Vision Processing and Frisbee Shooter Controller Design

John D. Russo and Travis Axtell

***Abstract*—This paper outlines a vision processing algorithm used during the 2013 FIRST Robotics Competition game Ultimate Ascent by FIRST Team 2035. The vision processing was performed off robot on the driver station laptop and relevant information was provided to the robot over the arena network, which improved the robot response time considerably when compared to processing on the on-board cRIO. This vision processing scheme won the Innovation in Control Award sponsored by Rockwell Automation at the Silicon Valley Regional. Once the robot sees the field targets, it estimates the distance to the target. A Frisbee shooter controller can then estimate the Frisbee flight path and choose shooter aim angle to improve the chance of gaining points.**

***Keywords*—FIRST Robotics Competition, vision processing, Frisbee, controller.**

I. Introduction

THE annual FIRST Robotics Competition game for 2013 had a requirement to launch WHAM-O Frisbees into three differently sized rectangular goals [1]–[3]. This task can be performed consistently without computer assistance using a predetermined robot field position. However, using the advanced features of the robot controller, it is possible to program the robot to actively perform acquisition, tracking and alignment of the robot towards the Frisbee goals. This is beneficial because the game rules limit how many Frisbees may be held by a robot, and collecting Frisbees takes precious match time, so increasing the efficiency of acquiring and launching the Frisbees can drastically improve game score.

The competition uses a laptop computer running a Dashboard and Driver Station programs to control the robot. The Dashboard provides feedback from on-board robot sensors and a AXIS webcam. The driver station allows for joystick input and transmits the user control data packets over the wireless network to the robot. The processing on the robot is performed on a National Instruments cRIO, which is a microprocessor computer. In the competition, this device can run one executable compiled from either: LabView, C++, Python, or Java.

This paper outlines the approach taken by FIRST Team 2035, The Robo-Rockin' Bots, for vision processing. This technique employs the modest driver station laptop to process the webcam images, resulting in quicker processing than on the cRIO. The resulting information from the laptop are simple image pixel coordinates, pixel lengths of bounding boxes surrounding detected targets, and an estimated distance to the targets. The driver station joystick buttons are programmed to point the robot at a specific target based on the image coordinate information. The original work on this image processing technique won the 2012 Silicon Valley Regional Creativity Award sponsored by Xerox, and the follow up work in 2013 won the Silicon Valley Regional Innovation in Control Award sponsored by Rockwell Automation.

Corresponding author email: jdr9012@gmail.com

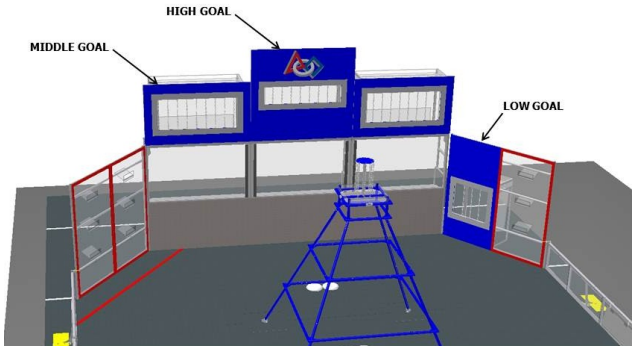

Fig. 1: The rectangular goals are located at three different heights on the field from [3].

The second section of this paper outlines an approach to use the distance estimation to evaluate a Frisbee launch angle from the robot. Although this approach was not demonstrated in the competition, this information is provided to give a complete reference to the intended direction of the robot programming. With the correct calibration, the robot Frisbee launching would be very consistent. The goal of this work is to allow the robot to shoot from any field position with high reliability of a successful Frisbee goal.

II. Vision Processing

The purpose of vision processing is to increase the repeatability of the Frisbee shots. Teams without this type of processing were able to shoot from predetermined positions on the field. Even under these circumstances, teams would routinely miss shots. A vision processing technique to identify the goals can increase the likelihood of improved shot performance from any field position where the shooter mechanism has the ability to hit the target.

The game competition used three rectangular Frisbee targets: a low target (29"×24"), a medium target (54"×21"), and a high target (54"×12") as shown in Fig. 1. These targets were surrounded by 4-inch retro-reflective tape for simplified vision processing when illuminated with a color LED ring light that surrounded the webcam aperture. These bright LEDs caused a strong single-color reflection which improved the signal-to-noise ratio for image processing.

The work presented in this paper began due to the McKaskle vision targets whitepaper [4], which outlined information of how to improve the signal-to-noise ratio of the square vision targets. Using this, the team developed a step-by-step procedure in the National Instruments Vision Assistant tool to identity field targets. Since the team uses Java programming on the cRIO, the Vision Assistant code was not able to be ported to the cRIO. Instead, the team modified the driver station laptop Dashboard program to include the Vision Assistant code. Additional features to improve calibration were also

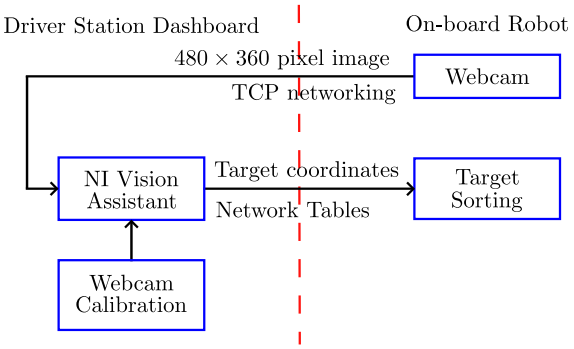

Fig. 2: The block diagram of where code is executed in the software between the driver station laptop and robot cRIO. The specific information and how it is transported between the hardware is shown for the arrows.

added to the Dashboard. The Vision Assistant processing output was then relayed to the robot cRIO over the arena network for robot control. The block diagram representation of the software layout is given in Fig. 2.

A. Vision Assistant

The vision assistant is a simple block-diagram utility to create step-by-step vision processing scripts for LabView. After testing the script using multiple input pictures to confirm correct operation, the LabView script is easy saved from the utility. This script can be added into another LabView program, such as the robot code (if using the LabView on the cRIO) or onto the Dashboard program. The script developed is shown in Fig. 7 on page 6. The descriptions of each component of the script are as follows:

Original Image LabView acquires the original image in HSL color format. This is the image that the webcam is outputting at roughly 30 fps. The image size for our use was 480×360 pixels.

Color Threshold Outputs a binary image where the pixel is True if the HSL threshold values are met, and False otherwise. This step finds the green-color response from the retro-reflective tape.

Convex Hull Fills an enclosed space (such as the retro-reflective tape perimeter in the binary image) to be a large solid rectangle. Convex means all the points on a line from a point on the filled area to another point in the filled area are still in the filled area.

Lookup Table Applies predefined lookup table transformations to the image to modify the dynamic intensity of regions in the image with poor contrast. We did not use this since we were using a binary image output from Color Threshold.

Size Check Removes areas that are too small to be real targets of interest. Sometimes the Color Threshold will return small areas of noise that are not actual targets. This step removes these areas to prevent issues with the following steps.

Filters Includes functions for smoothing, edge detection, and convolution. We used it for edge detection since we had a binary image.

Image Calibration Option to output real world units (inches rather than pixels) to the remaining blocks. We used pixels, but this was used to check to see if it is a reasonably-sized target during testing.

Shape Detection Isolates the rectangles as individual objects and ignores other shapes.

Particle Analysis Gives the bounding box parameters of each object, and the x, y pixel coordinates of the center of each rectangle.

Further details on the specifics of each of these blocks is available in [6]. The LabView script output is shown in Fig. 8 on page 6.

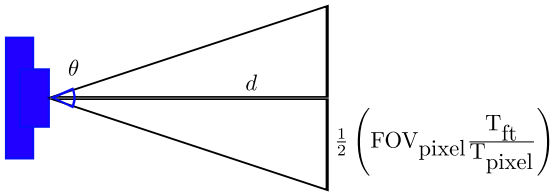

Fig. 3: A geometric model of a camera is the basis for estimating the distance to the target.

B. Labview Dashboard

The dashboard is modified to support several changes; the most important of which is that the vision processing script is added to the dashboard. A user display of the dashboard is shown in Fig. 9 on page 7. The modifications required some major changes to the underlying block diagram of the Front Panel, shown in Fig. 10 on page 7.

The calibration features added to the dashboard are three text fields that select numbers between 0 and 255 for the HSL values [5] used in the Vision Assistant for thresholding, and sliders that reflect the selected range. These were only changed when moving to a new arena, as the lighting conditions affect the optimal values for the Vision Assistant.

The vision processing algorithm estimates the distance from the robot to the vision target as

$$d = \frac{\frac{1}{2}\left(FOV_{\text{pixel}} \frac{T_{\text{ft}}}{T_{\text{pixel}}}\right)}{\tan\frac{\theta}{2}} \tag{1}$$

where d is the estimated distance to the target, FOV_{pixel} is the field of view measured in pixels, T_{ft} is the target height in feet, T_{pixel} is the target height in pixels, and θ is the camera's field of view measured in degrees or radians. A visual representation of Eq. (1) is in Fig. 3. The vertical heights of targets were used because lateral movement around the field results in large perspective changes for the horizontal width of targets. The FOV_{pixel} is known from the camera settings and T_{ft} is known, but does change based on which target is being used to determine the distance. The quality of the estimation decreases as a function of actual distance. This calculation is performed in LabView using the block diagram shown in Fig. 11 on page 8.

After the target distance is calculated, the values are transmitted outside of LabView over to the robot using NetworkTables. This is demonstrated in Fig. 12 on page 9.

C. Java Robot Code

Using the Java WPILIB implementation of NetworkTables, the coordinates for up to 6 targets are received on the robot. The Vision Assistant algorithm will correctly populate the list starting with target 0 up to as many targets are seen. However, the algorithm has no guarantee on the ordering of the targets. In fact, after processing one image of targets, the next image's target ordering may be completely different. Since it is unreliable to refer to a target as "target 0" since its position in the list may change, the targets must be sorted.

Sorting is very simple thanks to using some additional information about the target geometry and relative position between targets. The different shapes of targets can be used in a calculation of aspect ratio, which is simply width/height (both measured in pixels). The relative position of targets also is used to place the coordinate information into targets labelled "Top," "RMid," "LMid," "RBot" and "LBot" which is shorthand for Right/Left and Middle/Bottom. If only a single target is seen (*i.e.*, the robot is close and looking at only the bottom left target), then the target "Top" is always populated.

The complete robot code is available on GitHub [7]. The joystick operator interface code (`OI.java`) programmed five buttons for the targets. Thus, holding a single button would establish target lock on a particular target if available. The robot code then uses the WPILIB implementation of a PID controller (used as a P or PI controller only) to center the target in the field of view of the robot camera by rotating the entire robot. This centering happens very quickly, and now that the robot is centered on the target, the Frisbee launch angle needs to be determined for a successful shot.

III. Frisbee Trajectory Control

For the robot shooting Frisbees long distances (over, say 10 feet in length), the Frisbee flight path has an arc that can make aiming for the target difficult. The target Frisbee bins were 24 inches, 21 inches and 12 inches tall. Particularly the narrow opening requires the Frisbee flight path to be understood well enough to ensure a high probability of a successful shot from various field locations.

Many robots were limited to particular shooting locations on the field. In general, most of these positions also ensured a fairly straight Frisbee shot (and close distance) from the robot to the goal and there was no attempt to account for the Frisbee trajectory. Teams used contact with the field pyramid as a means to prevent interference from the opposing alliance (in accordance with the game rules). In addition, most robots could only load Frisbees at a loader station that was approximately 54 feet from the goals. If the trajectories for long distances were correctly estimated, the robot could stay near the loader station and launch Frisbees over the entire field. We encountered one team that did do this, but their shooting has to be calibrated manually by human and this involved many missed shots.

In this section, we describe a method that allows for robots to aim for the targets over a longer range that requires accounting for the Frisbee's trajectory. The robot design used by FIRST Team 2035 allows for a variable aim angle and shooting speed from a fixed height Frisbee turret. This technique estimates the aim angle, and estimating shooting velocity is also possible. The only required information during gameplay is the distance to the wall. This controller was never fully implemented on the robot, but is included to share the concept that the robot controller was headed towards.

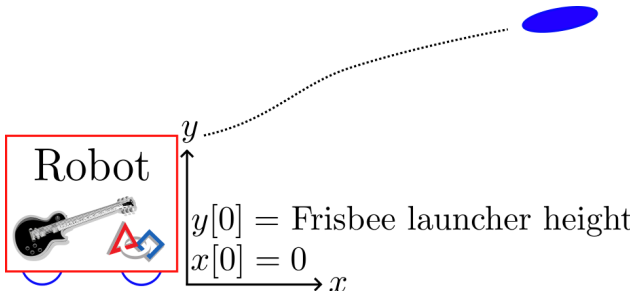

Fig. 4: The coordinate frame begins directly in front of the robot on the ground.

A. Frisbee Equations of Motion

Equations of motion are used to mathematically model the movement of an object (such as a Frisbee). In *kinematics*, the position of an object is described as a function of time. Describing a Frisbee's flight using mathematics has been the focus of a Master's thesis and journal article [8], [9]. To solve for the position, we simply create small steps in time and at each update the acceleration, velocity, and position. Although in this arrangement time is the independent variable, we do not care how long a Frisbee flight takes for our particular problem. The objective is that the Frisbee flight path has the correct height at the estimated distance from the goal. We evaluate the equations for as much time as necessary until the Frisbee has landed.

The equations of motion are nonlinear, which can complicate analysis (this is not a deterrent – simply a fact). A small change in a single parameter can have very different flight trajectory outcomes. Thus, accurate parameter estimation is important for a physically meaningful result. Unmodelled (or incorrectly modelled) parameters can be detrimental on the usefulness of a model. For example, these equations model ground distance and altitude, but they do not model the lateral motion of the Frisbee. If the Frisbee launcher causes a significant lateral motion, then we will have to compensate for this.

Before we describe the equations of motion, we first provide the coordinate frame that the equations exist in. This is shown pictorially in Fig. 4.

The drag and lift coefficients are specified by

$$\alpha_0 = \frac{-C_{L0}}{C_{L\alpha}} \tag{2}$$

$$C_D = C_{D0} + C_{D\alpha}\left(\alpha - \alpha_0\right)^2 \tag{3}$$

$$C_L = C_{L0} + C_{L\alpha}\alpha \tag{4}$$

where the parameters used are given in Table I. These values are only computed once because they do not change as a function of time (in this Frisbee model). These concepts are further explained in Aerodynamics literature as *lift* and *lift-induced drag*.

To solve for the Frisbee flight trajectory, we use Euler's method [10]. For each time step, we first compute the accelerations Δv_x and Δv_y.

TABLE I: All quantities used in evaluating the Frisbee Equations of Motion.

Quantity	Meaning	Value
α	Frisbee launch angle	Chosen [radians]
A	Frisbee area	0.0613 [m^2]
m	Mass of Frisbee	0.180 [kg] from [11]
d	Diameter of Frisbee	0.2779 [m] from [11]
h	Height of Frisbee	0.0348 [m] from [11]
g	Acceleration of gravity	-9.81 [m/s^2]
ρ	Density of air at sea level	1.23 [kg/m^3]
C_{D0}	Form drag	Estimated 0.18 [dimensionless]
$C_{D\alpha}$	Induced drag	Estimated 10 [dimensionless]
C_{L0}	Intercept	Estimated 0.33 [dimensionless]
$C_{L\alpha}$	Slope	Estimated 1.91 [dimensionless]
α_0	Lift angle offset	Computed in Eq. (2) [radians]
C_D	Drag coefficient	Computed in Eq. (3) [dimensionless]
C_L	Lift coefficient	Computed in Eq. (4) [dimensionless]
Δt	Time increment	Chosen 0.01 [s]
$\Delta v_x[n]$, $\Delta v_y[n]$	Change in velocity per time increment	Computed in Eq. (5) Computed in Eq. (6) [m/s^2]
$v_x[n]$, $v_y[n]$	Current velocity in x and y direction	Computed in Eq. (7) Computed in Eq. (8) [m/s]
$x[n]$, $y[n]$	Current position in x and y direction	Computed in Eq. (9) Computed in Eq. (10) [m]

$$\Delta v_x[n] = \frac{1}{2m}\rho(v_x[n-1])^2 AC_D, \tag{5}$$

$$\Delta v_y[n] = \left(g + \frac{1}{2m}\rho(v_x[n-1])^2 AC_L\right). \tag{6}$$

The square brackets indicate that we are forming a vector and n is the integer index. The actual time in the simulation can be determined using $t = n\Delta t$, though the time of flight is not of interest in this problem. A vector with N entries uses $n = 0, 1, 2, \ldots, N-1$.

Second, the velocity is then updated using

$$v_x[n] = v_x[n-1] + \Delta v_x[n]\Delta t, \tag{7}$$

$$v_y[n] = v_y[n-1] + \Delta v_y[n]\Delta t. \tag{8}$$

Finally, the positions are computed using

$$x[n] = x[n-1] + v_x[n]\Delta t, \tag{9}$$

$$y[n] = y[n-1] + v_y[n]\Delta t. \tag{10}$$

To initialize the solution, a Frisbee launch angle, α, is chosen. Then $v_x[0]$ and $v_y[0]$ can be estimated using

$$v_x[0] = V\cos\alpha,$$
$$v_y[0] = V\sin\alpha$$

where V is an estimate of the shooter mechanism linear velocity (from knowing the RPM rate) and the shooter launch angle. The initial position values can be taken to be $x[0] = 0$ and $y[0]$ is the height of the robot shooter in meters.

We continue this loop of evaluating equations (5) to (10) until $y[n] < 0$, indicating the Frisbee has landed.

B. Estimating Unknown Parameters

The referenced papers that identified the formulas for the Frisbee equation of motion cannot provide all the necessary information about the particular FIRST game Frisbee discs. The diameter, height and weight were specified by [11]. Some of these quantities are not known, but are estimated from a variety of techniques.

Unknown values can be substituted from another Frisbee, ideally as similar as possible. In several cases, we chose values close to the ones given by [9].

Estimating quantities can be done by simple physical observation over many trials and the only judgment can determine if the estimated values make physical sense. A Frisbee could be thrown many times and the maximum height, ground distance to the maximum height, and landing distance could be recorded. The difficulty is in knowing the Frisbee launch angle and if that launch angle is consistent with the shots. The shooting hardware for the robot could be utilized for this type of study, rather than a human throwing.

Another method for estimating the parameters is a rigorous analysis using math and engineering thought processes. An entire master's thesis has been performed on the flight dynamics of Frisbee flight [8]. The tools used in that thesis could be applied to the official FIRST game Frisbee to determine the unknowns. The thesis even uses optimization techniques to find the "best fit" of parameter values, which is an excellent use of that theory.

C. Evaluating the Equations for Different Frisbee Launch Angles

These equations can be evaluated on the cRIO for the current aim angle to position the robot the correct distance from the goals. However, we did not need to do this. A lookup table would suffice to choose a angle from the vision processing's current estimated distance from the target. The equations can be solved and the robot code need only have the distance and angle values. Many launch angle trajectories are shown in Figures 5 and 6.

The robot for FIRST Team 2035 was equipped with a gyro sensor on the Frisbee shooter. We would estimate our current shooting angle using this sensor. When the look up table specified a different angle to shoot at, the gyro would be used to ensure the shooter moves to the correct angle. Like the robot position controller in Sec. II-C, the shooting angle would be set through a PID controller.

Although we only built a look-up table for a shooting angle, it is possible to do this for an angle and velocity. If the shooting velocity of the Frisbee can be controlled (such as monitoring the RPM rate of the mechanism), then it would be possible to also control the velocity. In this arrangement, if the robot must

maintain its current position and adjust the aim angle and disc velocity, a Frisbee goal could still be completed. This would greatly enhance the opportunities to overcome obstacles met during game play.

IV. Conclusion

This paper presented a novel technique for vision processing of rectangular targets used during the 2013 FIRST Robotics Competition game Ultimate Ascent. The paper outlined the step-by-step procedure of using the driver station dashboard to process the vision targets and transmit information to the on-board Java program running on the robot.

The second contribution of this paper was a Frisbee trajectory based controller to assist in selecting launch angle for the robot. This is done to improve the likelihood of a successful shot into a goal since the Frisbee flight path is not a straight line over long distances. Although this was not implemented in time for the competition, the concept is explained for educational purposes.

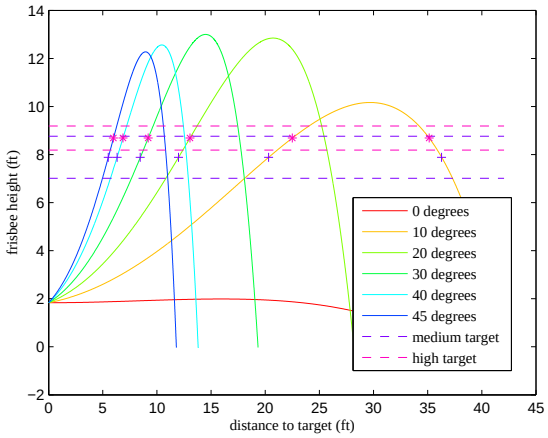

Fig. 5: The trajectories of six different Frisbee launch angles are shown. The top and bottom of the middle and high goals of Fig. 1 are shown as dashed lines. A tick mark is placed when the trajectory crosses through the middle of a goal (crossings with very steep slope are not marked). From these tick marks, we know what angle to shoot at from a given ground distance to the target.

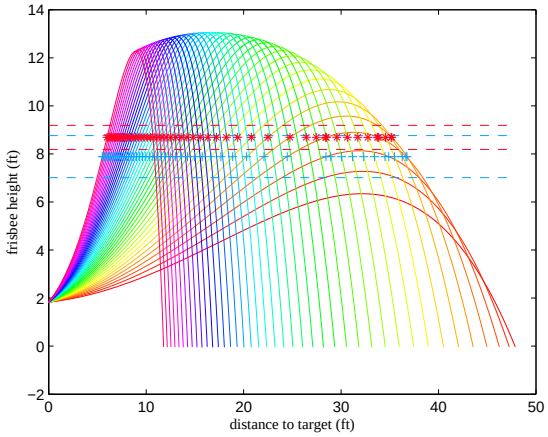

Fig. 6: The angles between 10° and 45° with 1° degree separation are shown. A look up table could quickly be established from this data to determine the appropriate angle for a given distance.

Acknowledgment

The authors would like to thank FIRST Team 2035 for putting up with the programming team. This work was simplified due to Greg McKaskle's whitepaper. This work was possible due to the sponsors of FIRST Team 2035: Carmel Unified School District, Monterey Bay Aquarium Research Institute, Naval Postgraduate School, FOCUS, Padre Parents, El Camino Machine and Welding, SJ Automation, KnappWerks and Carmel Valley Video.

References

[1] US FIRST website. [Online]. Available: http://www.usfirst.org/

[2] FIRST Robotics Competition Game. [Online]. Available: http://www.usfirst.org/roboticsprograms/frc/game-and-season-info

[3] FIRST Robotics Competition Game manual. [Online]. Available: http://frc-manual.usfirst.org/

[4] G. McKaskle. Vision Targets white paper. [Online]. Available: https://decibel.ni.com/content/docs/DOC-20173

[5] Wikipedia: HSL and HSV. [Online]. Available: https://en.wikipedia.org/wiki/HSL_and_HSV

[6] National Instruments Vision Assistant Tutorial. [Online]. Available: http://tinyurl.com/n92c88z

[7] FIRST Team 2035 Java code repository for year 2013. [Online]. Available: https://github.com/jesusrambo/ScraperBike2013

[8] S. Hummel, "Frisbee Flight Simulation and Throw Biomechanics," M.S. Thesis, Dept. Mech. Engr. Univ. California Davis, Davis, CA, 2003.

[9] V. R. Morrison, "The Physics of Frisbees," Electronic Journal of Classical Mechanics and Relativity, Mount Allison University, pp 1–12. Apr, 2005.

[10] Wikipedia: Euler's Method. [Online]. Available: https://en.wikipedia.org/wiki/Euler_method

[11] AndyMark 2013 FRC Game Pieces Frisbee. [Online]. Available: http://www.andymark.com/FRC-2013-Disc-p/frc-2013.htm

John Russo is a freshman at Bucknell University working towards a double major in Electrical Engineering and Neuroscience. His passion for robotics started five years ago with his involvement with FIRST team 2035. Under his leadership, the team received two prestigious awards for Innovation in Control, and Creativity in Design. His best summer memory was an internship under Dr. Timothy Chung at Naval Postgraduate School in Monterey, California developing communication protocols for swarming UAVs.

Travis Axtell began his involvement in FIRST robotics as a high school senior. He has mentored 3 FRC teams (342, 612, 2035) over 5 years in topics of programming and robot control. Travis graduated *cum laude* from Clemson University with a Bachelor of Science in Electrical Engineering and has a Master of Science in Electrical Engineering from the Naval Postgraduate School. He is a Department of Defense SMART Scholarship recipient and a Ph.D. candidate in Electrical Engineering performing research on space-based imaging systems.

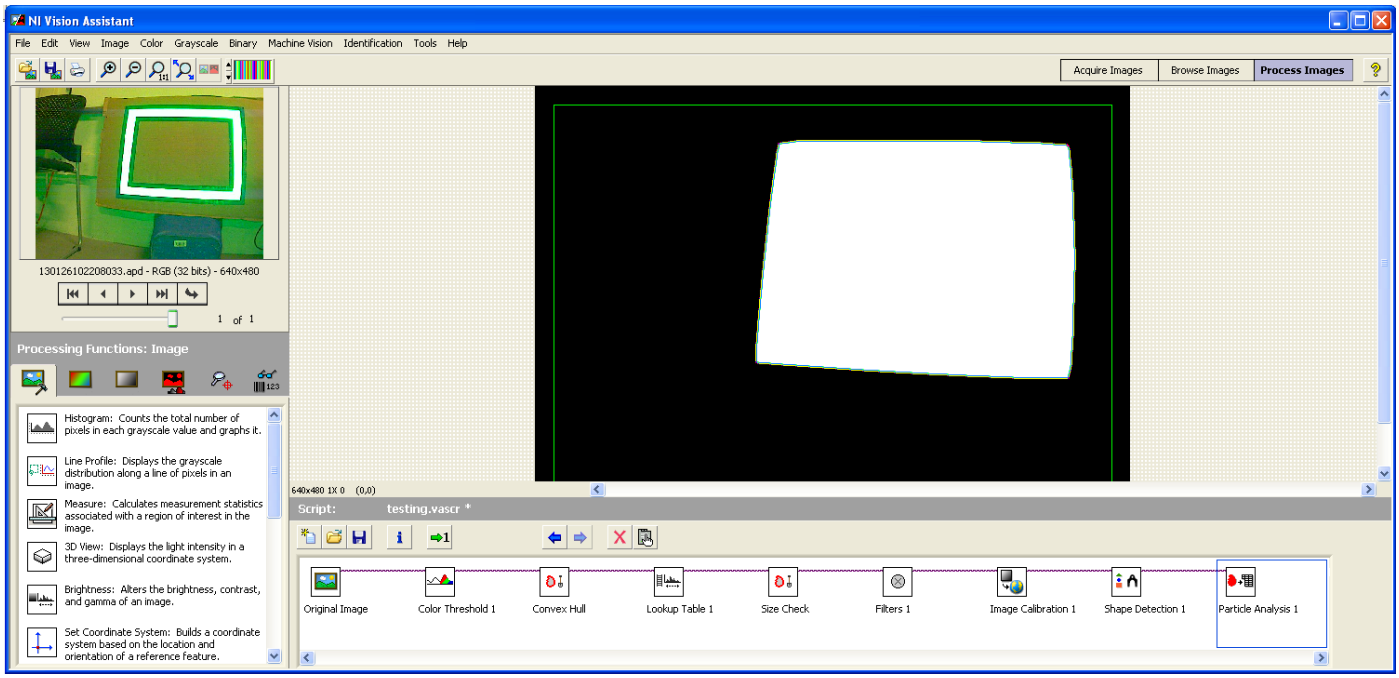

Fig. 7: The vision script layout using Vision Assistant. The input picture is in the upper left and the algorithm output is given in the large center image. The step-by-step procedure is listed along the bottom. This image is embedded at full resolution and can be copied outside of this PDF document.

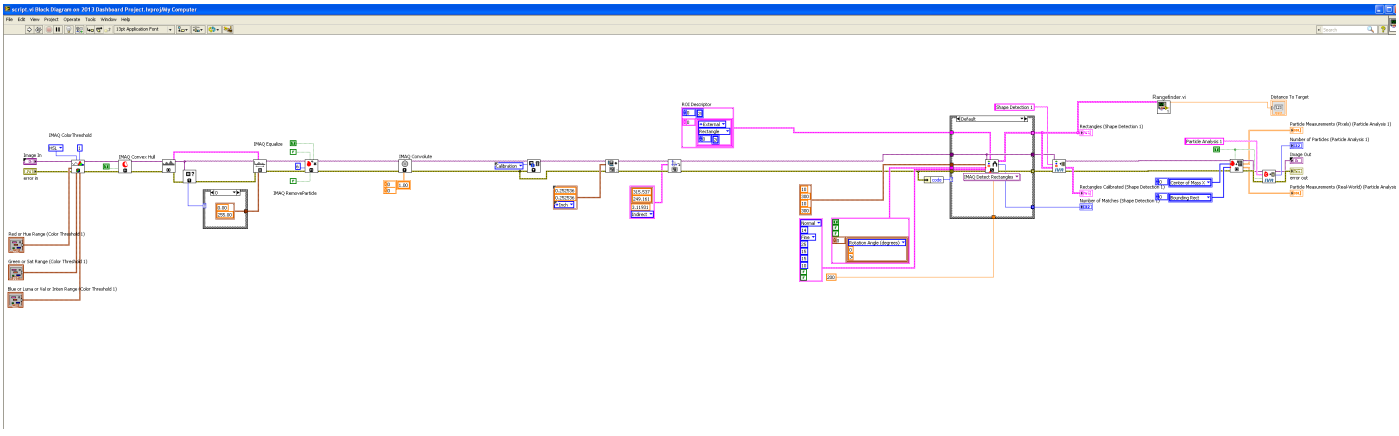

Fig. 8: This is the output script from the Vision Assistant. The only modification to the script is to use the calibration settings from the Dashboard Front Panel rather than explicit constants. The modification is at the far left. This image is embedded at full resolution and can be copied outside of this PDF document.

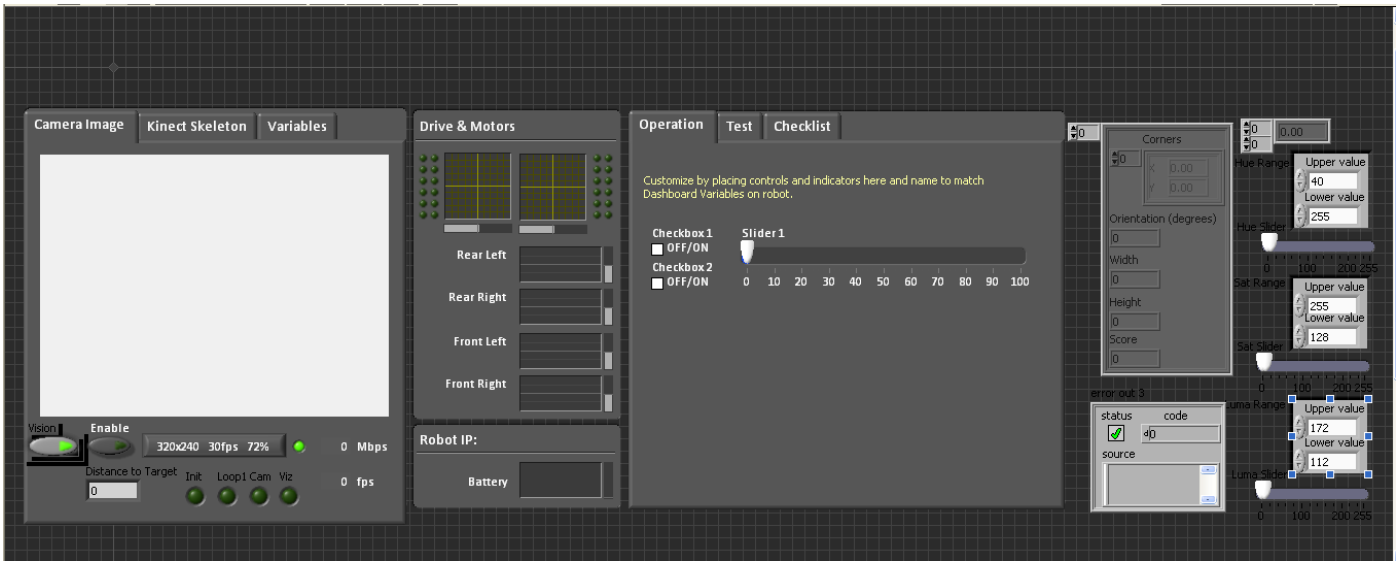

Fig. 9: Front Panel of the Dashboard display. The new vision processing option is shown on the left. There are several new panels on the far right that show network connection status and the HSL calibration settings. This image is embedded at full resolution and can be copied outside of this PDF document.

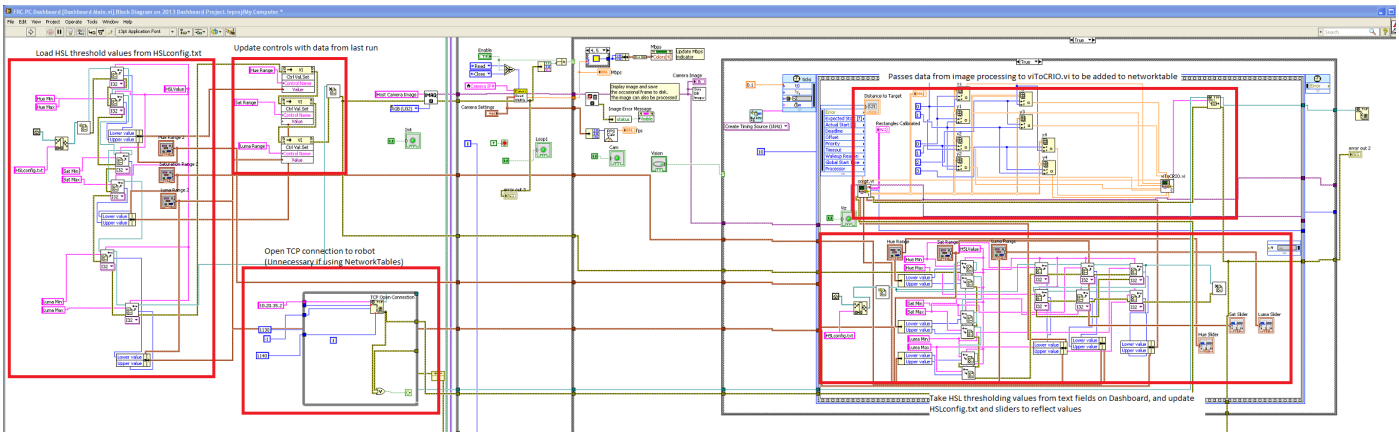

Fig. 10: This is the block diagram of the main display of the Dashboard. The webcam calibration settings and the option to enable vision processing are performed here. The vision processing script appears as a single icon (`script.vi`). This figure has markup comments to explain the purpose of the various groupings of blocks. This image is embedded at full resolution and can be copied outside of this PDF document.

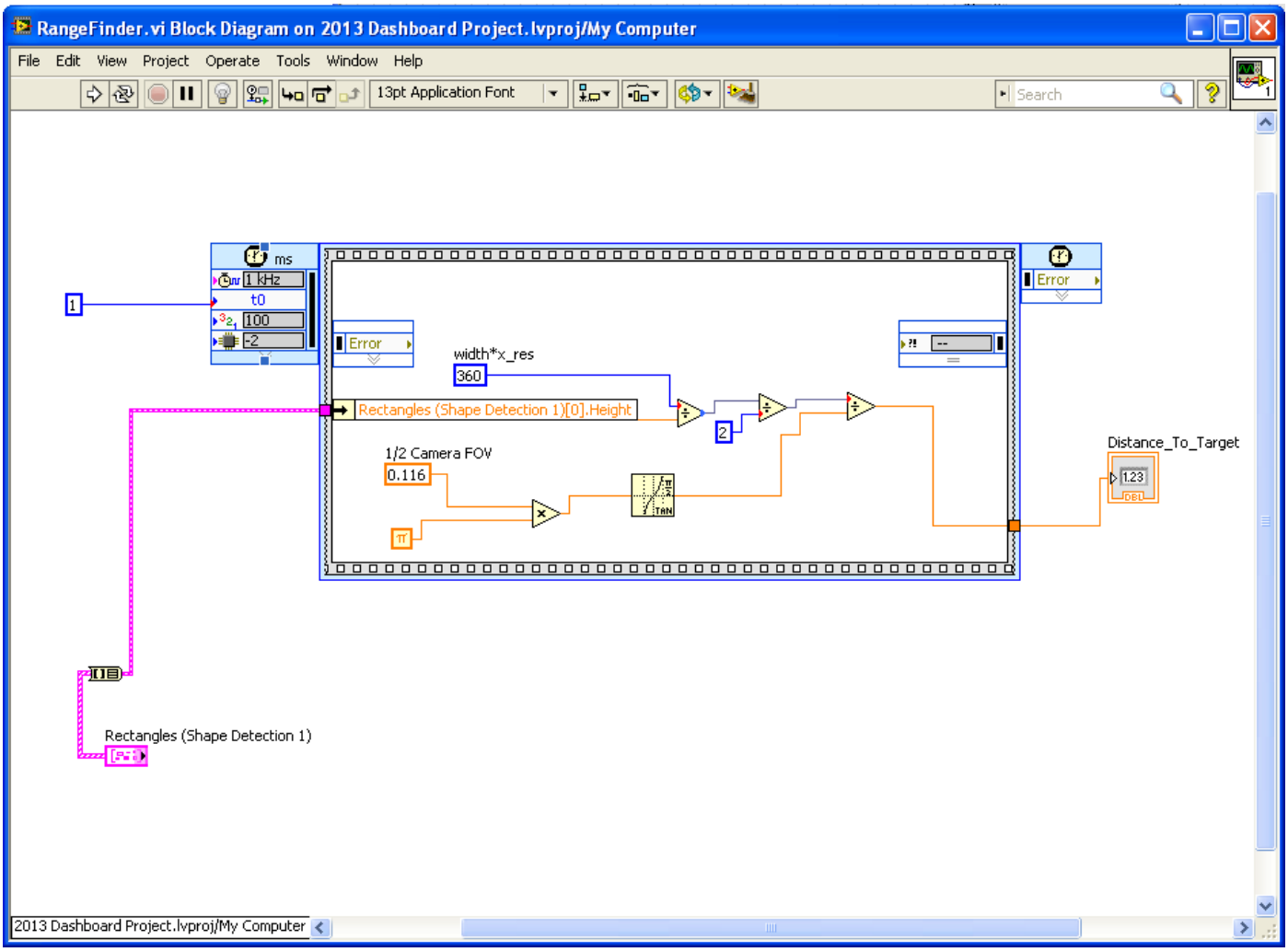

Fig. 11: The Range Finder is the distance calculation to the target. It always uses Target numbered 0, which guarantees if the algorithm sees any target that it is calculating a distance. This image is embedded at full resolution and can be copied outside of this PDF document.

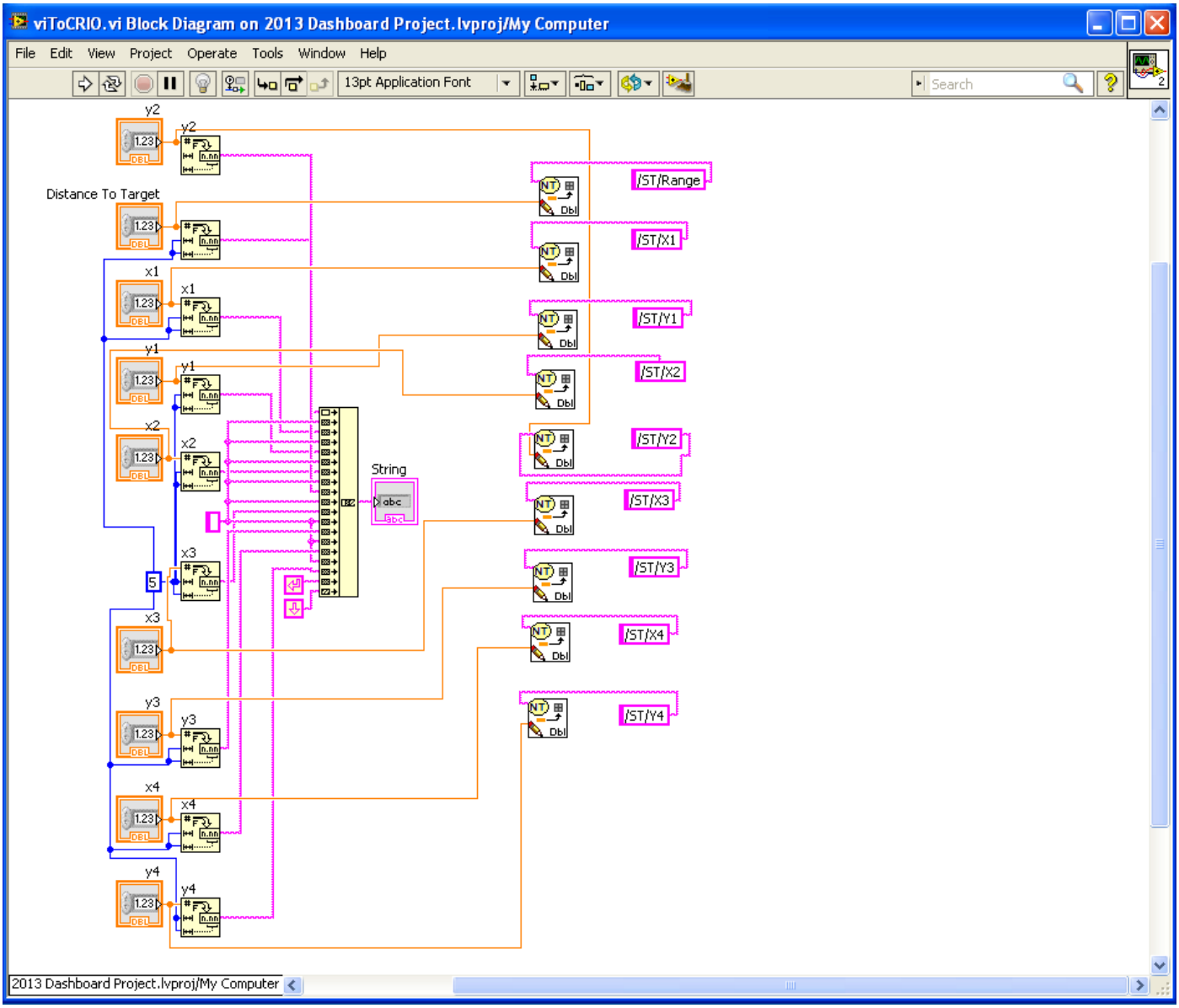

Fig. 12: This block diagram takes each of the targets center x and y pixel coordinate and places those values into separate entries in the NetworkTable. This image is embedded at full resolution and can be copied outside of this PDF document.