SLAM FOR DRONES

PAVAN SAI TEJA (2017ME10583)
SHASHANK (2017ME10572)
CHAITANYA RAM (2017ME10576)

Prof. Subir Kumar Saha

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, DELHI

AUG 2020

ABSTRACT

Developing and testing algorithms for autonomous vehicles in real world is an expensive and slow process. In this project we will present a new simulator (AirSim) built on Unreal Engine that provides realistic simulations for implementing Autonomous algorithms. [5] AirSim is an open source cross-platform simulator for autonomous vehicles It supports hardware-in-the loop (HITL) and software-in-the-loop (SITL) with popular different fight controllers. The simulator is designed to be extensible to accommodate new types of vehicles, sensors, and software protocols. The modular design of AirSim enables various components to be easily usable independently in other projects and 3D environments. With the wide application of LIDAR applications in many fields, the LIDAR based SLAM technologies has also developed rapidly. We demonstrate LIDAR based SLAM algorithms by first configuring a drone with LIDAR sensor and map the 3D simulator environment in different 3D environments/scenes.

CHAPTER 1

Introduction

The Future is coming, and we cannot stop it now. We will be having flying machines all above us very soon transporting goods and even people. We have to design our machines so that we stay save from these flying machines because they will be in the open environment unlike a Metro where if any, the damage will be restricted to the particular metro rail and not to the others moving on the road or staying in their home; or the roadways where enough research has been done on the safety measures, now there is a considerable amount of automation already in our cars and vehicles. The problem with flying vehicles like drones is that they are intrinsically very unstable, and if there is any catastrophic failure, they become much more insecure and unstable. To solve this problem, we must bring Autonomy to the drones - The most important part of this will be the awareness of the surroundings. This can be achieved by Simultaneous Localization and Mapping. This will allow the drones to map the environment continuously and help itself navigate through the obstacles. Also, SLAM will help us map places for general purpose usage like mapping architecture, art structures…

To test out these SLAM algorithms, simulation is a better alternative than real world testing considering cost, speed, and ease of implementation. Currently, this is a non-trivial task as simulated perception, environments and actuators are often simplistic and lack the richness or diversity of the real world. [6]For example, for robots that aim to use computer vision in outdoor environments, it may be important to model real-world complex objects such as trees, roads, lakes, electric poles and houses along with rendering that includes finer details such as soft shadows, specular reflections, diffused inter-reflections and so on. Similarly, it is important to develop more accurate models of system dynamics so that simulated behaviour closely mimics the real-world. AirSim is an open-source platform that aims to narrow the gap between simulation and reality to aid development of autonomous vehicles. The platform seeks to positively influence development and testing of autonomous algorithms.

CHAPTER 2

Literature Survey

For developing SLAM using simulation, there is a vast collection of simulators, addressing all of them exhaustively is beyond the scope of this paper. A few popular selections that are close to our work are mentioned here.

1. Gazebo [1] has been one the most popular simulation platforms for the research work. It has a modular design that allows to use different physics engines, sensor models and create 3D worlds. Gazebo goes beyond monolithic rigid body vehicles and can be used to simulate more general robots with links-and-joints architecture such as complex manipulator arms or biped robots. While Gazebo is fairly feature rich it has been difficult to create large scale complex visually rich environments

2. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles that are closer to the real world and it has lagged various advancements in rendering techniques made by platforms such as Unreal engine or Unity.

3. Other notable efforts include Hector [2] that primarily focuses on tight integration with popular middleware ROS and Gazebo. Similarly, Rotors [3] provides a modular framework to design Micro Aerial Vehicles and build algorithms for control and state estimation that can be tested in simulator. Finally, jMavSim [4] is easy to use simulator that was designed with a goal of testing PX4 firmware and devices.

We settled to use Airsim for its rich API support for interaction between Companion computer and the Simulation engine. This simulation uses a quadcopter equipped with various sensors and cameras. The possible sensors and cameras in our Microsoft AirSim simulated drone are. [5]

• inertial measurement unit (IMU) • magnetometer • global positioning system (GPS) • barometer • one dimensional distance sensor • 2D-lidar • 3D-lidar • regular camera • infrared camera • simulation-only depth camera

In a simulation, computational cost drives our choices, we need extremely powerful computers to process 2D stereo pair images to get 3D depth data. With lidar, the project becomes more implementable on decently powered laptops.

CHAPTER - 3

PROJECT OBJECTIVES AND WORK PLAN

Problem Definition/Motivation: Currently to implement a SLAM algorithm for autonomous navigation in real world, we must fetch different sensors like Lidar, Vision sensors, etc. But for regular university researchers it becomes impossible as cost of these sensors are way beyond their budget.

In our project, we ventured to find a 3D environment simulator with configurable spaces, robots, and sensors for testing algorithms in any user defined environments. This allows us to verify and debug correctness of our algorithms for SLAM in different situations/environments. This reconfigurability of spaces and environments is not easily possible in real world testing. Simulation makes testing cycle much easier.

OBJECTIVES OF THE WORK

In robotics, Simultaneous Localization and Mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment using sensors while simultaneously keeping track of the robot’s location within it.

The objective of this work is to develop LIDAR based SLAM algorithms that estimate the trajectory of the robot and create a 3D occupancy map of the environment from the 3D Lidar point clouds and previously estimated trajectory.

Using LIDAR point cloud data obtained from 3D simulator, we try to obtain coordinates of the (environment surface feature points) obstacles with respect to drone frame.

We then implement suitable LIDAR based SLAM algorithms to map the environment simultaneously while the robot is moving.

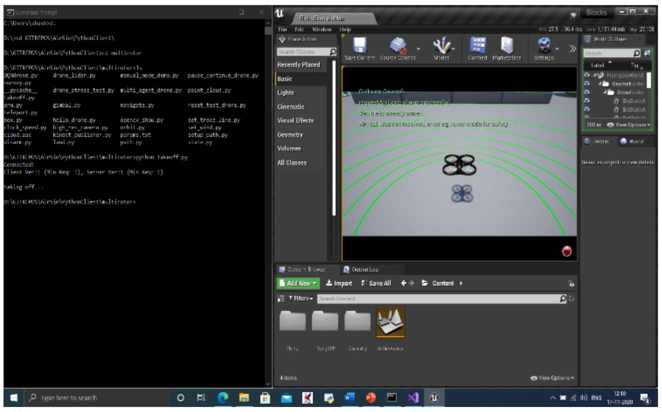
In our project, for simulating the drone and the environment we are using Microsoft’s Airsim project in Unreal Engine which supports hardware-in-the loop (HITL) and software-in-the-loop (SITL) with different fight controllers, our method makes testing algorithms cheaper and faster.

METHODOLOGY

- To simulate 3D environment, we install (latest) Unreal Engine 4.25.4 and build Microsoft's AirSim project that works with Unreal Engine (>4.24.0)
- We setup Python Clients to communicate with Simulation objects from the command line.
- We use Lidar data API to fetch point cloud data from the Unreal Engine simulation environment and map this data using an external python script.

Task Ahead:

- Point cloud is the raw 3D scan data, we need to study point cloud processing algorithms for feature estimation on which SLAM is implemented.
- We implement SLAM algorithms to map the environment simultaneously while the robot is moving.
- We demonstrate the repeatability of our algorithms by applying SLAM in various user defined environments.



```
C:\Users\shashod:

D:\>cd GITREPOS\AirSim\PythonClient\

D:\GITREPOS\AirSim\PythonClient>cd multirotor

D:\GITREPOS\AirSim\PythonClient\multirotor>ls
DQNdrone.py     drone_lidar.py        manual_mode_demo.py   pause_continue_drone.py
survey.py
__pycache__     drone_stress_test.py  multi_agent_drone.py  point_cloud.py
takeoff.py
arm.py          gimbal.py             navigate.py           reset_test_drone.py
teleport.py
box.py          hello_drone.py        opencv_show.py        set_trace_line.py
clock_speed.py  high_res_camera.py    orbit.py              set_wind.py
cloud.asc       kinect_publisher.py   params.txt            setup_path.py
disarm.py       land.py               path.py               state.py

D:\GITREPOS\AirSim\PythonClient\multirotor>python takeoff.py
Connected!
Client Ver:1 (Min Req: 1), Server Ver:1 (Min Req: 1)

taking off...

D:\GITREPOS\AirSim\PythonClient\multirotor>
```

Fig 1: CMD Prompt on left | 3D simulator on the right

GANTT CHART

TASK	Week 1-2	Week 3-4	Week 5-6	Week 7-8	Week 9-10	Week 11-12	Week 13-14
Literature Review about SLAM	🟦						
Evaluating Ros, Blender+Matlab, Microsoft AirSim	🟦	🟦					
Setting up AirSim Environment and collecting Data			🟦	🟦			
Plotting point cloud data and designing projectile of the drone for SLAM				🟦			
Studying Lidar SLAM and their implementation				🟦	🟦		
Testing our final code with multiple 3D environments					🟦	🟦	🟦
Report Making							🟦

CHAPTER 4

WORK PROGRESS

RELEVANT THEORY, EQUATIONS AND CODES

MATHEMATICAL DESCRIPTION OF SLAM PROBLEM

GIVEN

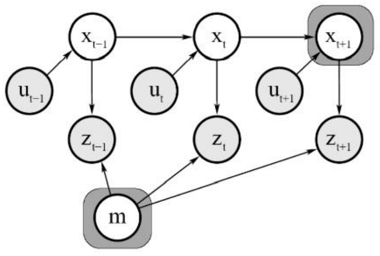
Robot Control inputs: $u_{1:T} = \{u_1, u_2, u_3 \ldots, u_T\}$

Observations: $z_{1:T} = \{z_1, z_2, z_3 \ldots, z_T\}$

SLAM PROBLEM IS TO FIND

Map: M

Path of the Robot: $x_{0:T} = \{x_1, x_2, x_3 \ldots, x_T\}$



$$p(x_{t+1}, m \mid z_{1:t+1}, u_{1:t+1})$$

Fig 2: Graphical Model

PLOTING LIDAR POINT CLOUD DATA USING CUSTOM PYTHON SCRIPT

```python
import numpy as np
import matplotlib.pyplot as plt

def myfunc(str):
    f = open(str,"r")
    f1 = f.readlines()
    a=len(f1)
    b = 3
    shape=(a,b)
    arr = np.full(shape,0,dtype= float)
    i=0
    for x in f1:
        result = x.split(' ')
        arr[i][0] = float(result[0])
        arr[i][1] = float(result[1])
        arr[i][2] = float(result[2])
        i=i+1
    return arr

matrix = myfunc("C:\\Users\\hp\\Downloads\\cloud3.asc")
# print(matrix)

# Data for three-dimensional scattered points
X=matrix[:,0]
Y=matrix[:,1]
Z=matrix[:,2]

ax = plt.axes(projection='3d')
ax.scatter3D(X, Y, Z, color = "green")
plt.show()
```

Description of the problem/case study

For Autonomous Navigation, we need

1. SLAM
2. Navigation
3. Motion Planning

But, in this project we are only addressing SLAM (Mapping and Localization). SLAM is the first step for building autonomous robots because without awareness of the environment robots can neither Navigate nor plan its trajectory

What is SLAM (Simultaneous Localization and Mapping)?

- Computing the robot's poses and the map of the environment at the same time
- Localization: estimating the robot's location
- Mapping: building a map
- SLAM: building a map and localizing the robot simultaneously

Exp. Setup

We are simulating a drone and its environment in Unreal Engine's AirSim project

- Download Unreal Engine
- Build AirSim Project
- Setup API control for Companion computer and the Simulation Engine
- Fetch Sensor Data using API Layer as shown in the figure below.

Matplotlib is a very useful python library for plotting and visualizing data.

- Import Matplotlib module to Python development and use it plot the data fetched from simulated Lidar sensor from the Unreal Engine's AirSim project environment.

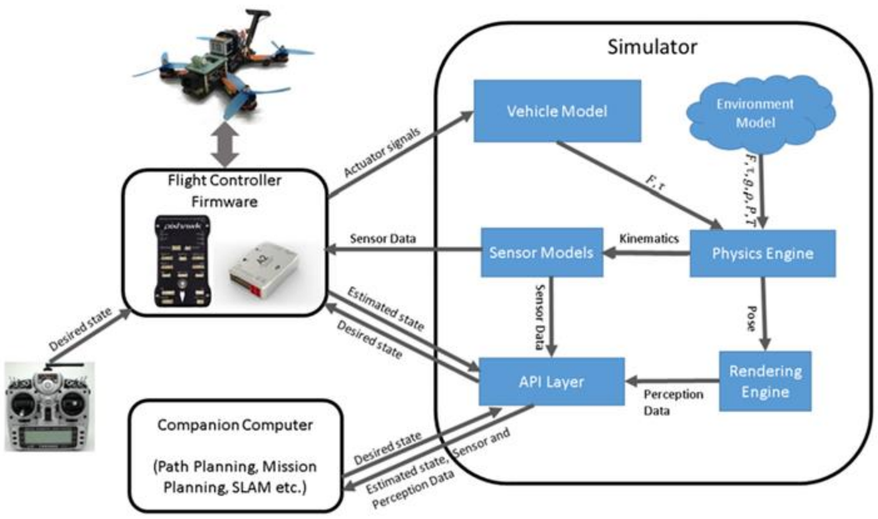
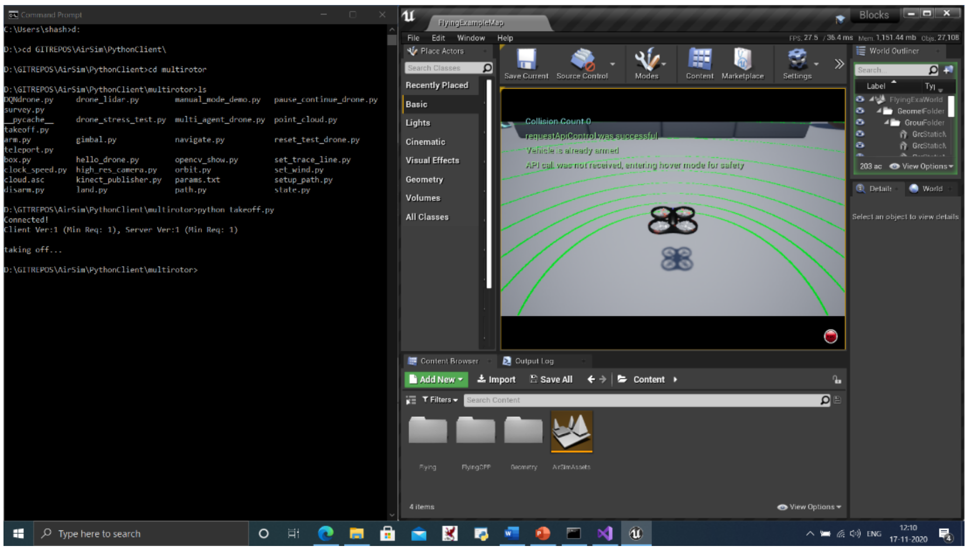


Figure 3:

Results and Discussion

- Airsim (3D Environment) Installation – Complete
- Python code of plotting 3D map from “.asc” format point cloud data is complete.



```
C:\Users\shash>d:

D:\>cd GITREPOS\AirSim\PythonClient\

D:\GITREPOS\AirSim\PythonClient>cd multirotor

D:\GITREPOS\AirSim\PythonClient\multirotor>ls
DQNdrone.py      drone_lidar.py        manual_mode_demo.py   pause_continue_drone.py
survey.py
__pycache__      drone_stress_test.py  multi_agent_drone.py  point_cloud.py
takeoff.py
arm.py           gimbal.py             navigate.py           reset_test_drone.py
teleport.py
box.py           hello_drone.py        opencv_show.py        set_trace_line.py
clock_speed.py   high_res_camera.py    orbit.py              set_wind.py
cloud.asc        kinect_publisher.py   params.txt            setup_path.py
disarm.py        land.py               path.py               state.py

D:\GITREPOS\AirSim\PythonClient\multirotor>python takeoff.py
Connected!
Client Ver:1 (Min Req: 1), Server Ver:1 (Min Req: 1)

taking off...

D:\GITREPOS\AirSim\PythonClient\multirotor>
```

```
Collision Count 0
requestApiControl was successful
Vehicle is already armed
API call was not received, entering hover mode for safety
```

Figure 4:

Discussion:

Currently, fetching of Lidar data has some issues. When we try to plot the received data as shown in the below figure 5 – it is not showing expected output. We must work on properly fetching Lidar data.

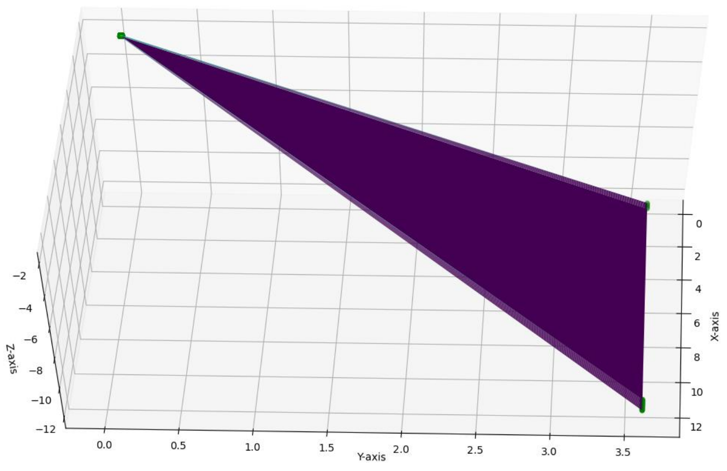


Figure 5:

CONCLUSION

Setup of the environment is complete. The task at our hands is to figure out a way to extract lidar data perfectly and to understand and implement suitable SLAM algorithm for Lidar data in the simulation. An example would be Graph Based SLAM algorithm.

We must implement the SLAM algorithms to map the environment simultaneously while the robot is moving. We must demonstrate the repeatability of our algorithms by applying SLAM in various user defined environments.

After the completion of SLAM, we can focus on Navigation, Motion Planning and Trajectory Planning for a complete autonomous package.

REFERENCES

1. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IROS (2004)

2. Meyer, J., Sendobry, A., Kohlbrecher, S., Klingauf, U., Von Stryk, O.: Comprehensive simulation of quadrotor uavs using ros and gazebo. In: SIMPAR, pp. 400–411. Springer (2012)

3. Furrer, F., Burri, M., Achtelik, M., Siegwart, R.: Rotorsa modular gazebo mav simulator framework. In: Robot Operating System (ROS), pp. 595–625. Springer (2016)

4. Babushkin, A.: Jmavsim. https://pixhawk.org/dev/hil/jmavsim

5. https://microsoft.github.io/AirSim/ (AirSim Documentation)

6. Cadena, Cesar & Carlone, Luca & Carrillo, Henry & Latif, Yasir & Scaramuzza, Davide & Neira, Jose & Reid, Ian & Leonard, John. (2016). Simultaneous Localization And Mapping: Present, Future, and the Robust-Perception Age. IEEE Transactions on Robotics. 32. 10.1109/TRO.2016.2624754.