2D Autonomous Mapping

Amukta, Tiffany, Kanupriyaa

Problem Statement

Autonomous vehicles are often used to explore difficult to reach places (i.e. Search and Rescue, Scientific Exploration). This project aims to create a low cost, educational demonstration of **autonomous mapping and navigation** by building a vehicle that explores a simulated environment.

Scope:

- Eliminate mechanical design challenges by reducing a real, textured environment to a **static environment with smooth ground and flat walls**
- We will build a ground vehicle capable of producing a **2D map of the maze**, traversing through it, and updating live changes to the maze

Requirements

1. Small wheeled vehicle capable of exploring entire maze in < 5 mins
    a. Maze must fit within 4ft x 4ft space (for demo)
    b. Must operate on battery for at least 2 hours
    c. Explores at rate of 20 ft / minute.
        i. Since maze design is variable, vehicle performance will be measured by distance/ time
    d. Vehicle must not hit walls
    e. Path must be optimized (vehicle traveling parallel to walls, minimize zigzagging)

2. Visualization of maze in 2D and location of vehicle relative to maze
    a. Visualization must update location of walls/obstacles as vehicle explores
    b. Vehicle (and visualization) must keep track of map coverage
    c. Vehicle must recognize areas it has explored already
    d. Localization error < 20% of vehicle dimension (width) or maze path width, or 1 inch

Challenges

- Developing a SLAM algorithm such that robot:
  - does not endlessly wander around (stuck in loops)
  - Is able to cover entire maze quickly
  - does not run out of battery in the middle of mapping
- Budget
  - Lidar and camera modules may potentially cost at least $100+ each
- ECE Areas:
  - Software
  - Signals

Solution: SLAM

We will build a software for our ground vehicle that simultaneously builds a map of the area and determines the robot's (own) position within the map using the SLAM (Simultaneous Localization and Mapping) method. The mix and match of these components determines what the final algorithm would be.

SLAM method components	
Mapping	Topological maps/ **Grid maps**
Sensing	**Laser scan (LIDAR)/ Visual feature based(Visual cameras)**/ Tactile sensors.

Solution: SLAM

Sensor modes	Landmark based/ **Raw data approach(Point clouds and images)**
Kinetics	**This is calculated using previous commands given to the vehicle, Odometry data**
Multiple Objects	Multiple object recognition is done through technology like JPDAF or PHD.
Loop closure	**Second algorithm to reset location priors/ Same algorithm using kinetic data collected**

Solution: Software

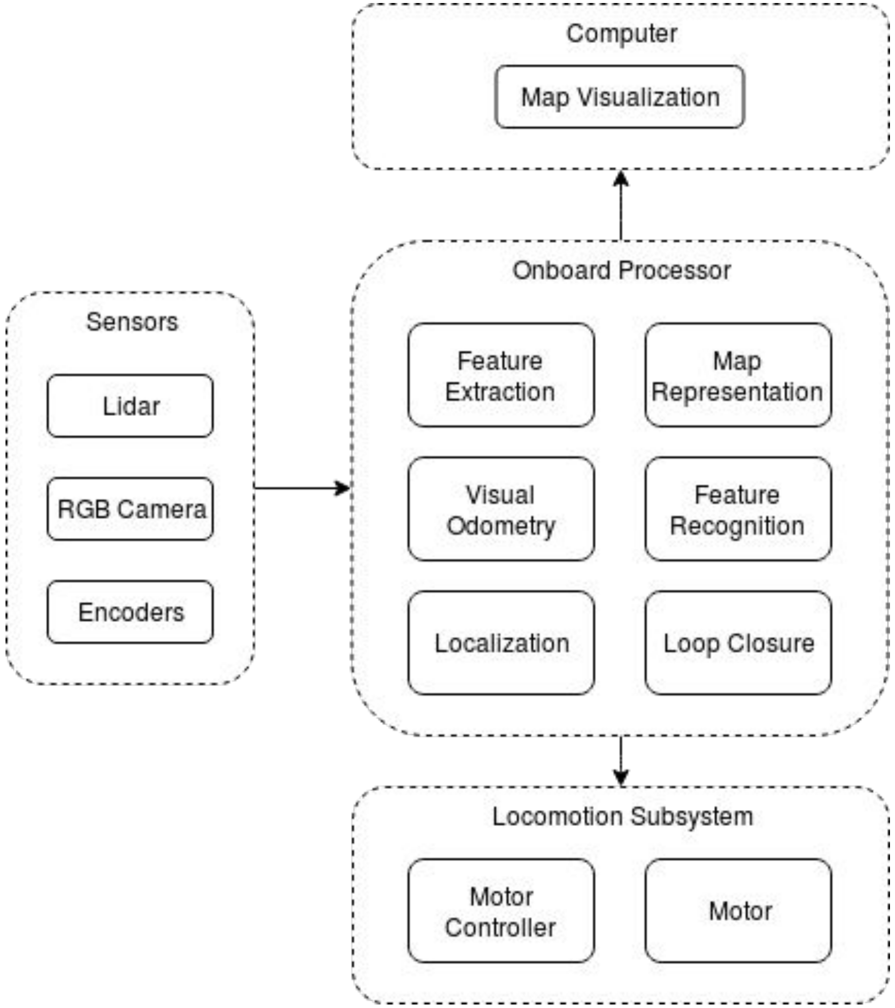
- We will be developing a web app using JS and Bootstrap for frontend UI and Django for the backend server. The computation for the backend is in python because most sensors and the raspberry pi come preloaded with python.
- According to our research (ongoing) we will be using python libraries such as numpy, scipy, matplotlib, cvxpy and pandas to process data.
- The Python Imaging Library will be used to generate the final png for the map.

Solution: Hardware

Sensor Type	Approx. Cost	Data
Single Point Lidar	$50 - $150	2D point cloud
Ranged Lidar	$450+	3D point cloud
Camera	$25 - $50	2D images
Depth Camera (Intel Realsense)	$200	2D images, depth information (0-8)

Solution: Hardware

- Processor: Raspberry Pi
- Lidar: Single point lidar is sufficient for 2D mapping
- Encoder: used as a benchmark to compare against feature based localization



Verification & Validation

Approach: build configurable maze to test various paths

1. Time trials of vehicle exploring maze, must complete within 5 min
   - Measure vehicle speed in straight line test, >20ft/min
   - Measure duration of one lidar/camera 360 degree scan and max distance vehicle can travel for next scan to collect overlapping image, verify vehicle can map at >20 ft/min
   - Maze must contain loops, hallways of varying width, and every type of 90 deg intersection
   - Vehicle must identify when it travels in a loop and recognize starting point
   - vehicle must keep track of map coverage, verified with coverage matrix, and return to intersections if it hasn't explored other paths. It must be able to minimize overlapping travel distance.
2. Visualization shows environment and location of vehicle in environment, and measured error for wall or vehicle placement must be < threshold

Work Breakdown Structure

Red = Amukta
Blue = Kanu
Yellow = Tiff

Lidar	Camera	Path planning	Visualization	Vehicle
Configurations to receive lidar points	Configurations to receive camera input	execute wall following and turns without collisions	Build web app and set up protocol to receive map info packets	Purchase or build suitable 2 wheeled chassis
Implement SLAM pipeline	Implement feature recognition	Minimize repeated routes when traveling to unexplored paths	Build visualization UI (with libraries)	Encoder and motor control IO with RPi
Implement loop closure	Testing	Implement map coverage matrix	Add vehicle live location, map coverage features	Implement benchmark localization with encoder data
Testing		Testing	Testing	Testing

Gantt Chart

TASK NAME	DURATION* (WORK DAYS)	TEAM MEMBER	PERCENT COMPLETE
Prototype 1 Goal: Simple user-centering feedback control loop running on skid steer base			
Research and order lidar and camera	2	Tiffany	0%
Figure out lidar and camera IO	4	Tiffany	0%
Set up Raspberry Pi	2	Amukta	0%
Set up a web application template	3	Kanu	0%
Install and learn python data libraries	1	Kanu	0%
Install and learn Python Imaging Module	2	Kanu	0%
Research SLAM Kinetics calculations	5	Tiffany	0%
Connect lidar and cam to software	4	Tiffany	0%
Research map representation	5	Kanu	0%
Configuration to recieve lidar points	7	Amukta	0%
Configuration to recieve camera input	5	Amukta	0%
Execute wall following and turns without collision	10	Tiffany	0%
Purchase or build robot chassis	10	Tiffany	0%
Implement SLAM pipeline	10	Kanu	0%
Implement map coverage matrix	5	Tiffany	0%
Implement feature recognition	5	Amukta	0%
Implement algorithm for loop closure	5	Kanu	0%
Build visualization UI for data	5	Kanu	0%
Build encoder and motor control		Tiffany	0%
Implement benchmark localization	3	Amukta	0%
Add vehicle live coverage	5	Amukta	0%
Add map coverage features	6	Kanu	0%
Build configurable maze	5	All	0%
Testing	3	All	0%