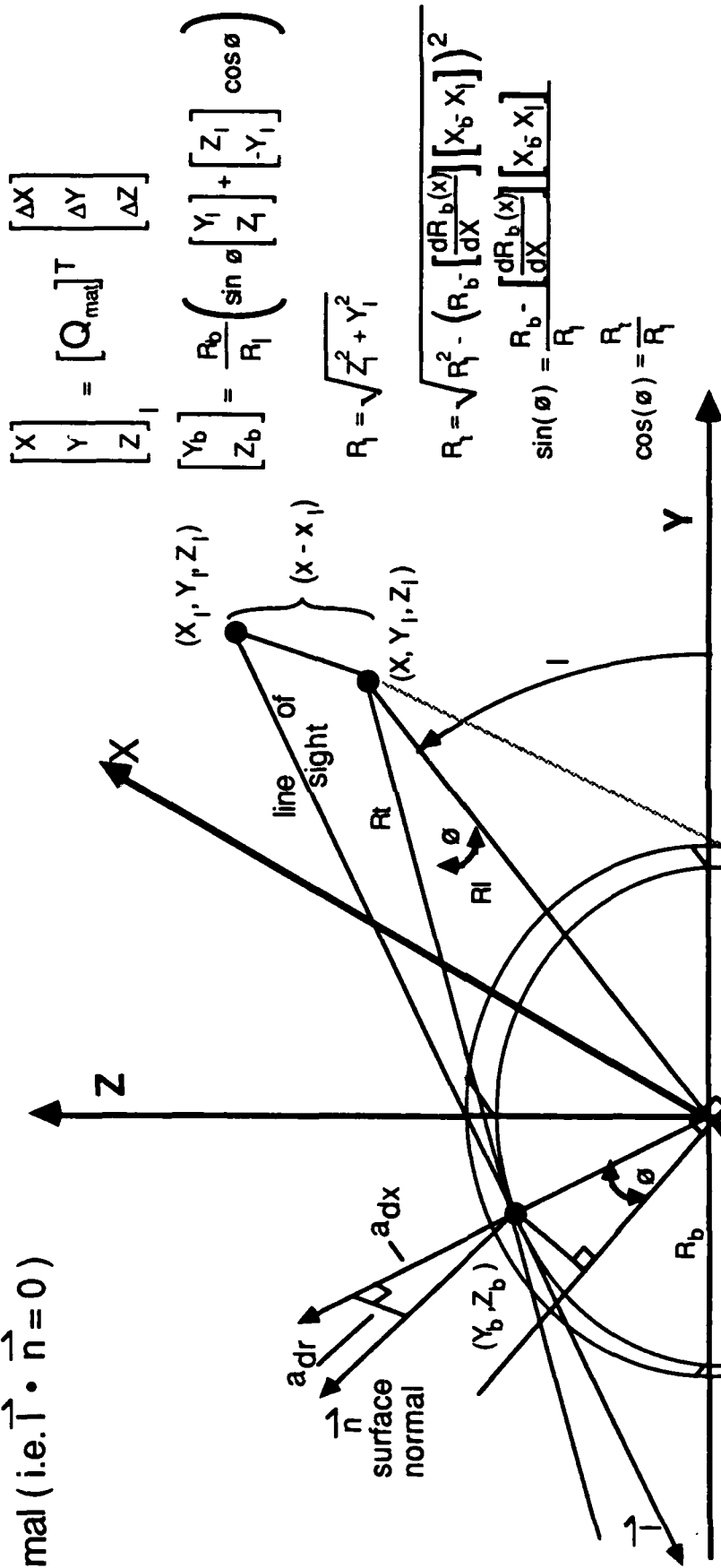
From several of the project reviews the model drawing algorithm discussed here became known as ERIM algorithm 3. For historical background, ERIM algorithm 1 used a linear algebra solution to find the visible edge by finding the plane through the axis of revolution of the body and normal to the plane containing the line-of-sight and the axis of revolution. This algorithm was used initially to increase the model drawing speed. ERIM algorithm 2 used a linear algebra solution to compute a point on the visible edge at each station along the axis, by projecting the line-of-sight into the plane normal to the axis of revolution and then computing the tangent points of that projection with the circle corresponding to the body. This corresponds to a cylindrical approximation. The resulting models were approximately equivalent to the original GADS algorithm and executed more rapidly. ERIM algorithm 3 uses a more complete linear algebra solution and is based on finding the line-of-sight which is orthogonal to the surface normal of the body of revolution at each station along the axis of revolution. This algorithm retains most of the speed of algorithm 2, handles the nearly end-on case correctly, and provides for a depth encoded outline of the model to be presented on the display.

2.1 LINEAR ALGEBRA SOLUTION FOR TANGENT POINT

Assume that the body of revolution is modeled by a series of truncated cones which have a common axis (herein taken to be the X-axis). This is the form used in the GADS model database. An additional assumption will be used: that each individual "cylinder" in the data base is smooth. The original data base often has objects with steps or slope changes combined. The reason for the additional assumptions is to give the user control over how the visible edges potentially created by steps and slope changes are displayed.

The linear algebra solution to the tangent point of a line-of-sight with a truncated cone is illustrated in Figure 1. The X-axis

Solution 3 is based on line-of-sight being orthogonal to the surface normal (i.e. $\vec{l} \cdot \vec{n} = 0$)



If R_l is imaginary the whole circle is in view (the viewing point lies within the extended cone). Thus plot the largest circle that is fully in view.

Figure 1 Illustration of Solution 3

("into the paper") is the axis of revolution of the cone. The subscripts b and l denote body and lens. The solution is based on computing the two lines-of-sight, $\bar{T}$, (a quadric equation) which are orthogonal to the normal to the surface of the truncated cone,

$$\bar{T} \cdot \bar{n} = 0 .$$

The two solutions, when they exist, correspond to the two visible edges of the cone which are on opposite sides of the cone. The visible edges are straight lines and correspond to the intersections of the cone with its tangent planes that pass through the origin of the line-of-sight. To derive the solution, first let the origin of the line-of-sight be transformed into the coordinate system of the body of revolution. The x component, $X_b - X_l$, is the distance from the lens to the plane normal to the axis of revolution at the station, X_b , of interest along the axis. The vector $[Y_l, Z_l]^T$ is the projection of the line from the lens to the axis of revolution onto the same plane, whose length is R_l . The radius of the body of revolution at X_b is $R_b(X_b)$, where functional notation will be dropped for simplicity. Thus $\bar{T}$ becomes

$$\bar{T} = \begin{bmatrix} X_b - X_l \\ Y_b - Y_l \\ Z_b - Z_l \end{bmatrix}$$

The surface normal depends on the position on the circle of revolution. The component of the normal in the plane of revolution is radial and thus proportional to the radius vector. The component in the x-direction is directly proportional to both the radius and the slope of the cone. Thus $\bar{n}$ becomes,

$$\bar{n} \propto \begin{bmatrix} -R_b \frac{dR_b(X_b)}{dx} \\ Y_b \\ Z_b \end{bmatrix} .$$

Setting $\vec{T} \cdot \vec{n} = 0$ gives

$$0 = -(X_b - X_1)R_b \frac{dR_b}{dx} + (Y_b - Y_1)Y_b + (Z_b - Z_1)Z_b$$

Because there are two unknowns Y_1 and Z_1 , a second equation is needed. The radius of revolution provides the needed constraint,

$$R_b^2 = Y_b^2 + Z_b^2.$$

Expressing the position on the circle of revolution in terms of components normal and colinear with the projection of the line-of-sight onto the plane of revolution gives

$$\begin{bmatrix} Y_b \\ Z_b \end{bmatrix} = \frac{R_b}{R_1} \left\{ \sin \phi \begin{bmatrix} Y_1 \\ Z_1 \end{bmatrix} + \cos \phi \begin{bmatrix} Z_1 \\ -Y_1 \end{bmatrix} \right\}$$

where ϕ is the angle between the lines from $[X_b, Y_1, Z_1]^T$ (the projected origin of the line-of-sight) to $[X_b, 0, 0]^T$ (the center of the circle of revolution) and to $[X_b, Y_b, Z_b]^T$ (the tangent point).

Combining these equations gives the following equations for $\sin \phi$ and $\cos \phi$.

$$\begin{aligned} 0 &= -\frac{dR_b}{dx} (X_b - X_1) R_b \\ &\quad + \frac{R_b^2}{R_1^2} \left[\sin \phi Y_1 + \cos \phi Z_1 - \frac{R_1}{R_b} Y_1 \right] \left[\sin \phi Y_1 + \cos \phi Z_1 \right] \\ &\quad + \frac{R_b^2}{R_1^2} \left[\sin \phi Z_1 - \cos \phi Y_1 - \frac{R_1}{R_b} Z_1 \right] \left[\sin \phi Z_1 - \cos \phi Y_1 \right] \\ 0 &= -\frac{dR_b}{dx} (X_b - X_1) + \frac{R_b}{R_1^2} \left[R_1^2 - \frac{R_1}{R_b} \left(R_1^2 \sin \phi \right) \right] \end{aligned}$$

$$\sin \phi = \frac{R_b - \frac{dR_b}{dx} (x_b - x_1)}{R_1}$$

$$\cos \phi = \frac{\sqrt{R_1^2 - \left[R_b - \frac{dR_b}{dx} (x_b - x_1) \right]^2}}{R_1} = \frac{R_t}{R_1}$$

where R_t is the length of the tangent from the origin of the line-of-sight to the cone extended to x_1 . If the value is imaginary, the origin of the line-of-sight (viewing point) lies within the extended cone and there are no visible edges.

2.2 REVISED DISPLAY ALGORITHM

The revisions to the model display algorithm have two purposes: Some implement the new display computations and some provide additional speed. The listing of the CIRCLE 3 module in Appendix A contains the important changes.

The general organization of the module has been changed to perform the display calculations in three passes over the data for each "cylinder" of the model. The first pass analyzes each truncated cone to determine which viewing case to use and compute intermediate results. The second pass computes the tangent points, marks beginning and end of visible edges, and determines where to draw circles. The third pass computes the data structure for driving the display routines. Before the first pass, the position of the camera is transformed into "cylinder" coordinates. The result is used to form "unit" vectors corresponding to the axis of revolution $[x_1, 0, 0]^T$, the projection of the origin of the line-of-sight onto the plane of revolution $[0, y_1, z_1]^T$, and normal to that projection in the same plane $[0, x_1, -y_1]^T$. These "unit" vectors are rotated (not translated) into the camera coordinate system. This allows the tangent points to be computed directly in camera coordinates, which saves time by not having

to rotate each point. The subroutines MM2 performs only rotation, compared to MM3 which performs both rotation and translation.

The first pass analyzes each cone by computing its slope, the radius

$$\left[R_b - \frac{dR_b}{dx} (x_b - x_1) \right]^2$$

of the extended cone at x_1 , and R_t . A total of seven cases are delineated:

1. Inside the extended cone between truncating plane and infinity.
2. Inside the truncated cone.
3. Inside the extended cone between truncating plane and apex.
4. Inside the extended cone beyond apex.
5. Outside the extended cone on apex side.
6. Outside the truncated cone between truncated planes (includes the degenerate case of a cylinder, $DRDX = 0$).
7. Outside the extended cone away from apex side.

The values of the slope and the radius are extended to the last point of the model so that the second pass will compute the second tangent point on the last truncated cone.

The second pass computes the points to be connected for the visible edge and where circles are to be drawn. Cases 3, 4, and 5 correspond to visible edges, and the tangent points are computed using the values from the first pass. Non-zero radius circles are drawn at both ends and wherever the "case" changes between visible and hidden or between two lines hidden on opposite sides of the extended cone apex.

The third pass computes the display data structure, first for the visible edges and then for the circles. It also stores the computed depth of each point for depth encoding of the lines on the display. This depth encoding is accomplished by the Cytocomputer after model movement stops, when requested by the operator.

3.0 GENERATING A THEORETICALLY BASED ERROR COVARIANCE MATRIX

The purpose of this effort was to develop a covariance matrix for the errors in position and orientation estimates obtained using manual GADS. The basic underlying assumption made is that a human operator matching a wireframe model to an image is using the location of edges and is minimizing the "overall" error of the match. This process has been modeled by a least squares estimation process, using linearized equations to relate the model parameters (position and orientation) to displacements of pixels in the image.

The wireframe model used by GADS is made from line segments which model planar structures (e.g., fins) and bodies of revolution. The bodies of revolution are modeled as a smoothly varying series of truncated cones, which result from rotating a curve made from a series of line segments. Thus, to develop the least squares estimator, we will examine the matching of a line segment to features in an image.

Assume that the line segment is drawn between two points of the model, $\bar{p}_1$ and $\bar{p}_2$ and that a set of edge features were matched to the line at locations $\bar{f}_i$, $i = 1, \dots, I$. Let the position and orientation parameters of the model be denoted as a vector $\bar{a}$,

$$\bar{a}^T = [t_x, t_y, t_z, r, p, y]$$

where t_x = Camera x translation, in inches from model origin

t_y = Camera y translation, in inches from model origin

t_z = Camera z translation, in inches from model origin

r = Model roll angle

p = Model pitch angle

y = Model yaw angle

The position, $\bar{x}^c = [x^c, y^c, z^c]^T$, in camera coordinates of a model point, $\bar{x}^m = [x^m, y^m, z^m]^T$, is given by

$$\begin{bmatrix} x^c \\ y^c \\ z^c \\ 1 \end{bmatrix} = \begin{bmatrix} c_p c_y & -s_p & -c_p s_y & t_x \\ c_r s_p c_y - s_r s_y & c_r c_p & -c_r s_p s_y - s_r c_y & t_y \\ s_r s_p c_y + c_r s_y & s_r c_p & -s_r s_p s_y + c_r c_y & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^m \\ y^m \\ z^m \\ 1 \end{bmatrix}$$

where s_p , c_p denote sine and cosine of the subscript p (pitch). A perspective transformation brings points in camera coordinates into image coordinates,

$$\bar{p} = \begin{bmatrix} x^c (z^c)^{-1} k_1 + k_2 \\ y^c (z^c)^{-1} k_3 + k_4 \end{bmatrix}$$

where k_1 , k_2 , k_3 , and k_4 are display-dependent constants.

In the matching process the operator adjusts the parameters, $\bar{a}$, so that the line segment has the same slope as the edge and is placed on the edge. This corresponds to "putting a straight line" through the points $(\bar{f}_i, i = 1, l)$ which represent the edge. Let $\bar{u}_l$ and $\bar{u}_n$ be unit vectors in the direction of the line segment and normal to it respectively,

$$\bar{u}_l = \frac{\bar{p}_1 - \bar{p}_2}{|\bar{p}_1 - \bar{p}_2|}, \quad \bar{u}_n = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \bar{u}_l.$$

Then the distances of each feature point from the line, d_i , and along the line, l_i , can be computed by

$$d_i = (\bar{f}_i - \bar{p}_c)^T \bar{u}_l,$$

$$l_i = (\bar{f}_i - \bar{p}_c)^T \bar{u}_n,$$

and

$$\bar{p}_c = \frac{1}{2} (\bar{p}_1 - \bar{p}_2) .$$

where $\bar{p}_c$ is the center of the line segment. For short line segments, only the displacement will be used now in the fitting process. The analysis can be expanded to include slope for long line segments. The least squares fit to an individual line segment of a displacement, Δd , along the normal is given by

$$\Delta d = \frac{1}{I} \sum_{i=1}^I d_i .$$

The errors of this process depend on the ability to locate edge features. We will assume that the variance, σ_d^2 , of the errors in locating the edge (measuring d_i) is constant over the image but dependent on image quality. However, the d_i will not be independent for closely spaced d_i . Therefore, we will assume a constant decorrelation distance, λ_d , along an edge which again depends on image quality. The variance of Δd can then be expressed approximately as

$$\sigma_{\Delta d}^2 = \frac{\lambda_d \sigma_d^2}{|\bar{p}_1 - \bar{p}_2|}$$

These displacements from each line segment of the model are "averaged" over the whole model by the operator in adjusting the model parameters, $\bar{a}$, to minimize the overall error. This process will be represented by the least squares fit of the displacements for line segments of the model using linearized equations. Let the displacements for all the line segments of the model be denoted by

$$\Delta d_j, j = 1, . . . , J,$$

The displacements can be expressed in terms of increments in the model parameters, $\bar{\Delta a}$, by linearizing about the estimate of the model param-

eters, $\bar{a}$. The resulting set of equations is

$$\Delta d_j = \bar{a}_j^T \bar{\Delta} a, j = 1, \dots, J$$

where $\bar{a}_j^T$ is the product of the normal and the matrix of partial derivatives of $\bar{P}_c$ with respect to each of the model parameters,

$$\bar{a}_j^T = \bar{u}_n^T(j) \left[\frac{\partial \bar{P}_c}{\partial t_x}, \frac{\partial \bar{P}_c}{\partial t_y}, \frac{\partial \bar{P}_c}{\partial t_z}, \frac{\partial \bar{P}_c}{\partial r}, \frac{\partial \bar{P}_c}{\partial p}, \frac{\partial \bar{P}_c}{\partial y} \right].$$

These equations can be weighted to be unit variance and combined into matrix form as

$$\bar{\Delta} d = A \bar{\Delta} a$$

where

$$\bar{\Delta} d = \begin{bmatrix} d_1/\sigma_{\Delta d_1} \\ d_2/\sigma_{\Delta d_2} \\ \vdots \\ d_j/\sigma_{\Delta d_j} \end{bmatrix}$$

and

$$A = \begin{bmatrix} \bar{a}_1^T/\sigma_{\Delta d_1} \\ \bar{a}_2^T/\sigma_{\Delta d_2} \\ \vdots \\ \bar{a}_j^T/\sigma_{\Delta d_j} \end{bmatrix}$$

The weighted least squares solution is computed using the generalized inverse of A and is given by

$$\bar{\alpha} = (A^T A)^{-1} A^T \bar{d} .$$

Given this relationship we can now derive a covariance based on small error approximations.

3.1 DERIVATION OF COVARIANCE MATRIX

The covariance matrix for the estimated parameters of the model is derived from the weighted least squares solution. The covariance of $\bar{\alpha}$ is given by

$$\text{Cov} (\bar{\alpha}) = E \left\{ \left[\bar{\alpha} - E\{\bar{\alpha}\} \right] \left[\bar{\alpha} - E\{\bar{\alpha}\} \right]^T \right\} .$$

Assume that the initial guess for $\bar{\alpha}$ is sufficiently close and that the fitting process has been iterated to a stable solution. Then the linearized weighted least squares solution derived above can be used to compute the covariance by taking $E\{\bar{\alpha}\}$ as the point of linearization. Under these conditions,

$$\left[\bar{\alpha} - E\{\bar{\alpha}\} \right] = \Delta \bar{\alpha} = (A^T A)^{-1} A^T \Delta \bar{d} .$$

$$\begin{aligned} \text{Cov} (\bar{\alpha}) &= E \left\{ \left[\Delta \bar{\alpha} \right] \left[\Delta \bar{\alpha} \right]^T \right\} \\ &= (A^T A)^{-1} A^T E \left\{ \Delta \bar{d} \Delta \bar{d}^T \right\} A (A^T A)^{-1} \\ &= (A^T A)^{-1} , \end{aligned}$$

because the displacements $\Delta \bar{d}$ were constructed to be a unit variance uncorrelated random process.

3.2 IMPLEMENTATION OF COVARIANCE COMPUTATION ALGORITHM

The algorithm for computing the covariance is an enhancement of the model drawing algorithm. After the end-points of a line segment to be drawn are computed, the contribution to the inverse of the covariance matrix is computed. After all line segments are considered, the matrix is inverted to form the covariance.

The contribution for each line segment is the outer product of $\bar{a}_j$ with itself normalized by the length of the line segment. Thus, the incremental contribution to the inverse covariance is given by

$$\lambda_d \sigma_d^2 \text{cov}^{-1}(\bar{a}) = \lambda_d \sigma_d^2 \text{cov}^{-1}(\bar{a}) + \left| \bar{p}_1(j) - \bar{p}_2(j) \right| \bar{a}_j \bar{a}_j^T$$

where $\bar{p}_1(j)$ is the pixel position of the beginning of the line segment,

$\bar{p}_1(j)$ is the pixel position of the beginning of the line segment,

$\bar{p}_2(j)$ is the pixel position of the end of the line segment, and

$\bar{a}_j$ is the projection of the partial derivative matrix onto the normal to the line segment.

The partial derivatives are computed using the chain rule. Thus, the partial derivatives of a position in pixel space is given in terms of camera coordinate partial derivatives by

$$\frac{\partial \bar{p}}{\partial a_i} = \begin{bmatrix} K_1(z^c)^{-2} \left[z^c \frac{\partial x^c}{\partial a_i} - x^c \frac{\partial z^c}{\partial a_i} \right] \\ K_3(z^c)^{-2} \left[z^c \frac{\partial y^c}{\partial a_i} - y^c \frac{\partial z^c}{\partial a_i} \right] \end{bmatrix},$$

where a_i represents any of the model parameters. The partial derivatives of the position in camera coordinates with respect to the camera translations are constants, either zero or unity,

$$1 = \frac{\partial x^C}{\partial t_x} = \frac{\partial y^C}{\partial t_y} = \frac{\partial z^C}{\partial t_z} ,$$

$$0 = \frac{\partial x^C}{\partial t_y} = \frac{\partial y^C}{\partial t_z} = \frac{\partial z^C}{\partial t_x} ,$$

and

$$0 = \frac{\partial x^C}{\partial t_z} = \frac{\partial y^C}{\partial t_x} = \frac{\partial z^C}{\partial t_y} .$$

The partial derivatives of the position in camera coordinates with respect to the model orientation angles are products of rotation matrices with the position vector of a model point,

$$\begin{bmatrix} \frac{\partial x^C}{\partial r} \\ \frac{\partial y^C}{\partial r} \\ \frac{\partial z^C}{\partial r} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ -S_r S_p C_y - C_r S_y & -S_r C_p & S_r S_p S_y - C_r C_y \\ C_r S_p C_y - C_r C_p & C_r C_p & -C_r S_p S_y - S_r C_y \end{bmatrix} \begin{bmatrix} x^m \\ y^m \\ z^m \end{bmatrix} ,$$

$$\begin{bmatrix} \frac{\partial x^C}{\partial p} \\ \frac{\partial y^C}{\partial p} \\ \frac{\partial z^C}{\partial p} \end{bmatrix} = \begin{bmatrix} -S_p C_y & -C_p & S_p S_y \\ C_r C_p C_y & -C_r S_p & -C_r C_p S_y \\ S_p C_p C_y & C_r C_y & -S_r C_p S_y \end{bmatrix} \begin{bmatrix} x^m \\ y^m \\ z^m \end{bmatrix} ,$$

and

$$\begin{bmatrix} \frac{\partial x^c}{\partial y} \\ \frac{\partial y^c}{\partial y} \\ \frac{\partial z^c}{\partial y} \end{bmatrix} = \begin{bmatrix} -C_p S_y & 0 & -C_p C_y \\ -C_r S_p S_y - S_r C_y & 0 & C_r S_p C_y + S_r S_y \\ S_r S_p S_y + C_r C_y & 0 & -S_r S_p C_y - C_r S_y \end{bmatrix} \begin{bmatrix} x^m \\ y^m \\ z^m \end{bmatrix} .$$

These rotation matrices are computed using a variation of the subroutine for the model transformation matrix.