DroneSLAM : Aerial Assistance for Sensorless Vehicles

Soham Patkar & Abhilash Balachandran

530.707 Robot Systems Programming - Project Report

Date : May 18th 2017

1. Introduction:

Vehicle planning in uncertainty is a trending topic. The need for planning and navigation in unreliable environments has greatly increased with the development of artificial intelligence algorithms and computing machinery.

One of the most challenging tasks in motion planning is the awareness of the existing surrounding. The absence of a 6 dimensional degree of freedom makes navigation with ground robots harder. Also, in scenarios where the navigating subject is not a robot, but rather a person driving a vehicle, the absence of depth sensors makes navigation and mapping highly challenging. Also trending is the need for different robotic systems to communicate with each other, sharing information to a mutual benefit.

Our project presents a simple solution of navigation for ground vehicles in uncertain terrain.

2. Proposed Goals:

In this project we seek to map and localize the 3d environment with the help of an aerial vehicle. The map is then utilized by a crude robot with just odometry information to navigate from one point to another whilst avoiding obstacles.

For building the map, we first find visual features in the map and then find their 3D pose in the environment. We assume we have a computer vision algorithm that does this for us, which is a fair assumption since there are several algorithms that do this for us. We use AR Tags as our 'visual features' to make our life easier. Further, we find all possible features in the environment, and their relative pose with respect to all other features. Eventually through some extrapolation, we generate a 3D reconstruction of the environment. The vehicle does not require a full 3D reconstruction, but the drone does need it to avoid collisions.

For the ground robot, all we need is information about the ground terrain and any areas it needs to avoid. We do this by providing an occupancy grid of the ground plane using the map developed by the drone. Once we have the occupancy grid, we can use motion planning in existing packages to reach a desired location while avoiding obstacles. We used the move-base library in the ros-navigation stack for our project.

3. Software:

3.1. Packages created:

a) map_building: We developed a package named "map_building" for building our map. The node is called 'make_wall' and it involves one callback function which is called everytime an AR tag is detected. This callback by itself call several functions which are actually responsible for the visualization.

The map_building function gets the pose data from the AR Tag and extrapolates it as much as possible. It assumes that every AR Tag represents a plane and it maintains a list of planes that it has seen. Every time the function is called it tries to find all possible intersection points and returns which ones are available and on what plane they are present.

The make_marker function is responsible for actually creating the map is RVIZ. It takes the plane equation and intersection points from the above function and draws the corresponding plane in RVIZ.

The updateOccupancyGrid function is responsible for creating the occupancy grid of the ground plane. It excecutes every time an AR Tag is detected on the ground plane, and also if an intersection point of the ground plane with any other plane is found. Eventually, it results in an occupancy grid where AR Tags on the bottom

plane are treated as obstacles and the polygon created by intersection points of the bottom plane as the boundary of the occupancy grid.

b) map_buidling_gazebo: Package with same functionalities for gazebo simulations. The Ardone differs significantly from the simulated drones. Hence, there are several variations between the 2 packages. However, the underlying logic remains the same.

b) ardrone_joy: To teleoperate the Ardrone using a joystick.

Link for git repository

3.2. Existing packages:

a. ar_track alvar: AR Tag tracking package for ros [http://wiki.ros.org/ar_track_alvar]

b. ardrone_autonomy: Ros driver for ar drone [https://ardrone-autonomy.readthedocs.io/en/latest/]

c. Gazebo – Simulation software with realistic physics engine [http://gazebosim.org]

d. Eigen – C++ Math Package, helps in easy matrix operations [http://gazebosim.org]

e. ros_navigation Stack – Move Base, amcl and fake Localization [http://wiki.ros.org/navigation]

f. EduMip software – Developed by Strawson, ROSified by Dr.Louis Whitcomb [https://dscl.lcsr.jhu.edu/ME530707_2017_EduMIP_ROS].

4. Hardware and Infrastructure :

4.1. Parrot AR Drone 2.0:

The Parrot AR Drone is developed by Parrot, a popular drone manufacturer and is one the most commonly used drones for hobbyist purposes like photography and videography. It has one front-facing camera and one bottom-facing camera. It has an internal IMU and a Sonar sensor for depth estimation. The drone uses the optical flow from the bottom facing camera along with IMU data for odometry and sonar sensor to estimate height.

Fig 1. Parrot AR Drone (a) Bottom View (b) Front View

4.2. EduMip:

EduMip is a educational robotics platform developed by Strawson Robotics. It has a Beaglebone Black microprocessor with a Robotics Cape shield. The Robotics Cape has an inbuilt IMU and can interface 8 motors and several sensors. A ROS package for self-balancing was created for the same by Prof Louis Whitcomb for the purpose of this course.

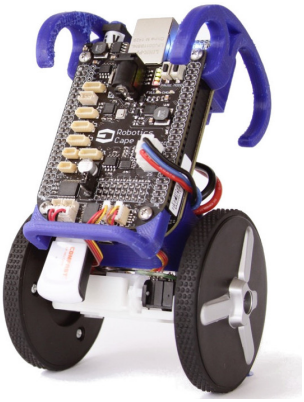


Fig. 2 EduMip

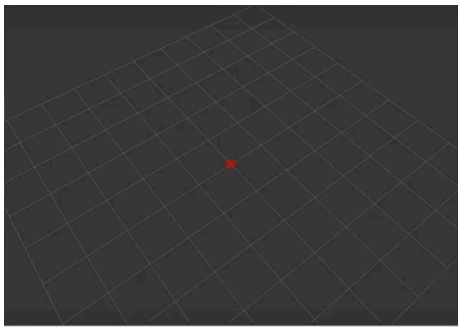
4.3. The testing setup:

To test this project we used a netted cage instead of a room with walls so as to avoid damage to the drone in case of crash. We assumed that this cage represents a room with the same dimensions. We put AR Tags in various locations to use as visual landmarks. We used one AR Tag on each of the walls to reduce the runtime of our program but it works the same for any number AR Tags placed anywhere.

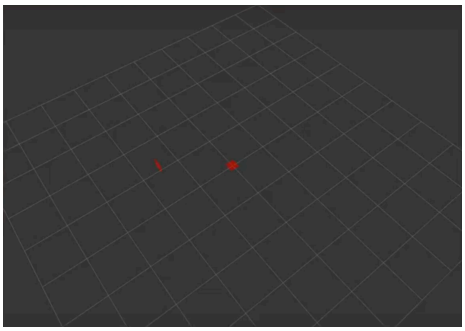
Fig 3. The testing setup.

Results

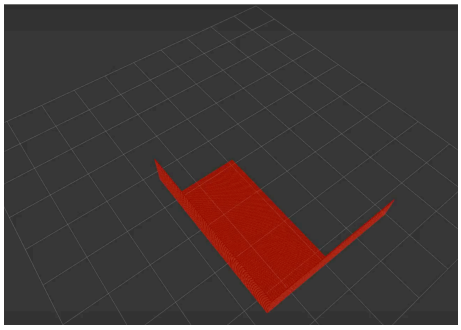
We were able to build geometric representations of the environment namely the cage that the drone is flying. Fig. 4 shows the building of the map



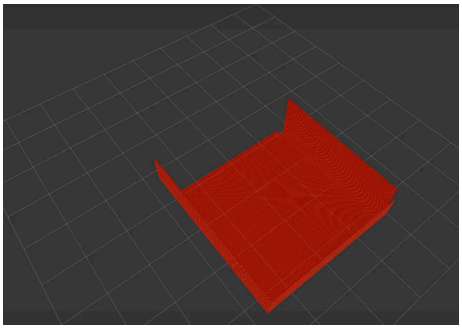
(a.) When one ar tag is detected



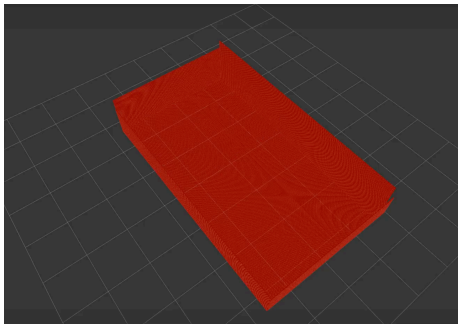
(b.) When two ar tags are detected



(c.) When three ar tags are detected



(d.) When four ar tags are detected



(e.) When five ar tags are detected

Fig. 4 Building of the map via ar tag detection v

As you can see, when the camera detects a single ar tag, we do not know its extent in space. So we draw a single polygon with the size of the ar tag. When we see three ar tags, we have a point of intersection, and can now draw planes. We keep drawing planes representing the cage with each ar tag we detect. After that, we map the obstacles on the bottom which are represented by ar tags.

The occupancy grid is shown in Fig.5. The white region is obstacle free and black region is the region occupied by obstacles.

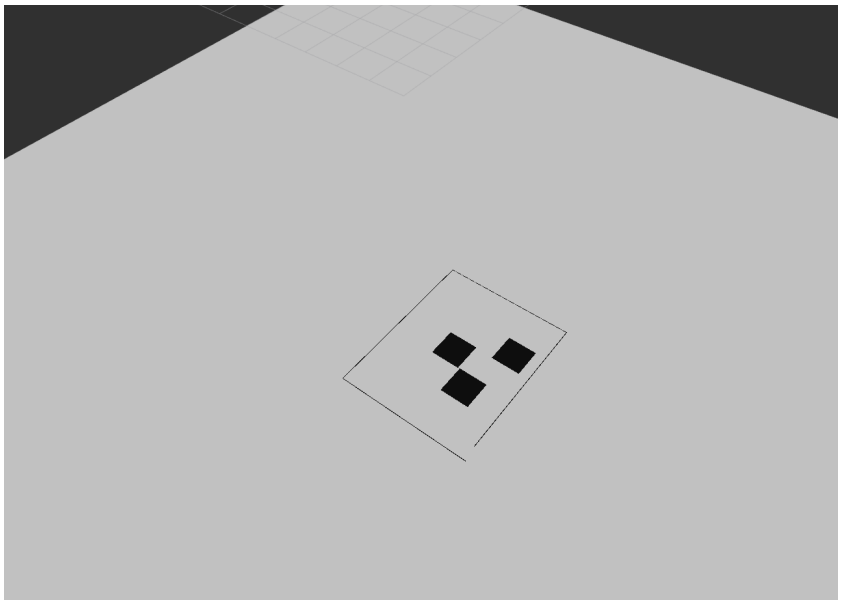


Fig 5. Occupancy grid. The black regions symbolize the obstacles. The Boundary visible is the boundary of the cage.

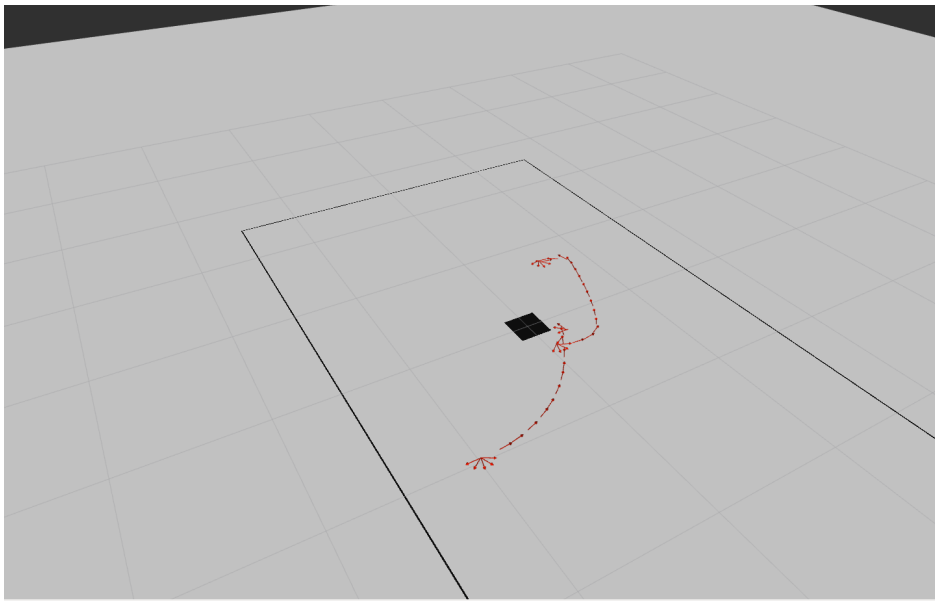


Fig. 6 shows the path planned by the EduMip in the occupancy grid using move base package. We used fake localization package to localize the EduMip and to plan using move-base.

Video Link

https://www.youtube.com/watch?v=YvCV_tB-hj0

Lessons Learned

- We learnt that **testing on real hardware is not the same as simulation**.
- Being aware of time stamps can make or break your program.
- Using plugins to fake sensor data can be a smart hack.
- Be careful when using AR Drone. We learnt not to not rejoice if it works in simulation. Gazebo is highly idealistic.
- Parrot AR Drone 2.0 is not a developer's drone. It has very less modularity and functionality. Do not use both its cameras together in the ar drone, you will get into trouble.

Suggestions for future projects

Be careful using AR drone. Parrot AR Drone 2.0 is not a developer's drone. It has very less modularity and functionality. Do not use both its cameras together in the AR drone, you will get into trouble. I would also suggest not to bank on simulations a lot.

References

[1] Engel, Jakob, Jürgen Sturm, and Daniel Cremers. "Scale-aware navigation of a low-cost quadrocopter with
a monocular camera." Robotics and Autonomous Systems 62.11 (2014): 1646-1656.

[2] Berat A. Erol, Satish Vaishnav, Joaquin D. Labrado, Patrick Benavidez, Mo Jamshidi, "Cloud-based control
and vSLAMthrough cooperative Mapping and Localization", World Automation Congress (WAC) 2016,

[3] Engel, Jakob, Jürgen Sturm, and Daniel Cremers. "Accurate figure flying with a quadrocopter using
onboard visual and inertial sensing." Imu 320 (2012): 240.

[4] Engel, Jakob, Jürgen Sturm, and Daniel Cremers. "Camera-based navigation of a low-cost quadrocopter."
Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012.