Pluto Rover - Autonomous Navigation

Pallav Hingu

Contents

Chapter 1

Analysis

1.1 Description of the Project

Space Exploration has advanced massively in recent years. Large companies such as NASA and SpaceX are taking large steps forward to explore and colonize new planets such as Mars. We know a lot about the planets in our solar system: We have pictures, samples and critical data about the atmospheres of them. This has opened a gateway allowing us access to further resources from the planets in our solar system. But, there is one "planet" that we have yet to explore in detail; a "planet" where no man-made object has never landed: Pluto. Having been recently described on Google as: *"Our favourite dwarf planet since 2006"*, we have yet to discover its surface and internal composition. To continue our research of the planets in our solar system, we will need to **send robotic rover on a research and exploration mission to Pluto.**

As Pluto is extremely far away from the Earth, there are many obstacles to sending a rover to Pluto. Whilst the most obvious is transportation, NASA have already proved that they are capable of doing this using their "New Horizons" interplanetary space probe launched as part of the "New Frontiers" program. But, after getting to Pluto, there are many challenges that a rover can face. Firstly, due to the large distance between Earth and Pluto, there is a considerable delay in receiving signals, and there is also a lot of data loss. This means that the rover cannot easily be controlled remotely as there would be a large delay in sending instructions as well as receiving camera/sensor feed, which means that the camera may not represent the actual situation of the rover when the feed is received on Earth. As this poses a large threat to the condition of the rover, the best approach to address the safety of the robot is to install an autonomous navigation software, which avoids all obstacles in it's path and uses other available sensors on the rover to get more information on the surrounds.

The information received using these sensors will be stored on the rover as well as sent to the nearest orbiting satellite, which would have been used to transport the rover as well. This satellite will then also store the data received from the rover's sensors and then will broadcast these results to Earth, allowing it to be analysed.

I will design a software with a friendly user interface (which will be on the companion app) that presents live information from the rover on a remote device, which can be on a satellite to monitor the rover and receive crucial data. The rover is on Pluto to detect the presence and location of metals on the surface. My software will utilize IoT protocols to present the output of the metal detector. It will also use a built-in IR camera to perform autonomous driving. A smart algorithm will also be used to generate the relative location of the rover on Pluto's surface according to its set landing location and landing orientation. This will allow the rover's position to be estimated, as GPS cannot be utilised on Pluto.

The aim of this mission is to detect if there are metals near the surface of Pluto, if yes, then to identify and record where.

1.2 Computational Methods

I think that this problem lends itself well to the computational methods such as abstraction and decomposition for a variety of reasons. The solution will be a software that controls the Pluto rover on a hardware level, allowing it full autonomous control as well as manual, using the rover's cameras and sensors. This will simply need to run on a computer as that will be the medium on which the images from the camera will be processes and the next course of action for the rover will be determined. There is no alternative solution for this problem that will not require a computer, due to the context of the problem, and the environment in which the solution will need to function (the harsh climate of Pluto).

1.2.1 Methods Used

Some of the computational methods used for the solution are:

- **Problem Decomposition**
  This large problem can be decomposed into many smaller steps. These steps can be thought to be the following:

  1. Use the rover's camera system to get some RAW data about the surroundings
  2. Add threshold to data to limit the amount of data being passed through the program
  3. Plot an image representation of the RAW data
  4. Use image processing to detect the obstacles in the image
  5. Use a smart algorithm to determine the action needed to avoid obstacles
  6. Send instructions to a driver board (allowing hardware control to motors)
  7. Use an algorithm on the driver board to control the motors
  8. Use a feedback loop to get a confirmed movement of the rover
  9. Calculate vector location of rover using its movement
  10. Get data from the metal detector
  11. Write vector location and metal detector data to local file
  12. Send data to server for companion app

  After these steps are completed, the problem can be said to be solved. The steps will allow the rover to react to obstacles in its surroundings, allowing for autonomous navigation. Also, the steps allow the rover to utilise a metal detector and send its data to a server for analysis, meaning that we can get more information about the surroundings of the rover and the surface of Pluto. The vector location calculation will also allow for the rover to be located using no other hardware components, using only the velocity and angular velocity of the rover.

- **Problem Recognition**
  The overall and apparat problem is autonomous navigation through an extreme terrain. But, the underlying and primary problem is the detection of obstacles and their location. If the obstacles can be located, this information can be passed through a set algorithm that will allow a motor response.

- **Thinking Procedurally - Divide and Conquer**
  All the decomposed steps above allow for the problem to be divided into smaller problems, which are easier to solve than the large overarching problem. Creating functions/modules to solve the smaller problems, and them combining them all a larger modular program is an example of divide and conquer as the large program had been divided onto smaller problems, that each formed modules that could easily be combined.

- **Abstraction**
  Abstraction plays a large role in ensuring that the program responds to input data immediately. The incoming data stream from the camera sensor contains a lot of information, most of which is irrelevant for this application. So, it is vital to utilise abstraction and extract useful information only from the camera's input data. In this case, we would add a threshold so that the data only represents obstacles a certain distance from the camera. This allows us to reduce the amount of information being fed into the program, decreasing its response time and increasing its speed. The obstacles that are detected by the camera but are further away from the camera are removed as they do not need to be considered yet, they do not pose a risk of collisions yet.

- **Logic Modelling**
  Logic modelling will allow me to illustrate how the program must work to fulfil the system requirements. It will also be helpful to depict the actual program's activities compared to its intended needs, allowing me to measure how successful the program is at fulfilling it's function and solving the problem. As this will help with the application evaluation, it will be useful when improving the program as it ensures that all the requirements are fulfilled well.

1.3 Stakeholders

The ideal stakeholders for this project would include the large space research firms and their representatives. They would be the clients that would use this software and adapt it for space exploration missions. But, as this is not possible right now, the stakeholders for my project will be people who have done research around artificial intelligence, space exploration and robotics.

My first stakeholder will be a family relative who has completed a PhD in artificial intelligence an is current a professor at a university. I will attempt to add some type of artificial intelligence within the software so that it can learn more from its surroundings and adapt to the current situation. The university professor will use my software to create a demonstration for young student visitors during university open days. This will allow the young students to get inspired and learn more about the future application of artificial intelligence. My software will be one of the suitable solutions for the university as it will be free to use and will have a simple concept, GUI and set-up. While many alternative demonstrations of artificial intelligence on robots require a long set-up time, my software will be very simple to get ready and will be reliable so that there are not unexpected surprises on university open days.

My second stakeholder will be my schools Young Engineers groups who have engineered a prototype Pluto rover for their competition this year. They will use this software to allow their prototype rover to move across a modelled terrain of Pluto. My solution will benefit this group highly as it will allow them to test their prototype, while also testing my software in the most realistic situation that I have access to. The solution will be most appropriate to this group's needs as it will provide them with a free software that is specifically designed for their rover's hardware and is optimised for the computational power that is available to them. Other solutions would either cost a lot of money or time and would need much more computation power, making them very expensive and complex.

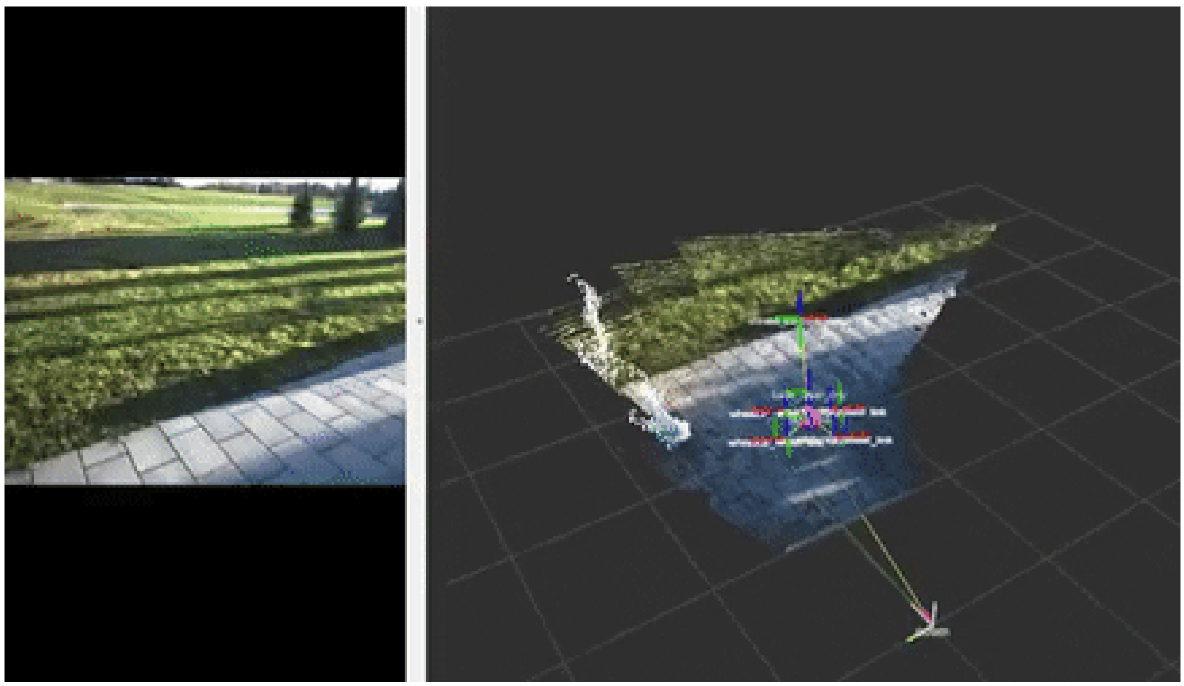
1.4 Research

1.4.1 Existing Solutions

As this is not a commonly commercially developed program, there are very few re-viewable examples available online. It is also very specific to the context, which may be a reason why the program is uncommon. But, if we widen the scope and look for software that are not specifically designed for this application but can be used in this context, we can find more programs online.

Examples:

- **RTAB-Map with ROS**



Overview

RTAB-Map (Real-Time Appearance Based Mapping) is an open-source application which uses different SLAM (Simultaneous localisation and mapping) approaches to generate a 3D dot-cloud of the surroundings of a sensor. It can be installed and used as a stand-alone application to generate 3D dot-clouds based on sensor inputs, or can be integrated into ROS (Robot Operating System) to allow the application to control hardware according to the 3D dot-cloud generated by a camera/sensor on a robot. It produces a 3d Map of the surrounds and uses localisation to detect the robot's location in relation to the 3d generated map. it then uses this to detect obstacles and find the most efficient way around the map to a given goal location.

ROS, which is essential in this application, is a robotics middle-ware and allows the RTAB-Map here to publish real-time messages to the appropriate nodes. In ROS, a node is a process that performs computation. Nodes are combined together in a graph and can communicate with each other to perform certain tasks. RTAB-Map, when integrated well with ROS, publishes messages to many external nodes, which communicate with hardware drivers in the system to move the robot according to RTAB-Map's instructions. As RTAB-Map can get a large image of the surroundings, it uses a smart algorithm to conduct path finding, meaning that it calculates the most efficient route to reach a destination while avoiding obstacles.

As this is a complicated process, it requires a lot of advanced hardware and a high amount of processing power. ROS is also extremely difficult to use, as well as install correctly. ROS is used for many complex robots, like the Robonaut 2 aboard the International Space Station.

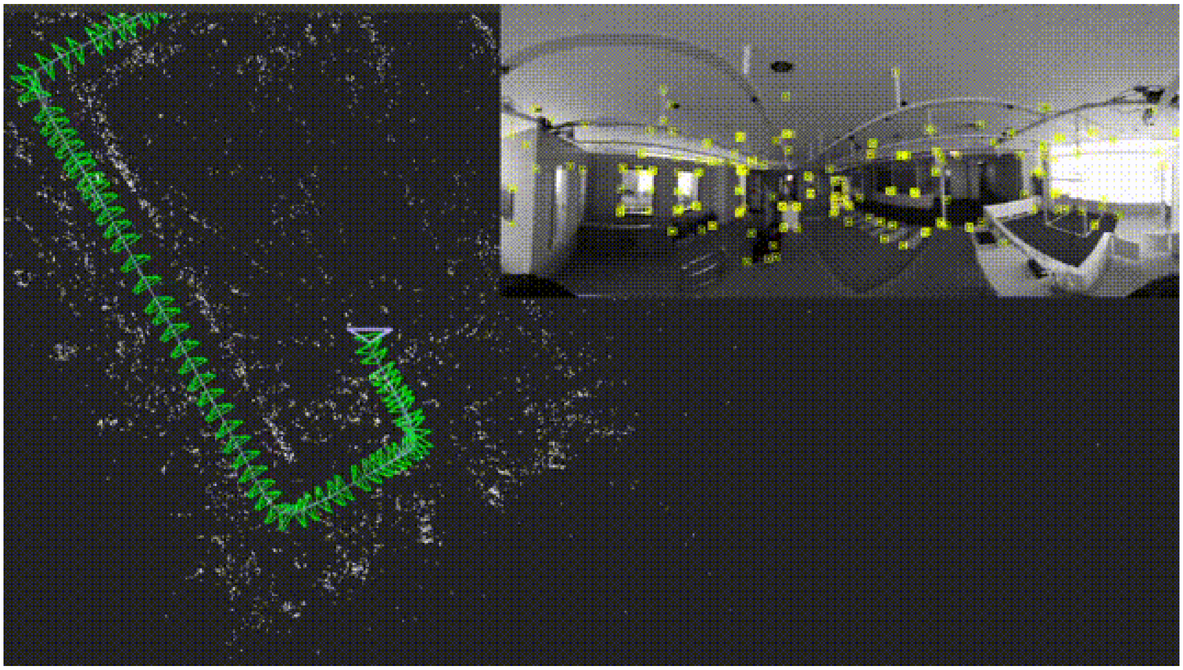
The Graphical User Interface(GUI) that it uses displays lots of information, such as the 3d Dot-map created of the surroundings in real-time, the raw images that are being captured, and the calculation of odometry (the relative location of the vehicle). This is a simple but effective and advanced user interface that provides a lot of data that can be useful for environmental analysis, but is not hard to use. As this application is not specific to a problem, the GUI can be adjusted to display less or more information, or different information by tweaking the settings. This allows the application to be adapted to many different situations, making it ideal even for an autonomous Pluto Rover.

Parts that I can apply to my solution

From this application, there are many things that I can use in my application:

- *Display Odometry* - the application displays the odometry (relative location) of the vehicle on the GUI. This will allow the relative location of the rover to be displayed on the GUI so that the Rover can be located.
- *Hardware Control* - the software can control hardware directly by converting the output into electrical signals for the components. This will allow the Rover's motors to be controlled by my application directly.
- *Navigation* - this application can determine movement to navigate around a map with a set objective. This will be useful to move the Rover around the surface of Pluto with an objective to detect metal.
- *GUI settings* - the use of GUI settings allows the displayed information on the GUI to be adjusted according to user preference or need. This can also be helpful in low resource scenarios as displaying less information means that the program runs smoother and faster. This could be helpful on the Rover as there are limited computational resources and so the adjustment of the GUI to the scenario may be helpful in improving performance.

- **OpenVSLAM**



Overview

OpenVSLAM is a monocular, stereo and RGB-D visual SLAM system similar to RTAB-Map. It is a software that uses pattern recognition to detect movement and estimate current location of the object, its odometry. But, the large advantage with this application is that it is compatible with many different cameras and so can be used for a variety of different projects. This system is also completely modular as it is designed by encapsulating several functions in separated components with APIs.

When comparing OpenVSLAM with RTAB-Map, we can see that RTAB-Map displays a lot more data. This is because it creates a dot-cloud with as much detail as possible to generate an accurate computational representation of its surroundings. While this data may be useful for many applications, it is not essential for obstacle detection and autonomous navigation. On then other hand, OpenVSLAM generates a simple but informative 2D birds eye map of the movement of the vehicle. It also creates a simpler dot-cloud by only detecting the presence of obstacles and not their shape, size or colour. This allows it to use minimal relevant data to navigate, which is optimal as it reduces the amount of data being processing, reducing processing time, latency and hardware demand.

The minimal approach on this monocular SLAM software makes it more suited for our application than RTAB-Map as we have limited hardware, and an overload of unnecessary data may product uncertainty in the reliability of the software.

Parts that I can apply to my solution

There are a few things from OpenVSLAM that I can apply to my solution:

- *Input Abstraction* - the ability to abstract the relevant information form the input device to suit the need of the software. This allows the software to be more efficient as well as reliable.
- *GUI Simplicity* - the GUI for OpenVSLAM is very simple and effective as it displays essential data making it easy to read, ensuring that there is not an overload of data being displayed to cause software lag as well as user confusion.

1.4.2 Primary Research

Questions for Interview

To ensure that an interview with the stakeholders is as effective as possible, the questions to be asked will require planning so that all required information related to the solution can be communicated.

I will first interview the ***Artificial Intelligence Professor*** with the following questions.

Background Questions:

1. *How important do you think artificial intelligence to be for the Pluto rover?*
2. *In this case, what can artificial intelligence do that a standard algorithm cannot?*
3. *How will adding artificial intelligence impact the hardware utilisation of the program?*

Questions for client requirements:

1. *What are your expectations for the GUI?*
2. *What hardware would you be able to use for running this software?*
3. *What should you be able to control remotely?*
4. *Would you prefer using a companion app on a computer or wiring up the embedded computer?*
5. *Is there anything else you would like to add?*

Apart from the above question, I may ask follow up questions or ask them to elaborate on certain answers.

After completing the interview with the Professor, I will interview the ***Young Engineers group*** with the following questions.

Questions for client requirements:

1. *What are your expectations for the GUI?*
2. *What hardware would you be able to use for running this software?*
3. *What should you be able to control remotely?*
4. *Would you prefer using a companion app on a computer over wiring up the embedded computer?*
5. *Is there anything else you would like to add?*

These questions are the same as the client requirement questions from the Artificial Intelligence Professor, allowing us to compare and create a collective requirements list to suit both stakeholders.

I have included some background questions for the Professor so that I can use the answers when attempting to implementing AI into my program.

Interview

Artificial Intelligence Professor:

Background Questions:

1. ***How important do you think artificial intelligence to be for the Pluto rover?***

   "I think that the use of artificial intelligence is extremely important in a rover, especially if it made for Pluto as the distance between the user and the rover will be very far. Therefore, we will need something to control the rover locally without relying on a human, and so artificial intelligence will be needed."

2. ***In this case, what can artificial intelligence do that a standard algorithm cannot?***

   "Well there are many things that an artificial intelligence model can be used for in this scenario. For example, it can be used to move the rover by observing its surroundings, it can be used to predict and recognize hazards. Unlike standard algorithms, it can learn more from the information it is observing, allowing it to perform better in unaccounted situations."

3. ***How will adding artificial intelligence impact the hardware utilisation of the program?***

   "Well, there are many different ways of implementing artificial intelligence, so it does depend on your chosen method. But, in general, AI utilises more hardware than a standard algorithm, especially if it is constantly learning from it's environment. So, I would say that adding AI to the program will increase its hardware requirements quite dramatically."

Questions for client requirements:

1. ***What are your expectations for the GUI?***

   "In this case, it would be most effective to use a simple GUI that allows control over all vital hardware features that are accessible using the software. I would also expect to see some sort of statistics on the current state of hardware or the output of mounted sensors."

2. ***What hardware would you be able to use for running this software?***

   "For demonstrations with this software, I would have access to a computer and possibly the model rover needed for the physical demonstration. It would be ideal if I could use any computer to control the rover."

3. ***What should you be able to control remotely?***

   "As the software is designed for long range control, I would expect options for manual control for as many things as possible. there are many things that can go wrong and so the user should have the choice to control all hardware possible remotely, as well as the movement of the rover obviously."

4. ***Would you prefer using a companion app on a computer or wiring up the embedded computer?***

   "It would definitely be easier to use a companion app, as well as much more professional. Wiring would increase the chances of errors and make everything much more fiddly during demonstrations, and so I would definitely prefer a companion app. But, the companion app should be easy to use and should be reliable, having quick setup times as well to make it robust."

5. ***Is there anything else you would like to add?***

   "I think I have said everything I would like to. Thank you."

<u>Young Engineers Group:</u>

1. ***What are your expectations for the GUI?***

   "A clean user interface with complex functionality allowing full control of the rover. We vision the GUI with a clean look presenting the location of the rover and a few buttons that allow full control of the rover."

2. ***What hardware would you be able to use for running this software?***

   "To run the embedded app, we have a Raspberry Pi 3 that is built into our model rover. Alongside, we also have an Arduino PCD that is hardwired to all rover sensors and motors. The Arduino can also communicate with the Raspberry Pi. If needed, we also have access to a remote laptop. Regarding the rover's sensors, it has a Kinect 360 RGB-D camera along with a magnetic coil for metal detection and a receiver for a RC transmitter. The Rover also has 4 DC motors that can be used for movement. This is the hardware that we would ideally run the application on."

3. ***What should you be able to control remotely?***

   "It would be nice to be able to control the rover's movement remotely to control the rover as required during testing and demonstrations. We should also be able to switch quickly between autonomous and manual control, which would helpful during presentations of the rover at events."

4. ***Would you prefer using a companion app on a computer over wiring up the embedded computer?***

   "It would be helpful to have the option use a companion app but we would also want to be able to control the rover with a RC transmitter so that it can be manually controlled without having do download specific software."

5. ***Is there anything else you would like to add?***

   "One thing we would like to add is that the Kinect sensor can see objects that are not directly in front of the rover. This may need to be considered during development."

1.5 Features of the Proposed Solution (System Goals)

1.6 Software and Hardware Requirements

1.7 Success Criteria

1.8 Limitations