Flipkart GRiD 2.0

Autonomous Indoor Drone - Round 3 Template

Team Name: AeroBITS

Institute Name: BITS Pilani

Team Member Details

All the members in our team are active members of the Radio-Control Club at BITS Pilani. We involve in building drone and aerial Vehicle as hobby and participate in different competitions

1. **Akshit Patel**:

a.) Internship at CSIR – CEERI. CSIR is ranked 12th according to Scimago in World Among Government Institution- Department Of Science & Technology.
b.) Research paper- Under review process. Title - Parameter Estimation and Comparative Analysis of Control Design Techniques for BLDC Hub Motor. In this research I and co-authors designed a way to estimate motor parameter and then simulate it in MATLAB. After simulating motor, we designed a closed loop control of motor using various normal and advance PID techniques. Advance methods include self-tuning Adaptive-Fuzzy PID and Genetic Algorithm for controlling motor.

Motor used was part of E-tricycle for disabled and elderly. [LinkedIn] [Research paper link]

2. **Effy John:** [LinkedIn]
3. **Raunak Banthia**: Drone Enthusiast pursuing Civil Engineering. Built multiple fixed wing RC planes. Intermediate proficiency in multiple CAD softwares including Autodesk Fusion360, AutoCAD, SolidWorks and Revit. Solid gasp of concepts involving structural analysis. [LinkedIn]
4. **Rishabh Jain:** [LinkedIn] Internship at Ultratech Cement where I tries to increase the efficiency of motors by applying SPRS and other methods.Currently pursuing major in "Electronics and instrumentation" branch at Bits Pilani .

Some of the accolades of team (comprised of four member as mentioned above)-

1.) Battle of Waterloo (boat racing) in Apogee'19, acquired 1st position
2.) Hovercraft competition Apogee'19, acquired 2nd position

Past Project we four as a team undertook -

1.) Quadcopter using KK 2.0 flight controller
2) Line follower and path finding robot using Arduino and IR sensors
3.) Currently working on building pesticide spraying hex-copter

On - going projects https://drive.google.com/drive/folders/1oktp8P8gfRQQ1KgVw75gvh7IcxKnPhgI?usp=sharing

Disclaimer

- We won't be using expensive Occipital Core 3D for tracking and depth image data. We would be using simple monocular images and depth data provided by Kinect camera for avoiding obstacles in between gates
- This eliminate use of optical flow sensor and lidar lite as well as velocity estimation will be directly obtained from VIO using OpenVins [1]
- GPS, Lidar and Optical Flow have been removed as they were adding to unecessary cost
- Limited computing will be done on raspberry pi and computational heavy process will be shifted to our PC's and communication will be done through Wi-fi
- We are using same drone frame but with different motor and thrust combination. [6]
- Currently the dataset used is EUROC[2] for VIO which is different from KITTI or Tartan Air as mentioned in previous proposal
- Obstacle avoidance has been redesigned using Kino dynamic path searching algorithm[3] and B-Spline Optimisation, which is different from avoidance library as mentioned in our last proposal.
- Updated Architecture can be found in Appendix A, and is explained in proof of concept video
- No separate safety switch for protection as we will directly command it from GCS or terminal if anything goes wrong
- Every Data mentioned by us in this slide is present in https://drive.google.com/drive/folders/1oktp8P8gfRQQ1KgVw75gvh7lcxKnPhgI?usp=sharing

Technical Specifications

1.) A Powerful combination of ROS and PX4. ROS will be running on onboard companion computer i.e., Raspberrypi and will communicate to ROS running on PC. Flexible shifting of computation will be done using ROS so as to confirm there is no lag in signals and actuation

2.) Complete indoor navigation without GPS. We will use highly efficient pre trained models OpenVIns for odometry estimation using monocular lens data.

3.) Complete testing of algorithm is done using SITL(Simulation in the loop) PX4 and gazebo using python and C++. This also enable us to perform HITL(Hardware in the loop) testing which is basicallytesting hardware with simulated one before putting it to actual drone. This will enable us to shift computation between onboard and offboard companions as required.(Mostlyobstacle avoidance + gate detection in offboard and VIO estimation onboard)

4.) Auto take-off and landing feature with the help of Ground Control Station or terminal on offboard companion(PC). Commands from offboard will be communicated via Wi-Fi

5.) We can have complete control of drone with virtual joystick and command drone manuallyif anything goes wrong. Also we can simplyactivate the failsafe mode(in our case its set to landing) by simplyturning off the inputs from companion(PC)

6.) QGroundControl enables us to establish Ground Control Station both on desktop and mobile (both IOS and Android) for sending mission commands to for travelling certain specified waypoints with GPS(for outdoor), live video stream, arming, disarming drone and manymore other features. Though controlling companion computer with signals from QGC mobile might be difficult but possible.

7.) GPS based QGround Control features like mission planning, specified trajectoryand other autonomous features willbe done with the help of fake GPS plugin. Basically we fake PX4 with simulated GPS data formed with VIO.

8.) The drone features a retractable landing gear which can be controlled remotelyor to be programmed to work automatically. This improves the maneuverabilityof the quad and improve the field of view of the camera.

9.) OpenVins for visual inertial odometryusing monocular lens data. This libraryhas been tested extensively on EUROC dataset for aggressive racing drone. It produces odometrydata with 7% error in aggressive flight. Considering our flight won't be that aggressive we will have results with error less than 2%

10.) Obstacle avoidance is featured through Fast Planner from HKUST robotics group. We used Kino dynamic path planning and B spline optimization for generating smooth trajectories.

11.) Drone features is expected to fly for 10min (in our case as it will not have changes in thrust) in single charge with payload capacity of 5kg with autonomous control

Aerodynamic & Payload Calculations

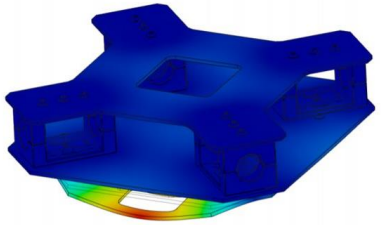
1. To design the drone we have taken 2 things to be absolute, it should carry a payload of 2 kg along with its all-up-weight and it should be able to go across the gates with ease.
2. To calculate the maximum payload that the drone can carry at best efficiency, we have used eCalc calculator. The analysis requires basic data of the components selected to get the ideal result. eCalc calculator for xcopter is used extensively by drone enthusiasts to test their prebuilds. The base constraint while choosing the components is the payload. For the drone to support a certain weight we consider the frame next followed by the type of motor that defines the thrust produced.
3. The all up weight comprises of the frame + basic components including camera(2.5kg)and the payload (2kg).
4. The thrust produced per arm by the drone is 2.5 kg and the total thrust is 10 kg.
5. The thrust to weight ratio for a quadcopter with a payload is kept at a lower value to gain more stability.
6. The frame weight is exceptionally light as it adopts a Toray 3K carbon fiber cloth woven on a carbon fiber board.
7. The drone flies in a ‘X’ formation so the effective length is less than the diameter of 600mm (not including propellers).
8. The center of gravity axis passes through the center of the drone with a little offset due to the battery which has been kept on the top part of the drone.
9. As the drone frame is symmetric in nature the only asymmetric components are the battery and the payload attached at opposite ends of the drone. The container will feature straps to keep the payload as centered as possible. The drone has a clearance of 170 mm when it is stationary.
10. The container will have more than enough space to hold the payload.

Basic Calculations and Motor, ESC combo for required flight time

- Battery Calculations [8]
  1. Rating 60C 4000mah
  2. Two such batteries will provide 8000mah
  3. Current discharge at any point possible= 60C*4000mah*2 = 480A
- Thrust Calculation [11]
  1. Total Weight of drone 4.5kg
  2. Thus we target for 10kg thrust, hence 2.5kg per arm
- Motor Calculation [9]
  1. Each motor will draw 15.1A for creating 2.5kg thrust
  2. Four Motor will draw 60.4A which can be easily provided by batteries
- Thrust to weight ratio approximately 2:1
- As if each motor draw 15.1A (or a bit more than that)then we will be able to reach sufficient height and then hover there with 1:1 ratio. So, max current wont in any case exceed 30A for us. More the current more torque and hence more thrust. So esc should of 30A
- Flight Time approximate with this data considering motor require at each point of time 60.4A will be
  1. 8A current can be provided for 1 hour
  2. So, (8/60.4)*60 = 7.9 min, so this is the expected flight time each motor produce 2.5kg at each point of time.
  3. So flight time is expected to be more than that.

Structural & Stability Analysis (with & without payload)

- The frame adopts a Toray 3K carbon fiber cloth woven on a carbon fiber board with 3K hollow twill pure carbon fiber carbon fiber. The design is perfected by CNC machining and the design standards higher than similar products, the full set of rack weighs only 476 grams. Carbon fiber provides a higher yield strength at a much lower weight. [7]
- To evaluate the stability of the drone we have taken the CAD design of the arm and the center separately. The Static analysis has been done on the key structural parts of the drone. The analysis is done separately on fusion 360 for both.



- The results are extremely promising as the internal stresses and the displacement is well within safety factors. We have added a live simulation recording as well. The full report is uploaded on the Drive.

- To perform the stability analysis, we have calculated the stress, strain and the displacement of the arm and the central body of the frame when thrust and/or weight is added to it. The output result is promising, and the displacement is minimal with the thrust at maximum.

- For the static analysis of a single arm of the Tarot 650 quadcopter, we have kept one end of the arm fixed with zero moment and the other end with the motor (with have a maximum thrust of 1.5 kg ~ 1.75 kg) to have an upward load of at the very least 15.0 N. The arm deflects in the upward direction with stress occurring at the starting position of the arm. The design load is taken to be 30N for the simulation.

- For the analysis of the frame we have taken the payload to have a weight of maximum 20N and additional drive weight which includes the battery to be 30N as well. The arm holders are kept at a static position and the result dictates that the central frame would deform well under safety factors. The design load is taken as 100N. However further tests need to be conducted with a physical model to improve and confirm the results. [5]
- Reports are available here.

CAD File with Payload & Drone Components

- The CAD file has been designed to resemble the actual build as closely as possible. The primary design which includes the payload and the internal components has been utilized to create designs for static analysis and simulations. Each individual file is upload on Drive.
- The designs for the static analysis are the skeleton of the drone that is, its frame (arm and center of the frame).
- The design used for the Gazebo simulation only differ in internal components and file type. The design had to be optimized for smoother operation.
- The payload has been attached on the bottom of the drone inside a box with fixed dimensions. The casing for the battery and the center frame are kept as separate entities but it is subject to change depending on the physical model.
- Links for - https://drive.google.com/drive/folders/1oktp8P8gfRQQ1KgVw75gvh7lcxKnPhgI?usp=sharing

1.) Complete Model with internal parts
2.) All Components
3.) Arm Assembly
4.) Frame (includes arm and center plate)
5.) Center frame

Autonomous Flight Algorithm Details

For all three subtopics we will be using odometry data provided by monocular vision. Using that, we will provide set point to MAVROS, MAVROS will it convert to hardware actuation values using MAVLINK protocol

We compare setpoint and current position at 20Hz, and next best set point is queued meanwhile drone reaches to current define d setpoint. Next setpoint is not pushed as current setpoint until drone do not reach within sphere of radius 0.05m (which is a threshold set by us). Once the drone is in that radius next set point which was queued till now will be pushed as current setpoint and now drone will target that setpoint

I) Take Off and Landing Logic

- For taking off we command drone to change mode(i.e., from manual to offboard, using MAVROS service). After that we wait for 5 s to receive "success" message which means that drone has successfully switched to offboard mode. If it's not the case then we again wait for 5s to receive a success message and this continues.If offboard switch is successful, drone gets armed and takes off to first setpoint i.e., (0,0,3)(fixed).

- Drone keeps detecting gates from start itself. Drone keeps detecting nearest position of gate while its reaching to take off height and keep publishing(which is queued) the next set point for drone(i.e., nearest gate). After reaching to minimum threshold distance of set waypoint, drone gets new setpoint and navigate to that position.

- For landing once we reach final position which means drone is not able to detect no more gates drone moves forward for some d istance and offboard mode is switched off and then we use MAVROS service to land it properly

II) Flight Algorithm

We fuse data provided from camera and IMU to provide odometry data i.e., to calculate position(x,y,z + yaw, pitch, roll angels) and roll, pitch and yaw rates.

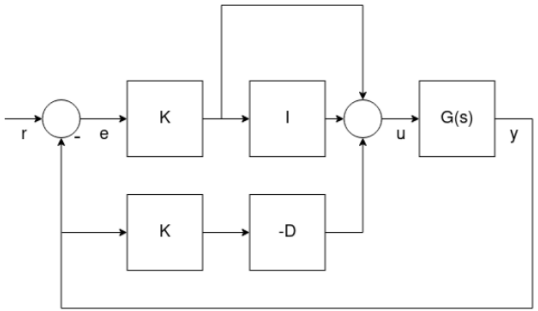
1.) Odometry Estimation on companion side - We will be using *OpenVINS[1]* for pose estimation, this library is two-time winner of IROS – VIO [4] competition which determined pose estimate using VIO + inertial measurement unit for aggressively moving racing whose data was provided using EUROC dataset with 7% translational error. It fits perfectly in our case as motion will not be that aggressive and lighting condition is also proper so we can use this for pose much accurate pose estimation using a simple monocular camera

2.) For px4 Firmware side odometry estimation -
Following things will be extracted from VIO
a.) Velocity in (x,y,z)
b.) Rotation Values (pitch, roll, yaw)
c.) Both Barometer or Vision can be used (according to performance, in simulation we used vision)
d.) Angular rates = GPS + IMU (GPS from VIO only as GPS will be virtually faked through GPS + additional remaining angular rates from default sensor on hardware)
d.) Parameter values specific to PX4, EKF2_AID_MASK = 321, EKF2_HGT_MODE = 3 (for Vision, barometer no change)

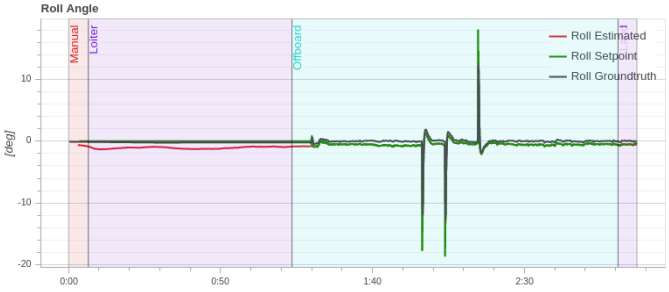
As we are not using PX4 supported tracking cameras we have modified PX4 pipeline to use our vision data(OpenVINS) with help of ROS. Refer to for detailed information

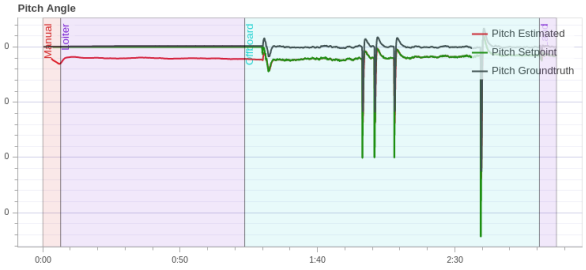
3.) (Only for using GCS features, won't affect offboard flight in any way) QGroundControl features will be disabled automatically if we don't use GPS data so we simulate GPS data with MAVROS fake-GPS plugin and create data for GPS using VIO itself. Basically we manually input latitude and longitude data to PX4, so it thinks that its getting actual GPS data.
This part is not possible in simulation as gazebo is already providing fake GPS data to us

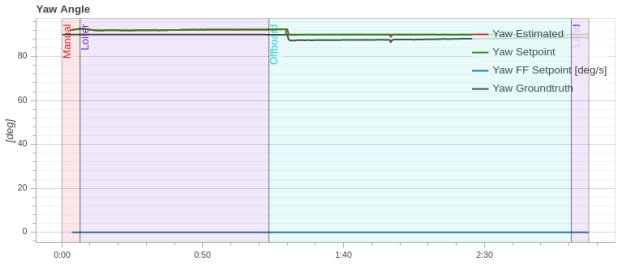
4.) Control Architecture and Tuning Constants of our model
PID architecture shown will be used by PX4 for controlling angular rates and angles, which leads to full control of drone except in Z direction. For that thrust input is directly controlled with PWM signal mapping. Motor mixing I.e., mapping all input to thrust values is done by defining mixer files (in our case we are using default mixer). Please refer [16]

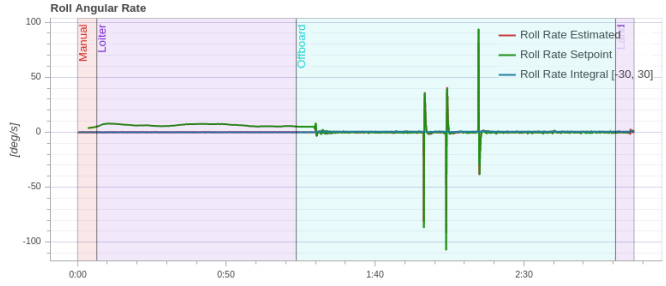


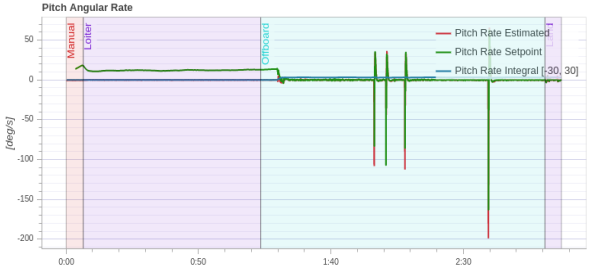
5.) Default PID constants worked great for us, below are some of the graphs explaining stability and tuning in our case, this all data has been generated using PID-analyzer for PX4, yaw angular rate needs some decrease in integral part due to sustained oscillations. We will prefer to tune constants on real drone as its difficult to tune them with PX4 SITL alone as changes in parameter are not clearly visible and then affecting some other parameter











Our complete Flight Review analysis along with PID analysis is uploaded on portal https://review.px4.io/plot_app?log=8c93f091-594b-4fbe-bc9e-e2d2489d466d . 3D analysis of our flight (by default IRIS will show up, that is the default for tool)can also be done on the same portal. It will show frame as 10031 which is does not exist in PX4 and is our AeroBITS model. To properly infer these graphs use [6] or [16]

Trip Management

Please refer to gate detection algorithm before reading this

1.) Firmware based (when drone is not in obstacle avoidance)

There are two main options available for trajectory management when actuation is provided. Jerk-Limited type Trajectory and Slew rate type trajectory. Jerk Trajectory generator limits jerk and acceleration and provides a smooth s –shaped position tracking with time. It is more suitable with vehicles which do not require quick position response and hence aligns properly with our study[14]

2.) When drone is in obstacle avoidance

In this trajectories will be generated by Fast Planner [14] which will input set of setpoints to px4 rather than following de fault trajectory generation. It uses kinedynamic path planning and use B-spline optimisation to smoothen the trajectory

Frame Detection Algorithm Details

We have so far tried multiple approaches towards gate detection. The preliminary approaches that we initially created, using basic Computer Vision programming is what is showcased in our demo video. We have since shifted to better methods that rely on basic Recurrent Neural Network (LSTM) systems, and R-CNN based systems that also have elements of the Snake Gate Detection algorithm embedded into it. The primary problem we were hitting upon while using the initial naïve approaches was that the gate was often incorrectly detected as a polygon with greater than 4 sides, due to jagged edges created as noise due to camera resolutions and changing angles of observation. Rather than relying on blurring, and correcting for errors, it was realized that applying the snake gate algorithm is significantly more efficient, as the algorithm (figuratively speaking) finds one point on the gate, and then "draws" a path across the screen along the gate, thereby locating the corners of the gate with extremely good accuracy, without too much computational complexity. This also helped us reduce the number of total polygon detections on screen, as the initial naïve approach would often detect random polygons on screen that weren't the gates, and these had to be ignored inside the algorithm based on Kalman filtering applied on the gates to predict the motion of the geometric centroid of the gates, and smooth out the noise. Now however, the system is being made much more robust, as the snake gate detection system combination no longer detects such false positives.

This is highly beneficial in terms of pose estimation, and since we can calculate the effective visual angle occupied by each gate, we have made the reasonable assumption that based on the position and pose of the rectangle open frame of the gate, one can reasonably estimate what the closest gate is, using an estimate of the visual field occupied by the solid angle covered by the four corners of the gate. The pose estimation will also have to be factored in here, as the visual angle occupied reduces when the gate is viewed side-on. Here, it is beneficial to work based on our current understanding of the actual dimensions of the gate, which enables the logic to differentiate between gates much faster. In the absence of said data, the algorithm will still work, but will require a slight amount of additional logical computation.

Frame Detection Algorithm Details (Continued)

The current idea for the logic system is as follows:

1. Image is detected using the camera setup

2. The image is masked for the colour range of the gates, and is then passed into the R-CNN and snake algorithm layer of the code, which spits out corner positions of the gates

3. A smoothing Kalman filter is applied, which prevent the motion of the drone from causing the algorithm from getting confused, the gates are tracked using the predictions of the filter, and de-noising also occurs. This way, even if there is a discontinuity in the gates, the snake gate algorithm's corrections, along with the filtering help us avoid anomalies.

4. Once the right gate and ordering is calculated, and pose estimates for the current gate and next gate are calculated, the algorithm passes this information onto the controller logic, which gives inputs to the MAVROS enabled system to translate the drone to a new location, and so and so forth, traversing the gates.

5. There will be visual discontinuities when the drone gets near a gate, as the gate is partially obscured away from the drone's field of view. A probability based estimate of its position will be made. When the drone gets really close to the gate, the default collision avoidance in the drone is disabled, and the drone control logic changes mode to "Gate traversal" mode. This is a very precise mode we have coded, where the translations are kept track of, along with accelerometer readings. Even though the safety of the collision avoidance is gone, since we know the exact position of the gate, and the drone, we can very carefully make the drone pass through the gate, after which collision avoidance is re-enabled.

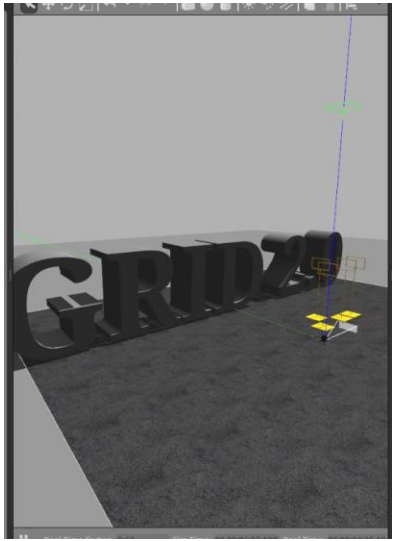
Gazebo Simulation & Assumptions

1. Various component(like esc,pixhawk module,battery etc) were not considered in gazebo simulation as it was leading to a file of 100Mb which was hard to simulate in any of available computers. As a result we approximated the weight of the components and used that as net weight of our drone
2. Lighting condition are same as that are provided by sun in gazebo
3. Due to heavy computational requirement for simulation + path planning we were not able to do that for more than 4 gates.
4. Default PID constants(Appendix B) worked with our simulation model, we cant tell for sure that they would work in real too
5. Our selected frame has retractable gears but we didn't put that control in simulation for computational ease
6. We are not simulating pose estimation from actual monocular camera so it might vary in real
7. Depth data used for simulation was taken with help of kinect camera plugin might not be accurate
8. Currently the threshold for setting next waypoint after current waypoint in navigation of drone is kept 0.05.
9. We used gazebo noise plugin to create noise in images received which adds to exactness of our solution
10. Blade flapping is ignored in all gazebo simulation

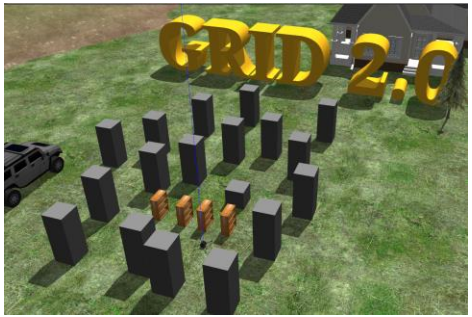
Relevant files for simulation

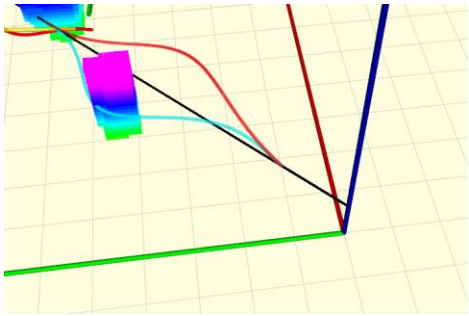
1.) All simulation videos in https://drive.google.com/drive/folders/1pO3sTPlFtnVwXywXCbGArTk9sI5-Rfiy?usp=sharing

Gate crossing



Obstacle avoidance





Other VIDEOS present in drive

a.) Local Planner(we integrated it with our model but can't get feed)
b.) Global Planner
c.) Using Qground Control with camera feed
d.) VIO estimate

Execution plan with timelines

Assuming we start on 20th August,2020

1.) Parts Collection – 20th - 31st August. We already have many parts in our aerodynamic lab at Pilani. If we can get access to it, we can when we reach there. Otherwise it might take around at least 5 – 10 days for all parts to reach and us start working with actual parts.

2.) Assembling and basic hover test– 31st - 2nd September We can have a basic flight test quickly, if all components are ready.

5.) Including obstacle avoidance in current PX4 pipeline successfully using fast planner and ROS – 20th – 5th September

3.) Testing 1 - Load testing, Disturbance test [6] and tuning constants for flight with manual controller (PlayStation JoyStick) – 2nd - 6th September

4.) Testing 2 - VIO stabilization in proper lighting conditions with data obtained from Camera Testing and IMU data from Pixhawk hardware, basic setpoints will be given to drone and tested if it's able to follow those – 2nd - 8th September

5.) Testing 3 – HITL (Hardware in the loop testing) We will test how algorithms perform in on real hardware(excluding camera) by setting up a connection. So real hardware will get feed from simulated camera and will control actuation of virtual drone

5.) Testing 3 - Gate detection algorithm without obstacle prevention testing- Gate will be loosely fixed hold by someone. If anything goes wrong could be removed easily. Re tuning algorithm constants (for both px4 firmware and gate detection algorithm) to get exact pose of gate in real world – 8-15th September

7) Testing 4 - Standalone obstacle avoidance in 2D (at fixed height) - with fast planner. -15th September –17th September

8.) Testing 5 - Final algorithm (Obstacle avoidance + Gate Detection) - 17th – 25th September

Component Details

NAME(QUANTITY)	DESCRIPTION
Tarot 650 Iron Man Frame(1)	Carbon fibre frame with 600mm diameter and 170mm ground clearance, the total frame weight is not more than 750g.
Camera(1)	Microsoft Xbox Kinect Camera
Brushless Motors(4)	4114/320 kv, weight – 148g 22 poles motor and has a 320 rpm/V of speed.
Set of Propellers(2)	15-inch,15 X 55Carbonfibrepropeller blade
30A 2S-6S BlHeli Platinum ESC (4)	7.4-22V,2S-6S lipo,30 A OPTO
Turnigy Heavy Duty Series 4000mAh 6S 60C – 120C (1)	4000mAh 6S 60C – 120C RCC battery
ESP 8266MODULE(FOR WIFI)(1)	Weight 10g:Size 10 x 5 x 5cm
HolyBro Pix32 Pixhawk PX4 2.4.6	Size: 3.8 X 5 X 1.5cm,Weight: 38 gm flight controller
Raspberry pi	RAM: 2GB,USB 3.0 port,2.4-5 GH Lan, Quadcore 64-bit Broadcom,2711 Cortex A72 processor

Component Sources, Price & Total Cost

- None of the product is discontinued and are currently available at time of writing, price and availability status are subject to change.

NAME(QUANTITIY)	SOURCE	PRICE(RS)
FRAME(1)	RC Product.in	8580
CAMERA(1)	Flipkart	4400
BRUSHLESSMOTORS(4)	LOCAL STORE(link for description of motor)	8000(4)
PROPELLER SET(2)	Robokits India	1960
30A 2S-6S BlHeli Platinum ESC	Amazon	3200(4)
Turnigy Heavy Duty Series 4000mAh 6S 60C - 120C	Amazon	7352(2)
ESP 8266MODULE(FOR WIFI)(1)	ESP8266 Module (For	259
Holybro Pix32 Pixhawk PX4 2.4.6	Amazon	8,671.74
Raspberry pi	Electronics Comp	2685
	Total	45107.74

Current Progress

Completed -

1.) We have now become well familiar with PX4 Firmware architecture and have got connected to its helpful community. We are at a sdtage so that we can modify px4 architecture as and when required.

2.) Frame selected after comprehensive structural and aerodynamic stability testing including detailed calculation of flight time and efficiency(with errors considered). Results are shown before.

3.) Complete simulation of real drone in CAD format and also in gazebo has been done. We are able to fly drone properly at some fixed height and make it cross fixed setpoints.[7]

4.) PID constant analysis along with attitude hold, thrust data all has been provided precisely for simulation. [7]

5.) We can successfully detect gate and get coordinates of its center of gate with the help of simulated monocular camera with noise included. Then we direct drone to reach that position and meanwhile compute position of next gate. We tested our algorithm by autonomously crossing 4 gates

6.) Obstacle avoidance integrated with our model and can successfully calculate trajectory to avoid obstacle in between

Still need to work on –

1.) Obstacle avoidance still don't work precisely. Main issue is that it consider gates as obstacle sometimes

2.) VIO data in simulation was enabled directly with help of gazebo plugin, so it might be more accurate than OpenVins implemented on real camera. That is something which can be tested after we have hardware and real lighting conditions.

References

[1] Patrick Geneva, et al. "OpenVINS: A Research Platform for Visual-Inertial Estimation." *Proc. of the IEEE International Conference on Robotics and Automation*.

[2] "EUROC." https://projects.asl.ethz.ch/datasets/doku.php?id=kmavvisualinertialdatasets.

[3] Zhou, Boyu et al. "Robust and efficient quadrotor trajectory generation for fast autonomous flight". IEEE Robotics and Automation Letters 4. 4(2019): 3529–3536.

[4] UZH IROS 2019 Winner and there their papers. https://github.com/uzh-rpg/IROS2019-FPV-VIO-Competition.

[5] Kuantama, E., Craciun, D., & Tarca, R. (2016). QUADCOPTER BODY FRAME MODEL AND ANALYSIS*ANNALS OF THE ORADEA UNIVERSITY. Fascicle of Management and Technological Engineering., Volume XXV (XV), 2016/1*.

[6] Quadcopter Design for Payload Delivery. https://www.scirp.org/journal/paperinformation.aspx?paperid=69618 .

[7] Pounds, P., Bersak, D., & Dollar, A. (2012). Stability of small-scale UAV helicopters and quadrotors with added payload mass under PID control*Autonomous Robots, 33*

[8]*Guidline-Multi-Rotor, Rotors,* https://www.rotordronepro.com/guide-multirotor-motors/#visitor_pref_pop

[9] cjdavies.org Drone Build, https://cjdavies.org/blog/?p=4502

[10] Yilmaz, E., & Hu, J. (2018). CFD Study of Quadcopter Aerodynamics at Static Thrust Conditions

[11] Propeller Thrust Equation, & Downloadable Excel Spreadsheet Thrust Calculator, https://www.electricrcaircraftguy.com/2013/09/propeller-static-dynamic-thrust-equation.html

[12] Multicopter PID Tuning Guide https://docs.px4.io/master/en/config_mc/pid_tuning_guide_multicopter.html

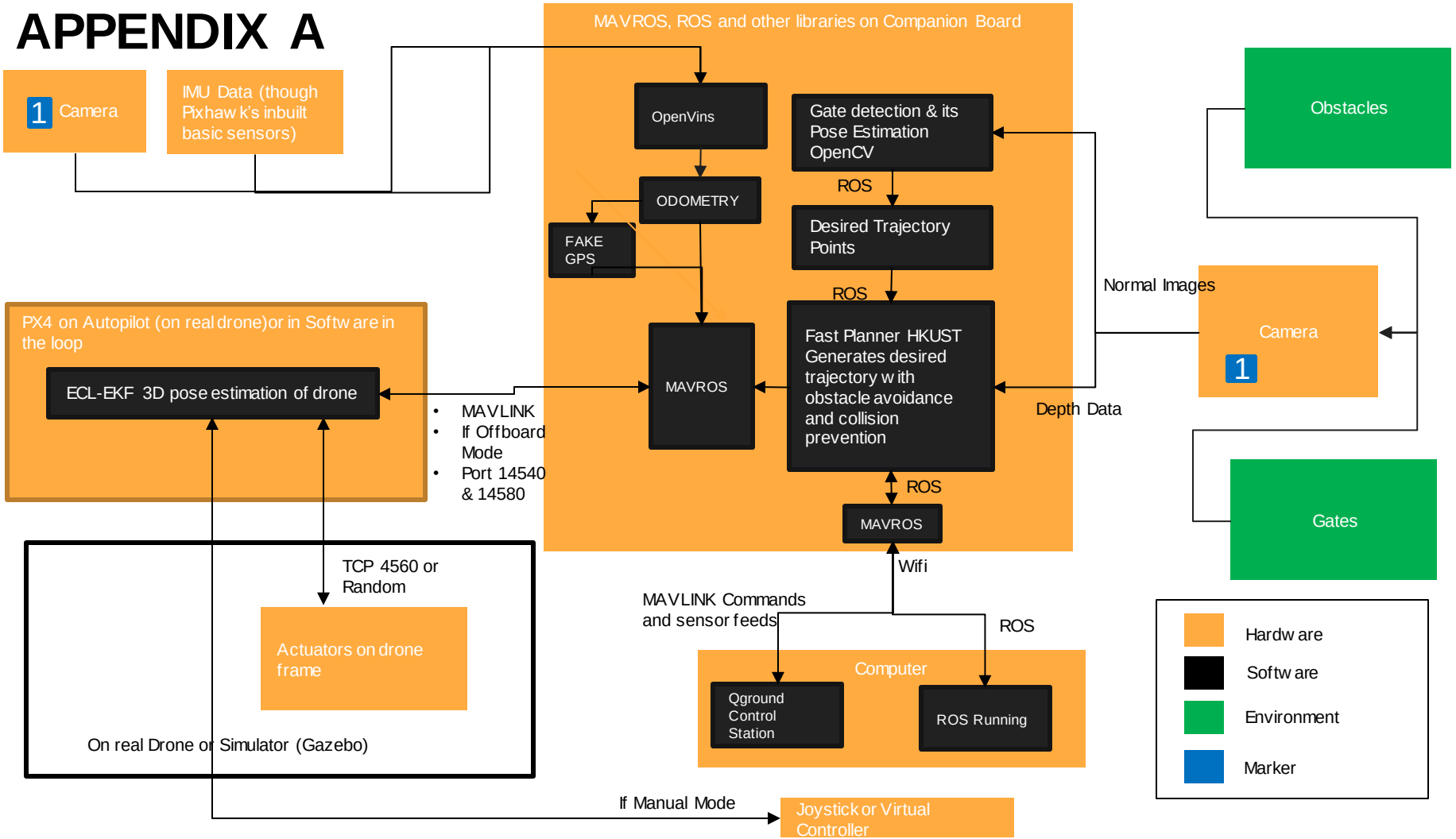
[13] Using Vision or Motion Capture Systems for Position Estimation https://dev.px4.io/master/en/ros/external_position_estimation.html

[14] Multicopter Setpoint Tuning (Trajectory Generator) https://docs.px4.io/master/en/config_mc/mc_trajectory_tuning.html

[15] Fast Planner, https://github.com/HKUST-Aerial-Robotics/Fast-Planner

[16] Using ECL-EKF, https://docs.px4.io/v1.9.0/en/advanced_config/tuning_the_ecl_ekf.html

APPENDIX A


MAVROS, ROS and other libraries on Companion Board
1 Camera
IMU Data (though Pixhawk's inbuilt basic sensors)
OpenVins
ODOMETRY
FAKE GPS
Gate detection & its Pose Estimation OpenCV
ROS
Desired Trajectory Points
ROS
Fast Planner HKUST Generates desired trajectory with obstacle avoidance and collision prevention
MAVROS
ROS
MAVROS
Obstacles
Normal Images
Camera
1
Depth Data
Gates
PX4 on Autopilot (on real drone) or in Software in the loop
ECL-EKF 3D pose estimation of drone
MAVLINK
If Offboard Mode
Port 14540 & 14580
TCP 4560 or Random
Actuators on drone frame
On real Drone or Simulator (Gazebo)
Wifi
MAVLINK Commands and sensor feeds
ROS
Computer
Qground Control Station
ROS Running
If Manual Mode
Joystick or Virtual Controller
Hardware
Software
Environment
Marker

Appendix B PID Constants

These PID constants work great for simulated drone
These can be directly copied to terminal one by one when firmware launch or by QGroundControl or creating a file in initiation folder of PX4. We prefer last one

```
param set MC_ROLL_P 7
param set MC_ROLLRATE_P 0.15
param set MC_ROLLRATE_I 0.05
param set MC_ROLLRATE_D 0.004
param set MC_PITCH_P 7
param set MC_PITCHRATE_P 0.15
param set MC_PITCHRATE_I 0.05
param set MC_PITCHRATE_D 0.004
param set MC_YAW_P 2.5
param set MC_YAWRATE_P 0.25
param set MC_YAWRATE_I 0.25
param set MC_YAWRATE_D 0
```


Flipkart
GRID 2.0