ECE 459 Final Project Report
Electronic Ladder Golf Team
Hayden Bader, Alyssa Henline, Ying Wang
Friday, December 11th, 2015

Initial System Design

Bola Detection

Initially, we envisioned that the system would use a system of RFID transmitters and at least three RFID receivers. The transmitters would be located within the bolas and would transmit once it has landed and stopped accelerating. The receivers would then triangulate the location of the bola based on the difference of the time that each RFID receiver acknowledges.

Several issues occurred with this design from its conception. First, the practical distance required in order to differentiate between the timing of the signals, given the speed of electromagnetic transmission, would be error prone due to the slightest differences in the environmental medium surrounding the sensors, sensor build, and other factors, as well as require an extremely accurate sensor (which would be prohibitively expensive) that will likely need to measure time in microseconds in the coarsest resolution and communicate with other sensors (which if it has any communications latency at all between the receivers would render the proposal null). All processing would likely require hardware able to time in terms of microseconds. The system would not be robust since any movement of the receiver and transmitter would likely not only disrupt the extremely sensitive sensors but also possibly create an inaccurate position signal. Finally, the location of the bola would likely be inaccurate enough to warrant the need for another sensor to confirm the output, which would render the need for the first sensor moot.

Another solution proposed was pressure sensors on the rungs themselves; when the bola lands on the rung, the pressure from the weight of the bola will trigger a change in the pressure felt by the pressure sensor, which will prompt a change in the score.

Several problems occurred with this proposal that prevented serious implementation. Pressure sensors available were either long strips or wide squares of pressure sensitive material; applying this sensor to the rung without disrupting the integrity of the sensor would be difficult as the sensor was not designed to be wrapped around a cylinder. If the setup needed to be taken apart or experienced rough usage, the need to expose the sensor due to the physical nature of the interaction with the sensor may wear out the sensor very quickly, especially due to the nature of impacts from bola throws. The system would be unable to differentiate between the different teams, unless one team's bolas had different weights than the other team's (in which case the game is unfair). Finally, the system would not be flexible enough to detect different cases the system would eventually need to support (What if the bola wrapped around the frame as well as the rung? What would happen if one bola knocked off another bola? What if the bola never hit the frame in the first place? What if the bola hung onto the frame precariously and then fell off?).

The final proposal for the initial system design for bola detection is a computer vision system composed of a camera and an image processing system locally on the embedded system or barring that on a microprocessor like the Arduino or a miniature computer like the Raspberry Pi. When the bola triggers the bola detection system, the camera takes a picture of the setup and transmits it to the Arduino. The Arduino takes the image and forwards it to another local computer, which will process the image and return a score to the frame. The system would likely be mounted on a frame on the ladder golf set, which would allow for a constant reference frame for the camera.

On the software side, the team is looking towards SLIC superpixel analysis in order to detect clusters of pixels similar to each other; once the superpixels have been iterated through, there should be enough comparison to a template image where the bolas can be detected and brought out, perhaps with some mean squared error threshold attached.

The benefits of this system include its increased robustness; there is no interaction at all with any moving objects. This will enable the system to remain more reliable than the other solutions considered. Another benefit is that it is cheaper; instead of many different pressure sensors or extremely sensitive RFID receivers, the camera module and the embedded vision cost of materials should be relatively small considering the limited area size and quality of the hardware needed for basic image processing.

The problem with this system include that nobody on the team knew enough about digital image processing and computer vision to create an algorithm; the knowledge base for constructing such a system would have to be made separately from the embedded system and require a significant investment of time and energy. Another worry remained from the limited computing power of the Raspberry Pi; whether it is fast enough in order to process the image remains a question to be found.

Final System Specification

Bola Detection

The final system proposed in the demonstration is a computer vision algorithm that works in tandem with a setup between a Raspberry Pi, an Arduino connected to the Raspberry Pi through USB, and a camera attached to the Arduino through digital I/O. The hardware aspect of the initial system specification did not change much throughout the project.

The computer vision algorithm was more complicated than originally thought. It is split into two components: rung detection and bola detection, both of which are described further below and in the results and discussion section.

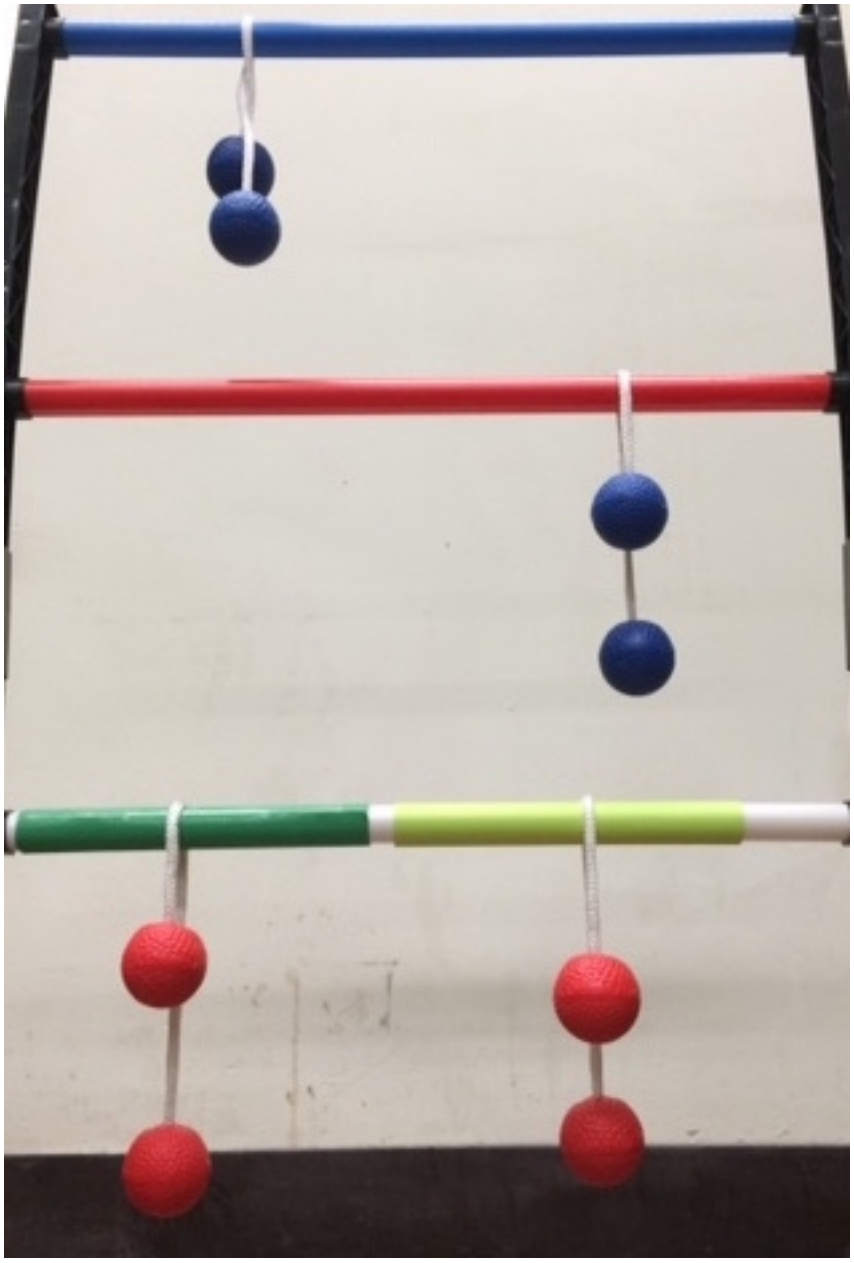

Figure 1: Initial test image, cropped to display only the frame (full resolution)

Figure 2: Example output from the vision system (full resolution)

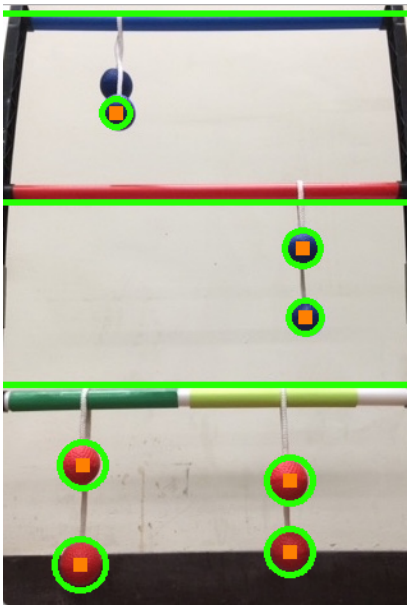

Figure 3: Result from test input through _final.py. This successful detection case will return a completely accurate score back to the frame.

For rung detection, the algorithm transforms the image several times and then it applies a Hough line transform to return the rung location. First, the image is converted into the HSV colorspace, which describes the hue/saturation/value of each pixel instead of combinations of red/green/blue. This allows for easier thresholding as hues are more similar to each other than combinations of individual colors.

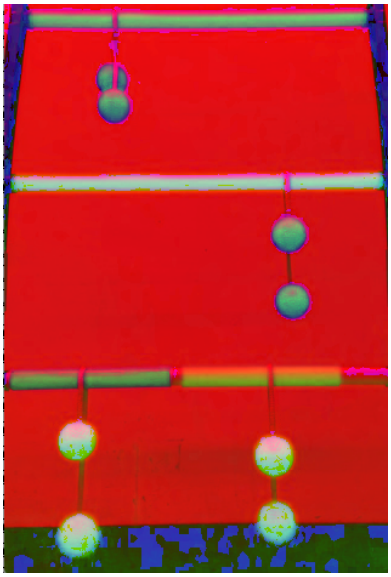

Figure 3: Test image transformed to the HSV colorspace

After defining some range of HSV values for a lower and upper bound, the algorithm thresholds out the desired color (whether blue, red, or green). This threshold should capture the rung and perhaps some peripheral colors given a laboratory background.

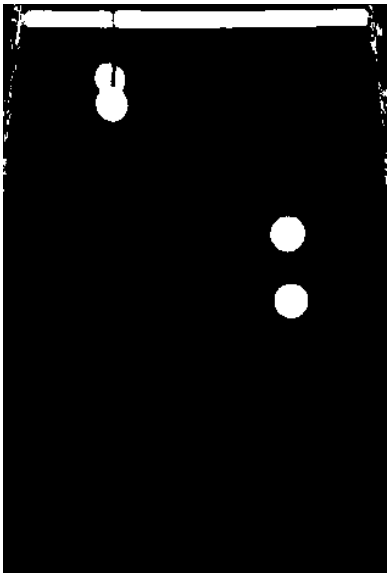

Figure 4: Output of color thresholding blue as a binary mask.

After thresholding, the resulting mask has a median blur applied to it; a median blur applies a kernel for each pixel and sets the median value of the kernel to be the pixel value. This removes any noise from the image by not allowing noisy pixels to influence the rest of the image processing sequence later on.

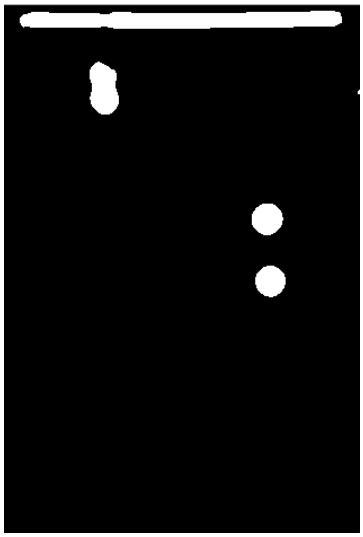

Figure 5: Binary mask after median blurring is applied.

Next, the Canny edge detector is applied to the binary mask, erasing any pixels that do not provide a gradient. This removes the weight of redundant pixels in terms of determining the eventual line coordinates.

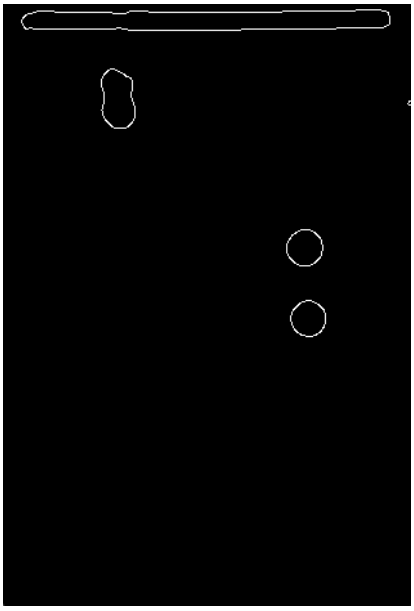

Figure 6: Binary mask after Canny edge detection is applied.

After the image has been processed, a series of Hough line transforms is applied to the remaining mask. The Hough line transform uses a polar coordinate system in order to detect the line, and the given method used in the algorithm converts the result back into Cartesian space. The key is using the prior knowledge that there should only be one rung, and given the image a line running through the rung should be the line giving the strongest correlation. Therefore, the algorithm iterates through a while loop, increasing the threshold and reapplying the Hough line transform until only one line is left; that line is assumed to be the rung. The line is then superimposed on top of the image to verify to the human eye the algorithm's accuracy. This is then repeated to detect all three rungs and display them on the image.

For bola detection, the algorithm performs a SLIC superpixel analysis on the image, and iterates through the list of superpixels. The number of superpixels the image is broken up into is a matter of parameter tuning and can be adjusted within the algorithm. A blank slate with the same width and height dimensions as the original image is instantiated. The algorithm proceeds to iterate over each superpixel, generating the mask and applying the image on top of the mask. The applied mask is then converted into the HSV colorspace and thresholded by a range of blue, red, and green HSV ranges. The HSV thresholds are determined through parameter tuning. The superpixel is then compared to a blank slate and the mean squared error is calculated in order to generate a value surmising the difference between the applied mask and a blank slate. If there is no difference, then there are no relevant pixels within the superpixel that would matter for bola detection, and they are not included. If the MSE is not zero, then the superpixel is added to the blank image. Sequentially, the blank image develops into an image that is composed of relevant superpixels.

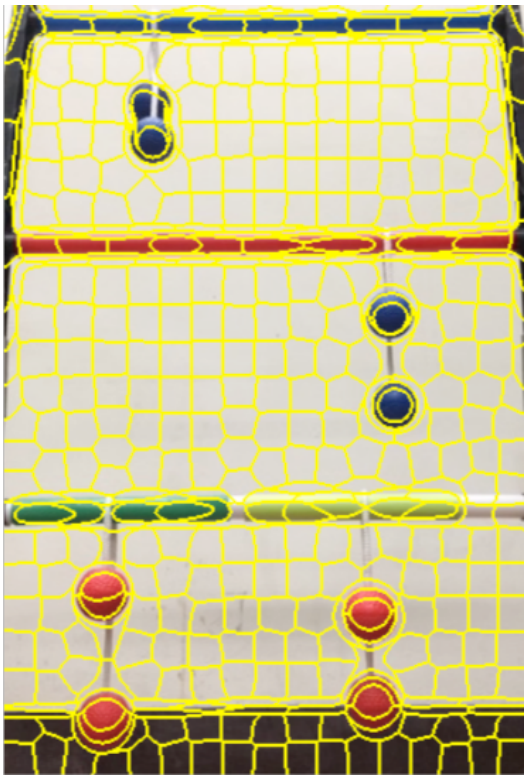

Figure 7: SLIC superpixel analysis output on the test image.

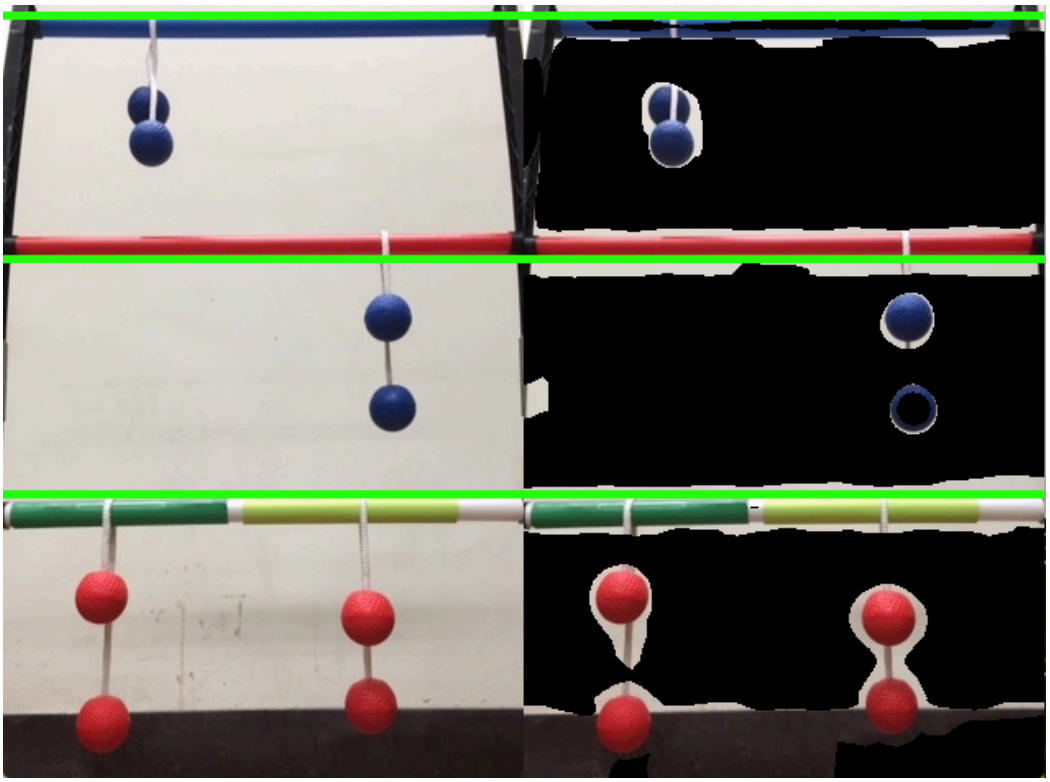

Figure 8: The original image compared to the thresholded image. All possible background is removed; removal of more background would involve decreasing the granularity at the expensive of time performance.

Once the superpixel analysis is complete, the comparison remains much like that of the rung detection. The algorithm thresholds the image again in order to better establish the location of a particular color, generates a binary mask based on the threshold, passes it through a median filter in order to remove any residual noise, applies a Canny edge detection to the image in order to remove any non-edge detected pixels, and then applies the Hough Circles Transform in order to detect the number of circles. The Hough Circles Transform uses a 3-space matrix and an accumulator in order to check the possible (x coordinate, y coordinate, radius) triple in order to see if the proposed circle thresholds above the user-established threshold value needed to confirm a circle is legitimate. If it is, a circle displaying the circle itself and a rectangle indicating the circle's center would be drawn on top of the image. This is done for both the red and the blue bolas separately, in order to differentiate between the two teams and keep their scores separate.

When all circles are detected, the scorekeeping portion of the algorithm begins. Given the list of rung's detected y-values, the minimum of each y-value array is generated and returned as the y-threshold for each detected bola. This divides the image into three regions where the bola could be located; the red rung zone, the blue rung zone, and the green rung zone, based on what the rung y-threshold values are. Depending on the circle's location based on its y-coordinate, it would fall into one of those sections. The score can then be calculated from the location where the bola is found, and given to the specific team based on which color the bola is. The score is

given by the division of the number of bolas by 2 added with the modulo of the number of bolas with 2.

The image also suffered from several deficiencies, including a peculiar blue glow in the left side of the image, as well as compression artifacting; the image processing algorithm introduces features to handle both cases. The compression artifacting results from the nature of the JPEG compression format, and indicates high frequency changes in pixel changes. To look into the nature of this, a segment of the image can be passed into a 2D Fourier transform to see what are the most frequently occurring frequencies in the frequency spectrum, and develop a filter to screen out those specific frequencies while leaving the others intact. The blue glow in the image makes color thresholding difficult; one solution is to isolate the particular subset of the image desired to capture (the bolas/rungs, and the blue glow), calculate the mean HSV values and the covariance matrix for that image subset, and establish a threshold based on the Mahalanobis distance for the rest of the image. The Mahalanobis distance is akin to a z-score for an n-dimensional Gaussian distribution; since the Gaussian distribution has the most entropy or randomness out of established distributions, it is the most forgiving out of all distributions; potential z-scoring for other distributions would remain stricter and less accurate. This is an alternative to the guess-and-check parameter tuning loop that occurred previously in the project.

Results and Discussion

Bola Detection

The computer vision algorithm portion of the project worked surprisingly well for a variety of representative images. The rung detection especially was particularly robust in the given setting even without changing any parameters; this was testing images from a 1080p iPhone camera to the 240p image available in the embedded vision platform. Bola detection proved to be a much harder problem, especially since the size of the bola varies due to due to distance, nor the shape of the bola due to occlusion, nor the expected number of bolas in the different regions available; the only assumptions available included the general color of the bola. Automatic methods would be required in order to differentiate the bola color and potentially the blue glow, and those would be different depending on the environment the system operates in, which would require manual isolation of those elements from a set of images.

The split of duties in rung detection and bola detection due to the different search spaces of the features in question. With rung detection, the number of rungs to detect was known, and the gradient of the rung remained more or less constant through the entire length of the rung. This meant that color thresholding could be inaccurate and still detect the rung with a high level of accuracy. Meanwhile, from the vision system's perspective, since there is no ability to determine the number of bolas that are supposed to be on the board (since that is the value that needs to be returned to the scorekeeper and the purpose of the system's existence), and bolas themselves will

appear to be different sizes at different distances, and suffer from occlusion among the different elements present on the frame setup already and thus two balls per bolas will be impossible to detect every time with a 2D image.

The overall design of the image processing algorithm changed much during the course of the project. Good design principles of the codebase were generally not obeyed, and significant refactoring would need to be done in order to make the code presentable in any meaningful manner. More importantly, the algorithm was designed in order so that the image would be able to fit it; this disposes of the information given to the algorithm, and renders the algorithm as inflexible and monolithic in terms of implementation, which should not take place in any software development process.

Analysis for Revision 2.0

Bola Detection

The hardware that the computer vision algorithm should definitely be improved for the second revision. Given the same time frame and the same budget, a stereovision setup should have been used and an embedded vision module should have been developed and deployed in lieu of the Raspberry Pi/Arduino/Camera setup that is currently in place. This setup would only be marginally more expensive, but would complicate the design.

While the computer vision algorithm does do a surprisingly decent job at detecting bolas and rungs in the images sent to it, the hardware setup is limited to the lab space and is not portable to any arbitrary environment. The images sent are extremely slow; 640p images take about a minute in order to reach the Raspberry Pi before proceeding to be sent to the server, to the point where the team had to amortize the delay by taking a picture only once per round. The camera quality was also lacking; distortion, compression artifacting, and color infidelity remained problems (though the image processing algorithm attempted to solve this issue through server-side software).

A stereo vision set would enable the system to differentiate the frame and bola arrangement by not only color, but also distance. This would be extremely useful in differentiating in actual tailgate environments, since there are many artificial and natural colors that would likely match to some degree any colors used for the rungs and bolas; it is far less likely that any items would match both the color as well as the distance threshold to become a false positive. An ideal stereo vision setup would enable the system to be background agnostic. This system is likely the simplest implementation that would accurately be able to complete this task in any given environment.

A PCB would fulfill the desire for the computer vision aspect portion of the project becoming an embedded system, while also improving performance as the image would not need to be transformed into serial data and streamed between different devices 32 bytes at a time.

A higher resolution camera would have been greatly appreciated, since more pixels would mean more data about the captured environment for the algorithm to process. A higher fidelity camera would have been welcome as well, since the original camera suffered from both a fisheye lens distortion, as well as the blue glow in the left hand side of the image. However, the camera component would have been more expensive in this instance since the component quality would have been increased, and the budget limit for the project would likely have been exceeded. The team would then have to work to secure additional funding for the project (which given the availability of funding sources on campus would not pose a significant obstacle), or scale down the scope of the project itself to sensor design (which if given would provide an extremely interesting divergent comparison to other teams).

Latency remained a bottleneck in our project since images are data intensive by their nature; since the current setup serially transmits an image 32 bytes at a time between the Arduino and the Raspberry Pi, communications is painfully slow. Communications via SPI or I2C directly on the Raspberry Pi GPIO pins would be preferential. If the system could eliminate latency entirely, it would be possible to keep a real time score by streaming images continuously, or take a burst of images while the bola was still accelerating in some direction and give the most accurate score, which should be the score returned by the most images.

If time permitted, the algorithm can be written as an ASIC and included on the PCB itself, without the need for streaming data anywhere; the OpenMV project is a good representation of this end goal. This scenario could only arise if the ICs available for this embedded system were available on the market; without that capability this end goal would be unlikely as ASIC design is in a category by itself in terms of difficulty. It may be possible to purchase a camera module such as the HackHD and create a PCB that processes the images coming from the camera module and integrates it.

Several components could be improved with the software side of image processing. If the algorithm ended up being written in OpenCV and deployed on a server with communications to the embedded system, it could be rewritten in C++ or Go instead of with Python bindings and gain a decent performance boost. The algorithm was written in Python due to the ease of writing and running Python, but interpretation is sub-optimal in terms of timing compared to C++ compilation and thus results in slower performance. Go would be pleasant to implement and compiles but bindings may not be available for all needed functions, in which case bindings would need to be written. Hardware acceleration could be developed for image processing; however, the code would need to be rewritten as single-threaded code does not translate into multi-threaded code.

No development of any kind of casing or housing for the embedded vision system was considered due to the low priority for a prototype that is exclusively tested indoors. However, for a real device to be used and presumably left outdoors, a casing is absolutely crucial for any kind of lasting solution, due to weather conditions as well as regular handling and usage. Time

permitting, a case would likely be developed for the embedded vision system using a CAD program like SolidWorks/Autodesk Inventor and printed through 3D printing services available at the university. The case, made out of PLA or ABS plastic and if it is relatively well designed, will be resistant to rain and weather conditions, dust/sand/physical irritants, electric shocks, non-malicious physical interaction, and remain cheap, reliable, and replaceable. If tolerances on a small print prove to be a constraint on developing a successful case at Duke, SLS or SLA printers should be able to provide a robust solution with the desired attributes listed above at the cost of price and printing and delivery time. If cost is a concern, a similar case could be laser cut from pieces of acrylic. Aesthetics would be improved from covering the case as well, which would improve the perception of professionalism of the system among potential users, and thus viability of the system on consumer markets would likely increase.

This system of bola detection ultimately uses only one type of sensor (vision); however, more sensors could be used in order to improve the accuracy of detection. The more methods to differentiate the bolas from other similar objects in the environment, the more likely that the system would be able to accurately count bolas. The embedded vision system could be complemented by the RFID system originally thought of, where when the thrown bola registers no acceleration, all bolas thrown before could transmit a signal to the bola detection platform and verify the embedded vision results. A UV-sensitive paint may cover the bola, allowing for easy detection by a UV-sensitive camera.

Fault tolerance should also be built into the embedded system in order to increase reliability. This is particularly important for this setup, since it is expected to be frequently moved, jostled, perhaps impacted with a missed bola throw, and endure harsh environmental conditions. While the proposed casing would lessen the impact of rough use, certain measures may be taken during the embedded system as well. Without a full understanding of fault tolerant embedded systems, mission-critical sensors may be duplicated in order to have at least one working sensor operating at all times while the other sensor can be maintained or updated, and error-correcting bits can be added to critical signals in order to ensure accuracy. These may be among the simplest fault tolerant mechanisms to implement, and likely the most feasible given the same budget and time frame.

Power usage is also a key factor in designing this system, as any system that requires more maintenance than the traditional ladder golf game in order to work properly is unlikely to be embraced by the intended audience. Solar panels, wind mills, and other sources of renewable energy can be incorporated in the structure in order to provide power during optimal days outside. Energy storage options like Li-ion or Li-Po batteries can be provided for short-term energy storage, although this solution may suffer from degradation over time as batteries get older.

Individual Contribution

Ying Wang

I contributed the majority of the software portion of the bola detection aspect of the project. In particular, I learned OpenCV and relearned Python for the project in order to get the computer vision aspect up and running. I learned or relearned both the concepts and the implementation of SLIC superpixel analysis, changing colorspaces, histogram equalization, bilateral filtering, Canny edge detection, Sobel/Laplacian/Scharr edge detection, colorspace conversion, template matching, Hough Line transform, Hough Circle Transform, adaptive and simple thresholding, Otsu and Riddler-Calvard thresholding, splitting and merging images into their respective channels, image arithmetic, and bitwise operations on images, among others. Some of these I implemented in my eventual function, and some were deemed unnecessary or unhelpful in the development of this particular algorithm and were so excluded.

Credits for significant contributors for the embedded vision portion of the project should be rendered to:

Professor Stacy Tantum for introducing the invaluable advanced statistics and DSP concepts without which my program would not be possible, such as 2D Fourier analysis for data compression issues as well as principles of the Hough Circles Transform, Hough Lines Transform, the HSV colorspace, Covariance matrix applications, Gaussian distribution traits, Mahalanobis distance, and image processing principles.

Professor Carlo Tomasi who introduced to me the integral image for template matching, the use of stereo vision for foreground separation and masking, and additional information about the Hough Lines Transform.

Dr. Adrian Rosebrock of University of Maryland, Baltimore for his book “Practical Python in OpenCV”, which taught me the principles of OpenCV and the basics of image processing within 48 hours as promised, and for his assistance and support through email.

Dr. Tomasz Malisiewicz of MIT CSAIL for information on template matching techniques and advice.