e-Yantra Robotics Competition - 2018

Theme and Implementation Analysis – Hungry Bird

HB#5374

Team leader name	Divyansh Malhotra
College	Bharati Vidyapeeth's College Of Engineering, New Delhi
Email	divyansh.sgs@gmail.com
Date	02-01-2019

Scope and Preparing the Arena

Q1 a. State the scope of the theme assigned to you. (5)

Hungry bird aims to recapture the fundamentals of nature where a bird is trying to use its aerodynamic body, into a real-world scenario where we make use of drones to achieve a task of same complexity: navigation through a constrained environment. This not only helps to build a system which models the navigation, but also to use it in specific applications like natural disasters where a drone, as small as PlutoX, can steer through the obstacles and minute points. The theme also exploits the use of a Virtual 3D world to emulate real-world scenarios for more efficient testing.

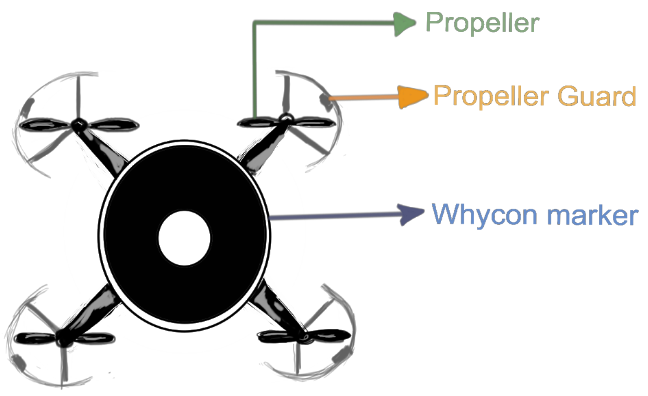


Fig1. PlutoX Drone

PlutoX is a small maneuverable drone that can be used to traverse through difficult terrain.

b. Attach the Final Arena Images. (5)

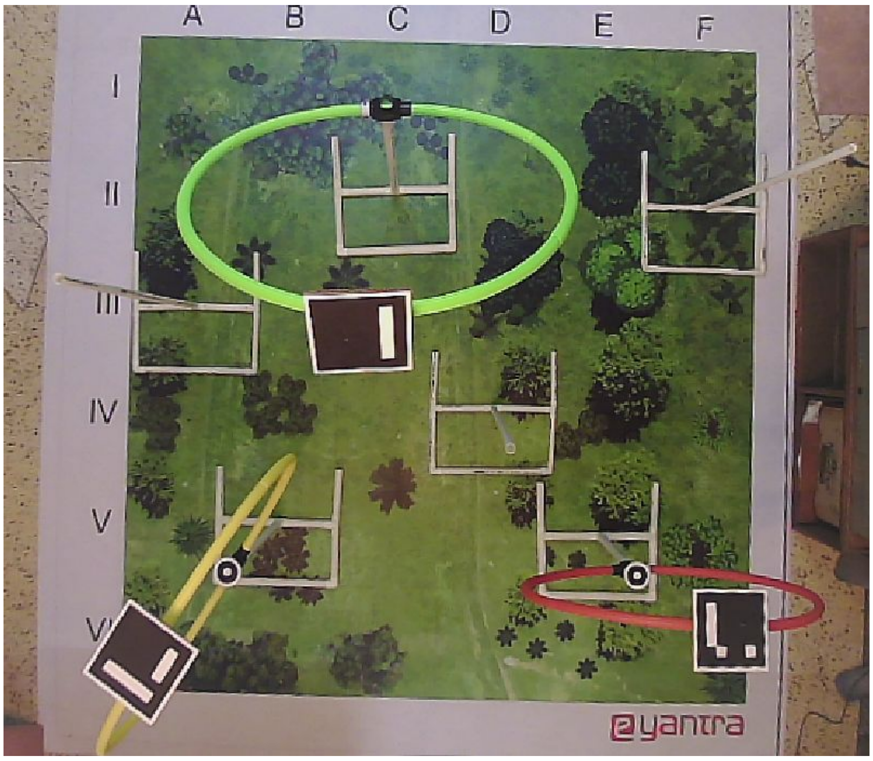


Fig2. Top View

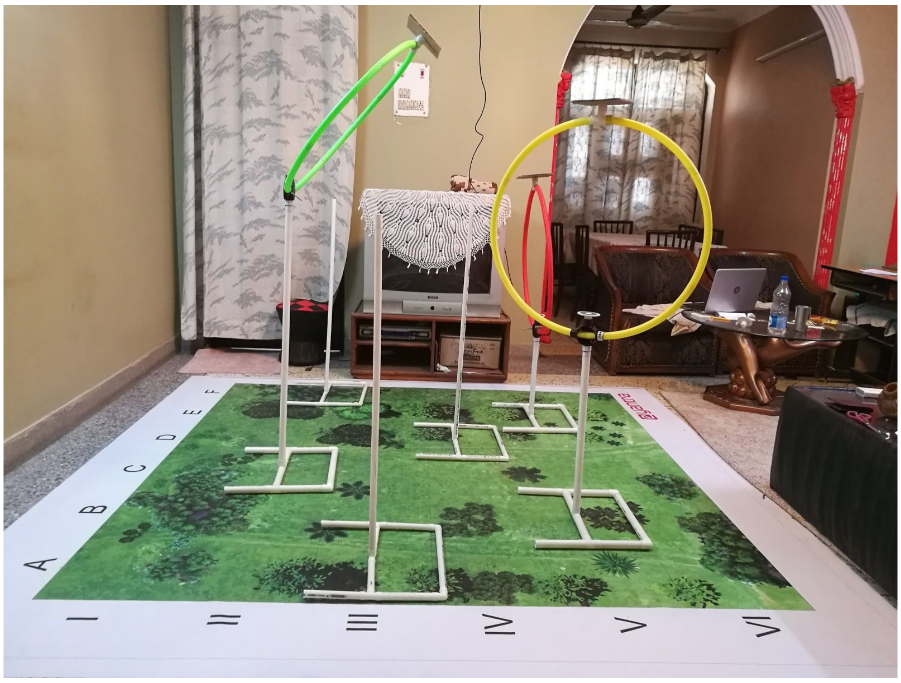


Fig3. Angle 1

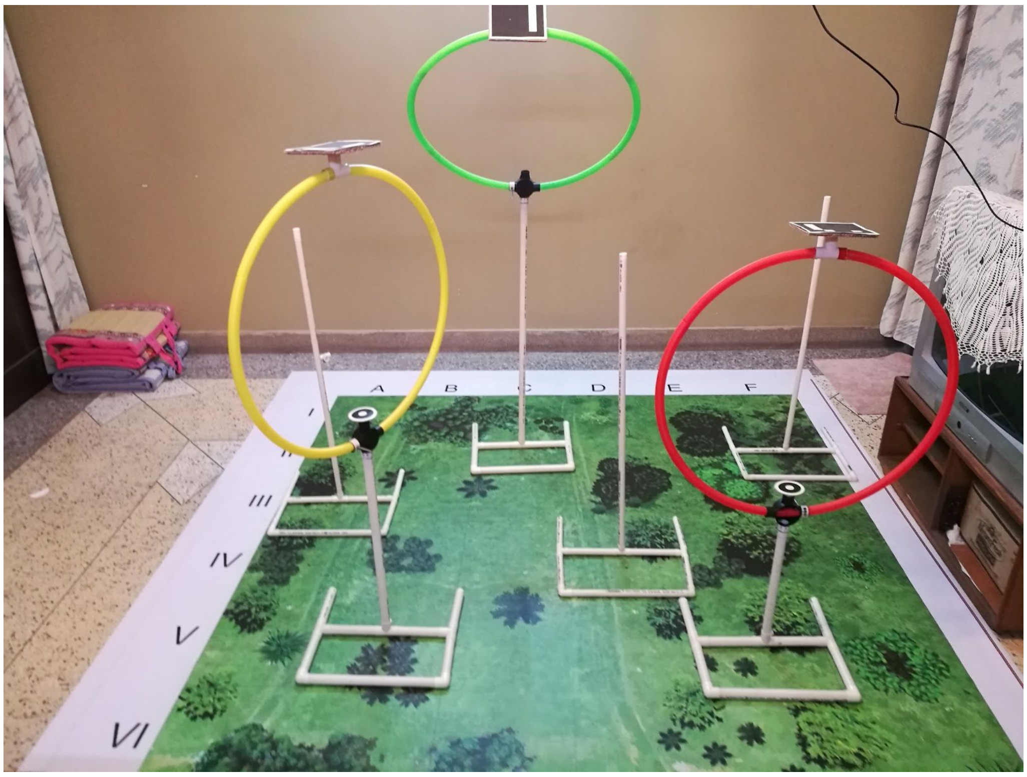
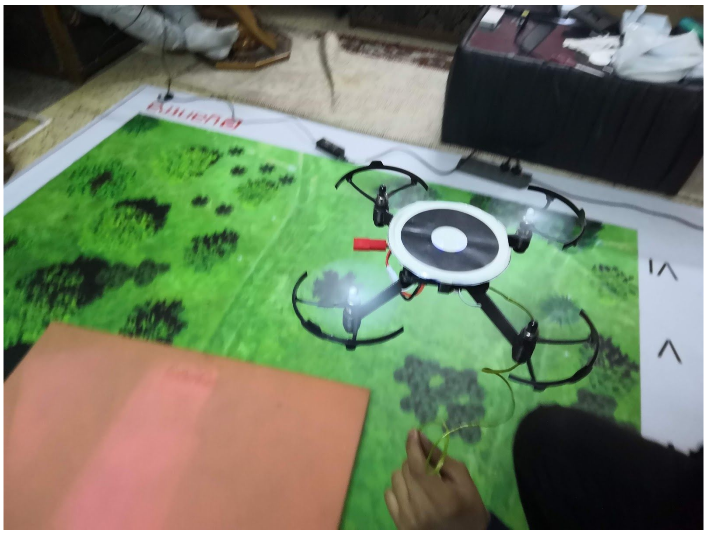


Fig4. Angle 2

Testing your knowledge (theme analysis and rulebook-related)

Q2. How will you ensure that while tuning the PID value, Drone will not crash? (5)

We are aware that the crashing of the drone can result in breaking of the propeller guards, which would then make the propellers vulnerable. To avoid the drone from suffering a crash on a hard surface, during testing, we covered the flex and surrounding area with a layer of foam. Also, we attached a lightweight thread below the drone. One of the teammates would always stand near the premise to stop it from crashing on walls by diverting the drone by pulling on the thread.


.
Fig5. Foam is placed on possible crash sites

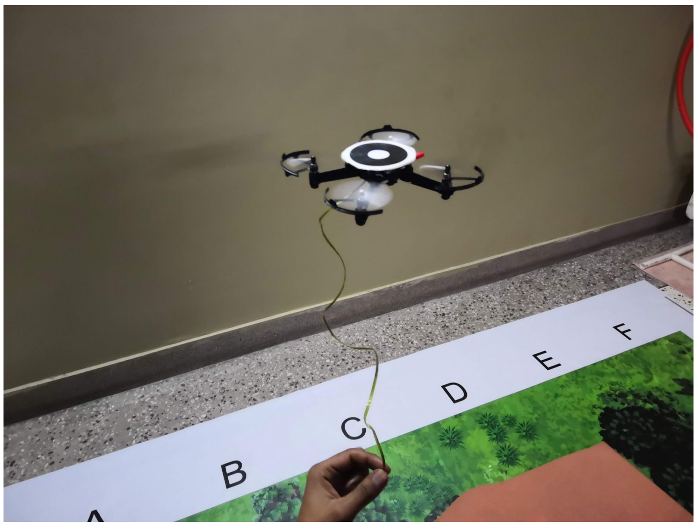


Fig6. Thread is attached below the drone for commanding it in case of a possible crash

But these still left a factor of risk. After a few initial runs, we observed that we could easily make out when a crash is inevitable. Also, if the drone runs out of the camera view, then the PID tuning cannot be done, and a crash is again possible. To avoid these, we developed simple coding solutions:

1. Whenever the drone runs out of the camera's view, the values of the roll and pitch are reversed(1500-the last set error) for a few iterations and then disarmed. The reversal is done to get it back in the testing area.
2. The key input 'D' or '7' was used for emergency situations to immediately disarm the drone to prevent any crashes.

Using the above two solutions and the preventive measures discussed before, we can ensure that the drone does not crash while tuning the PID values.
We can also land the drone smoothly using the key 'r' as stated in the drone_command node.

Q3. How will you navigate the drone through the hoops. Do you anticipate to loose input from the overhead camera while the drone is crossing the hoop? If yes, how are you planning to deal with this scenario? (5)

We would navigate the hoops on reaching the final points of the paths given the OMPL (Open Motion Planning Library) Plugin. When we reach the final point of the path generated, we check the error to be less than a particular threshold, which is taken with consideration of the loop's girth (maximum effective girth is of Sal Tree since it is closest to the camera, being of the highest height). Thus, the drone is considered to be at the final point or the hoop, if the error is within that range of threshold.

Yes, we anticipate to lose the input from the overhead camera. Thereby, the threshold helps us to compensate for exact detection of the drone at the final point/hoop. We take the last values of pitch, roll, and throttle and use the same values to pass in the same direction as it was travelling to pass through the hoop, while we lose the input of whycon from the overhead camera. Using this, the velocity of the drone is maintained in the same direction for the time it is not detected by the overhead camera.

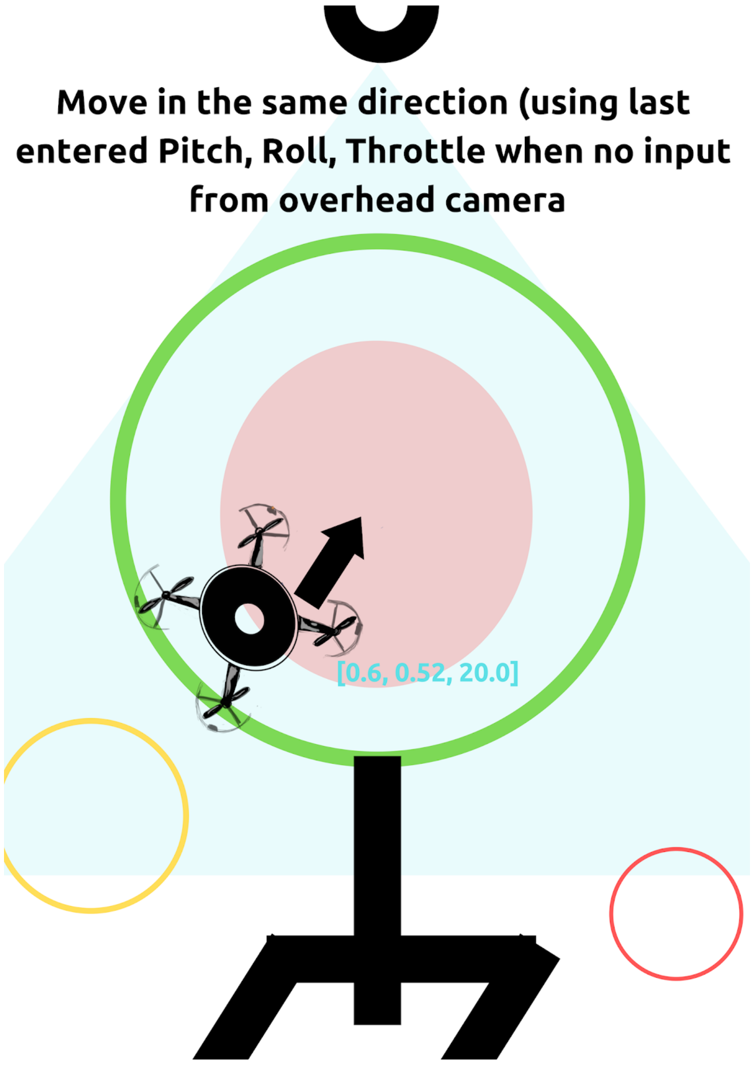

Move in the same direction (using last entered Pitch, Roll, Throttle when no input from overhead camera
[0.6, 0.52, 20.0]

Fig7. We continue moving with the same values when the whycon is not visible

Q4. What is the trend of WhyCon co-ordinates in a single real-world z-plane? (5)

While the camera is originally fish-eyed, after calibration, the resultant image is flatter. But still, it does not reach an ideal state. The x and y whycon coordinates are not as affected as much as the z coordinate by this conundrum. This leads to some variation in the height(z) values in a single real-world z plane, which should be ideally constant throughout the plane.

The observed values for extreme points of the arena are tabulated below:

Position	**Whycon z-coordinate**
Centre	33.47
Top-left	29.80
Top-right	30.82
Bottom-left	29.42
Bottom-right	29.58

The values are at their maximum at the centre and then decrease in an almost similar fashion in all directions running outside the centre.
The challenges offered by this are:

1. Scaling factor of whycon to real-world coordinates for emulation
2. PID tuning
3. Traversal of drone.

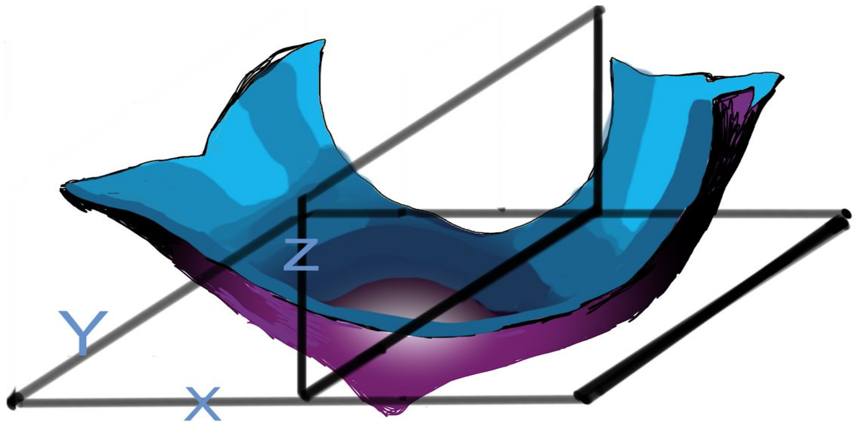


Fig8: Variation in z axis when moving in x-y plane

While there is no straightforward method to overcome this, we decided that we can find the scaling factor by comparing values for different positions in the arena. While it showed some non-linear behaviour, we can reach an approximate relation in the form of a+bz. This gave us good results for combating the first challenge i.e. emulation.

Q5. What will be your strategy to earn maximum points in a run? (10)

There are many constraints, especially involving the battery life that limits the flight time. Hence our aim is to reduce the time required to complete the task. This helps us to increase the factor(420-T) and also provides us with an opportunity to gain the bonus points.
A fine tuned PID, which in itself is a challenge, can help us to prevent collisions with the loops of the food-trees and the non-food trees. It will also help us to facilitate a smooth landing for the drone.
The formula to calculate the score is:

Total score: (420 - T) + (FP * 200) + (HC * 200) - (CP * 20) - RP + LB + B

- T is the total time in seconds to complete the task.
- FP is the Food picking point:
    - Scored when Bird crosses the hoop. This point is awarded only if the Bird crosses the plane of the hoop.
    - No FP for subsequent crossing until it crosses Home Tree
- HC is the Homecoming point:
    - Scored when Bird crosses the Home tree. This point is awarded only if the Bird crosses the plane of the hoop.
- CP is a collision penalty applied:
    - for each collision between Bird and trees/
- RP is a reposition penalty applied:
    - RP = 20 * (e (Tp/2) -1),
    - Where,
        - Tp is the number of times the Bird is being repositioned.
- LB is the landing bonus of 30 points if the Bird lands at the Start.
- B is a bonus of 100 points awarded:
    - When the task is completed within 10 minutes,
    - no penalty is incurred.

Hence, we are currently aiming to minimise T(thus maximising 420-T), minimise CP and procure LB and B. With optimum path planning using the OMPL library and proper PID tuning, we will be able to procure the FP and HC and reduce possibilities of CP and RP. In this way, we aim to maximise our score.

We aim to be prepared for the several cases possible where the battery runs out, whycon detection is poor etc. We need to find a balancing point between the number of points in a path for better accuracy and time taken.

Algorithm Analysis

Q6. Draw a flowchart illustrating the algorithm you propose to use for theme implementation. (10)

Flowchart for real-time emulation of the environment in V-REP

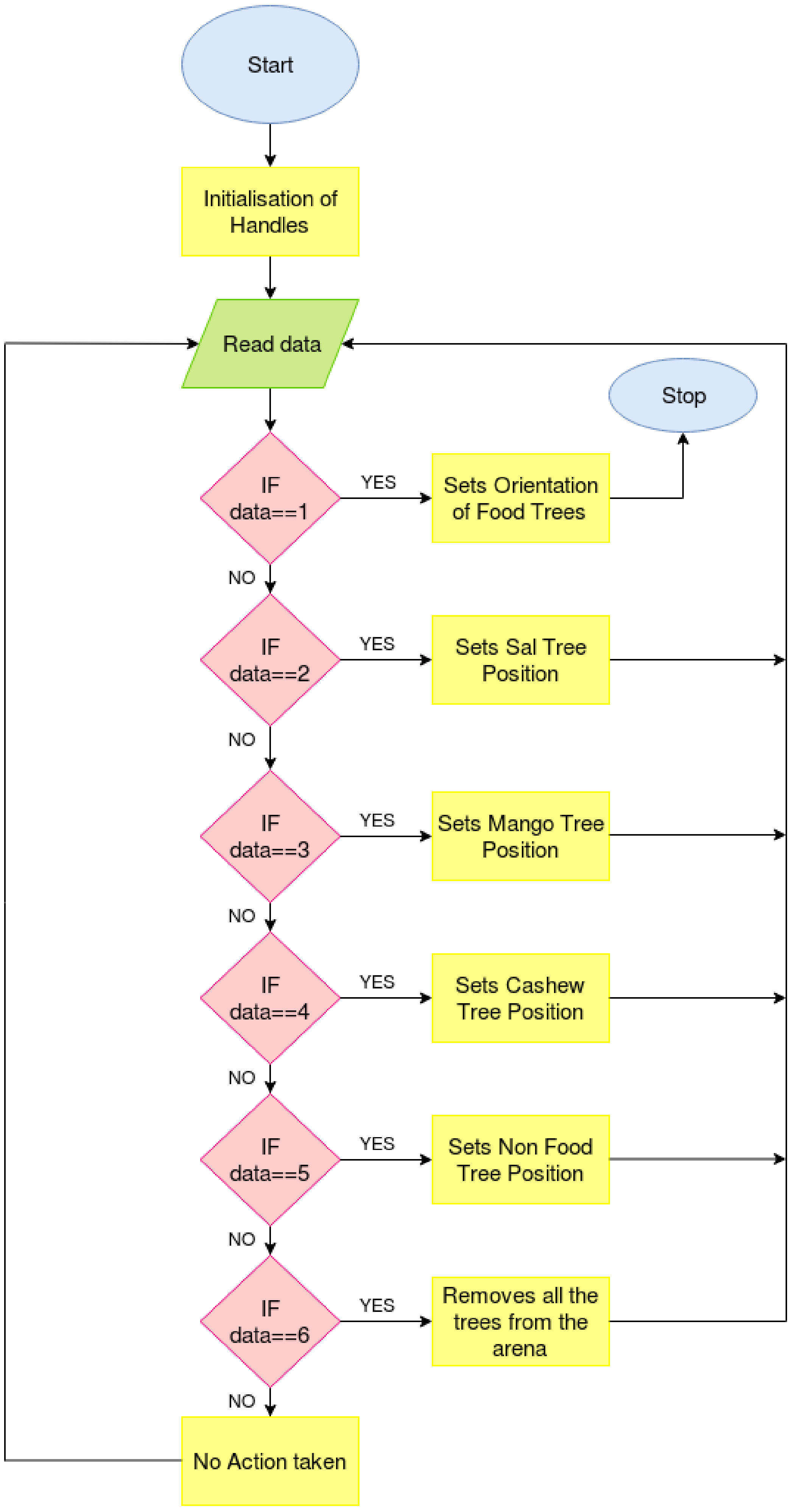
1) sysCall_init() - This function basically adds all the required handles and subscribes to each of their respective topics.

Object	Object Handler Name
Green Hoop	Orientation_hoop1
Yellow Hoop	Orientation_hoop2
Red Hoop	Orientation_hoop3
Cashew Tree	Position_hoop3
Mango Tree	Position_hoop2
Sal Tree	Position_hoop1
No Fruit Tree	obstacle_1

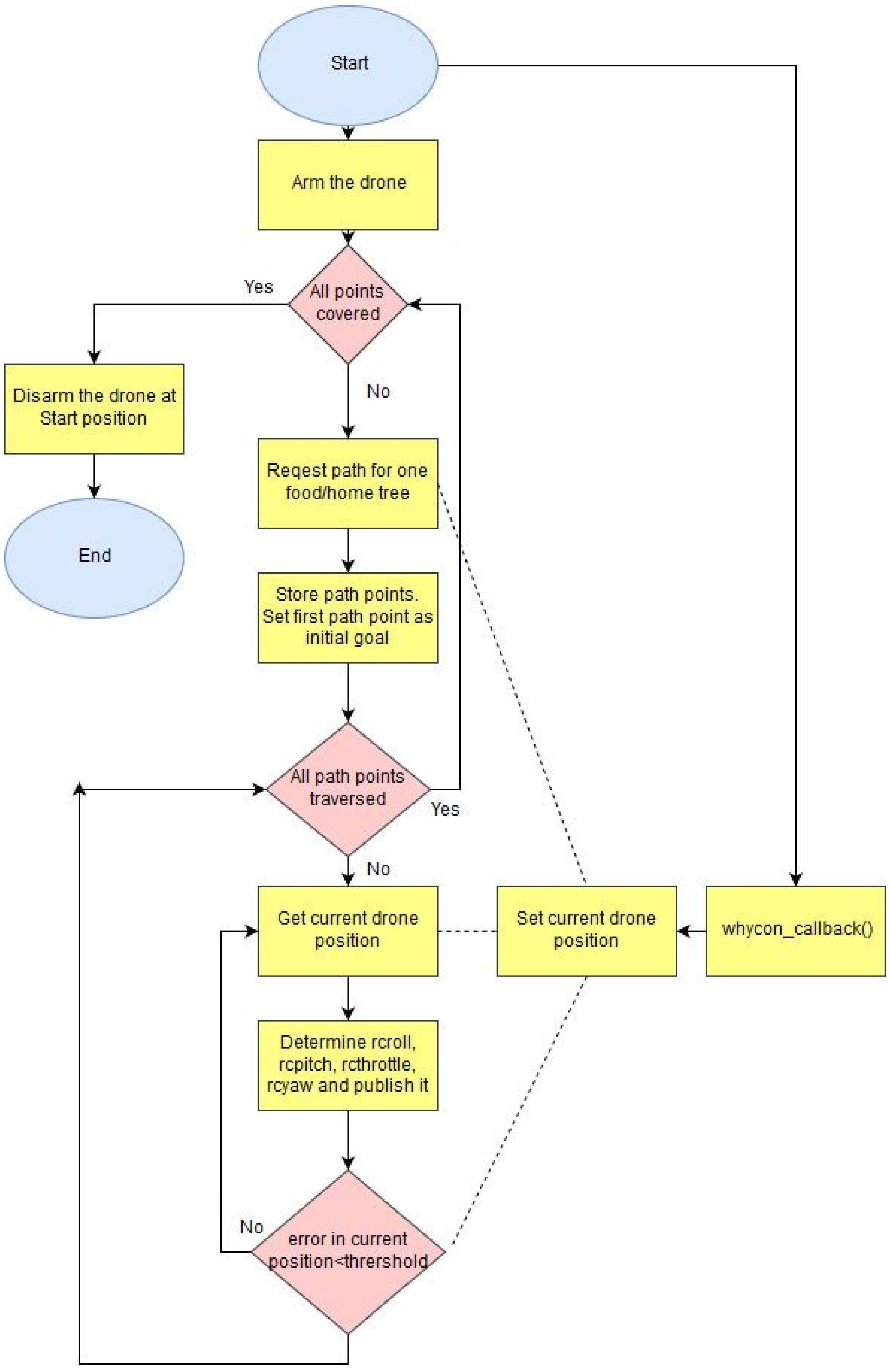
2) aruco_callback(): It gets the orientation of ArUco marker through camera and sets the orientation of each loop using Orientation_hoop dummy.

3) whycon_callback(): It gets the position of the whycon marker and sets the position of trees as well as objects using Position_hoop dummy.

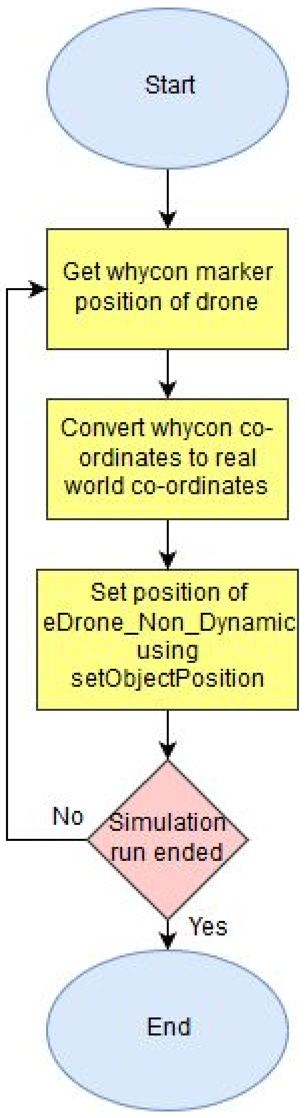
4) key_callback(): It takes the input from user, reads the input and further sets or unset the position and orientation of trees.


Start
Initialisation of Handles
Read data
Stop
IF data==1
YES
Sets Orientation of Food Trees
NO
IF data==2
YES
Sets Sal Tree Position
NO
IF data==3
YES
Sets Mango Tree Position
NO
IF data==4
YES
Sets Cashew Tree Position
NO
IF data==5
YES
Sets Non Food Tree Position
NO
IF data==6
YES
Removes all the trees from the arena
NO
No Action taken

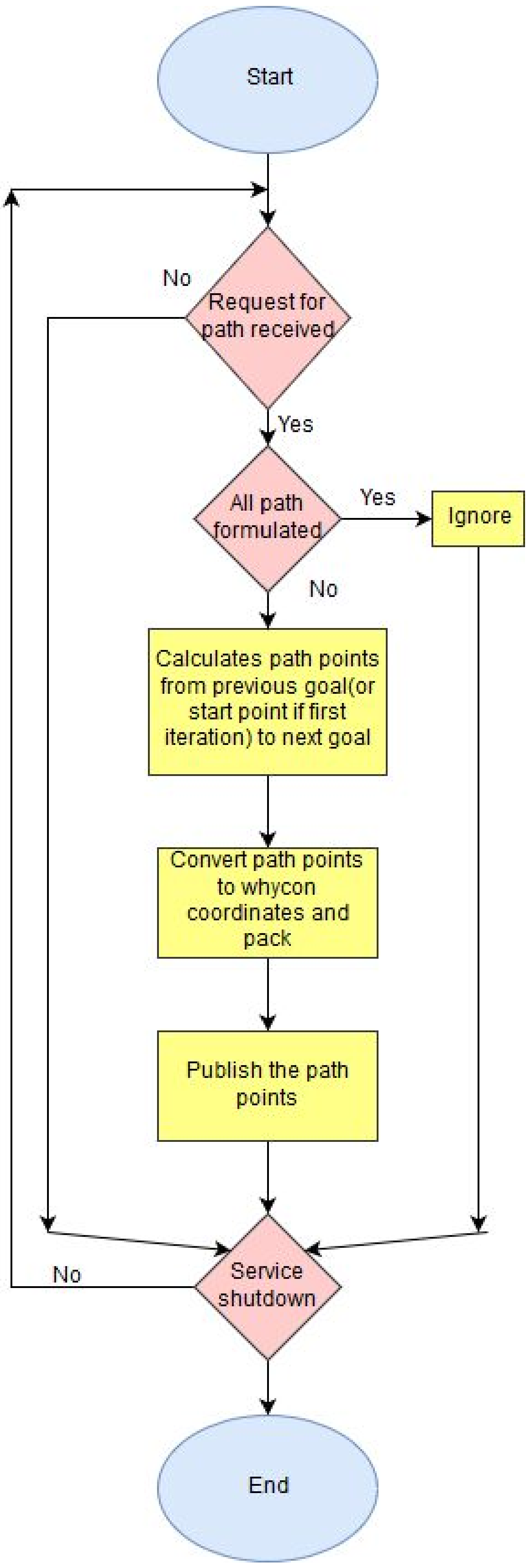
Flowchart for Drone Traversal in V-REP


Start
Arm the drone
All points covered
Yes
No
Disarm the drone at Start position
End
Reqest path for one food/home tree
Store path points. Set first path point as initial goal
All path points traversed
Yes
No
Get current drone position
Set current drone position
whycon_callback()
Determine rcroll, rcpitch, rcthrottle, rcyaw and publish it
No
error in current position<thrershold

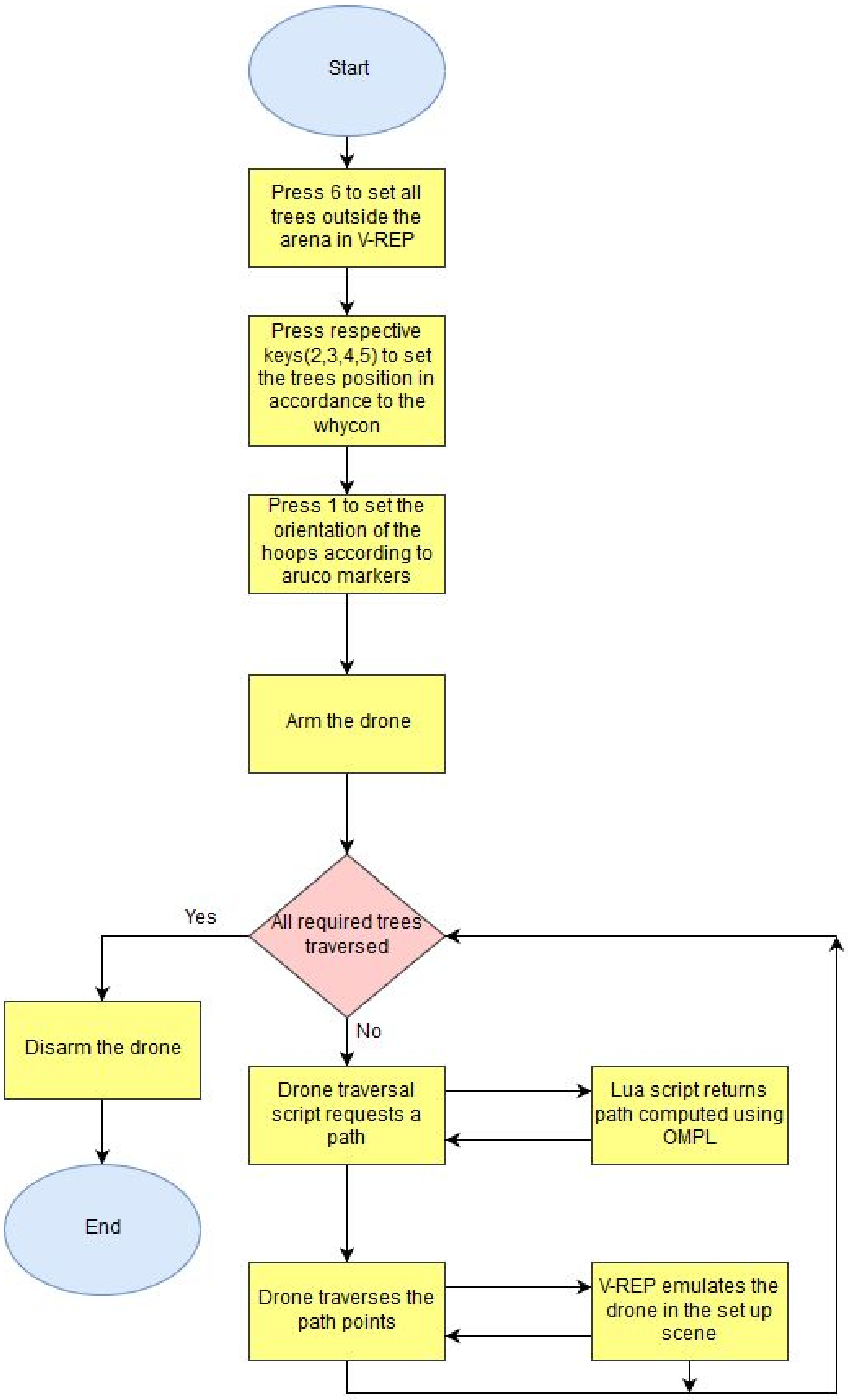
Flowchart for Drone Emulation in V-REP


Start
Get whycon marker position of drone
Convert whycon co-ordinates to real world co-ordinates
Set position of eDrone_Non_Dynamic using setObjectPosition
Simulation run ended
No
Yes
End

Flowchart for path planning in V-REP


Start
No
Request for path received
Yes
All path formulated
Yes
Ignore
No
Calculates path points from previous goal(or start point if first iteration) to next goal
Convert path points to whycon coordinates and pack
Publish the path points
No
Service shutdown
End

Flowchart for complete flow of the task:



Start
Press 6 to set all trees outside the arena in V-REP
Press respective keys(2,3,4,5) to set the trees position in accordance to the whycon
Press 1 to set the orientation of the hoops according to aruco markers
Arm the drone
All required trees traversed
Yes
No
Disarm the drone
End
Drone traversal script requests a path
Lua script returns path computed using OMPL
Drone traverses the path points
V-REP emulates the drone in the set up scene

Challenges

Q7. What are the major challenges that you can anticipate in addressing this theme and how do you propose to tackle them? (5)

1. Dependence of coordinates on Calibration: There is a fish-eye effect in the camera which is flattened using calibration. However, doing calibration may have different effects based on how many points we have covered while doing the calibration. This results in a slight change in the z axis coordinates when we move in x and y direction. To rectify this, we use the scaling factor of z axis (Whycon to real-world conversion) in the form of a+bz, where a and b are constants found by taking the average value from all the extreme ends of x and y axis.
2. Communication dependency on laptop: Based on the speed of laptop, the process of PID is significantly affected, as slower laptops tend to communicate slower and hence fetch and give position of whycon (drone) taking more time. This is also seen when the processed pitch, roll, throttle and yaw values are being processed and sent to the drone. Due to this, we see a slight variation of PID values in different laptops, and hence set them accordingly.
3. Whycon re-identification from overhead camera: There were many barriers which disrupted the detection of whycon marker like the illumination conditions and high speed of drone. To rectify this, we use more light sources for brighter conditions. High speed of the drone is put to limit using limit controls on all the parameters -pitch, roll, yaw and throttle- within a particular range e.g: 1425 to 1575. Further, there was re-identification of whycon marker due to the LED lights on the Primus V3R. To stop this, we added one more layer of opaque paper to prevent much light from passing through.
4. Real world PID tuning in a constraint area: Due to the limited bounded region, PID tuning became a challenging process, as a result, our drone used to move out of arena repetitively. This also increases the risk of collision with obstacles.
   However, to avoid drone collision, we covered the flex and surrounding area with foam. One of the teammates would always stand near the drone to avoid crashing and dragging it back to the arena region using an attached string.
5. Different PID characteristics at different voltage supply: The value of Kp, Ki, Kd vary slightly with the different voltage levels. For now, we tuned our drone at used fully charged battery levels for PID tuning. To overcome this, we tend to employ adaptive PID parameters, which change with the change in the voltage levels.
6. Facing problem in yaw tuning, because it does not depend on the whycon coordinate information. Hence, we took the yaw values from the drone board itself, and tuned it.