The Control & Software Architecture of a Ball-catching Drone

Arijit Dasgupta, Arjun Agrawal and Jasper Tan

National University of Singapore ME4232: Small Aircraft and Unmanned Aerial Vehicles

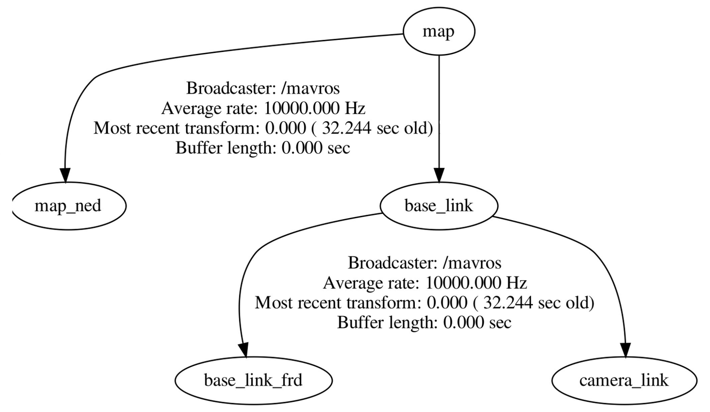
Introduction

The motivation of the project is to produce a versatile projectile detection and navigation stack for drones. In this study, we push the boundaries of research done in the area by removing reliance on motion capture and instead making use of an *onboard depth camera*. This project details the:

- conceptualisation,
- simulation, and
- testing of a projectile catching drone.

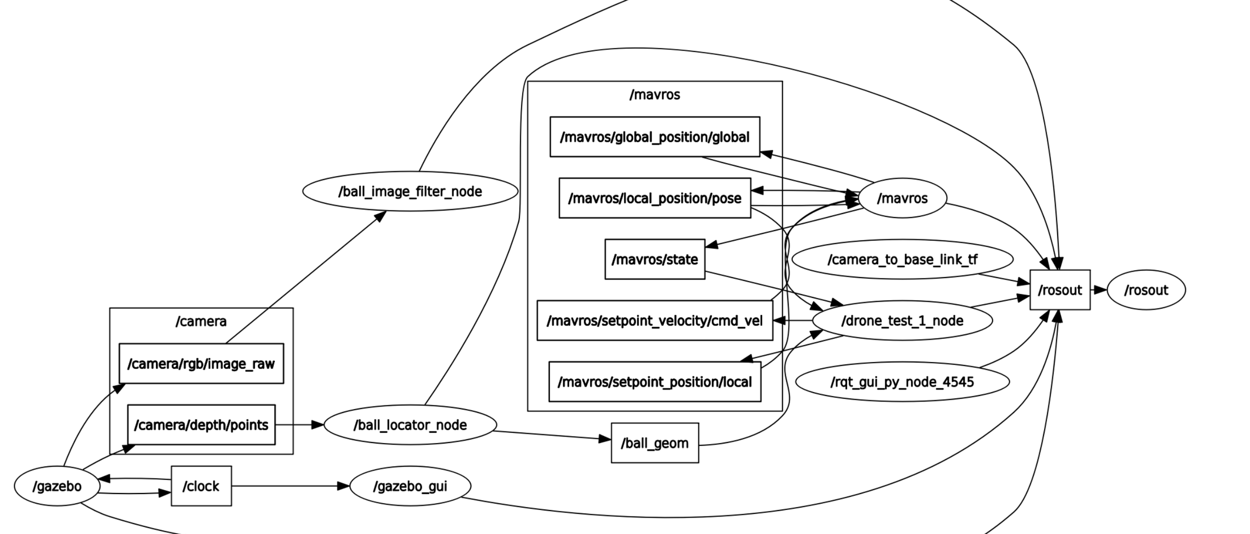
ROS is the primary software framework. There are three main functions of ball catching drone:

1. Object Detection: Colour and shape detection is used to detect objects. From the depth image, coordinates of the projectile are acquired to localise the ball.
2. Projectile Prediction: Trajectory prediction algorithm used to predict the ball's path



3. Path Planning: Used to plan an efficient path for the drone to intercept the ball

The TF tree and RQT graph shows how each function is used in the navigation stack:



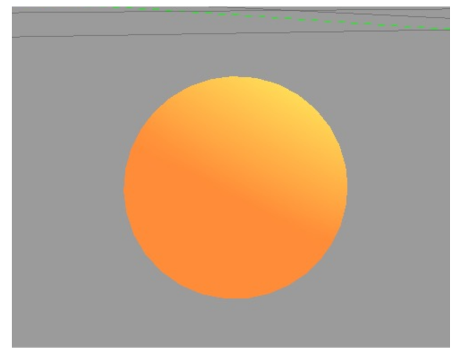
Object Detection

The image from the depth camera need to be segmented in order to identify and localise the position of the ball. In this project, the Realsense Depth Camera D435i was used.

Detection code was written in C++ and with minimal use of libraries to ensure operational frequency of at least 30Hz.

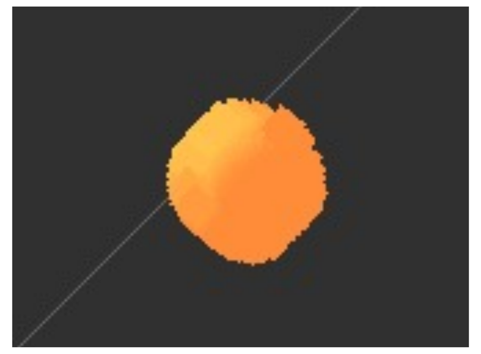
- Thresholding is used to create an array for points which fit the RGB criteria
- Points greater than two standard deviations away from the mean can be removed, as these points are likely background noise or ball frindges
- The mean X and Y position of the ball is found – centroid location
- The centroid pose is published as a pose message on ROS
- Frame Transforms (TF tree) is used to find the ball pose in the map frame

This node publishes the ball position (X_{Ball}, Y_{Ball}, Z_{Ball}).



Ball in Gazebo Simulation

Result of colour thresholding - Rviz Camera Frame



Result of colour thresholding + removal of noise - Rviz Camera Frame

Trajectory Prediction

In order to predict the future path of the ball, a path planning method is implemented using numerical methods based on basic kinematics and dynamic equations to predict and publish the trajectory of the ball. The process has three steps:

1. Finding the velocity: from the observed position of the ball (X_{Ball}, Y_{Ball}, Z_{Ball}), the velocity (U_{Ball}, U_{Ball}, U_{Ball}) can be measured. A regression and queue (link-list) data structure is used to calculate the ball velocity and perform calculations is real time.

$$U_{ball} = \frac{n\sum tX_{ball} - \sum x \sum t}{n\sum t^2 - (\sum t)^2}$$

2. Determine the acceleration: Two forces act on the ball, weight and drag, hence we need to first calculate the values of these forces.
   1. Weight: 0.026487N as net mass is 2.7g
   2. Drag (D_r): We need drag coefficient (C_D) and magnitude of velocity $|V|$
      - Drag Coefficient: Function of Reynold's number estimated at $\frac{VD}{v} \cong 2700$. With the equation below we can estimate $C_D \cong 0.420$.

$$C_D = \frac{24}{Re} + \frac{2.6(\frac{Re}{5.0})}{1+(\frac{Re}{5.0})^{1.52}} + \frac{0.411(\frac{Re}{2.63\times10^5})^{-7.94}}{1+(\frac{Re}{2.63\times10^5})^{-8.00}} + \frac{0.25(\frac{Re}{10^6})}{1+(\frac{Re}{10^6})}$$

      - Absolute Velocity: Use equation below

$$|V| = \sqrt{U_{ball}^2 + V_{ball}^2 + W_{ball}^2}$$

      - Hence, with the equation below we can find the value of drag (D_r)

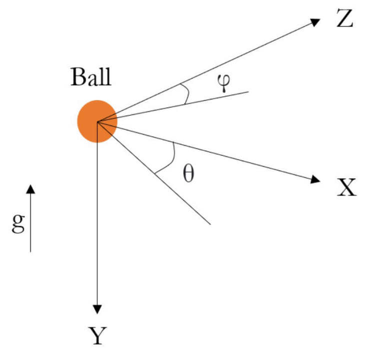
$$D_r = \frac{1}{2}\rho C_D |V|^2 A$$

   3. Finally, the free body diagram of the can be resolved to calculate the acceleration vectors of the body

$$a_x = -\frac{D_r}{m}\sin\phi\cos\theta$$

$$a_y = g - \frac{D_r}{m}\sin\phi\sin\theta$$

$$a_z = -\frac{D_r}{m}\cos\phi\cos\theta$$



3. The position and velocity of the ball is for the next time step set by the time step parameter t_{step} and is published.

$$X_{ball} = X_{ball} + U_{ball}t_{step} + \frac{1}{2}a_x t_{step}^2$$

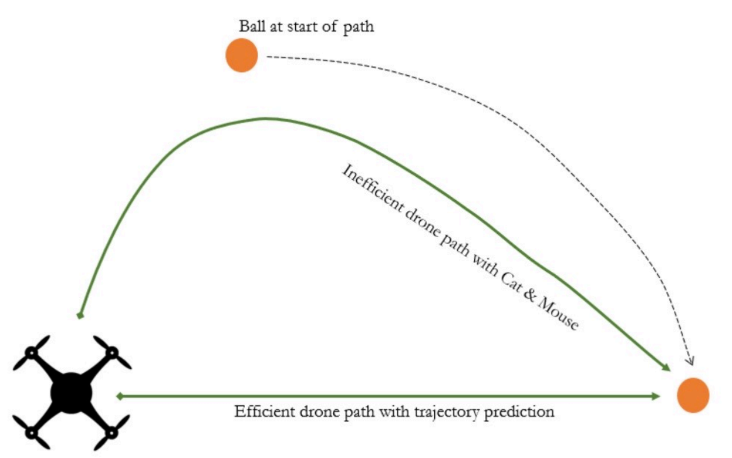
$$U_{ball} = U_{ball} + a_x t_{step}$$

Drone Path Planning

Three major methods were used to carry out drone path planning.

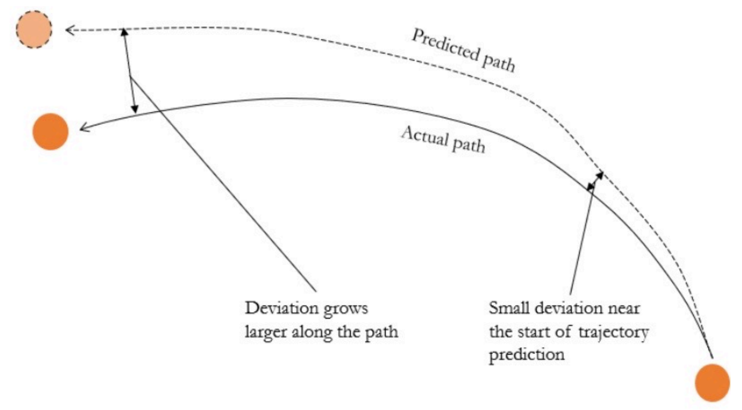
Method 1: Cat & Mouse

- The goal point of the drone is the current position of the ball
- As long as the ball is in the view of the depth camera, the drone will chase the ball
- **Con:** The method is computationally efficient but could be lead to inefficient paths



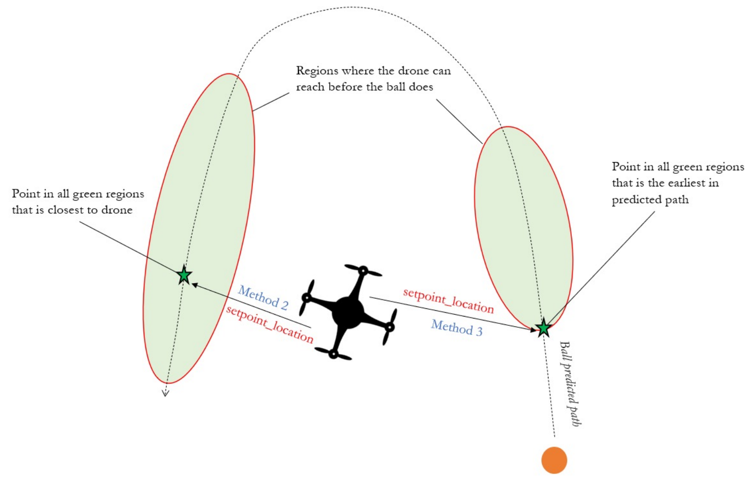
Method 2: Prediction with Shortest Path

- Developed to produce more efficient paths
- Method uses the drone's kinodynamic capabilities and the ball's *time-based* predicted trajectory to identify a point on the ball's trajectory the drone can reach *before* the ball
- **Con:** the trajectory prediction may have small errors which get magnified if the intercept point is far from the ball position. This problem can be alleviated as the ball's predicted trajectory is continuously recalculated allowing the drone to adjust inflight.

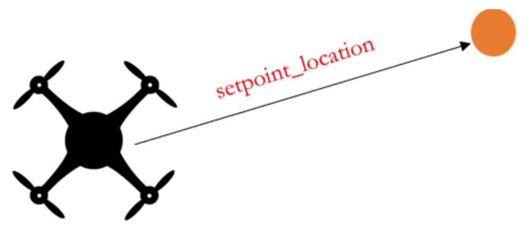


Method 3: Prediction with Fastest Path

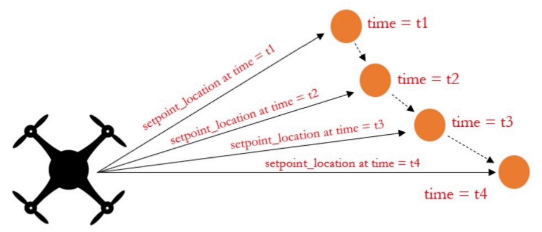
- Modified from Method 2
- path planner more aggressively chooses the closest point of interception
- Hence the Drone does not wait for the ball
- **Con:** Higher risk of missing and highly dependent on trajectory planner to be accurate



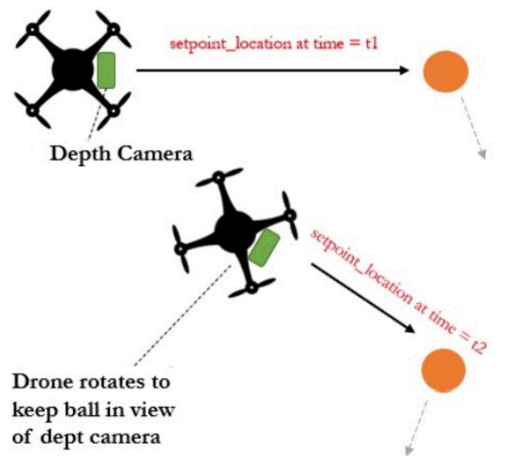
Testing Scenarios



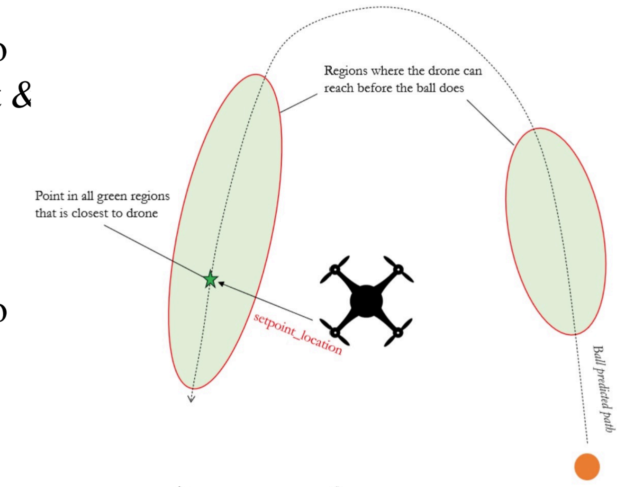
Scenario A: Drone path to stationary ball via the Cat & Mouse method



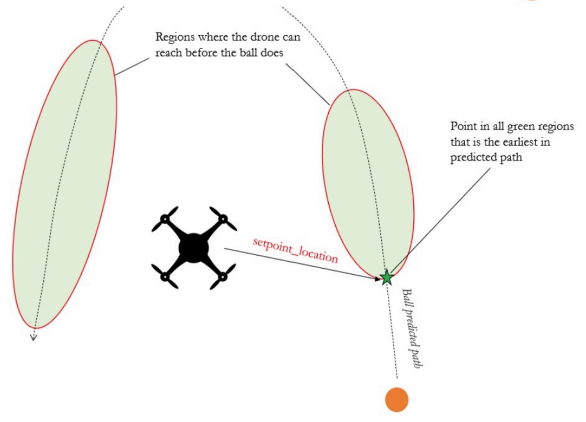
Scenario B: Drone path to moving ball via the Cat & Mouse method at different time



Scenario C: Drone path to moving ball via the Cat & Mouse method with yaw rotation



Scenario D: Drone path to a moving ball via the Prediction with Shortest Path method



Scenario E: Drone path to a moving ball via the Prediction with Fastest Path method

The simple Cat & Mouse method can tackle the first three scenarios. With the yaw control, the drone was able to keep the ball in its camera frame.

In order to tackle Scenario D the Shortest Path method was adopted.

Scenario	Method 1	Method 2	Method 3
A	✓	✓	✓
B	✓	✓	✓
C	✓	✓	✓
D	✗	✓	✗
E	✗	✗	✓

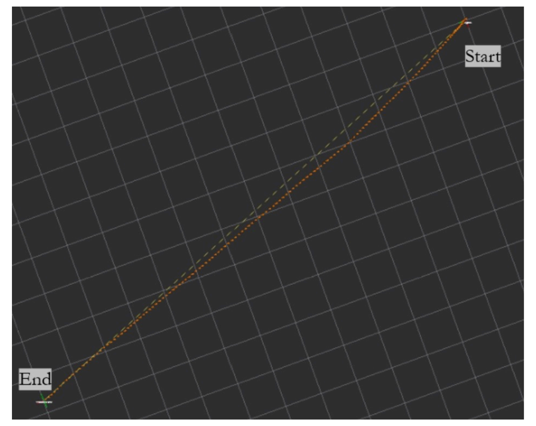
The table above summarises the methods used to tackle each of the aforementioned scenarios. All methods are able to resolve scenarios A, B and C. Method 2 can resolve scenario D, while Method 3 can resolve scenario E. Method 1 was unable to resolve either scenarios.

Simulation Testing

Scenario A

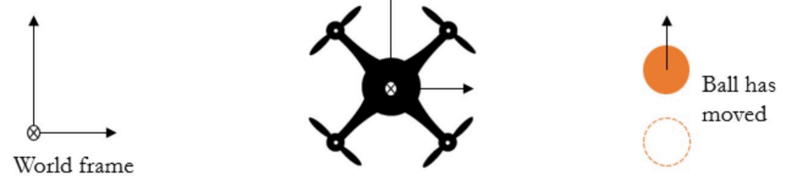
Provided valuable insights into the operation of the drone. Drone moves directly towards the ball placed directly at the centre of the field of view. However, once the drone pitches to accelerate the ball was often out of the camera frame. Two-prong approach was taken to resolve this:

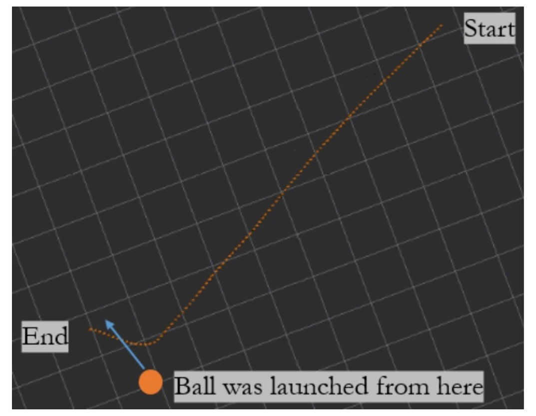
1. The acceleration of the drone was reduced
2. The drone continued to move in the direction of its last goal setpoint if the ball abruptly disappeared from the camera frame



Scenario B

The ball is placed a distance away and moves at a constant velocity across the camera frame.

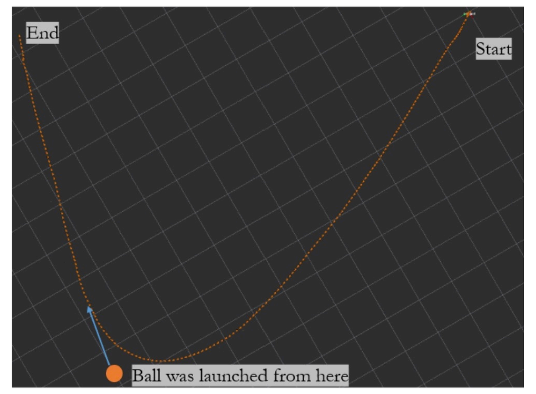




Drone moves towards the ball. It takes a sharp turn near the end to chase the ball but is unable to intercept.

Scenario C

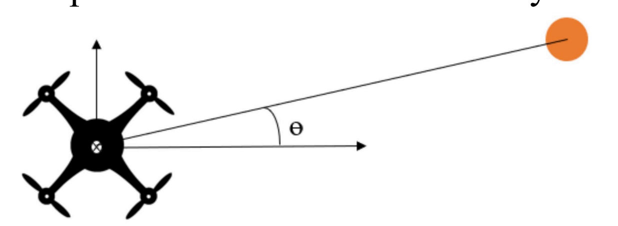
The drone is able to yaw, hence circumventing the issues faced in scenario B. The yawing motion is triggered by an increase in the θ angle beyond a threshold value. This will yaw the drone until the ball is back in the centre of the frame.

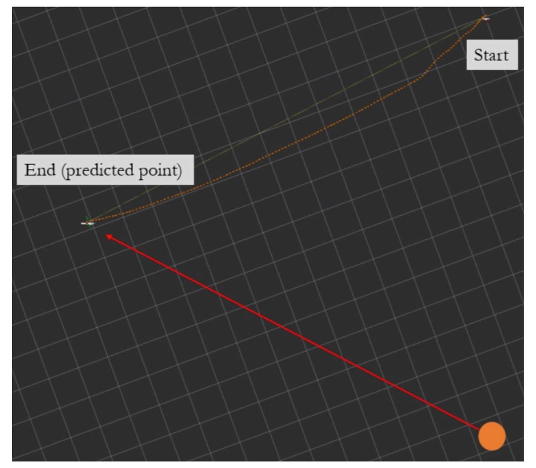


Scenario D

With the help of projective tracking and the drone kinodynamics, we were able to place the drone at a location before the ball reached that point

Here the drone takes a relatively straight path to a predicted point where the ball eventually ended up





Findings and Conclusion

We managed to simulate a drone which is able to identify and plan a path towards an object moving in a predictable manner.

A limitation of the depth camera was that it was unable to provide the object location with enough precision. The ball location had significant variance even when stationary.

The project has made good headway into the study of object catching drones.

In future iterations, more experimentation can be carried out to explore different sensor suites such as other depth cameras and lidars for the purposed of object detection and tracking. Along with the use of state estimators and particle filters to reduce sensor noise.

When considering hardware implementation, significant consideration needs to be given to the odometry system of the drone as the PX4 autopilot does not provide adequate performance in real world implantation. One interesting method to tackle this problem is the use of Lidar Odometry and Mapping.

Acknowledgements

We would like to express our gratitude to Dr Sutthiphong Srigrarom (Spot) for teaching us the fundamentals of UAS and drone technology, and exposing us to many aspects of unmanned aerial systems like drone racing, efforts by the industry and much more.