AD

AD-E404 541

Technical Report ARMET-TR-20033

AN ACCURATE METHOD FOR CALCULATING THE SYMMETRY AXIS OF AN OBJECT IN A QUASI-BINARY, HIGH-RESOLUTION DIGITAL IMAGE

Kenneth Hohnecker

March 2024

U.S. ARMY COMBAT CAPABILITIES DEVELOPMENT COMMAND ARMAMENTS CENTER

Munitions Engineering and Technology Center

Picatinny Arsenal, New Jersey

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
March 2024	Final	20190301	20200330

4. TITLE AND SUBTITLE
An Accurate Method for Calculating the Symmetry Axis of an Object in a Quasi-binary, High-resolution Digital Image

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	5e. TASK NUMBER	5f. WORK UNIT NUMBER

6. AUTHOR(S)
Kenneth Hohnecker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army DEVCOM AC, METC
Munitions Systems Directorate
(FCDD-ACM-MI)
Picatinny Arsenal, NJ 07806-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
N\A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
U.S. Army DEVCOM AC, ESIC Knowledge & Process Management Office (FCDD-ACE-K) Picatinny Arsenal, NJ 07806-5000		Technical Report ARMET-TR-20033

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The purpose of this paper is to derive an algorithm for determining the angle (with respect to horizontal) of the axis of symmetry of an object in a gray-scale image. In order for this method to be effective, the image must have sufficient resolution (examples in this paper had on the order of 800,000 pixels). In addition, the image must be quasi-binary, i.e., the background must be relatively homogeneous, and the object of interest must be the largest object in the image.

The motivation for this algorithm is to calculate yaw angles of projectiles during flight whose images are captured and processed by the Automated Small-Arms Photogrammetry (ASAP) software (written in MATLAB) at the U.S. Army Combat Capabilities Development Command (DEVCOM) Armaments Center (AC), Picatinny Arsenal, NJ. This algorithm has several benefits: it is robust (works even with blurry, truncated, or marred images), it is relatively simple to code, and it requires minimal *a priori* knowledge of the shape of the object.

15. SUBJECT TERMS

Automated Small-Arms Photogrammetry (ASAP) High-speed yaw capture Angle of attack
Projectile flight Image recognition

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	c. THIS PAGE		
U	U	U	SAR	33

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER (Include area code)
Kenneth Hohnecker	(973) 724-7633

Standard Form 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

CONTENTS

FIGURES

ACKNOWLEDGMENTS

The author would like to thank Dr. Ryan J. Decker for his invaluable feedback on this paper and for his contributions to the field.

INTRODUCTION

The purpose of this paper is to derive an algorithm for determining the angle (with respect to horizontal) of the axis of symmetry of an object in a gray-scale image. In order for this method to be effective, the image must have sufficient resolution (examples in this paper had on the order of 800,000 pixels). In addition, the image must be quasi-binary, i.e., the background must be relatively homogeneous, and the object of interest must be the largest object in the image. For a more generalized image-analyzing method, see reference 1.

The motivation for this algorithm is to calculate yaw angles of projectiles during flight whose images are captured and processed by the Automated Small-Arms Photogrammetry (ASAP) software (written in MATLAB) at the U.S. Army Combat Capabilities Development Command (DEVCOM) Armaments Center (AC), Picatinny Arsenal, NJ. This algorithm has several benefits: it is robust (works even with blurry, truncated, or marred images), it is relatively simple to code, and it requires minimal *a priori* knowledge of the shape of the object.[1]

ALGORITHM

Convert Gray-scale Image to Black and White

Suppose there is a gray-scale image, Φ, that contains pixels with weights zero (black) to one (white). It is assumed that the object of interest, Ω, in the image has one axis of symmetry, is roughly homogeneous in shade, and is much darker than the background, i.e., the pixel weights of Ω are nearly constant and much less than the background.

The team wanted to estimate a pixel cut-off weight, w_c, below which a pixel is considered part of Ω and above which a pixel is considered part of the background. If Ω is not expected to cover more than τ percentage of Φ, then the pixel weight of the (τ)th percentile pixel weight is a good guess at the representative background pixel weight. This weight can be called w_τ. The minimum pixel weight, w_{min}, of Φ is a representative pixel weight for Ω, since Ω is assumed to be much darker than the background and nearly uniform in shade. w_c will fall somewhere between w_τ and w_{min} and can be approximated as the average of these two weights, as seen in equation 1:

$$w_c = \frac{w_\tau + w_{min}}{2}. \tag{1}$$

Φ is thus converted to a black and white (BW) image using w_c as the threshold value to distinguish between white and black pixels.

[1] The user only needs to know if the axis of symmetry is the minimum or maximum principal axis, whether the background of Φ is darker or lighter than Ω, and the maximum percentage of Φ that Ω will cover. If the background of Φ is darker than Ω, replace w_{min} with w_{max} and replace τ with $100 - \tau$.

Locate Boundary of Object

MATLAB's `bwboundaries` function is used to trace any disjoint objects within Φ. Per the team's assumptions, the boundary with the largest interior area is Ω. Let $\partial\Omega$ be the set of boundary pixels of Ω, and let

$$\partial\vec{\Omega}_j = \begin{bmatrix} x_j \\ y_j \end{bmatrix}, \ 1 \leq j \leq N \tag{2}$$

be the vector containing the x and y coordinates of the jth pixel in $\partial\Omega$.[2]

Find Axis of Symmetry of Object

Estimate Axis of Symmetry Using Moments of Inertia

As an initial approximation to the axis of symmetry, it is first assumed that Ω, as observed in Φ, is indeed perfectly symmetric. Since an axis of symmetry is also a principal axis (ref. 1), the estimation problem is reduced to finding the principal axes of Ω. Furthermore, it will be assumed that the axis of symmetry of Ω is the minimum principal axis (true for virtually every projectile that is stable in flight).[3] Let L be a line that passes through the centroid of Ω. The perpendicular distance, $D_\perp(x, y)$, between L and the point (x, y) is

$$D_\perp(x,y) = \left| \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} c_x \\ c_y \end{bmatrix} \right) \cdot \vec{n} \right|, \tag{3}$$

where c_x and c_y are the x and y coordinates of the centroid of Ω, respectively, and $\vec{n}$ is a unit vector normal to L. The moment of inertia, M, about L is thus

$$M = \iint_\Omega D_\perp^2 \, dx dy. \tag{4}$$

Assume for now that the axis of symmetry is closer to a horizontal line than a vertical line. Since it is also assumed that the axis of symmetry is the minimum principal axis, this implies that $I_{xx} \leq I_{yy}$, where I_{xx} is the moment of inertia about the x-axis and I_{yy} is the moment of inertia about the y-axis, respectively. In this case, L will be represented with the equation

$$y = a(x - c_x) + c_y, \ I_{xx} \leq I_{yy}. \tag{5}$$

---

[2] In this paper, whenever x and y are referenced using subscripts, they are the coordinates of a pixel in $\partial\Omega$. Whenever x and y are referenced without subscripts, they are just generic points in $\mathbb{R}^2$.

[3] If necessary, the algorithm from this point on can be modified if it is assumed that the axis of symmetry is the maximum principal axis.

Therefore[4]

$$\vec{n}(a) = \frac{1}{\sqrt{1+a^2}} \begin{bmatrix} -a \\ 1 \end{bmatrix}. \tag{6}$$

Substituting equations 3 and 6 into equation 4 and expanding yields

$$\begin{aligned} M &= \frac{1}{1+a^2} \iint_{\Omega} \left[a^2(x-c_x)^2 - 2a(x-c_x)(y-c_y) + (y-c_y)^2 \right] dxdy \\ &= \frac{1}{1+a^2} \iint_{\tilde{\Omega}} \left(a^2\tilde{x}^2 - 2a\tilde{x}\tilde{y} + \tilde{y}^2 \right) d\tilde{x}d\tilde{y}, \end{aligned} \tag{7}$$

where the tilde above a variable represents a shifted coordinate system such that the centroid of Ω is at the origin. The principal axes occur when M is at an extremum. The values of a that extremize M (call these a_*) can be found by setting the derivative of M with respect to a to zero and solving for a_*, i.e.,

$$\begin{aligned} 0 &= \frac{dM}{da} \\ &= \frac{\left(1+a_*^2\right) \iint_{\tilde{\Omega}} \left(2a_*\tilde{x}^2 - 2\tilde{x}\tilde{y} \right) d\tilde{x}d\tilde{y} - 2a_* \iint_{\tilde{\Omega}} \left(a_*^2\tilde{x}^2 - 2a_*\tilde{x}\tilde{y} + \tilde{y}^2 \right) d\tilde{x}d\tilde{y}}{\left(1+a_*^2\right)^2} \\ \implies 0 &= a_*^2 I_{xy} + a_* \left(I_{yy} - I_{xx} \right) - I_{xy} \\ \implies a_* &= \frac{-\left(I_{yy} - I_{xx} \right) \pm \sqrt{\left(I_{yy} - I_{xx} \right)^2 + 4I_{xy}^2}}{2I_{xy}}, \end{aligned} \tag{8}$$

where I_{xy} is the product moment of inertia of Ω. In order for the assumptions (the axis of symmetry is closer to a horizontal line and $I_{xx} \leq I_{yy}$) to be consistent, the positive root in equation 8 must be chosen. Therefore,

$$a_* = \begin{cases} \frac{2T_a}{1+\sqrt{1+4T_a^2}}, & I_{xy} \leq I_{yy} - I_{xx}, \\ \frac{2}{T_a+\sqrt{T_a^2+4}}, & I_{xy} > I_{yy} - I_{xx}, \end{cases} \tag{9}$$

Where

$$T_a = \begin{cases} \frac{I_{xy}}{I_{yy}-I_{xx}}, & I_{xy} \leq I_{yy} - I_{xx}, \\ \frac{I_{yy}-I_{xx}}{I_{xy}}, & I_{xy} > I_{yy} - I_{xx}. \end{cases} \tag{10}$$

[4] The convention that $\vec{n}(a)$ is always pointing in the positive y direction has been chosen.

A similar derivation can be made if it is assumed that the axis of symmetry is closer to a vertical line than a horizontal line. In this case, the equation for L can be expressed as

$$x = \ell(y - c_y) + c_x, \; I_{xx} > I_{yy}. \tag{11}$$

The unit normal vector to L is then[5]

$$\vec{n}(\ell) = \frac{1}{\sqrt{1+\ell^2}} \begin{bmatrix} 1 \\ -\ell \end{bmatrix} \tag{12}$$

and the value of ℓ that extremizes the moment about L is

$$\ell_* = \begin{cases} \frac{2T_\ell}{1+\sqrt{1+4T_\ell^2}}, & I_{xy} \leq I_{xx} - I_{yy}, \\ \frac{2}{T_\ell+\sqrt{T_\ell^2+4}}, & I_{xy} > I_{xx} - I_{yy}, \end{cases} \tag{13}$$

Where

$$T_\ell = \begin{cases} \frac{I_{xy}}{I_{xx}-I_{yy}}, & I_{xy} \leq I_{xx} - I_{yy}, \\ \frac{I_{xx}-I_{yy}}{I_{xy}}, & I_{xy} > I_{xx} - I_{yy}. \end{cases} \tag{14}$$

Thus, the axis of symmetry can be approximated as using equations 5 or 11 and substituting in a_* or ℓ_*, respectively.

Refine Axis of Symmetry Using Maximum Reflective Symmetry

Suppose there is an arbitrary object Ω with one axis of symmetry S. Now suppose the line G passes through the centroid of Ω and reflect Ω about G; call the reflected object Ω' (see fig. 1). Observe that the area of $\Omega \cap \Omega'$ is equal to the area of Ω when $G = S$. Thus, the area of $\Omega \cap \Omega'$ is maximized when $G = S$, since the area of $\Omega \cap \Omega'$ cannot be greater than the area of Ω. Based on this observation, the area of $\Omega \cap \Omega'$ (call this area Λ) will be used as a measure of symmetry. The advantage to this metric is that any artificial asymmetries of Ω due to the image's truncated field-of-view and/or extraneous objects are ignored (the intersection of an image and its reflection about an axis is always symmetric) and the only goal is to find the axis that produces the largest symmetric image possible. The reflection, $\overrightarrow{P'}$, of an arbitrary point $\vec{P}$ about the line G is given by

[5] The convention that $\vec{n}$(a) is always pointing in the positive x direction has been chosen.

$$\vec{P}' = \vec{P} - 2\left[\left(\vec{P} - \vec{R}\right) \cdot \vec{n}_G\right] \vec{n}_G, \tag{15}$$

Where

$$\vec{R} = \begin{bmatrix} r_x \\ r_y \end{bmatrix} \tag{16}$$

is some point along G and $\vec{n}_G$ is a unit vector normal to G.

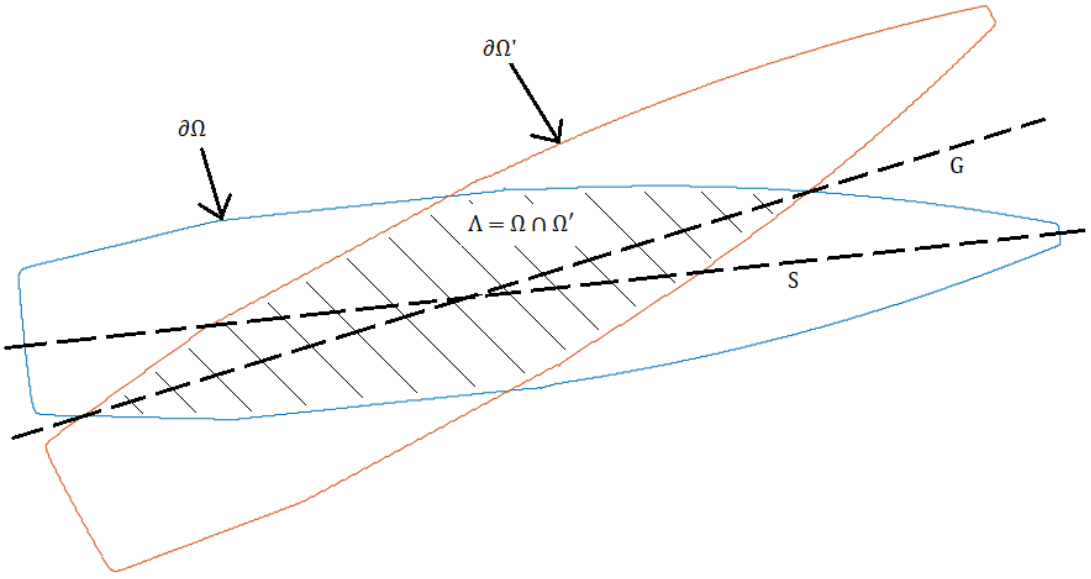

Figure 1
Diagram for maximum reflective symmetry method

MATLAB `polyshape` objects and functions are used to find the intersection of the projectile and its reflection. The `fminunc` function is used to minimize $-\Lambda$ (thus maximizing Λ), with initial guesses for $\vec{R}$ and the axis of symmetry provided by the centroid location and axis slope calculated using the method in the Estimate Axis of Symmetry Using Moments of Inertia section, respectively. If the initial guess results in using equation 5 to express the axis of symmetry, then r_x will be fixed and r_y varied during the maximization of Λ.[6] If the initial guess results in using equation 11 to express the axis of symmetry, then r_y will be fixed and r_x varied during the maximization of Λ.

[6] If r_y were fixed and r_x varied, the scenario of the axis of symmetry being a horizontal line might be encountered, in which case no amount of varying r_x would be able to translate G relative to the initial guess. The converse argument holds if the axis of symmetry is closer to a vertical line.

Calculation of Centroid Coordinates and Moments of Inertia

The centroid coordinates and moments of inertia can be calculated as follows:

$$c_x = \frac{1}{A} \iint_{\Omega} x dx dy, \tag{17}$$

$$c_y = \frac{1}{A} \iint_{\Omega} y dx dy, \tag{18}$$

$$I_{xx} = \iint_{\tilde{\Omega}} \tilde{y}^2 d\tilde{x} d\tilde{y}, \tag{19}$$

$$I_{yy} = \iint_{\tilde{\Omega}} \tilde{x}^2 d\tilde{x} d\tilde{y}, \tag{20}$$

And

$$I_{xy} = \iint_{\tilde{\Omega}} \tilde{x}\tilde{y} d\tilde{x} d\tilde{y}, \tag{21}$$

Where

$$A = \iint_{\Omega} dx dy \tag{22}$$

is the area of Ω. Using Green's Theorem, equation 22 becomes

$$A = \frac{1}{2} \int_{\partial\Omega} (x dy - y dx) . \tag{23}$$

Due to pixelation, $\partial\Omega$ only consists of horizontal and vertical lines. Thus, the integral in equation 23 can be separated into two parts

$$A = \left[\frac{1}{2}\int_{\partial\Omega}(xdy - ydx)\right]_{vert} + \left[\frac{1}{2}\int_{\partial\Omega}(xdy - ydx)\right]_{horz}. \tag{24}$$

For the vertical portions of $\partial\Omega$, dx = 0 and x is constant; conversely, dy = 0 and y is constant for horizontal lines. Therefore, equation 24 becomes

$$\begin{aligned} A &= \frac{1}{2}\sum_{\substack{j=1\\ x_j=x_{j+1}}}^{N} x_j \int_{y_j}^{y_{j+1}} dy - \frac{1}{2}\sum_{\substack{j=1\\ y_j=y_{j+1}}}^{N} y_j \int_{x_j}^{x_{j+1}} dx \\ &= \frac{1}{2}\sum_{\substack{j=1\\ x_j=x_{j+1}}}^{N} x_j\left(y_{j+1} - y_j\right) - \frac{1}{2}\sum_{\substack{j=1\\ y_j=y_{j+1}}}^{N} y_j\left(x_{j+1} - x_j\right) \\ &= \frac{1}{2}\sum_{\substack{j=1\\ x_j=x_{j+1}}}^{N} \left[x_j\left(y_{j+1} - y_j\right) - y_j\left(x_{j+1} - x_j\right)\right] + \frac{1}{2}\sum_{\substack{j=1\\ y_j=y_{j+1}}}^{N} \left[x_j\left(y_{j+1} - y_j\right) - y_j\left(x_{j+1} - x_j\right)\right] \end{aligned} \tag{25}$$

Since the vertical and horizontal sections of $\partial\Omega$ are mutually exclusive, the two sums in equation 25 can be combined to get

$$A = \frac{1}{2}\sum_{j=1}^{N}\left(x_j\delta_{y,j} - y_j\delta_{x,j}\right), \tag{26}$$

where[7]

$$\delta_{x,j} = x_{j+1} - x_j \tag{27}$$

And

$$\delta_{y,j} = y_{j+1} - y_j. \tag{28}$$

[7] $\partial\Omega$ is assumed to be a closed loop; therefore, $x_0 = x_N$ and $x_{N+1} = x_1$. Similar equalities hold for all indexed variables on $\partial\Omega$.

Using a similar procedure as for A, equations 17 through 21 can be calculated as follows:

$$c_x = \frac{1}{2A}\sum_{j=1}^{N} x_j^2 \delta_{y,j} \tag{29}$$

$$c_y = -\frac{1}{2A}\sum_{j=1}^{N} y_j^2 \delta_{x,j} \tag{30}$$

$$I_{xx} = -\frac{1}{3}\sum_{j=1}^{N} \tilde{y}_j^3 \delta_{x,j} \tag{31}$$

$$I_{yy} = \frac{1}{3}\sum_{j=1}^{N} \tilde{x}_j^3 \delta_{y,j} \tag{32}$$

$$I_{xy} = \frac{1}{8}\sum_{j=1}^{N} \left(\tilde{x}_j^2 \tilde{y}_{j+1}^2 - \tilde{y}_j^2 \tilde{x}_{j+1}^2\right). \tag{33}$$

EXPERIMENTAL RESULTS

The previous algorithm was implemented using the custom MATLAB function `angleFinder()` in the appendix. The angle of the axis of symmetry in figure 2 was manually measured by taking the midpoint of the meplat (nose) and boattail (rear) and calculating the angle with respect to the horizontal of the line formed by these two points. Figure 2 was then rotated, truncated, blurred, and marred in various ways (see figs. 3 through 12) to test the robustness of the algorithm. The angles calculated by `angleFinder()`, using $\tau = 50$, were then compared to the manually measured results (see table 1).[8]

[8] Note that the actual MATLAB code in the appendix returns the negative of the angle, since MATLAB flips the image when it's imported.

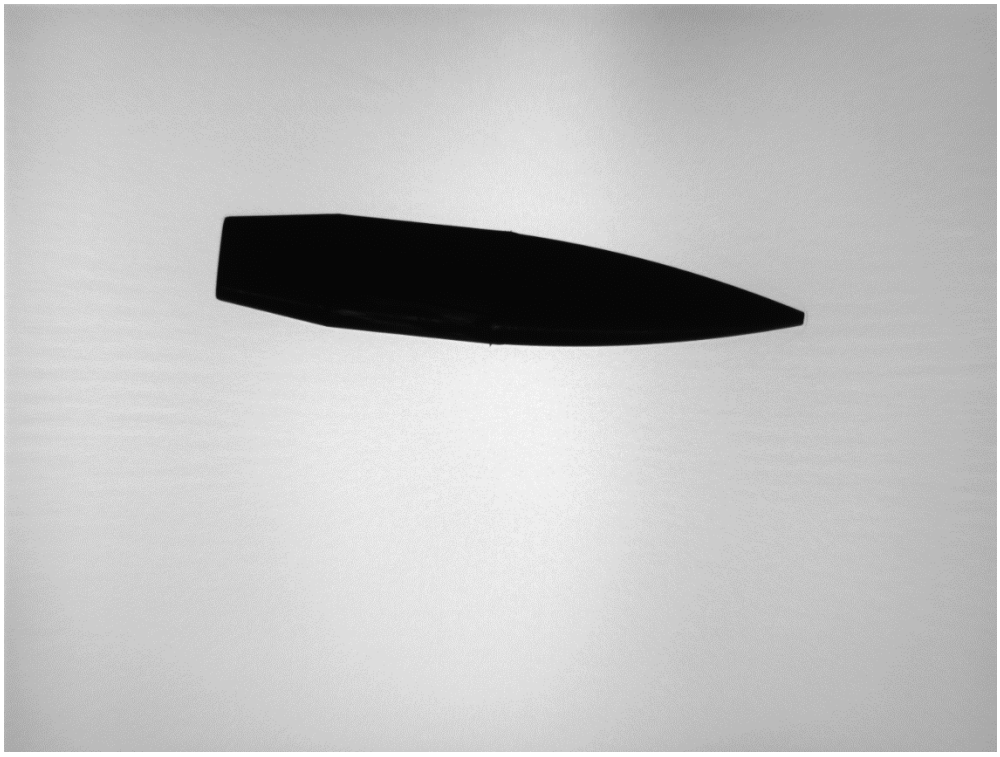

Figure 2
Original image

Figure 3
Figure 2 shifted right

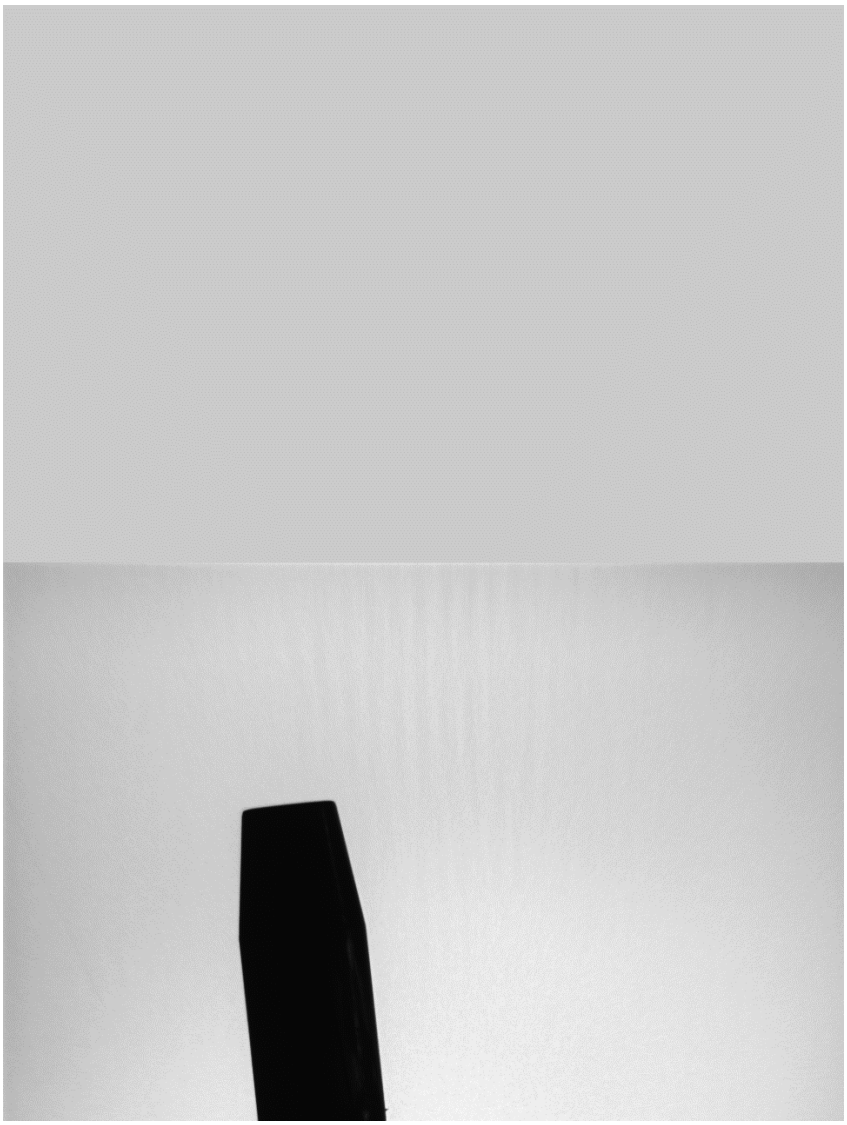

Figure 4
Figure 2 shifted right and rotated

Figure 5
Figure 2 shifted left and blurred

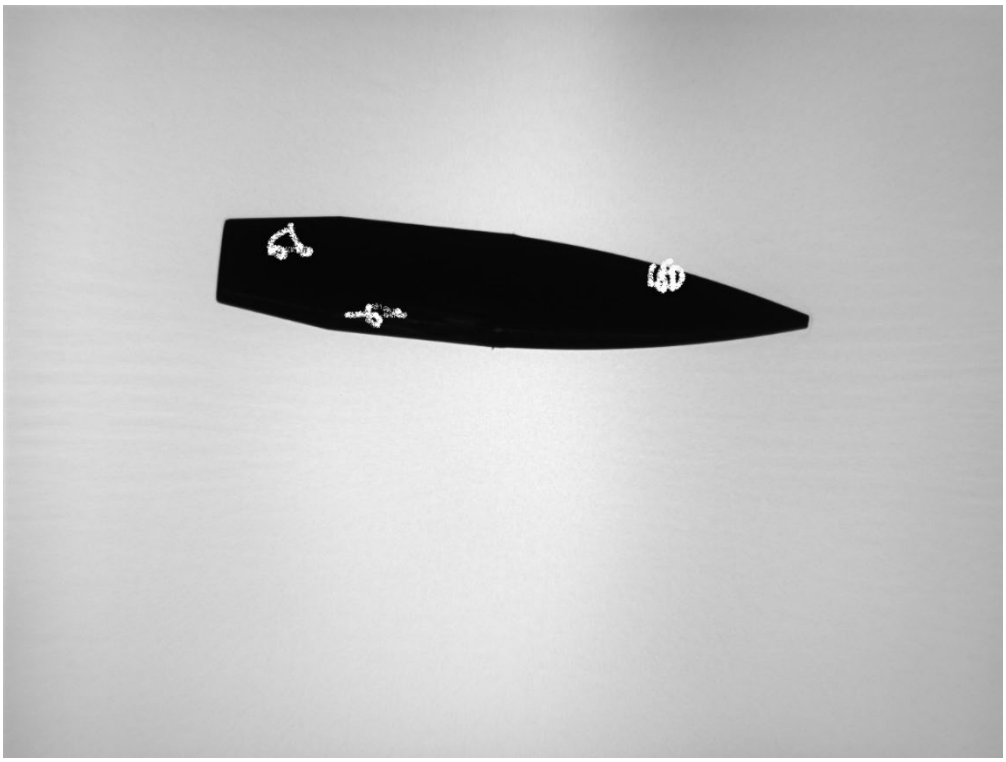

Figure 6
Figure 2 with random anomalies

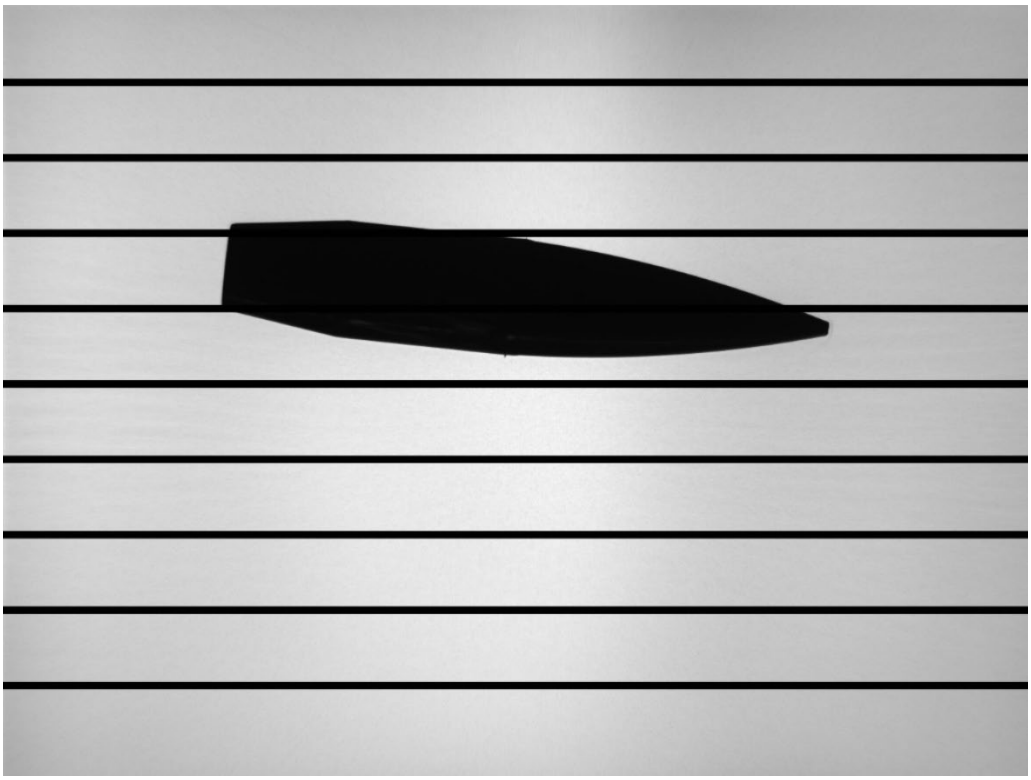

Figure 7
Figure 2 with horizontal line anomalies

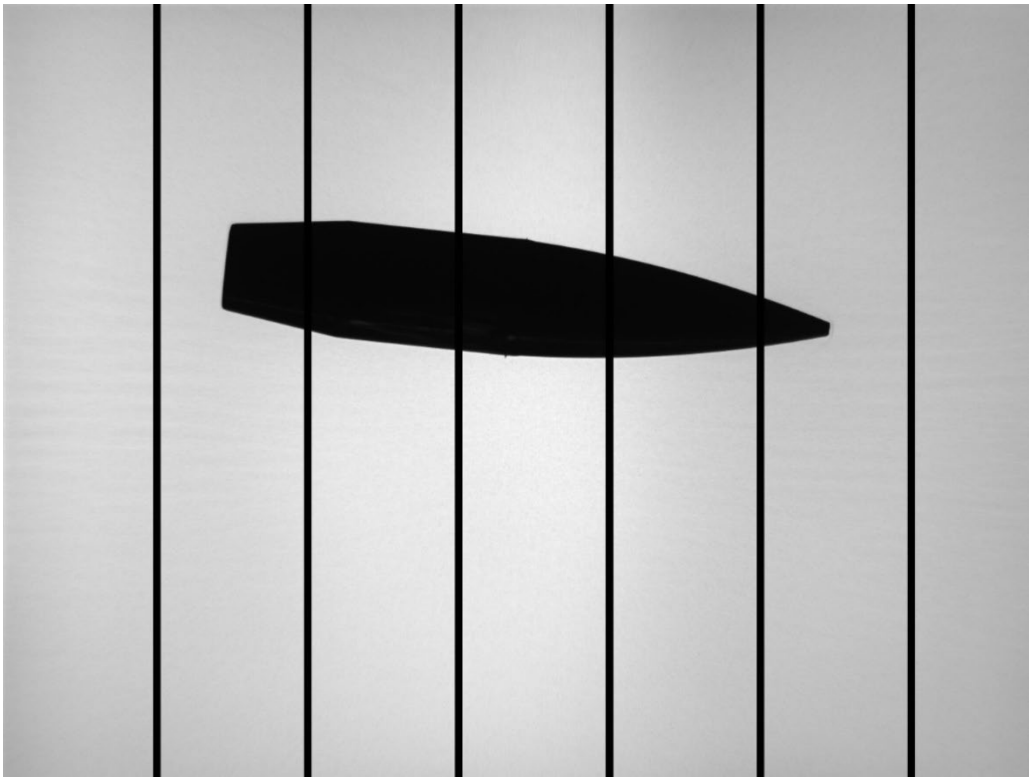

Figure 8
Figure 2 with vertical line anomalies

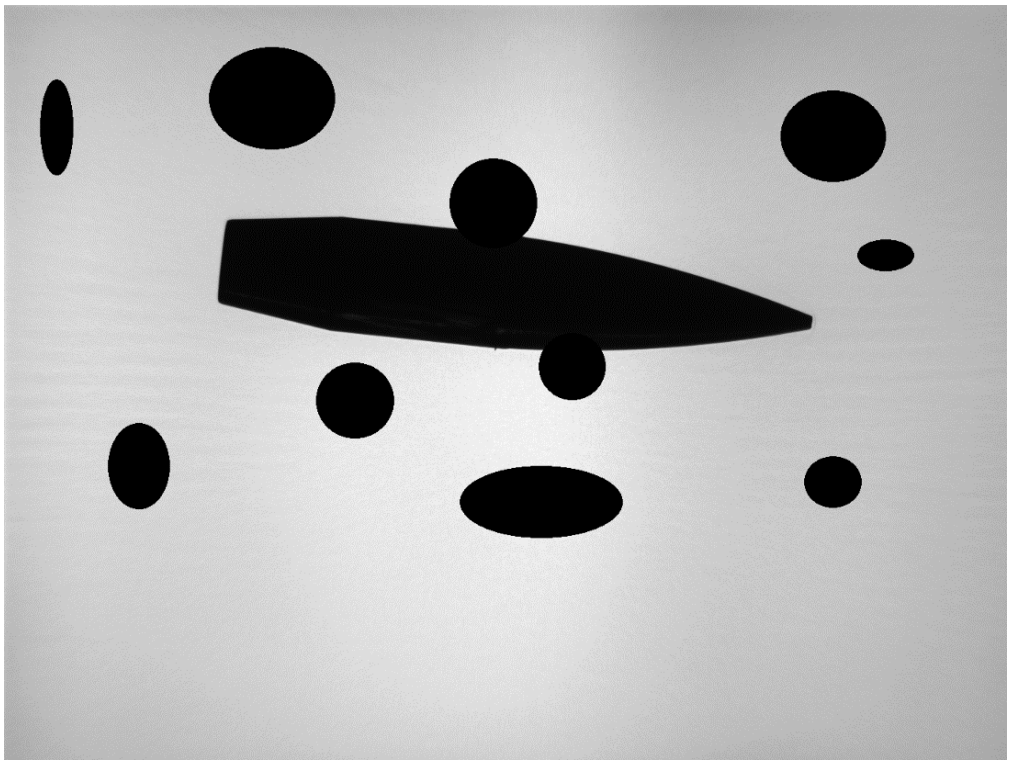

Figure 9
Figure 2 with elliptical anomalies

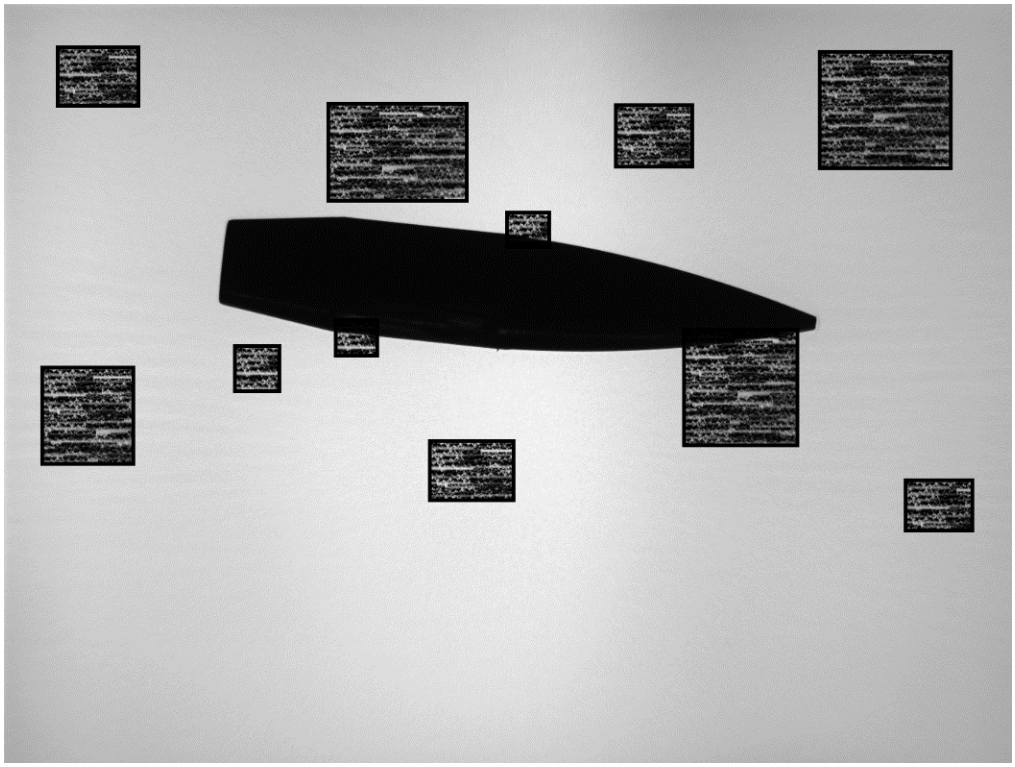

Figure 10
Figure 2 with rectangular anomalies

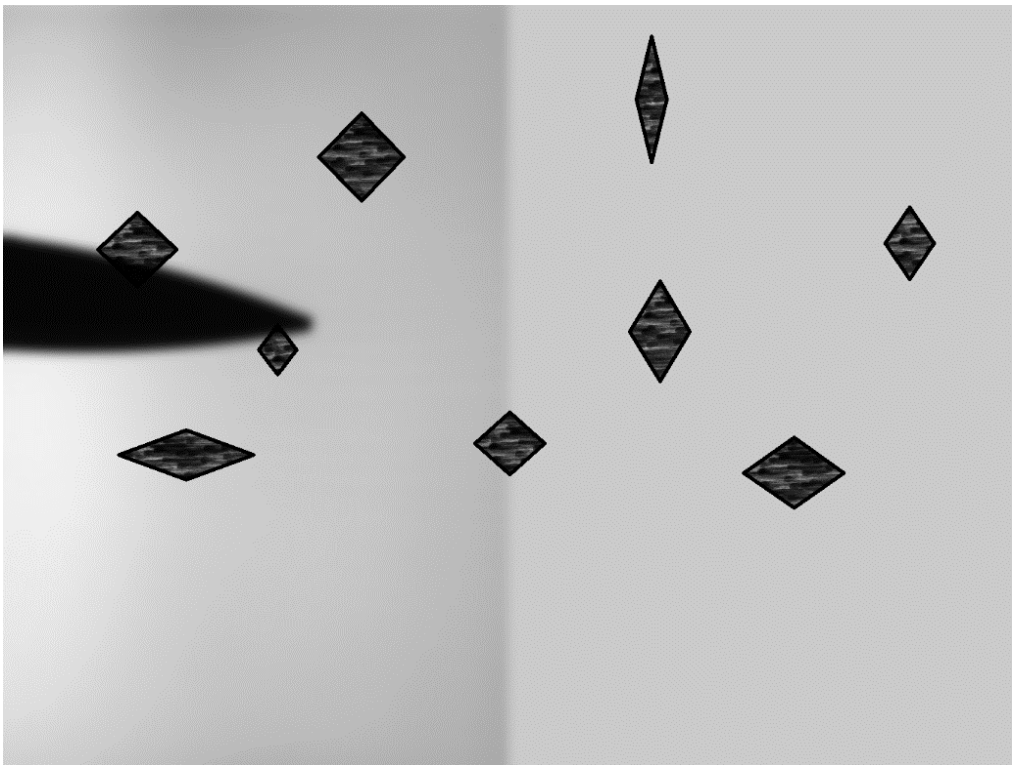

Figure 11
Figure 2 shifted left, blurred, and with diamond-shaped anomalies

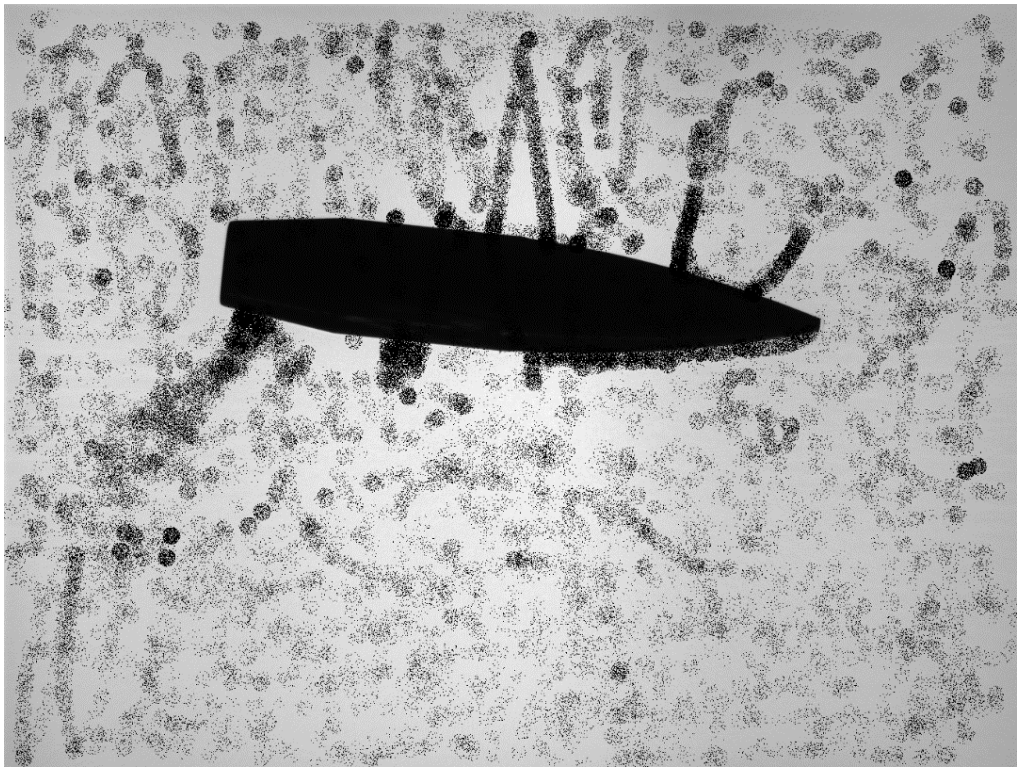

Figure 12
Figure 2 with significant noise

Table 1
Manual measurements versus algorithm measurements

Figure	Manually measured (deg)	angleFinder() (deg)	Error (deg)
2	-5.907	-5.916	9E-3
3	-5.907	-6.151	2.4E-1
4	-84.093	-83.873	2.2E-1
5	-5.907	-5.810	9.7E-2
6	-5.907	-5.927	2.0E-2
7	-5.907	-5.923	1.6E-2
8	-5.907	-5.913	6E-3
9	-5.907	-5.929	2.2E-2
10	-5.907	-6.089	1.8E-1
11	-5.907	-5.847	6.0E-2
12	-5.907	-6.491	5.8E-1

CONCLUSIONS

An algorithm was derived to calculate the angle of the axis of symmetry of an object in an image. This algorithm requires minimal *a priori* knowledge of the shape of the projectile, and it is accurate even if the image is blurry, truncated, or marred. Various runs were performed in the Experimental Results section of the report; the error when the object was fully contained in the image, in focus, with no blemishes was on the order of 0.01 deg, averaged on the order of 0.2 deg, and, in the most extreme case, was on the order of 0.6 deg.

REFERENCES

1. Decker, R. J., Kölsch, M. N., and Yakimenko, O. A., “An Automated Method for Computer Vision Analysis of Cannon-Launched Projectile Video,” Journal of Testing and Evaluation, Vol. 42, No. 5, September 2014.

2. Hibbeler, R. C., Engineering Mechanics: Dynamics, Twelfth Edition, Pearson Education, 2010.

APPENDIX
MATLAB CODE

```
function [theta, c_x, c_y, x, y] = angleFinder(Phi, maxOmegaPerc)
%% inputs
% Phi = grayscale image

% maxOmegaPerc = the max percentage of the image the object of interest is
% expected to cover; the smaller this number is, the more background noise
% is acceptable

%% image processing

w_min = min(Phi(:)); % minimum pixel value
% (tau)th percentile pixel weight
w_tau = prctile(Phi(:), maxOmegaPerc);

% threshold for cuttoff between dark and light is halfway between the
% darkest pixel and the (tau)th percentile pixel (which is
% representative of the background pixel value)
w_c = double(w_tau + w_min)/2;

% convert image to black and white
Phi = imbinarize(Phi, w_c/255);

[h, w] = size(Phi); % width and height of image in pixels

% add perimeter of white pixels to image, prevents border interference when
% using bwboundaries
Phi = [true(h, 1), Phi, true(h, 1)];
Phi = [true(1, w+2); Phi; true(1, w+2)];

%% find projectile boundary

% find possible boundaries of Omega
delOmega = bwboundaries(Phi);
numBoundaries = length(delOmega); % number of possible boundaries

A = 0; % initialize area of object of interest
% find object of interest
for j = 2:numBoundaries % skip the first boundary as it is just the border
    x = delOmega{j}(:, 2); % x coordinates of current boundary
    y = delOmega{j}(:, 1); % y coordinates of current boundary
    N = length(x); % number of current boundary points
```

```
    % difference between adjacent x coordinates on boundary
    deltax = zeros(N, 1);
    deltax(1:end-1) = x(2:end) - x(1:end-1);
    deltax(end) = x(1) - x(end);

    % difference between adjacent y coordinates on boundary
    deltay = zeros(N, 1);
    deltay(1:end-1) = y(2:end) - y(1:end-1);
    deltay(end) = y(1) - y(end);

    % area of the interior of current boundary
    Atemp = dot(x, deltay) - dot(y, deltax)/2;

    % if the current object is larger than the previous largest object,
    % replace
    if Atemp > A
        A = Atemp;
        maxA_idx = j;
    end
end

% boundary points of largest object
delOmega = delOmega{maxA_idx};

% store x,y coordinates of boundary points
x = delOmega(:, 2);
y = delOmega(:, 1);

%% Estimate slope of axis of symmetry of projectile and centroid location

% intermediate variables
N = length(x); % number of boundary points
% difference between adjacent x coordinates on boundary
deltax = zeros(N, 1);
deltax(1:end-1) = x(2:end) - x(1:end-1);
deltax(end) = x(1) - x(end);
% difference between adjacent y coordinates on boundary
deltay = zeros(N, 1);
deltay(1:end-1) = y(2:end) - y(1:end-1);
deltay(end) = y(1) - y(end);

% coordinates for centroid
c_x = abs(dot(x.^2, deltay))/A/2;
c_y = abs(dot(y.^2, deltax))/A/2;
```

```
% shifted coordinates so that centroid is at origin
x_tilde = x - c_x;
y_tilde = y - c_y;

% moments of inertia
Ixx = abs(dot(y_tilde.^3, deltax))/3;
Iyy = abs(dot(x_tilde.^3, deltay))/3;
Ixy = abs((x_tilde(end)*y_tilde(1))^2 - (y_tilde(end)*x_tilde(1))^2 ...
    + sum((x_tilde(1:end-1).*y_tilde(2:end)).^2 ...
    - (y_tilde(1:end-1).*x_tilde(2:end)).^2))/8;

% calculate slope of axis of symmetry; if axis of symetry is closer to a
% horizontal line, represent line as y = a(x - c1) + c2; if axis of symetry
% is closer to a vertical line, represent line as x = 1(y - c2) + c1
if Ixx <= Iyy % axis of symmetry is closer to a horizontal line
    horzFlag = true;
    deltaI = Iyy - Ixx;
    if Ixy <= deltaI
        Ta = Ixy/deltaI;
        a_star = 2*Ta/(1 + sqrt(1 + 4*Ta^2));
    else
        Ta = deltaI/Ixy;
        a_star = 2/(Ta + sqrt(Ta^2 + 4));
    end
else % axis of symmetry is closer to a vertical line
    horzFlag = false;
    deltaI = Ixx - Iyy;
    if Ixy <= deltaI
        Tl = Ixy/deltaI;
        l_star = 2*Tl/(1 + sqrt(1 + 4*Tl^2));
    else
        Tl = deltaI/Ixy;
        l_star = 2/(Tl + sqrt(Tl^2 + 4));
    end
end

%% Refine axis of symmetry and centroid location using reflective symmetry

% Create a polygon of the boundary points
delOmega = polyshape(x, y);
% initial guess at x coordinate of point that lies on axis of symmetry
c1 = c_x;
% initial guess at y coordinate of point that lies on axis of symmetry
c2 = c_y;

% find angle of axis of symmetry and the point along that line that lies
```

```
% closest to the previously calculated CG, use this as the new CG
if horzFlag % use a_star and vary c2
    min_values = fminunc(@(var) -Lambda(var, c1, delOmega, horzFlag),...
        [a_star, c2]);
    a_star = min_values(1);
    c2 = min_values(2);
    tempCG = [c1; c2] + ([c_x - c1, c_y - c2]*[1; a_star])/sqrt(1 + a_star^2);
    theta = atan(a_star);
else % use l_star and vary c1
    min_values = fminunc(@(var) -Lambda(var, c2, delOmega, horzFlag),...
        [l_star, c1]);
    l_star = min_values(1);
    c1 = min_values(2);
    tempCG = [c1; c2] + ([c_x - c1, c_y - c2]*[l_star; 1])/sqrt(1 + l_star^2);
    theta = acot(l_star);
end

% new CG location
c_x = tempCG(1);
c_y = tempCG(2);


% Given a polygon shape 'delOmega', calculate the area of the intersection
% of 'delOmega' with its reflection about the axis 'G' with slope 'a_star'
% or 'l_star' that passes through the point (r_x, r_y).
function L = Lambda(var, c, delOmega, horzFlag)

if horzFlag % values to use if axis is closer to a horizontal line
    r_y = var(2);
    r_x = c;
    a_star = var(1);
    nvec = [-a_star; 1]/sqrt(1 + a_star^2);
else % values to use if axis is closer to a vertical line
    r_x = var(2);
    r_y = c;
    l_star = var(1);
    nvec = [1; -l_star]/sqrt(1 + l_star^2);
end

M = length(delOmega.Vertices(:, 1)); % number of unique vertices on delOmega

% Reflect vertices of 'delOmega' about the axis 'G'
delOmega_prime = delOmega.Vertices - 2*((delOmega.Vertices...
    - repmat([r_x, r_y], M, 1))*nvec).*nvec';

% Create a polyshape structure 'delOmega_prime' from the reflected vertices
```

```
% of 'delOmega'
delOmega_prime = polyshape(delOmega_prime(:, 1), delOmega_prime(:, 2));

% Find intersection of 'delOmega' and 'delOmega_prime'
polyint = intersect(delOmega, delOmega_prime);

% Compute area of intersection region
L = area(polyint);
```

DISTRIBUTION LIST

U.S. Army DEVCOM AC
ATTN: FCDD-ACE-K
Picatinny Arsenal, NJ 07806-5000

Defense Technical Information Center (DTIC)
ATTN: Accessions Division
8725 John J. Kingman Road, Ste 0944
Fort Belvoir, VA 22060-6218

GIDEP Operations Center
P.O. Box 8000
Corona, CA 91718-8000
gidep@gidep.org

REVIEW AND APPROVAL OF ARDEC TECHNICAL REPORTS

An Accurate Method for Calculating the Symmetry Axis of an Object in a Quasi-Binary, High-Resolution Digital Image
Title

_____
Date received by LCSD

___Ken Hohnecker_____
Author/Project Engineer

_____
Report number (to be assigned by LCSD)

_x7633_____ 65N_____ ____FCDD-ACM-MI_____
Extension Building Author's/Project Engineers Office (Division, Laboratory, Symbol)

PART 1. **Must be signed before the report can be edited.**

a. The draft copy of this report has been reviewed for technical accuracy and is approved for editing.

b. Use Distribution Statement A_X___, B____, C____, D____, E____, F____ or X ____ for the reason checked on the continuation of this form.

1. If Statement A is selected, the report will be released to the National Technical Information Service (NTIS) for sale to the general public. Only unclassified reports whose distribution is not limited or controlled in any way are released to NTIS.

2. If Statement B, C, D, E, F, or X is selected, the report will be released to the Defense Technical Information Center (DTIC) which will limit distribution according to the conditions indicated in the statement.

c. The distribution list for this report has been reviewed for accuracy and completeness.

Signed – Mark Nicolich 3/30/2020
Division Chief (Date)

PART 2. To be signed either when draft report is submitted or after review of reproduction copy.

This report is approved for publication.

Signed – Kip Hess 3/15/2024
Division Chief (Date)

SMCAR Form 49, 20 Dec 06 supersedes SMCAR Form 49, 1 Nov 94.