

B. TECH EXPLORATORY PROJECT ON

**“DRONE DELIVERY SYSTEM”**

BY

**Garvit Singhal (21135054)**

**Harsh Kumar Panchal (21135058)**

**Hemanth Patel (21135060)**

**Manik Chandra (21135079)**

**Sayyam Jain (21135119)**

Department of Mechanical Engineering

Under the esteemed guidance of:

**Dr. Pawan Sharma**

Department of Mechanical Engineering

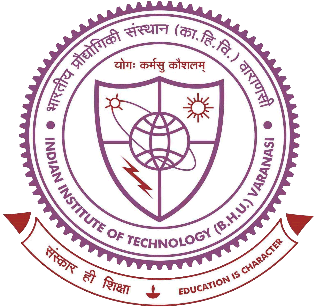
Indian Institute of Technology (BHU) Varanasi

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude towards **Dr. PAWAN SHARMA** (Department of Mechanical Engineering) for their guidance and constant supervision throughout this project. His willingness to share his vast knowledge helped us understand this project and its manifestations deeply. Without his support, we would not have been able to complete this project successfully.

We would also like to express our most profound obligation towards **AEROMODELING CLUB(IITBHU)** and **ROBOTICS CLUB(IITBHU)** for providing us with their significant expertise in drones and providing us with resources which allowed us to bring our imagination into physical existence.

Finally, we would like to thank our friends, family, and people who supported us in our efforts during this project.

**Department of Mechanical Engineering**

**यांत्रिक इंजीनियरिंग विभाग**

**CERTIFICATE**

This is to certify that the project entitled, **“Drone Delivery System,”** submitted for the exploratory project in Part-2 Mechanical Engineering, Indian Institute of Technology (BHU) Varanasi - 221005, is a bonafide project work carried out by the following students under my guidance and supervision:

1. Name – Garvit Singhal (Roll no. - 21135054)
2. Name – Harsh Kumar Panchal (Roll no. - 21135058)
3. Name – Hemanth Patel (Roll no. - 21135060)
4. Name – Manik Chandra (Roll no. - 21135079)
5. Name – Sayyam Jain (Roll no. - 21135119)

Project Supervisor: Dr. Santosh Kumar

Dr. Pawan Sharma Head of Department

Mechanical Engineering

IIT-BHU, Varanasi

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Serial Number** | **Topic** | **Page Number** |
| 1. | Abstract | 5 |
| 2. | Introduction | 6 |
| 2.1 | Applications | 6-7 |
| 3. | Material and Methods | 8 |
| 3.1 | Description of hardware components | 8-9 |
| 3.2 | Description of software components | 10-13 |
| 3.3 | Localization | 14-16 |
| 3.4 | Mapping | 17-18 |
| 3.5 | Motion Planning and Obstacle Avoidance | 19-20 |
| 4. | Conclusion | 21 |
| 4.1 | Difficulties Faced | 22 |
| 4.2 | Future workplan | 23 |
| 4.3 | Achievements | 23 |
| 5. | References | 24 |

**ABSTRACT**

Our project involves creating a multirotor flying drone that can hover stably, use image processing to find and pick up an object using an electromagnet or clamps, and navigate to a bin to drop the object based on image processing. The drone must be activated through a soft signal or have a push button switch to start. Our drone must have a picking device, such as a claw or electromagnet, to pick up packages weighing less than 50 gm. The package is a thermacol block with a thin MS sheet on top to make it magnetic. The metal can be covered with coloured paper. There will be a total of two packages - one with an Aruco code and one that is coloured, loosely held in place in a slot to ensure the draft of the drone does not throw them around. The final task involves the drone taking off from a designated landing pad, picking up the packages one at a time, dropping them in the drop zone, repeating these steps until all packages have been dropped, and returning to the base without touching the ground. The arena size is 10x10ft with a 12-inch fence around it, and the landing pad is a circle of diameter 2 feet, while the drop zone is 2x2ft with a 12-inch fence around it.

**INTRODUCTION**

Drones have become increasingly popular due to their versatility and ability to perform a wide range of tasks. However, flying a drone indoors can be challenging due to the lack of GPS signal and obstacles in the environment. In this project, we aimed to overcome these challenges by automating the flight of an F450 drone using an onboard computer and enabling it to fly indoors with the help of Intel RealSense for localization, as well as April Tags and Aruco Markers. This project demonstrates the potential of using advanced computer vision and autonomous flight control technologies for indoor drone operations, with potential applications in various industries such as search and rescue, inspection, and delivery.

**Applications**

1. **Search and Rescue:** In emergency situations, drones can be deployed to quickly scan indoor structures for survivors, and this project can help navigate the drone around obstacles and localize the survivors' position accurately.
2. **Inspection:** The use of drones for inspection purposes is increasing, and the ability to fly indoors with precise localization can help inspect hard-to-reach areas such as pipelines, ducts, and ventilation systems.

**Figure 1: Search and Rescue**

1. **Delivery:** Drones have been used for delivering packages, and this project's technology can enable delivery in indoor environments, such as delivering medication within hospitals.

**Figure 2:Medical Deliveries**

1. **Security:** Drones can be deployed to enhance security in indoor spaces, such as monitoring restricted areas, and with this project's technology, the drone can navigate through the environment and scan the area for suspicious activities.

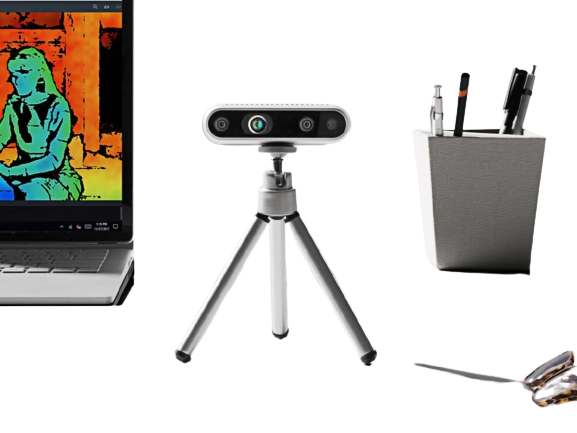
**MATERIALS AND METHODS**

**Description of Hardware Components**

1. **Pixhawk 2.4.8 flight controller:** The Pixhawk 2.4.8 is an open-source flight controller used in drones that offers precise and reliable control through a range of sensors including accelerometers, gyroscopes, magnetometers, barometers, and GPS. It supports a variety of communication protocols, making it easy to integrate with other hardware and software systems. 

**Figure 3:Pixhawk Flight Controller Figure 4:Jetson Nano**

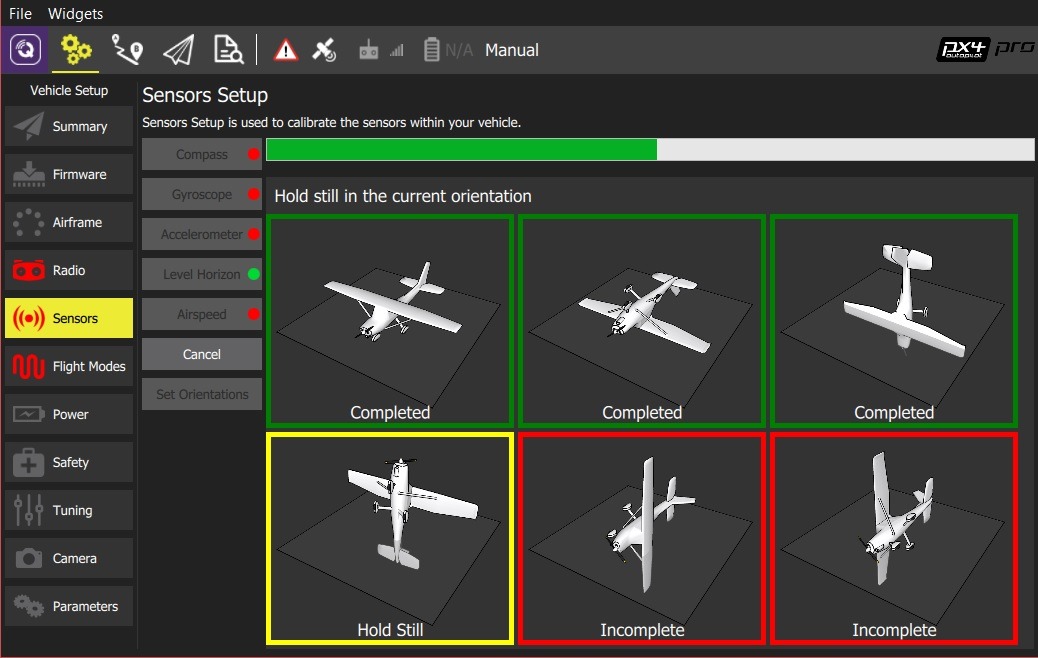
1. **NVIDIA Jetson Nano:** The Jetson Nano is a small but powerful computer designed for edge computing applications. It was used as an onboard computer on a drone to process vision position and perform waypoint navigation. Equipped with a camera, it processed images to determine the drone's position and orientation. The Jetson Nano ran on Ubuntu Linux and used ROS for coordination between processes, providing real-time navigation and control.
2. **Intel Realsense D435i:** The Intel RealSense D435i is a depth-sensing camera used in various applications, including robotics and drones, for accurate spatial mapping. In the project, the D435i camera was mounted on the drone to detect April tags and estimate vision position in real-time. The camera also provided raw imu data for mapping or sensor fusion, improving the localization system.

** **

**Figure 5:Intel Realsense D435i Figure 6:Lidar Sensor**

1. **TF02-Pro LIDAR Distance Ranging Sensor:** The TF02-Pro LIDAR Distance Ranging Sensor is a high-precision ranging sensor designed for use in drones and other robotics applications. It utilizes LIDAR technology to measure distances with high accuracy, even in challenging lighting and weather conditions. In our drone, the TF02-Pro sensor was used to measure the height of the drone above the ground, which was then fused with EKF parameters in the PIXHAWK flight controller for pose estimation. This enabled the drone to maintain a stable and precise height above the ground, even in GPS denied conditions.
2. **30A Electronic Speed Controller:** 30A ESC is a component used in drones and other remote-controlled vehicles to control the speed of the motor.
3. **935KVA BLDC motor:** BLDC (Brushless DC) motors are commonly used in drones due to their high power-to-weight ratio and efficiency. They provide the necessary propulsion for the drone's movement and are controlled by the flight controller to maintain stability and manoeuvrability during flight.

**Description of Software Components**

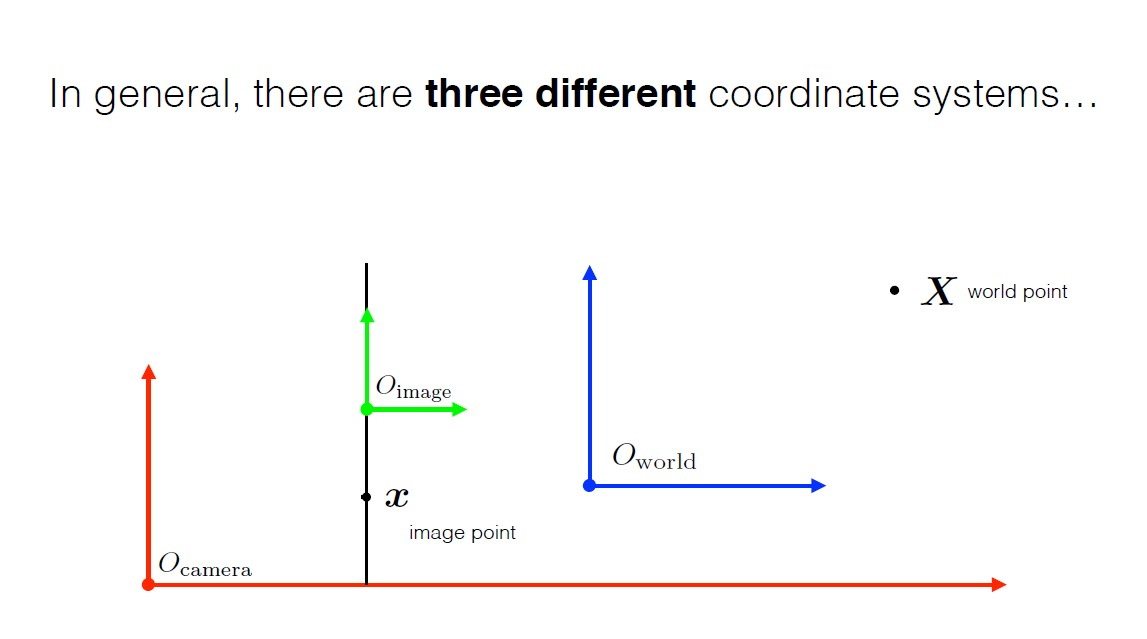
1. **Robot Operating System:** ROS (Robot Operating System) is an open-source robotics framework used for developing and controlling robotic systems. It provides a collection of tools, libraries, and conventions that help developers build complex robotic systems easily. ROS allows developers to write software in a modular and reusable way, making it easy to integrate various sensors, actuators, and other components. In our drone, we used ROS on the onboard computer for coordination between all the ongoing process. Subscriber publisher model helped us to efficiently transfer data be it camera images or odometry from node to node and communicate with ground control station using MAVROS (MAVLink ROS Interface). We installed the ROS environment on the drone's onboard computer and used the AprilTag ROS package to detect the visual markers. We also used MAVROS to communicate with the drone's autopilot, which enabled us to control the drone's movement and perform waypoint navigation. The use of ROS in our drone allowed us to develop a robust and flexible system that could adapt to different use cases and environments. By leveraging the power of ROS, we were able to integrate various sensors and components seamlessly, enabling us to develop a fully functional and reliable drone platform**.**
2. **MavROS:** MAVROS (MAVLink ROS Interface) is a ROS (Robot Operating System) package that provides a communication interface between the ROS environment and MAVLink-based systems such as Pixhawk and Ardupilot. MAVLink is a protocol used for communication between unmanned vehicles and ground control stations. It is designed to be lightweight and robust, making it ideal for use in small embedded systems such as drones. In our project, we used MAVROS to communicate with our drone's autopilot (PX4), which was running on the Pixhawk flight controller. We used MAVROS to send commands to the autopilot, such as setting waypoints for the drone to follow, and receiving status updates from the autopilot, such as the drone's acceleration imu data and battery level etc. We also used MAVROS to send a vision\_position message to the Pixhawk flight controller. This message contains information about the drone's position and orientation in space, as determined by our onboard vision system using april tags. By sending this message to the Pixhawk, we were able to provide the autopilot with more accurate position and orientation data in GPS denied environment. This allowed us to perform more precise and reliable flight .
3. **QGroundControl:** In our project, we used QGroundControl to set parameters for our drone's onboard computer to take position data from the vision position system, which was based on the Intel RealSense D435i camera and April tags. We configured the QGC to receive and process the vision\_position message sent by our onboard computer, which contained the position and orientation of the drone as determined by the vision system. By using QGC in conjunction with our onboard vision system, we were able to develop an autonomous drone platform that could localize using April tags and navigate to waypoints based on the vision data. This allowed us to perform a wide range of autonomous tasks, from simple position hold and waypoint navigation to more complex missions such as search and rescue and environmental monitoring. Overall, QGroundControl represents a powerful tool to develop autonomous drone systems. Its flexible and customizable interface, combined with its real-time telemetry monitoring capabilities, makes it an ideal platform for developing and testing autonomous drone missions.

**Figure 7:QGroundControl**

1. **PX4 Firmware:** PX4 firmware is an open-source autopilot software that provides a flexible and modular framework for controlling unmanned vehicles. It includes a range of flight control algorithms that enable precise control of the vehicle's attitude, altitude, and position, as well as support for GPS navigation and telemetry data transmission. It also includes a modular architecture that allows for the addition of custom modules and plugins, enabling developers to tailor the firmware to their specific needs. To make an autonomous drone based on visual odometry, we used PX4 firmware as the flight control software. We integrated a visual odometry system, which was based on the Intel RealSense D435i camera and April tags, with the firmware to provide accurate position and orientation information for the drone. We configured the PX4 firmware to receive and process the vision\_position message sent by the onboard computer, which contained the position and orientation of the drone as determined by the visual odometry system. Using the position and orientation information from the visual odometry system, we programmed the drone to perform autonomous tasks, such as takeoff, landing, and waypoint navigation. The flexibility and customizability of the PX4 firmware allowed us to tailor the drone's behavior to our specific needs, enabling us to develop a highly capable and reliable autonomous drone platform.
2. **Apriltags\_ros :** The apriltag\_ros package/stack is a popular open-source software library in the field of robotics and computer vision. It provides a powerful set of tools for detecting AprilTags, a type of visual marker that can be used for localization and navigation. We used the apriltag\_ros package in our drone to detect and identify AprilTags, and to estimate the drone's position and orientation in real-time. Using the package, we were able to extract high-precision visual odometry data from the camera, which we then used to control the drone's flight path. We used ROS to publish this data, enabling other software components on the drone to access it and use it for various purposes, such as controlling the drone's motion and trajectory. Overall, the apriltag\_ros package was a crucial component in our drone's localization and navigation system, allowing us to achieve accurate and reliable control over the drone's movement and behavior.
3. **OpenCV:** OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It has a wide range of functions and tools for image processing, object detection, and machine learning applications. OpenCV is used in a variety of applications, including robotics, surveillance, and augmented reality. In our drone, we used OpenCV to detect and track colored boxes in the camera's field of view. We used the library's color filtering functions to isolate the colored boxes in the image, and then used its feature detection functions to extract the pixel coordinates of the boxes. Once we had these pixel coordinates, we used a combination of camera calibration and coordinate transformation techniques to convert them to global coordinates relative to the drone's origin. Overall, the use of OpenCV in our drone allowed us to achieve accurate and reliable detection and tracking of colored objects in real-time, providing valuable information for navigation and control. The library's extensive feature set and ease of use make it a popular choice for computer vision applications in robotics and beyond.

**Localization**

Our team has developed an autonomous drone for indoor delivery purposes using Pixhawk 2.4.8 and px4 firmware. Since we cannot use GPS indoors, we had to find alternative localization methods that would allow us to achieve accurate box picking while matching Pixhawk's required vision pose rate of 40-60 Hz. The size of the arena was 10 x 10 feet, and the boxes were 3x3 inches, so we needed an error less than 3-2 inches for accurate box picking.



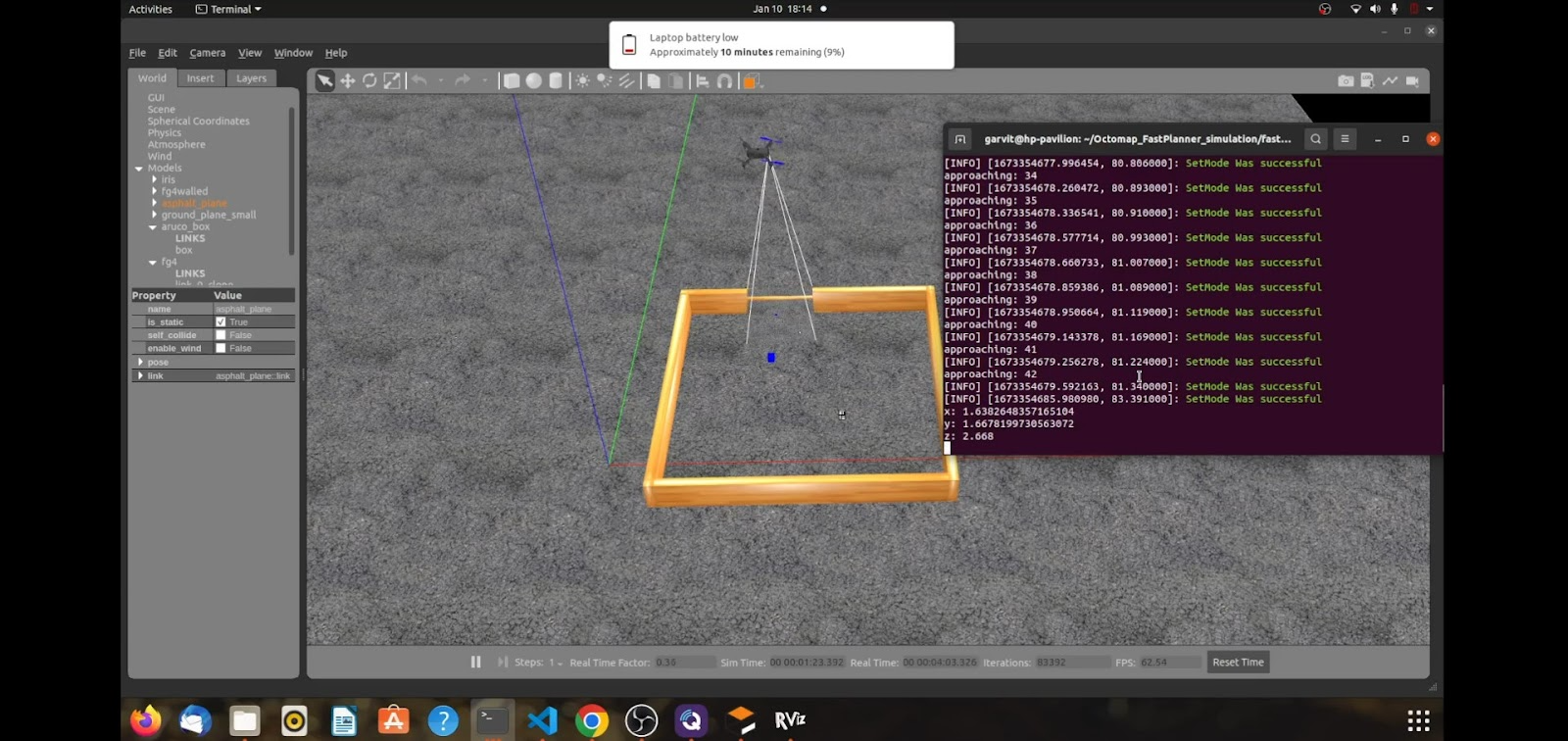
**Figure 8:Coordinate Systems**

Our initial approach was to use two stereo images provided by the Intel RealSense D435i camera for the RTab mapping in ROS to generate visual odometry. RTab mapping is a ROS package that provides real-time dense 3D mapping using RGB-D cameras. Using RTab mapping required a high computational power, and the desired rate did not match Pixhawk's requirement. This approach was also limited to situations where some disturbance or obstacle is present in the camera frame for the vision pose to be updated. Despite these limitations, we found that this approach worked fine on our laptop.

We then switched to the idea of using voxblox mapping and localization in that map. Voxblox is a volumetric mapping library that provides a real-time 3D mesh of the environment. Our plan was to create a voxblox map and then use it for localization. However, we found that working with the depth point cloud from the Intel RealSense D435i required high computation, which was not feasible for our project.

We then proceeded to use Apriltags localization. Apriltags are visual fiducial markers that are widely used for localization in robotics. We created a grid of 18 Apriltags, each of size 14.3 cm. We placed 16 tags inside the arena in a 4x4 configuration and the remaining two outside the arena for landing and dropping zone location. In ROS, we fed the location, size, and orientation of each tag in the tag bundle. When the overhead camera on the drone moves and the Apriltags come into the camera frame, the average of the pose from each tag is taken, and it is published on a ROS topic called "/tagOdometry".

To accurately determine the height of the drone in mm, we used a 1-d LIDAR. When the drone gets tilted, the apparent height increases, so by knowing the angle of tilt, we calculated the actual height. A ROS node was written to calculate the actual height and fuse it with /tagOdometry. The fused data was then published to the vision pose through Mavros.



**Figure 9:Localization**

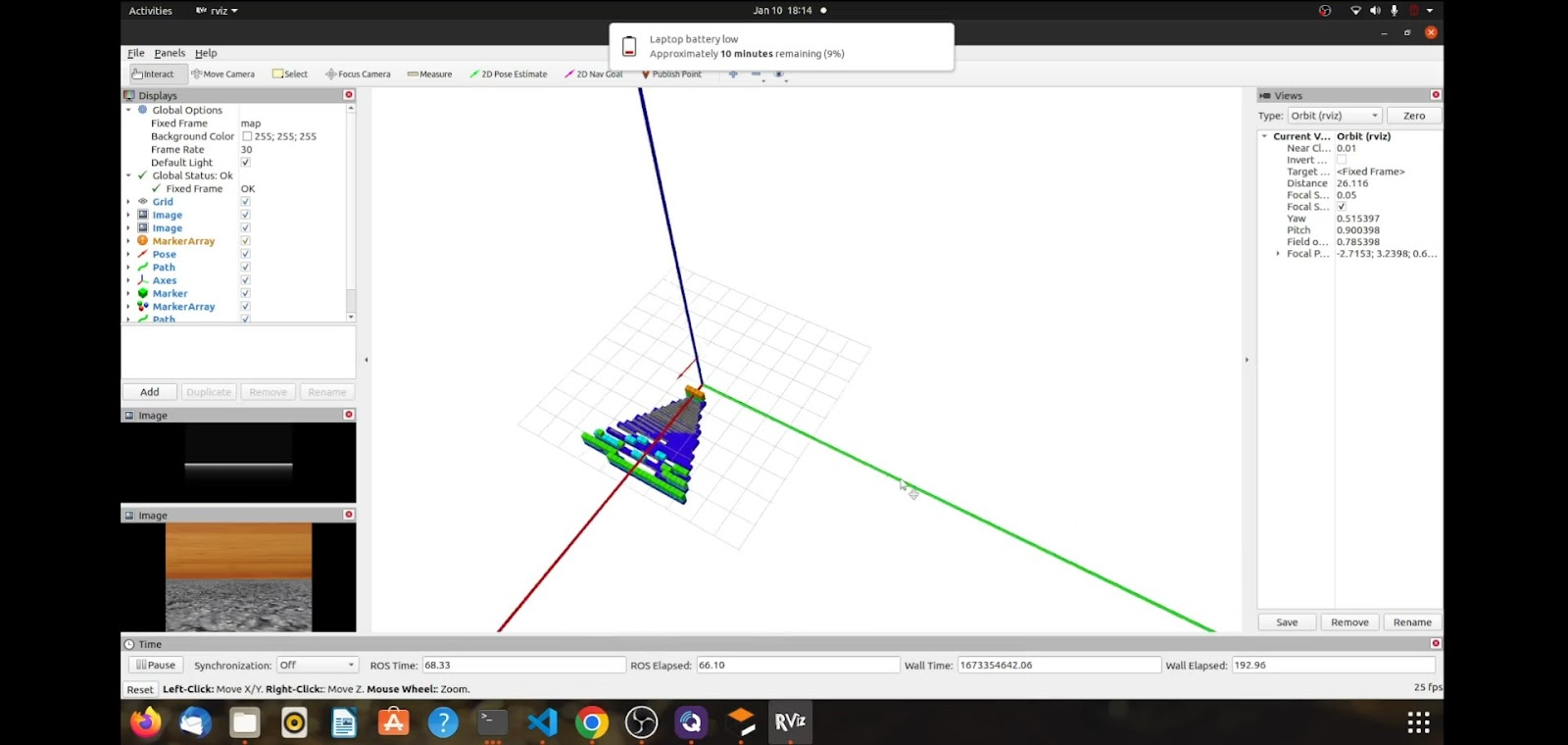
Mavros is a ROS package that provides communication drivers for various autopilots with MAVLink communication protocol support. The vision poses published on Mavros receives to the Pixhawk. Pixhawk has an internal IMU device, and using the EKF filter, this vision pose fuses with IMU data to generate a local pose at the desired rate by px4 firmware.

In order to accurately determine the drone's position using Apriltags, it was necessary to perform camera calibration using a monocular camera calibration. Two chessboard grids of different shapes and sizes were used to calibrate the camera accurately.

In conclusion, our team utilized various localization methods to develop an autonomous drone for indoor delivery purposes. Our initial approach of using RTab mapping was limited by high computational power requirements, and we then explored the use of voxblox mapping, which also proved to be computationally intensive. Finally, we settled on using Apriltags localization, which allowed us to achieve the desired accuracy while matching Pixhawk's required vision pose rate.

**Mapping**

We began our exploration of mapping methods with octomapping, a technique that is highly compatible with both ROS Melodic and ROS Noetic. Octomapping involves creating an occupancy grid using a probabilistic octree data structure. Each voxel in the octree represents a small region of the environment, and the state of each voxel can be either free (unoccupied), occupied, or unknown. By updating the state of voxels based on sensor data, octomapping enables the creation of dynamic maps of the environment.

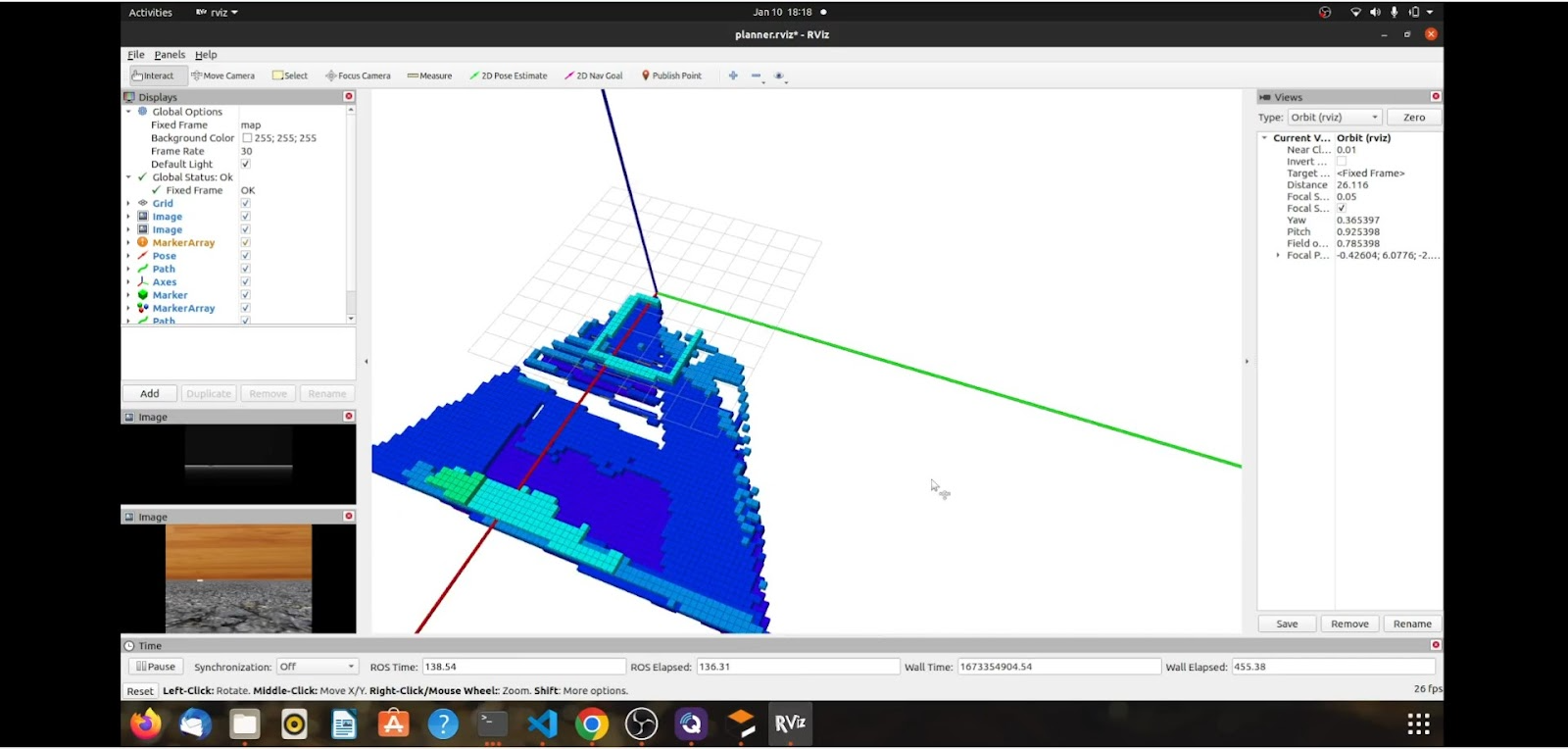


**Figure 10:Rtabmapping**

We utilized octomapping in conjunction with fast planning for obstacle avoidance. Our aim was to enable the drone to navigate to a point-to-point location using the generated dynamic map to pick up the boxes. However, we encountered a challenge when the size of the voxel boxes in the map was larger than the size of the box we wanted to locate. Decreasing the voxel size led to high computation power requirements, and so to overcome this issue, we used a downward-facing camera to locate the box. We used octomap solely for obstacle avoidance, but we were unable to use mapping on hardware due to the high processing power requirements. Since there were no obstacles in the environment, the obstacle avoidance component was overcleaning.

We also explored voxblox mapping, which was compatible with ROS Melodic. This technique involves generating a map of the environment on a laptop, which is then used for localization. However, since our drone was on ROS Noetic, we did not use voxblox mapping.

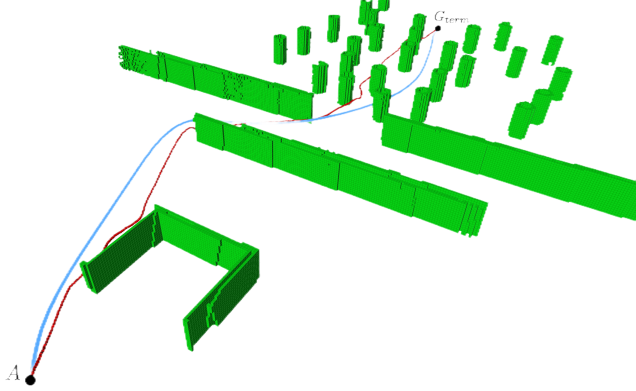
In conclusion, we explored octomapping and voxblox mapping techniques for mapping in our autonomous drone delivery system. While octomapping showed promise for obstacle avoidance, we faced challenges related to the size of voxel boxes and computation power requirements. We were unable to use voxblox mapping due to compatibility issues. Ultimately, we chose not to use mapping on hardware and instead relied on sensor data to navigate the drone in the environment.



**Figure 11:Octomap of arena**

**Motion Planning and Obstacle Avoidance**

In the simulation phase, we utilized octomapping with fast planning to generate a dynamic map that updated as the drone progressed through the mission. Fast planning is an optimization method that improves the efficiency of the path planning algorithm, allowing for the shortest path to be obtained in the unoccupied voxels for obstacle avoidance. The map was created on a laptop, and the A\* algorithm was used to generate the shortest path to the goal point. The A\* algorithm is a widely used pathfinding algorithm that calculates the shortest path between two points, taking into account the obstacles present.



**Figure 12:Obstacle Avoidance**

A ROS node named Planner was developed to generate the planned path, which was then passed to a controller node to execute it. The controller node subscribed to MAVROS topics, which allowed it to take the path and follow it. The sub-goal points in the path were sent through MAVROS set goal points topic, and the service ensured that each point was reached before moving onto the next. The downward camera was used for Apriltag localization and for detecting coloured boxes and Aruco boxes in the local map. Using OpenCV, the coloured boxes in the image were masked out, and the distance to the box from the drone was calculated using meters to pixel ratio. The distance was then transformed to the world frame to generate the goal point array for the coloured boxes. The Aruco box location was detected using the aruco\_ros package, and the information was updated in the goal array. A master control node was developed to generate the goal array, and the planner and controller nodes were called to instruct the drone to move to the box locations in the required order using the generated path.

In the actual hardware, we faced some limitations due to the high computation power and power supply requirements. We were unable to use Intel RealSense for mapping, so we relied solely on Apriltag localization. Obstacle avoidance was also deemed unnecessary due to the constraints of the hardware. Therefore, we developed a direct node that enabled the drone to take off in onboard mode and fly to the center of the arena where the entire arena was covered, and the goal array was generated. The drone was then sent to the box location using the MAVROS set goal point, where it executed its delivery task.

In conclusion, our team has developed a comprehensive motion planning and obstacle avoidance system for the autonomous drone, which utilizes various algorithms and components to enable efficient delivery operations. The simulation and actual hardware phases were both challenging, but we were able to overcome the obstacles and develop a functioning system. The report provides a detailed overview of the methodology used and can be used as a reference for future research and development in this field.

**CONCLUSION**

In conclusion, our project successfully demonstrated the potential for using advanced technologies for indoor autonomous flight of drones. We were able to modify the F450 drone and develop software to enable it to fly autonomously and navigate around obstacles. However, we also faced challenges in delivering payloads due to limitations in the design of our electromagnet, which impacted the efficiency of the drone. Future improvements to the localization system could enhance the accuracy and precision of payload delivery, making this technology even more valuable in various industries, including search and rescue operations and indoor structure inspections. Overall, our project contributed to the advancement of drone technology and highlighted the potential for continued innovation in this field.



**Figure 13:Final Arena**



**Figure 14:Final Prototype**

**Difficulties Faced**

1. **Lack of GPS:** GPS signals are often weak or non-existent indoors, which makes it difficult for the drone to accurately determine its location. This means that the drone needs to rely on other sensors such as accelerometers, gyroscopes, and cameras to navigate its surroundings. The position estimated by OpenCV was at lower frequency than required by EKF for sensor fusion resulting in inaccurate position.
2. **Communication Interference**: Indoor environments can also have interference from other wireless devices, such as Wi-Fi routers or Bluetooth devices, which can disrupt the communication between the drone and the controller.
3. **Power Constraints**: The sensors and computing components needed for autonomy can consume significant amounts of power, reducing the drone's operating time, especially Jetson Nano, on which onboard processing is done, leading to its 100% CPU usage.
4. **Lighting Conditions:** Indoor environments can have varying lighting conditions, affecting the drone's ability to perceive its surroundings. For example, low light conditions can make it difficult for the drone's cameras to see obstacles, while bright lights can cause glare and distort the images captured.

**Future Workplan**

1. Using a Buck-boost voltage regulator to maintain a steady voltage supply to all components reduces voltage fluctuation and thus less power interruption on nano and minimizes the energy lost as heat. This can prevent voltage spikes or dips that could damage or disrupt the operation of the components.
2. By dedicating one computer to handle mapping, the other computer can be used exclusively to process data from Realsense, thus reducing CPU usage and heating. Additionally, using two onboard computers can allow for parallel processing of data, which can significantly improve the speed and efficiency of data processing.
3. Using a Mo-Cap system in combination with a nano device can provide more accurate and real-time tracking of motion, which can be useful for applications such as autonomous navigation, obstacle avoidance, and target tracking.

**Achievements**

Our team is proud to have achieved significant success in two prestigious competitions. At the **Robotics Conclave Technex 2023** hosted by IIT BHU, our team secured the **1st place** with our exceptional robotics project. Additionally, we were among the **top 5 teams under 3000** participants in the highly competitive **Flipkart Grid 4.0** challenge. We are motivated to continue to push the boundaries of innovation and achieve even greater success in future projects.

**REFERENCES**

1. <http://wiki.ros.org/apriltag_ros> - Apriltag\_ros package
2. <http://wiki.ros.org/aruco_detect> - Aruco\_detect package
3. <https://github.com/IntelRealSense/realsense-ros/tree/ros1-legacy> - IntelRealsense
4. <https://github.com/ros-drivers/usb_cam> - usbcam\_ros docs
5. <https://github.com/introlab/rtabmap_ros> - rtabmap
6. <https://github.com/ethz-asl/voxblox> - voxblox
7. <https://github.com/OctoMap/octomap> - octomap
8. <https://github.com/mavlink/mavros> - mavros
9. <https://docs.px4.io/main/en/companion_computer/> - px4 firmware docs
10. <https://github.com/behnamasadi/ROS_monocular_camera> - camera calibration