An Autonomous Navigation and Object Tracking System for Aerial Multirotor Robots with Obstacle Avoidance in GPS Denied Environments

Syed Hashmath Ulla T A[1], Shivkumar Umadi[1], Vaibhav Bajaj[2], Sushrith Arkal[2], Anush S Kumar[2], Nithin Bodanapu[2], Pallavi M P[2], Upasana Sridhar[2], Himanshu Sharma[1], Nilesh Pattanayak[3]

[1]*Department of Mechanical Engineering*
[2]*Department of Computer Science & Engineering*
[3]*Department of Electronics & Communication Engineering*
PES University, Bangalore

ABSTRACT

The most crucial factors impeding the adoption of autonomous vehicle systems are the identification and tracking of targets, avoidance of obstacles and operation without the aid of GPS or stationary points of reference. This paper proposes and elaborates solutions aimed at a multirotor aerial vehicle under these constraints, with a focus on indoor flight.The choice of Multirotor aerial vehicles stems from their versatility in terms of indoor and outdoor flight. Few other vehicles have the manoeuvrability to fly within narrow corridors and tunnels. Our solution incorporates real-time active path planning, assisted with dynamic obstacle avoidance, for an efficient and collision-free path to the destination, along with object tracking and interaction algorithms aided by computer vision.

1. INTRODUCTION

Team Aeolus will be representing PES University, India at the 2018 edition of International Aerial Robotics Competition (IARC). In this paper we describe the approach used this year, including the details of the system comprising of navigation systems, obstacle and bot detection approaches and the algorithms incorporated into the solution to the problem posed by Mission 7a.

1.1. Problem Statement

To push the limits of advancements in the field of aerial robotics, International Aerial Robotics Competition had put forth its 7th Mission in 2014 which is now entering its 5th year. This challenge involves interaction of the multirotor with ground bots and navigation in a sterile environment with no external aid or stationary point of reference for localisation. The mission is divided into stages - 7a and 7b. In the former stage, the ground robots are to be herded towards the edge of the 20x20 m^2 arena which is demarcated by a green line. The aerial robot controls the direction of the motion of the ground robots by touching the tactile switch on the top of the ground bots. These bots change their motion every 20s in a randomized fashion. When this switch is tapped once the robot turns 45° clockwise and when blocked in the direction of motion, it changes its course by 180°- we shall refer to these actions as ***interactions*** henceforth. The stage is considered successful when at least four of the ten ground bots cross the green line in the given ten minutes. The latter stage involves two or more aerial robots competing against each other to guide the maximum number of robots across the assigned line to each multirotor within the given time limit.

1.2. **Algorithm**

The approach used is based on the greedy heuristic that a bot closer to the green line requires less effort to get it across. A single bot remains the focus of the multirotor's path planning decisions. The multirotor only interacts with this bot until it crosses the green line, at which point it must identify a new target.

The algorithm consists of 2 phases:

- Scouting
- Hunting

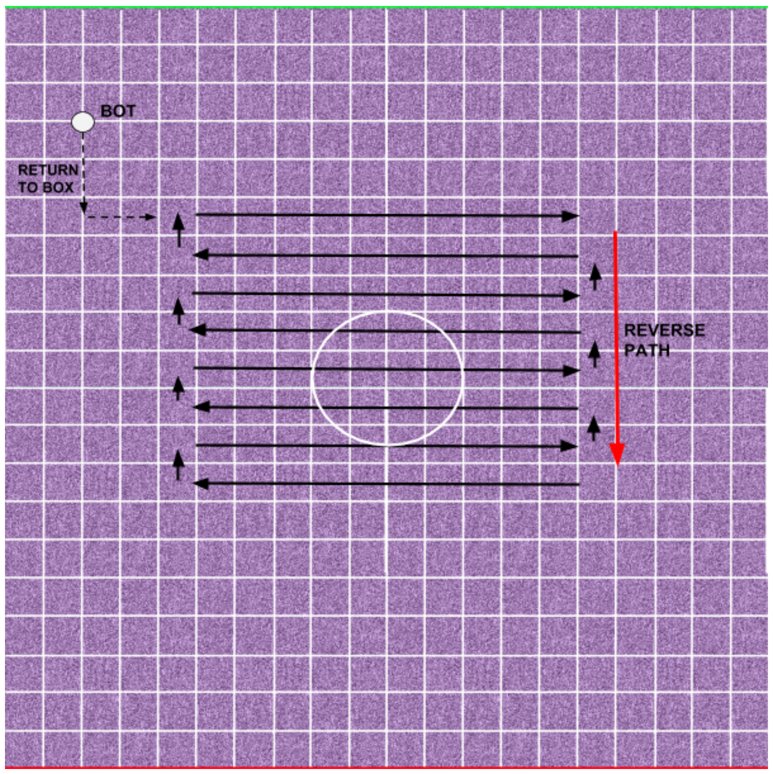


Fig. 1: Scouting Multirotor Robot Motions

1.2.1. Scouting:

This phase scans (scouts) for ground bots and uses a probability density based algorithm to designate a target based on the current position of the bot. The search path can be enclosed in a area of the size 12×8 metres called the Region of Interest (RoI). This restriction is a consequence of the field of view of the forward-facing camera. The path is an alternating left-to-right-to-left sweeping scan of the RoI with the multirotor robot starting between the centre and the green lines and slowly working forward moving closer towards the green line before mirroring its path backward toward the red line.

1.2.2. Hunting:

Once a target has been selected the hunting phase begins. This involves following the ground bot and performing interactions. After the target bot has successfully reached the destination the multirotor robot restart the scouting phase from its home position within the RoI.

Algorithm 1 botChaser

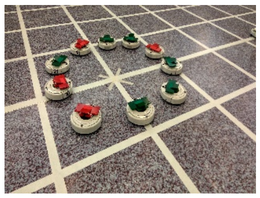
```
 1: procedure BOTSCANNER
 2:     while ROS OK do
 3:         Go OFFBOARD
 4:         ARM
 5:         if armed then
 6:             if current position == target position then
 7:                 $detect$ ← vision_data() # Looks for a target bot
 8:                 if detect then
 9:                     $target\_postion$ ← bot_location()
10:                 else
11:                     $target\_postion$ ← next_point()
12:                 $obstruction$ ← lidar_data()
13:                 if obstruction then
14:                     alternative_path() # Store next point as a safe point
15:                 Publish the final decided next point
16:             else
17:                 Wait till target position is reached
[1]
```

1.3. **Vision**

The algorithm relies on being able to find and track both target bots and obstacles. We have 2 nodes for this purpose, the color detection based target finder and the LIDAR based obstacle avoidance system.

1.3.1. Ground Bot Detection:

The ground bot detection is accomplished using algorithms developed with computer vision strategies. The bots are detected and tracked using colour feature detection algorithms[1]. These algorithms are optimized for situations involving glare.


(a) Input Image

(b) Output Image

The responsibilities of the algorithm can be categorised into the following parts detailed in the following sections:

- Detection
- Decision to attack
- Tracking

1.3.1.1. Detection:

- Previous Approach:
  You Only Look Once (YOLO): It is a machine learning model meant for efficient real time object detection and classification. The neural network divides the image into regions and predicts bounding boxes and probabilities for each region. It looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike systems like RCNN which require thousands for a single image. YOLO is known for its quick detection and lower computational requirements but there is a trade off with respect to accuracy, as compared to other techniques like RCNN, etc. We used pre-trained weights to train the model with over 1000 annotated images and the generated model had an accuracy of detecting a roomba of over 80%.

- Current Approach:
  Color Detection : The Ground Bots as stated have specific colored flaps on top of them. Due to varying light conditions and natural factors, we use a range of values to determine a certain color. This method gives a binary mask where pixels are of value 1 in the regions of the image where a ground bot is present. Once we have the mask we search for contours of at least minimum threshold area so as to avoid unnecessary noise.

- Decision to Attack :-
  The roomba's going in the wrong direction(away from the green line) are the ones we aim to target. One of these ground bots is chosen to pursued by the multirotor.

- Tracking :-
  For tracking a ground bot we use a fisheye camera (F = 2.8 mm and 180 degree diagonal F.O.V). The images captured from the fisheye camera are straightened. We use color detection to track a roomba over multiple frames. We use a polynomial fitting algorithm to fit a line into the set of points used to track the ground bot. Once we have the line we predict an interception point for the ground bot and the multirotor so that we can tap it and turn it.

1.3.2. Obstacle Detection and Avoidance:
Obstacle Detection is performed using an omnidirectional LIDAR[2], which uses a laser to scan the surroundings. A vector denoting the position of each obstacle detected is published to a ROS topic. All the vectors are calculated with the LIDAR unit as the origin. Each obstacle is also assigned an ID in order to track their motion patterns over multiple frames.Obstacles are qualified as potential threats when they are within a radius of 1.5m. The position information over multiple frames is indicative of the most probable path of the obstacle and an obstacle avoidance action is not performed until it is verified that the predicted path with the flight path of the multirotor and the obstacle is deemed a threat. In the event of multiple threats, the evasive action avoids the closest obstacle.

The Quadcopter performs Obstacle Avoidance by:

- Moving 1m along its horizontal axis, in the direction away from that of the approaching obstacle(determined by the angle of approach).
- This is repeated until the LIDAR does not detect any more approaching threats.
- This allows the multirotor to move the minimal distance required to be safe from a collision and hence has a chance to continue tracking the target bot it was following before detecting the thread.
- In case the original target is out of our field of view the multirotor begins to scan the arena in search of another target.

1.4. **Navigation**

Navigation is performed by a finite state automaton which makes its decisions based on data collected from the various sensor specific programs. This communication of information between the different processing nodes takes place over the Robot Operating System (ROS)[3], which is essentially a network where all the sensors and the sensor specific code are individual nodes which can communicate with each other using ros topics or ros services(strongly typed messaging queues). The Vision and the LIDAR programs publish information to specific named ros topics. The navigation program subscribes to these topics and based on the information it receives, it decides where to move next. Navigation is aided by Pixhawk native ECL-implemented Extended Kalman Filter(EKF)[4] state estimator in conjunction with Visual Odometry using the ZED Camera.

2. PAYLOAD

2.1. **Sensors onboard**

We employ a range of sensors to aid in the process of stabilization, maneuvering, navigation and tracking tasks.

2.1.1. Omni-directional LIDAR:
Manufactured by SLAMTEC, RPLidar A2 [2] is a 360° 2D laser scanner with a distance range of (0.15 m-6 m), distance resolution of <0.5 mm, angular range of 360° with resolution of 0.9°, scanning frequency of 10 Hz . The sampling frequency is ≤ 4000 Hz and the laser used has a

wavelength of 785 nm and 3 mW power adhering to FDA Class I laser safety regulations. Data transfer occurs over a serial communication.

2.1.2. Fish Eye Camera:
Manufactured by ELP, ELP-USBFHD01M [5], is a camera equipped with a fish eye lens outputs frames with a resolution of 1920×1080 interfaced via USB, at a rate of 30 frames per second. It has a 180 degree diagonal field of view with a focal length of 2.1mm. This camera is positioned below the multirotor facing the ground.

2.1.3. Depth Sensing Camera:
Manufactured by Stereo Labs, ZED Stereo Camera [6], is a 3D camera for depth sensing and motion tracking. It comes with a 6-DoF positional tracking with an accuracy of +/- 1mm, with a range between 0.5m-20m, capable of recording 720p videos at 60fps, and operating frequency of 100Hz. The camera's field of view extends to 110°. The superior technology of the camera aids in both, accurate ground bot position estimation as well as position estimation of the multirotor in space. It is connected to the processor via USB 3.0.

2.1.4. IMU Sensors:
The flight controller unit consists of three IMU sensors (accelerometer, gyrometer and magnetometer). InvenSense MPU9250, ICM20948 and/or ICM20648 are the first and third set of IMU sensors [7] and ST Micro L3GD20+LSM303D or InvenSense ICM2076xx act as backup sensors. The first set of sensors are placed on the primary board of the FCU while the third set is placed on the board isolated from vibrations. The readings from these different sensor sets are placed on separate buses. They are responsible for providing the inertial measurement required for maneuvering the multirotor robot. The gyrometer readings, signifying the aircraft's rotation about its own axis, are required for a stable flight and used internally by the PID controller. The accelerometer readings provide the acceleration and acceleration forces in all the 3 axes and aid in altitude control of the aircraft. This information can be used for position estimation but is known to be error prone. The magnetometer or the compass, is responsible for providing information about the earth's magnetic field and inferring the magnetic north.

2.1.5. Barometric sensors:
The flight controller unit consists of two MS5611 barometric sensors[8]. It measures the air pressure and in theory can be used to predict the altitude but needs frequent re-calibration.

2.2. **Communication Systems**

The communication system is used as a means to command the vehicle and record its vitals. This can be accomplished via both wired and wireless approaches. Currently the system is configured to communicate via a wireless approach. It is equipped with telemetry radios connected to the FCU via an I2C link. The matching telemetry unit communicates via a serial to USB link to a ground station, which performs the functions of monitoring and commanding the aerial robot via the MAVLink 2 protocol[9].

2.3. **Processing Unit**

We use the Nvidia Jetson TX1 [10], [11] as the onboard processing unit that takes care of the navigation systems, computer vision and communication with the FCU. The specification of the unit is as follows:

- NVIDIA Maxwell™ GPU with 256 NVIDIA® CUDA® Cores
- Quad-core ARM® Cortex®-A57 MPCore Processor
- 4 GB LPDDR4 Memory
- 16 GB eMMC 5.1 Flash Storage
- Secondary Memory: 128GB SDXC memory

2.4. Flight Controller Unit

Pixhawk [7], manufactured by 3DR is the flight controller unit onboard the multirotor. It is an open source autopilot. The role of this unit is to read the input data from the various IMUs and sensors, perform calculations and as an output, manipulate the speed of the motors to execute required maneuvers. It is equipped with a 32-bit ARM Cortex M4 core with FPU operating at a frequency of 168 Mhz, 256 KB RAM, and 2 MB Flash memory. The sensors onboard are discussed in detail in the Sections 2.1.6 and 2.1.7.

The Pixhawk can utilise 3 different power sources from a USB input, power module, and servo rail input for redundancy. For IO operations, it comes with an I2C port, 2 CAN ports, 5 UART ports and SPI. The motors are connected to the servo rail and the RC receiver is connected to the RC port. The optical flow module connects to the I2C port on the FCU module.

3. Aerial Vehicle

3.1. Propulsion system

The multirotor uses 15$\times$5.5 inches propellers to provide sufficient lift. The propellers also work most efficiently at mid throttle ranges i.e. 50-55% throttle. The multirotor is designed such that at 50% throttle the weight of the multirotor is equal to its thrust.

These propellers are driven by Tiger© MN4010 [12]. They have comparatively low current ratings between 4.2A and 14.6A at 22.2V in 50% to 100% throttle range respectively. They operate at a maximum current rating efficiency of 84% between 3-8A at a temperature of 52°C, ensuring longer flight times.

T-motor 40A ESCs are used to drive the motors. These ESCs are recommended for the Tiger motors for 4S-6S configurations. The ESCs are flashed with Simon K firmware and calibrated through the Pixhawk to avoid de-sync during flight. The ESCs are connected to the motors using bullet connectors which can handle currents up to 50A.

3.2. State Estimation and Control

The state estimate of the multirotor is obtained by FCU's Inertial Measurement Unit's (IMU). The fusion is carried out by the ECL implementation of the EKF state estimation algorithm. The algorithm's parameters have been tuned on observations derived from manual flight and errors in measurement of sensor data inferred by analysis of flight logs. Further, the state estimation is converted to corresponding PID to control the multirotor. This is implemented in the internals of the PX4 v1.7.3 flight stack.

3.3. **Flight Termination System**

Even the best engineering designs are bound to fail in certain hostile conditions in the event of any system malfunction. In a multirotor there are several sub-systems and failure of any one of them could be precarious. To ensure safety of the user or individuals around this multirotor, we have introduced several safety measures into the system. If the autonomous or path planning system fails, the multirotor is programmed to return to its starting position and land. In the event of failure of positioning system, the aerial vehicle shifts to manual Radio-Controlled mode. For any other breakdown, we can disengage the power to the Electronic Speed Controllers (ESCs) and motor by using a radio-linked switch (kill switch) which works independent of all other communication links and computing system. This will immediately terminate the flight.

3.4. **Power Supply Unit**

The system is powered by Lithium Polymer batteries rated at 4000mAH at 22.2V. Further, the individual subsystems receive their rated voltage from a custom-designed voltage divider circuit. Considering the current rating of the batteries and the estimated payload, the theoretical upper limit of the flight time with this battery and configuration of the system is about 20 minutes.

4. System Architecture

4.1. **Overall System Architecture**

The system consists of 2 cameras, a forward-facing stereo ZED camera and a downward-facing wide-angle camera. The system initially uses the ZED camera to scan the RoI to detect any bots that are headed in the wrong direction. The direction of the bots is then corrected. Further, to interact with the bots, the wide-angle camera is engaged in order to exercise fine control over the interaction. The interactions are then repeated until the bot reaches the green line and this process is repeated for all the bots.

The bot detection sub-system interacts with the navigation architecture to command the multirotor robot to perform the interactions. The navigation module abstracts the vision sub-system from the intricacies of the low-level communication The interactions are accomplished by the navigation architecture, which in turn sends low-level commands via MAVLink v2 protocol to the FCU.

4.2. **Navigation Architecture**

The LIDAR sensor is used to detect the presence of obstacles in the environment. It can give a 2D snapshot of distances of various obstacles at a point in time.

The LIDAR sensor's data is consumed by the Lidar_Node, which is a software module in the form of a ROS Node, which provides a wrapper that abstracts the low-level details of serial communication that is performed to obtain data from the LIDAR sensor, and in turn provides the readings of the LIDAR sensor in a programmer friendly manner over ROS strongly typed messaging queues.

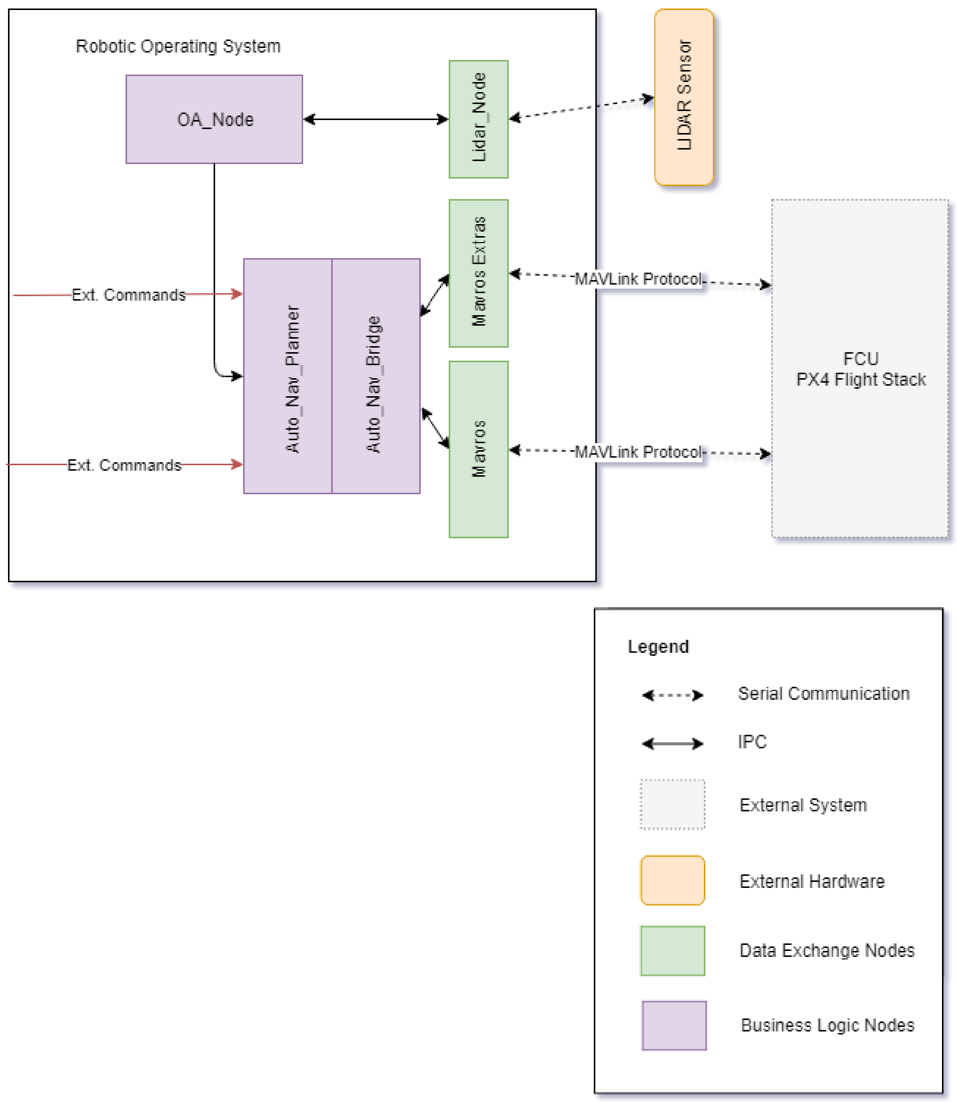


Fig. 3: Navigation System Architecture

The OA_Node is the module responsible for obstacle detection. It uses the Lidar_Node to obtain readings from the LIDAR sensor and performs analysis and computation over multiple instances or snapshots of LIDAR readings to detect obstacles and in turn provides details of obstacle presence. The node also performs data filtering. It restricts the distance to accurate range of the lidar, this parameter is configurable by the user.

The Mavros and Mavros_Extras are ROS Nodes that abstract details of communication over serial or WiFi link with the MAVLink protocol for communication with the FCU.

The FCU receives messages over MAVLink that contain information and commands that it must follow. The FCU runs the PX4 firmware that runs on top of NuttX.

The Auto_Nav_Bridge node is the module responsible for low level communication with the FCU via MAVLink. It also takes care of publishing rates requirements from the MAVLink protocol and the PX4 Firmware. It also plays crucial role of continued publishing the current point on failure of planner node.

The Auto_Nav_Planner node is the module responsible for taking high level decisions to enable the aerial robot to navigate to the provided destination coordinates without colliding or touching the obstacles. It interacts with the OA and Bridge nodes to accomplish this. This module is the entry point for clients/users to use the system. It acts like a facade to the system, hiding its internal complexities and providing a simple interface for the client to use the system.

4.2.1. Enhancements over previous architecture:
This section will detail enhancements over the previous iterations of the system architecture.[13] The goal is to enhance existing features and develop new features for more stable and consistent navigation as well as object tracking. We have shifted our implementation for the navigation module from the optical flow(which gives inconsistent values) to using external odometry such as the ZED Camera which provides position coordinates using Visual Odometry[14]. Since external odometry is used in place of internal IMUs to track the multirotors position and since the external odometry is not directly connected to the Pixhawk, the Pixhawk is unaware of its position and hence navigation by publishing setpoints to the Pixhawk is no longer possible. The navigation code decides the next setpoint to go to, and by implementing PID in the navigation code, the required motion of the multirotor in the x,y and z axis to reach the next setpoint is determined. These values are fed to the Pixhawk using Command Velocity Messages which then applies the appropriate thrust in the 3 axis to move the multirotor to its destination as decided by the navigation code. The new enhanced architecture uses the new ECL implemented EKF state estimator which fuses more data than LPE to estimate the robots altitude and is also compatible with the latest version of the MAVLink protocol (v2). The main enhancement made in Vision is the use of a bottom facing fisheye camera which is capable of detecting bots over a larger field of view compared to the camera used previously.

5. System Robustness

5.1. **EMI/RCI**

As discussed previously in Section 2.1.6, the aerial vehicle has a flight controller with embedded accelerometer, gyroscope and magnetometer. All these sensors are used alongside, in order to enhance the performance of the flight controller. These sensors are highly susceptible to external noise. The sampling frequency used by the flight controller unit is 1KHz, hence any other electromagnetic signal which has frequency less than 500Hz can corrupt the sensor values. The ESCs and motors draw high amperage of current at varying frequencies from about 1KHz to 300KHz and therefore it is extremely likely that the EM waves generated by them can corrupt the sensor data.

Additionally, the internal or external communication systems can introduce electromagnetic interference. The solution to this setback is electromagnetic shielding of the sensors and the wires carrying the signals or high amperage current to drive the motor. The EM shield is grounded with the body ground, which connects the universal ground of every circuit. By placing the IMU's and ESCs strategically, the coupling can be reduced. Coaxial cables are preferentially used wherever possible. We have matched the impedance of every AC signal carrying conductor to reduce the reflection.

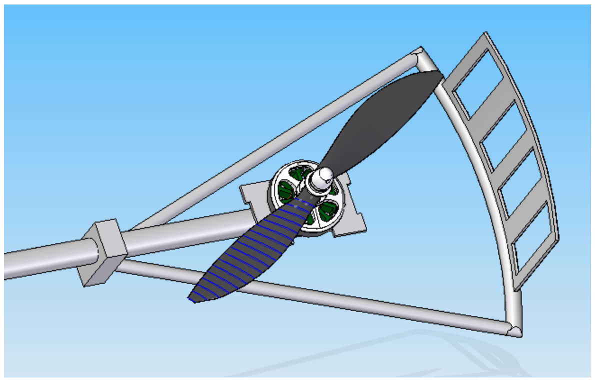


Fig. 4: Propeller guard

5.2. **Vibration**

Multirotor vibrations are one of the most critical problems faced during flight, take-off and landing. They need to be reduced in order to stabilize the multirotor. While taking off, landing and hovering at low altitude, ground effect occurs which causes considerable vibrations[15]. In order to avoid that, the take-off and landing time has been minimized and hovering will be done at a higher altitude, so as to have negligible ground effect. During landing, the vibrations have been overcome by introducing simple, cost effective bush dampers right below the landing gear. The four legs of the landing gear are all connected as one unit to reduce imbalance that causes vibrations as well.

Rotating motors inevitably create vibrations that also adversely affect the sensors and visual display. We have used dampeners, currently foam, around motors and the other electronic parts to reduce these vibrations to a large extent. Along with the foam, there are rubber bushes under the flight controller unit to reduce the error in the sensor readings. Nut and screws turn loose due to vibrations and hence have been plated with thin sheets of foam to keep them intact. Our previous propeller guards were curved and closed structures. This caused the small amount of air flowing horizontally to hit them and induce vibrations. We have now introduced a Gated curved surface surrounding the propellers so as to allow maximum flow of air outwards.

5.3. **Safety**

While test flying the autonomous multirotor, all nearby team members stay 2 meters away from it at all times. In addition, the frame is always tethered with two ropes to prevent it from harming anyone or from crashing.

To ensure the safety of the drone and the team members, we use prop guards attached to the arms of the multirotor. In cases of doubt the drone will stay grounded until the issue in question is found and resolved. Our prop guards took multiple iterations, these have been 3D printed using ABS lament given its strength, flexibility, machinability, and high temperature resistance. We wanted a design that would not interfere with the prop wash, but still be able to avoid the multirotor from destroying itself in the unfortunate event if collides with objects around it.

6. Acknowledgements

We extend our gratitude to our institution, PES University, for its constant support through the development of the project. We would like to sincerely thank **Prof. Carine V D'Souza** for her continued guidance. We also appreciate the efforts of **Sankeerthana M** – Ground Bot Electronics and Killswitch Design, **Ahana Kumar** – Marketing, **Vishwavadana R** – Marketing and Logistics, **Harish Krishnan** – Logistics, **Chaitanya Premkumar** – Fabrication, **Nayantara M** – System Electronics, **Sankirthana KN** – Ground Bot Electronics, **Suyash Khetawat** – System Electronics, for their contribution in the development of this project.

References

[1] M. Deswal and N. Sharma, "A fast hsv image color and texture detection and image conversion algorithm," *International Journal of Science and Research (IJSR)*, vol. 3, no. 6, 2014.

[2] SLAMTEC, *RPLIDAR A2 Specifications*, http://bucket.download.slamtec.com/004eb70efdaba0d30a559d7efc60b4bc6bc257fc/LD204_SLAMTEC_rplidar_datasheet_A2M4_v1.0_en.pdf, 2016.

[3] Open Source Robotics Foundation, "About ros," http://wiki.ros.org/, 2017.

[4] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," pp. 3068 – 3068 – 12, 1997. [Online]. Available: https://doi.org/10.1117/12.280797

[5] ELP Cameras, *ELP Wide Angle Camera Module Datasheet*, http://www.elpcctv.com/wide-angle-full-hd-usb-camera-module-1080p-usb20-ov2710-color-sensor-mjpeg-with-21mm-lens-p-204.html.

[6] Stereo Labs, "About zed stereo camera," https://www.stereolabs.com/], 2017.

[7] Pixhawk, *Pixhawk Autopilot*, https://pixhawk.org/modules/pixhawk, 2013.

[8] TE Connectivity, *MS5611 Berometric Sensor Datasheet*, http://www.te.com/commerce/DocumentDelivery/DDEController?Action=showdoc&DocId=Data+Sheet\%7FMS5611-01BA03\%7FB\%7Fpdf\%7FEnglish\%7FENG_DS_MS5611-01BA03_B.pdf\%7FCAT-BLPS0036.

[9] L. Meier, *MAVLink Micro Air Vehicle Communication Protocol*, http://qgroundcontrol.org/mavlink/start.

[10] NVidia, *Jetson TX1 Developer Kit*, http://developer.download.nvidia.com/embedded/L4T/r23_Release_v1.0/NVIDIA_Jetson_TX1_Developer_Kit_User_Guide.pdf, 2015.

[11] Nvidia, *Nvidia Jetson TX1 System-on-Module*, http://developer2.download.nvidia.com/assets/embedded/secure/jetson/TX1/docs/JetsonTX1_Module_DataSheet_DS07224010v1.1.pdf, 2015.

[12] T-Motor, *T-Motor Data Sheet*, http://www.rctigermotor.com/html/2013/Navigator_0910/38.html, 2017.

[13] P. Singhania, Siddharth R.N., S. Das, and A. Kalkunte Suresh, "Autonomous navigation of a multirotor using visual odometry and dynamic obstacle avoidance, pes university," http://www.aerialroboticscompetition.org/assets/downloads/2017SymposiumPapers/PESUniversity.pdf, 2017.

[14] I. Afanasyev and I. Ibragimov, "Comparison of ros-based visual slam methods in homogeneous indoor environment," https://www.researchgate.net/publication/320623436_Comparison_of_ROS-based_Visual_SLAM_methods_in_homogeneous_indoor_environment.

[15] P. Sanchez-Cuevas, G. Heredia, and A. Ollero, "Characterization of the aerodynamic ground effect and its influence in multirotor control," *International Journal of Aerospace Engineering*, vol. 2017, 2017.