www.DeepakPublishing.com

McBryde, R. et al. (2016): JoSS, Vol. 5, No. 1 pp. 435–448
(Peer-reviewed article available at www.jossonline.com)

www. JoSSonline.com

End-to-End Testing of a Dual Use Imaging Sensor for Small Satellites

Christopher R. McBryde and E. Glenn Lightsey

School of Aerospace Engineering, Georgia Institute of Technology
Atlanta, Georgia, USA

Abstract

In this research, a methodology has been developed for on-the-ground testing of visual navigation sensors for use on small satellites such as CubeSats. This process consists of a series of tests, including simulation, software-in-the-loop testing, and hardware-in-the-loop validation. Simulation consists of generating virtual images using specified camera and satellite parameters. Software-in-the-loop testing processes these virtual images through the visual navigation algorithms and verifies the results. Hardware-in-the-loop validation is performed using the same tests with actual images captured in situations analogous to those expected on orbit. The tests are intended to demonstrate both far field (e.g. star camera) and near field (e.g. proximity navigation) functions of dual use imaging sensors. Collectively, the analysis and testing suite creates a relatively simple validation process for imaging sensors, such as those primarily intended for use on small satellites.

1. Introduction

1.1. Role of imaging in small satellites

Low-cost, reliable imaging systems are vital for future small satellite applications. Imaging systems are a key component in proximity operations, a necessary capability for commonly proposed small satellite missions involving constellations or rendezvous with a target spacecraft. Star tracking algorithms provide precision attitude determination in sunlight or eclipse, facilitating operations like targeted imaging or high data rate radio communication. Reliable imaging on small satellites greatly expands the mission types that they can perform and needs to be accessible to small satellite providers who do not have the large budget or staff of larger satellite teams.

1.2. Proximity operations

Technical advances in vision processing systems are enabling new capabilities in spacecraft autonomous visual navigation. The role of the infrared camera on the SpaceX Dragon capsule autonomous docking system on its resupply mission to the International Space Station in 2012 showed that vision systems can be used for autonomous proximity sensing and relative navigation. The potential applications of this technology include important operations: satellite formation flying, autonomous rendezvous and docking, coordinated relative control, and on-orbit inspection, servicing, and assembly. Moreover, an autonomous object detection system can be used for space debris hazard mitigation and collision avoidance. Advances in imag-

Corresponding Author: Christopher McBryde, mcbryde@gatech.edu
Publication History: Submitted – 03/14/15; Revision Accepted – 01/25/16; Published – 02/19/16

ing systems and their associated electronics are making vision-based navigation systems less expensive and physically smaller than ever before, enabling their use on low-cost small spacecraft such as CubeSats.

Several new algorithms for vision-based measurement processing have been developed for applications outside of aerospace engineering that are very promising in terms of their robustness and simplicity. For example, the "blobber" algorithm presented by Walker determines the center of brightness (COB) of a body, which for a convex object is close to its geometric center (Walker and Spencer, 2012). The algorithm allows useful information to be extracted from an image even under poor optical conditions, including limited lighting and focus. Modeling and tracking algorithms can also be drawn from outside the aerospace field. The Learning Switching Dynamic Models for Object Tracking (Celeux et al., 2004) is an example of such a process. These new algorithms can be combined with traditional aerospace pose navigation algorithms, which require that the pattern of the object must be identifiable and known in the target object frame of reference (Haralick et al., 1989; Tweddle, 2010)

1.1. Star tracking

For small satellites to be viable as scientific and technological platforms, they must be capable of accurate attitude determination. In other words, the satellite must be able to determine its orientation well enough to ensure mission success. The most direct way to accomplish that goal is via star trackers. However, traditional star trackers are large and contain light baffles that would exceed the limited volume of many small satellites, including CubeSats; miniature star trackers that can fit within a CubeSat structure are a fairly recent development. For example, commercial devices have emerged that fit within the CubeSat form factor (Enright et al., 2010). Huffman (Huffman et al., 2006) presented a thorough survey of available star tracking algorithms, but they were not implemented in hardware. Pong has presented an algorithm for using the camera from ExoplanetSat for star tracking (Pong et al., 2012).

1.2. Contributions

Clearly, advances in sensor hardware and processing algorithms and technology have expanded the potential for imaging applications on small satellites. However, as important as these capabilities are, it is equally important that they perform as expected during their mission. Hardware and software must operate successfully during the device's first and all subsequent uses on-orbit. Thus, simulation, software-in-the-loop testing, and hardware-in-the-loop testing are critical for small satellites providers to prove the performance of their devices prior to flight.

This research presents a two-fold solution for these satellite developers. First, it demonstrates performance of a low-cost vision sensor that can be used both for proximity operations and star tracker-based attitude determination. This dual utility is valuable since it provides more functionality within the same mass, volume, and power costs of a single sensor. Secondly, procedures are presented to test the performance of this device in three pre-launch program stages: simulation of images using provided camera parameters, software-in-the-loop testing, and hardware-in-the-loop testing. Using the presented sensor as an example, the simulation and testing procedures provide a methodology for verifying the capabilities of imaging sensors in a lower budget small satellite program. Algorithms and test procedures will be presented first for proximity operations and then for star tracking.

2. Proximity Operations

2.1. Algorithm

To provide relative position information in the vicinity of another, non-cooperative body, the imaging sensor employs the blobber algorithm, developed by Walker (Walker and Spencer, 2012). This algorithm acquires relative position information in a two-stage process. First, a unit vector to the second body is determined by locating the center of brightness on the image plane. Then, using 2-D image coordinates and the camera geometry, a 3-D unit vector approximately directed toward the center of the body is determined in the imaging vehicle's body-fixed reference frame.

The algorithm then estimates the range to the object. This stage involves several steps. First, the major and minor axes of the blob are determined, as well as the ratio of the lengths of these axes, called the axis ratio. Next, estimates for the maximum and minimum projected areas are found using numerical methods. The projected area, known as A/A_0, is the ratio of the area viewed on the two-dimensional image plane to the minimum such area. For example, A_0 for a 3U CubeSat would be the area of its smallest face, 10 cm^2. A distribution of possible projected areas versus the axis ratio is found using randomly generated orientations of the target object. An example distribution for a 3U CubeSat is given in Figure 1.

A maximum and minimum area curve is determined from this distribution of points, from which an estimated range can be found via Equation 1 (Walker and Spencer, 2012).

$$\rho_{mean} = f\sqrt{\frac{A_{mean}}{N_{blob} \cdot p^2}} \tag{1}$$

In this equation, f is focal length of the camera, A_{mean} is the average projected area for a given ratio of axes, N_{blob} is number of pixels the imaged object illuminates, and p^2 is the physical area of one pixel on the sensor.

2.2. Simulation

The first step in performing end-to-end testing of the proximity operations sensor is the generation of simulated image measurements. The primary purpose of these simulated images is for use in the later software-in-the-loop testing. However, they are also important for algorithm validation. The accuracy and utility of the software-in-the-loop tests can be determined later by comparing the simulated images to actual images taken with the hardware.

2.2.1. Types of reflectance

Reflected light from an object is broken into three categories: ambient, diffuse, and specular (Lekner, 1987). Ambient reflected light is due to a light source that is sufficiently scattered so that its direction of origin is unknown. An example of this type of reflection is sunlight on a cloudy day. The surroundings are clearly being illuminated, but the clouds scatter the

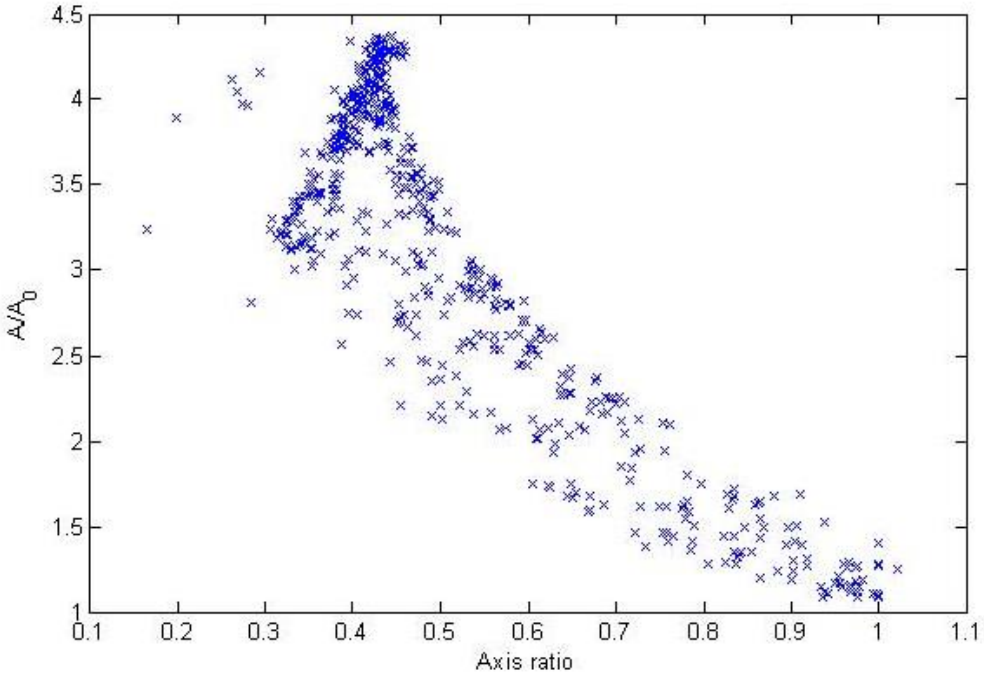

Figure 1.Example of projected area distribution for a 3U CubeSat.

sunlight so that no shadows exist from which to determine the sun's direction. The next type of reflection is diffuse, or Lambertian. Diffuse reflection results from a directional light source striking a matte, or rough, surface. Diffuse reflection is scattered in all directions, with a greater reflected intensity the smaller the angle is between the light source and the normal direction of the surface. The last type of reflection is specular. Specular reflection results in a "glossy" appearance and follows Snell's law: the angle of incidence equals the angle of reflection. A sketch of the difference between Lambertian and specular reflection is given in Figure 2.

2.2.2. Primitive shapes

The first step in the generation of simulated images is the definition of primitive shapes. These building blocks are combined to form the optical target as well as the imaging satellite. The shapes that were selected for this step were cuboid, sphere, and cylinder. A generation function creates a list of points making up the surface of the shape given the dimensions, surface resolution, relative position, and relative orientation. The surface resolution is defined as the number of points per unit surface area of the primitive shape. Each of these points is then assigned a normal direction for use in calculating the interaction of the body with simulated light sources. The point list combined with the associated normal directions is called the point cloud.

The first primitive shape selected is the rectangular cuboid, referred to hereafter as "cuboid." Cuboids are defined as convex polyhedra bounded by six rectangular faces. This shape is common in satellite design, forming the basis of components like solar panels or a CubeSat. Since the cuboid faces consist of rectangles, the parameter of interest for generating the surface is the pixel pitch, or number of pixels per unit length. That value is determined from the surface resolution by taking its square root. The edge points of the cuboid are evenly distributed on each of the 12 edges with a distance between them equal to the pixel pitch. These edges are assigned a normal direction of zero so that they reflect only ambient radiation. The absence of Lambertian or specular radiation represents the discontinuity between the normal directions of each face. Next, the points on each of the eight faces are generated. One coordinate direction is held constant and the points are evenly distributed, separated in the other two coordinate directions by a distance equal to the pixel pitch. Each face is assigned a normal of 1 or −1 in the direction that was held constant during the loop, depending on location (left/right, top/bottom, or front/back). Finally, the point cloud locations and normal directions are rotated and translated based on the relative position and translation information.

The second primitive shape is a sphere. The distribution of points evenly across a sphere is not as trivial as for a cuboid. Deserno (Deserno, 2004) presents two potential remedies for this unequal distribution. The

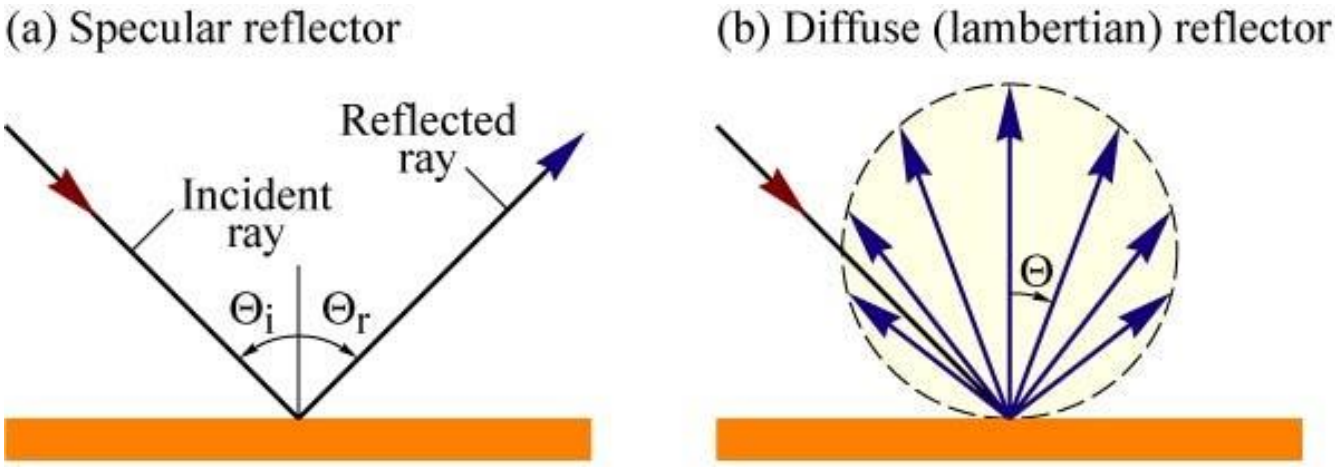

Figure 2.Specular versus Lambertian reflection (Schubert, 2006).

first is to randomly distribute a given number of points over the surface area. The second is to choose circles of latitude d_θ and distribute on them points along d_φ, such that the area per point $d_\theta d_\varphi$ remains approximately constant. The second is the strategy that was implemented. While the point distribution algorithm is more complicated, the normal direction calculation for each point is simpler: it is the unit vector to the point's location relative to the geometric center. Similarly to the cuboid process, the point cloud locations and normal directions are then rotated and translated.

The last primitive shape is a cylinder. Generating the point cloud for the cylinder uses a combination of the cuboid and sphere processes. The points for the top and bottom of the cylinder follow from a two-dimensional version of the algorithm from Deserno (2004). The normal directions are simply positive-and negative-z unit vectors. The body points of the cylinder are generated by "laying out" the face flat and creating a grid of points on that face. The normal directions for the face are found using the unit vectors to each point similarly to the sphere. Finally, the points are rotated and translated. Figure 3 shows all three primitive images.

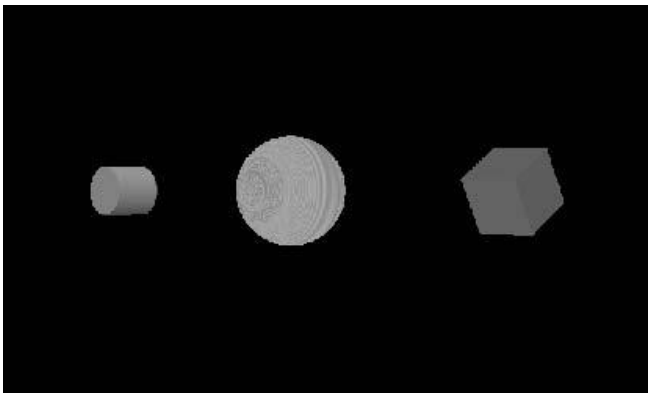

Figure 3. Examples of generated images (left: cylinder, center: sphere, right: cuboid).

2.2.3. Satellite configuration file

Once the algorithms have been developed to create the point clouds for the various primitive shapes, a method is needed to combine them into more complex shapes, as well as describe their motion both relative to space and relative to the other primitive shapes that comprise the object. This step is accomplished via a configuration file. The text file consists of three parts. The first two lines describe the camera geometry, including resolution, pixel size, and other parameters, as well as the camera's initial location in three dimensional space. The first line also specifies the direction of the light source. The third line describes the camera's motion, if any, in inertial space. The next part describes the motion of the imaging satellite overall. This information includes the number of primitive shapes that describe the body, the body's initial location and the subsequent motion of the body. Finally, the last part describes the primitive shapes themselves: their shapes, sizes, optical characteristics and relative position and motion to the camera-centered frame. There can be any number of components, indicated by a parameter in the body section.

2.2.4. Simulated images generation

Once the configuration, motion, and optical characteristics of the satellite have been specified, a series of simulated images is generated. These images can be combined into a video to ascertain data about the velocity of the object. First, all of the pertinent information is extracted from the configuration file. These parameters determine the number of frames of the series that must be generated. For each frame, the relative position of the target object to the camera as well as its visual parameters are passed into an image generation subfunction. This subfunction determines which points on the satellite are visible to the camera, eliminating points that are obscured either by another part of the same primitive shape or by another shape in the satellite. For each of these visible points, the reflected light is determined by the point's normal direction, the direction to the light source, and the direction to the camera. Each material reflects light in a ratio of ambient, Lambertian, and specular reflection given by a coefficients k_a, k_l, and k_s. The other necessary parameters are I_a the amount of ambient light near the body, which is an estimate and can be ignored if all light sources are directional, and I_i the intensity of the of the light source. The vector N is the normal

direction for a point on the surface of the object as generated in section 2.2.2. The unit vector L points to the light source and unit vector V points to the observer, in this case the simulated camera. Both of these vectors originate at the point on the surface. The vector R is an intermediate vector necessary for the calculation of specular reflection. The dot product of R with V is defined by Equation 2. These vectors are shown in Figure 4. The specular exponent n relates to the apparent smoothness of the object. These parameters are combined to give the illumination of a point by the Equation 3.

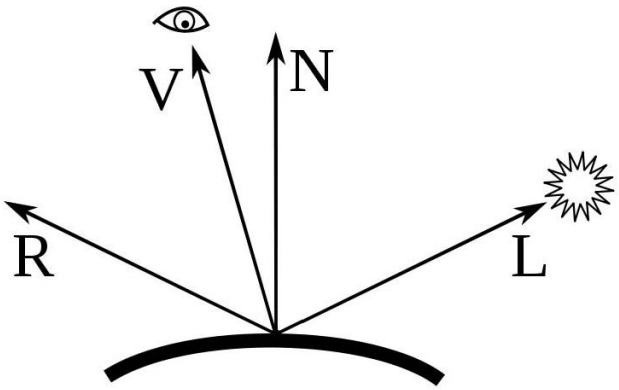

Figure 4. Reflection model vectors.

$$\mathrm{R}\cdot\mathrm{V}=2(\mathrm{N}\cdot\mathrm{V})(\mathrm{N}\cdot\mathrm{L})-(\mathrm{V}\cdot\mathrm{L}) \quad (2)$$

$$I = I_a k_a + I_i(k_l(\mathrm{N}\cdot\mathrm{L})+k_s(\mathrm{R}\cdot\mathrm{V})^n) \quad (3)$$

2.3. Software-in-the-loop testing

To test the proximity operations algorithm, a software-in-the-loop test is used. Images are simulated using given parameters and then these simulated images are passed through the algorithm. Once that is completed, the results are compared to a truth value.

The satellite and camera parameters are application specific, but to make the test representative of typical low-cost small satellite applications, the following parameters were used. For the satellite, a 3U CubeSat sized object was generated, having dimensions of 10 by 10 by 30 centimeters. For the camera, the parameters for a real sensor that is planned for use on an upcoming CubeSat mission are used, given in Table 1.

Table 1. Simulated Camera Parameters

Resolution (px)	1024×768
Pixel size (μm)	4.65×4.65
Focal Length (mm)	6

It is important when selecting hardware to consider the maximum and minimum resolvable range. These values are based on the pixel size μ, focal length f, and characteristic dimension of the object h. For this sensor hardware and target object, the theoretical resolvable range R is between 8.4 cm and 64 m. These values are the distances at which the object fills the entire frame and at which the object covers four square pixels. Resolvable ranges can be found by constructing similar triangles, one using the focal length and image sensor and one using the range and object size (Figure 5). The thresholds on either extreme can be adjusted based on sensing requirements and application.

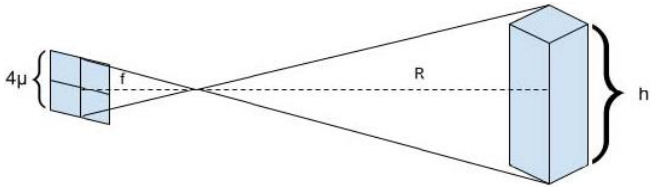

Figure 5. Resolvable range calculation.

Once the hardware was specified, a path was chosen for the motion of the satellite. The path was designed to extend over various ranges as well as orientations of the satellite. These parameters are given in Table 2.

Table 2. Satellite Motion Parameters

State	x	y	z
Velocity (m·frame^{-1})	0.02	0.02	0.4
Rotation (deg·frame^{-1})	5	5	5

A simulated movie is then generated from the camera and satellite parameters as well as the specified relative motion. The movie was then passed into the proximity operations algorithm, which found the relative position of the body in each frame. The simulated movie is frame-rate independent. In some tests, a

specified frame rate could be necessary. For example, velocity may need to be estimated, as well as position. In that case, the velocity and rotation parameters should be calculated so that those values agree with the desired relative motion and frame rate of the camera. These results are compared with the true motion in Figure 6.

The results show that the average tracking of the satellite is quite good. There is some variation around frames 16 through 27, which can be explained by the changing orientation of the target satellite. The blobber algorithm determines an estimate for the range based on the projected area and the ratio of axes. As can be seen in Figure 1, multiple projected ranges can be associated with a given axis ratio, hence the error. Additionally, a greater error exists in the z-direction, since that is the direction of the camera boresight. Thus, an error in the range will show up most prevalently in the z-axis instead of the x-or y-axis. Nevertheless, the algorithm tracks the satellite quite well, considering the limited image quality and knowledge of the object being tracked, especially at the shorter ranges.

To test the blobber algorithm more rigorously, a second software-in-the-loop test was performed. For this test, 11 ranges were tested under 9 random orientations. To more accurately determine the performance of the algorithm, the ranges were distributed logarithmically from 1 m to 100 m. In total, 99 different orientations were tested. The results are given in Figure 7. The second test reinforces the results of the first. The algorithm performs well at shorter ranges, but begins to underestimate the range starting at around 10 m. The average results for the orientations at each range are given Table 3.

This analysis backs up the general trend in Figures 6 and 7. The blobber algorithm performance degrades as the range increases. A likely explanation for this phenomenon is that the smaller the object becomes in the image plane, the fewer pixels it illuminates. That, in turn, worsens the axis ratio estimate upon which the projected area and range estimates are based. From a higher-level perspective, this feedback is valuable to satellite developers before launch and highlights the utility of simulation and software-in-the-loop testing.

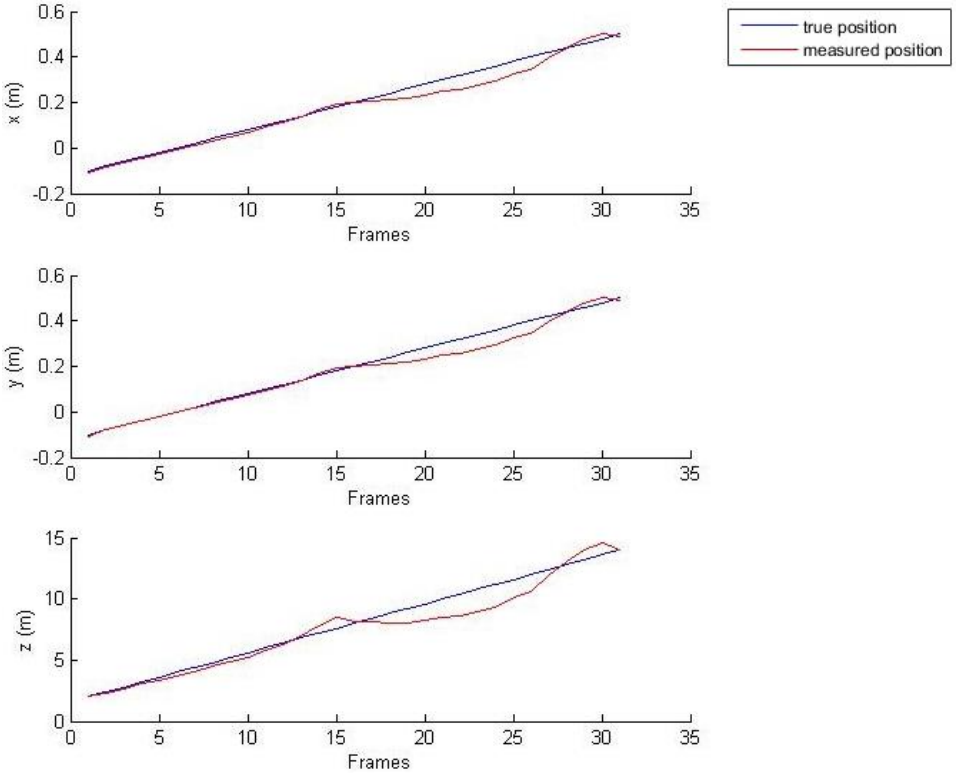

Figure 6. Software-in-the-loop results.

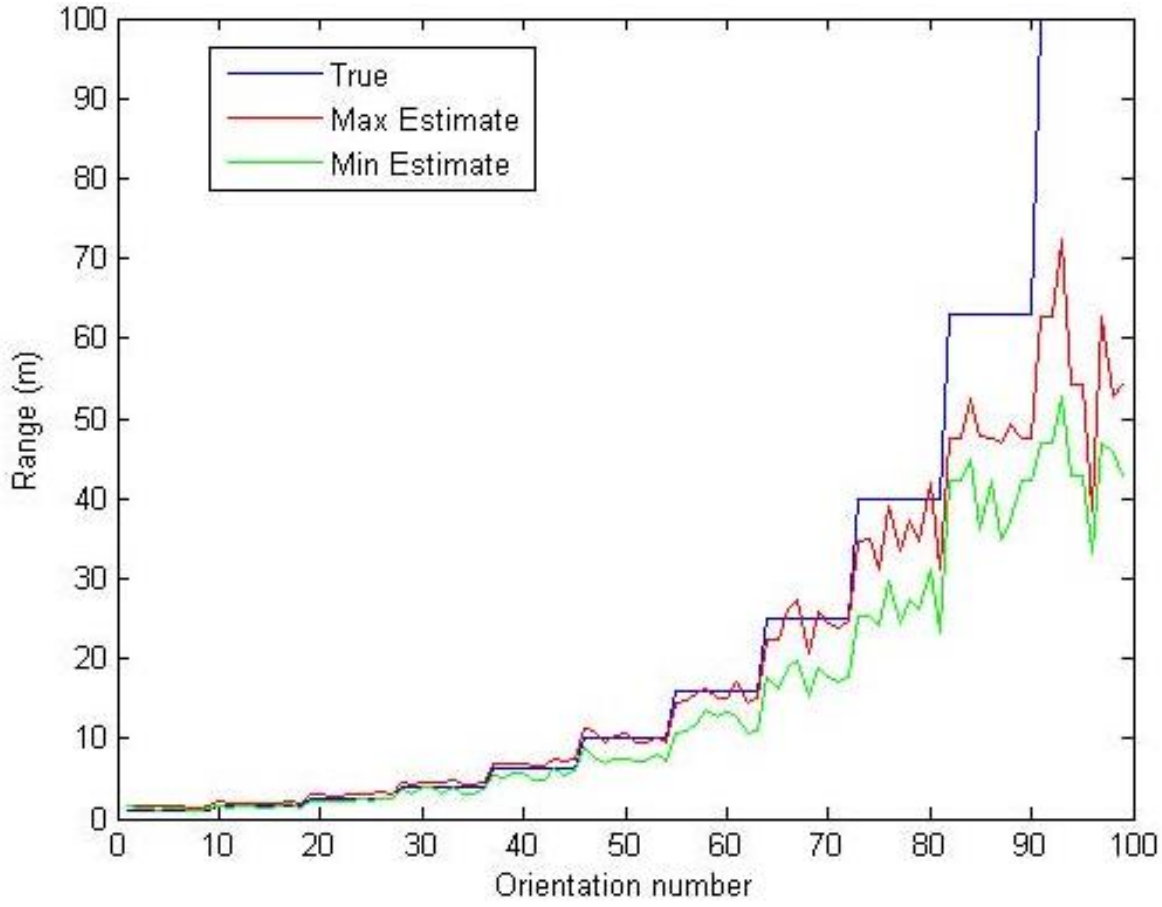

Figure 7. True range (blue) compared with maximum (red) and minimum (green) estimated ranges.

Table 3. Algorithm Performance by True Range

True range (m)	Avg est range (m)	Avg abs error (m)	Avg rel error (%)	True ranges within min-max (%)
1.000	1.392	0.3921	39.21	0
1.585	1.738	0.1840	11.61	77.78
2.512	2.537	0.1961	7.808	88.89
3.981	3.837	0.3991	10.03	77.78
6.310	5.951	0.4992	7.912	88.89
10.00	8.685	1.345	13.45	44.44
15.85	13.11	2.734	17.25	22.22
25.12	19.99	5.131	20.43	33.33
39.81	30.15	9.657	24.26	11.11
63.10	42.40	20.69	32.79	0
100.0	50.50	49.50	49.50	0

2.4. Hardware-in-the-loop validation

In order to perform a hardware-in-the-loop validation for the proximity operations algorithm, the following process was used. First, a 3U CubeSat analog was constructed with the dimensions of 10x10x30 cm. Next, a camera was analyzed using the process presented by Bouguet (2013) to determine the exact focal length. This value was combined with the camera geometry to yield the following key parameters given in Table 4.

Table 4. Camera Parameters

Resolution (px)	2048 × 1536
Pixel size (μm)	2.9475 × 2.9475
Focal Length (mm)	4.356

Next, a series of images was taken of the satellite analog. To determine the performance of the blobber algorithm at different orientations, all of the images were taken with the analog at a distance of one meter. These images were taken under different lighting conditions and different orientations. The images were then processed using the blobber algorithm described in section 2.1 after undergoing various image processes. First, the images were inverted, since the analog was black on a light background as opposed to the simulated images, which were light on a dark background.

These images were then converted from red-green-blue color to binary black-white using a thresholding parameter. If any of the color channels met or exceeded the threshold, the pixel was considered a 1; otherwise, it became a 0. Since this camera used an 8-bit sensor, the maximum brightness value was 255 and the threshold parameters were chosen to be near the higher end of the range 0-255, to focus on the foreground object.

Figure 8 shows the range results for various thresholding parameters. The error bars represent the maximum and minimum range generated by the blobber algorithm. Overall, the hardware-in-the-loop provides reasonable verification of the blobber algorithm using physical measurements. The average residual is 8.14 cm, or an error of 8.14%, for the best performing

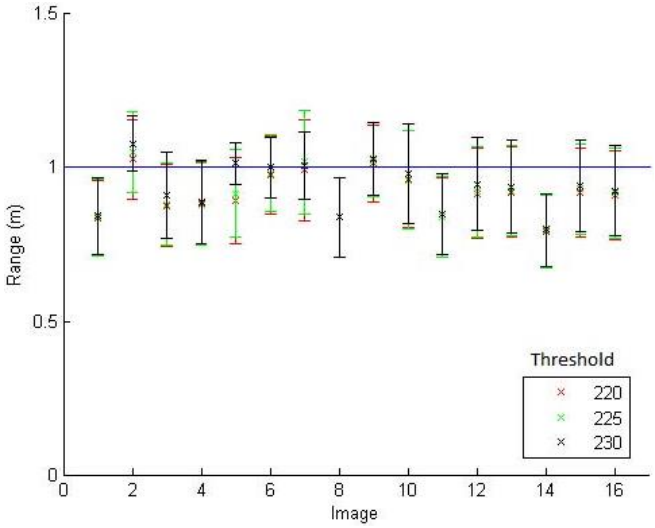

Figure 8. Range results for various threshold parameters.

threshold of 230. In addition, the true range falls within the error bars for 75% of the images. In an effort to improve the image performance, the thresholded images were opened and closed. Opening an image is a processing technique in which the image is eroded and then dilated and is used to eliminate small foreground objects. Closing is the inverse process, when an image is dilated and then eroded. This technique closes holes in foreground objects. A more detailed description of both processes can be found in Haralick's work (Haralick and Shapiro, 1992). Opening and closing the image did not provide any improvement to the performance, and the results were very similar to Figure 8. The projected area is determined from the number of foreground pixels, and since a level of uncertainly already exists relating the projected area to the axis ratio, the performance is not greatly improved or reduced by a process which more accurately accounts for the true number of foreground pixels. The overall result provides an on-the-ground verification of the blobber algorithm analogous to the demonstration of the star tracking algorithm described in section 3.4.

3. Star Tracking

3.1. Algorithm

The star tracking algorithm employed in this example is known as the voting method and is described

by Kolomenkin, et al. (2008). Each potential star pair is compared to an on-board catalog of star pairs and possible identities for each potential star pair are identified. When the list of star pairs for an image has been evaluated, the most frequently chosen catalog star is assigned to its respective candidate star in the image. The proposed angular distances as determined by the catalog are then compared to the actual distances, and if they match to within a given tolerance, those two identities receive a "vote." At the end of the process, the identities with a high number of votes are confirmed. Their unit vectors in the inertial frame are then passed along with the sensor frame unit vectors to the attitude determination algorithm. That algorithm was the method outlined by Markley (1988), which finds the attitude via a singular value decomposition of a 3x3 matrix constructed from the vector representations in the observed and reference frames.

The voting method was chosen because it is extremely robust to poor quality images that can produce measurement artifacts known as false stars. In tests by Kolomenkin et al. (2008), the voting method retained the correctly identified stars until the number of false stars was as much as three times the number of actual stars, and it accomplished this without restarting the algorithm. Also, the lower resolution performance of inexpensive imaging sensors does not significantly affect the accuracy of the voting method. Using defocusing and centroiding, the sensor attains subpixel accuracy. In terms of memory requirements, a catalog of star pairs used with the voting method is about the same as those for other star identification methods, with some other methods needing more memory for star triples or even grids of potential images. Considering these criteria of robustness, resolution, and memory requirements, the voting method was chosen for star tracking on small satellites.

3.2. Simulation

To verify functionality of the attitude determination capability of the same sensor, a simulation was conducted to demonstrate the operation of the star tracking algorithm. The setup for the analysis consisted of picking a random attitude to simulate and generating the corresponding star field image that would be observed by the camera pointed in that attitude. The attitude here is defined as orientation of the camera relative to the Earth-centered inertial frame. To select the attitude, three random angles were selected between −180 degrees and 180 degrees. These angles were used in a 1-2-3 rotation sequence to generate an attitude direction cosine matrix. This attitude was combined with the camera parameters and a star catalog to create the simulated image. Further details on the simulation process can be found in McBryde (2012). This process was repeated a total of 200 times, resulting in a series of 200 randomly generated simulated images.

3.3. Software-in-the-loop testing

These images were then processed by the star tracker software, which outputs an estimated attitude solution according to the algorithm specified in section 3.1. The true and estimated attitudes were compared to find the accuracy of the star tracker measurement. The error about each axis was then calculated as shown in Eqs. 4, 5, 6, and 7.

$$\Theta = \mathrm{R}_{calc}^{-1}\mathrm{R}_{actual} = \begin{bmatrix} \approx 1 & -\phi_z & \phi_y \\ \phi_z & \approx 1 & -\phi_x \\ -\phi_y & \phi_x & \approx 1 \end{bmatrix} \quad (4)$$

$$\epsilon_x = 90° - \cos^{-1}\phi_x \quad (5)$$

$$\epsilon_y = 90° - \cos^{-1}\phi_y \quad (6)$$

$$\epsilon_z = 90° - \cos^{-1}\phi_z \quad (7)$$

The attitude error between the estimated attitude and simulated true attitude was calculated for each axis for each of 200 randomly generated images. The algorithm was considered to have failed if the error was greater than 100 arcseconds for any of the three axes. Since the target accuracy for this system is arcminute level (single digit arcminutes), 100 arcseconds is on the low end of this range in order to account for real-world factors that might affect the accuracy. Given these conditions, the star tracking algorithm successfully found the correct simulated attitude

within an acceptable error 191 out of 200 times, for a reliability of 95.5%.

Looking first at the failures, only twice out of 200 times, or 1%, did the algorithm report an incorrect attitude for which it reported obtaining a confident match. The match confidence is determined by the value T, called the match threshold. The match threshold is the maximum number of votes received by a star in the verification step of the voting method. If $T \leq 0$, then no star received any votes, indicating a poor match. On orbit, this would be output as an error condition and the attitude solution would not be used. Only two reported attitude solutions were incorrect despite having $T > 0$. These cases could present a problem, since they would be reported by the algorithm as a correct result. After visual analysis of both failure cases, the simulated images have at least one star pair in very close proximity, which fooled the algorithm into thinking there was only one star and caused the failed identification. Further refinement of the star catalog or algorithm could be used to counteract this case if necessary, but since it represents only 1% of the total test cases, the algorithm may be acceptable as is, depending on the sensor requirements.

For the 191 successful cases, the average performance can be seen in Table 5. Theoretically, performance should improve when more stars are observed.

Table 5. Average Successful Case Performance

Stars generated	11.4607
Stars observed	10.1152
T	8.6230
ϵ_x (arcsec)	10.6237
ϵ_y (arcsec)	7.7998
ϵ_z (arcsec)	6.4789

Figure 9 shows the error for each axis averaged over the number of stars observed. The image generation process is computationally intensive, but could be augmented with additional computational power to run more than 200 cases for better statistical results.

The cases where fewer stars are observed do have the highest errors for the x-and y-axes, and almost

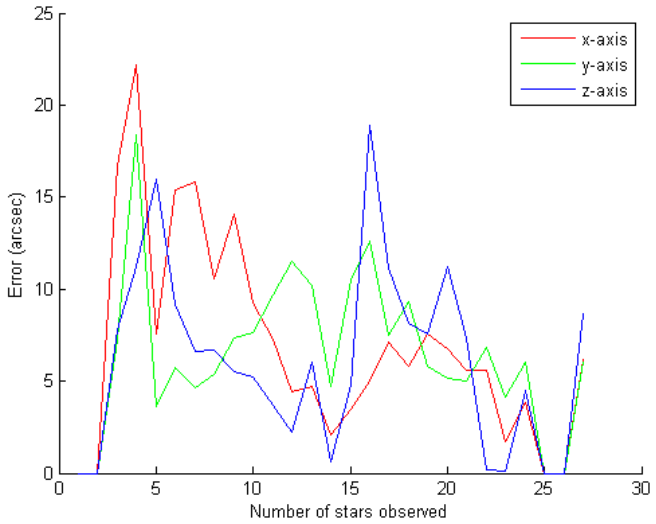

Figure 9. Error in each axis vs. number of stars observed.

highest for the z-axis, but there appears to be no consistent downward correlation as the number of stars observed is increased. This observation is verified by an root-sum-square analysis of the error across the three axes, shown in Figure 10. The phenomenon could be related to the star clustering error described in the failed identification cases.

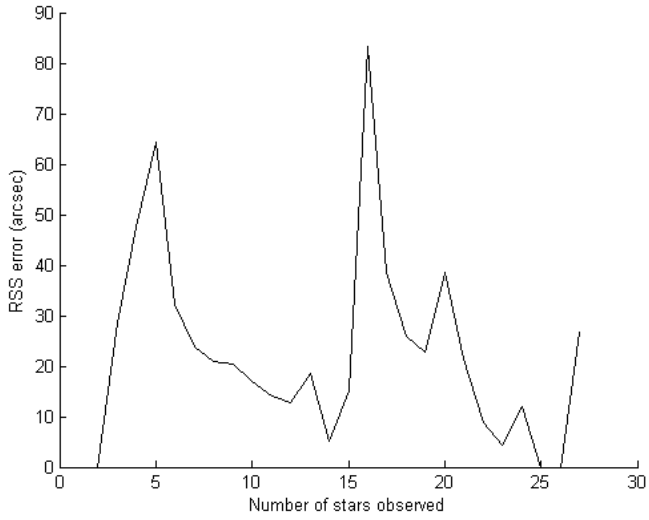

Figure 10. RSS error vs. number of stars observed.

The graphs from Figures 9 and 10 highlight some interesting results. First, even though all physical error has been removed from the test, the errors in the three

axes are not zero. Looking through the process for potential error sources, the only likely contributor is the centroiding algorithm. The algorithm uses a slightly defocused image and measurement centroiding to achieve subpixel accuracy in the estimate of each star's location in the image. Despite achieving subpixel accuracy, the centroiding function still has some error from the true location of the center of the star. Errors as small as 10 arcseconds should not concern most low-cost small satellite developers looking to add star tracking to their satellites unless their mission requires more accuracy.

Another noteworthy metric is the average number of stars viewed, which was about ten. Even for a 20 degree FOV, this number is somewhat high and is the result of the brightness mask on the stars being set to magnitude 5.5. This choice was made to ensure that enough stars existed in each image for analysis. The purpose of the computer-generated images was not to assess the real world performance of the star tracker, given all of the ideal assumptions that were made. Rather, it was to verify the functionality of the algorithm. Further tests were conducted to determine physical hardware performance in the presence of additional measurement error sources.

3.4. Hardware-in-the-loop validation

To conduct hardware-in-the-loop night sky testing, an experimental setup was created that could be used outside the lab. The physical setup of the test is as follows. A Matrix Vision mvBlueFOX-M121G camera and Schneider Optics lens were fixed to a camera tripod mount which was then attached to a high-quality tripod. It was important to make sure that the tripod was level, to enable the calculation of the true attitude. The tripod camera setup was taken to two different dark sky locations in Texas on two different nights. In addition to the Schneider Optics lens, two other lenses were used to take test images: the Edmund Optics 12 mm lens and the Navitar 6 mm lens. These lenses have a wider field-of-view and can view more bright stars.

The software setup was as follows. The camera was connected via a USB cable to a laptop running the mvImpact ACQUIRE software provided by Matrix Vision. The program used was called wxPropView and allowed the viewing of the camera output as well as saving the resulting image. In addition, a GPS receiver was used to determine the latitude and longitude of the location of the camera and the time of the image. This information was used to calculate the true attitude of the camera at the time of the image.

The results of the night sky testing were promising, but incomplete. The best success was on a set of calibrated images taken with the Navitar 6 mm lens. This image was run through the star identification process. The stars were all correctly identified as compared to the listing of star names in the constellation. The next step was to calculate the attitude and then apply the transformation to the unit vectors in the inertial coordinate frame. The resulting unit vectors should match the unit vectors derived from the image, which are given in the camera coordinate frame. When this process was performed on the image set, the result was quite good for some of the images. Figure 11 shows the predicted and actual observed positions overlaid on

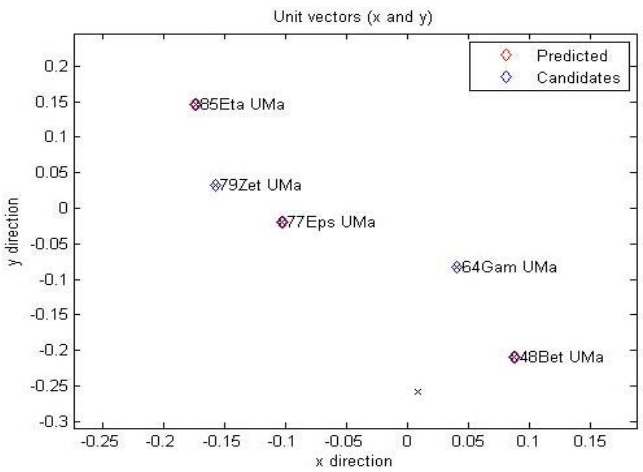

Figure 11. Predicted and candidate unit vectors using star identification results

top of each other. For two of the stars, the discrepancy is small enough that it cannot be seen on the graph. Table 6 shows the error for each identified star as well as its magnitude. The identification numbers reference the Yale Bright Star Catalog (YBSC).

These results from the night sky testing show the viability of the star identification algorithm in a real-world situation. Not every image in the two sets

Table 6. Error and Magnitude for Each Star Observed in Figure 11.

YBSC ID	Name	Error (arcseconds)	Magnitude
5182	η-UMa	110.0	1.86
5045	ζ-UMa	26.8	2.27
4896	ϵ-UMa	59.8	1.77
4545	γ-UMa	58.8	2.44
4286	β-UMa	114.1	2.37

returned an attitude solution, and the likely cause results from the calibration of the camera system. The calibration parameters vary from camera system to camera system, and even from environment to environment. Further investigation will go into the cases without a solution to determine why a solution was not returned. In those cases, the algorithm indicated the lack of a solution. It is also important to note that the comparison for these tests was between the measured image of the star on the image and its projected location based on the attitude solution. A true attitude in an Earth-centered inertial frame was not calculated; if it were, errors such as GPS accuracy, timing accuracy, and tripod alignment and calibration would all be important factors. Similarly to the proximity operations tests, these results provide a useful proof of concept, but also demonstrate the importance of proper hardware calibration to obtain good results.

4. Conclusion

The methodology presented in this research demonstrates an approach for small satellite designers to perform faithful and accurate tests of their visual navigation sensors before launch. Used either individually or as a whole, the simulation, software-in-the-loop testing, and hardware-in-the-loop validation techniques lower the risk assessment for low-cost small satellite missions. In addition, the process shows the viability of a dual-use sensor on these missions. This combination has the advantage of saving volume, mass, and power by combining multiple utilities into a single sensor on missions when all three resources are at a premium.

4.1. Planned Demonstration on Satellites

The visual navigation system outlined in this research is in a unique position to be put to a practical test in space. The design is included on two separately developed student satellite missions. These 3U CubeSat missions are known as Bevo-2 and ARMADILLO (Atmospheric Measurement And Detection of sub-mILLimeter Objects), scheduled to fly by 2016. The parameters that were used in this study represented real sensors that are baselined for these missions to be used on-orbit.

4.2. Future work

There are still some remaining issues with obtaining the best results from the night sky testing, but these can be resolved with further testing and refinement of the star identification algorithm. One potential issue is the differing focus requirements for star tracking, which must take images of stars at near infinity, and imaging, which usually focuses on nearby objects. Due to the small focal length of the camera used for the tests, “near infinity” in this case is on the order of a few meters, so it should not drastically affect the imaging capability of the camera to image both stars and other objects that are tens of meters away. However, spacecraft providers who would like the increased attitude determination accuracy that comes with a longer focal length would have to consider the resulting increase in the minimum distance for proximity operations. On the proximity operations side, further work must be done in refining the hardware-in-the-¬loop validation. This work includes testing the camera at various ranges and the testing of different image processing techniques to improve the accuracy of the ranges.

Acknowledgments

The authors wish to acknowledge the work of the students in the Texas Spacecraft Laboratory on the satellites on which this dual use imaging sensor will fly. In particular, we would like to thank Andrew Fear,

Karl McDonald, and Brian O'Connor for their assistance. This work was sponsored in part by NASA contracts NNX09M51A and NNX15AD26H.

References

Bouguet, J. (2013): Camera Calibration Toolbox for MATLAB. Available at: http://www.vision.caltech.edu/ bouguetj/calib_doc/ (last accessed Feb. 19, 2015).

Celeux, G., Nascimento, J., and Marques, J. (2004): Learning Switching Dynamic Models for Objects Tracking. *Pattern Recognition*, Vol. 37, pp. 1841–1853.

Deserno, M. (2004): How to Generate Equidistributed Points on the Surface of a Sphere. Available at: http://www.cmu. edu/biolphys/deserno/pdf/ sphere_equi.pdf (last accessed Jul. 30, 2015).

Enright, J. et al. (2010): Towards Star Tracker Only Attitude Estimation, in Proc. *24th Annu. AIAA/USU Conf. on Small Satellites*, Logan, UT.

Haralick, R. et al. (1989): Pose Estimation from Corresponding Point Data. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19(6), pp. 1426–1446.

Haralick, R. and Shapiro, L. (1992): *Computer and Robot Vision*, Vol. 1. Boston: Addison-Wesley Publishing Co.

Huffman, K. et al. (2006): Designing Star Trackers to Meet Micro-satellite Requirements, in *SpaceOps 2006 Conf.*, Rome, Italy.

Kolomenkin, M. et al. (2008): Geometric Voting Algorithm for Star Trackers. *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 44(2), pp. 441–456.

Lekner, J. (1987): *Theory of Reflection, of Electromagnetic and Particle Waves*. Dordrecht, Netherlands: Springer.

Markley, F. (1988): Attitude Determination Using Vector Observations and the Singular Value Decomposition. *J. Astronaut. Sci.*, Vol. 38(3), pp. 245–258.

McBryde, C. (2012): A Star Tracker Design for CubeSats. Master's thesis, University of Texas at Austin.

Pong, C. et al. (2012): High-precision Pointing and Attitude Determination and Control on Exoplanetsat, in *AIAA Guidance, Navigation, and Control Conf.*, Minneapolis, MN.

Schubert, E. (2006): Light Emitting Diodes.org. Available at: http://www.ecse.rpi.edu/~schubert/ Light-Emitting-Diodes-dot-org (last accessed Oct. 4, 2015).

Tweddle, B. (2010): Computer Vision Based Navigation for Spacecraft Proximity Operations. Master's thesis, Massachusetts Institute of Technology.

Walker, L. and Spencer, D. (2012): Automated Proximity Operations Using Image-based Relative Navigation, in *26th Annu. USU/AIAA Conf. on Small Satellites*, Logan, UT.