Autonomous Quadcopter Ball Catching Using Kinect Sensor Tracking

Computational Biomedicine Image and Modeling Center

RUTGERS Aresty Research Center for Undergraduates

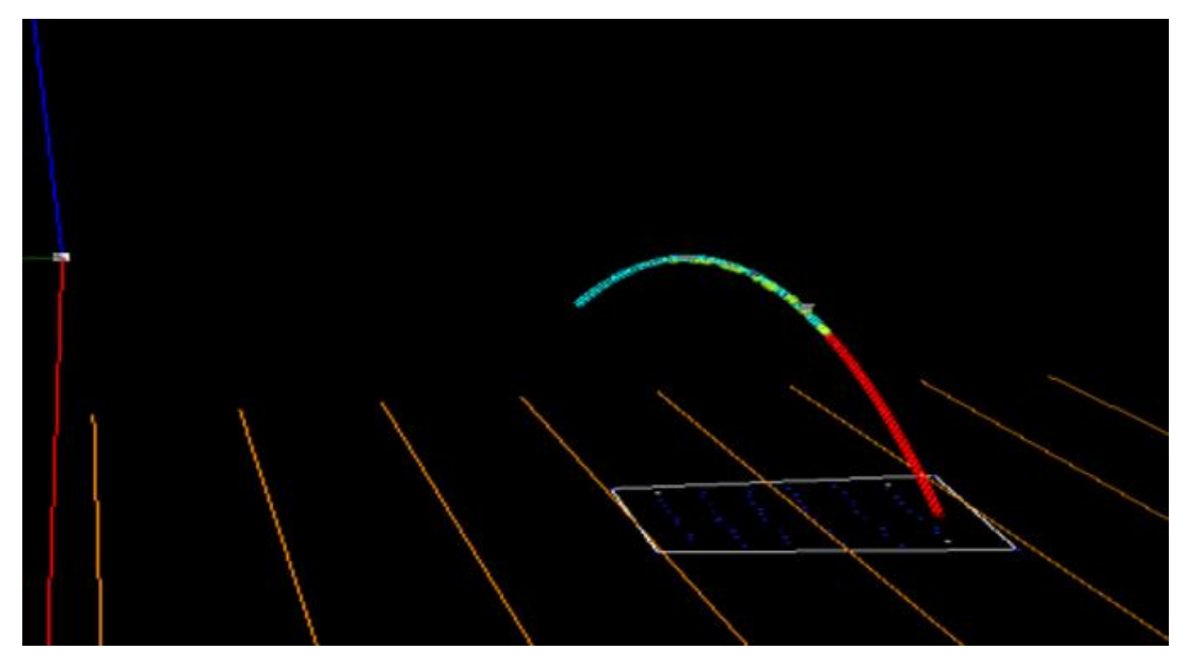
Alessandro Orsini
Mentor: Sejong Yoon

Abstract

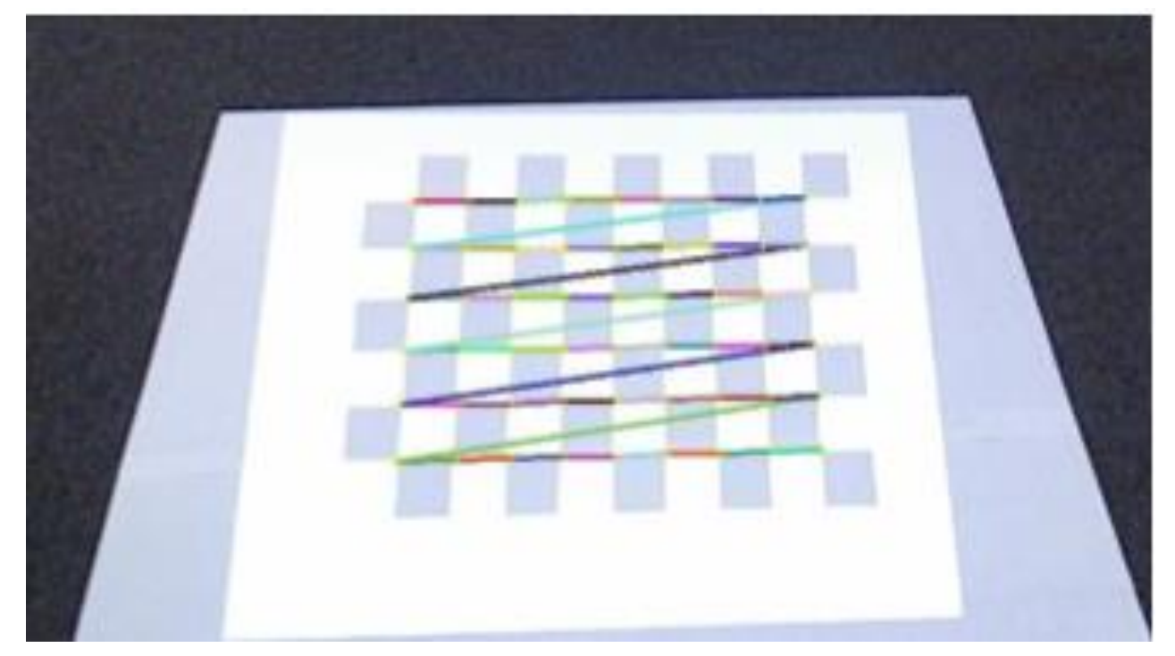
Quadcopters are Unmanned Aerial Vehicles (UAVs) whose simple design and ease of maneuverability make them optimal for autonomously performing tasks for use in the military, agriculture, and film. In this project we are working to create a seamless integration between a Microsoft Kinect sensor and a Parrot AR.Drone quadcopter to track objects to maneuver the quadcopter and complete the objective of catching a ball. To accomplish this, different software was developed and tested to quickly detect circular outlines, predict the trajectory, and efficiently move the drone. We were able to develop accurate ball tracking and drone movement, but determined a signal delay has been impeding the goal of catching a ball. We have collected data about predicted trajectory accuracy, drone reaction speed, and drone weight limit to determine the capabilities of our system and identify points where improvement is possible. The quadcopter was able to catch a ball in simple cases, where the ball flies straight up and then back down, but failed with more difficult trajectories due to the lag. In future work, the reaction time of the quadcopter would need to be decreased, through either improved software or improved hardware.

Background

This project builds on much of the work of Rutgers graduate John Bartos. When he stopped working on the project, John had developed a way to track both balls and the quadcopter. He also did trajectory estimation and predicted where active balls would land. Landing positions were displayed using a projector that was calibrated with the Kinect. Finally he got the drone to move smoothly using a PID controller.

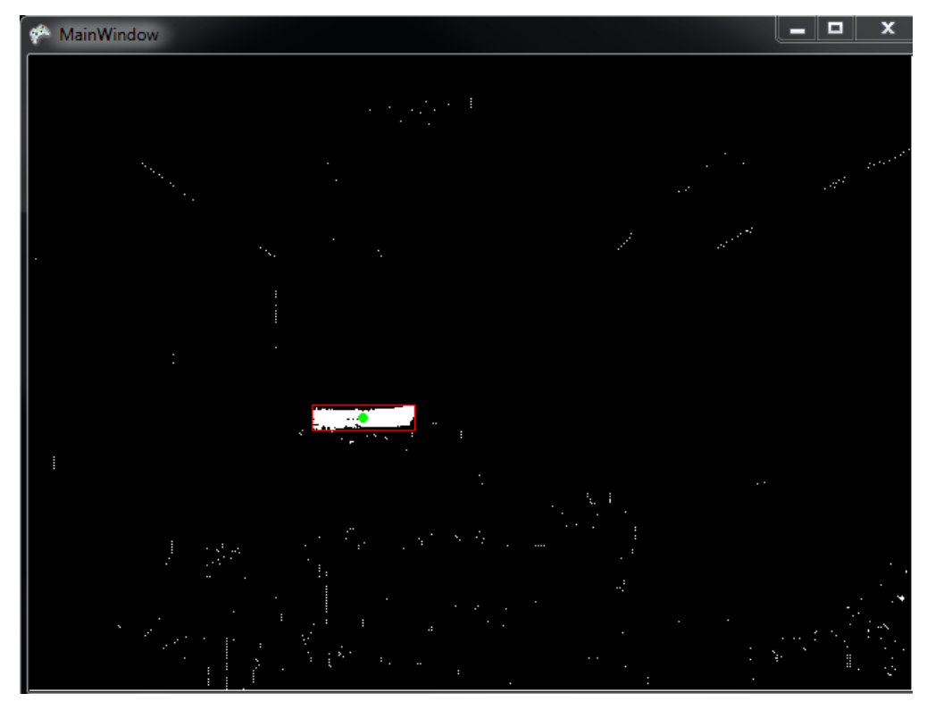


Trajectory Prediction is done when an active ball is present. Once enough frames are present a quadratic fit is done on the data to produce a trajectory equation.



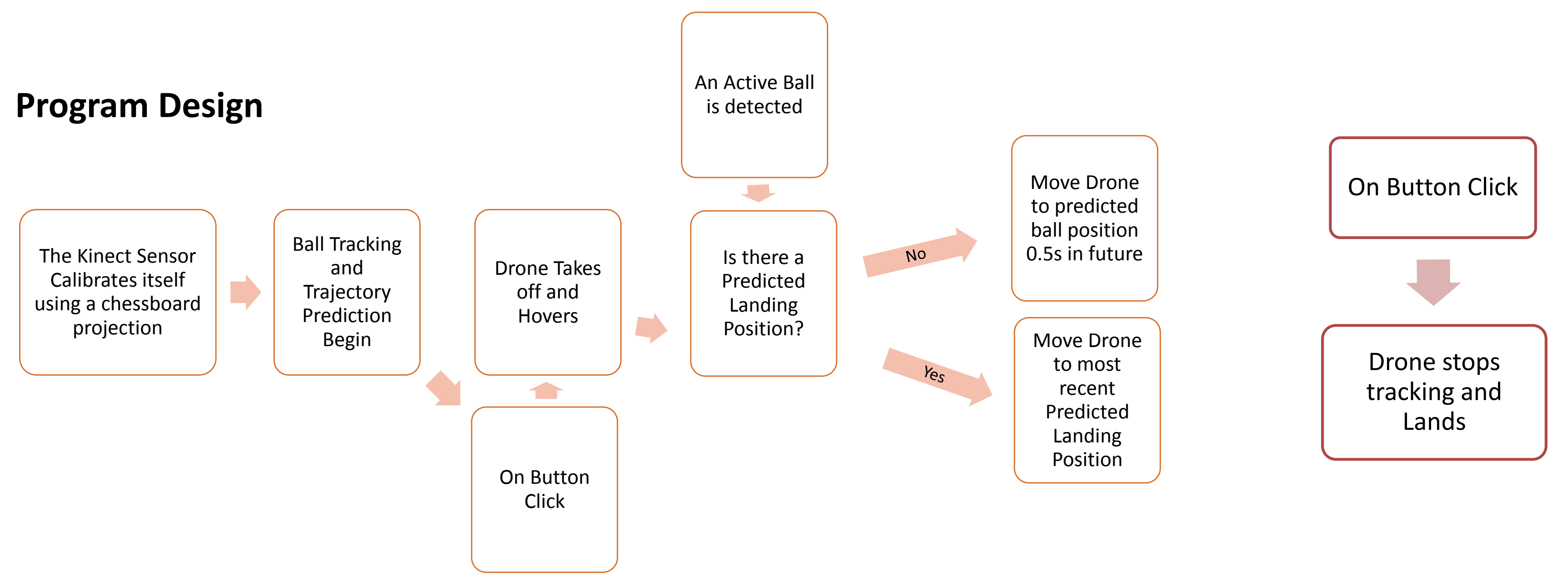
Calibration is done by first using the Kinect sensor to find the corners of the chessboard. The system is then calibrated for the projectors position.

Drone Tracking is performed using a distinct color of tape that is wrapped around the basket on the drone. The other colors are filtered out and the location of the blob is saved from the Kinect

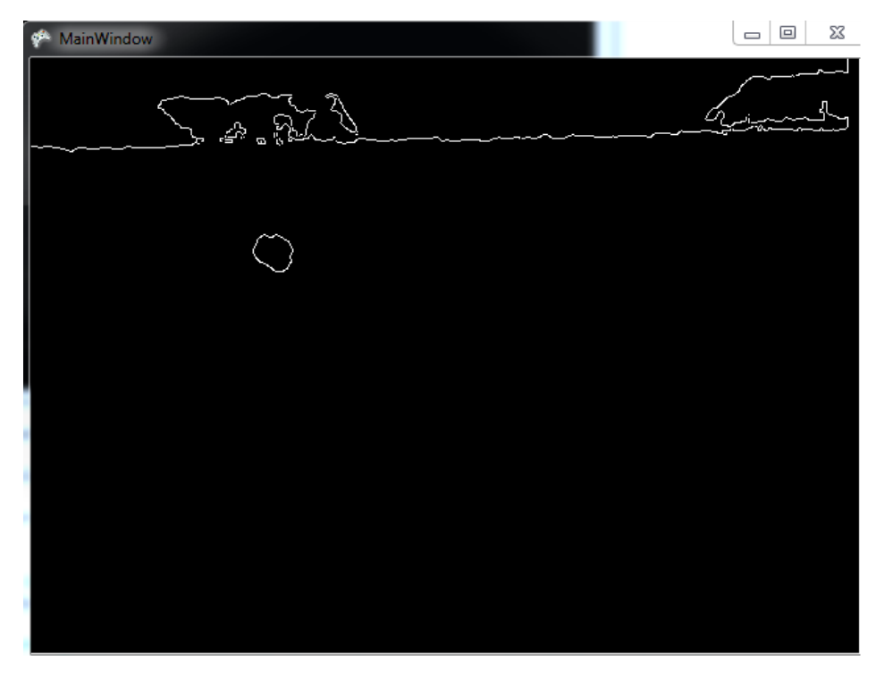


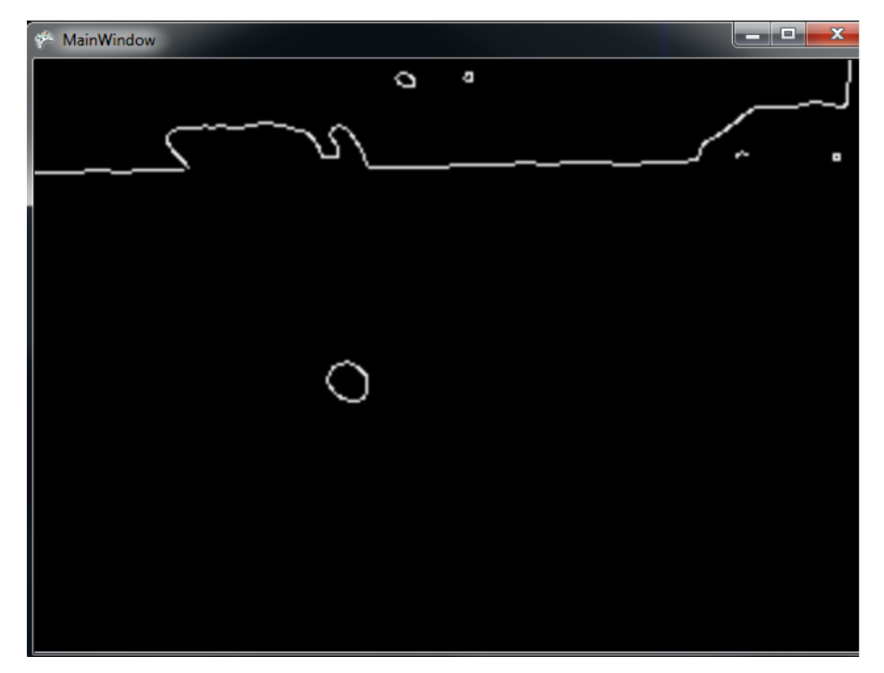
Methods

Program Design

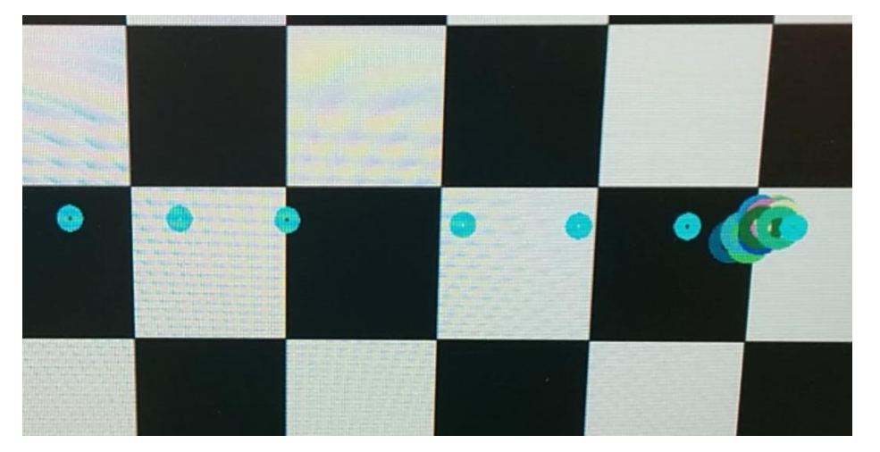

The Kinect Sensor Calibrates itself using a chessboard projection
Ball Tracking and Trajectory Prediction Begin
Drone Takes off and Hovers
On Button Click
An Active Ball is detected
Is there a Predicted Landing Position?
No
Yes
Move Drone to predicted ball position 0.5s in future
Move Drone to most recent Predicted Landing Position
On Button Click
Drone stops tracking and Lands

For **Detection**, the Kinect sensor captures image data and converts the objects in the frame to outlines and then checks if they are circular and within the correct width threshold to be a ball. Through usage, we discovered that the fluorescent lights were causing the Kinect to detect blobs that were not actually present. We were able to use noise reduction through Emgu CV to prevent these blobs from being detected and incorrectly tracked.


Ball Detection Before Smoothing


Ball Detection After Smoothing

To help visualize the ball location data that the drone would utilize to move to locations small blue markers were projected onto the floor for every ball detected. The predicted landing positions of the ball were displayed with larger circular markers.


Marker display being projected onto a chess board background

Adjusting the **Drone Movement** so the drone would move to catch a ball after it processed the data was a major component of this project. We changed our method of movement control several times.

- Initially we used a PID controller to move to a predicted landing position when it was calculated. This worked well, but was far too slow for ball catching because for drone to move smoothly it could only go at 25% its max speed.
- We changed the program, so the drone would move to every location of the ball detected until an accurate prediction was calculated. Then it would use the PID controller to reach the predicted position.
- We determined the PID controller was the biggest bottleneck and replaced it with straight-line drone commands at full speed.
- Due to the time it takes the drone to accelerate, the drone is sent to the ball's predicted position .5 seconds in the future.

Performance Data

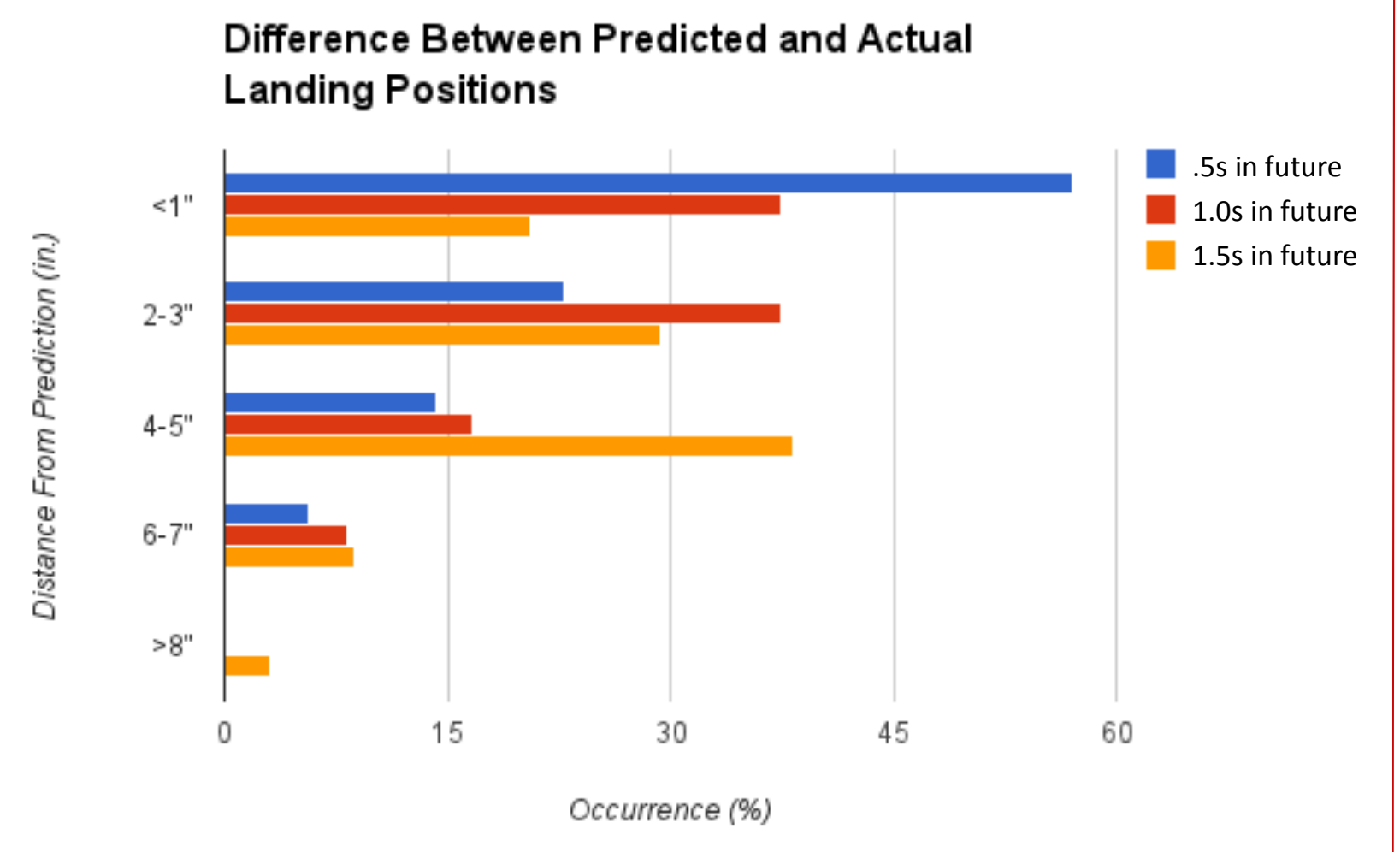
Time to Move Drone 1ft (in ms)

This test measured how long after receiving a command it took the drone to accelerate and travel 1ft.

Trials	12
Mean	933 ms
Min	901 ms
Max	987 ms
Range	86 ms
Std. Dev.	28.7 ms

Ball Prediction Accuracy (in.)

This test measured the accuracy of the trajectory prediction for 0.5s, 1s, and 1.5s in the future.



Max Ball Load of Drone

This test measured the carrying capacity of the drone the effects that carrying more balls has on it.

# of Balls	Flying?	Motor Strain?	Drifting?
1	Yes	No	No
2	Yes	No	No
3	Yes	Yes	No
4	Yes	Yes	Yes
5	Yes	Yes	Yes
6	Yes	Yes	Yes
7	No	-	-

Observations and Results

Tracking

- Accurate ball tracking system captured balls in environment for each frame received from Kinect. No "fake" balls are detected due to noise reduction in image.
- Each ball displays a small marker on projector board with its detected location.
- Range of colors filtered out was narrow enough to avoid tracking the wrong object, yet always found the location of the drone.

Movement

- Initially a Proportion-Integral-Derivative (PID) Controller was used to have the drone move smoothly to a specified location, but was too slow.
- A proportion controller used much less processing and was almost as accurate, but still had a slow max speed.
- We determined that the best results would come from going at max speed towards the target position and then using the hover command to stop
- For the drone to receive a command and accelerate for 1ft it takes on average 933 milliseconds, which makes it more difficult to reach the ball on time.

Ball Catching

- The drone was able to catch the ball in simple scenarios, such as when the ball went straight up and down or had a trajectory close to that
- When the trajectory was more complicated, the drone executes the correct commands, but does not reach the ball in time because when it changes direction the acceleration takes too long.
- The drone comes very close with large arcs that only require it to move in one direction, so with a few changes we hope to succeed with this scenario.

Future Work

Performance Improvement

- Improved tracking device such as Windows Kinect Sensor 2.0
- Updated quadcopter drone with improved signal response time and acceleration
- Larger testing room to give quadcopter more time to react

Additional Testing

- Additional testing that would be useful could be quantitatively measuring how close the drone was to catching the ball for different scenarios to see where improvements could be made.

References

[1] Bartos, John. "Kinect Tracking and Autonomous Control." Text. May 28, 2014.
[2] Obal Carnero. "cvBlob." Internet: http://code.google.com/p/cvblob/, May. 28, 2014
[3] Bradski, G. "OpenCV." OpenCV: Open Source Computer Vision. N.p., n.d. Web. 28 May 2014. <http://opencv.org/>.
[4] Arumugam, Prabu. "Image Denoising." Internet: http://codeding.com/articles/image-noise-removal , 28 May 2014.
[5] Balanukhin, Ruslan, P. 2013. The AR.Drone 2.0 controlling library for C#/.NET and Mono, with video support. https://github.com/Ruslan-B/AR.Drone.