Visual Vector Sensor Implementation for Tabletop Spacecraft Simulators

David T. Gerhardt[1] and Stephanie A. Goellner[2]
Virginia Tech, Blacksburg, VA 24060

Sarah Hefter[3]
Virginia Tech, Blacksburg, VA 24060

and

Laura L Jones[4]
Virginia Tech, Blacksburg, VA 24060

Methods for the development and implementation of a webcam-based visual attitude sensor are determined for use on a tabletop spacecraft simulator. The simulator, located in Virginia Tech's Space Systems Simulation Laboratory (SSSL), consists of a flat honeycomb platform that rests on a hemispherical air bearing, enabling the craft to fully rotate around its vertical axis and providing a ten degree range of motion about the other two axes. In order to determine the simulator's orientation relative to an inertial laboratory, an easily identifiable target image is fixed to the ceiling within the camera's field of view. The camera captures an image and the sensor attitude determination code is executed. This code processes the resulting data to first identify the target shape and then interpret those results to output a set of angles corresponding to the orientation of the simulator. This complete algorithm, which can be implemented regardless of the type of camera, is described in detail, and a discussion of preliminary stationary attitude results is presented.

Nomenclature

θ = rotation angle about $\hat{n}_3$

φ_x = tilt angle about $\hat{n}_1$

φ_y = tilt angle about $\hat{n}_2$

I. Introduction

OPERATIONAL spacecraft establish attitude using onboard sensors, which may include star trackers, sun sensors, or magnetometers. Spacecraft simulators provide a cost-effective method for studying the dynamic response of spacecraft. Simulators also require attitude determination sensors, but using the same space-qualified equipment is both unnecessary and cost prohibitive. The Visual Vector Sensor system developed in Virginia Tech's

[1] Undergraduate Student, Department of Aerospace Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, Student Members AIAA.
[2] Undergraduate Student, Department of Aerospace Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, Student Members AIAA.
[3] Undergraduate Student, Department of Aerospace Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, Student Members AIAA.
[4] Undergraduate Student, Department of Aerospace Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061, Student Members AIAA.

Space Systems Simulation Laboratory (SSSL) is a cost-effective attitude determination system that uses a generic webcam to visually determine the orientation of the simulator.

The Visual Vector Sensor system has five main components: a tabletop spacecraft simulator, a PC-104 flight computer, a target figure, a generic webcam, and C++ code that captures images and extracts the rotation and tilt angles. The centroid of the target figure (an acute isosceles triangle) is placed on the ceiling of the laboratory, directly above the center of rotation of the simulator, and the most acute angle of the triangle is pointed in the direction of the standard inertial x-axis used for all attitude determination systems on the simulator[2]. The code that controls the Visual Vector Sensor system instructs the camera to capture an image of the target triangle, and then implements a series of functions to identify the centroid and vertices of the triangle. The vertices are then sorted to determine which location corresponds to the most acute angle of the triangle. This information is passed to another set of functions that determine the orientation of the simulator and output three angles corresponding to the rotation around the three inertial axes. The following sections describe the equipment used in this research and the algorithm developed to process the image and determine the orientation angles of the simulator. A brief discussion of preliminary implementation and static results is then presented.

II. Research Equipment

A. Spacecraft Simulators and Simulation Hardware

The Distributed Spacecraft Attitude Control System Simulator (DSACSS) is used to study and develop distributed control laws for pointing and spacing of formation flying satellites. The DSACSS consists of three simulators, two that include software and hardware, and one that includes only software. The simulators with hardware float on air bearings to simulate a frictionless dynamic environment and are capable of hosting a payload of up to 300 lb[1]. The Visual Vector Sensor system is being implemented on one of these simulators, named Whorl-1 (shown in Figure 1), which is arranged in a tabletop configuration that provides full freedom of motion in yaw and +/- 5 degrees of motion in both pitch and roll[1].

Whorl-1 is currently equipped with a PC/104+ flight computer that houses a 733MHz processor, a flash drive with a Linux Fedora operating system, and two USB ports[1]. This simulator's attitude determination sensors include a three-axis accelerometer, rate gyros, and a magnetometer. The accelerometer can sense accelerations up to +/- 1.5g in each axis, and the rate gyro can sense rates up to +/- 75 degrees/second in each axis[1]. The Visual Vector Sensor will provide an independent measure of the orientation of the simulator and thus will enable a better informed estimate of the actual orientation of Whorl-1.

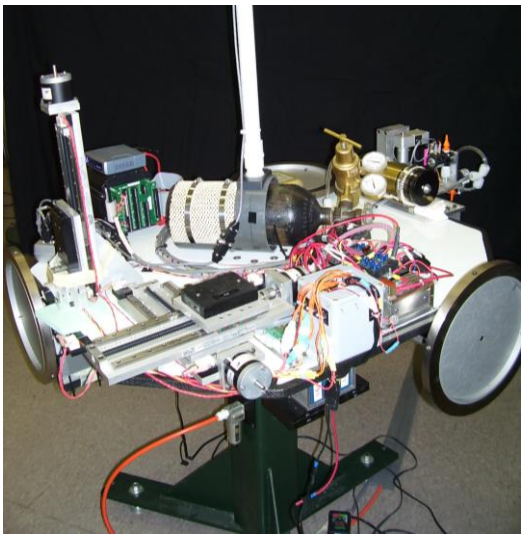

Figure 1: *Whorl-1 is a spacecraft that consists of a table-top platform on a central hemispherical air bearing.*

Figure 2: *Target image that includes a white acute isosceles triangle on a black background.*

B. Target Figure

The target figure affixed to the ceiling is a key component of the sensor system design, with constraints necessitating a simple figure that could be readily identified by a low-resolution camera, yet still provides the complete attitude of the simulator. The final target figure is a white acute isosceles triangle on a black background, as seen in Figure 2, and was chosen based on the recommendations from previous research in the SSSL[2]. This figure is easily identified in an image due to the contrasting colors (even with a black and white camera) and sharp edges, which can be found with conventional edge-detection methods. The fact that the isosceles triangle has one unique angle (the most acute angle) allows the image to identify a specific direction relative to the inertial frame, aiding in the attitude determination.

The figure is made from standard poster board and was measured to have a height of 22 in and a width of 15 in and cut to an accuracy of 1/32". This size was empirically chosen based on the desire to maximize the size of the triangle to increase the accuracy of the system while keeping the figure small enough to be seen in the camera's field of view at any possible orientation. The black poster board, which measures three feet by four feet, was placed such that it does not cover any lights on the ceiling in order to minimize any disturbance to the everyday environment of the lab. The centroid of the triangle is placed above Whorl-1, as closely as possible to the center of rotation of the Whorl. This task was accomplished using a series of plummets and laser pointers to provide an estimated position for the triangle. Because of the inexact nature of this method, an error correction will need to be added into the code to account for this systematic discrepancy. This error correction has not yet been implemented in the SSSL, as the final location of the spacecraft simulator has not yet been finalized; therefore the actual value for this error is still unknown.

Figure 3: *The Visual Vector Sensor webcam is a low-resolution off-the-shelf webcam that has been incorporated into the attitude determination system of the tabletop spacecraft simulator.*

C. Attitude Webcam

The hardware for the sensor in the Visual Vector Sensor system is a simple 320 x 240 pixel webcam created by Creative Labs, Inc and intended for use on home computers, Figure 3. The algorithm developed for use in the SSSL was intended to accommodate upgrades to this camera. However, the particular camera used in this implementation of the sensor, the Video Blaster Web Cam 3, model number CT6840, is no longer supported by Creative Labs, Inc., due to its age.

III. System Framework

A. Coordinate Frames

In order to relate the orientation of the webcam, simulator, and laboratory, it is necessary to define the various coordinate frames that exist in this system. Figure 4 describes the three significant frames: the inertial frame, the body frame, and the camera frame. The inertial laboratory frame, denoted by the $\hat{n}$ axes, is standard for all of the hardware in the lab and has been arbitrarily described in relation to the walls of the lab. The $\hat{n}_3$ axis is defined as the direction from the floor to the ceiling of the lab, and the rotation angle of the simulator is defined as its movement about this axis. The target isosceles triangle is fixed to the ceiling such that the $\hat{n}_1$ axis points in the direction of the most acute angle on the triangle. The remaining axis, $\hat{n}_2$, follows the right-hand-rule.

By definition the $\hat{n}_1$ axis is coincident with the first body frame axis, $\hat{b}_1$, when Whorl-1 has a rotation angle of zero. In case of zero tilit, the third axis of the inertial frame, $\hat{n}_3$, and the third axis of the body frame, $\hat{b}_3$, are also coincident. The tilt angle is defined as the motion of the simulator about the inertial $\hat{n}_1$ and $\hat{n}_2$ axes. The remaining coordinate frame, the camera-fixed frame (denoted by the $\hat{c}$ axes), has the same orientation as the body frame except for an offset of its origin. This is the frame from which all of the images are seen.

The webcam is fixed on Whorl-1 and experiences the same rotations as the simulator. The target triangle itself defines the inertial $\hat{n}_1$ and $\hat{n}_2$ axes. Also in each image, the camera-fixed frame is defined, since the edges of the image run parallel to the $\hat{c}_1$ and $\hat{c}_2$ axes. These coordinate frames set up the framework for the solution algorithm for the Visual Vector Sensor system.

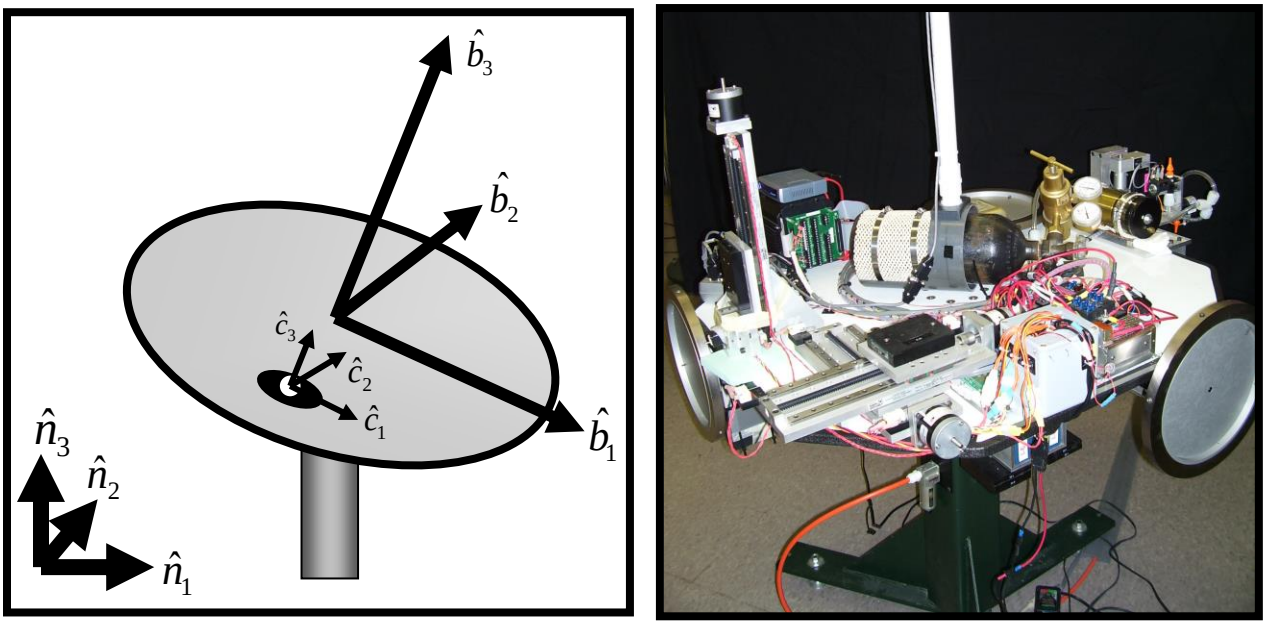

Figure 4: *Whorl-1 has three coordinate frames for the Visual Vector Sensor system: the non-accelerating inertial frame that is fixed to the lab (denoted by the $\hat{n}$ axes), the body-fixed frame that rotates with Whorl-1(denoted by the $\hat{b}$ axes), and the camera-fixed frame with an offset origin at the location of the camera (denoted by the $\hat{c}$ axes).*

B. OpenCV Implementation

OpenCV is an open-source library of C++ functions that allows for image processing, available from the website http://sourceforge.net/projects/opencvlibrary/. This library was a resource for image processing on the Visual Vector Sensor system in order to access camera initialization functions, optimized edge detection methods, and other processes, thereby reducing the need to redevelop code that has already been written. The OpenCV package was uncompressed, installed, and configured for use on the PC-104 flight computer on Whorl-1 using dynamic linking and a workbench PC-104 for testing purposes.

C. The Link between Orientation and the Captured Image

In order to develop the algorithm that would be capable of determining the three angles, it is important to understand how changes in the webcam's position and changes in the Whorl's orientation affect the image itself. Thus, to better describe the images gathered by the webcam, Figure 5 shows what the images look like during rotation and tilting of the Whorl. The first row shows how the images change with a rotation about the inertial $\hat{n}_3$ axis, and the second row shows how the images change with a tilt about the inertial $\hat{n}_1$ axis for different positions of the camera. For each photo, to the left is a corresponding representation of the Whorl, the target Figure, and the various frames. Clearly, during a rotation the centroid of the triangle stays in approximately the same location on

the image, while the triangle points in different directions. Through a tilt angle, the centroid is offset from its original location. These features of the relationship between the orientation of the Whorl and the image captured by the camera are critical in determining the angles about each axis.

D. Description of Image Capture and Processing

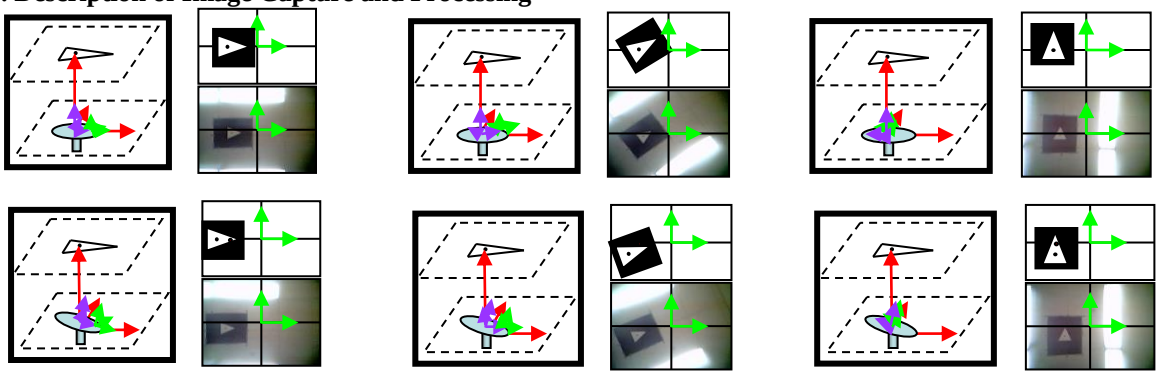

Figure 5: *Camera image through a tilted rotation rotates the triangle and displaces the centroid relative to the zero-tilt location of the centroid*

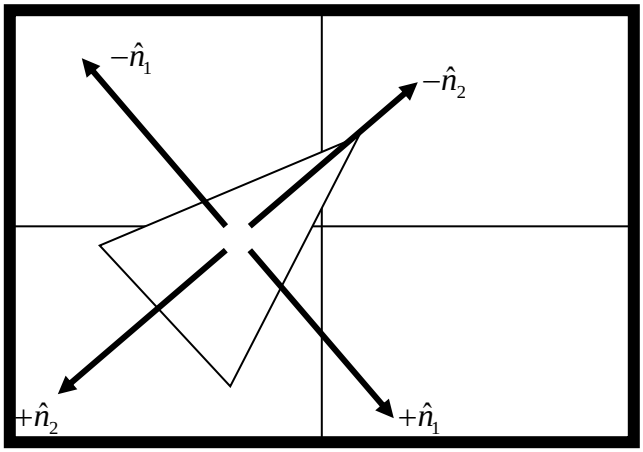

Figure 6: *The general motion of the triangle on the image when pure tilt angles are applied.*

Filtration and processing of the images are complex parts of this system. Filtration is needed before determining the centroid of the triangle and its furthest vertex, which is the essential information for this system's attitude determination scheme. In order to determine the centroid and vertices of the triangle, the image of the test figure taken by the webcam, must be converted to a 240 x 320 matrix of pixel values, such as the matrix in Figure 7. These values range from 0 to 255, where 0 is pure black, and 255 is pure white. Although the poster board is black and the triangle is white, the lighting environment is such that the resulting matrix values are only different by approximately 20 - 30 points between the poster and the triangle. Thus, images taken by the webcam must be filtered to produce useable results. The attitude determination algorithm, which includes image filtration and processing, is discussed in depth in Section IV.

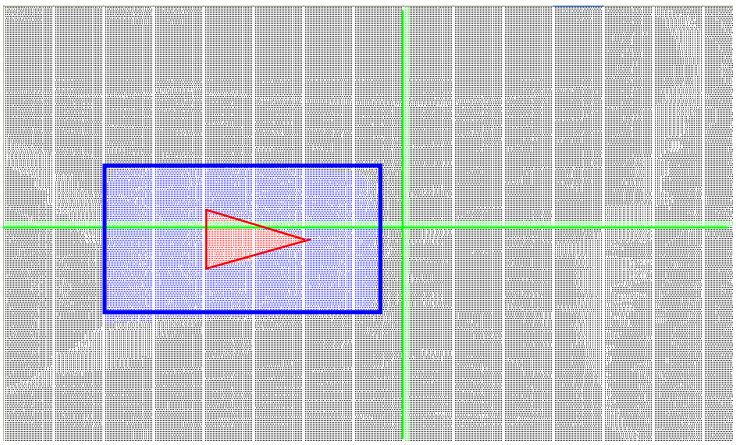

Figure 7: *The photographs captured by the camera sensor are read into the software component of the system as a matrix of pixel numbers much like this one. Here, values between 90 and 120 in the triangle have been colored red, and values under 90 in the poster have been colored blue. The middle row and middle column have been highlighted green to demonstrate where the camera axes are in relation to the triangle.*

IV. Attitude Determination Algorithm

A. Image Acquisition

An image is captured from the webcam with the function cvCaptureFromCAM(CV_CAP_ANY) in the OpenCV library, which finds a suitable camera device in any USB drive on the PC104. The function cvQueryFrame() then takes an image from the video stream capture, which can be processed much like a saved file. An example of a typical camera image is shown in Figure 8, where the gradient of the lights highlights the sharp edges of the triangle. The image is read in as grayscale, which allows faster computation, and removes unnecessary information. Next the image is converted to a data type (8uC1) which allows other OpenCV functions to process it.

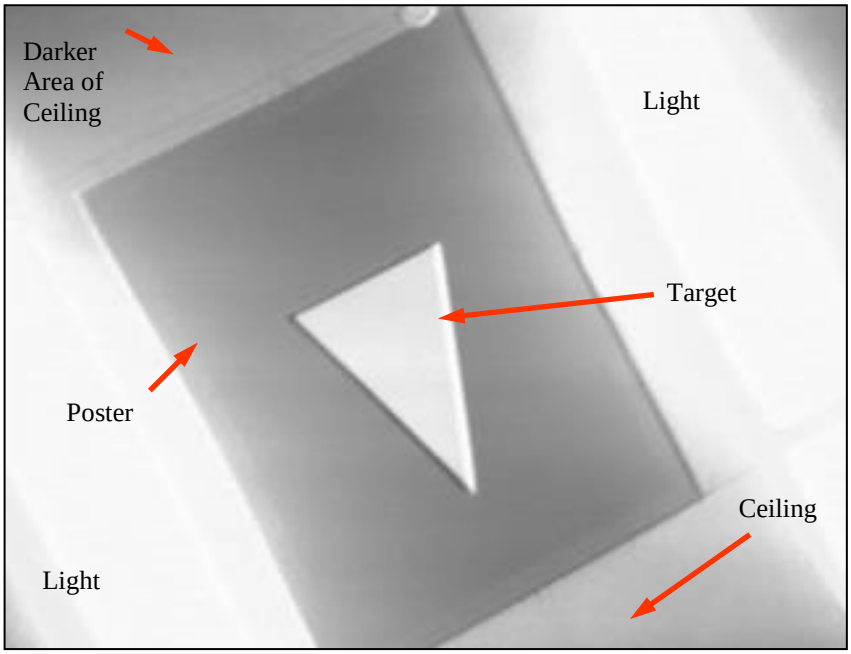

Figure 8: Typical image of Figure collected by the Webcam.

B. Image Processing

After the image is acquired, cvCornerMinEigenVal() calculates the eigenvalues of each pixel's neighborhood. The function CvGoodFeaturesToTrack() then uses these eigenvalues to determine the strongest features (sharpest points) within the image. For the purposes of this project, the function is set to find the three most prominent features, which generally correspond to the vertices of the triangle within the image- efforts are being made to ensure that the vertices are selected every time. A second constraint requires that the features be at least 45 pixels apart, which prevents the same vertex from being detected twice (note that this number was selected based on the resolution of the camera). The vertex locations are then stored in an array for later use.

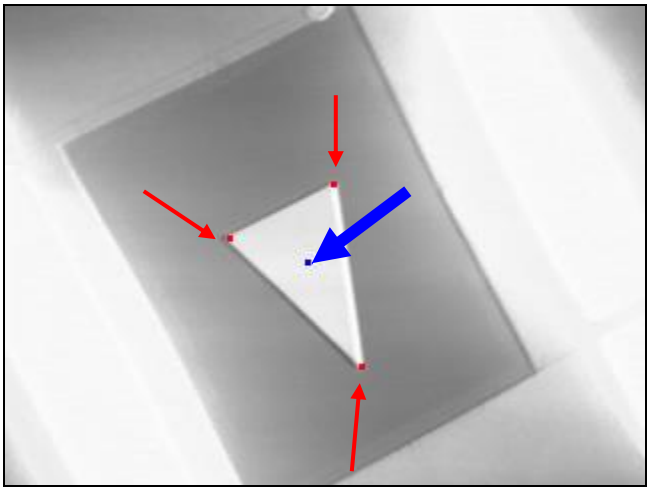

Figure 9: *Effect of FindVertex() on original image. Detected vertices are marked with the red thin arrows and the centroid is marked with the thick blue arrow.*

C. Data Processing

After the strongest features are detected, the furthest vertex of the triangle is found using the distances between vertices, and placed at the beginning of the vertex array for identification purposes. Next, the centroid is analytically determined based on the vertices; the location of the centroid is also stored in an array for later use.

D. Error Checking

Error checking is needed to verify the results of cvGoodFeaturesToTrack(), which could identify outliers, such as personnel standing over the spacecraft simulator, as features. The function outputs the three most prominent features of the image, regardless of how eccentric the triangle that is formed by these points. Currently, the only error check involves computing the area of the triangle from its vertices and matching the tabulated area with the known area of the actual triangle, with a set margin of error given for tilt and other various effects. This margin then determines whether an actual triangle has been computed, or an error has occurred in processing. If an error has occurred, the data from the current image must not be used.

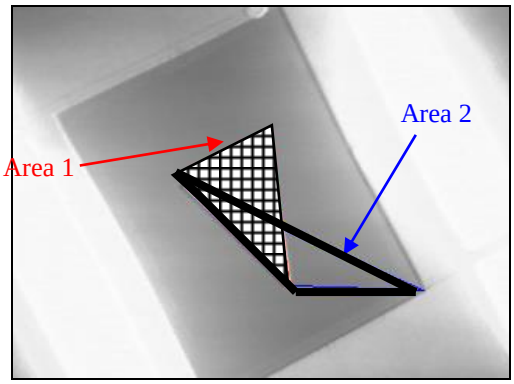

Figure 10: *Image from webcam that shows the true triangle area in crosshatching and the erroneous area based on false vertex detection outlined in bold.*

E. Attitude Determination Inputs

The next phase in the algorithm calls for the actual calculation of the camera's attitude relative to the inertial lab. The output of this portion of the code consists of three angles, each about the inertial rotational $\hat{n}_3$ axis and the inertial tilt axes $\hat{n}_1$ and $\hat{n}_2$. These methods require the following inputs in order to determine the geometry of the problem: the location of the centroid of the target triangle, the location of the vertex at the most acute angle on the triangle, the distance of the camera from the center of rotation (in engineering units), the distance from the target to the center of rotation (in engineering units), and the location of the centroid when the simulator is not tilted (also known as the zero-tilt centroid). These input values are marked in Figure 11.

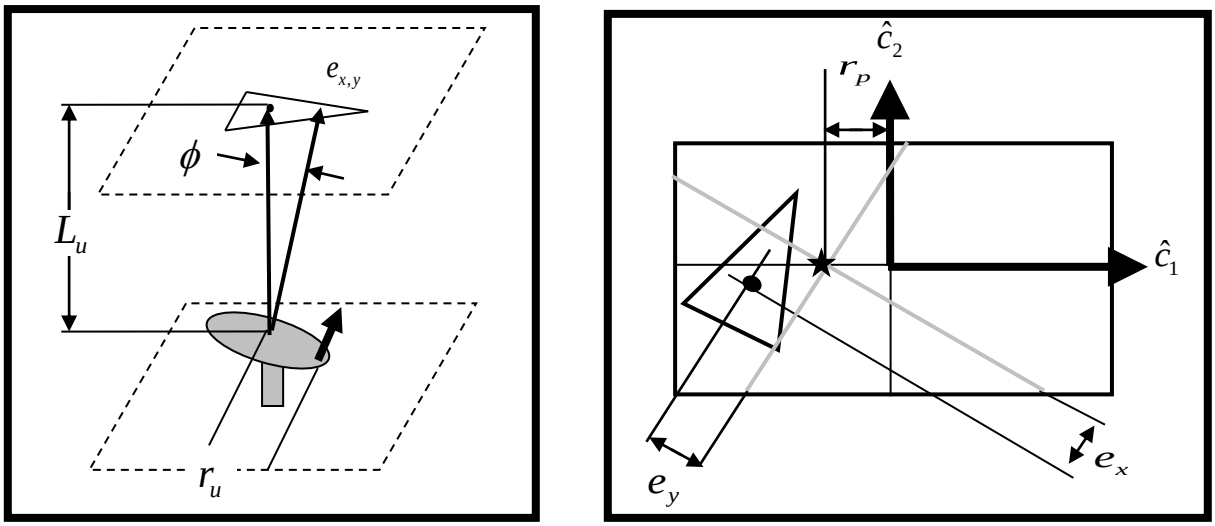

Figure 11: *The definition of key parameters in the attitude determination algorithm. The star location in Figure b is the location of the zero-tilt centroid.*

F. Rotation Angle Determination

The rotation angle, or the angle about the inertial $\hat{n}_3$ axis, can be determined from a simple vector product relation. The function that determines the rotation angle creates a vector from the stored triangle centroid location to the location of the furthest vertex, which represents the direction of the $\hat{n}_1$ axis. The camera frame, which is aligned with the matrix of pixels gathered from the image processing portion of the algorithm, is compared to this vector to find the angle between the inertial frame of the lab and the camera frame (which is fixed with respect to the body frame). This relation, shown in Equation 1, produces the rotational orientation of the spacecraft simulator.

$$\theta = \cos^{-1}\left(\frac{\underline{c}_1 \bullet \underline{n}_1}{|\underline{c}_1||\underline{n}_1|}\right) \quad (1)$$

Figure 12 illustrates this angle, θ, which represents the rotation of the Whorl about the $\hat{n}_3$ axis.

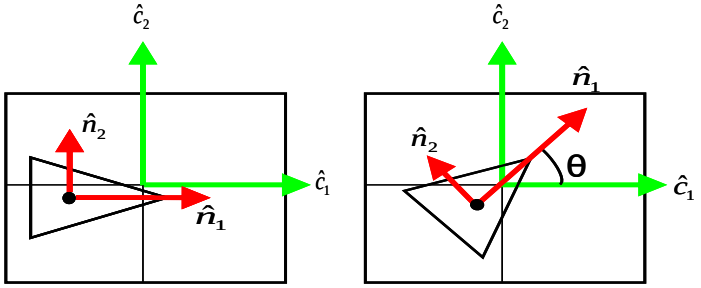

Figure 12. *The rotation angle about the* $\hat{n}_3$ *axis is found by identifying the angle between the* $\hat{c}_1$ *and the* $\hat{n}_1$ axes.

G. Tilt Angle Determination

When the tilt angles are both zero and Whorl-1 experiences a pure rotation, the centroid of the triangle would ideally remain in the same location on the image (that is, the same pixel location). Once this location is empirically established, the non-zero tilt angles can be determined simply from establishing how many pixels the centroid has moved from this zero-tilt location, and in which direction it has moved relative to the triangle. Given the inputs to this portion of the code, it is simplest to take the matrix indices and shift the points such that all of the values are relative to a frame where the $\hat{e}_1$ direction points towards the most acute angle in the triangle.

The algorithm calls for this shift in two steps. First, it must transform the matrix indices (the "frame" in which all of the image processing is done and the "frame" in which the coordinates are sent to the code)into the zero-tilt

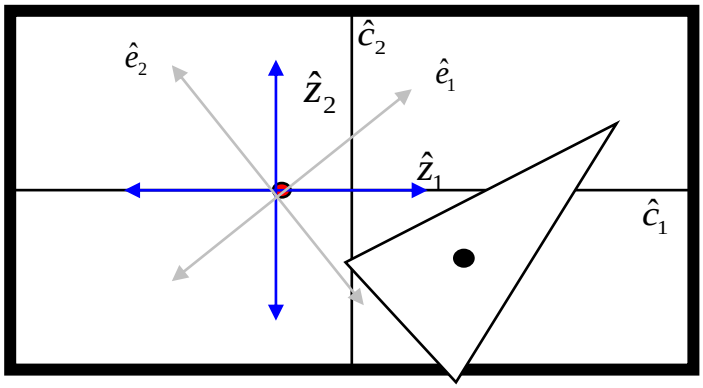

Figure 13: *The* z *coordinate frame is centered at the zero-tilt centroid location, and the* e *coordinate frame, which has the same zero location, is rotated by the rotation angle* θ.

frame. The zero tilt frame, shown in Figure 13, has its origin at the pixel location of the centroid when there is no tilt, and has its $\hat{z}_1$ and $\hat{z}_2$ axes are in the same direction as the $\hat{c}_1$ and $\hat{c}_2$ axes, respectively.

In order to change these values to the zero-tilt frame, the code then determines the center of the entire image by finding the height and width of the matrix and dividing by two, then rounding down to the smaller row or column. This rounding is necessary; otherwise no image would be able to produce a zero tilt or rotation. The index of each point is then changed into a position in the zero-tilt frame using equations 2 and 3:

$$Z_x = \left(Column\#_- \frac{imageWidth}{2} - ZeroTiltXOffset \right) \qquad (2)$$

$$Z_y = -1\left(Row\#_- \frac{imageHeight}{2} - ZeroTiltYOffset \right) \qquad (3)$$

where the Row# and Column# are the original matrix indices and the ZeroTiltOffsets (both X and Y) are inputs derived from the system calibration, corresponding to the distance in pixels of the webcam's offset from the center of rotation.

The second step in the code uses a rotation matrix to shift all of the coordinates in to a frame where the $\hat{e}_1$ axis is in the same direction as the $\hat{n}_1$ axis (i.e., the direction of the vector from the current location of the centroid to the current location of the furthest vertex.) The rotation matrix uses the rotation angle, so it is important that the rotation angle be calculated first, and then passed to the tilt angle algorithms. The rotation matrix has the form:

$$\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} Z_x \\ Z_y \end{bmatrix} \qquad (4)$$

The "e frame" is aligned with the rotation of the triangle so that the x components indicate the location in the direction of the most acute angle on the triangle. Thus, the e_x and e_y components found from the rotation matrix are exactly the distances needed to compute the final tilt angles.

The final portion of the code simply needs to translate these e_x and e_y components into actual angles. This is accomplished by finding the pixel/inch ratio (note that the inches could be any standard unit so long as it is consistent, and inches are just used for clarity) and using geometry to calculate the final angle. The relations and values that govern this problem are shown in Figure 11 and Figure 13. The distance r_u from the Whorl's center of rotation to the center of the camera's lens (in inches) and the zero-tilt x position offset of the centroid r_p (in pixels) should correspond to the same distance. Thus, these values can provide the pixel/inch ratio required for the solution.

The vertical distance L_u between the untilted Whorl and the location of the poster board (in this case, the ceiling) can now be converted to its equivalent pixel value. This value, and the offset values e_x or e_y form two legs of a right triangle, which are shown in Figure 11. Thus, it is a simple matter to determine the tilt angles ϕ_{n1} and ϕ_{n2} using trigonometry, as shown in Equations 5 and 6.

$$\phi_{n1} = \tan^{-1}\left(\frac{e_y}{(r_p / r_u)L_u}\right) \quad (5)$$

$$\phi_{n2} = \tan^{-1}\left(\frac{e_x}{(r_p / r_u)L_u}\right) \quad (6)$$

IV. Testing and Future Work

A. Testing

The Visual Vector Sensor system has now entered the testing phase of the system development. Preliminary tests indicate that the system works for all rotation angles, and given a particular static orientation, the system will output the appropriate attitude every time, indicating a highly precise sensor. More detailed testing, especially for the tilt angle determination, cannot be conducted until the camera is mounted to the simulator, since the initial values (which depend on the geometry of the system) need to be established before these algorithms can work. Much of the planned testing has the goal of determining the initial values, then finding the accuracy and precision of the system once it is running on the simulator. Ultimately, this battery of tests will provide feedback on common errors or system issues that will help shape any future algorithm development to prevent these problems.

B. Error Detection Warnings

In order to ensure that if the Visual Vector Sensor system fails it does so in a predictable and correctable manner, a set of error modes need to be developed within the algorithm. These modes will consist of a series of error-checking tests that will ensure the data provided is valid information for the simulator's attitude. While the current algorithm includes an area-check to verify that the target triangle is the identified object in the image, more can be done to make the system more intelligent. For example, the lighting in the laboratory has a significant effect on the quality of the image captured by the Visual Vector Sensor system, which may lead to variations in the calculated attitude as the lighting in the lab changes. It is possible to alert the end users of the system to this concern by computing an average brightness of the image, and outputting a warning message when the value lies outside a given threshold. Other warnings and error detection methods may be added according to the results of the testing phase of this project, including but not limited to warnings for returning output angles that are not physically possible on the simulator, detecting objects obstructing the view of the target image, and larger than expected uncertainties that may be the result of the simulator or camera being knocked out of place.

C. Mounting the Sensor

Once the system has been tested for reliability and robustness, the webcam will need to be permanently mounted to Whorl-1 before the final phase of testing can be completed. The location of the sensor must be carefully selected to ensure that the target image will always be visible in the field of view of the camera, regardless of the attitude of the simulator or the placement of other components on the simulator. The ideal placement for the sensor will be determined experimentally. After the webcam is affixed to the simulator, the initial values for the algorithm can be precisely determined and the Visual Vector Sensor system can be subjected to its final set of tests. The completion of this task is pending the final placement of Whorl-1 in relation to the laboratory.

V. Conclusion

The Visual Vector Sensor system is an operational but not yet fully implemented attitude determination system that can be applied to any tabletop spacecraft simulator. The hardware consists of five elements: the spacecraft simulator, the flight computer onboard the simulator, the target figure (in this case, an acute isosceles triangle), the camera, and the C++ code for computing the attitude of the simulator. Each of theses elements has been described, with particular attention devoted to the algorithm for calculating these results. The basic concept behind this algorithm is to capture an image with the camera, process that image to identify the vertices of the triangle and calculate the centroid and vertex of the most acute angle of that triangle. Once these values are determined and corrected to ensure accuracy, they are passed to the angle determination processes which use trigonometry and the various coordinate frame definitions to identify the angles describing the orientation of the simulator. Preliminary testing indicates that this system is functioning as planned, but further testing is necessary to determine its accuracy and to build in more error-detection methods. The camera also needs to be mounted to the simulator permanently before much of this testing can be completed. Currently, the system is tentatively described as operational, since it does capture images and produce reasonable results. Because the algorithm is not dependent on the geometry of the Space Systems Simulation Lab or the particular model of the camera, this system can be implemented on any tabletop simulator interested in visually determining the attitude of their craft.

References

[1] Kowalchuk and Hall. "Distributed Spacecraft Attitude Control Simulator: Feedback Control Capabilities and Visualization Techniques". Presented at the 7th International Symposium on Quantitative Feedback Theory (QFT) and Robust Frequency Domain Design Methods. Lawrence, Kansas. 9-11 August 2005.

[2] Woodward, Emily. "Camera-Based Attitude Determination for Satellite Simulation Applications". Space Systems Simulation Laboratory, Blacksburg, VA. 8 December 2004.