Quadcopter Control to Evade Obstacles

Xingyu Wang, Xinyue Shi, Gautham Narayan Narasimhan, Shubham Sarkar
wangxingyu369@gmail.com, xinyues@andrew.cmu.edu, gauthamn@andrew.cmu.edu, shubhams@andrew.cmu.edu

***Abstract*—We present in this report a robust obstacle avoidance system using the Crazyflie drone. We have shown that the Crazyflie can successfully dosge a beach ball thrown at it. This whitepaper presents a software framework using ROS Kinetic that is able to localise the Crazyflie in real-time using the Optitrack motion capture system. We have also implemented a PID controller and a Planner on our hardware framework to robustly dodge obstacles.**

***Index Terms*—Motion Planning, Multi-goal A*, realtime replan, RRT, motion primitives, PID Control**

I. Problem Description

Design of an obstacle avoidance system is critical for robotics application concerning autonomous vehicles, industrial robotics, defense and autonomy based industries. In this project, we limit our concern towards defense and civil rescue mission applications.

An interesting thing about these missions are that the drone are used for low altitude operations. At such low altitude, drone is expected to be have contact with multiple obstacles in its path. As these drone are autonomously driven Unmanned Aerial Vehicles (UAVs), control system based design must be used such that it avoids obstacles which have a probability of colliding with the drone surface. The design of obstacle avoiding system should be performed such that it avoids obstacle only if it is within the collision zone of the drone and continues its trajectory to reach its final point with minimum efforts.

II. Related Literature

Previous investigations in this domain has discussed multiple methods and sensors to devise obstacle avoidance systems such as using neural network based control system design incorporating laser and infrared sensors by Chi and Lee [1]. Also, Kim et al. [2] worked on using vision system based approach to detect the object if its more than 2 meters and uses ultrasonic sensor based edge detection of obstacle for less than 2 meters distance. Using optimization such as Mixed Integer Linear Programming method to obtain an optimal path for avoiding collision between multiple aircraft was proposed by Richard and How [3]. The work of Lin and Saripalli [4] focused on developing a real-time path planning algorithm which assisted UAVs to avoid collision with another aircraft considering uncertainty of motion of the other aircraft. They used reachability sets consisting the information of all reachable points to avoid obstacles at a given time. A method for avoiding collision of UAVs was developed by Zhiyong et al. [6] based on the minimum angle shift. Chakravarthy and Ghose [7], introduced a collision cone to detect the trajectory of an obstacle and to avoid it if its within the collision cone of the UAV.

Figure 1 shows an example figure of collision cone based approach used by Chakravarthy and Ghose [5].

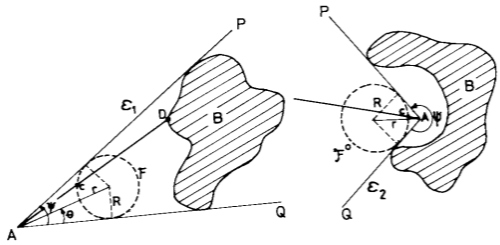

Fig. 1. The collision cone based method introduced by Chakravarthy and Ghose tries to map the edge of the obstacle by using tangents generated from a point to estimate the co-ordinate of the point it needs to reach to avoid collision[7]

III. System Modeling

The problem of designing an obstacle avoidance controller for Crazyflie can be decomposed into the following sub-parts: sensing, planning, and control. The sensing part is responsible for measuring the location and orientation of the drone and obstacle in 3D space. The planning part is used to predict the trajectory of the obstacle to determine if it will collide with the drone. If collision is predicted, the planner will generate a path for the drone to follow in order to dodge the obstacles. Finally, the controller is crucial to control the drone to fly to a reference point in space. The controller should be designed aggressively in order to dodge the obstacles as fast as possible.

A. Sensing System Modeling

In this project, the sensing of drone's location and orientation, as well as the obstacle's location, is solely handled by the OptiTrack Motion Capture Systems. Using this system, we modeled both drone and obstacle as rigid body objects in an open 3D space covered by OptiTrack cameras. It is assumed that the OptiTrack system's outputs are fast and accurate within the tolerance of the other subsystems.

B. Planner Modeling

In this project, the drone is modeled to be a point robot that can move omnidirectionally. The shape of the drone is modeled to be a sphere with a radius equal to the largest distance from its center to the edge of the propeller. The obstacle used in this project is a large beach ball that will be thrown by humans towards the drone. Once thrown, the

motion of the obstacle is assumed to be a parabola. Under these assumptions, the interaction between the drone and obstacle can be modeled as 2 spheres with one of them fixed in space and the other undergoing parabolic motion. The collision happens when the distance between the centers of the 2 spheres is less than the sum of their radius. The planner takes measurement data from OptiTrack system and outputs a reference location for the controller to follow.

C. Controller Modeling

The controller used in this project is Proportional-Integral-Derivative (PID) controller, which is one of the most common controllers used for quadrotor control. Linear Quadratic Regulator (LQR) and Model Predictive Controller (MPC) are considered during the proposal phase of the project. However, due to time limitation and the difficulty of effective system identification of quadrotor system, they are not used in the controller design. The controller of a quad-rotor consists of attitude control and position control. The state of the system can be defined as below.

$$s = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \\ \gamma \\ \dot{\gamma} \end{bmatrix}, u = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{1}$$

The quadrotor system is a nonlinear system. Comparing to design controller directly to the quadrotor system dynamics, designing a controller to error dynamics is easier and more convenient for a PID controller. Therefore, in the design section, controller design to the error dynamics will be discussed. The controller design of this problem is separated into attitude controller design and position controller design. In this case, the attitude controller is already provided by the PID controller in the firmware. The task is to design a PID feedback controller for position control of the drone.

Figure 2 shows an example of the PID feedback control block diagram. In the next section, detailed designs will be discussed about the implementation of the controller.

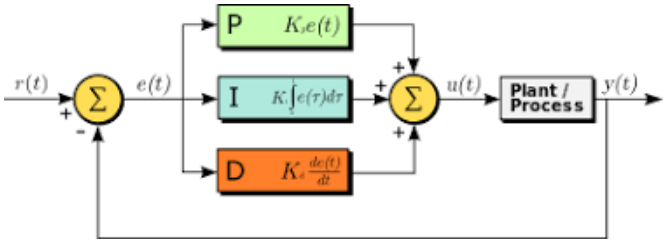

Fig. 2. PID Control loop

Figure 3 shows an example of errors.

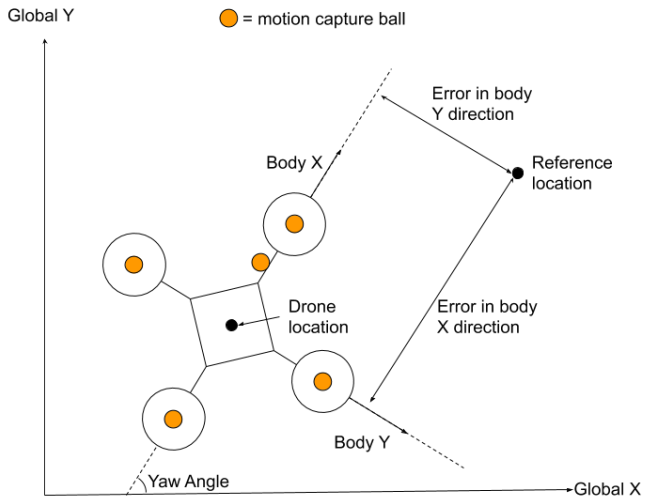

Fig. 3. PID control error in the Crazyflie's reference frame

IV. Controller Design

A. Controller Design

The position feedback from the motion capture system is in global coordinates. It provides us (x_g, y_g, z_g). The control is in body frame, so convert the global frame error to body frame error by the following equation.

$$e_{xg} = x_{des} - x_g \tag{2}$$

$$e_{yg} = y_{des} - y_g \tag{3}$$

$$e_z = z_{des} - z_g \tag{4}$$

$$e_x = e_{xg}cos(\gamma) + e_{yg}sin(\gamma) \tag{5}$$

$$e_y = e_{xg}sin(\gamma) - e_{yg}cos(\gamma) \tag{6}$$

Once the body frame error is known, a low pass filter is designed to get the derivative of the error $(\dot{e}_x, \dot{e}_y, \dot{e}_z)$. The sum of the previous error, $(\sum_{k=t_0}^{T} e_x, \sum_{k=t_0}^{T} e_y, \sum_{k=0}^{T} e_z)$ can be obtained by increment the previous error value. To prevent the integral term of PID to overflow the variable in computer, the summation is reset to 0 at some time step after a time period $T - t_0$ from t_0. The command outputs directly control the joystick commands to the drone through XBox360. Then a PID controller for hover control can be implemented as the following.

$$u_x = K_{Px}e_x + K_{Dx}\dot{e}_x + K_{Ix}(\sum_{k=t_0}^{T} e_x) \tag{7}$$

$$u_y = K_{Py}e_y + K_{Dy}\dot{e}_y + K_{Iy}(\sum_{k=t_0}^{T} e_y) \tag{8}$$

$$u_z = K_{Pz}e_z + K_{Dz}\dot{e}_z + K_{Iz}(\sum_{k=t_0}^{T} e_z) + C_{hover} \tag{9}$$

Notice the hover input bias term C_{hover} in the z-direction. Hover input term counteracts the effects of gravity, and is obtained by testing. Increment this value until the drone starts to takeoff gives the rough value of hover input.

When the quadrotor is required to fly aggressively. A feedforward controller can be added to the original feedback control loop. However, due to time limitation and hardware issues, this controller was not implemented. Here is a brief explanation for the feedforward section. A switch is designed to turn on the feedforward term when the drone detects the obstacle approaching, and the feedforward term will be activated for a short time to activate aggressive maneuver but later cutoff for stability.

B. Planner Design

The planner models the drone and obstacle as two spheres. The planner design focuses on predicting the obstacle trajectory under free fall assumption and generating a new reference location to allow the drone to dodge the obstacle while traveling the shortest distance. The planner takes measurement data about the location and orientation of the drone and obstacle from OptiTrack system. The planner then calculates the obstacle's speed and heading using current and previous measurement data. Once the distance between the obstacle and the drone is less than a threshold, the planner starts to predict the obstacle trajectory. Using numerical methods, the planner predicts the obstacle location at multiple time steps into the future. At each time step, the planner then calculates the distance between the two objects to determine if there is a collision. Once a collision is detected, the planner generates a new reference point for the drone to follow. In order to dodge the obstacles as fast as possible, the new reference point will be located perpendicular to the obstacle's heading direction. Once the obstacle flies past the drone, or if there is no collision, the planner sets the reference point back to the default reference point.

Figure 4 shows how planner generates a new reference point that results in minimum travel distance for the drone. In this project, two methods for generating new reference points were used. The first method generates the new point perpendicular to the obstacle's velocity direction only in the horizontal plane. That is, the drone will maintain its altitude when dodging. On the other hand, the second method finds the perpendicular direction in 3D space. That is, the drone will change its altitude during dodging maneuver. In this project, the performance of these two methods was tested and compared in order to show which one is more effective.

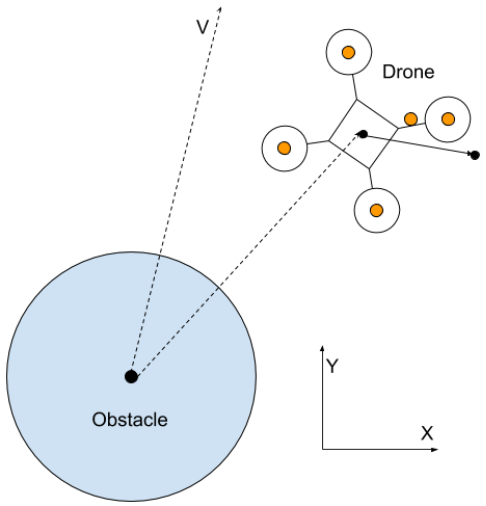

Fig. 4. Top view of how the planner works
This diagram shows the planner generates a new reference location for the drone to dodge the obstacles. In order to travel for the shortest distance, the new reference location has to be located perpendicular to the direction of obstacle's velocity vector. Two different methods were used and compared in this project. One method only finds the new location at the same horizontal plane while the other method allows the new location to be at different height. The new location is calculated by projecting the position vector from obstacle to drone onto the obstacle's velocity vector by using dot product. Then the perpendicular direction is found by subtracting the projection vector from the position vector. If using the 1st method, the z coordinate for all vectors can be ignored during calculation.

V. Simulation

A. Prediction Algorithm Simulation

The performance of the planner was simulated in Python in order to explore its correctness and tune its parameters. In this simulation, we explore the the accuracy of the prediction algorithm by comparing the predicted trajectory with the actual trajectory of the obstacle thrown by a human. The actual trajectory data was collected by throwing the obstacle into the air and use OptiTrack system to record its position information while it is doing free fall motion. After collecting the data, we used Python to predict the obstacle trajectory by using only the first few points of the actual trajectory data. The prediction algorithm is as follows. Given the first few points of the actual data, we assume that the obstacle is in free fall motion. We used linear regression to find the velocity vector that best fits these data points. Then, we predict the obstacle position into the future using numerical methods with very small time steps. Finally, we plot both trajectories together in a 3D plot for comparison, which can be seen in figure 5.

From the figure, it can be seen that the actual obstacle trajectory is somewhat tilted and does not undergo a perfect parabolic motion. This is possibly due to the fact that the obstacle is a light-weight ball with large radius. The aerodynamic force may play an important role during the motion. Therefore, there is a significant difference between the predicted and actual motion of the obstacle as the trajectory gets longer. As a result, re-planning is needed at each short time interval in order to correct the error and improve prediction accuracy.

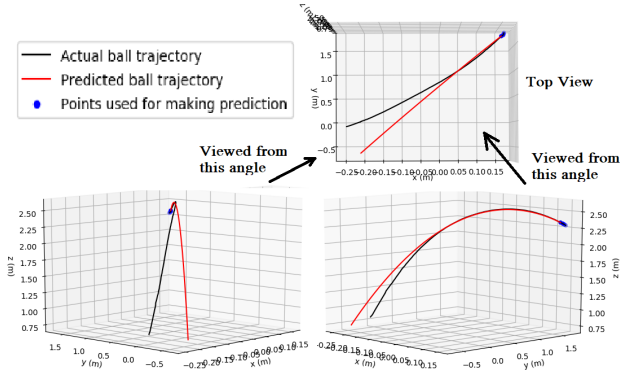

Fig. 5. Comparison between the predicted and actual obstacle trajectory
The actual obstacle trajectory does not undergo a perfect parabolic motion due to its shape and weight. There is a significant difference between the predicted and actual motion of the obstacle as the trajectory gets longer. As a result, re-planning is needed at each short time interval in order to improve prediction accuracy.

B. Obstacle Avoidance Algorithm Simulation

In this simulation, the planner was given the location data of the drone and obstacle in order to simulate the obstacle avoidance algorithm. Assuming the predicted trajectory of the obstacle is correct, we visualized the result in a 3D plot to show the proposed new reference point using both dodging methods discussed in the previous section. During the simulation, many parameters were carefully tuned in order to achieve maximum calculation efficiency. These parameters include the radius of the obstacle and drone, the number of time steps predicted into the future, and the distance between the old and new reference point. The simulation result for a sample scenario can be seen in figure 6.

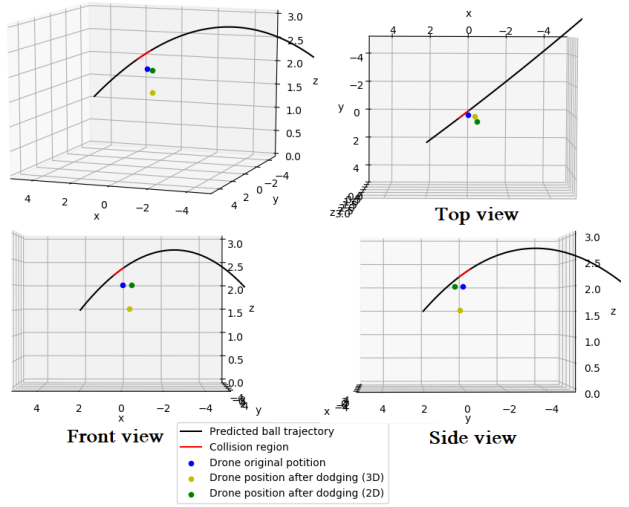

Fig. 6. Planner simulation for obstacle prediction and dodging
All units are in meters. This visualization shows an example of the planner predicts the obstacle trajectory and generates new reference points to avoid collision. The reference points generated using two different methods are both shown in this figure. The yellow dot represents the new reference location when the drone is allowed to change its altitude. The green dot shows the new reference location at the same height as the old reference point.

From the result, it can be seen that the planner is able to use the predicted obstacle trajectory to find a new reference location for the drone. The red part of the trajectory represents the collision region, and the yellow and green dots represent the new reference location that make the drone to avoid collision while traveling the shortest distance at the same time. The green dot shows the new drone location that is constrained to stay in the same altitude, and the yellow dots represents the new reference location without the restriction of maintaining altitude. From the simulation, it is clear that the planner is able to generate efficient plans to dodge obstacles.

VI. Hardware Implementation and Testing

A. Sensing Implementation

The project uses OptiTrack sensing system alone to get position data. In order to get readings correctly, both drone and obstacle has to mount motion capture balls or stickers. In this implementation, the drone has mounted 5 motion capture balls. Four of them are mounted on its 4 legs, and the last one is mounted in the middle of one of its legs that points towards the positive body x-axis. These 5 balls forms a rigid body in the OptiTrack system. The balls are arranged in such a way to make sure that the extra weight is distributed evenly on the drone, and there is enough space between each ball to allow the sensing system to have clear reading. In order to get consistent yaw angle reading, the motion capture balls are mounted asymmetrically on the drone. For the same reason, the obstacle is also mounted with OptiTrack stickers. Due to the fact that the obstacle is a sphere, it is not necessary to read its orientation information. Thus, the OptiTrack stickers are mounted on the obstacle symmetrically. The obstacle is also painted to cover its reflective surface and prevent false sensor readings. Figure 7 shows the actual sensing system used in this project.

Fig. 7. Hardware implementation of the sensing system
The drone has 5 motion capture balls attached to ensure correct measurement from sensing system. The motion capture balls are mounted asymmetrically on the drone in order to get consistent yaw angle reading. The obstacle is painted to cover its reflective surface and prevent false sensor readings. The obstacle is also mounted with OptiTrack stickers for position measurement

B. Software Architecture and Controller Implementation

We use the ROS Kinetic framework to control the Crazyflie. ROS abstracts away the burden of synchronizing messages from various sources, nodes and helps us focus on just writing the controller.

Our goal is to control the Pitch, Roll and Thrust of the Crazyflie. Yaw control is not required because we assume that

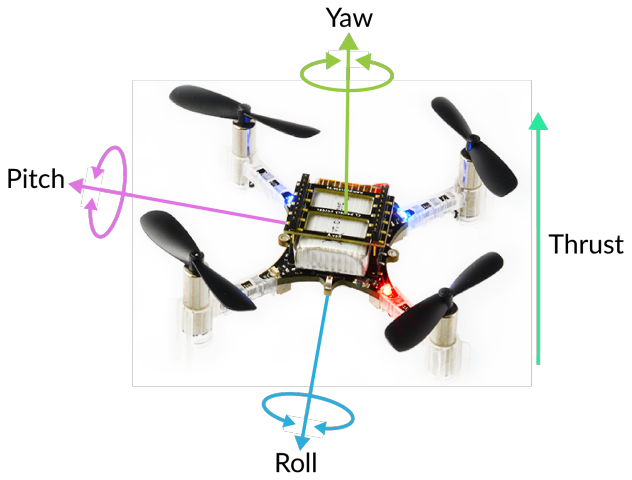

Fig. 8. Crazyflie Roll, Pitch and Yaw axis

the Crazyflie is always controlled in its own reference frame, hence according to the Crazyflie it is always pointing forward. In order to do this we compute the PID controller error with respect to the Crazyflie's reference frame and not the global reference frame.

We use a PID controller to provide higher level control commands to the on board controller on the Crazyflie. The message passing happens through the /cmd_vel rostopic. Hence we can simply send the joystick command and the Crazyflie action server will pick it up and forward it to the drone using provided drivers by BitCraze. There are other methods to provide control inputs to the Crazyflie such as using the BitCraze python API, however we found that sending joystick commands was one of the easiest things to get up and running and was fairly robust for our application needs.

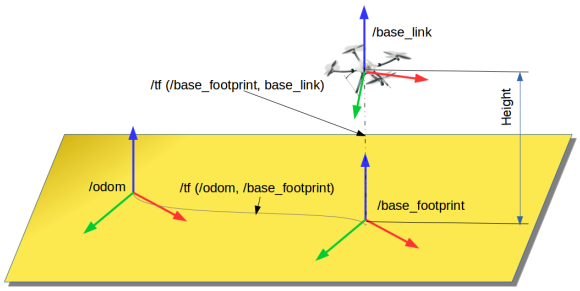

Fig. 9. ROS TF frames for Crazyflie and base

The ROS framework uses a package called TF to sync up the reference frames between various ROS nodes and the transforms between them. Hence we can hand over transformation computations such as converting between Euler angles and quaternion representations of rotations to this package.

The localization of the Crazyflie is performed using the

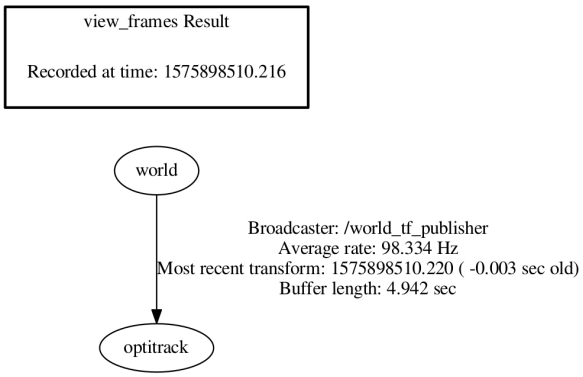

Fig. 10. ROS TF transform and topic for Optitrack data

Optitrack Motion Capture System. This system using multiple IR cameras to track IR reflective markers in a given indoor space. As long at least three markers are visible, the Optitrack system can triangulate the position of a rigid body and provide the coordinates of the centroid at 100 Hz. We again use ROS to transfer these coordinates between the ROS master running on the Optitrack computer and local computer running our PID controller.

We use a ROS subscriber to listen to these messages and will know the state of the Crazyflie. We use this for computing the L1 error between the target position coordinates and current position coordinates independently in X and Y coordinates.

Once the PID computes the linear velocity required to move the Crazyflie, it send these values over the /cmd_vel rostopic to the Crazyflie action sever.

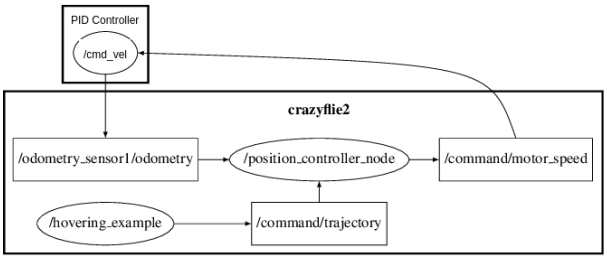

Fig. 11. Crazyflie on-board controller ROS flowchart

The Crazyflie action server is constantly listening for control inputs over the /cmd_vel topic. As soon as it gets a message, it uses the Crazyflie radio to broadcast the message using the provided 2.4 GHz radio transmitter. This is a commonly used Tx-Rx setup to transfer joystick messages between ground station and drones as it provides sufficient bandwidth at the same time having little attenuation of the signals.

Once the Crazyflie on-board ROS node receives the joystick command, it uses the on-board position controller node to compute motor speed values. The position controller node uses its own PID controller to compute the motor Pulse width

modulation (PWM) amplitude and frequency.

In summary, the controller loop starts with receiving the position coordinates of the Crazyflie from the Optitrack motion capture system, local computer has a PID controller that provides the required velocity to reach target. The on-board micro-processor then uses these values to control the Motor PWM using another on-board PID.

C. Testing Method

In the testing phase, controller design was tested along with the path planning algorithm. According to the program, the drone was expected to evade the obstacle based on the trajectory of the ball. We used two point mapping to predict the expected trajectory of the obstacle (ball in this case). Hence, during the testing phase, the obstacle as thrown in all directions and the ball used to choose an escape point from the obstacle's trajectory if there is a probability of collision. After choosing the escape point co-ordinates, the drone moves in that direction to evade collision.

VII. Demo Results

Testing was performed for three dimensions, where the drone displayed accurate results for horizontal, diagonal, vertical evasion from the obstacle. Limited performance was observed in the evasion when the ball was directly dropped on the drone from top or thrown up towards the drone. We analysed the reason and found out that this could be solved by using much better trajectory prediction as like using more points to predict much accurate trajectory of the obstacle.

The performance of our program could be observed by using the following YouTube link, where we have documented the drone performance to evade obstacles:

https://www.youtube.com/watch?v=cF44kJlIQ-8

Also, cases of failure are also documented along with some more successfully evasion of the obstacle:

https://www.youtube.com/watch?v=cF44kJlIQ-8

VIII. Conclusion

The demo results indicate that the our controller and path planning algorithm has achieved baseline performance. We performed testing and the results helped us conclude that the path planning could be improved considering more points for prediction of trajectory, and there is a small delay, leading to the drone crashing if the obstacle is projected at high speed. Also, the controller performance could be improved to achieve much more aggressive control and evade objects with minimum efforts. Having discussed all the shortcomings, the controller design performance was tuned to ensure there is a good aggressiveness in evasion obstacle. The two point based trajectory prediction helped us evade objects in most of the directions of the obstacle projected and thus achieved satisfactory results. Hence, the documented testing and demo results through the video links, support our claim and the performance was verified by multiple tests. Inference can be drawn from this that the problem of avoiding obstacles at lower altitude for drones was thoroughly investigated and designs were built to achieve the goal within which baseline design performance is observed and meets the basic objective of the project. There is a scope for more critical work in future where the shortcomings of this project could be studied and solution could be determined.

References

[1]Chi K.H., Lee M.R., 'Obstacle Avoidance in Mobile Robot Using Neural Network. Proceedings of the International Conference on Consumer Electronics', Communications and Networks (CECNet), Xianning, China. 11–13 March 2011, pp. 5082–5085.

[2]Kim P.G., Park C.G., Jong Y.H., Yun J.H., Mo E.J., Kim C.S., Jie M.S., Hwang S.C., Lee K.W., 'Obstacle avoidance of a mobile robot using vision system and ultrasonic sensor', Comp. Sci. (LNCS) 2007, 4681:545–553.

[3]Richards A and How JP., 'Aircraft trajectory planning with collision avoidance using mixed integer linear programming', Proceedings of the American control conference, Vol. 3, Anchorage, AK, 8–10 May 2012, pp. 1936–1941. IEEE.

[4]Lin Y and Saripalli S., 'Collision avoidance for UAVs using reachable sets', 2015 International conference on unmanned aircraft systems (ICUAS), Denver, Colorado, USA, 9–12 June 2015, pp. 226–235. IEEE.

[5]Budiyanto A, Cahyadi A, Adji TB, et al., 'UAV obstacle avoidance using potential field under dynamic environment', International conference on control, electronics, renewable energy and communications (ICCEREC), Bandung, Indonesia, 27–29 August 2015, pp. 187–192. IEEE.

[6] T. Bresciani, "Modelling, Identification and Control of a Quadrotor Helicopter", Master's thesis, Lund University, Sweden, 2008.

[7]Zhiyong X, Xiuxia Y, Yi Z, et al., 'Study on dynamic guidance obstacle avoidance of UAV based on the minimum angle shift', Int J Sec Appl 2016; 10(3): 393–404.

[8]Chakravarthy A and Ghose D., 'Obstacle avoidance in a dynamic environment: a collision cone approach', IEEE Trans Syst Man Cybern 1998, 28(5): 562–574.

[9] S. Bouabdallah, A. Noth, and R. Siegwart. PID vs LQ control techniques applied to an indoor micro quadrotor. In International Conference on Intelligent Robots and Systems 2004, volume 3, pages 2451 – 2456 vol.3, 2004.