Autonomous New Planet Rover Report

By Jiajia Chen For DGMD S-17 2021 Robotics, Autonomous Vehicles, Drones, and Artificial Intelligence

Abstract

The purpose of this project is to study the feasibility and challenges of the computing systems for autonomous rovers that will be used to explore new planets and moons. Out of scope for this project are physical constraints such as gravity and radiation. The motivation here is to evaluate modern techniques from Computer Vision, Machine Learning and Deep Learning to perform obstacle avoidance, object classification and object following, and to consider their application towards this futuristic scenario in which rovers help humans explore outer space while ensuring self preservation by preventing collisions, and autonomously making decisions to set goals and to navigate towards those goals. While the 6 levels of autonomy are defined in the human world, these autonomous rovers for space exploration will fit into Level 5 where they must perform in conditions not ever experienced by humans. I demonstrate in this project the capabilities of an autonomous rover to explore a test environment for which it has not been precisely trained and yet achieve its goals. The project revealed the computing challenges in the physical world, with bottlenecks in memory, networking, and speed of storage that can defeat the objectives of autonomy, thereby highlighting the need for efficient utilization of computing resources. Work on this project also revealed the need for constant recalibration of the autonomous systems for small physical variances such as wheels not perfectly aligned and causing a very slight sideways deviation when the intended path is straight. Future work on this project includes continuous learning to understand the rovers' unexplored environment with deep reinforcement learning.

1. Introduction

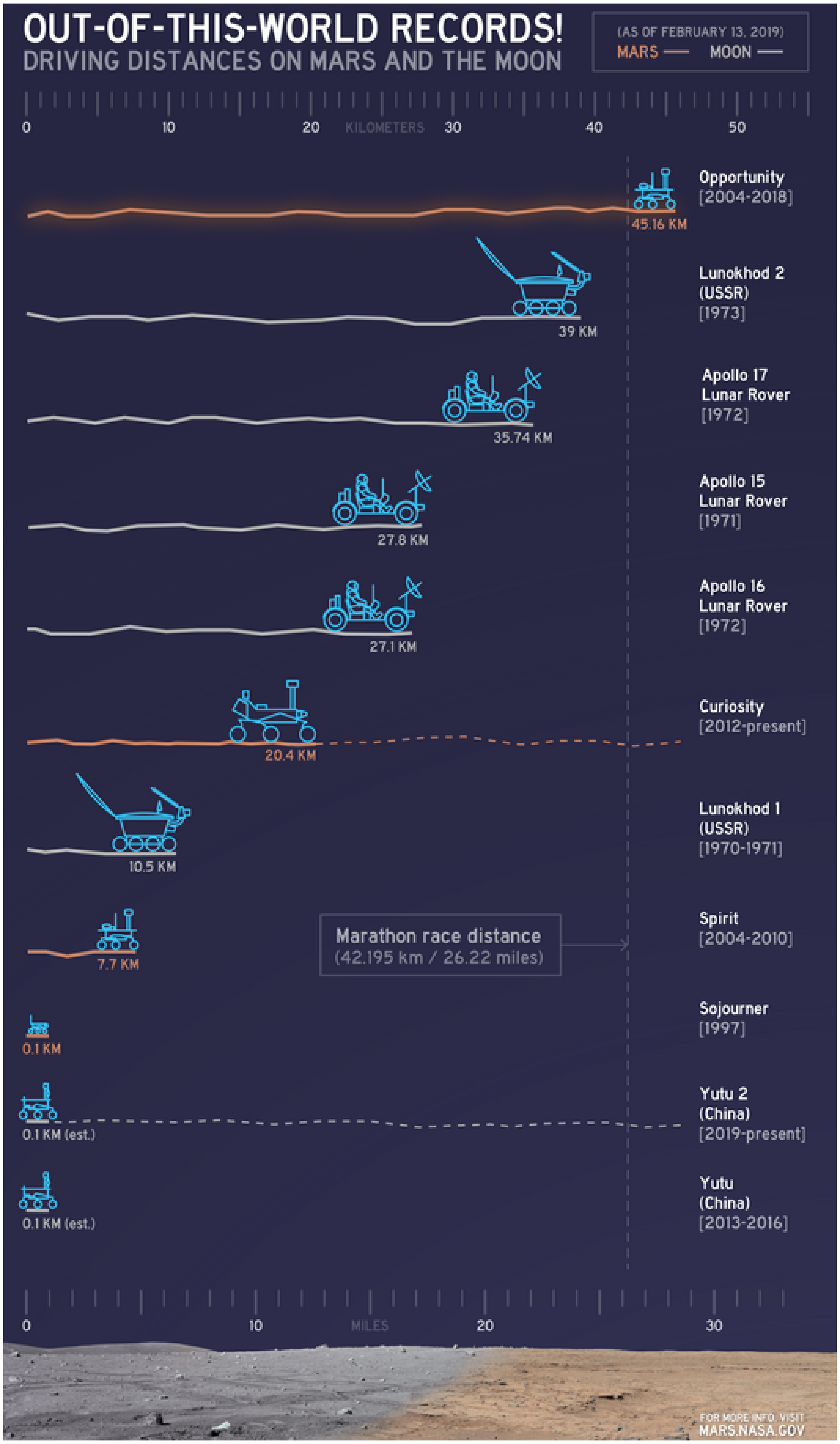
The first commercial manned spacecrafts are launching with Virgin Galactic, Blue Origin, SpaceX, etc. Over the next few decades, these companies and government organizations will reinvest to explore even further in outer space. Planets and moons will be our targets, both within and outside our solar system. However, it would be both inefficient and too dangerous for humans to be going to each planet or moon ourselves. We will need robots to help us create spacecraft in space, and launch them with autonomous drones and autonomous rovers onboard. In this project, I focus on the autonomous rovers that will operate at the surface of unexplored planets and moons.

1.1 Project Goals

In this project, I build a physical prototype of an Autonomous New Planet Rover, a robot at Level 5 autonomy on an open course. I focus on the rovers that will operate at the surface of unexplored planets and moons, specifically on their computing systems for self-preservation, decision making, and navigation. The ultimate goal is to build an autonomous new planet rover which keeps on exploring interesting new territories with transmission of data about discoveries of interest at the surface of the new planet or moon, while protecting itself from damage by avoiding crashing into obstacles or falling over edges. To achieve more of the ultimate goal in this project, I added to the original goal of collision avoidance the goals of object identification, autonomous determination of objects of interest, and autonomous planning and navigation towards such objects of interest.

1.2 Level of Autonomy

My rover operates at Level 5 autonomy, with no human attention. This kind of rover can make its own decisions about where to go and how to protect itself. Rovers on unexplored planets and moons need to continuously navigate at the surface, avoid damage from collisions or falls, and pay very close attention to specific topics of interest. NASA's Mars rovers accomplish these through human intervention in perception, planning and control, but such human-driven operation takes time to process and it takes between 3 and 21 minutes to transmit data each way to or from Mars. During those 42 minutes or more that the rover is waiting on Mars for a human to tell it what to do next, a lot can happen in real-time in its environment that will damage its equipment. While such human-managed exploration is inefficient and risky on Mars, it becomes infeasible beyond Mars. Hence Level 5 of autonomy is required for the application of rovers towards exploration of new planets and moons. However, Level 5 of autonomy comes with its own challenges, some of which my project identifies.


OUT-OF-THIS-WORLD RECORDS!
DRIVING DISTANCES ON MARS AND THE MOON
(AS OF FEBRUARY 13, 2019)
MARS
MOON
0
10
20
KILOMETERS
30
40
50
Opportunity [2004-2018]
45.16 KM
Lunokhod 2 (USSR) [1973]
39 KM
Apollo 17 Lunar Rover [1972]
35.74 KM
Apollo 15 Lunar Rover [1971]
27.8 KM
Apollo 16 Lunar Rover [1972]
27.1 KM
Curiosity [2012-present]
20.4 KM
Lunokhod 1 (USSR) [1970-1971]
10.5 KM
Spirit [2004-2010]
7.7 KM
Marathon race distance (42.195 km / 26.22 miles)
Sojourner [1997]
0.1 KM
Yutu 2 (China) [2019-present]
0.1 KM (est.)
Yutu (China) [2013-2016]
0.1 KM (est.)
0
10
MILES
20
30
FOR MORE INFO, VISIT MARS.NASA.GOV

1.3 Description

In this project, I create a physical prototype of a fully autonomous rover that will operate in environments for which it cannot be fully trained, protect itself from damage, identify objects or events of interest, make local planning and control decisions to continuously explore the planet's surface, steer intentionally to get sufficiently close to objects of interest, and then stop at a safe distance to observe and transmit to us its new discoveries. In my rover, I use images to determine objects of interest, but these could be easily replaced on the rovers that we send to another planet or moon with scientific equipment to identify molecular structures, minerals, life forms, etc, that are of specific interest for a specific mission. To accomplish these objectives, I use physical hardware and ensure its satisfactory operation under test conditions, setup and configure software systems to perform all necessary tasks, gather training data and perform transfer learning on top of existing neural network models, program for multiple use-cases to operate together as one system, and test for successful operation of each component and for the integrated system of the rover. I record videos and capture photos to study and report my findings, and I gather all code, data and artifacts to prepare the reports and presentations.

2. Related Work

While I could not find a similar project as mine operating at a similar budget of less than $300 and with only a month to build, test, and report, I found some related thinking from NASA in the following:

- Future Planetary Rovers May Make Their Own Decisions by Michael Schirber July 05, 2012.
  While the title suggests decision-making, the author is describing

decisions related to taking images to document its surroundings, not self-navigation or self-preservation.

- Sample Return Robot Challenge for Autonomous Technology
  The goal of this challenge in which several entries have participated each year, the robot must navigate challenging terrain and return predetermined samples within 30 minutes.
- Meet "Hedgehog": Engineers build cube-like rover for exploration of asteroids, comets
  The astronomers must give the hedgehog a plan, and inform it about its location, and then this hedgehog will make some local decisions
- NASA Robots Compete Underground in DARPA Challenge
  These rovers map objects within 5 feet while driving around autonomously within underground tunnels.

3. Team Organization

This is a solo project, with no other team members because I have only odd hours late night Pacific time after work and family commitments to focus on the project. I did this project entirely on my own.

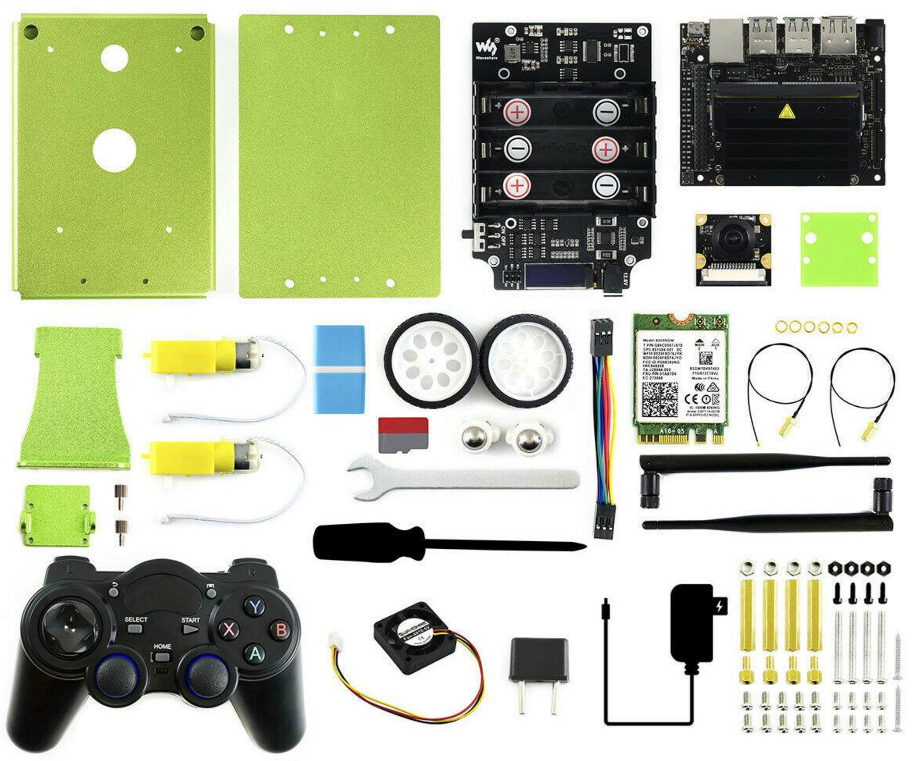
4. Hardware, Software and Development Tools

This is a physical prototype. It is important to pay careful attention to the hardware and infrastructure, software, setup, data collection, and model training to get this functioning right.

4.1 Hardware & Infrastructure - Minimum Requirements

- Jetson Nano 4GB
- Waveshare Jetbot AI Kit with single camera
- SD card with at least 64GB

- Laptop
- Ethernet wired & WiFi Network
- Monitor, keyboard, mouse
- Physical test environment for the rover to explore



4.2 Software - Minimum Requirements

- Jetbot latest image from Nvidia / Waveshare
- Jupyter Lab server running on the Jetbot which also offers Python and its standard libraries
- Balena Etcher software on the laptop to write the image to the SD card

- Ubuntu/Windows OS on laptop, and browser (Chrome, Firefox, etc)
- Torch, OpenCV, Numpy, etc Python libraries on the Jetbot via Jupyter Lab

4.3 Project Setup

- Following the guidance at the following pages, obtain the Jetbot image and write it to the SD card using Balena Etcher https://github.com/NVIDIA-AI-IOT/jetbot/wiki/Create-SD-Card-Image-From-Scratch https://www.waveshare.com/wiki/JetBot_AI_Kit The Balena Etcher software worked better on Windows, Ubuntu has more troubleshooting capabilities for issues with Jetbot
- Assemble the Waveshare kit into a Jetbot using instructions here https://www.waveshare.com/wiki/JetBot_AI_Kit_Assemble_Manual
- Connect the Jetbot to a monitor, keyboard and mouse, and connect a wired connection for the Jetbot.
    - Update all Ubuntu software on the Jetbot and set up a known username and password
    - Set up the WiFi connection for the Jetbot to operate untethered
    - Make sure the Jetbot's IP address shows up on the little Waveshare OLED screen for both the wired and wireless connections
    - Detach the wired connection and make sure that the Jetbot still gets a wireless connection when untethered when turned off and on
- Once the wireless connection works on the Jetbot, from a laptop computer go to http://<jetbot_ip_address>:8888 and enter the Jetbot's username and password

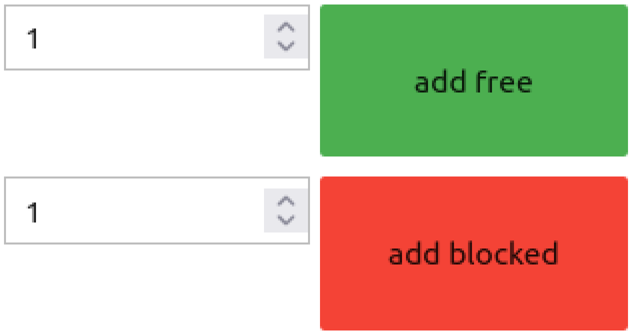
- Try the Nvidia examples starting with basic motion to make sure that the Jetbot works
- For coding, the Jupyter Lab site provides all functions and serves as a development environment with Python included. To install additional libraries, either do so from the Jetbot setup above with a monitor, keyboard and mouse, or type ! followed by the terminal command, such as:
  !pip install keras
- To use my code, download from Github my repository. Transfer to the Jetbot by drag-drop into Jupyter Lab. First run all cells in *rover_model_training.ipynb*, which will generate the *dataset* folder and the model *best_model.pth* on the Jetbot. Download *ssd_mobilenet_v2_coco.engine* from https://jetbot.org/master/examples/object_following.html and place it in the same folder as the ipynb files. Set up your physical environment and place the Jetbot where you wish to test the rover's autonomous features. Run all cells in *rover_logic.ipynb*, except the final cell which stops the Jetbot. To change from teddy bear to another class, find its number in *ms-coco-labels.txt* and enter the new number in *rover_logic.ipynb* in the text area (not in the code) where you see the number 88 currently.

4.4 Data Collection

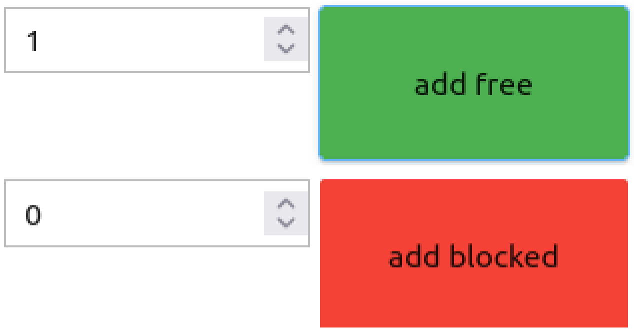
250 images of 2 classes -

- Blocked = Jetbot will hit the object in the image if it moves forward
- Free = Jetbot can safely move forward

```
[5]: display(image)
     display(widgets.HBox([free_count, free_button]))
     display(widgets.HBox([blocked_count, blocked_button]))
```



```
[5]: display(image)
     display(widgets.HBox([free_count, free_button]))
     display(widgets.HBox([blocked_count, blocked_button]))
```



4.5 Machine Learning and Deep Learning Models

I used the following two models:

- AlexNet - To perform transfer learning to train 2 classes of Obstacle Avoidance
- COCO - To perform object identification into generic classes, rather than specific object names, as would be the case with an unexplored planet or moon

4.6 Changes to The Plan

The project was not smooth sailing from start to finish.

- Given sufficient time, a rover on an unexplored planet or moon could theoretically explore all accessible parts of that planet with random motion. Rovers don't have an infinite lifetime and they also need to weather harsh conditions that limit their lifetime. To help the rover more efficiently complete its mission, I added intentional navigation towards objects of interest identified from long distances. To mimic this use-case, I added to my goals object identification, detection of interesting objects, and steering intentionally towards such objects to explore them further. To implement this use-case, I did object identification, motion or direction planning, and object following, in addition to collision avoidance.
- I had intended to teach the rover not only collision avoidance, but also fall avoidance, such as over a cliff or over the edge of a crater. However, with only 1 camera in the Waveshare kit and no other sensors, the training data and the live images have no relative altitude information for points within the image. Monocular depth perception lacks such altitude information too as the amount of floor visible above the bottom edge of the image does not reflect the relative altitude of different parts of that floor. That amount of floor visible at the edge of a crater can appear to be sufficient to move forward if the soil looks the same outside and inside a crater. Note that the term depth in monocular depth perception does not equal altitude, rather it is about depth in perspective images. With edge detection and additional sensors including more cameras, such altitude perception could be improved. However, without additional sensors, I decided to set aside fall prevention for future work.

- The Sparkfun 2.1 kit was a lot of work to assemble. The chassis in the kit has fastening holes that do not line up with the holes in the Nano. I have no intention of drilling holes into the Nano. Next, screws that came with the Sparkfun kit could not even go into the holes for the Nano. The Sparkfun's OLED screen never turned on. Even if it had turned on, it could not provide an IP address because the WiFi module that came with the Sparkfun kit has device drivers for Windows and Mac, but not for Linux which runs on the Nano. To hack through any of these incompatibilities, I would need an HDMI connection to the monitor, but the HDMI was intermittent and unreliable. I abandoned plans of using the Sparkfun kit.

4.7 Troubleshooting

- Make sure that the Nvidia examples work with the Jetbot
- Make sure that the SD card is at least 64GB and is rated U3 V30 or close to such read/write speeds to avoid storage bottlenecks
- The Jetbot must operate close enough to your WiFi router to have a good connection and avoid network bottlenecks.
- Use a wired connection to troubleshoot camera lag issues
- Reduce frames per second (fps) to 10 or lower if sufficient for your application, as higher fps overloads the Jetson Nano processor, memory and network stack
- Reduce motor speeds and turn gain to make sure the movement of the Jetbot is at a speed manageable for bottlenecks with the system, but set it high enough to generate sufficient torque for the given surface; I needed motors to run at least at 0.2 to generate sufficient torque on my carpeted surface

5. List of Milestones

- 07/04/2021: Have the Wareshare Jetbot vehicle assembled and the laptop and the development environment set up.
- 07/11/2021: Start a preliminary design of the architecture and collecting/tagging data. Confirm viability of Jetbot and Waveshare equipment.
- 07/18/2021: Finish a few iterations of using pre-trained AlexNet and adding 250+ samples collected by myself from the training and test environment in which my device will work. Iterate based on the performance of road tests and finalize the project plan.
- 07/25/2021: Complete training with sufficient training data of all the scenarios in which the rover will operate. Validate with a lot of testing that the model works. Deploy a model to the Jetbot, and explore ways to improve performance of a real world rover on other planets or moons.
- 08/01/2021: Prepare all the artifacts needed for the final project.

6. Results and Discussion

6.1 Data Collection

I collected 250 images with the robot's single camera placed in front of random objects that I did not use in its final test environment. I took the photos using the data_collection.ipynb Jupyter notebook under the obstacle_avoidance folder, and classified the objects as blocked or free to move forward.

6.2 Model Training

Because the rover is meant to be fully autonomous and explore new environments where training is not going to be possible before the rover gets to the unexplored planet or moon, I did not train the Jetbot specifically for objects similar to those in my test environment. However, I did transfer learning to training the robot on what I considered to be a safe distance to avoid collision, and I used another model for general classification of objects that help the Jetbot make decisions about where to go next.

6.2.1 Training For Collision Avoidance:

I trained the Jetbot to classify images of its surroundings into 2 categories -

- Blocked: An immediate obstacle hindering further movement in the forward direction, or
- Free: Safe for further movement in the forward direction.

I did this training by applying transfer learning over the existing AlexNet model using the 250 images I captured as mentioned in the Data Collection section above.
My model uses monocular depth perception to distinguish between the two classes of situations. Peeling the layers of the resulting neural network, one can see that the model distinguishes two classes based on the amount of floor it can see between the bottom of the image and the bottom of the nearest object in front of it.
Empirically this can be confirmed by moving the camera and retesting what the Jetbot considers blocked or free - if the camera is pointing down to the floor, it get closer to an object that it considers a blocking object because it believes there's more safe space to go forward, and if the camera is pointing straight in front it can see less of the floor and hence it stops far from the obstacle.

6.2.2 Training For Object Identification:

I could have used various existing convolutional neural networks available for object identification, such as AlexNet, VGG16, etc. However, doing so would have deviated from the scenario of the rover on an unexplored planet where it might encounter objects never seen before. In addition, a large model would overload the Jetbot's limited memory and computational resources. To achieve even more general classification I chose the COCO model to classify objects with the nearest class of objects known to humans. As a result, the Jetbot is intentionally set up to be error prone in its identification.

6.3 Use Cases

My project implements several use-cases that together mimic the operations of the Jetbot on an unexplored planet or moon. The Jetbot autonomously determines whether

6.3.1 Collision Avoidance

Using the neural network that classifies images into blocked or free, my rover determines on its own whether it is currently obstructed from moving forward or not. If the rover determines that it is blocked, it will turn its wheels and thereby its fixed camera to the left. In the ideal situation, the camera should turn without having to turn the entire Jetbot, but within the limitations of this project, the trade-off was acceptable. At 5 FPS, the Jetbot is still seeing several images during its left turn, and it determines whether it is safe to move in the direction of view of its camera. It will continue to turn to the left until it is safe to move. In the case of a lagging camera due to bottlenecks mentioned above in the Troubleshooting section, the Jetbot may end up rotating infinitely. This is an error condition to resolve beyond the prototype where after a fixed number of rotations, the Jetbot pauses to clear out its queue of images and overcome the lagging camera. When the rover determines that it is not obstructed, it continues to move forward.

6.3.2 Object Identification

My rover is able to analyze any image it sees from its camera and immediately classify most shapes in its view, even though the classification is not accurate as will be the case on an unexplored planet or moon. Using bounding boxes, I display the classification, and the confidence level of such classification.

6.3.3 Goal Setting

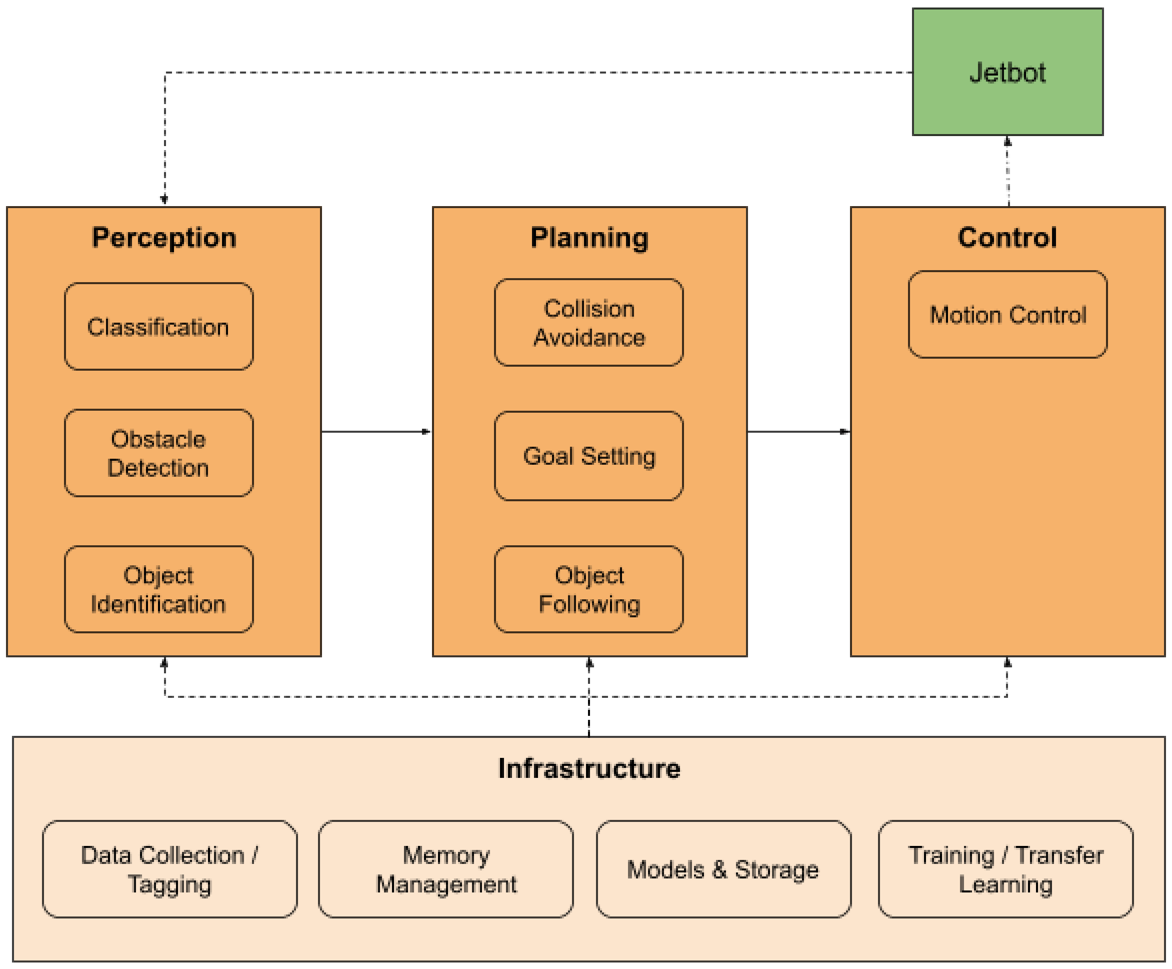
Among the objects identified may at times be certain objects of special interest that the Jetbot decides it must go investigate. When my rover locates an object of interest in any image in its camera's view, it immediately sets that object as a goal, turns itself in the direction of that object, and begins to move in that direction. This autonomous decision-making about the rover's own goals on an unexplored planet or moon is an important feature of my rover. In situations where it sees multiple such objects of interest in a single image in its camera's view, my rover prioritizes the object with the highest confidence of belonging to the class of special interest. For my prototype rover, the object of interest is an object on earth, such as children's toys that can be seen in a camera image. However, on an unexplored planet, the item of interest may be a molecule, a structure or texture or shape, a solid or liquid or gas, a life-form, or any variety of things that we may need to explore. Such a real life robot could also explore multiple such classes at once using different sensors beyond my simple camera, and my project is designed to add additional sensors as appropriate.

6.3.4 Object Following

With Collision Avoidance in place, the Jetbot is already protected from crashing into the object of interest. Hence now it can move towards the object of interest. When the rover is close enough to the object of interest, it will stop at a safe distance due to the Collision Avoidance system I mentioned above. This is also typically the safe distance to observe, gather data, and transmit such data to a satellite spacecraft around the planet or moon, or all the way back to us on Earth. In the 4.5 billion years of our planet, mobile life forms large enough to destroy such a rover have been alive only recently. The rover would need to be programmed for backward motion to run away from a threatening mobile alien species, and in such a situation, it would need a camera on the back side too so that it does not collide into objects in its back. However, my rover is already capable of following a mobile object of interest while maintaining a safe distance to avoid collisions.

6.4 System Architecture

The following are components of the system of my rover.



6.5 Results & Discussion

After resolving several challenges, the Jetbot based rover is able to achieve most of the intended goals to varying degrees of accuracy.

6.5.1 Obstacle Avoidance

The rover avoids obstacles 100% of the time when the object is in front of the rover's single camera. However, the rover does not have egomotion to understand its own dimensions, especially dimensions relative to the 3D space it occupies. As a result, when an object is on its side, the rover may at times collide or run over an obstacle with one of its edges. This problem can be reduced by providing the Jetbot better training data to also classify as obstacles objects that are close to the left or right bottom of the image, not only in front of the image.

6.5.2 Object Identification

The rover is able to see objects 10+ feet away in the test environment. Depending on resolution of the camera and the training data, the rover may see objects farther away. Within an image, the rover recognizes in real-time well over 50% of the objects as objects that belong to one of the classes in the COCO dataset. For the real-life rover that operates on an unexplored planet or moon, the model would need to be efficient enough to classify most and only the potential objects we expect to observe on those planets or moons. In addition, the rover should be learning from its surroundings as minerals, rocks, life forms, etc, of specific types may be common in that environment and their traits may be different from objects we recognize on Earth. This implies that the data gathered on the planet or moon must be fed into a pipeline that revises the object classification model running on the rover.

6.5.3 Autonomous Goal Setting & Object Following

It will be an important feature for an autonomous rover to set and adjust its navigation goals on the unexplored planet or moon. My rover prioritizes the highest probable object of the target class, adjusts it's orientation to point its camera and thereby the rover's body towards the target, and then moves forward towards that goal. At each available image, the rover re-evaluates the objects in its view, resets its goal, and readjusts its orientation.

This is where my rover's problems begin with goal setting and object following. First, the rover's two motors do not move perfectly straight ahead. I have neither been able to perfectly calibrate these motors from Waveshare, nor received an exchange for them. The motors I have make the robot turn very slightly leftwards. This leftward motion changes the robot's angle of view of the target object. What looked like a teddy bear from an angle earlier now looks slightly different from an angle slightly to the left. As the rover gets closer to the target object, this slight leftward motion forms a larger angle of deviation from the earlier angle from which the target object looked like a teddy bear. Unless the rover is fully trained to classify a teddy bear correctly from every angle, it may at some point decide that what it sees from closer and from a different angle is no longer a teddy bear.

This also brings up another problem, that the rover moving towards a target does not remember the goal. With no coordinates available to remember the exact position of the target and no GPS. If the rover comes across an obstacle along the way, it will successfully avoid the obstacle, but while doing so it often can no longer see the target object, and hence it forgets its goal and moves on to other motion.

The addition of a mechanism to remember the absolute position of an object in addition to the relative position of the rover would be valuable to successfully do goal setting and object following.

6.5.4 Hardware, Infrastructure And Efficient Software

One of the most significant revelations of this project has been the minute attention to detail required to get every aspect of the rover absolutely right. I have listed them in the Troubleshooting steps above. Working with hardware reveals all kinds of real-life issues, many of which would be difficult to simulate even if the simulation writer knows about these possibilities. There will absolutely be bottlenecks in hardware, networking, processing, memory, storage, etc, so based on the use-cases of priority, you have to make several trade-offs. Training for fully autonomous vehicles on a remote planet requires training not only for safe states, but also for how to get back to a safe state from an unsafe one. Because most of the time, the next state of the rover is most likely to be similar to the previous state (e.g. moving towards an object of interest), humans use our short-term memory to decide our actions, and the rover would greatly benefit from similar cumulative short-term memory which rapidly decays the weight of old information to make decisions such as which direction to move in the absence of GPS and map on a new planet or moon. The infrastructure and hardware must be chosen to minimize bottlenecks expected in the system, while the dataset, models, software architecture, etc, must also be efficient to minimize situations where the bottleneck may come into play. Of course, with gravity, radiation, planetary weather, etc, the systems will need to build in redundancy, failure recovery, etc, too.

7. Conclusions

- Working with actual hardware exposes many real-life issues.
- There will absolutely be bottlenecks in hardware, networking, processing, memory, storage, etc, so based on the use cases of priority, you have to make several trade-offs.

- Training for fully autonomous vehicles on a remote planet requires training not only for safe states, but also for how to get back to a safe state from an unsafe one.

8. Future Work

- Add more cameras and sensors to get better perception of relative altitude of different points in the image
- Apply deep reinforcement learning
- motion backwards away from moving objects
- Take the rover out among the soil, pebbles, rocks, ditches, etc

9. References

- https://github.com/NVIDIA-AI-IOT/jetbot/wiki/Create-SD-Card-Image-From-Scratch
- https://www.waveshare.com/wiki/JetBot_AI_Kit
- Future Planetary Rovers May Make Their Own Decisions by Michael Schirber July 05, 2012. https://www.space.com/16414-future-planetary-rovers-decisions-software.html
- Sample Return Robot Challenge for Autonomous Technology https://www.nasa.gov/directorates/spacetech/centennial_challenges/feature/2016_sample_return_robot_challenge_award.html
- Meet "Hedgehog": Engineers build cube-like rover for exploration of asteroids, comets https://phys.org/news/2016-02-hedgehog-cube-like-rover-exploration-asteroids.html
- NASA Robots Compete Underground in DARPA Challenge https://www.nasa.gov/feature/jpl/nasa-robots-compete-underg

round-in-darpa-challenge