Vision Algorithm for the Solar Aspect System of the HEROES Mission

Alex Cramer
NASA Goddard Space Flight Center
Greenbelt, Maryland 20771
alexander.cramer@nasa.gov

***Abstract*—This work covers the design and test of a machine vision algorithm for generating high-accuracy pitch and yaw pointing solutions relative to the sun for the High Energy Replicated Optics to Explore the Sun (HEROES) mission. It describes how images were constructed by focusing an image of the sun onto a plate printed with a pattern of small fiducial markers. Images of this plate were processed in real time to determine relative position of the balloon payload to the sun. The algorithm is broken into four problems: circle detection, fiducial detection, fiducial identification, and image registration. Circle detection is handled by an "Average Intersection" method, fiducial detection by a matched filter approach, identification with an ad-hoc method based on the spacing between fiducials, and image registration with a simple least squares fit. Performance is verified on a combination of artificially generated images, test data recorded on the ground, and images from the 2013 flight.**

TABLE OF CONTENTS

1. INTRODUCTION

The problem of generating solar aspect is common to many missions. Knowledge of aspect relative to the sun is especially important for solar observatories, but it is also commonly used by other spacecraft as part of a suite of sensors for determining attitude. Additionally, any system employing solar panels can also make use of this knowledge to optimize harvested power. In the case of the High Energy Replicated Optics to Explore the Sun (HEROES) mission, solar aspect was determined by processing images of a carefully constructed scene onboard the balloon in real time. The approach taken for this Pitch and Yaw Aspect System is different from that taken for other similar systems because it achieved fine solar pointing with an accuracy of 20 arcseconds using relatively inexpensive electronics. In contrast, many similar systems require the use of carefully tuned photo-detectors, FPGAs and other purpose-built electronic hardware. In the case of the HEROES sun sensor, an off-the-shelf camera and computer were used.

This paper is organized as follows. Section 2 provides background on the PYAS optical and mechanical elements, the scene observed by the PYAS, and assumptions made about that scene. Section 3 covers existing work. It addresses both existing sun-trackers, and existing algorithms that could be applied to this specific computer vision problem. Section 4 outlines the details of the algorithm used by the PYAS. Section 5 covers performance of the algorithm. This includes performance on synthetic test data and pre-flight test data. This section also provides a brief summary of the results from the 2013 flight of the HEROES payload. Finally, Section 6 provides a summary of algorithm performance, some lessons learned, and recommended changes for any future PYAS system based on this same design.

2. BACKGROUND

This section covers some background for the HEROES mission and the place of the PYAS in that mission, specifically performance requirements levied on the PYAS. The relevant details of the PYAS optical, mechanical and electrical hardware are given, as well as a discussion of how these come together to form the PYAS scene. There will also be a list of simplifying assumptions made in the algorithm design. These assumptions are justified based on the PYAS hardware, and are important for keeping the PYAS processing problem tractable. The final PYAS algorithm was shaped by these technical details, simplifying assumptions, and mission requirements.

PYAS System Requirements

The PYAS was expected to provide pitch and yaw offsets from a target anywhere on the sun with a accuracy of 20 arcseconds to the gondola control system (CTL), and to do so with a cadence of 1 Hz. Each frame also needed to be stored for post-processing on the ground. These solutions had to be generated any time the sun was within the field of view (FOV) of the PYAS. The HEROES CTL was capable of pointing with an accuracy of $\pm 1^\circ$ in pitch and yaw on coarse sensing alone, at which point it depended on fine sensing to stabilize on a target. For solar pointing, the target could lie anywhere on the sun. Ensuring the sun was always fully on the screen after coarse pointing, regardless of where the target was placed on the sun, would require at least a 3° FOV. The PYAS was only able to achieve a 2.8° FOV, but because of the narrow FOV of the HEROES telescope, only a coarse solution was necessary at the edges of the FOV. Solutions at the edges only had to be accurate enough to tell the CTL roughly where to slew to bring the sun fully onto the screen. A summary of the relevant requirements on the SAS are shown in Table 1.

PYAS Hardware and Scene

A simplified diagram of the PYAS is given in Figure 2. A CCD camera observes plate onto which the sun is projected. The optics for handling this projection consist of an IR filter, band-pass filter, and a plano-convex lens with 3m focal length. This plate itself was covered in a printed pattern of

Table 1. **PYAS Requirements**

Requirement	
Cadence	1 Hz
Accuracy	$<$ 20 arcseconds
Field of View	2.8° (fine)
	$>$ 3.3° (coarse)

cross-shaped fiducial markers as shown in Figure 2. The hardware is described in detail in **??**. The identity of each fiducial mark is encoded in the distance between it and adjacent markers, so that a minimum of 3 adjacent non-collinear fiducials are required for identification. The entire assembly was then surrounded with baffles to cut down on stray light, ensuring the focused solar image was the only source of illumination on the fiducial plate. Taking advantage of precise knowledge of the locations of each fiducial marker relative to the HEROES scientific payload, the hope was to determine the payloads orientation relative to the sun by locating the projected image of the sun relative to the fiducials.

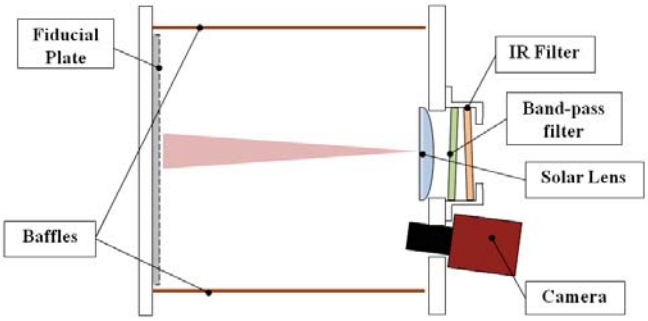

Figure 1. Rough diagram of the PYAS

The HEROES electronics of immediate interest to the PYAS would be the SAS computer stack and the camera. The two communicated through a gigabit ethernet connection. A summary of the relevant specs is given in Table 2. The SAS computer performed all PYAS image processing, as well as handling image storage, command, and telemetry functions for the SAS subsystem. Doing all this on a single 1.6 Ghz core meant that the PYAS algorithm had limited processing power available. As for the camera, the detector size of 966 pixels in the short axis, combined with 2.8° FOV, gave each image pixel an approximate width of 10.6 arcseconds. This meant that to do better than 20 arcseconds accuracy the algorithm would need to resolve to the sub-pixel level.

An example of the images seen by the PYAS is shown in Figure 2. Images are rectangular with a small circular region illuminated by the sun. This solar projection falls into a larger circular area defined by the fiducial screen. A diagram of the entire screen is shown in Figure 2. The screen is sized to span nearly the entire image along the short axis. Within the solar projection it is possible to see several cross-shaped fiducial markers. Once the visible fiducials in a frame are located and identified, they can be used to perform registration between the image plane and the gondola coordinate system. Location and orientation of the fiducial plate and each of its fiducial markers was measured relative to the HEROES x-ray telescope. This calibration data bridged the gap from knowledge of the suns location on the fiducial screen to knowledge of the HEROES telescope's offset from target.

Table 2. **PYAS Electronics**

Camera	
Camera Model	Imperx IGV-B1310
Sensor Size	1296x966 pixels
Bit Depth	12 bit monochrome
Max Frame rate	26 FPS
Computer	
SBC Model	Cool RoadRunner-945GSE
Processor	Intel Atom N270
	Single-core, 1.6 GHz
Storage	256 GB SSD (x2)
Write Speed	$<$260 MB/s

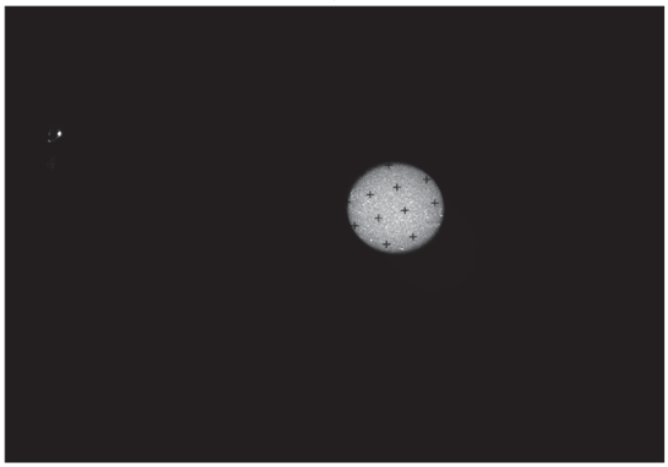

Figure 2. Sample of a raw frame taken by the PYAS camera

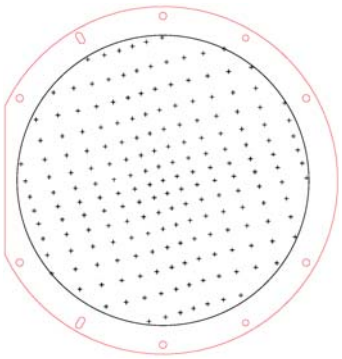

Figure 3. Layout of the PYAS fiducial pattern

PYAS Assumptions

The setup described above gives a baseline for what the PYAS image processing problem might have to handle. There are scenarios not described above, however, which could severely alter the scene observed by the PYAS camera. Accounting for every possible anomaly would very quickly push the algorithm's complexity to a point where it had no chance of meeting the cadence requirement. Instead some basic assumptions were made about how PYAS frames might vary during flight, with the goal of keeping the PYAS problem

Table 3. Algorithm Assumptions

	Assumption
1	There will be only one large bright object present in each frame
2	Optics and camera are approximately parallel to the fiducial plate
2a	Solar image is circular rather than elliptical
2b	Projective affects from the camera orientation can be approximated with a similarity transform
2c	Clocking of fiducial plate relative to the camera is negligible
2d	Change in distance and clocking between fiducial plate and camera will be small
3	The projected solar image will not be under or overexposed

tractable. These assumptions are listed in Table 3.

Given the design of the PYAS and the SAS as a whole, it is reasonable to make these assumptions about the scene, especially with regard to geometry. Baffling protects against large illuminated patches, while construction and mounting of the PYAS ensures that the camera, optics, and fiducial plate are all roughly parallel and will not rotate relative to each other. Finally required exposure should be constant during a single balloon flight, and can be set manually from the ground at the start of observation. Without these assumptions the problem would still be tractable, but perhaps not possible with limited processing power. Each portion of the PYAS algorithm can leverage these assumptions to justify simpler methods than might be required without. The next section considers many methods that could applied to the PYAS problem, some of which would have made it robust to the failure of these assumptions.

3. Related Work

The problem here is framed as solely a computer vision problem, where the design of the SAS hardware, especially fiducial pattern and solar optics, is considered to be fixed. The larger problem is that of sun detection and aspect determination, for which there is an extensive history. These pointing problems are likely very similar to the problem faced by the larger SAS system, so it is important to examine how past missions have solved problems like the one faced by HEROES. Existing work in computer vision is also directly applicable, especially any solutions to similar shape-finding and image registration problems.

Sun Sensors

Sun sensing has many applications, some with requirements very different than fine pointing. The range of applications has led to a wide array of cadence, FOV, and accuracy requirements, and as a result there are many approaches to sun detection. The simplest employ a small number of photodetectors with finely tuned electronics. An exmaple of these would be quad-cells like the Lockheed Intermediate Sun Sensor (LISS) and the uSS, the former of which has similar FOV and better performance than the PYAS [1] [2]. More recent designs gravitate toward detector arrays on the order of hundreds of pixels square, using custom cut apertures to generate simple computer vision problems. For example the Micro Digital Sun Sensor (μDSS) uses a pinhole aperture to generate a 10-20 pixel-wide image image of the sun on its sensor, which it locates quite accurately with a simple centroid algorithm implemented on an FPGA [3]. A similar but more complicated sensor called the Micro Sun Sensor (MSS) adds multiple pinholes for redundancy, and identifies individual projections by measuring inter-projection distances [4]. The concept of encoding identity of simple geometric marks in their spacing is one that used in the PYAS as well.

Table 4. Sun Sensors

	Design	FOV	Accuracy	Cadence
LISS	Quad-cell	6°	7 arcsec	Cont.
uSS	Quad-cell	120°	0.15°	Cont.
uDSS	CMOS array	94°	36 arcsec	10 Hz
MSS	CMOS array	120°	0.2°	2 Hz
RHESSI SAS	Linear CCDs	1.53°	0.4 arcsec	128 Hz

The Solar Aspect System of the RHESSI spacecraft falls somewhere between an imaging array and a quadcell [5]. The RHESSI SAS employed three linear CCD sensors, each with 2048 elements rotated 120° relative to eachother. By projecting the sun onto these sensors with a 1.55m focal length lens, the RHESSI SAS achieved a plate scale of 1.73 arcseconds/pixel, with each individual CCD having a FOV of about 1°. When the solar image was fully visible for a detector the SAS software could determine the locations of the solar limb and from these compute midpoint of the sun along that particular detector's chord through the solar disk. With the sun fully visible by two CCDs, the solar center could be estimated. Ideally the sun would be visible on all three. In that case, knowledge of midpoint of the sun on each CCD and the relative angle between CCDs would allow for a precise estimate solar aspect. A summmary of the perfomance of this and other sun sensors is given in Table 4.

Existing Algorithms

The scene the PYAS observed was actually quite benign when compared to some in modern computer vision. Camera and target were coupled together mechanically, and illumination of the scene was very tightly controlled. The result is that the PYAS problem decomposed into very basic image processing tasks, each of which had a well-established history. The task of locating the sun is one of finding a circle of known radius. The fiducial detection task is one of detecting small geometric markers and is another common machine vision problem. Finally the task of generating a mapping from pixel to gondola space is an image registration problem. This section will briefly review relevant literature for each of these tasks.

Circle or Ellipse Detection—The simplest method for determining location of a large circle like the solar image is a centroid of pixel intensity. Both of the imaging sensors in the previous chapter used this method to achieve sub-pixel accuracy [3], [4], and the same method has also been used in detection of circular fiducials where it is was favored for simplicity and accuracy [6] [7] [8]. However, all of these

cases treated small circles on the order of 10-20 pixels wide, not 180 wide like the sun in a PYAS image. Centroids can be sensitive to occlusions, irregular illumination, and sensor noise, all of which become more likely for larger shapes. In this case especially, fiducials represent a known, unavoidable occlusion. Other classic approaches to circle finding would be a matched filter or Hough Transform, but both of these are also better suited to smaller circles and smaller image [9] [10]. To search an entire PYAS frame with either method would be far too intensive computationally.

Edge detection and curve fitting is an attack that is better suited to larch circles than centroiding or matched filtering, and that scales better than Hough. After passing the image through an edge detection filter, the task becomes one of generating a set of parameters that fit a circle through the edge pixels. This can be framed as a least squares problem, and has been studied fairly extensively for both circles and ellipses [11] [12] [13]. Beyond the least squares approach, there are methods that focus on determining the center of the circle described by a set of points, then deriving radius as the average of the distance from each point to this center. One of these is Average Intersection, using triplets of non-co-linear points and treating them as endpoints of chords through a circle [14]. The bisectors of a circles chords intersect at its center, so the average of these intersections should provide a good estimate of the circle center. This method was shown to be sensitive to noise on the edge of the circle and to situations where chords were too short, i.e. points used to calculate an individual bisector were too close together, but is very similar to the method used by RHESSI. A modification of this method was ultimately used on the PYAS as well.

Fiducial Mark Detection—The task of finding PYAS fiducial markers is again a shape finding problem. Unlike the sun-finding task where it could be assumed there was only one circle in the frame, here there are guaranteed to be multiple fiducial markers. As with circle-finding, the task of locating small geometric marks has been present in machine vision for a long time. Although there is no direct requirement on the accuracy of fiducial marker locations, the goal for this stage was to once again to resolve to sub-pixel levels. This is a fairly common requirement for fiducial markers, so most of the methods considered here resolve fiducial location to sub-pixel level.

Regardless of the marker shape, the method most often employed for locating these kinds of fiducials is template matching. Common methods to further refine the locations from a template matching approach include centroiding, intensity interpolation, correlation interpolation, and curve fitting. Centroiding is by far the simplest and most popular, but was shown to work best on convex shapes like circles and diamonds, with degraded performance on cross-shaped marks [6] [7] [8]. Centroiding can also be used in a correlation image to refine peak location. With a sharp enough spike in correlation the problem becomes very similar to light stripe detection in one dimension, or to detection of point sources in two dimensions as would happen in a star tracker or a system using LEDs as active fiducials [15] [16]. Tests on centroid estimation in this context show it to be able to narrow down peak position to fractions of a pixel, but to be sensitive to window size, noise, and threshold [16] [17] [18].

Instead of using a centroid, it would also be possible to refine the pixel location returned by a matched filter by using interpolation. In intensity, this would mean up-sampling both image and template. Accuracy in the estimate of a fiducials location should increase proportionally with the up-sampling factor. This approach is analyzed in great detail in [19], where an algorithm is devised that reduces the cost both in required computation and storage. By comparison to most other methods this approach is extremely accurate, showing errors on the order of .005 pixels. However, this accuracy comes at the price of high computational cost. Even with the measures taken to reduce computation time, [19] concedes that correlation interpolation is faster. In the case of correlation interpolation, the idea is to perform a least squares fit of a paraboloid surface to the region around the maximum in correlation, using the maximum of this surface as the refined location of the shape [20]. This method was shown to be accurate down to 1/16 of a pixel for a wide array of image features.

A template matching approach followed by a local centroid is the most common fiducial detection method for small, simple fiducials. This method is easiest when applied to small, filled convex shapes like circles. Circles are the most popular due to their being easy to locate with a simple centroid and their being rotationally symmetric, something which cross marks are not. Because of their poor performance under centroiding, cross shaped fiducials require either an interpolation of either intensity or correlation to refine their location to sub-pixel. Centroid of correlation is computationally simpler than interpolating in either correlation or intensity, and has comparable performance.

Image Registration— The last problem to be tackled with the PYAS is one of image registration. Camera position in the PYAS setup was somewhat arbitrary, and certainly not calibrated. All calibration of the PYAS was done to determine position and orientation of the fiducial screen pattern relative to the other HEROES systems. The last section considered how fiducials on the screen could be located, and the ad-hoc methods developed for identifying them will be described in the next section. Once identified, each fiducial in the PYAS image offers a single point correspondence between the image plane and the plane of the fiducial screen. At the worst then this should be a problem of computing a projective transformation, which is addressed in [21]. The construction of the PYAS actually constrains this to something much closer to a simpler similarity transform. Furthermore there is very little rotation between the fiducial plate and the camera. This means that the problem of finding the mapping between sensor and plate amounts to determining the scale and offset between the two coordinate systems, requiring only three parameters to properly characterize the mapping. This problem was solved in the PYAS with a linear least-squares approach.

4. PYAS ALGORITHM

At a top level the algorithm can be broken into a few simpler problems. The solution to each individual problem can be developed separately, as long as it delivers the proper data to the next stage. This flow is illustrated in Figure 4. Where relevant, the coordinate space is listed. The ultimate goal is to determine solar center in gondola coordinates, or at least in a space which can be easily mapped to the same space as the gondola controller's coordinate system, such as location on the fiducial screen.

The four tasks are to locate the center of the sun, locate the visible fiducial markers, identify the visible fiducials, and transform the sun's location from image or pixel coordinates

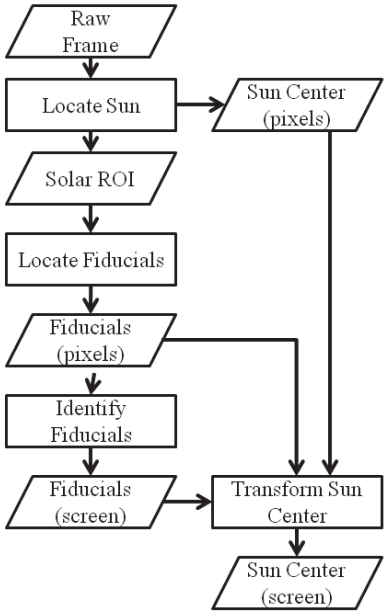

Figure 4. Flow chart of PYAS Algorithm

to a location on the fiducial screen. The entire algorithm was executed on each successive frame, depending on very little information from the previous frame. The only information carried over was the previous sun location in pixels, used in the first stage for placing chords.

Locate Sun Center

This stage takes in the raw PYAS frame and must determine both the location of the sun's center and an ROI containing the illuminated patch of the fiducial screen. This stage must determine the sun's location to better than a pixel in order to meet the 20 arcseond knowledge requirement. Once the sun's location is known, the ROI is just a neighborhood around that point. This neighborhood could be sized based on knowledge both of the sun's size given the time of year and PYAS plate scale, as well as knowledge of the payload's max slew rate based on past flights of the HERO payload.

An approach very similar to the RHESSI SAS can be taken if individual rows and columns of the PYAS detector are treated as independent linear CCDs. This is very similar to the Average Intersection method of finding the center of a circle, except it constrains points on a chord to be on the same row or column of the image, avoiding issues that arise when the two edge points are too close together. At a minimum the midpoints of a single row and single column could be combined to estimate the sun's center. The more robust approach taken with the PYAS was to take multiple rows and columns, locate the limb crossings and their midpoint for each, and average all of these midpoints to arrive at an estimate of the solar center.

This does not address how to place these chords: how to pick which rows and columns of the image to analyze. Ideally all chords are placed on the sun, which is possible if we have prior knowledge about the sun's location. Otherwise chords need to be placed evenly across the entire image. To take advantage of any prior knowledge there were two states. If the previous frame had determined a valid solar solution, then the previous ROI location could be recycled. In this case chords were placed evenly through the ROI. If the most recent frame had not produced a valid solution, chords were placed evenly across the entire image. This is illustrated in Figure 4

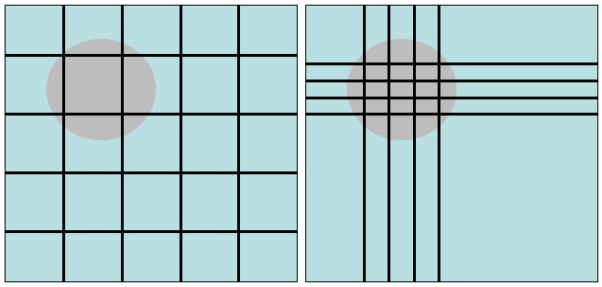

Figure 5. Placement of chords in an image when previous sun location is either known or unknown.

Potential limb crossings were determined by searching each chord for pixels immediately above a brightness threshold. Ideally there would be two of these per valid chord: one transitioning onto , and one transitioning back down from the solar disk. Ideally all chords would either meet these criteria, or have no crossings at all. Any chord that did not meet this criteria could be rejected. The two threshold crossings are considered to be approximate limb crossing locations. For each of these crossings, a local linear fit is generated. This is used in an attempt to refine the threshold crossing to the exact point at which the solar limb crossed the brightness threshold. Since the solar limb has a very stable brightness profile, points of identical brightness on the limb are spaced the same distance from the center of the sun.

This approach still needs some refinement. There are several features which do not meet this model, and foremost among them are fiducial marks. As these are black marks against a white background they can produce spurious limb crossings whenever they cross a chord. Because of the fiducial marker arrangement, it is highly likely that a chord will cross at least one marker. The solution is to both set the threshold below the brightness of a fiducial marker, and to reject any pairs of threshold crossings with slopes and spacing that could correspond to a fiducial.

In addition to fiducial markers, there were other non-ideal features which cropped up in both tests and during flight. Although the optical path was enclosed by baffles, stray light was still present due to bleed through joints in the baffling. This is sometimes visible as a bright patches or patches around the edge of the fiducial screen. Carefully sealing the baffle joints to light cut down on this. Hot pixels and scratches in the fiducial screen also posed a problem, usually as one or more saturated pixels. These were removed by adjusting their intensity value to the 99th percentile of brightness across the image, effectively removing any intensity bright outliers.

The algorithm for sun-finding is summarized by the pseudo-code in Algorithm 1. This code requires the current frame and the previous ROI location as inputs. It also needs a threshold, K, for defining limb location as a function of total brightness of the sun, and a number of chords to use per axis, N.

Outputs of this stage are the $SunCenter$ and a ROI which should contain the patch of frame illuminated by the sun. It also returns the 1st and 99th percentile cutoffs for pixel brightness ($RobustMin$ and $RobustMax$)

Algorithm 1 LocateSun

Require: **CurrentFrame**, $PastROI$, K, N
Ensure: $SunCenter$, **ROI**, $RobustMax$, $RobustMin$
  $RobustMax \leftarrow$ 99th percentile of $CurrentFrame$
  $RobustMin \leftarrow$ 1st percentile of $CurrentFrame$
  $Threshold \leftarrow$ K% of $RobustMax$
  if $PastROI$ is valid
    Chords $\leftarrow$N rows and N columns spaced evenly across $PastROI$ in **CurrentFrame**
  else
    Chords $\leftarrow$N rows and N columns spaced evenly across entire **CurrentFrame**
  for each **Chord** in **Chords**
    for each $ThisPixel$ at $ThisIndex$ in **Chord**
      if $((ThisPixel > Threshold)$ & $(PastPixel < Threshold))$
        Edges $\leftarrow ThisIndex$
      else if $((ThisPixel < Threshold)$ & $(PastPixel > Threshold))$
        Edges $\leftarrow -PastIndex$
      for each $ThisEdge$ in **Edges**
        if $(|ThisEdge| - |PastEdge|) < FiducialLength$
          remove $ThisEdge$ and $PastEdge$ from **Edges**
      if $((numel(\mathbf{Edges}) = \mathbf{2})$ & $(FirstEdge > 0)$ & $(LastEdge < 0))$
        for each $Edge$ in **Edges**
          Nbhd $\leftarrow$ pixels before and after pixel at $Edge$
          $Line \leftarrow$ Least squares fit line to **Nbhd**
          $Limb \leftarrow Index$ s.t. $Line(index) = Threshold$
          if Chord is a row
            RowLimbs $\leftarrow Limb$
          else Chord is a column
            ColLimbs $\leftarrow Limb$
  $SunCenter \leftarrow [mean(\mathbf{ColLimbs})\ mean(\mathbf{RowLimbs})]$
  ROI $\leftarrow$ Subset of **CurrentFrame** around $SunCenter$

Locate Fiducials

The ROI determined in the previous stage should contain the only illuminated fiducials in the PYAS frame. The goal in this stage is to locate fiducial markers in the ROI, again down to a sub-pixel level. Several potential methods were entertained for locating fiducials, but ultimately the simplest was deemed to be a form of template matching. While other considered methods might have been significantly faster, template matching proved to be extremely effective without requiring complicated logic and error checking that more ad-hoc methods would have needed. Thanks to the reduced size of the solar ROI and the small size of fiducial markers, convolution with a mask has a low enough computation time to be viable. An edge-based template based on the work in [9] provided extremely strong responses to fiducials which in turn were easy to refine down to sub-pixel level.

In the ideal case the edge-based template matching algorithm mentioned above worked basically out of the box. The image could be convolved with a pre-generated matched filter to generate a response image. Examples of each are shown in Figure 4. Setting a threshold on response and searching for local maxima above that threshold was sufficient to find nearly all fully-illuminated fiducials, and even some partially-illuminated marks. A thresholded centroid of the response image around each local max refined the location of the peak response, and therefore of the fiducial, to better than pixel level.

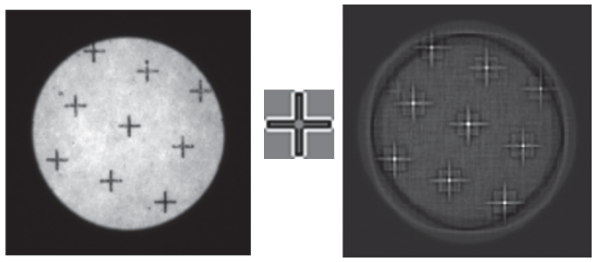

Figure 6. From left to right: Example of a PYAS ROI, edge-based template, convolution image.

As in sun-finding stage, there are a few edge cases here that can hinder determination of fiducial locations. The matched filter really only generated spurious detections for sharp edges. Bbaffle leaks were too dim and dispersed to generate a strong enough response. On the other hand scratches proved to have much more impact. Scratches to the fiducial plate showed up as long saturated strips of pixels,and were often able to produce responses to the fiducial template that were large enough to register as fiducial locations. This seemed to be a result of their simply having a significantly higher brightness value than their neighbors. To avoid this complication, any point above the 99th percentile was set to the 99% value. This was very effective at removing undesirable peaks in the image while affecting a small number of pixels overall. With these points removed, thresholds for detection could be tuned to produce reliable detection with very few false-positives. Thresholds were set in terms of standard deviations above the mean pixel brightness value, which made them fairly robust to changing intensity of the solar image. To further protect against false-positives, if multiple detections occurred within a fiducial length of each other, only the highest of these responses was kept as a valid response. The rest were discarded.

The pseudocode in Algorithm 2 describes this stage in the algorithm. The inputs to this section of code are the **ROI** from the previous stage, the $RobustMax$ computed in the previous stage, and a few parameters. The $K1$ and $K2$ parameters specify how many standard deviations above the mean to set thresholds for detection and centroiding respectively. The $NbhdSize$ variable sets the size of the region around each fiducial to use when computing a centroid.

The output to the next stage is the **Fiducials** variable, a list of the coordinates of fiducials in pixel space. As long as there are at least three non-collinear fiducials, it is possible to proceed to the next step and attempt to identify the detected marks.

Identify Fiducials

Once fiducials are located, they must be identified. As discussed in the introduction, fiducials are spaced on a rotated grid. For each row of fiducials, vertical spacing between fiducials increases by a fixed amount, while horizontal spacing between fiducials is unique depending on location in the grid. The reverse is true down columns of the grid. The idea then is to find pairs of fiducials which meet the fixed spacing, and then identify the pair based on the varied spacing.

Algorithm 2 LocateFiducials

Require: $ROI, RobustMax, K1, K2, NbhdSize$
Ensure: **Fiducials**
  for each $Pixel$ at $Index$ in **ROI**
    if $Pixel > RobustMax$
      $Pixel \leftarrow RobustMax$
  $Mean = Mean$(All $Pixel$ in **ROI**)
  $Std = StdDev$(All $Pixel$ in **ROI**)
  $Threshold = Mean + K1 * Std$
  load **FiducialTemplate**
  Response $\leftarrow conv2$(**ROI**, **FiducialTemplate**)
  for each $Pixel$ in at $Index$ in **Response**
    if $Pixel > Threshold$
      if $Pixel$ is a local maximum
        for each $Fiducial$ in **Fiducials**
          if (a fiducial at $Index$ would overlap a fiducial at $Fiducial$)
            if $Pixel >$ **Response**$(Fiducial)$
              $Fiducial \leftarrow Index$
            else
              skip to next $Fiducial$
          else
            Fiducials $\leftarrow$ **Index**
  if $numel$(**Fiducials**) $>$ **12**
    keep 12 $Fiducial$ in **Fiducials** with largest **Response** value
  $Threshold = Mean + K2 * Std$
  for each $Fiducial$ in **Fiducials**
    $Nbhd \leftarrow$ subimage of **Response** of size $NbhdSize$ centered at $Fiducial$
    $Refinement = centroid(Pixel$ in $Nbhd$ s.t. $Pixel > Threshold)$
    $Fiducial = Fiducial + Refinement$

Identification revolves around finding pairs of neighboring fiducials. These are spaced by a fixed distance in one axis, and a varying distance in the other which is used for identification. Identification proceeds by searching the entire list of fiducials for pairs of the correct fixed spacing either along either pixel rows or pixel columns. The result is two lists of pairs, one where the fixed distance along rows is valid, and one for column spacing. Each list is then searched for pairs with valid fiducial spacing in the other axis. If a valid pair is found, both members are given the ID for that distance. For example, if a pair of fiducials was found to be spaced by 15 pixels along rows, it would be deemed a column pair. If the spacing along columns was 45 pixels, then this pair would get column IDs of 0 and 1 respectively. Each would need to be a member of a valid row pair to get the other half of its ID. This spacing is illustrated in Figure 4.

If after this first pass there are fiducials which were not deemed to be members of any valid pairs, they are discarded. Usually these were false-positives from the previous stage. Tight thresholds were necessary on allowable row/column distances to prevent false positives, since false positives had the potential to adversely affect the mapping generated in the next stage. For this reason it was preferable to potentially have more missed detections as opposed to false positives.

After attempting to ID all the fiducials present, and dropping those without even a partial ID, there still are likely some with half of a valid ID pair. These are fiducials which only belonged to one pair, or to two co-linear pairs. To attempt to get around these partial IDs, fiducials missing say a row

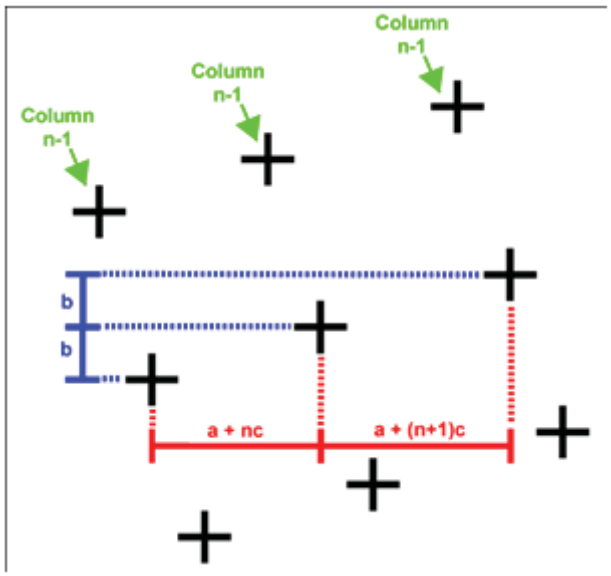

Figure 7. Fiducial Spacing. Fixed spacing is marked 'b' down the rows of fiducials. ID spacing is shown between columns.,

ID can take the value of an adjacent fiducials row ID if the two are in a column pair, and vice versa. Propagating IDs like this does have a drawback because a single incorrectly identified fiducial has the potential to corrupt the IDs of adjacent fiducials. It is also possible for adjacent fiducials to disagree over which row/column a common neighbor belongs to. In cases like these it is almost guaranteed that one of the adjacent fiducials has an invalid ID, but it is impossible to tell from this dispute alone which of the two is incorrectly identified. At absolute most there were only ever 12 fiducials visible, and often this number was closer to 4 or 5. With so few, it becomes very difficult to use any kind of majority vote to determine which fiducials were identified correctly. Ultimately the simplest strategy was taken for any disputes over ID of a given fiducial: if full ID for a fiducial could not be determined, the fiducial was simply ignored.

The pseudocode below summarizes the logic used when identifying fiducial markers. As input it takes simply the locations, in pixel space, of identified fiducials. Also required at this point are the actual spacings for the fiducial grid. The FixedSpacing is the vertical spacing between adjacent pixels on the same row, or vice versa for columns. The IdSpacings are the spacing between rows or columns of fiducials. In addition to this information, we add a tol variable, to capture how close to the expected distances pairs need to be.

Output here are two lists, one of fiducials, and the other a list of equal length with corresponding fiducial IDs. The list of fiducials is ideally identical to the one given as input, but in the event that any fiducial was not able to be identified in the previous step, it is removed from the list. As with previous stages, these outputs need to be vetted before continuing to use them to generate a mapping, and mapping the solar center to screen coordinates. To generate a mapping, there should be at least 3 fiducials in the list, and their IDs should indicate that they are not co-linear: all three IDs should not belong to the same row or column. As long as this holds true, then it should be possible to map from pixel space to screen coordinates given the fiducials and IDs found here.

Algorithm 3 IdentifyFiducials

Require: $\mathbf{Fiducials}, Tol, FixedSpacing, \mathbf{IdSpacings}$
Ensure: $\mathbf{FiducialIDs}$
  $\mathbf{AllPairs} \leftarrow$ all pairs of fiducials in $\mathbf{Fiducials}$
  for each $Pair$ in $\mathbf{AllPairs}$
    if $(RowSpacing(Pair) FixedSpacing) < Tol$
      $\mathbf{ColPairs} \leftarrow Pair$
    if $(ColSpacing(Pair) FixedSpacing) < Tol$
      $\mathbf{RowPairs} \leftarrow Pair$
  for each $Axis$ in $[Row, Column]$
    for each $Pair$ in $\mathbf{AxisPairs}$
      for each $IdSpacing$ at Id in $\mathbf{IdSpacings}$
        if (AxisSpacing($Pair$) ¡ $IdSpacing$) ¡ Tol
          for each $Fiducial$ in $Pair$
            $Fiducial.\mathbf{AxisVotes} \leftarrow Id$
    for each $Fiducial$ in $\mathbf{Fiducials}$
      if mode($Fiducial.\mathbf{AxisVotes}$) is unique
        $Fiducial.ID(Axis)$ = mode($Fiducial.\mathbf{AxisVotes}$)
      else
        $Fiducial.ID(Axis)$ = UNKNOWN
  for each $Axis$ in $[Row, Column]$
    for each $Pair$ in $\mathbf{OtherAxisPairs}$
      if $Pair(Member).ID(Axis)$ is unknown
        if $Pair(OtherMemeber).ID(Axis)$ is known
          $Pair(Member).ID(Axis) \leftarrow Pair(OtherMember).ID(Axis)$
    for each $Fiducial$ in $\mathbf{Fiducials}$
      if $Fiducial.ID(Axis)$ is unknown
        if mode($Fiducial.\mathbf{AxisVotes}$) is unique
          $Fiducial.ID(Axis)$ = mode($Fiducial.\mathbf{AxisVotes}$)
  for each $Fiducial$ in $\mathbf{Fiducials}$
    if (($Fiducial.ID(Row)$ is unknown — ($Fiducial.ID(Col)$ is unknown))
      Remove $Fiducial$ from $Fiducials$

Transform Sun Center

Once fiducials are identified, their true location in screen coordinates can be generated from calibration data. Their locations on the screen are known from the printing process, and calibrations before flight. The transformation from screen coordinates to gondola coordinates was also measured pre-flight. The goal at this stage is to generate a mapping from pixel to screen coordinates and to apply this mapping to the sun location found in the first stage. The assumption that mapping from plate to sensor is a scale and shift is used in this stage. This means that a simple linear fit can be used in each axis independently to model the change in coordinates from pixel space to fiducial screen. The only requirement levied on the previous stage is that there be a sufficient number of pairs to do a proper fit.

As mentioned previously, this stage accepts a list of fiducial locations in pixel space and their corresponding IDs. The ID locations on the fiducial screen, and therefore relative to the gondola coordinate system, are pre-determined by calibration on the ground. Mappings were generated with a simple linear least squares fit.

This stage outputs the linear mappings generated, as well as the sun center location in mils relative to the center of the fiducial grid. The mappings are checked to verify that the identified scale factor is approximately correct. The

Algorithm 4 TransformSunCenter

Require: $\mathbf{Fiducials}$, $\mathbf{FiducialIDs}$, $\mathbf{ScreenLocations}$, $SunCenter$
Ensure: $Mappings, ScreenCenter$
  $ScreenFiducials \leftarrow ScreenLocations(FiducialIDs)$
  for each $Axis$ in $[Row, Column]$
    $\mathbf{Mappings}(Axis) \leftarrow$LinearFit($\mathbf{Fiducials}(Axis)$, $\mathbf{ScreenFiducials}(Axis)$)
    $ScreenCenter(Axis) \leftarrow \mathbf{Mappings}(SunCenter)$

ScreenCenter is passed to another module for final adjustment into gondola coordinates, and then used to generate offset from the desired solar target.

Summary

The PYAS algorithm outlined above was used to generate pointing solutions for the HEROES balloon on its 2013 flight. The basic flow was established early in development, and functionality of each stage was fleshed out in sequence. Changes were made to each stage to account for disturbances, misalignments and the like as they were detected during testing. Much of the code was prototyped in MATLAB, but ultimately implemented in C++ using the OpenCV library. The next section will go over testing of the algorithm against artificial data in MATLAB, as well as performance and testing pre-flight with the OpenCV implementation.

5. PERFORMANCE AND TESTING

The task of verifying PYAS performance is about as complicated as the PYAS problem itself. Most of this arises from difficulty in getting a ground-truth for the instrument. In the case of the sun and fiducial finding, which were inspected most heavily, artificially generated data offers the only chance to test the algorithm against a known solution. The downside is that it is difficult to generate good approximations of the images actually generated by the PYAS. Verifying the PYAS on actual test observations obviously does not have this problem. On the other hand, while test observations are obviously an excellent representative of what will be seen on flight, they lack any ground truth at all. Inspection cant really offer solutions to a sub-pixel level, which leaves either a priori knowledge of the pointing of the PYAS system during a test, or another algorithm which has already been proven to work. The latter is impossible while only using images generated by the PYAS, and the former is impossible without an elaborate test setup able to adjust PYAS position relative to the sun in a precise way.

To verify requirements for the algorithm, sun tracking and fiducial detection are verified against synthetic test data. Description of how this data was generated and how it was used to test each segment of the algorithm is given as well. In contrast, the effects of image registration will be considered directly on data from full system tests. Finally, overall system performance will be assessed based on data from the 2013 HEROES flight itself.

Synthetic Data

Sun Finding—In the case of the sun finder, artificial data was created by drawing a filled circle at a scale of 1 arcseconds per pixel then down-sampling the image by a factor 10.6 to approximate the ideal PYAS image scale. After a simple Gaussian blur and the addition of some white noise, these

Table 5. **Parameters for synthetic sun test images**

Parameter	Value
Range of centers	-5 to 5 arcseconds in each axis
Sun diameter	180 pixels
Oversampling factor (arcsecond to pixels)	10.7

Table 6. **Parameters for test of performance vs number of chords**

Parameter	Value
Noise Std Dev	5 counts
Threshold	180 pixels
Number of Chords (N)	3 to 90 per axis

images are a fair approximation of actual PYAS images, with ground truth given by the scaled coordinates of the sun center in the original image. The PYAS sun tracking algorithm was tested against these images, and performance was measured against number of chords and level of white noise. Noise level varied, but the parameters below describe the other parameters of the set of test images.

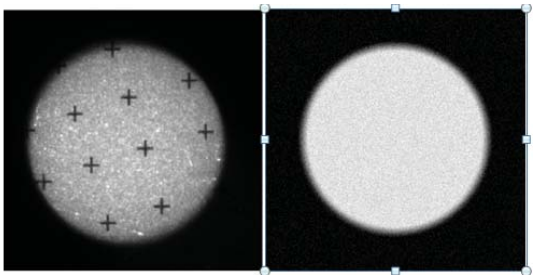

Figure 8. Cropped PYAS image from sun test data (left). Synthetic sun image used in testing (right)

This stage of the PYAS algorithm has two parameters which can be tuned to affect response: threshold level K and number of chords N. In all tests the threshold K was set to 50% of max brightness. In this test, N was varied to see the effect of the number of chords on accuracy. The standard deviation of the white noise used in this portion of the test was held fixed at a level similar to that of ground test data. The parameters for this test are listed in Table 6.

The results are shown in Figure 5. The center determined by this algorithm was compared to the ideal center, and the magnitude of this error was used to measure performance. The plot below shows the 3σ error in arcseconds over all positions for a given number of chords. Theres clearly a sharp improvement in performance initially, which starts to flatten out after 10 chords per axis. Based on this test the number of chords to be used in flight was set at 10.

The next test used a fixed number of chords and varied the noise level to see how sensitive this method is to increased noise. A range of values were used for the noise σ: from noise-free to an order of magnitude worse than the expected noise level. Parameters for this test are listed in the table 7.

As before, performance can be measured in terms of 3σ error

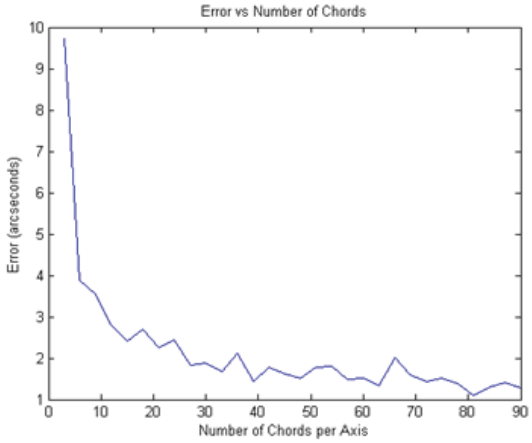

Figure 9. Performance vs. number of chords. Error shown is 3σ of error values for each parameter value.

Table 7. **Parameters for test of chord finding vs noise**

Parameter	Value
Noise Std Dev	.01 to 100 counts
Threshold	105 counts
Number of Chords (N)	10 per axis

versus noise level. Also of interest, however, is number of chords dropped. The likelihood of a chord failing to meet the criteria listed above for a valid chord increases with noise level. At a certain point no valid chords will be found in the image. Although it is unlikely that such a high level of noise would be encountered on flight, it was of interest in development to see how robust this method is to high levels of noise. Plots of both error vs. noise and dropped chords vs. noise are shown in Figure 5.

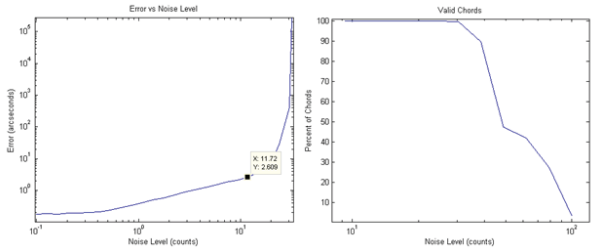

Figure 10. Plots of error and percentage of dropped chords vs. noise level. Noise level is measured in bits in the intensity, and chords are dropped when they fail to meet the criteria outlined in 4

The plot of error shows that beyond noise levels of 15 counts the sun tracking algorithm fails to meet requirements at all. At the even higher level of 30 counts the number of valid chords plummets, corresponding to a spike in error. The hope was that noise levels in flight data would be significantly lower.

The last test of this algorithm was designed to see how it behaves in the presence of fiducial markers. For this, a simulated screen image was generated using the same method used in generating the sun image: an image was generated at higher scale and then scaled down to true size. Portions of this screen were then "illuminated" by multiplying regions of

Table 8. Parameters for test of sun-detection in the presence of fiducial markers

Parameter	Value
Noise Std Dev	5 counts
Threshold	105 counts
Number of Chords (N)	10 per axis
Sun center	(0 arcseconds, 0 arcseconds)
Background	One of 81 subsets of fiducial pattern
Trials per background	10

the screen image with the synthetic solar image, giving test images like the one shown in the Figure 5.

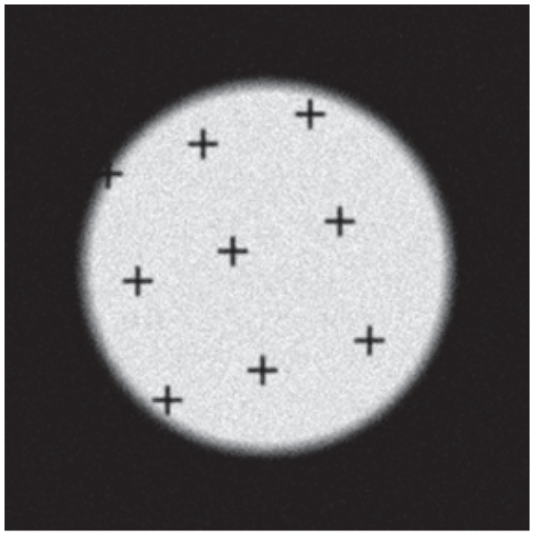

Figure 11. Test image containing fiducial markers

The parameters for this battery of testing are listed in the Table 8. The goal here was to see how well this algorithm can handle encountering fiducial markers. While certainly not perfect, comparing this to the flight data at the start of this section shows that it is a reasonable approximation of what true PYAS images look like.

The bulk of values fall below this threshold, but it is clear that there is a subset of frames where this algorithm fails to meet the requirement. Even though most values fall below the threshold, the bound on error here is much higher than what was seen in tests without fiducials present. The reason for this is visible in the example frame shown above, and was overlooked in the planning stages of the PYAS code. Looking at the top left fiducial in the sample image above, it is clear that if a chord were to fall horizontally across the fiducial it would be impossible to tell where the true edge of the sun was. The right edge of the horizontal arm of that fiducial would be detected as the edge of the sun, while the actual edge lies closer to the left edge of the marker. This chord would introduce an offset, and given that there are only 10 chords at most through the whole image, one error can introduce a significant pull on the estimate of the center.

Fiducial Detection—Artificial fiducial marker data was described briefly in the previous section, where an artificial

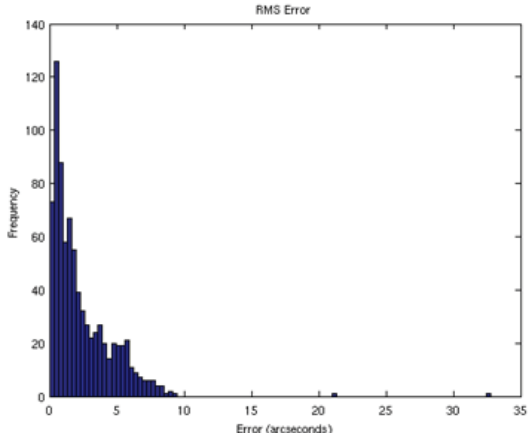

Figure 12. Histogram of error levels for trials in the test of chord-based sun-detection in the presence of fiducial markers. Almost all tests fell within the 10 arcseconds requirement.

Table 9. Performance of various fiducial location methods

Method	3σ RSS error
No Refinement	4.2 arcseconds
Centroid (FLIGHT)	1.55 arcseconds
Centroid (Improved)	.68 arcseconds

fiducial screen was used to test the sun tracker. In this section rather than the entire screen, only images of a single fiducial will be considered. With the threshold set properly, the fiducial detection was able to determine fiducial location to a pixel level for fully-illuminated fiducials without any issues. Of interest here is how accurately the refinement method described in Section 4 can resolve sub-pixel locations.

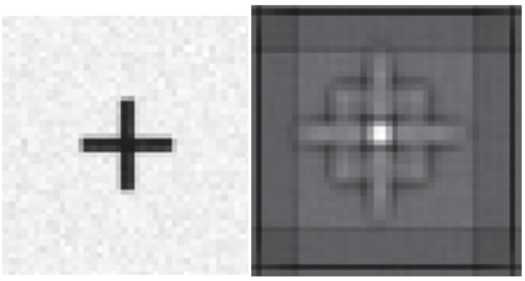

Figure 13. Example of a fiducial test image, and the corresponding correlation image

To test this, a set of test images was generated where the location of the fiducial mark varied across 6 different locations in mils on the fiducial screen in each axis. All 36 of these locations would produce a peak in the correlation response at the same pixel. The error in fiducial location without any refinement is listed in the table below. Also included are the results of refining with a centroid, which was the method taken with this algorithm.

Error for taking no action is what would be expected, the equivalent of about half a pixel. Refining the location of the peak with a centroid was able to cut that value in half. Centroiding to determine correlation peak was chosen because

the method is fast, and because during development there were problems forming a well-conditioned paraboloid fitting problem. Paraboloid fitting would have been the preferred approach here, and was shown in the literature to perform nearly an order of magnitude better than the performance seen here. The reason for poor performance of the centroid is a systematic error described below.

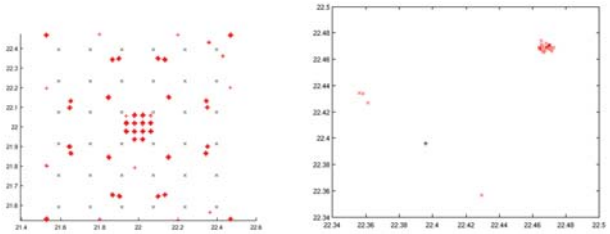

Figure 14. At left, true fiducial locations are shown in black, all corresponding to the same integer pixel location. Attempts to refine fiducial location to sub-pixel are shown in red. Clearly there is systematic error present. At right, the same is shown for a single point.

The graph on the left of 5 shows a plot of predicted locations in red vs. true locations in blue for all the locations tested. The right shows an example for a single point. Part of this seems to be intrinsic to using a centroid to refine position of a peak, but there is also a component here caused by not removing the threshold value from the pixel values before computing the centroid. Points farther from the center have a greater moment on the resulting solution, and by not subtracting the threshold, they also have more "mass." The results of subtracting this value when computing centroid are shown in Figure 5.

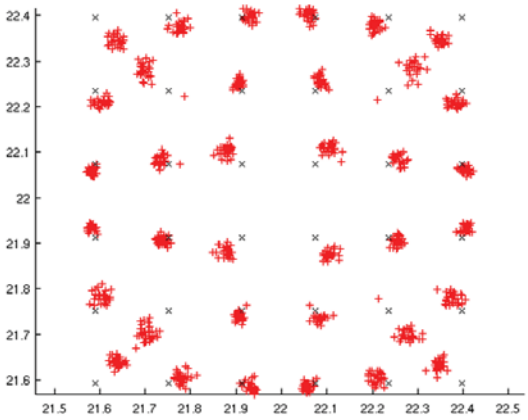

Figure 15. An improvement on the method fiducial detection used on flight.

Although clearly an improvement, there is still systematic error in this estimate. Several alternate methods for refining fiducial locations were considered earlier, and one of these would likely be a better candidate than the centroid-based refinement described here.

Test Data

Several tests were done of the PYAS system before flight, the vast majority of which were performed while attached to the HEROES payload. Performance of the sun tracking portion of the code cannot readily be assessed because the PYAS is folded into the larger HEROES control loop. On the other hand, the fiducial plate and camera are fixed relative to each other, so performance of the mapping computation can be assessed directly. For the fiducial location and identification, inspection shows that missed detections and false-positives are uncommon, but quantifying these values would require inspection of every frame. Instead, this section will look at the jitter in fiducial locations and at how jitter in fiducial locations propagates through to the final aspect solution.

Fiducials—Although the sun may move while the gondola pans to track a solar target, the fiducial pattern is fixed relative to the camera. Therefore motion of the pattern relative to the camera should be fairly small. Although some of the jitter on fiducials may come from motion of the gondola or flex of the PYAS, assuming all noise present is due to the detection algorithm is a fair worst-case assumption. The plots below show scatter of fiducial measurements and RSS error vs. the mean fiducial location for what should have been a static observation.

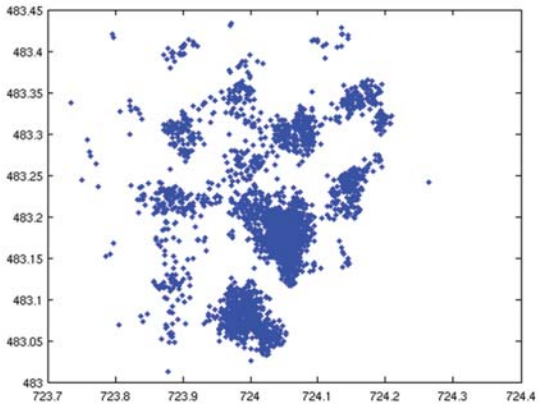

Figure 16. Spread of fiducial locations from a test of the PYAS system

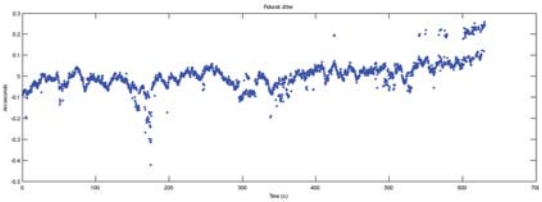

Figure 17. Fiducial location from a ground test, in a single dimension, plotted as a time series

It is interesting to see that there is some structure to the data. The quantization seen in the 2D scatter is likely a result of the systematic error shown to be present in the centroid method for refining fiducial location. This same quantization is even visible in the time series data after 800s, where there the data is clearly bimodal. As for the periodic nature of the time series, a 25s cadence could reasonably be motion of the gondola.

Registration—In tests with synthetic data it was concluded that the sun tracking algorithm in the presence of fiducials and noise is only trustworthy to about a pixel, or 10 arcseconds. This is an approximation which does not factor in effects of mapping from pixel space to the coordinate space of the fiducial screen. The effect of the mapping could be computed analytically, either assuming a Gaussian distribution on the location of the fiducials, or approximating one by computing the standard deviation of the jitter plotted above. Instead,

Table 10. **Measured noise values for image registration parameters**

	Variance
$\sigma^2_{m_1}$	1.045 (mils/pixel)^2
$\sigma^2_{m_2}$	1.579 (mils/pixel)^2
$\sigma^2_{b_1}$	5.089 mils^2
$\sigma^2_{b_2}$	3.111 mils^2
σ^2_x, σ^2_y	.0978 pixels^2

however, direct measurements of jitter on the parameters of the fit can be used. The mapping in question takes the form of two linear fits, one in each axis. The parameters of these fits were computed for every frame in an observation of a fixed target, and their variances are given in the Table 10, along with an estimate of the variance of the points in the sun center.

Combining all these values together is a requiers an expression for the covariance matrix of a noisy point mapped with a noisy transformation H, which is given in [29]. The worst case estimate of covariance for the coordinates of the sun center is $14.4mils^2$. Adjusting for plate scale the total error estimated to have standard deviation of 6.512 arcseconds. Comparing to the 3σ bounds being used earlier, that puts the estimate for error in the computed solution at 19.5 arcseconds, just barely under the 20 arcsecond requirement.

Summary and Flight Performance

To review, there were a handful of requirements levied on the SAS which were relevant to the PYAS algorithm. Accuracy was the driving requirement and has been the focus of this chapter. Software also had a direct impact on cadence, and this requirement has been qualitatively addressed throughout this work. The design of the algorithm was made with cadence in mind, which helped in making image saving, rather than processing, the tall pole from a cadence standpoint. Finally FOV was mentioned as a requirement but also not addressed explicitly. The PYAS frame was sized so that the entire sun was fully visible anywhere in a +/-1 degree range. Outside of this range the chord-finding method will return a degraded solution, and it becomes increasingly unlikely to have sufficient fiducials to register the PYAS frame to the fiducial screen.

Ideally the flight processing algorithm would have enough margin to double as post-flight algorithm as well. Unfortunately the PYAS flight algorithm overshot its target accuracy level and appears to only barely meet the knowledge requirement for flight. The flight was successful however, and lessons learned in development and testing of the algorithm apply directly to the post-flight data processing. Recommendations for improved accuracy when processing PYAS frames will be described in the next and last chapter.

As for the flight, the HEROES payload had a successful launch in September of 2013 [30]. The flight lasted over 24 hours, and the payload was at float for 21 of those. Solar pointing time was 7 hours, and the PYAS system provided aspect solutions for the duration. During these observations, the HEROES control loop was required to point the payload with pitch-yaw jitter having a 50th percentile of 1 arcminute. Measurements taken by the PYAS-F show that over the duration of the solar pointing period, the HEROES payload was within 10 arcseconds of the target in elevation and 30 arcseconds of the target in azimuth 50% of the time. Even factoring in 20 arcseconds of potential error in aspect knowledge, the HEROES payload still meets pointing requirements. The flight knowledge requirement levied on the PYAS-F is rolled into this jitter requirement, and meeting it means the HEROES telescope was pointed successfully.

6. CONCLUSIONS

Software

Lessons learned from the flight processing algorithm for the PYAS could be applied to the post-processing of data on the ground, and possibly to future systems based on or similar to the PYAS. For the ground processing, this could be as little as making minor tweaks to the flight code or as much as a completely new algorithm. Some aspects of the flight algorithm worked adequately: fiducial identification and mapping from pixels to screen for example. However the algorithm would benefit from a major change to its overall structure, and then adjustments to the first two stages.

First and simplest, the method for refinement of fiducial locations needs to be changed. The systematic error present in the correlation centroid could be eliminated by switching to a parabolic surface fit to the correlation peaks. Outside of that systematic error, however, the matched filter approach worked quite well for locating fiducials. Next to be modified would be the sun finding method. The Average Intersection was attractive because of its nature as a heritage approach, being very similar to the method used by RHESSI in its SAS. RHESSI did not have to contend with fiducial marks, however, which can corrupt the solar limb and provide a disturbance when they occur on the body of the sun. They will affect any edge-based approach, meaning either their locations need to be predicted and masked out, or a different approach to sun detection has to be considered.

Moving away from edge-based methods to something like circle-enclosing instead [22]. Fiducials effectively subtract portions of the sun, but enough of the edge is present that an enclosing method should give a good estimate of the center. Another alternative would be to revisit the use of a simple centroid. Centroids were ruled out early in the development of the PYAS algorithm partly because fiducial markers would occlude portions of the sun and affect the centroid. This may have been more of a problem if large convex shapes like circles or diamonds were used as marks, but the cross marks used in the PYAS darken at worst 56 pixels per mark. At most that means about 3% of the sun might be occluded. Its possible that the effect of this on the centroid of the sun would be minimal. Converting to a binary image and using a morphological closing could further reduce the impact of fiducials on a centroid estimate.

Hardware

In addition to direct changes to the PYAS algorithm, it would indirectly benefit from changes to hardware, specifically the SAS computer and the fiducial pattern. The first of these would be to try to upgrade the SAS computer. The initial goal of a 10 Hz cadence proved to be impossible with the selected hardware. A faster processor would certainly help here, and there are certainly processors more powerful than the Atom. Additionally, more time needs to be devoted to the problem of storing images. Images in the SAS were stored as uncompressed FITS files, and file I/O ended up being a

major bottleneck in the processing pipeline. PYAS frames have a great deal of empty space, and if not video, even lossless image compression would cut down on raw file size and might help to improve file write times.

The fiducial pattern could stand to be changed as well. Cross-shaped fiducial markers are easy for humans to locate, but circular markers may be more suited to machine vision applications. They are easier to locate precisely via a simple centroid, and although the PYAS did not encounter problems with rotations, circular marks would be much more robust to potential rotation between the camera and screen. The method of encoding fiducial ID in inter-fiducial distances should also be re-addressed. The current method is very sensitive to changes in scale, and required adjustment to parameters in the identification code between test configuration and mounting on the gondola. Augmented reality literature is full of fiducial markers that carry information payloads, and there are even patterns of relatively small marks that still carry information without relying on inter-fiducial distance. For example, the fiducial marks used in the 2-axis encoder described in [23] rely on a pattern of small marks which can deliver position knowledge when only part of the pattern is visible.

The PYAS system successfully provided fine pitch-yaw knowledge to the HEROES pointing control system during solar observation. It exceeded cadence requirements, and analysis shows that it managed to meet pointing knowledge requirements, if barely. Solar pointing data from flight appears to have met jitter requirements in both axes as well. Because it was being developed in parallel with the PYAS hardware, the PYAS algorithm had to be capable of handling a wider array of possible scenes. Now that the PYAS optics, fiducial pattern, and camera are finalized, however, and the concerns about processing time have been completely removed, it should be possible to extract much finer pitch and yaw knowledge from the recorded PYAS frames. Further systems based on a PYAS would also benefit from a more powerful computer and a modified fiducial pattern.

REFERENCES

[1] T. T. Tarshis and G. T. Sakoda, "A Second Generation Sun Sensor for Sounding Rocket Applications," pp. 85–90, 1979.

[2] P. Ortega, G. López-rodríguez, J. Ricart, M. Domínguez, L. M. Castañer, S. Member, J. M. Quero, C. L. Tarrida, J. García, M. Reina, and A. Gras, "A Miniaturized Two Axis Sun Sensor for Attitude Control of Nano-Satellites," *IEEE Sensors Journal*, vol. 10, no. 10, pp. 1623–1632, 2010.

[3] N. Xie and A. J. P. Theuwissen, "A Miniaturized Micro-Digital Sun Sensor by Means of Low-power Low-noise CMOS Imager," *IEEE Sensors Journal*, vol. 14, no. 1, pp. 96–103, 2014.

[4] S. Mobasser, C. Liebe, and J. Naegle, "Flight Qualified Micro Sun Sensor for Mars Applications," *Proceedings of 2nd International Conference on Recent Advances in Space Technologies, 2005. RAST 2005.*, vol. 3, pp. 234–239, 2005.

[5] R. Henneck, J. Bialkowski, F. Burri, M. Fivian, W. Hajdas, A. Mchedlishvili, P. Ming, K. Thomsen, J. Welte, A. Zehnder, B. Dennis, G. Hurford, D. Curtis, and D. Pankow, "The Solar Aspect System (SAS) for the High Energy Solar Spectroscopic Imager HESSI," *SPIE Conference on EUV, X-Ray and Gamma-Ray Instrumentation for Astronomy*, vol. 3765, no. July, pp. 771–776, 1999.

[6] X. Fernàndez and J. Amat, "Research on Small Fiducial Mark Use for Robotic Manipulation and Alignment of Opthalmic Lenses," *IEEE Conference on Emerging Technologies and Factory Automation*, pp. 1143–1146, 1999.

[7] C. B. Bose and I. Amir, "Design of Fiducials for Accurate Registration Using Machine Vision," *IEEE Transactions on Pattern Anaylsis and Machine Intelligence*, vol. 12, no. 12, pp. 1196–1200, 1990.

[8] M. Tichem and M. Cohen, "Sub-micrometer Registration of Fiducial Marks Using Machine Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 8, pp. 6–9, 1994.

[9] H. Moon, R. Chellappa, and A. Rosenfeld, "Optimal Edge-Based Shape Detection." *IEEE Transactions on Image Processing*, vol. 11, no. 11, pp. 1209–1226, 2002.

[10] R. C. Gonzalez and R. E. Woods, *Digial Image Processing*. Upper Saddle River, NJ: Prentice Hall, 2008.

[11] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct Least Square Fitting of Ellipses," *Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 476–480, 1999.

[12] W. Gander, G. H. Golub, and R. Strebel, "Least-squares fitting of circles and ellipses," *BIT*, vol. 34, no. 4, pp. 558–578, Dec. 1994.

[13] I. D. Coope, "Circle Fitting by Linear and Nonlinear Least Squares," *Journal of Optimization Theory and Applications*, vol. 76, no. 2, pp. 381–388, Feb. 1993.

[14] D. Umbach and K. N. Jones, "A Few Methods for Fitting Circles to Data," *IEEE Transactions on Instrumentation and Measurement*, vol. 52, no. 6, pp. 1881–1885, 2003.

[15] B. F. Alexander and K. C. Ng, "Elimination of Systematic Error in Subpixel Accuracy Centroid Estimation," *Optical Engineering*, vol. 30, no. 9, pp. 1320–1331, 1991.

[16] S. S. Welch, "Effects of Window Size and Shape on Accuracy of Subpixel Centroid Estimation of Target Images," *NASA Technical Paper*, no. 3331, 1993.

[17] S. Lee, "Pointing Accuracy Improvement using Model-Based Noise Reduction Method," *Proceedings of SPIE*, pp. 65–71, Apr. 2002.

[18] D. K. Naidu and R. B. Fisher, "A Comparative Analysis of Algorithms for Determining the Peak Position of a Stripe to Sub-pixel Accuracy," *Procedings of the British Machine Vision Conference 1991*, pp. 28.1–28.9, 1991.

[19] Q. Tian and M. N. Huhns, "Algorithms for Subpixel Registration," *Computer Vision, Graphics, and Image Processing*, vol. 35, pp. 220–233, 1986.

[20] S. S. Gleason, M. A. Hunt, and W. B. Jatko, "Subpixel measurement of image features based on paraboiloid surface fit," vol. 1386, pp. 135–144, 1990.

[21] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, UK: Cambridge University Press, 2000.

[22] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," *New Results and Trends in Computer Science*, vol. 555, pp. 359–370, 1991.

[23] D. B. Leviton, T. Anderjaska, J. Badger, T. Capon,

C. Davis, B. Dicks, W. Eichhorn, M. Garza, C. Guishard, S. Haghani, C. Hakun, P. Haney, D. Happs, L. Hovmand, M. Kadari, J. Kirk, R. Nyquist, F. D. Robinson, J. Sullivan, and E. Wilson, "Cryogenic optical position encoders for mechanisms in the JWST optical telescope element simulator (OSIM)," *Proceedings of SPIE*, vol. 8863, Sep. 2013.

BIOGRAPHY

Alex Cramer *received B.S. degrees in mathematics and electrical engineering from UMD in 2009, and an M.S. degree from the same in electrical engineering in 2014, with a focus on computer vision. He is currently an Electrical Engineer at NASA Goddard Space Flight center in the Electromechanical Systems branch. He worked as the PYAS software engineer on the HEROES mission.*