Date: April 29th, 2019

From: Adam Kwok

To: Doctor Kaputa and anyone else interested in my work

Subject: Simulink Modifications and Tracking Faces and Tennis Balls using PyQT and OpenCV

Introduction

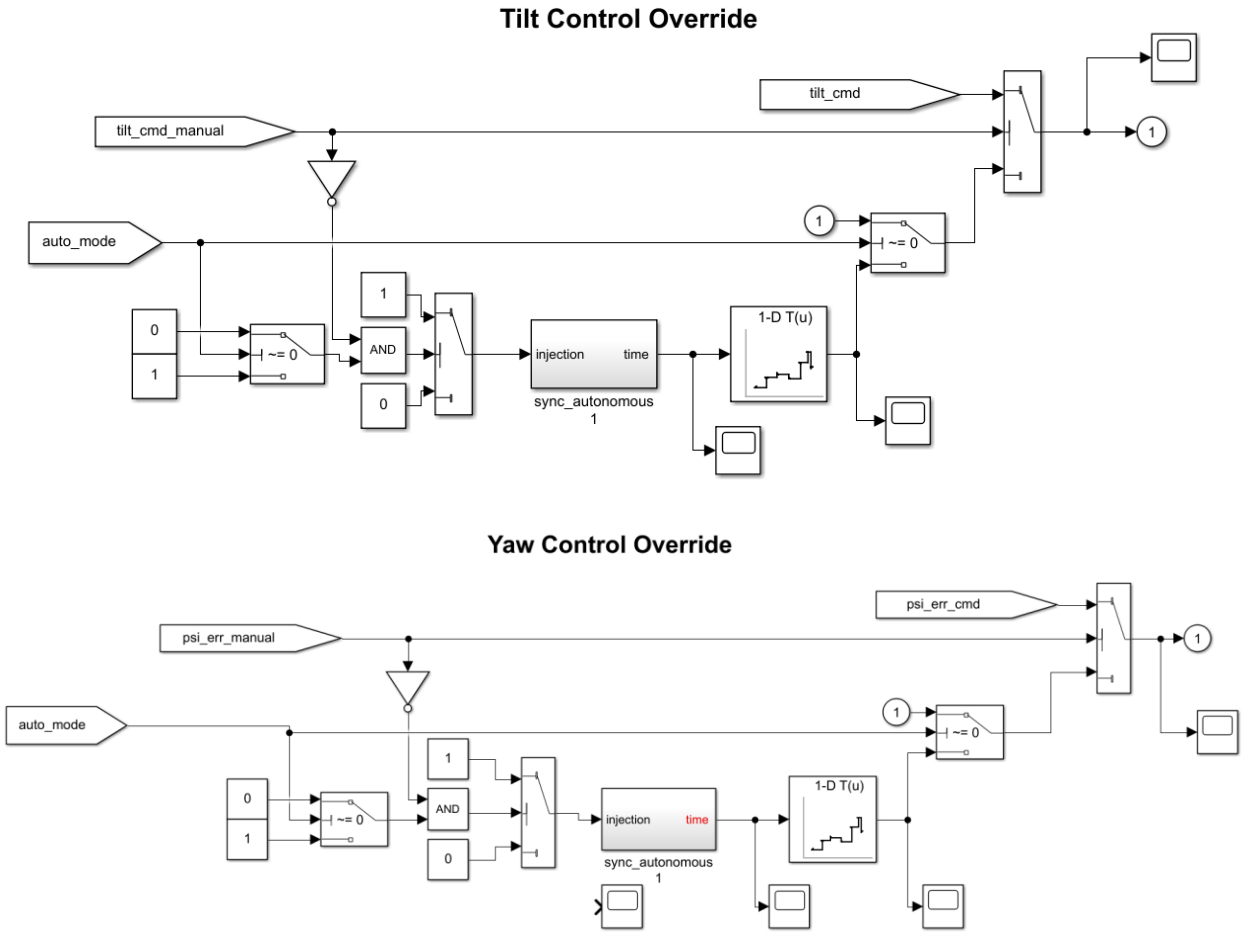
For the CPET-563 drone project, I utilized Simulink for communicating with the transmitter, ADC, camera outputs, and PWM AXI. PyQt4 running OpenCV was utilized for the detection and tracking of tennis balls and utilized a Haar cascade to track faces. The important factors in this project were to have maximum accuracy and results with the least amount of work. These requirements were to ensure the following:

- Minimal cost for the customer
- All requirements work and to an acceptable degree of functionality and accuracy

The below analysis investigates the limitations and error cases established and recognized as well as underlines the current functionality. This solution was designed to implement the controls for a Snickerdoodle based drone for manual control, autonomous navigation, and autonomous tracking of both tennis balls and faces.

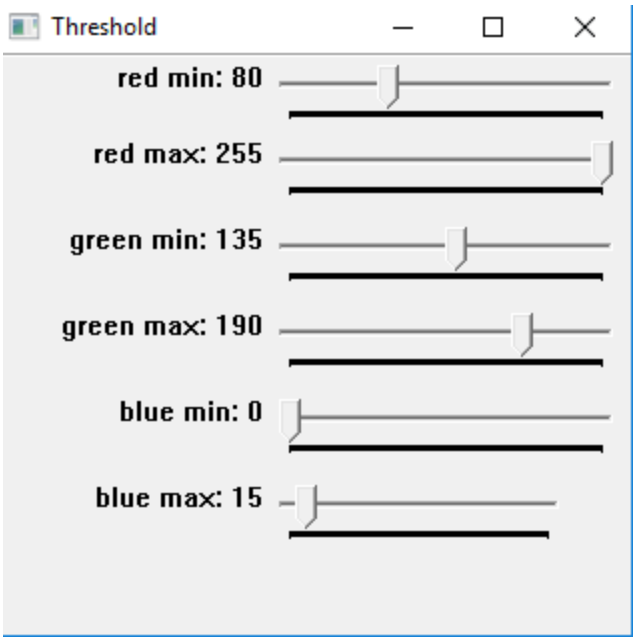
Analysis

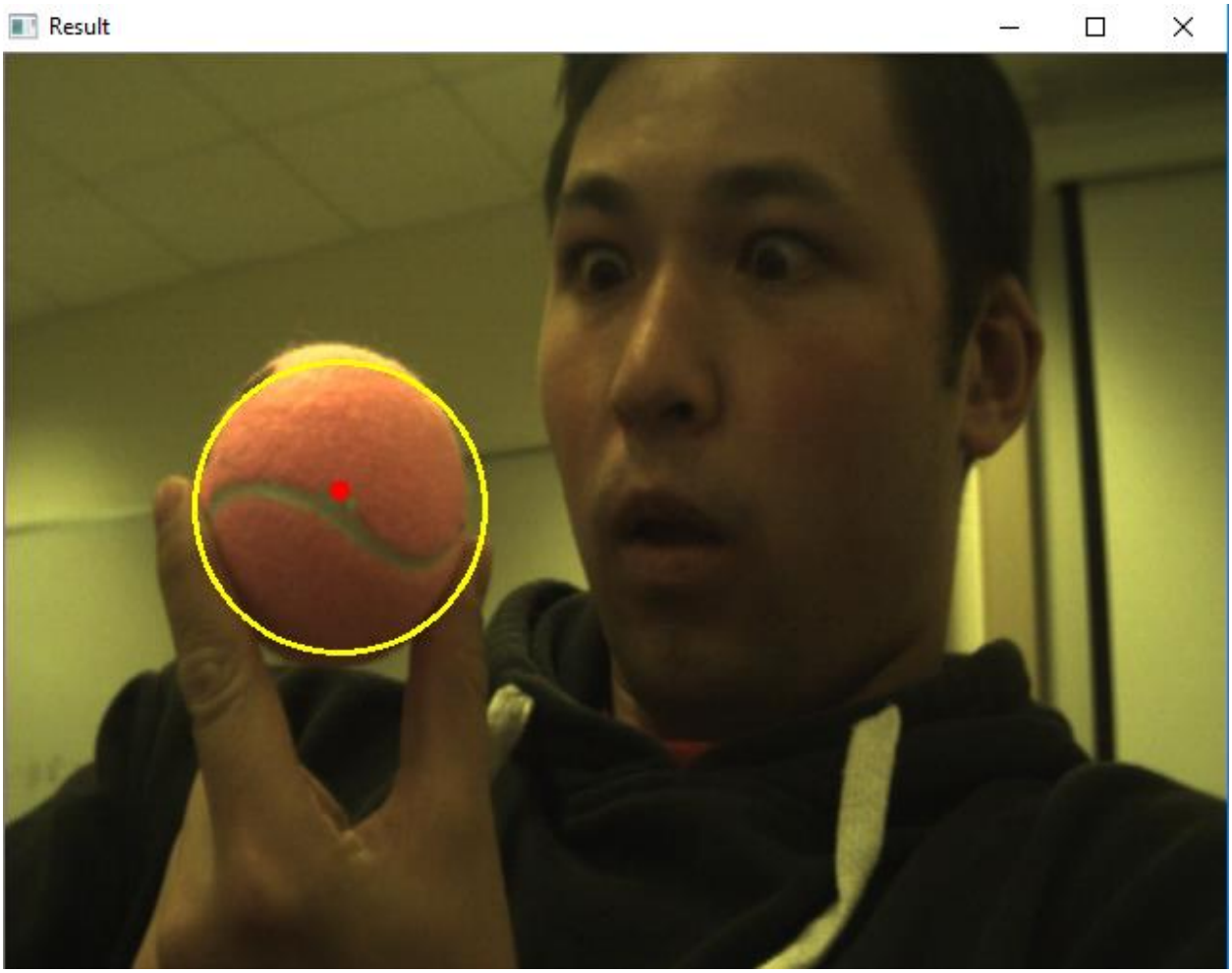
With the goals of minimizing cost while maintaining a high degree of functionality, the Simulink was modified from a previous design that was only configured for manual control and only had measures in place for tracking mode that needed modification for accurate movement functionality. On indication that the drone autonomous mode was in navigation mode and was also in autonomous mode instead of manual mode, the navigation mode instructions were passed to both the pitch and yaw commands.



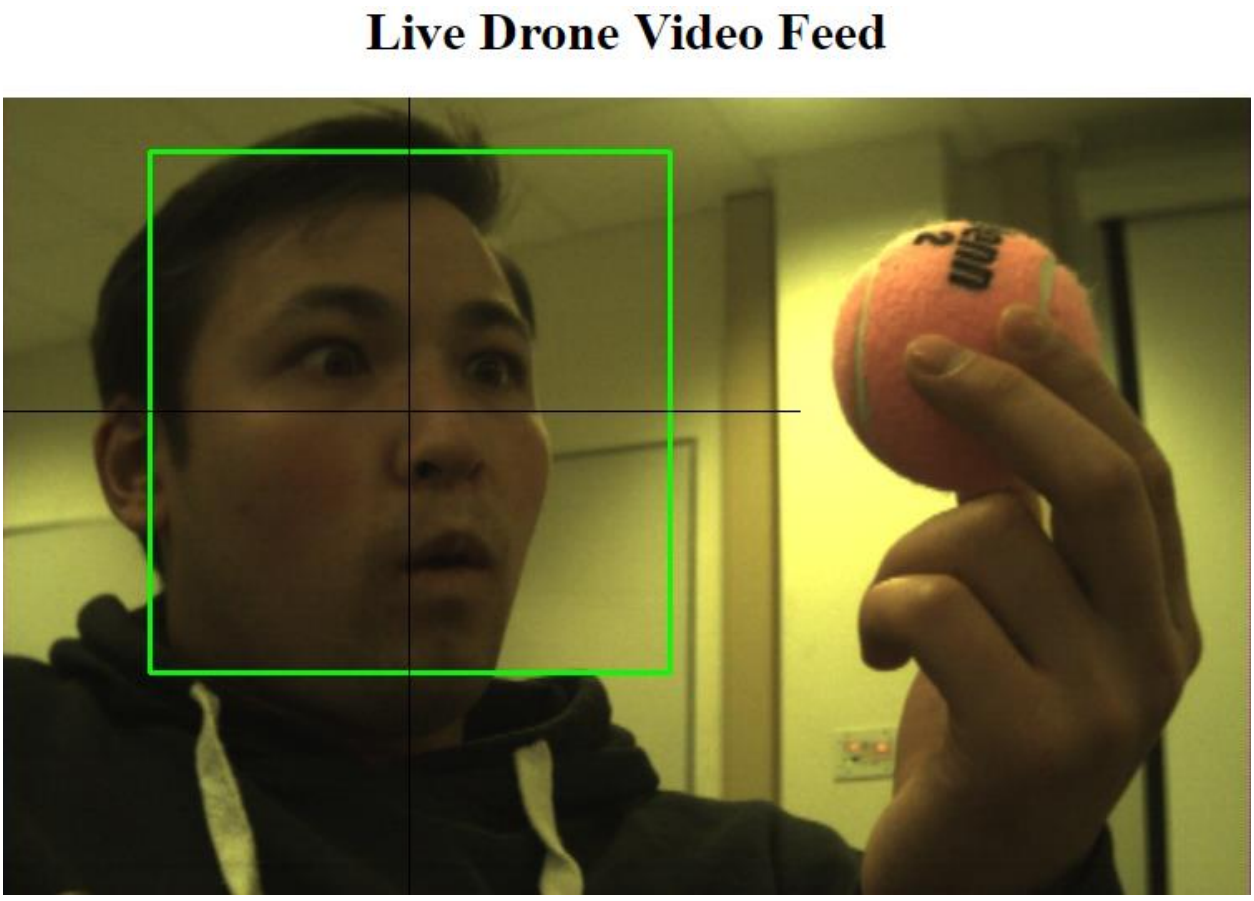
Other tweaks to the Simulink include modifying the low-duty cycle mode to account for faulty motors and modifying gain values associated with camera XY coordinate pitch and yaw instructions.

With the goal of detecting tennis balls, the OpenCV application was focused on homing in on color and shape. This meant that the minimums and maximums of red, green, and blue were established for the necessary shade of pink being tracked. These values were established utilizing PyQT4 software developed for a previous project and translated into the new tracking system in a tracking system that utilized the same Gaussian blurring as well as the HSV color filtering.


Threshold
red min: 80
red max: 255
green min: 135
green max: 190
blue min: 0
blue max: 15


Result

Face tracking was completed utilizing PyQT4 running OpenCV to run the image through a haar cascade and then utilized an equation based off of the Pythagoran theorem to select coordinates of the tracked face closest to the middle of the screen. Tracking was completed before the tracked image coordinates were sent to the Simulink referenced memory addresses and the tracked image was sent to a server for visual representation.



Conclusion

- Because the Simulink required little modification cost was driven low, however functionality can be improved and made more comprehensive with different priorities
- Because the code provides clear x and y coordinates for the centroid, the drone is provided with a single point to track which aids in each of clarity.
- Since error was found tracking when there was a similar color in the same image or when high-contrast conditions were present, environment should be taken into consideration for tracking
- Because of long-computation time for face tracking, computation being completed off-board or faster hardware will be required for better results
- Obscuring the tennis ball results in a skewed centroid, but still tracks the exposed portion well enough to be considered accurate