Team Description Paper of Indian Institute of Technology Kharagpur for IARC

Aditya Agarwal, Soumyadeep Mukherjee, Manash Pratim Das,
Gaurav Gardi, Sairam K, Kumar Ankit, Kumar Krishna Agarwal,
Vishnu Sharma

ABSTRACT

This paper reports the current preparation strategy of Aerial Robotics Kharagpur, participating in IARC Mission 7 2016. Our main goal includes robust, indoor localization in GPS denied environments supported by optical flow sensors. Other features like ground i-robots detection and differentiation between the target bots and patrol bots is also discussed, followed by a brief description of the herding algorithm AI algorithm to be used.

1. INTRODUCTION

Our research group, Aerial Robotics Kharagpur, started in January 2015 with IARC being the prime target. Hence we started on with the problem statement of IARC mission 7. Our team was organized into two major domains, which are "controls" and "software". Keeping Robotics Operating System (ROS) as the base we developed our simulation environment in ROS and Gazebo in order to speed up the software development and testing while the hardware gets ready. As mission 7 is based indoor, hence we made April Tags based indoor true value setup. We have a website[18] and a GitHub Organization[19].

2. SIMULATION

Gazebo, an open source robotics simulator is used to simulate the robot along with the mathematical, physical and visualization model. It also emulates the environment with the physics and other interactive robots. We have make ROS plugin for the behaviour of i-robots, whereas the quadcopter model, is made on top of hector_gazebo, supported by JSBSim, ArduCopter, RotorS, MAVROS, ardupilot_sitl_gazebo plugin.

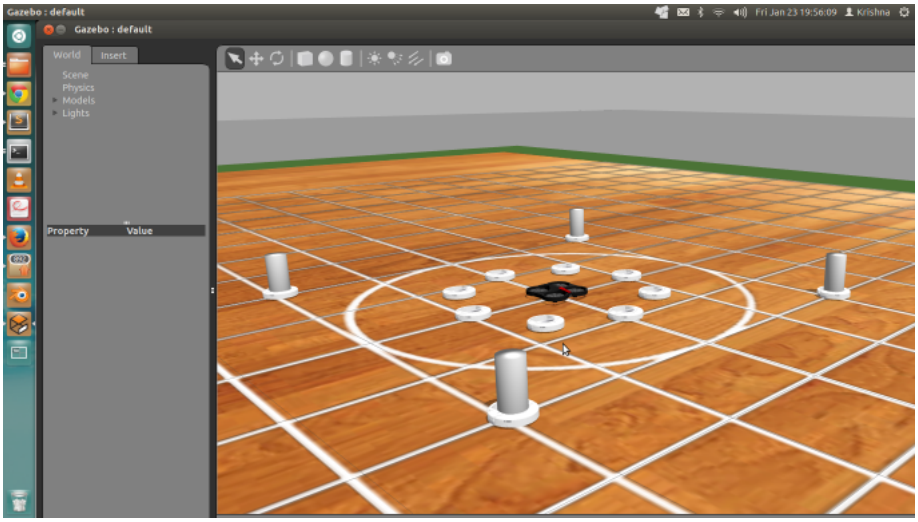


Figure 1. Gazebo Simulator.

3. OVERALL SYSTEM DESIGN

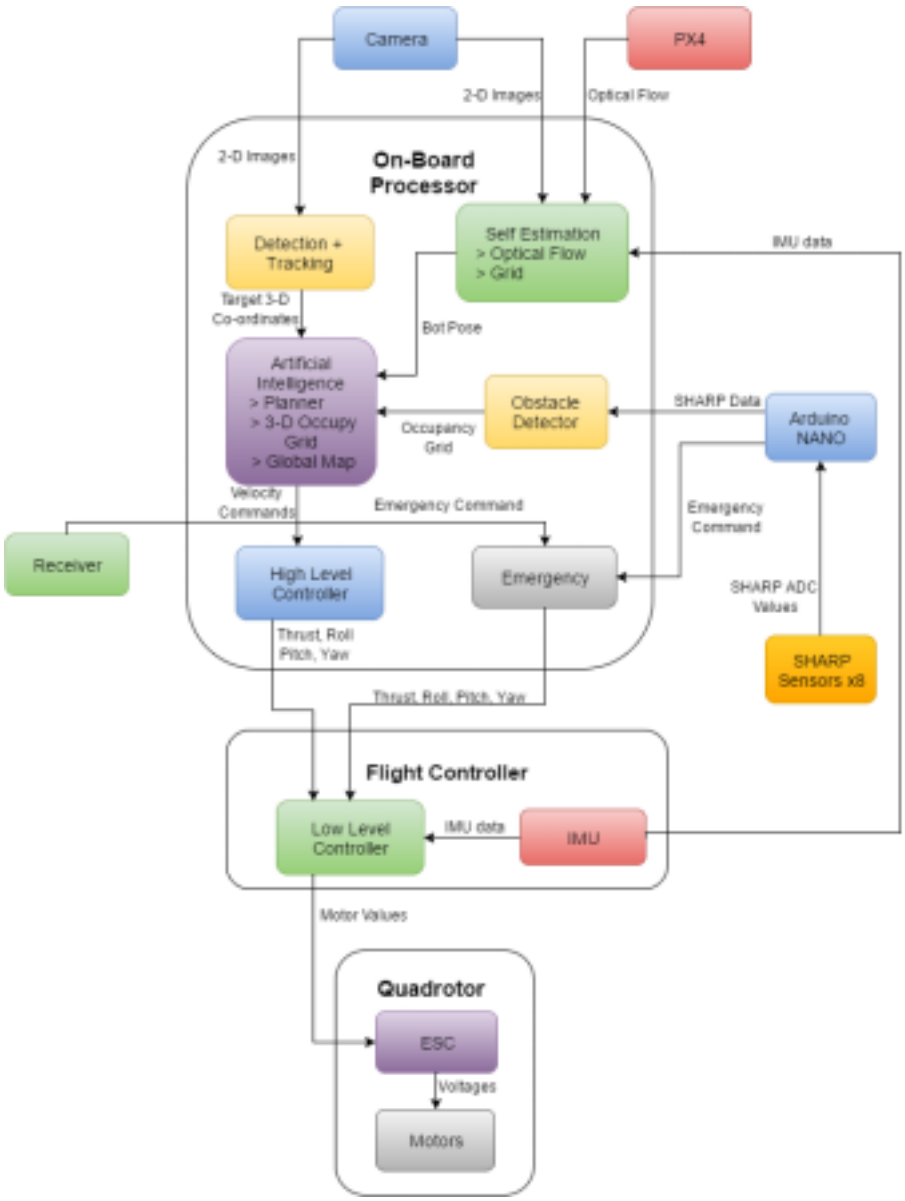


Figure 2. The above is our system design.

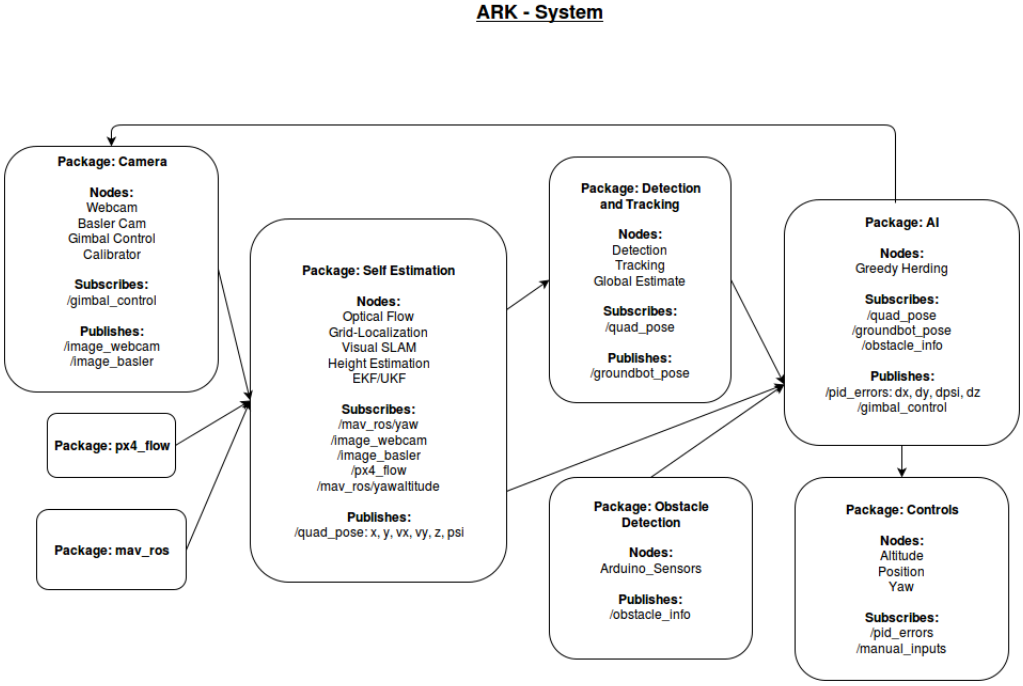


Figure 3. The above is our ROS based framework.

The diagram above shows the ROS packages, the ROS nodes inside them and the ROS topics they deal with. This is the software control part which runs on the onboard computer.

4. LOCALIZATION

For indoor localization of the quadcopter, we use optical flow and grid localization methods, based on px4flow optical flow sensor and bottom camera respectively. The sensor data is fed into, an EKF/UKF state estimator that gives the estimated quadcopter pose.

4.1. Grid Localization

4.1.1. Detecting the Grid

- The image is acquired from the on-board cameras on the quadcopter.
- The lines in the image are first filtered out and isolated by using segmentation, implemented using the Canny edge detector.

Figure 4. Image before canny is applied.

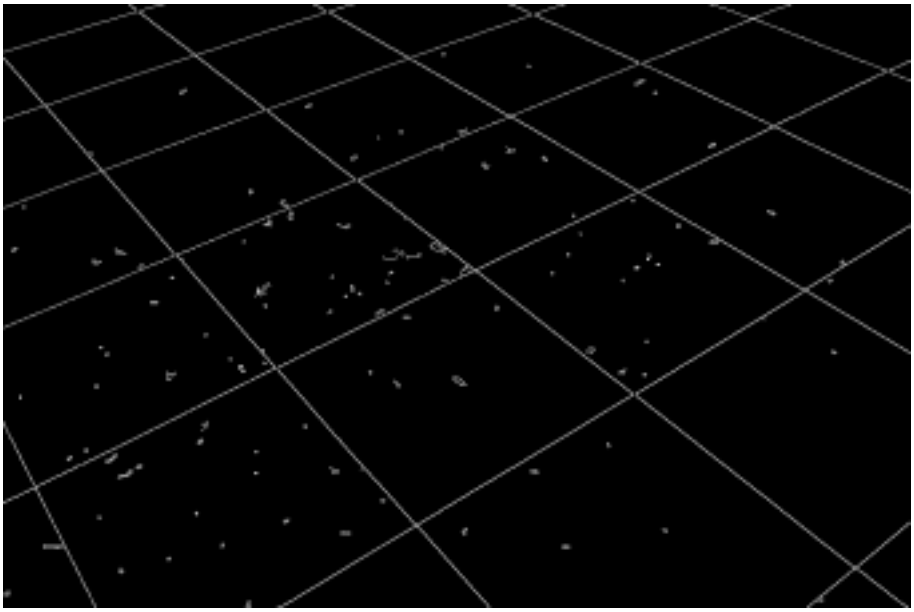


Figure 5. Image after Canny edge detection is applied.

- Once Canny is applied, erosion and dilation is performed on the image to remove noise.
- The lines are identified by using a Hough transform on the image.
- After the Hough transform is applied and the lines found, the intersection points of these lines are identified as nodes of the grid.

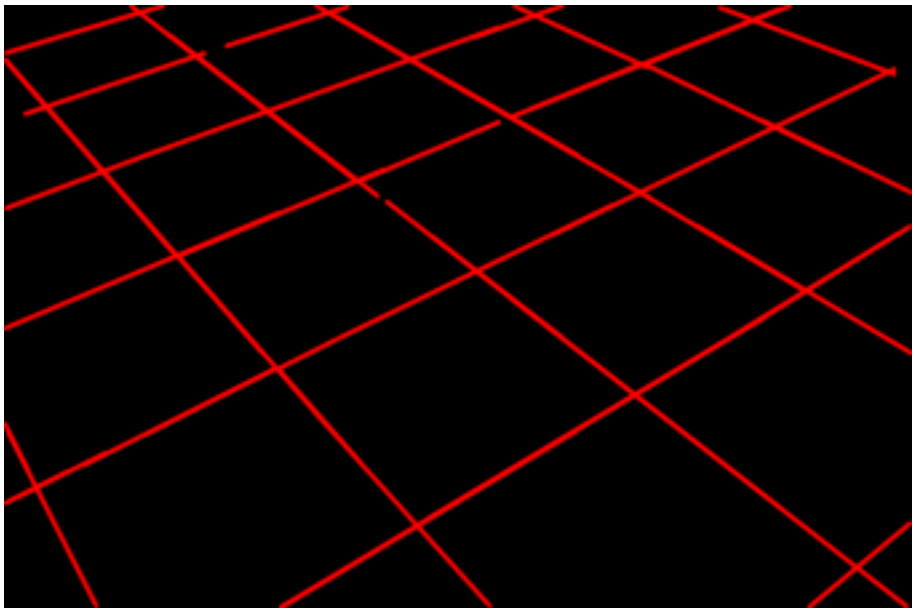


Figure 6. Image after Hough transform is applied(on Canny filtered image).

4.1.2. Grid Following

This is achieved by Line following.

- The destination node is found.
- If the robot is required to proceed from point (x1,y1) to (x2,y2), the robot first finds the nodes that are between these two points.
- It then traverses the grid by following the lines connecting these nodes, keeping either x coordinate or y coordinate constant while following each line.
- At each node, commands are sent to the robot to turn left, right or to move forward or backward to traverse the lines as described above.

4.2. Optical Flow

We are using px4flow optical flow sensor, which is interfaced to the onboard computer through USB. We use px4flow_node[17], to get the optical values that are fed to the state estimation package. The quadcopter's position given by optical flow is further corrected by that given by the grid localization.

5. GROUND ROBOT DETECTION AND TRACKING

5.1. Detection

5.1.1. Ellipse Detection

- The detection of each Ground bot will be done with a modified form of Randomized Hough Transform(RHT), fully described in reference, to detect ellipses that correspond to the edges of the bots.
- Two points are selected as the ends of a major axis, and a third point on the assumed ellipse is selected randomly and the vote of the accumulator is done on the length of the minor axis.

5.1.2. Hog Based SVM Detection

- Train HOG based classifier by using multiple images of ground robot and then run SVM detector.

5.1.3. Template Matching

- OpenCV Template Matching which is a technique for finding areas of an image that match (are similar) to a template image (patch).

5.1.4. YUV Based Segmentation

- YUV Based Cube Segmentation of Ground Robots followed by blob detection and extraction.

5.2. Position Estimation

Once the bots are detected, the noise associated with the dynamic observables of the moving bot will be filtered out using a Kalman filter to enable tracking of the bot. This is achieved by the following steps:

- The Kalman filter takes in the measured position of the bot(which in this case is the centre of the ellipse detected by the RHT) as well as its velocity from the video feed.
- The position can also be estimated by integrating velocity over time (which incorporates the covariance between the position and velocity). This is called the predicted position.
- The error associated with each of these quantities is also found by calculating the expected noise in the readings. The error is estimated as a Gaussian function.
- These two quantities (measured and predicted positions) are compared and the best guess of the bot's position is made by considering it to be the configuration for which both estimates are most likely after incorporating the associated errors.

5.3. Tracking of Multiple Robots

- Having found the most probable position of each bot using the Kalman filter, the next step is to track multiple bots.
- A cost matrix is created which incorporates the direction in which the bots had been moving, the distance of the updated estimates of positions from the previous estimates, the expected collisions as well as the expected turns.
- The cost matrix is run through the Hungarian algorithm to associate the updated positions with the previous positions, thus giving an identity to each bot, and enabling multiple bot tracking.

6. OBSTACLE AVOIDANCE

IARC consists of moving obstacle robots which need to be avoided when they are on the way of the desired trajectory and path.

Obstacles Description: 4 Ground Robots with cylinders attached on top of them to form vertical moving obstacles.

Avoidance Algorithm Overview:
The Obstacle Avoidance in IARC does not need to be global as the number of obstacles are less in number. So, we use eight sharp sensors attached on all four arms and between two propellers which measures the distance between the sensor and nearest obstacle. If the obstacle is closer than a threshold then multiple kinds of action can be taken:

- Wait for Obstacle to Move out of Path
- Avoid Obstacle by Creating a New Trajectory around the obstacle and come back to desired trajectory.

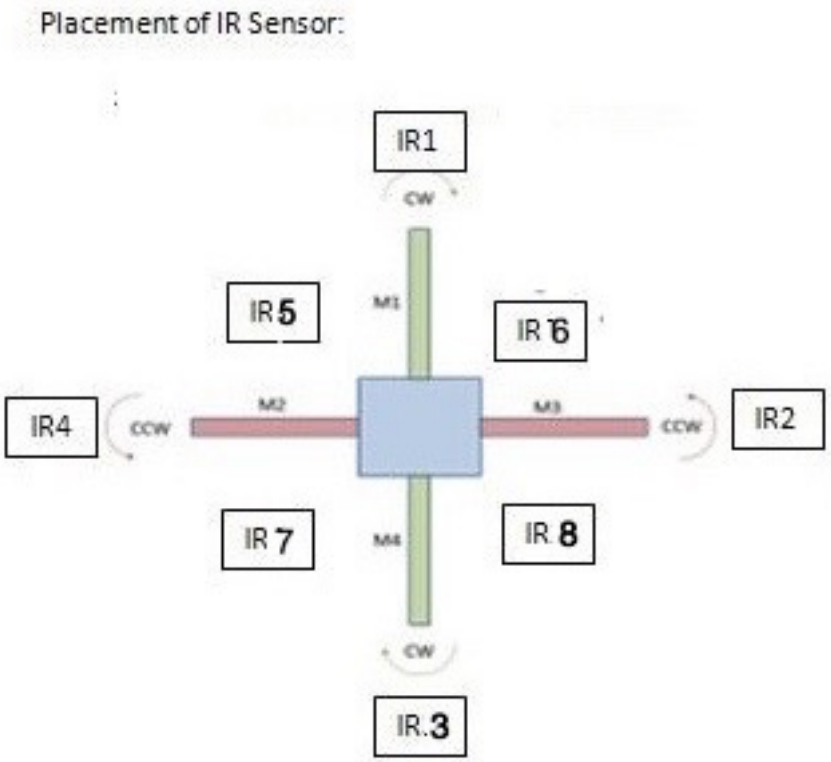


Figure 7. Placement of IR sensor on the quadcopter.

Procedure I:

- Interrupt Control for IR is triggered as soon as an obstacle comes closer than 40cm from any IR.
- Depending on IR from 1-8 and comparing with velocity direction of robot, in case they are same, the robot is halted at the same position or commanded to follow line in reverse till last node is found.

Procedure II:

- Trajectory Generation Module is used to generate trajectories from node to node via the lines or directly.
- This trajectory generation is recalculated on obstacle trigger and a visible graph is created around the obstacle and followed till next grid node.

7. SYSTEM CONTROL

7.1. APM

We have used APM which runs the ArduCopter-3.3 firmware. ArduCopter is a nearly feature-complete open source UAV firmware. Thus our high level control is utilizes the features of ArduCopter to its fullest. Since, most of ArduCopter's autonomous features uses GPS, we used the APM in manual mode along with rc overrides mavlink messages to send signals to control the quadcopter. In this way APM thinks that the quadcopter is being controlled by an RC controller but actually it is controlled autonomously by an onboard computer.

7.2. MAVROS

Since APM communicates in Micro Air Vehicle Link(MAVLink) protocol and both our ground station and onboard computer uses Robot Operating System(ROS), we used MAVROS for communication between onboard computer and APM. MAVROS is a MAVLink extendable communication node for ROS.

7.3. PID control

We have used a two layer PID. The low level control is being taken care of by the ArduCopter. The high level controller, to control position in three dimension and yaw is implemented using MAVROS, in a script which runs on the onboard computer. The scripts changes the mode of ArduCopter to "AltHold" mode and computes the RC Override signal which is the output of PID controller and sends it to the APM.

8. IARC ROBOT DESCRIPTION

Figure 8. Our quadcopter.

8.1. Configuration

- Odroid XU4 : High Level Controller
- APM : Low Level Controller
- ESC : Electronic Speed Controller, 45A OPTO
- LiPo : 11.1V, 3s, 5000mAh, 20C
- Motors: 850kv BLDC
- Propellers : 11" x 4.7"
- Camera : 30fps, Field of View - 78 degrees, Aspect Ratio - 16:9
- Receiver : 6 channel PPM
- Frame : Glass Fibre X frame

9. HERDING ALGORITHM AI

9.1. Greedy Method

- The motif of this algorithm is to stop the ground bots from crossing a line while shifting the line ahead if the area behind is cleared of the bots.
- Before takeoff, the quadcopter will be fed with the direction information about the green and red lines. The direction information will tell where is the red line (east/west/north/south).
- Once the quadcopter takes off, it will hold altitude at 1.5m (from ground) and hold position on the nearest node while holding its yaw.
- As it knows the general direction of the red line, so it turns on the node to a heading that is aligned with a grid line along the general direction of the red line.
- It then follows the grid to:
  - Detect the position of red line. Let us call it L0
  - Track back to the line ahead of it. L1.
  - Move to the end of L1. L(present) = L1.
- The quadcopter will now start making "search clear" maneuver. This maneuver is a loop that involves:
  - Moving along line L(present) to search for ground bots.
  - If ground bots are found, then it will mark the location and predict trajectory of each one of them.
  - Once the search is complete it will target the ground bots it encountered and make strategic landing for each one of them to turn them in the general direction of the green line. This process will clear the particular section it can view while moving along the line and will ensure section present in L $<$ L(present)
  - Once the section is clear it will go to an end of L(present) ++. And continue the loop.
- So we keep clearing the area and pushing the ground bots until they cross the green line.

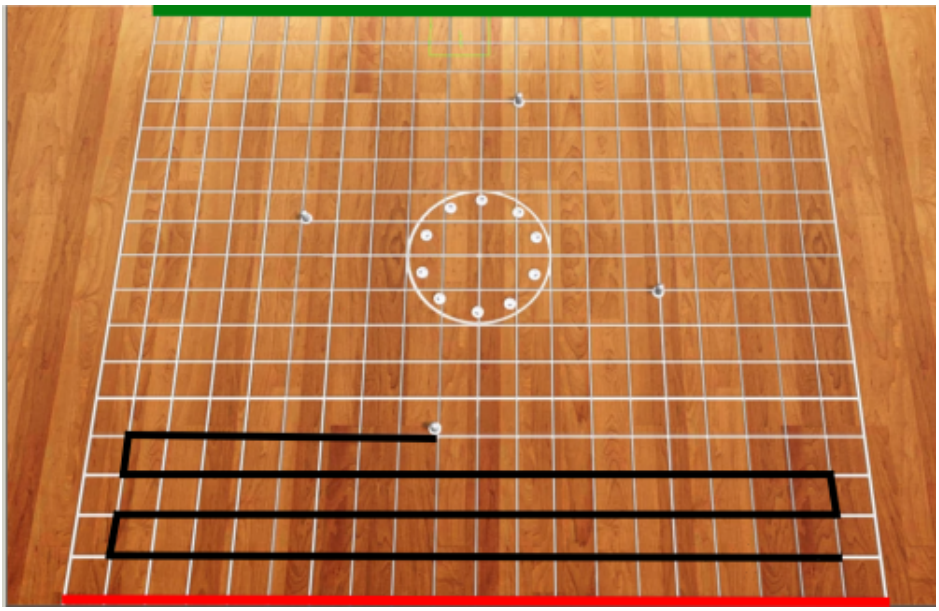


Figure 9. Herding algorithm visualization.

Assumption: Our robot is faster than the ground robots.

10. EMERGENCY KILL SWITCH

We have three level of control on the quadcopter: Wifi-software, standard Radio Control link and "IARC Common Kill Switch"[16] as given in IARC website. As required in the competition, we have a hardware independent safety mechanism to act as a kill switch in case of an emergency that will cut power to the electronic speed controllers. RC control link can be used for manual override control of the quadcopter or to change ArduCopter mode to LAND, while the same can be achieved by MAVROS commands over Wifi link to the on-board computer.

11. TRUE VALUE SETUP

Setup of a ground truth system is essential to register ground truth data. Obtained data can be used to evaluate, test and benchmark algorithms related to localization, mapping and motion planning in quadcopters. This setup should be robust to illumination changes, shadows and arena modifications. Such setup would even help in many different strategies to work upon further, like tracking of system, game strategies and for easy debugging.

Depth perception from conventional stereo vision is based on the triangulation principle as mentioned. [1, 2, 6, 8, 9]. The 3D position of quadcopter can be obtained by the intersection of the two rays passing through the camera optical centers and its 2D projections in each image, but object disparity is an important cue for position estimation. We use two cameras arranged on a baseline, such that their view fields overlap at the desired object distance. By taking a picture with each camera, we capture the scene from two different viewpoints. Hence, a correspondence needs to be established between two images regions that correspond to some physical feature in space. Then the depth can be reconstructed using

triangulation, which is discussed further.
We use April Tags[15] placed on the quadcopter to track it.

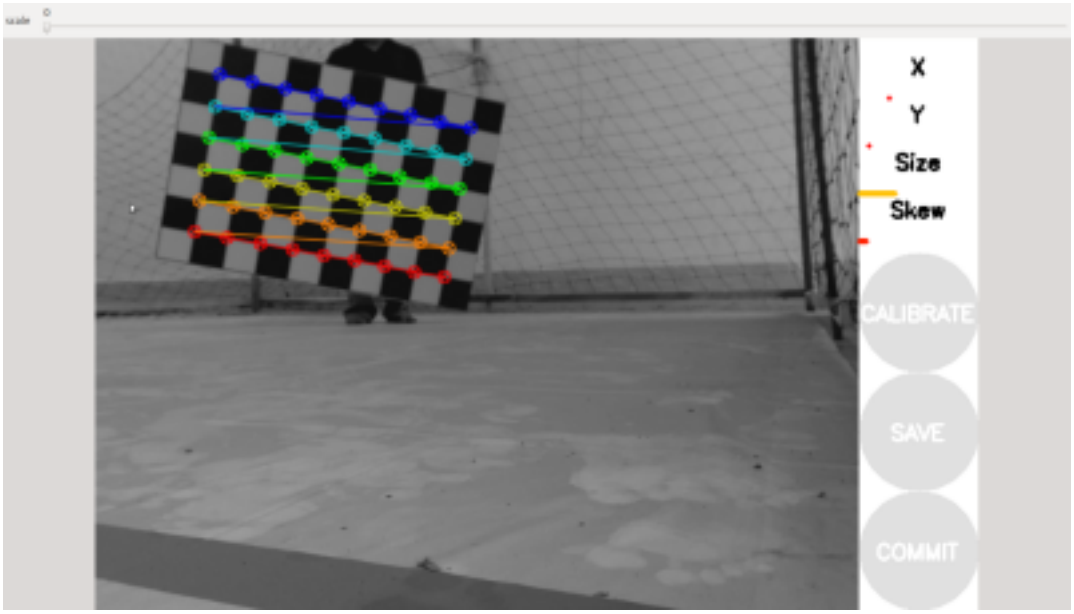


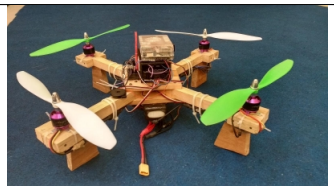
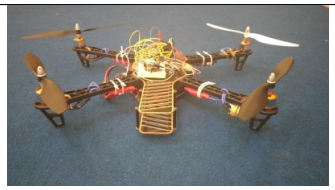
Figure 10. Left and right Camera Calibration using 9*6 checkerboard

12. TESTING

Figure 11. Sample testing arena

We have an indoor testing arena with sample grid floor as in IARC arena and two iRobots. We have safety harness to tie up the quadcopter while testing various controls and PID tuning.
We tested the AI (Herding) algorithms on the simulator. We tied the quadcopter with various allowed degree of freedom to test altitude hold, yaw hold, node hold and grid following algorithms on the real quadcopter in our arena.
We tested various quadcopter frames, onboard computers, flight control boards, cameras and other hardware components like motors, electronic speed controllers,

power distribution system etc. The following are a the combinations we tested.

		
Frame	Wooden X Frame	Plastic X Frame
Processor	Raspberry Pi	Odroid XU4
Controller	Pixhawk	Arduino NANO
IMU	Pixhawk inbuilt	GY87
ESC	30A BEC	40A BEC
Battery	LiPo:14.8V, 4s, 5000mAh, 20C	LiPo: 11.1V, 3s, 5000mAh, 20C
Motors	1000kv BLDC	980kv BLDC
Propellers	11" x 4.7"	10" x 4.5"
Protocol	Mavlink	MultiWii
Payload	600gms	400gms

13. References

1. "Aerodynamics for Engineering Students". E. L. Houghton, P.W.Carpenter.
2. "Fundamentals of Aerodynamics". John David Anderson.
3. "Engineer's Aerodynamics". S. M. Yahya.
4. "Modelling and Control of a Large Quadrotor Robot". P. Pounds, R. Mahony, and P. Corke.
5. "Design principles of large quadrotors for practical applications," P. Pounds and R. Mahony
6. "Towards autonomous indoor micro VTOL". Samir Bouabdallah, Samir Bouabdallah and Roland Siegwart.
7. "Usability assessment of drone technology as safety inspection tools". Javier Irizarry, Masoud Gheisari and Bruce N. Walker, Associate.
8. Honig, Z. (2011) "T-Hawk UAV enters Fukushima danger zone, returns with video." 6:48PM April 21, 2011, retrieved on April 22, 2011.
9. Zibreg. C. (2011) "Awesome use of an iPad and the Parrot AR Drone."
10. "Surveillance Planning with Localization Uncertainty for UAVs". Jan Faigl, T Krajnık, V Vonásek, L Preucil
11. "Collocated interaction with flying robots". Wai Shan Ng and Ehud Sharlin
12. "Flying sports assistant: external visual imagery representation for sports training." Higuchi, K., Shimada, T., and Rekimoto, J.
13. "Simulator aero model implementation." T. S. Alderete, NASA Ames Research Center, Moffett Field, California.
14. "Nonlinear observer design and sliding mode control of four rotors helicopter". H. Bouadi and M. Tadjine
15. "AprilTag: A robust and flexible multi-purpose fiducial system". Edwin Olson
16. The IARC common safety switch
17. px4flow_node
18. Aerial Robotics Kharagpur: Website
19. Aerial Robotics Kharagpur: GitHub organization