Autonomous Navigation of a Multirotor using Visual Odometry and Dynamic Obstacle Avoidance

Pranav Singhania[1]
Department of Information Science Engineering
Siddharth R N[1]
Department of Mechanical Engineering
Sovan Das[1], Akshay Kalkunte Suresh[1]
Department of Telecommunication Engineering

[1]*PES Institute of Technology, Bangalore*

ABSTRACT

In the present day, aerial vehicles such as micro fixed-wings or multirotor systems are important assets for industrial growth. These vehicles are generally autonomous or semi-autonomous and have a wide field of applications in military, mining or forest surveillance. All these applications are highly dependent on GPS (Global Positioning System) localization. Our project involves the design, assembly and configuration of a fully autonomous multirotor capable of manoeuvring in a GPS denied environment without the aid of large stationary points of reference. The multirotor is capable of interacting with external autonomous moving objects using a combination of control and computer vision algorithms. It creates efficient routes in flight for navigation whilst avoiding obstacles during the motion.

1. INTRODUCTION

Team Aeolus will be representing PES University, India at the 2017 edition of International Aerial Robotics Competition(IARC). We describe below our understanding of the problem statement, the planned solution, sensors and other equipment on board and the system architecture we use to solve the mission.

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 4th 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 cm^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°. 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. **Approach**

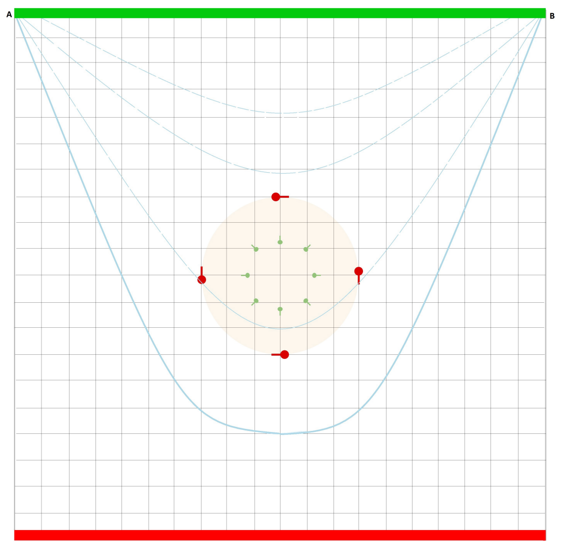

Fig. 1: Navigation algorithm

1.2.1. Path Planning:

We have devised an algorithm which mimics the natural process of herding animals forward. As seen in figure 1, the quadcopter moves in a parabolic path from one vertex of the green line to the other. The paths axis of symmetry is perpendicular to the green line. For the first traversal of the parabola, the directrix of the parabola lies on the red line parallel to the green line. The same parabolic path is traversed twice by the quadcopter, from point A to point B and back from point B to point A. This dual traversal is one pass and with every pass, the focus of the parabola is shifted marginally towards the green line. As the focus shifts towards the green line, the parabola ultimately becomes a straight-line running along the green line.

We detect the ground bots using the camera array. For every ground bot in the field of view of the quadcopter during the parabolic motion, the movement of the ground bot is observed, and if the movement is favorable (towards the green line) the ground bot is left as is, but if the movement of the ground bot is unfavorable (towards any of the red lines), the quadcopter breaks out of its parabolic motion and navigates to the location of the ground bot and taps it once to stimulate a 45-degree clockwise rotation. After tapping the respective ground bot, the quadcopter falls back to following the parabolic path it was traversing prior to spotting the ground bot. This process continues in iteration for every pass of the quadcopter until the path of the quadcopter reaches the green line and moves on top of it. We follow a priority based approach to navigate the quadcopter as per decreasing order of priority, as stated below:

- Obstacle detection and avoidance
- Ground bot detection
- Parabolic path motion

The first priority is to avoid any obstacles. Hence, if any obstacle comes as close as 2m to the quadcopter, the quadcopter flies above the maximum height of the tallest obstacle i.e 2m for a fixed period of 8 seconds thereby allowing the obstacle to have passed the quadcopters path of motion. The quadcopter descends back to the altitude of 1.5m post the period of 8 seconds and continues navigating. In the case LIDAR [1] does not detect any obstacle within the threshold area, the vision system observes the ground bots while the quadcopter is in parabolic motion. If the impending direction of motion of the ground bot is away from the green line, the quadcopter captures the then current location of the ground bot and begins navigating towards it. Once the quadcopter reaches the location, it re-checks for the new updated location of the ground bot at the given time instance, since the ground bot would have made considerable motion during the quadcopters movement towards the ground bot. The quadcopter then reroutes to the new precise location of the ground bot reducing the possible margin of error and descends on top of the bot triggering a clockwise motion. The quadcopter then returns to following the parabolic path from where it broke off when it discovered the ground bot. The entire process described above is performed recursively until the end of each trial.

1.2.2. Obstacle/Ground bot Detection:

The obstacles are detected with the help of omnidirectional LIDAR [1]. During the quadcopters movement described in the previous section, if the quadcopter encounters an obstacle bot, it raises its altitude to 2.5 meters to avoid the obstacle bot for a fixed duration of 6 seconds and then descends to the altitude it was at previously.

We detect ground bots using the camera array. We use the characteristic colors (red, green) of the top plate as an indicator of a ground bot. We threshold the frames from the camera array to detect regions in its view corresponding to the colors on the top plates. As the rest of the arena is otherwise devoid of those characteristic colors, color detection is effective in detecting the ground bots. We use the HSV (Hue, Saturation, Value) color space in detecting the color as it is more robust to detect changes in brightness. Upon thresholding the camera frames, there could be small isolated areas detected as the top plate which could either be due to noise in the image or other small distant objects having the same characteristic color as that of the ground bot top plate. We remove such erroneous detections using morphological opening which is achieved by an erosion followed by a dilation of the frame. We avoid discontinuities in the detection region of the top plate, which could be due to the presence of noise in the image, using morphological closing which is performed by a dilation followed by an erosion of the frame. The resulting detected area corresponds to a single ground bot.

1.2.3. Navigation:

We navigate the arena with the help of visual cues observed from the camera array mounted on the quadcopter, providing a 137° field of view. The most prominent of the cues are the grid lines. We use canny edge detection [2], for detecting the edges in the camera frames and Hough Transform [3] to separate out the lines corresponding to the grids. We pre-process the camera frame by smoothening the image using a gaussian blur filter. This allows the removal of the effect of noise in the grid blocks and makes the task of detecting the grid lines easier. Canny edge detection uses pixel gradients to detected edges in the image and outputs a binary image with thin edges indicated. Hough Transform filters the image with edges to recognize only the grid lines. Duplication of detection of grid lines due to the camera angles is avoided by combining multiple detections in very close proximity into one.

To aid visual navigation, we employ Local Position Estimation (LPE) with the help of an optical flow sensor and a downward facing LIDAR. We track the movement of the quadcopter in the arena using LPE. We compound the position information from LPE, our knowledge of arena dimensions and the visual cues to pin point a location in the arena with a high° of accuracy. We track the motion of each individual object of interest, (the ground bots) by drawing a trajectory joining the positions of the object in successive frames. We then extrapolate this trajectory to obtain a sense of the direction the object is headed towards.

2. PAYLOAD

2.1. **Sensors on-board**

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

2.1.1. Omni-directional LIDAR:

Manufactured by SLAMTEC, RPLIDAR A2 [1] is a 360° 2D laser scanner with a distance range of 0.15m to 6m, distance resolution of <0.5mm, angular range of 360 with resolution of 0.9, scanning frequency of 10 Hz. The sampling frequency is <=4000Hz and the laser used has a wavelength of 785nm and 3mW power adhering to FDA Class I laser safety regulations. It is connected to the processor via USB.

2.1.2. Optical Flow:

Manufactured by 3DR Robotics, PX4flow [4] sensor uses a 752480 MT9V034 image sensor with 16 mm M12 lens fitted with an IR blocking filter, capable of capturing images coupled with a 168 MHz Cortex M4F built-in processor accompanied with 128 + 64 KB of primary memory, the device is capable of capturing and processing 4x4 binned images at 400 Hz. It comes along with a MAXBotix sonar sensor which yields the vertical distance of the quadcopter relative to the ground with a lower threshold of 0.3 m. All of the aforementioned sensors and processors work in tandem to produce the flow data from which the relative velocity and position of the quadcopter in the X-Y plane can be inferred. It is also observed that the validity of the optical flow readings are dependent on the maximum horizontal velocity of the quadcopter and its distance from the ground, for example when flying at a height of 1m, for the position of the vehicle to be estimated accurately, the horizontal velocity cannot be more than 2.4m/s. It is connected to the FCU via I2C (Inter-Integrated Circuit), described in the subsequent sections.

2.1.3. Rangefinder:
Manufactured by Garmin, LIDAR Lite V3 [5], is an optical laser range finder with the resolution of 1cm, accuracy of +/-2.5cm for distances more than 1m, and a measurable range extending between 5 cm and 40 m, and operating at a frequency of 500 Hz. It is used to improve upon the accuracy with respect to the vertical distance of the quadcopter relative to the ground and keep the measurements from the MAXBotix sensor as a fallback mechanism in the event of the LIDAR malfunctioning mid-flight. It is connected to the FCU via PWM connected to the auxiliary rail of FCU.

2.1.4. Camera:
Manufactured by Microsoft, Microsoft LifeCam HD-3000 [6], is a USB camera which outputs frames with a resolution of 1280x720 pixels, at a rate of 30 frames per second. It has a 68.5° diagonal field of view. Two such cameras are used for vision based navigation, to detect the ground bots and estimate their location using the grid lines on the arena. It is connected to the processor via USB.

2.1.5. Depth Sensing Camera:
Manufactured by Stereo Labs, ZED Stereo Camera [7], 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 cameras 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 quadcopter in 3D space. It is connected to the processor via USB 3.0.

2.1.6. IMU Sensors:
The flight controller unit consists of 3 redundant IMU sensors (accelerometer, gyrometer and magnetometer). InvenSense MPU9250, ICM20948 and/or ICM20648 are the first and third set of IMU sensors [8] 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 quadcopter. The gyrometer readings signifying the aircrafts rotation about its own axes are required for a stable flight, used internally by the PID controller. The accelerometer readings provide the acceleration and acceleration forces in all the 3 axes and aid in attitude 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 earths magnetic field and infer the magnetic north.

2.1.7. Barometric sensors:
The flight controller unit consists of 2 redundant MS5611 barometric sensors. It measures the air pressure and in theory can be used to predict the altitude but need frequent re-calibration.

2.2. **Communication Systems**

The communication link used by an aerial vehicle is one of the most vital systems on-board. The unit can be wired and wireless. The wireless system transmits real time acquired vehicle status and it permits the user to perform application level modifications during flight. The primary wireless link which is used to send and receive the acquired data and autonomous control parameters is WiFi. The WiFi unit on board supports 802.11ac [9] with 2x2 MIMO enabling higher data rates and long range. The next communication unit used on board is a 915MHz telemetry which establishes connection between the ground station and flight controller unit. This helps in monitoring the vehicle status and flight parameters. The third unit and one of the most essential sub-unit is the 2.4GHz radio link. This enables connectivity with a remote controller for manual flight. This link has 8 channels and each channel is Pulse Width Modulated. The last wireless communication unit is 433MHz safety switch which is used for flight termination. The wired communication protocols used on-board are USB, I2C and UART (Universal Asynchronous Receiver/Transmitter). The processing unit is connected to the flight controller unit using USB, and the peripherals for localisation connected to the flight controller unit use I2C as it is a robust protocol against noise coupling and also supports upto 127 slave devices connectivity to one master. The UART protocol is used to connect the 915MHz telemetry to the flight controller unit.

2.3. **Processing Unit**

We use the Nvidia Jetson TX1 [10], [11] as the on-board processing unit. It is responsible for processing the video streams from the two cameras, determining obstacles from the LIDAR rangefinder, deciding the path of the quad-copter and accordingly control the flight control unit. 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

The most resource demanding process on the on-board processor is the execution of vision tasks which could greatly benefit from Graphical Processing Unit (GPU) acceleration. In the previous iteration of the competition we used the Intel® NUC™ [12] for our processing tasks but with this iteration we upgraded our processing unit to Jetson TX1 with an Nvidia Maxwell™ GPU. The Jetson TX1 with Nvidia Maxwell™ GPU heavily outperforms the Intel® NUC™ with Intel® HD Graphics 5500 in massive vision tasks. This is evident by the glmark2 gpu benchmark score of 1589 scored by the Jetson TX1 as opposed to the 781 scored by the Intel® NUC™. On a CPU intensive task of counting prime numbers upto 99,999, the Jetson TX1 had an average execution time of 44s as opposed to 107s by the Intel® NUC™ with a Intel® Core™ i3 processor. For these reasons, we have made the decision to use the Jetson platform.

We extend the USB interface on the Jetson TX1 with a 4-port PCI-e USB extender. We interface the 2 USB cameras, the depth sensing camera and the omni-directoinal LIDAR through these ports.

2.4. **Flight Controller Unit**

Pixhawk 2.1 [8], [13], manufactured by Proficnc is the flight controller unit on-board our quad-copter. The role of this unit is to read the input data from the sensors, perform calculations and as an output, manipulate the speed of the motors to execute required maneuvers. It is an open source autopilot with major improvements over its predecessor, Pixhawk. In order to improve the IMU sensor readings, as mentioned in section 2.1.6, Pixhawk 2.1 comes with 3 redundant IMU systems, which are isolated from each other, and are also dampened in order to avoid flight vibrations from corrupting the data. It comes along with a decoupled IMU and FMU (Flight Management Unit) in order to reduce interference to the sensors. It uses a 32-bit ARM Cortex M4 core with FPU operating at a frequency of 168 Mhz, 256 KB RAM, 2 MB Flash memory and a 32-bit failsafe co-processor [14]. The sensors on-board are discussed in detail in the sections 2.1.6 and 2.1.7.

Unlike the Pixhawk, Pixhawk 2.1 removes the power management from the FMU and removes the servo rail as the primary source of backup power for the FMU. It comes with a single independent 5V supply for the flight controller and its peripherals. It also provides under/over-voltage protection, over-current protection and thermal protection. FMU and IO work at 3.3V. It can be powered via USB or through the 2 brick ports. For I/O operations, it comes with an I2C port, 2 CAN ports, 4 UART ports, a console port and a HMI port. The motors are connected to the servo rail and the RC receiver is connected to the RC port. The optical flow connects to the I2C port and omni-directional LIDAR connects to the PWM auxiliary rail.

3. AERIAL VEHICLE

3.1. **Propulsion system**

The multirotor uses $15 \times 5.5''$ 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 [15]. 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. **Control**

The onboard processor generates set points based on the approach as discussed in section 1.2. The Flight Control Unit receives these set points through MAVLINK. Following this, the optical flow receives intermediate dynamic movements dX, dY, Z coordinates through the I2C protocol from the Flight Controller. Altitude feedback is given by the LIDAR Rangefinder which along with the optical flow values use LPE (Local Position Estimator) an Extended Kalman Filter [16], to generate the multirotor 3D position and velocity. The new position and velocity is updated on the on board position estimator along with the readings from inertial sensors. To maintain the global position of the multirotor with respect to the point of takeoff, each intermediate movement

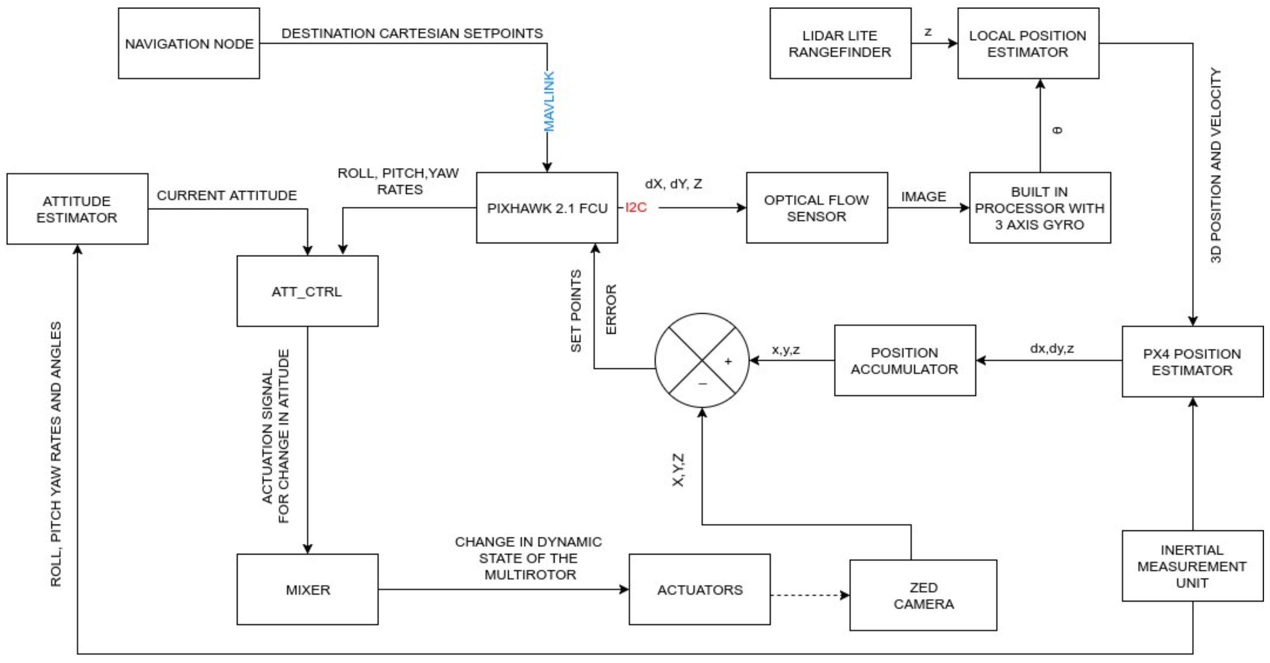

Fig. 2: Control Architecture

in the x and y axes is added into an accumulator which is initialized as the Cartesian origin at the point of takeoff . This accumulator maintains a dynamic record on the current global position in the Cartesian axes. The ZED camera which is mounted on the front of multirotor is used to provide global position feedback. The difference between the Cartesian coordinates in the global position accumulator and the Cartesian coordinates returned by the ZED is the error in local position estimation. The error along with the current altitude inputs back to the optical flow sensor present in the closed loop. Based on the error response the multirotor makes small movements till the global position error is reduced to zero. ID (Integral Derivative) tuning is done to ensure lesser settling time and quicker response in the global position response characteristics.

3.3. **Flight Termination System**

Despite considerable efforts, even the best of engineering designs are bound to fail in certain hostile conditions, due to system malfunctions. To ensure safety of the user and the individuals around the multirotor, several safety measures have been introduced into the system. In a multirotor, there are several sub-systems and failure of any one of them could be hazardous. If the autonomous or path planning system fails, the multirotor is programmed to return to it's starting position and land. In the event of failure of positioning system, the aerial vehicle will be shifted to manual Radio-Controlled mode. For any other breakdown, the power to the Electronics Speed Controllers (ESCs) and motors can be disengaged by using a radio-linked switch which works independent of all other communication links and computing system. This will immediately terminate the flight.

3.4. **Power Supply Unit**

We have used a Lithium Polymer 6-cell (22.2V) battery to power the multirotor. The battery can hold upto 8000mAH of charge and can provide a maximum burst current of 45C. Each sub-unit of the system demands different voltage ratings such as the NVIDIA Jetson TX1 requires 12V at 1.5A, Pixhawk 2.1 requires 5V at 100mA and so on. To distribute this power from the battery at different ratings custom built DC-DC converters have been used. The theoretical upper limit of the flight time with this battery with the configuration of the multirotor mentioned in this paper is about 20 minutes.

4. SYSTEM OVERVIEW

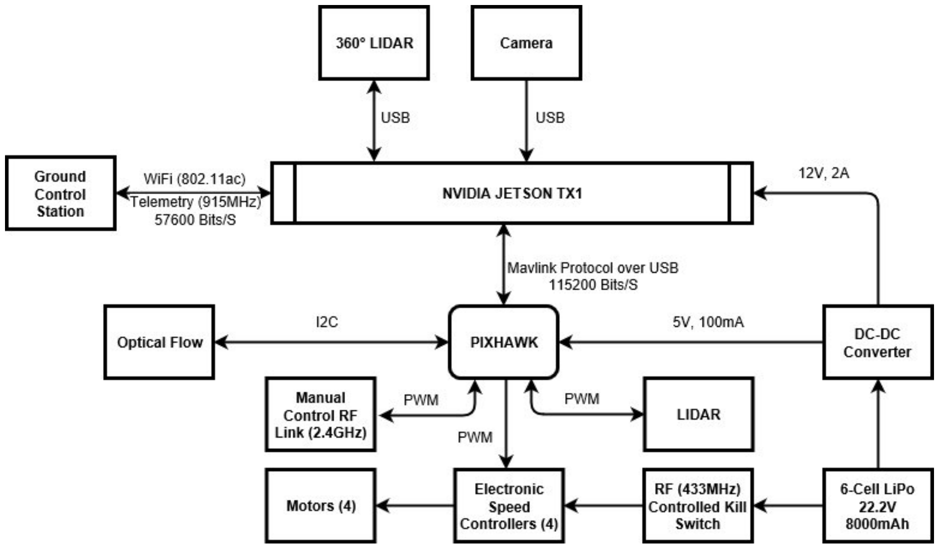

Fig. 3: System Overview

Figure 3 describes the abstract overview of the multirotor and the associated on-board equipment.

5. SYSTEM INTEGRATION AND OPERATIONS

In this section, we discuss the integration of various sensors with the processor and the FCU.

ROS (Robotic Operating System) [17] is a framework for writing robust robot software with ease. ROS supports various libraries that aids interfacing with many devices at an abstraction level. It has focused development in the field of hardware interoperability and extensive documentation on the firmwares of various hardware. We use this as a binding and data synchronisation system for our application that runs on the onboard processor for efficient and fast processing and data in real time. We use the ROS Kinetic release [18] for our application.

OpenCV [19] is an open source library for vision applications written in C/C++ with bindings for C, C++, Python and Java. It is designed for real-time vision tasks and for computational efficiency. It uses OpenCL [20] which enables it to use hardware acceleration, independent of the hardware platform on which it runs. We use the C++ interface of OpenCV 2.4.11 built with Nvidia Jetson TX1 (L4T) support for our vision tasks, which supports CUDA [21] acceleration. The OpenCV program reads the video streams from the camera array and processes the image frames to locate the ground bots as described in section 1.2.2. It communicates with the other systems by publishing the location of a ground bot to a ROS topic.

MAVROS [22] is a subset ROS package which allows two way communication between the autopilots and ground control stations via the MAVLink communication protocol. In our use case, this package plays a pivotal role wherein the onboard processor obtains feedback from the various sensors via their data streams referred to by their respective topic names (ROS Topics, similar to message queues) published by their respective packages running on the system. Our applications subscribe to the necessary data streams in order to take autonomous decisions or carry out precise calculations for navigation, and take countermeasures against erroneous data and for the safety of the multirotor.

The sensor and the associated ROS Topics are as follows:

- RPLIDAR A2 - /scan (sensor_msgs/LaserScan)
- PX4Flow - /mavros_extras/px4flow/raw/optical_flow_rad (mavros_msgs/OpticalFlowRad)
- ZED Camera - /camera/odom

The processor obtains the local position estimation from the FCU through the topic, ‘mavros/local _position/pose’ and guides the FCU to the given location using the topic, ‘mavros/set-point_positio-n/ local’.

Figure 4 describes the code architecture that implements our navigation algorithm.

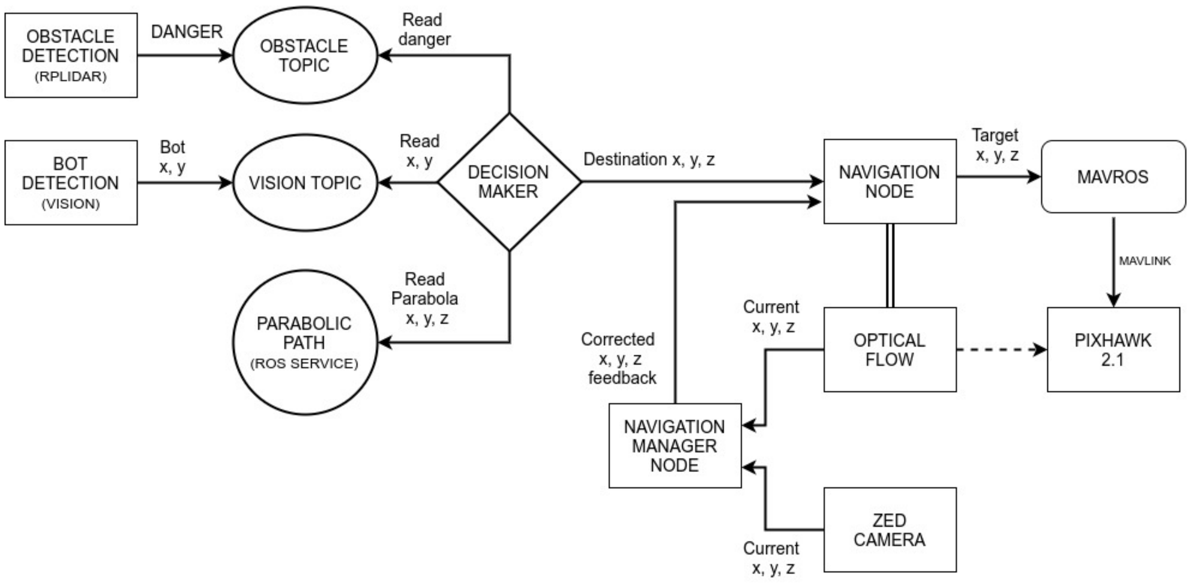

Fig. 4: Navigation Code Architecture

In the earlier model of navigation [12], we had relied on the local position estimator only for position estimation of the destination and the multirotor’s position with respect to the launch site. With this system we had higher probability of error propagation. To overcome this issue, we introduced another redundant local positioning sensor. we use ZED cameras visual odometry and position estimation. Hence, we use two independent position estimation streams for error rectification. This rectification is done by the navigation node manager (fig. 4). The navigation manager node compares the current position of the vehicle with respect to the FCUs sensors with the ZED cameras position estimation at a set interval. By using mean squared error estimator the error is estimated and the flight path is corrected.

6. System Robustness

6.1. EMI/RCI

As discussed in section 2.1.6, the aerial vehicle is equipped with accelerometer, gyroscope and magnetometer, 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 500KHz can corrupt the sensor values. The ESCs and motors draw high amperage of current at a varying frequency of 300KHz to 1KHz, therefore it is extremely likely that the EM waves generated by them can corrupt the sensor data.

Also, the internal or external communication systems can introduce electromagnetic interference. The solution to this setback is electromagnetic shielding 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 IMUs and ESCs strategically the coupling can be reduced. Preferably coaxial cables are used wherever possible and also the source and sink impedance of every AC signal carrying connection is taken care to reduce the reflection due to impedance mismatch.

6.2. Propeller Safety

Figure 5 displays the design for the propeller guards to prevent causing any harm to the by-standers. The propeller guards have been 3D printed using ABS filament given its strength, flexibility, machinability, and high temperature resistance. The guards are held in place with the aid of two carbon fibre rods as shown in figure 5.

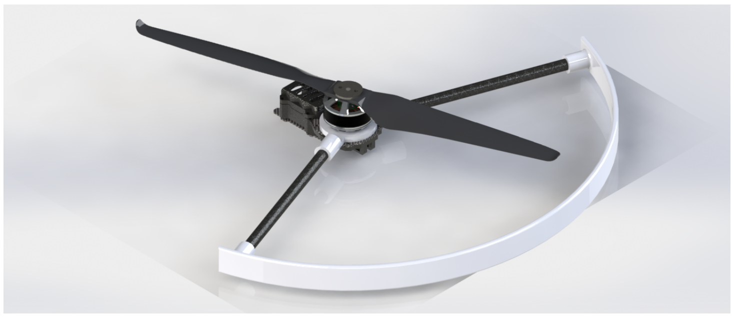

Fig. 5: Propeller Guards

7. Acknowledgement

We extend our gratitude to our college, PES University, for their constant support through the course of this project. We also appreciate the efforts of Syed Hashmath Ulla T A, Anush S Kumar, Sushrith Arkal, Sai Shravan, Shivkumar Umadi, Upasana Sridhar, Sankirthana KN, Sankeerthana M, Nilesh Pattanayak, Suyash Khetawat and Chaitanya Premkumar for their contribution in the development of this project.

References

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

[2] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[3] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic hough transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119–137, 2000.

[4] L. Meier, *PX4Flow Specifications*, https://pixhawk.org/modules/px4flow.

[5] GARMIN, *LIDAR Lite v3 Specifications*, https://buy.garmin.com/en-US/US/p/557294#specs, 2017.

[6] Microsoft, *Technical Data Sheet*, http://download.microsoft.com/download/0/9/5/0952776D-7A26-40E1-80C4-76D73FC729DF/TDS_LifeCamHD-3000.pdf, 2016.

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

[8] proficnc, *Pixhawk v2 Feature Overview*, www.proficnc.com/index.php?controller=attachment&id_attachment=5, 2017.

[9] M. Gast, *802.11Ac: A Survival Guide*, 1st ed. O'Reilly Media, Inc., 2013.

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

[12] S. Das, A. Kalkunte Suresh, and R. Siddharth, "Autonomous object tracking and obstacle avoiding multirotor of team aeolus, pes university," http://www.aerialroboticscompetition.org/2016SymposiumPapers/PESUniversity.pdf, 2016.

[13] proficnc, "Pixhawk v2," http://www.proficnc.com/, 2017.

[14] A. D. Team, "Pixhawk v2 specifications," http://ardupilot.org/copter/docs/common-pixhawk2-overview.html, 2016.

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

[16] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Intelligent Autonomous Systems 13*. Springer, 2016, pp. 335–348.

[17] O. S. R. Foundation, "About ros," http://wiki.ros.org/, 2017.

[18] ——, "Ros kinetic release," http://wiki.ros.org/kinetic, 2016.

[19] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[20] J. E. Stone, D. Gohara, and G. Shi, "Opencl: A parallel programming standard for heterogeneous computing systems," *Computing in science & engineering*, vol. 12, no. 3, pp. 66–73, 2010.

[21] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1365490.1365500

[22] O. S. R. Foundation, "About mavros," http://wiki.ros.org/mavros, 2017.