ECE 445 | Design Document
Multi-Agent State Estimation in a Partially-Observable Environment

Team 76
Kourosh Arasteh
Junwon Choi
TA: Amr Martini
Introduction

Objective

Many military base locations and military construction sites in developing countries are often GPS-denied and/or contested environments[1]. For such construction projects, collaborative robots pose an attractive solution to the problem of distributed, large-scale construction. However, without a robust mapping process for the environment, it would be impossible to develop a plan for distributed autonomous construction. An accurate and dynamic mapping of the region(s) of interest is necessary before autonomous construction robots can be deployed. Therefore, our problem statement is as follows:

How do we keep track of a large, sparse map between several ground agents, without using GPS or long-range localization technologies?

![Diagram](image)

Our research aims to solve the problem statement above by utilizing multiple unique agents, each tasked with one aspect of our multi-faceted mapping and localization objective. By utilizing 2 ground agents and 1 “eye-in-the-sky” agent, we’ll be able to discretize mapping to our ground agents and localization to our sky agent. Our ground agents’ sole objective
is to move about our small-scale sandbox and capture small “patches” of map data, and our sky agent’s sole objective is to determine the location and orientation of each ground agent. Ground agents relay their patches to a central controller agent (in our case, a PC) and the controller agent receives location data for each patch from the sky agent. Our controller then takes all patches of map data and their respective location data to accurately stitch them into one complete map of our entire environment. Our approach presents 2 novel advantages: considerably lower per-agent cost and significantly reduced computational resources required throughout the mapping process. We’re able to observe these advantages because instead of having one intelligent agent performing both localization and mapping, we have multiple “dummy” agents each focused on one single task, thereby decreasing computational complexity, material cost, and overall development time. A further novel aspect could be achieved in one of our growth goals, utilizing agents of different designs and capabilities to map our environment in a scheme that plays to each agent’s unique strengths.

Background

In applications of collaborative robotics, keeping track of the state of the environment is often attempted using individual agents that simultaneously perform localization and mapping onboard. To do so, these agents require powerful computation capabilities and constant communication with both GPS satellites and D-GPS towers. However, there are applications of robots like these in locations that are GPS-denied or contested to the point that extraneous long-range communication should be avoided or is unavailable entirely.

The inspiration behind our project and the problem we’re trying to solve comes from the Army Corps of Engineers and the current challenge they face in trying to map and characterize a construction site prior to construction. The development of a miniaturized testing environment will lend a robust testbed to develop novel algorithms in collaborative construction robotics problems.
High-Level Requirements

1. Sky agent should be able to localize the ground agent's position within ±5cm and orientation within ±15° within an 8' x 8' stage.

2. System should be able to stitch 64 LIDAR pointclouds, each measuring 1' x 1', from multiple ground agents with zero overlap into the global map using location and orientation data obtained from the sky agent.

3. System should be able to detect and create bounding boxes for objects within the stage similar in size to the ground agent (20cm² to 0.1m² top area).
Design
Block Diagram

Figure 2
# Requirements & Verification

## 1) 7.4V LiPo Battery

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The LiPo battery must be able to provide 7.4 V (±0.2 V) to the rest of the power subsystem for proper operation of the DC-DC converter. | 1. Connect the battery leads to an oscilloscope.  
2. Measure the output of the LiPo battery and make sure that it's providing 7.4 V (±0.2 V).  
3. The LiPo battery must be tested at various states of operational charge to ensure that the battery consistently provides the expected output regardless of its current charge level. |

## 2) Low Voltage Cutoff

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The cutoff device must open the circuit bridge between the LiPo battery and the DC-DC converter if power is in range 6.0V-6.5V. | 1. Attach a power supply to cutoff device in the same fashion that the LiPo battery would be connected.  
2. Using the power supply, sweep voltage from 7.4 V down to 0 V and note at which voltage the cutoff device operates.  
3. Cutoff device should operate in the 6.0V to 6.5V band. |
### 3) 7.4V→5V DC-DC Converter

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The DC-DC converter must be able to convert battery voltages from 6.5V-8.0V to 5.0V (±0.5V) at peak current draw, 1500mA, at scan startup of the LIDAR. | 1. Attach a power supply to the DC-DC Converter in the same fashion that the LiPo battery would be connected.  
2. Connect the output of the DC-DC Converter to an oscilloscope to measure the converted output voltage.  
3. Using the power supply, sweep voltage from 5.0 V to 8.0 V.  
4. We should see an output voltage of 5.0 V (±0.5V) |

### 4) H-Bridge Motor Controller

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The motor controller must be able to switch and pass 5.0 V (±0.5V) to both DC motors. | 1. Use an Arduino to code a simple testbench for left and right motor movement.  
2. Attach the outputs of the motor controller to an oscilloscope to read output voltage.  
3. Upon running the Arduino testbench, we should see the output voltage switch between +5.0 V and -5.0 V as the motor changes direction between left and right. |
### 5) Left & Right DG01D DC Motor

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| Both left and right DC motors must operate in both clockwise and counterclockwise directions in a range of 5.0 V (±0.5V). | 1. Connect the DC motor to a power supply in the same fashion that the motor controller would be connected.  
2. Using the power supply, sweep the voltage from 4.0 V to 6.0 V.  
3. We should observe that the motor operates between 4.5 V and 5.5 V.  
4. Repeat the same process with negative voltages (reverse polarity) to check that the motor is properly functioning in both clockwise and counterclockwise directions. |

### 6) Left & Right Motor Encoders

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| Each encoder must provide the orientation of its motor ±1° and be queried at a sampling rate of 500Hz. | 1. Utilizing an arduino as logger, we will hand rotate each encoder one degree around the entire 360 degree range, using a protractor as a reference.  
2. We will log the recorded encoder values and verify that each degree is properly captured by the encoder. |
### 7) 5V→3.3V DC-DC Converter

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| The DC-DC converter must be able to convert the 5.0 V power source voltages from 5V (±0.5V) to 3.3V (±0.3V) in order to power the Wi-Fi chip, which specifically requires a 3.3 V power source. | 1. Attach a power supply to the DC-DC Converter in the same fashion that the 5.0 V power source would be connected.  
2. Connected the output of the DC-DC Converter to an oscilloscope to measure the converted output voltage.  
3. Using the power supply, sweep voltage from 4.0 V to 5.5 V.  
4. We should see an output voltage of 5.0 V (±0.5V) |

### 8) ROS Controller

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
</table>
| Our controller agent must be able to send and receive data packets to/from our ground agents and sky agents via ROS nodes. | 1. Initialize all ROS nodes for ground agents, sky agent, and controller agent.  
2. To verify that the ground agents are communicating with the controller agent, observe the ground image processing ROS node that ground agents are publishing to and make sure proper map data (LIDAR or stereo camera) is sent.  
3. To verify that the sky agent is communicating with the controller agent, make sure that the sky agent is properly publishing 2D image data to the sky camera ROS node. |
Plots
Resolution Graph provided for RPLidar A2M8

![Resolution Graph](image3)

Example Point Cloud Output Plot of RPLidar A2M8

![Example Point Cloud](image4)
An important consideration in our system design is the total necessary data throughput of all the communication channels and components handling the local patch mapping data. The selected LIDAR, the RPLIDAR A2M8, has an angular resolution between (0.45°, 1.3°), a sample duration of 0.25 ms, and a typical scan rate of 10 Hz. We see below that we require a throughput of 4000 samples/second.

\[
10 \frac{\text{rotations}}{\text{second}} \times 360 \frac{\text{degrees}}{\text{rotation}} \times 1 \frac{\text{sample}}{0.9 \text{degrees}} \times 32 \frac{\text{bits}}{\text{sample}} = 16k \text{Bps}
\]
With a 32bit float representing distance in meters, we see that our system requires a 16kBps throughput rate. This informed our choice of the ATMega2560 MCU, which has a clock speed of 16MHz, and the ESP32, which can map and store up to 4MB/s in SRAM for packet partitioning and communication.

**Risk Analysis**

A large issue that may risk the progress of our project is being able to successfully calibrate all of the sensors at play. Our ground agents will be equipped with LIDARs and the sky agent will be some form of a cheap RGB camera. It is possible we may run into issues combining location data with our map data because our ground agent and sky agent use two entirely different sensors. We can overcome this potential obstacle by including additional metadata about the data packets that our controller agent expects to receive. By doing so, the controller agent can better understand the how data from each sensor is related and provides the stitching algorithm with more context. This also reduces the complexity of our stitching algorithm.

Another risk factor is the stitching algorithm itself. The completion of our environment map relies on our controller agent’s ability to stitch patches of map data together using location and orientation data provided by the sky agent. Developing an image processing algorithm to accurately stitch
all of our ground agent patches together may prove to be difficult, especially if incoming data from each respective agent isn’t necessarily uniform.

The wireless connection aspect of our project could also pose a risk because of the size-intensive nature of LIDAR pointclouds. Hardware on our ground agents could pose limitations on how fast these LIDAR pointclouds are relayed back to our controller agent. We will have to make sure that the pointclouds generated for each patch of map data are as small and distinct as possible. By doing so, we can guarantee that we’re passing the smallest pointcloud necessary to form a complete and comprehensive map.

**Safety & Ethics**

Our project includes several safety hazards that should be clearly identified. Most notably, our ground agents are equipped with LIDARs which are laser devices. The inclusion of LIDARs pose risk of vision damage should an individual look directly into the area where lasers are emitted. The RPLidar A2M8 has been certified by the American National Safety Institute (ANSI) as an FDA Class 1 Laser[2]. This means that it cannot emit laser radiation at known hazard levels (Class 1 lasers are less harmful than barcode scanners). However, Class 1 lasers can still become harmful if viewed through optical aids (such as binoculars or magnifying glasses) for extended periods of time. As a result, the LIDAR should never be brought to eye-level or looked directly into when powered on.

Another safety hazard involves the lithium-polymer batteries used to power our ground agents. Lithium-polymer batteries are necessary as a power source because of the significant amount of power consumed by the LIDAR for mapping. As a result, misuse of these batteries or carelessness can lead to extremely dangerous explosions if overcharged or introduced to extremely hot temperatures. Furthermore, over-discharging these batteries can lead to unpredictable behavior, and even explosions. We need to maintain charging temperatures between 0°C and 60°C and make sure to utilize our aforementioned low-voltage cutoff elements throughout the testing and implementation process.[3][4]
A potential safety hazard we must keep in mind is the possibility of our ground robots interfering with human movement and traffic. Although small, our robots could get in the way of people moving back and forth and can be a tripping hazard. For the scope of our project, this hazard is address with a boxed-in stage where the robots will be operating. The borders of our stage will be adequate enough in strength and size to stop a robot should it try to veer out-of-bounds.

Our project has one main risk in ethics violation, and that’s regarding privacy. At a high level, our system can map and localize any given area using multiple ground agents and an eye-in-the-sky. Unfortunately, it is possible that an individual, or a group of individuals, with malicious intent can use our technology to acquire localized map data of a property or region without consent. This is a direct violation of principle 1.6 in ACM’s Code of Ethics, which states that computing professionals have a responsibility to respect the privacy of the public and other professionals. [7] Misuse of our technology is also a violation of IEEE’s Ethics Code #1, which states that engineers should hold the welfare of the public paramount, and strive to comply with ethical design. [5] This is reiterated in the IEEECS Code of Ethics section 3.12, mentioning that we must ‘Work to develop software and related documents that respect the privacy of those who will be affected by that software. [6] To avoid misuse of the technologies introduced in our project, we plan to only run our project in a lab environment and avoid any case of mapping private property. Furthermore, we will maintain this technology as a research project, not for commercial distribution, which will avoid the issue of the public using the technology to invade privacy.
# Parts Cost

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Unit Cost</th>
<th>Qty</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP Lidar A2-M8</td>
<td>$299.00</td>
<td>2</td>
<td>$598.00</td>
</tr>
<tr>
<td>ELP Dual Lens Camera</td>
<td>$64.99</td>
<td>2</td>
<td>129.98</td>
</tr>
<tr>
<td>Turnigy LiPo 4000mAh</td>
<td>$55.08</td>
<td>3</td>
<td>165.24</td>
</tr>
<tr>
<td>ATMega2560 MCU</td>
<td>$12.35</td>
<td>4</td>
<td>49.40</td>
</tr>
<tr>
<td>Logitech Webcam C310</td>
<td>$27.84</td>
<td>1</td>
<td>27.84</td>
</tr>
<tr>
<td>ESP-8266 Wi-Fi Chip</td>
<td>$1.60</td>
<td>4</td>
<td>6.40</td>
</tr>
<tr>
<td>Mountain Ark Tracked Robot Chassis</td>
<td>$49.99</td>
<td>2</td>
<td>99.98</td>
</tr>
<tr>
<td>Cryton Dual-Channel Motor Driver</td>
<td>$18.29</td>
<td>2</td>
<td>36.58</td>
</tr>
<tr>
<td>Raspberry Pi 3B+</td>
<td>$35.00</td>
<td>1</td>
<td>35.00</td>
</tr>
<tr>
<td>70&quot; Tripod</td>
<td>$39.99</td>
<td>1</td>
<td>39.99</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$1188.41</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

# Labor Cost

Utilizing the expected BS CompE salary
2 engineers * $40/hr * 12hrs/week * 10 weeks = $9,600

# Timeline
<table>
<thead>
<tr>
<th>Date</th>
<th>Events</th>
</tr>
</thead>
</table>
| 2/15  | ● Design Document drafted  
      ● ROS software structure prototyped on PC                         |
| 2/22  | ● Circuit Schematics completed  
      ● Design Document Completed  
      ● Design Review Scheduled                                                |
| 3/1   | ● Parts Ordered  
      ● ROS software full implemented in PC module-testing                  |
| 3/8   | ● Wi-Fi protocols over ROS between all agents up and functional, able to transmit and unpack data  
      ● Firmware written for all components                                    |
| 3/15  | ● First-Round PCB submitted for review  
      ● Robot Chassis Control Unit-Testing completed  
      ● Webcam Control Unit-Testing completed                                  |
| 3/22  | ● All ROS Software Structures fully implemented for Controller Agent and Sky Agent |
| 3/29  | ● Final PCB submitted for review                                         |
| 4/5   | ● Ground Agent physical completion and testing  
      ● Ground Agent software completion and testing  
      ● Camera-based localization implemented and tested                     |
| 4/12  | ● Local map-patch passing implemented and tested from both Ground Agents  
      ● Ground Agent motion control loop implemented and tested                |
| 4/19  | ● Mock Demo                                                              |
| 4/26  | ● Demo  
      ● Mock Presentation                                                    |
| 5/3   | ● Final Presentation  
      ● Final Report  
      ● Lab Notebooks                                                          |

Citations


