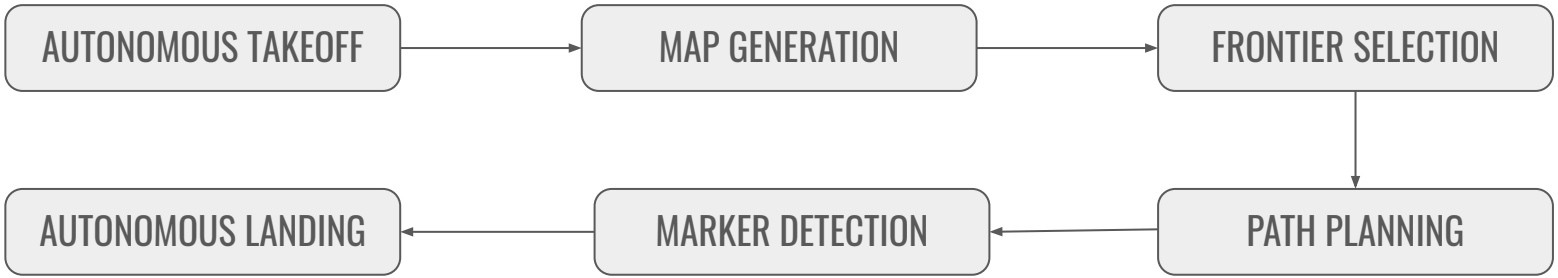
D.R.D.O DGRE's
VISION BASED OBSTACLE AVOIDANCE DRONE

TEAM : 16

INTER IIT TECH MEET'21

Approach



AUTONOMOUS TAKEOFF AND LANDING

Distance based logic for accurate autonomous landing

MAP GENERATION

Voxblox for map generation

Dynamically sized map that makes use of voxel hashing

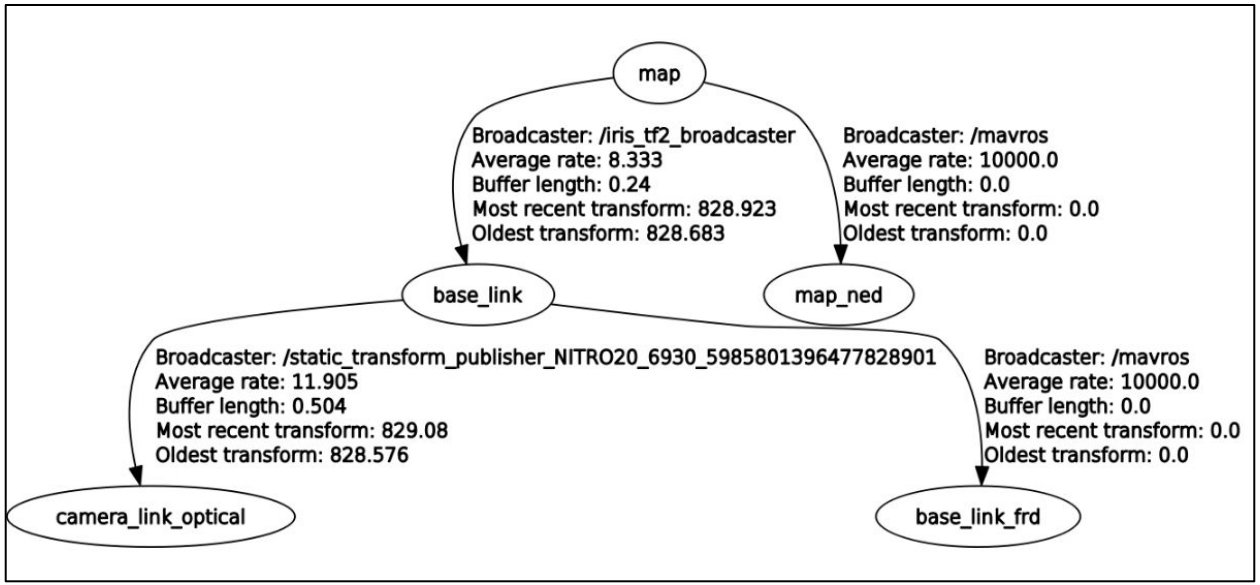
Approach (contd.)

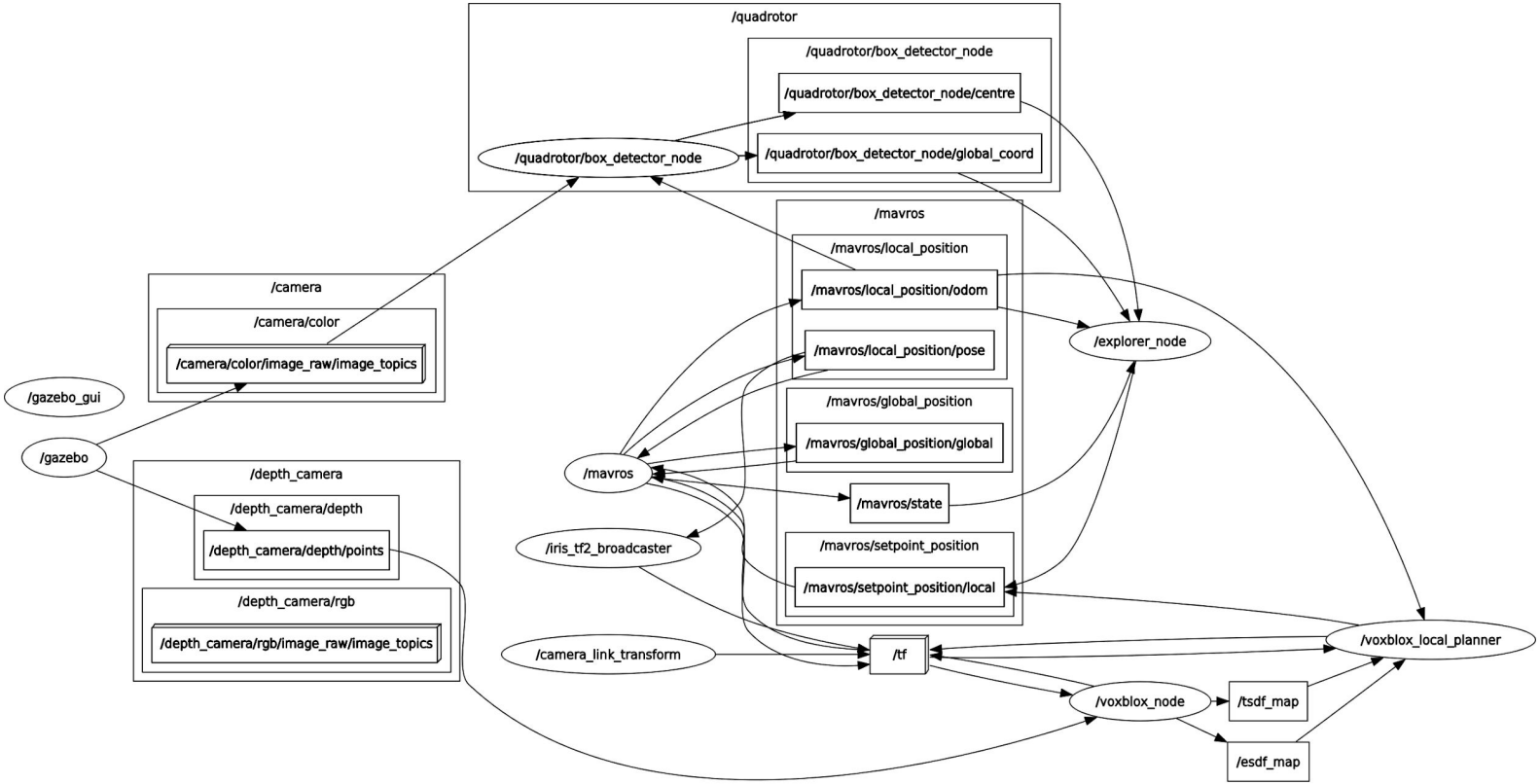
FRONTIER SELECTION	Select the best frontier possible using a distance score Frontier farthest in the direction of UAV selected is first
PATH FINDING	Path is calculated between two points by generating a random graph and running A* algorithm on it
MARKER DETECTION	ArUco module of OpenCV library for detection of ArUco marker Frame transformations to get the marker coordinates in map(global) frame

Fixes In TF Tree

Static_transform_publisher between the base_link and camera_link_optical with a transformation of (0 0 0 -1.57 0 -1.57)

TF static broadcaster which uses the mavros odometry and publishes frame transformations between map and base_link via TF





Software Architecture: Graph

Software Architecture: Nodes

voxblox_node

Receives point cloud data from the depth camera

Handles generation of ESDF, TSDF map for the planner

explorer

Integrates the takeoff, landing, detection and planning modules

Publishes final setpoints for landing of the quadrotor

box_detector_node

Uses the downward facing camera to detect the ArUco marker and calculate its position provided it has the id 0 using OpenCV

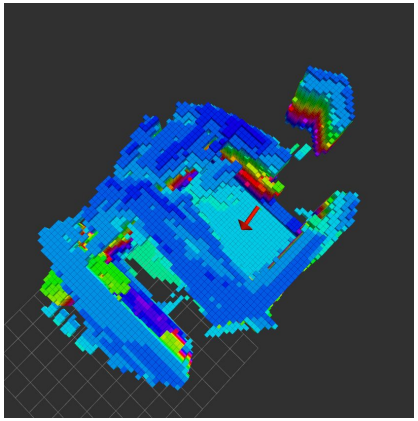
voxblox_local_planner

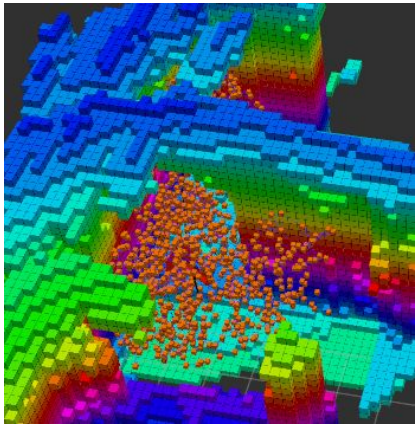
Receives the ESDF/TSDF maps generated by the Voxblox server

Publishes the selected frontiers as waypoints

Voxblox Local Planner

- Runs the Map Generation, Frontier Selection and Path Finding Algorithms
- Map Generation : Generate ESDF and TSDF maps. Sensor data is used to create TSDF map which is then propagated to create ESDF map
- Classify Voxels as FREE, OCCUPIED OR UNKNOWN
- Frontier : Voxel that has more than one unknown neighbour





- Frontiers are then clustered together and scores are assigned to the frontier centres
- The best frontier based on a yaw and distance metric is then selected
- Finally, A* algorithm is used to plan a path to the frontier

Path Finding

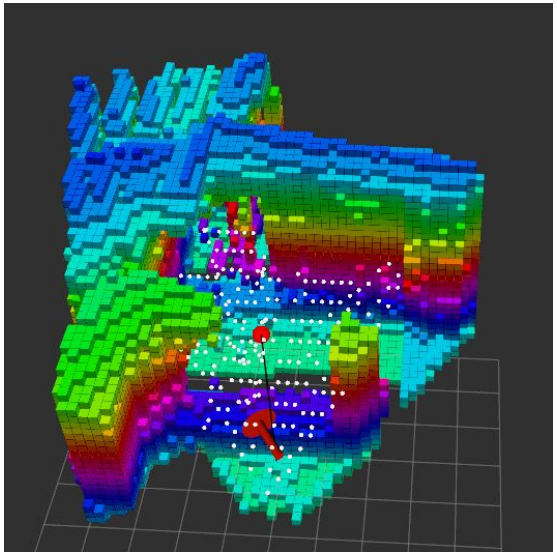
Generate Random Graph by sampling points within box defined by start & end points

Connect Graph by using Rtree.

Run A* on the graph

Shorten the graph by checking each pair of points for a free path in between

The shortened path is published to MAVROS and tracked in planner node



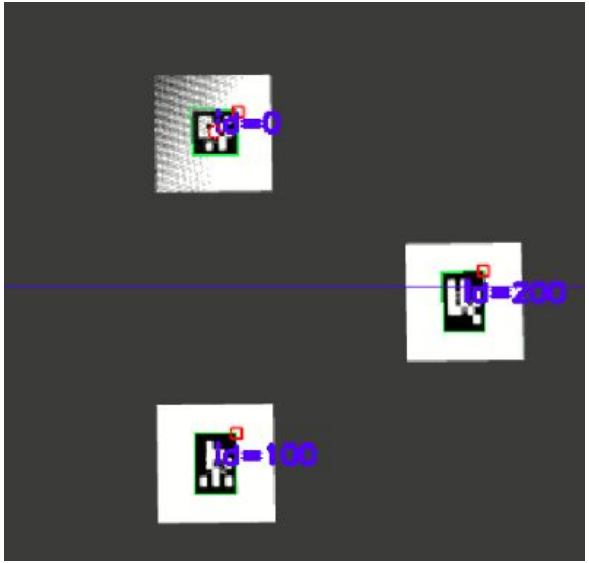
On failure, second attempt is made by increasing the sampling density and size of sampling box

If it fails again, last attempt is made by trying to find a path to the point nearest to the endpoint

If everything mentioned earlier fails, then the waypoint is marked infeasible and no path is returned

Aruco Detection & Landing

- cv::aruco::detectMarkers() function used to get the markerIds, and their corresponding marker_corners
- Got camera frame coordinates of the marker by using functions of this library
- Used the cv::aruco::estimatePoseSingleMarkers() function and performed frame transformations global coordinates of the required aruco marker are obtained
- For accurate landing of quadrotor, used distance based thresholding in x-y plane, on the current odometry of the quadrotor



Centre Detection → Camera frame Coordinates → Quad frame Coordinates → Global frame coordinates → Quadcopter

Previously tried approaches

- NBVP :- Getting stuck in local minima
- AEPlanner :- Issues with exploration in closed places
- Fast Planner :- Assumes world map known, a priori

Scope of improvement

- Improvements in frontier quality
- Addition of a safety node, updating the map at a faster rate and checking for collision
- Path smoothing using trajectory generation

Performance Analysis

Computation	**Cost**
Gazebo	~1.5 cores
RVIZ(Visualization)	~ 0.5 cores
Voxblox + Planner	~1.5 cores

Local System Specifications	
OS	Ubuntu 18.04
Processor	Intel® Core™ i7-10750H CPU @ 2.60GHz × 12
Graphics	GeForce GTX 1660 Ti/PCIe/SSE2

THANK YOU