To: Dr. Satici, Nardos Ashenafi, Brian Higgins

From: Robotic Vision Team: Daniel Pullicar, Mason Cannon, Roscoe Ambrose, Haston LaGrone

Subject: Biweekly Update, 1/16/23 – 1/29/23

Date: January 29, 2023

Hello again professors. This is a short update on the progress our group has made since our last update.

# Work Completed

The final two weeks of January has brought with it some unique challenges and important discoveries for the team. A handful of what appeared to be straight forward problems where significantly more complex than we initially thought. The main issue we encountered was that adding a new camera with a higher framerate was not as simple as plugging it in. With that being said, there has been important progress on all the individual tasks taken on by the group.

Daniel Pullicar:

* Completed coding on Extrinsic Calibration
* Calibration data is now stored and recalled, so it only needs to be done once.

Mason Cannon:

* Completed looking at dilation and erosion filters to add into the in-code structure. They are useful but not for what we need. The program we have works just as well without the dilation and erosion filters added into it.

Roscoe Ambrose:

* Added functionality from Haston’s script to the main script that extracts total runtime, total frames captured, FPS that the program can keep up with, processing speed, and puck detecting accuracy.
* Added 3 more camera parameters to GUI (CAP\_PROP\_FRAME\_WIDTH, CAP\_PROP\_FRAME\_HEIGHT, and CAP\_PROP\_FPS).

Haston LaGrone:

* Implemented a version of the HoughCircles detection program that captures video at 60+ FPS.
* Discovered how to override the default API and other settings OpenCV uses when interfacing with web cameras.
* Confirmed that Canny Edge Detection can capture FMO’s.
* Acquired a web camera capable of capturing video up to 90 FPS.
* Can rudimentarily filter contours from Canny Edge Detection.
* Found the lost camera.

# Areas That Need More Attention

* High Frame Rate Detection

OpenCV utilizes a default API back end and video capture settings when using a web camera. In order to change these, and thus utilize a higher frame rate camera, the settings need to access by the program and modified in a very particular order. The default API needs to be toggled to match the API compatible with the camera (this can vary from one device to the next depending on its OS.) From this point the encoding format, frame dimensions, and FPS must be set in descending order. If any value in the list does not match a camera prerequisite or is out of this order, OpenCV will revert to default settings. These default settings will even override settings in the camera toggled through other applications. This issue has finally been tackled but consumed a considerable amount of effort to decipher.

* Image Point to Object Point Conversion

Some of the steps required to convert the puck’s image points to the world-frame object points are still hazy. This problem is being studied but may need some guidance from Dr. S. if we get stuck.

# Work Pending

The team’s focus in the coming weeks continues to be on creating the pieces needed to create a cohesive prototype of the detection system. Each group member will be continuing to focus on their assigned area of the project. The delay with the high FPS camera pushed the detection goals into this two-week period unfortunately. A functional version of the intrinsic calibration program has been completed and been put through initial testing. We believe that the coming two weeks will contain a significant amount of testing and debugging.

Daniel Pullicar:

* Create a calibration procedure that includes an accuracy-check.
* Convert Image Points to World Points.
* Work with Roscoe to add GUI elements to facilitate calibration.

Mason Cannon:

* Look more into the FOURCC and understand the compression ratios and which ones are useful and not useful.
* Looking into the canny edge detection function to understand how it works.

Roscoe Ambrose:

* Research what OpenCV settings FOURCC and CAP\_DSHOW are and how they work.
  + FOURCC may need to be added to the GUI, but not with a slider.
* Work with Daniel on adding calibration into the master script.
  + Add useful comments/print statements to show before and after comparison of settings.
* Explore CNN (Convolution-Neural-Network) with example from Nardos.

Haston LaGrone:

* Investigate program efficiency with new detection method.
* Investigate program accuracy with new detection method.
* Finish GitHub branch setup.
* Test overall functionality and practicality of a 60+ FPS implementation.

# Conclusion:

The last two weeks have not been as decisive as the team would have wished. With the detection method solution still not entirely clear, it is unknown how much more or less time will be required to successfully identify the puck. The ideal outcome is that the improved program that can utilize the higher frame rate handles the fast-moving object problem. With that out of the way the team can dedicate its efforts to an actual first pass at implementation. Many of the components are close to functional but need more testing to ensure they are effective.