University of Tartu
Faculty of Science and Technology
Institute of Technology

Uku Ilves

Towards Large-Scale Automated Mission Planning for Robotic Lunar Operations - Early Prototyping

Bachelor's thesis (12 ECTS)
Computer Engineering

Supervisors:
MSc Rando Avarmaa
MSc Quazi Saimoon Islam
MSc Hans Teras

Tartu 2024

Abstract

Towards Large-Scale Automated Mission Planning for Robotic Lunar Operations - Early Prototyping

This thesis presents the development of a Vehicle Routing Problem (VRP) solver prototype designed in particular for lunar surface mission planning. The prototype manages time dependent lunar shadows, demonstrating its utility for further assessment in simulation. Its transferability, relying on a celestial body's Digital Elevation Model (DEM) and solar data using the Skyfield API, makes it applicable for other extraterrestrial contexts within our solar system.

As large scale space mission planning trends toward increased automation, this thesis presents an early prototype addressing parts of this shift. By tackling the challenges posed by time dependent shadows and automating the route planning for the mission planner, this thesis contributes to the field of lunar exploration.

Despite the solver's effectiveness in managing time dependent shadows, the single threaded nature of the algorithm limits the processing speed of the solver. However the potential enhancements of this VRP planner prototype offer promises for a more efficient and accurate mission planning tool for future extraterrestrial missions.

CERCS: T120 Systems engineering, T320 Space technology[1]

Keywords: Mission planning, VRP, Global planning, Local planning, Shadow map, Horizon map

Resümee

Suuremastaabiline Automatiseeritud Missiooni Planeerimine Robotitele Kuupinnal - Varajane Prototüüpimine

Antud lõputöö tutvustab sõidukite teekonna planeerimise probleemi lahendaja prototüübi väljatöötamist, mis on mõeldud eelkõige kuupinnal missioonide planeerimiseks. Antud prototüüp haldab ajast sõltuvaid kuuvarje, näidates selle kasulikkust simulatsioonis edasiseks hindamiseks. Selle ülekantavus, mis tugineb taevakeha digitaalsele kõrgusmudelile (DEM) ja päikese andmetele, kasutades Skyfield API-t, muudab selle kohaldatavaks ka muudes maavälistes kontekstides meie päikesesüsteemis.

Kuna suuremahuliste kosmosemissioonide planeerimise suundumused on suurenenud automatiseerimise suunas, esitleb see lõputöö varajast prototüüpi, mis käsitleb selle suundumuste osi. Ajast sõltuvate varjude tekitatud väljakutsetega tegelemisel ja missiooniplaneerija jaoks teekonna planeerimise automatiseerimisel aitab käesolev lõputöö kaasa Kuu uurimise valdkonda.

Vaatamata lahendaja efektiivsusele ajast sõltuvate varjude haldamisel, piirab algoritmi ühe lõimeline olemus lahendaja töötlemiskiirust. Siiski pakuvad selle VRP planeerija prototüübi võimalikud täiustused lubadusi tõhusamaks ja täpsemaks missioonide planeerimise tööriistaks tulevaste maaväliste missioonide jaoks.

CERCS: T120 Süsteemitehnoloogia, T320 Kosmosetehnoloogia [1]

Märksõnad: Missiooniplaneerimine, VRP, Globaalne planeerimine, Lokaalne planeerimine, Varjukaart, Horisondikaart

List Of Figures

Contents

Abbreviations And Definitions

TSP - Traveling Salesman problem

VRP - Vehicle Routing Problem

POI - Point of Interest

DEM - Digital Elevation Model

CUDA - Compute Unified Device Architecture

GPU - Graphics Processing Unit

CPU - Central Processing Unit

Path - points between two points of interest

Point of interest - The location for carrying out a part of the mission

1. Introduction

The resurgence of lunar exploration marked by ambitious programs like NASA's Artemis, China's Lunar Exploration Programme, India's Chandrayaan mission and various private sector initiatives, highlights a renewed global commitment to the Moon. These programs not only aim to expand the human presence but also to leverage robotic missions for scientific gains. As more missions get announced and their objectives grow more complex, the necessity for mission planning systems becomes increasingly critical.

Navigating the lunar surface presents many challenges due to its harsh and unique environment. Historically methods such as blind driving, teleoperating and autonomous obstacle avoidance have been utilized, however these approaches alone face limitations due to factors like the vast distances involved, limited on board computing power which in turn introduce substantial communications latencies, and excessive waiting times on the obstacle avoidance algorithms, making traveling larger distances impractical.

Historically to overcome these challenges, space missions involving rovers most often employ a combination of local and global planning systems:

- **Local planning:** Often referred to as autonomous obstacle avoidance, involving real time navigation in an environment that is unknown, requiring the rover to employ remote sensing and localization algorithms to avoid hazards.
- **Global planning:** Involves planning the routes of the entire mission based on a known environment to optimize travel objectives over longer distances and durations.

However the integration of these systems into comprehensive mission level planners is a recent development. Mission planning encompasses the allocation of resources, scheduling of activities, mission parameters, communication windows for the entire duration of the mission, adapting to changing parameters and/or objectives. Previously the University of Tartu and Milrem Robotics, developed a vehicle routing system to use it as a mission planner by solving the vehicle route planning problem (VRP) between points of interest (POI-s). VRP is used to find the optimal route between POI-s which is important when minimal travel distance is important and/or other factors like having efficient battery use.[4]

The developed mission planner exhibits trade offs, accounting for dynamic shadow movement that lead to long computation times of the schedules. Additionally the dynamic path planner checked for shadow presence during each iteration and to manage the complexity of the real time shadow movements the system used a slack parameter to approximate shadows. This parameter is fine tuned on a per mission basis which may not always yield optimal results. [4]

The VRP solution results in a mission plan which includes the order in which points of interest should be visited along with a schedule that includes estimated arrival and departure times.[4]

This thesis aims to circumvent some of the drawbacks present in the mission planner developed by the collaboration of the University of Tartu and Milrem Robotics, specifically the long computation time and approximation of shadows with the slack parameter.

By developing automated mission planning systems for extraterrestrial operations it is possible to increase the distances covered and increase the scientific output of space missions. These systems are designed to minimize human intervention and to reduce risks during the lifetime of a mission.

2. Literature Overview

This section gives an overview over the critical aspects of vehicle routing that need to be addressed for the lunar surface. The examination of Lunar shadows offers insight into the impact moving shadows have over route planning. The planning strategies introduce existing strategies already used in extraterrestrial missions and current mission planners give an overview over how ongoing missions in space deal with navigation, where vehicle routing for mission planning gives an overview of the downsides in current planners and introduces the idea of large scale mission planning.

2.1. Lunar Shadows

Lunar surface shadows are a significant environmental factor when it comes to mission planning. The characteristics of the cast shadows, specifically their rapid movement near the Lunar terminator and having extensive length have direct implications on mission safety. Because there is no thick atmosphere on the moon to distribute heat, areas in shadow can have temperatures as low as -210 °C. Since days and nights on the moon last very long, rovers would need adequate equipment to handle around 14 days of extremely cold temperatures or avoid them. [5,6]
Since the moon doesn’t have a significant atmosphere, there is also no scattering or reflection of light on the moon, this results in dark shadows with no indirect illumination in contrast to Earth where shadows can be dimly lit. This is also a problem for visual odometry, since it is so dark in a shadow a robot can’t make out shapes or surface features, having also established that shadowed regions are also extremely cold it is desirable to avoid those regions as much as possible. [7]

2.2. Current Mission Planners

Despite advancements being made in large scale mission planning, ongoing space missions employ two styles of mission planning: global and local, which are often combined.
In local planning which is also referred to as tactical planning the environment is totally unknown or partially known and it requires the robot to employ remote sensing and localization algorithms to avoid obstacles which primarily arise from the low

resolution of a DEM where small, but still significant terrain features are lost. This however is costly since it is done on limited hardware available on the robot.

In global planning often referred to as strategic planning, the environment where the planning takes place is based on previously collected data, for example the lunar digital elevation model. This implies that a path can be precomputed instead of computing it in real time on a resource limited robot in space. This approach is particularly advantageous in scenarios where real time path computation on resource limited robots in space is impractical. By utilizing previously remotely sensed environmental information, global planning results in the efficient preplanning of routes, mitigating the computational burden during navigation. [14]

Despite its benefits global planning is not without its limitations, for example the lunar digital elevation models have a resolution which is above a meter per pixel [24], which means that there is no complete information available, the planned route may avoid obstacles on the digital elevation model, but when testing the planned route on the lunar surface it may encounter a small crater or a large obstacle on its path that was not seen due to the low resolution of the digital elevation model [24].

Despite having a significant drawback it is still useful to have a general idea of how the complete mission route looks like and then making minor adjustments where problems arise, and to solve that autonomous navigation often also uses a combination of local path planning in conjunction with global planning. In local planning the initial path is planned globally and followed until a problem arises. [14] The idea is to plan a global path with the global planner which outputs a set of points to visit in order to reach a point of interest and use a local planner to avoid the hazards. The following figure demonstrates how a local planner works on the twin Mars exploration rovers

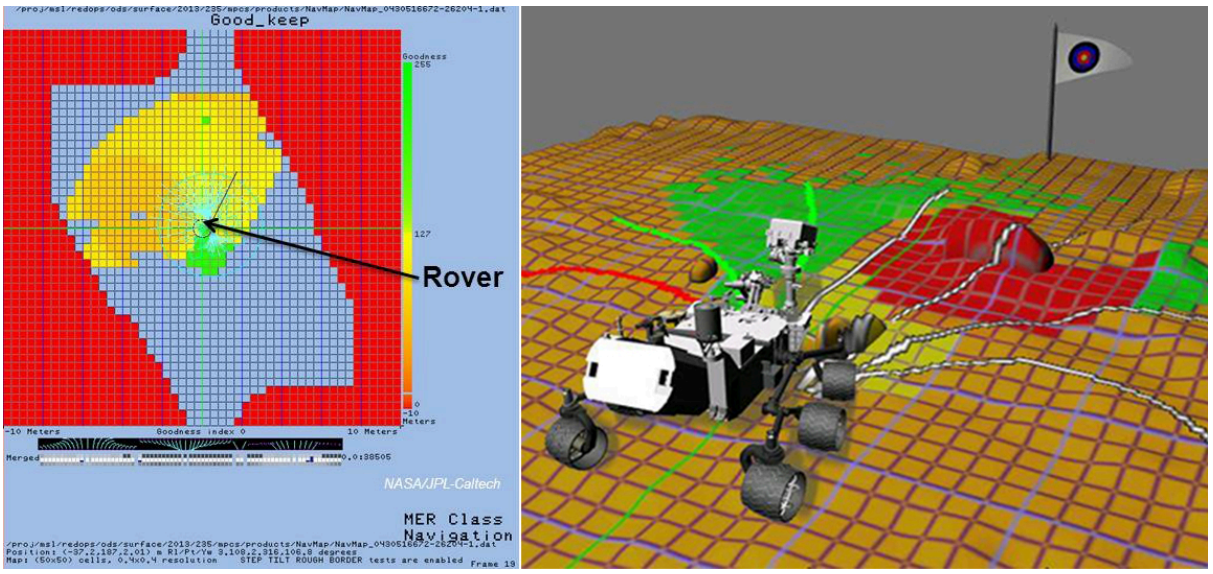


Figure 1. GESTALT choosing the best candidate arc on an elevation map [11]

An example of planetary mission planning is Grid-based Estimation of Surface Traversability Applied to Local Terrain (GESTALT) by NASA which is used on both the twin Mars exploration rovers [8]. The drawback of GESTALT is the cost matrix, it is small since it is an elevation map produced by the stereo vision module, being limited by the camera's range of vision [11]. Additionally the twin Mars exploration rovers used a global path planner to go from point A to point B using the AutoNav system extended with a pathfinding algorithm called Field D* . The following figure shows the algorithm's route.

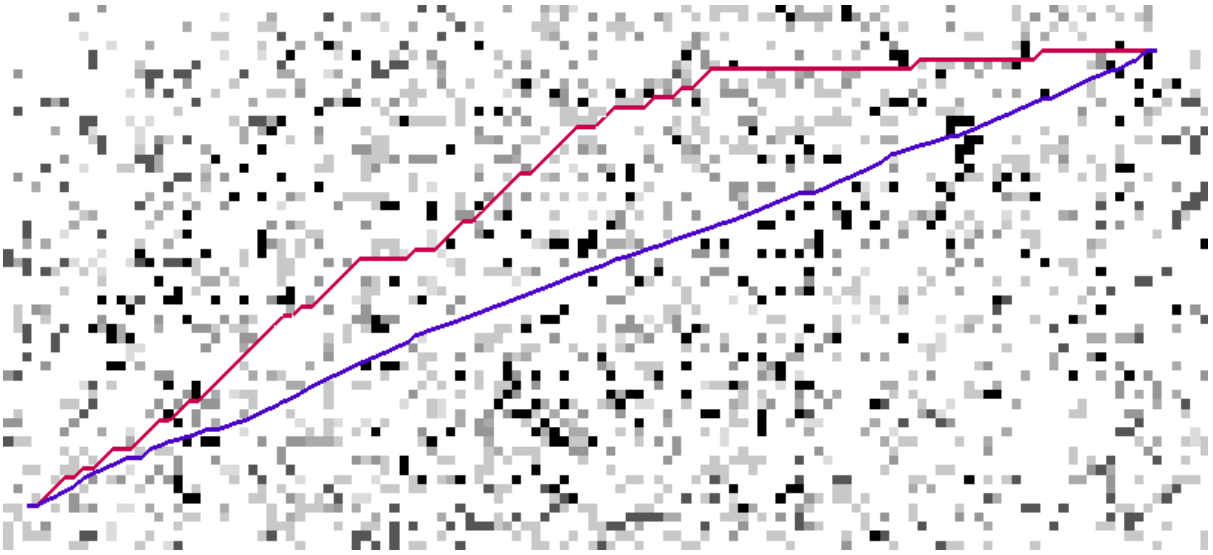


Figure 2. Paths produced by D Lite (red) and Field D* (blue) [12]*

On Figure 2 the traversable area is depicted in white and everything else represents an obstacle. From the Figure 2 it is seen that Field D* produces a straighter and shorter path to the endpoint. The advantages of employing a global planning approach are becoming evident, creating a larger virtual environment that incorporates surface features results in a reduction in path length compared to both local planning strategies and the other pathfinding algorithm. Moreover, expanding the size of this virtually reconstructed environment would further amplify these benefits - being able to plan a larger scale mission.[8]

The Perseverance Mars rover was proposed to use a global planner that uses a grid that is 100 meters wide and long and the grid is centered around the rover where each cell is 1 meter by 1 meter. The cost calculation took into account terrain data in a 2 m radius of the cell and the roughness of the terrain[11, p.13]. Despite using these two strategies in conjunction and using the new autonomous driving algorithm on the Perseverance rover there still arise situations where it is not possible to continue the mission without tele operating the rover since the algorithms themselves are not always able to handle different environmental situations [13].

Currently the majority of mission planning tools available, whether open or proprietary, are primarily focused on space bound missions, mainly dealing with orbital mechanics. These tools are designed to optimize trajectories and orbits of space missions and conduct debris analysis in low Earth orbit[18]. However these tools don’t include mission planning on extraterrestrial surfaces.

The Volatiles Investigating Polar Exploration Rover (VIPER) mission announced by NASA represents an advancement in this area. It is proposed to use the System Health Enabled Real-Time Planning Advisor (SHERPA) which is designed to be the main global mission planning system for the lunar mission [19][20]. Additionally The rover will use TSP with time windows to solve the route planning problem [21]. Given that the VIPER mission is still in the development phase, detailed technical documentation on SHERPA is not yet available. The information currently accessible primarily comes from presentations available through NASA’s Technical Reports Server and their blog, which outline the anticipated functionalities and objectives of VIPER [20][21].

2.3. Vehicle Routing for Mission Planning

For large scale mission planning it is expected that the automated planner presents the user with an efficient route through a number of POI-s. So far the existing solutions in space[11, 12] have used small cost matrices to plan a path between the rover and the destination, while this is an adequate solution to traverse between two points, it falls short taking into account multiple POI-s. The complexity increases and finding the minimal traveling distance between an order of points is computationally more expensive. To solve such a problem there are a few algorithms like the Traveling Salesperson and Vehicle Routing where the prerequisite is a cost matrix similar to the ones the rovers already used.

The main idea of the Vehicle Routing Problem (VRP) is to find a set of routes for vehicles that minimizes travel time between locations, however it is important to note that when there is only one vehicle the VRP reduces to the Traveling Salesperson Problem. [10]

Finding the order of scientifically interesting points to visit while minimizing the total traveling distance is an example of the Traveling salesperson problem (TSP). TSP consists of finding the shortest tour while visiting each city exactly once. This problem is one of the most famous problems in combinatorial optimization, having wide use cases like in very large scale integration placement problems or scheduling, specifically the Vehicle Routing Problem in the current context. [15]

One method of solving TSP is by brute force approach, this however is computationally very expensive, having O(n!) time complexity [26], this would be feasible while $n < 10$ but quickly becomes infeasible to calculate as $n = 15$ would result in over a trillion permutations of cities. Since TSP is computationally very expensive to calculate the exact best route, researchers have come up with heuristics to approximate the optimal route. Such as 2-opt and 3-opt and the Lin and Kernighan heuristic[15]

There are more variations of the VRP to optimize for other constraints, like optimizing for time windows or carrying capacity[2], but this thesis focuses on minimizing travel distance, since accounting for the above constraint examples would be outside the scope of the initial prototyping phase of the planner development and would add complexity, additionally distance traveled can be converted to traversal time under the assumption of fixed traversal speed. [9, 10]

Existing VRP solutions developed for terrestrial use can’t be directly transferred to other celestial bodies. This primarily arises from the absence of data such as having no digital elevation model or needing to account for unique environmental factors such as lunar shadows, which implies modifying the VRP algorithm to suit the needs of extraterrestrial planning.

Commercial solutions employ open and proprietary solutions[9] which are developed specifically for Earth conditions, which can’t be directly transferred to extraterrestrial surfaces due to their unique and different conditions, like not having an atmosphere resulting in dark shadows with no indirect illumination or long lunar day and night cycles. Moreover, these solutions minimize the traveling distance by calculating the cost matrix of the respective terrain which results in a static cost matrix, however it is not sufficient to only calculate the terrain costs on a lunar mission, since the presence of shadows will dictate the departure time if the POI is in shadow, which if left unaccounted for could result in a premature end of a mission.

3. Problem statement

The ideal large scale mission planner for lunar or similar extraterrestrial environments would integrate multiple variables to ensure an optimal mission plan. Such a system would accurately account for varying dynamic shadows, which can impact solar power availability and thermal conditions for rover operations. It would also adapt routes based on terrain features, vehicle specific features such as weight, speed, energy consumption and predefined communication windows essential for maintaining contact with Earth based control stations.

This thesis represents an initial step towards realizing such a mission planning solution, with the aim to develop a proof of concept VRP solver with custom behavior capable of planning routes on the lunar surface. This solver is tailored to plan a route accounting for the dynamic shadows cast by lunar topography which shift over time due to the Moon's rotation. The VRP output should also be as close as possible to an optimal solution.

To solve the problem the problem was divided into two main subproblems:

1. Developing a method to know if a location on the lunar elevation map is in shadow at a given time
2. Develop a VRP solver which can account for shadows when planning a route

4. Requirements

Having established that an ideal mission planner accounts for multiple different parameters, it concludes to one computationally complex task, thus it is necessary to define requirements for the planner to ensure the operations of the planner happen in a timely manner.

4.1. System Requirements

- The VRP planner must be capable of operating effectively within a 5km^2 operating area, this is an internal requirement, ensuring comprehensive coverage for an initial prototype
- The planner should include a stopping behavior when entering shadowed areas, since the shadows pass quicker than the rover is able to move, the behavior still results in a near optimal behavior.
- The algorithm should be efficient: due to the substantial computational demands of large scale mission planning, ensuring timely route generation.
- The DEM must not be downsampled below its original resolution due to the already low resolution of 1,5 meters per pixel.
- User defined points of interest: User must have the ability to specify points of interest that the algorithm plans a route around.
- Route optimality: The algorithm should generate routes that are as close as possible to the optimal route given the computational and data constraints.
- Reasonable calculation timeframe: Route planning should be completed within a timeframe that is practical for near real time operational planning.

4.2. Hardware Limitations

- Memory limitations: The hardware used to support the VRP planner must not exceed 64 GB of ram usage. This limitation offers a comfortable buffer between the expected computational load and other expected system operations of the workstation.
- Computation limitations: The VRP planner must be able to be used on a higher end workstation with a high end GPU and CPU

5. Methods

The implementation involved solving the Vehicle Routing Problem and the creation of a VRP solver capable of accounting for shadow movements over time. Python was selected as the programming language for implementation, favored for its simplicity and efficiency in rapid prototyping compared to lower level languages. The Apollo 15 DEM used in the thesis is accessible on NASA's moon trek web portal [23].

The chosen approach for addressing the VRP planner involved the use of horizon maps and creating a custom VRP planning solution capable of querying a location on the map to know if it is in shadow at a given time. This approach was chosen to enhance the route planning capabilities of the planner. The inclusion of horizon maps allowed for the representation of environmental changes, specifically the movement of shadows which could directly affect the mission outcome. Figure 3 illustrates the high level overview of the implementation:

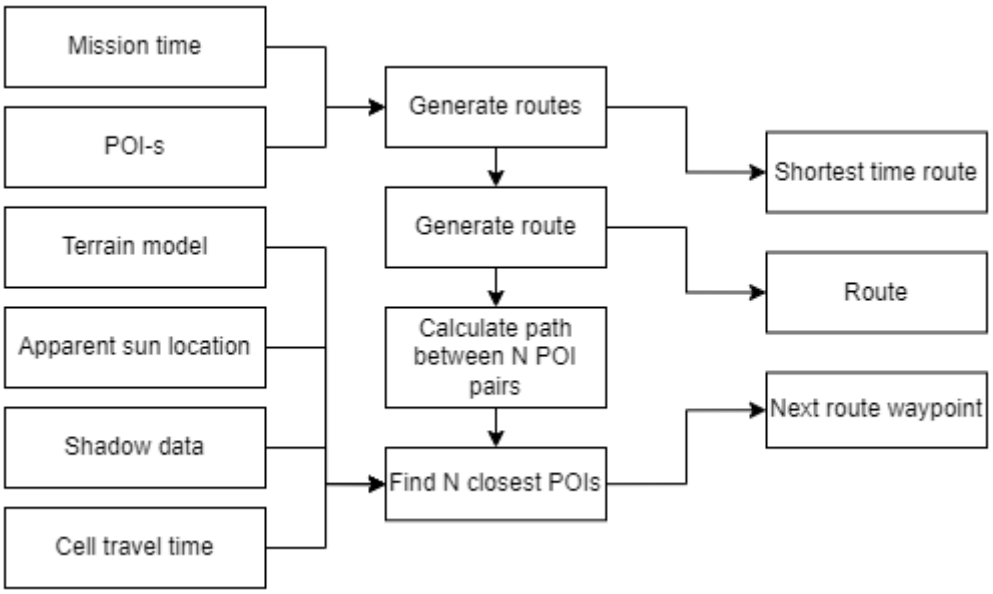


Figure 3. Functional diagram of the VRP solver

The functional diagram illustrated in Figure 3 provides an overview of the logic flow for the VRP solver. The primary objective of the VRP solver is to find the shortest route while accounting for moving dynamic shadows which vary over time. The functional diagram consists of Inputs, functions and outputs:

Inputs:

- Mission time: The time when route planning starts and used to keep track of the overall elapsed mission time when generating each individual route
- POI-s: User specified points of interest that are used to plan the mission route around
- Terrain model: Used to look up shadows for a given location and in acquiring the apparent sun location.
- Shadow data: Used to check if a given location is in shadow at a given Sun's azimuth
- Cell traversal time: A fixed unit of time used as a cost that is added to mission time as the vehicle traverses each cell.

Functions:

- Generate routes: used to generate a user specified amount of routes and chooses the route that takes the least amount of time to complete.
- Generate route: Helper function, generates a single route
- Calculate path between N POI pairs: Helper function, used to calculate a path between POI pairs
- Find N closest POIs: used to find a user specified amount of nearest POIs and randomly returning one of the found POIs, which is the next waypoint in the route.

Outputs:

- Shortest time route: The route with the shortest traversal time and effectively shortest distance since traversal speed is a constant.
- Route: A single mission route, which may or may not be optimal.
- Next route waypoint: Used as the next destination when constructing a route.

5.1. Shadow Lookups

In computer graphics a straightforward method for projecting shadows involves a two pass rendering process. Initially rays are cast from the perspective of the sun, during this the distances traveled by each ray are saved in the z buffer. In the second pass the z buffer is tested from the perspective of the observer. If the stored z buffer value is less than the distance to a specific pixel then that pixel is in shadow [16]. Given

the focus on a sun centric view rather than the player the process can be reduced to a single rendering pass further simplifying the shadow mapping process.

Two ways were identified to check the shadow state of a given location - calculating the ray from the queried location towards the sun and checking whether the ray intersects with the terrain or calculating the angle of the horizon for each location on the operating area and testing it against the sun's altitude.

The implementation involved creating a horizon map by calculating the angle of the horizon for all the given locations on the operating area and then testing the locations against the Sun's altitude above the horizon to determine if they are in shadow. An initial prototype was created prior to the final solution using CUDA acceleration for ray tracing to generate a shadow map offering potentially vastly faster performance generating it, but was scrapped due to the errors in generating the map which can be read more in detail in Appendix 5.

The decision to compute horizon angles for shadow lookup was influenced by the availability of pre-existing computational geospatial tools, specifically WhiteboxTools[3] includes a utility designed to calculate horizon angles thereby eliminating the need for developing the functionality from the bottom up. Additionally the required solar altitude and azimuth data were obtainable using the Skyfield[25] library. Despite the computations being performed on a CPU the process of generating the horizon maps proved to be computationally efficient. A horizon map which is analogous to a DEM contains only the horizon angle corresponding to a specific sun azimuth for each location on the DEM. This approach is illustrated in Figure 4, where an example generated horizon map can be seen.

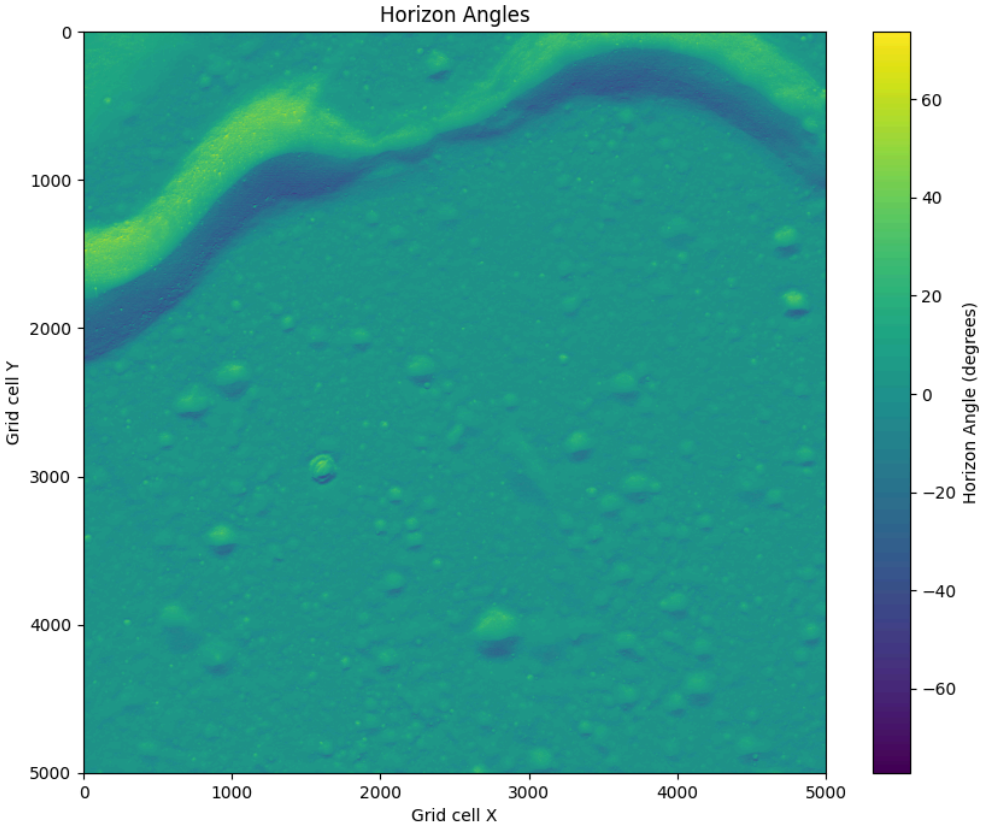


Figure 4. Example calculated horizon angles for a given azimuth angle where grid cell X and Y are the indices of the DEM on their respective axis.

To create a shadow map from an example horizon map as seen on Figure 4 it is necessary to test the horizon angle against the Sun's altitude using the following formula:

$$location\ in\ shadow\ =\ sun\ altitude\ <\ angle\ of\ horizon\ at\ location$$

To verify that the horizon angles were correctly calculated a shadow map was created using the above formula on each location on the operating area of the DEM. The following example can be seen on Figure 5.

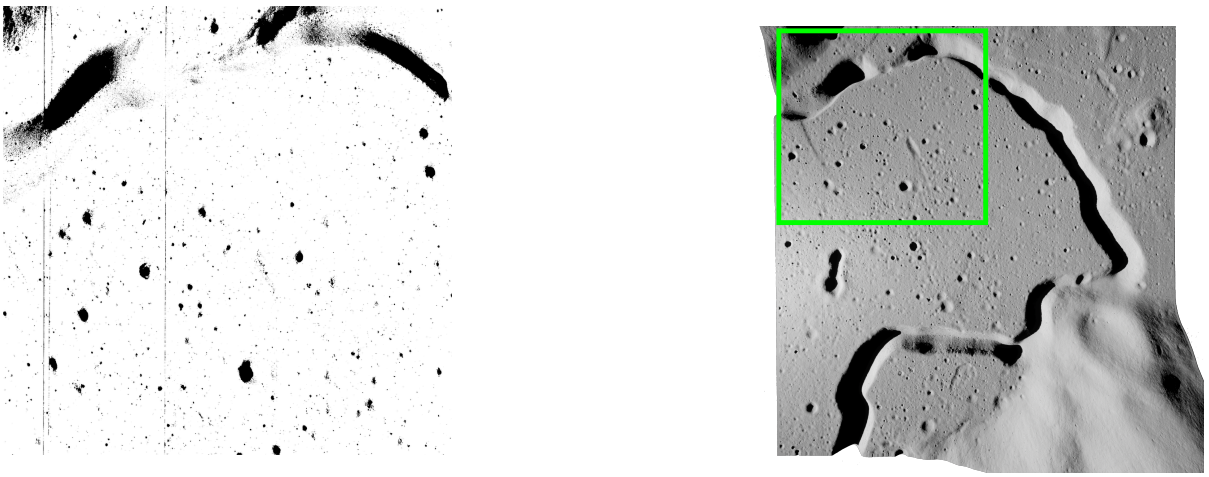


Figure 5. Example created shadow map (left) compared to the same area highlighted in green which is produced in simulation (right) at 2024-02-01_T00:00:00

From Figure 5 the created shadow map is binary, resulting in white representing illumination and black representing no illumination. Comparing the shadow map against the simulation which was created in ULYSSES [22], it can be concluded that the two are quite similar.

The last step in this process is creating all the horizon angles (now referred to as horizon maps) spanning from 0 to 360 degrees with one degree intervals. These individual horizon maps were combined into a single dictionary resulting in a vast lookup table. This resulted in each horizon map occupying approximately 100 MB, with the combined horizon map totaling approximately 35 GB.

5.2. VRP Solver with Dynamic Shadow Lookups

Instead of influencing the costs in a cost matrix with shadow maps, it was chosen to plan the route in the time dimension. This was desirable, because it reduced the dimensions of the problem allowing for a simpler creation of the cost matrix for the VRP algorithm which allowed the representation of environmental changes, specifically shadow movements. Furthermore time can be converted to distance traveled with the assumption of fixed traveling speed and energy spent is directly related to time spent traversing, additionally planning temporally allowed to use time spent in shadow as an additional cost as well as allowing to easily add timestamps to each point in a line between POI pairs allowing for a simpler mission planner output creation.

The first step of the VRP algorithm is to plan a user specified amount of routes and picking the best route from the planned routes. To create the route a function is called to create a random route. The following pseudocode section explains the logic of the function:

Algorithm 1: Building a route randomly

```
Function find_route_random(start, pois, mission_start_time, N=4)
    Initialize route as list containing start
    Initialize final_cost to 0
    Initialize timestamps_of_paths as empty list
    Initialize paths_between_pois as empty list
    Set poi_start_time to mission_start_time
    While length of route is less than length of pois:
        Set not_visited as list of pois not in route
        Call find_random_closest_point with last element of route, not_visited,
poi_start_time, N=1
        Get next_node_index, cost, path_to_poi, timestamps_of_path from result
        Set timestamp_at_arrival to last element of timestamps_of_path
        Append not_visited[next_node_index] to route
        Append path_to_poi to paths_between_pois
        Add cost to final_cost
        Append timestamps_of_path to timestamps_of_paths
        Set poi_start_time to timestamp_at_arrival
    Return route, final_cost, paths_between_pois, timestamps_of_paths
```

To build a single route it was decided to find the closest point that will be used as the next waypoint in the route, it is done by generating all the blue points as seen on figure 6 between all POI pairs, then iterating through the line points doing shadow lookup to calculate the cost of reaching the POI.

Iterating through time allows the use of time required to traverse one grid cell as a cost. If the cell has no restrictions the cost is only the time it takes to traverse one cell. However if the cell is in shadow the cost is the amount of time the cell is in shadow which is expressed as an integer multiple of the time it takes to traverse one cell. This cost is only accumulated if the rover reaches a cell that is in shadow.

To find the closest point the function starts by calculating the sun's altitude and azimuth at the beginning of each iteration and then iterates through the lines and their respective points between POI pairs, getting the location of each point on the

line between the POI pair. It checks if the point is in shadow, if it is not, the index array gets incremented, illustrating that one cell has been traversed and adds a timestamp for that point since it is assumed that the rover would traverse one cell in 5 minutes with a fixed speed.

If the point is in shadow the index does not get incremented and the algorithm keeps incrementing the cost for that point by adding one cell traversal time to it, effectively resulting in the rover stopping in shadow until the shadow passes. Encountering an index error while iterating through the points in a line means that a new POI has been reached and a new line between POI pairs gets iterated through repeating the above process again.

This process runs until a user specified amount of nearest random POI-s has been found, if the POI-s have not been found yet, the mission time is incremented by 5 minutes, effectively illustrating that a line point has been traversed.

Once the POI-s have been found one POI is randomly chosen from the found POI-s and added as the next route destination from the origin, this returns the line between the origin and POI along with its respective timestamps for the points in the line, which is later used to output the best planned route into a json file.

There are two common ways of expressing a line between POI pairs: a line that is capable of being drawn diagonally(without manhattan restriction) or a Manhattan restricted line - resulting in only horizontal and vertical movements to form a line.

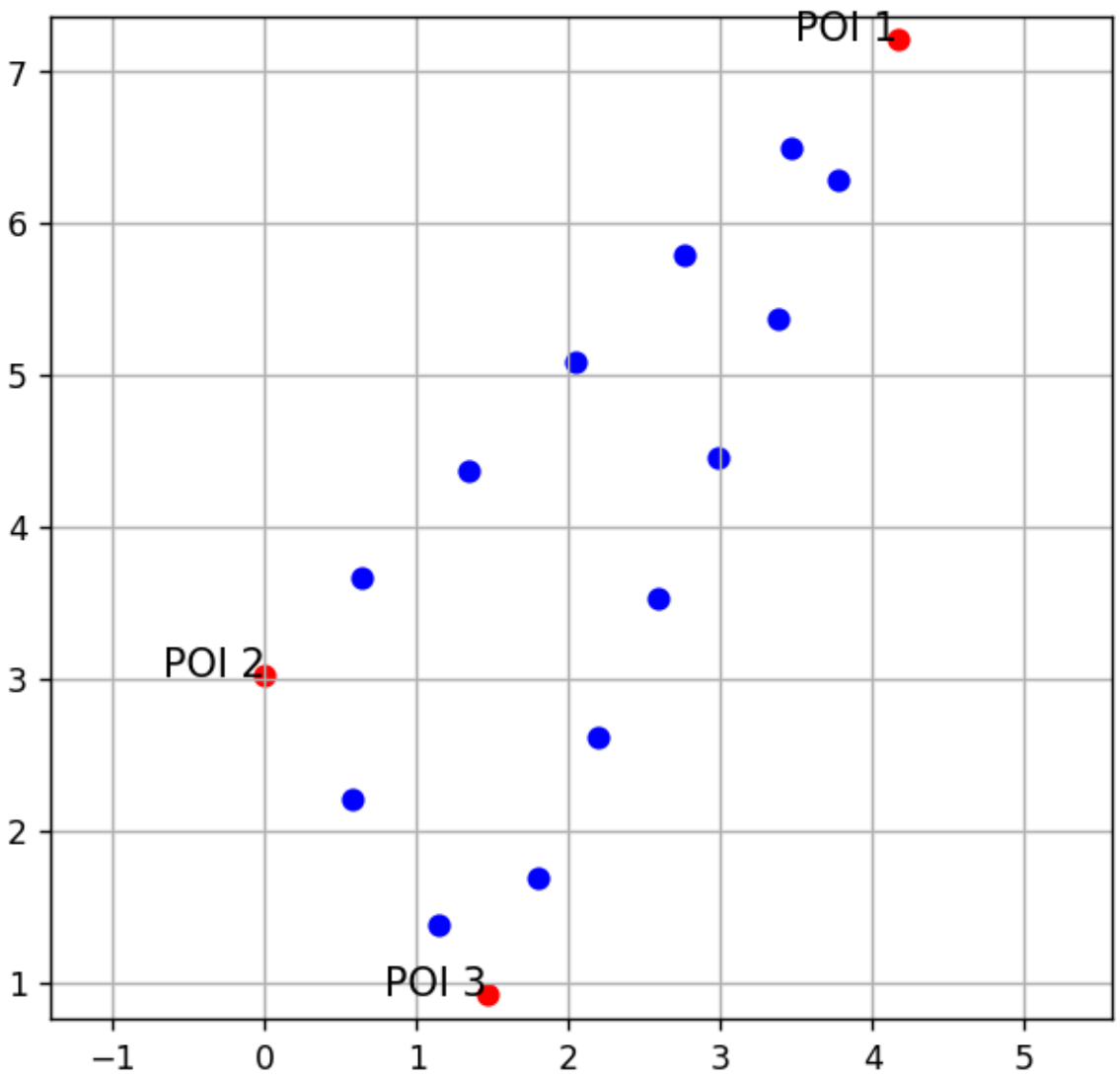


Figure 6. Example of creating equal distance points between POI pairs (blue).

On Figure 6 it is seen that the last blue point between each POI is shorter than the other ones, this is not a cause for concern considering the resolution of the DEM is 1,5 m/px. Getting the position wrong by up to 1,5m was not an issue as it simplified the route planning significantly.

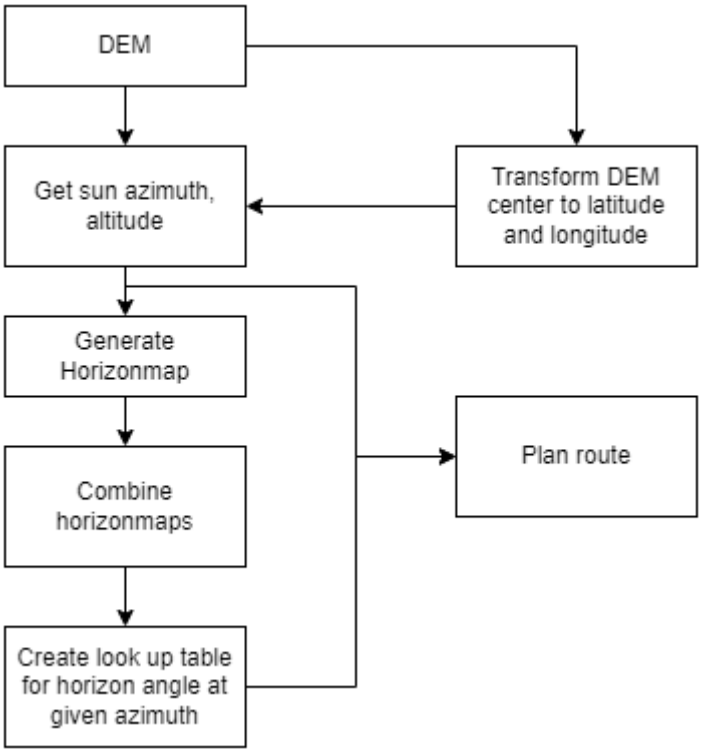


Figure 7. software system diagram

Figure 7 shows the overall system architecture of the VRP solver and the design choices taken to plan a route where:

- **DEM:** is used to get the Sun data, specifically altitude above horizon and azimuth, by transforming the center location of the DEM to latitude and longitude.
- **Sun data:** is used in route planning as well as creating horizon maps for a given azimuth
- **Horizon map:** is used to check if a given location is in shadow for the specified azimuth the horizon map was created for
- **Lookup table:** is the result of the combined horizon maps spanning from 0 to 360 degrees for the azimuth, enabling the checking of if a location is in shadow for any azimuth.
- **Plan route:** uses the sun data and the lookup table to account for varying shadows and plans a route.

6. Results

As a result of the thesis a route planning script was written and the flow diagram of the logic in the script is illustrated in Figure 8.

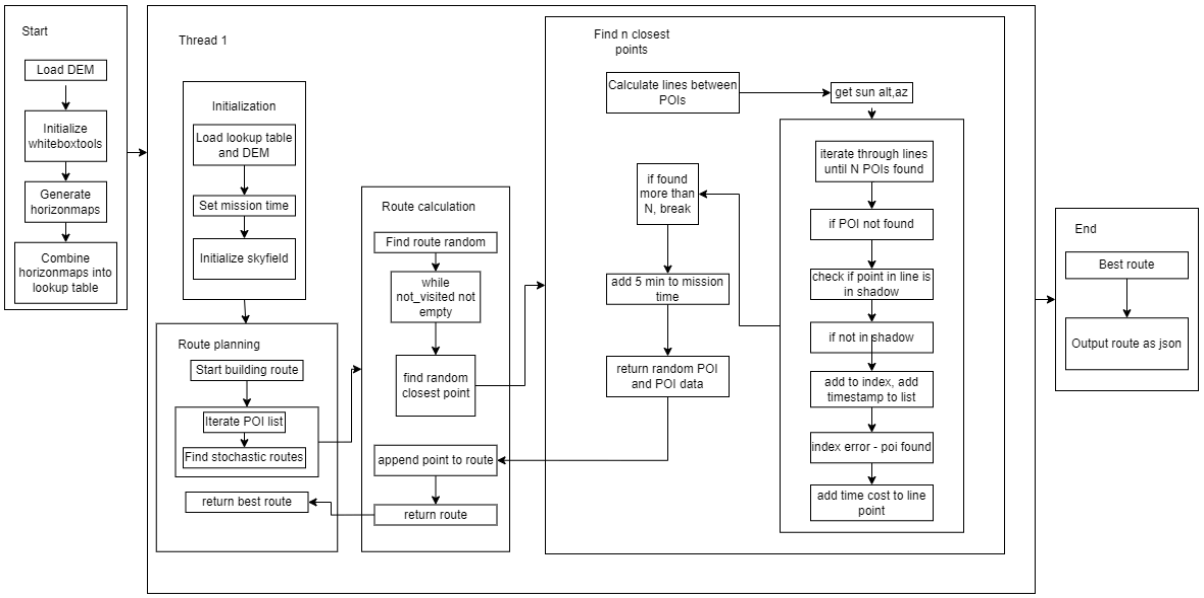


Figure 8. Software logic flow diagram

The route planning script consists of four primary functions:

- **Find stochastic route:** in charge of returning the best route from a set of randomly calculated routes,
- **Find route random:** in charge of building a single route by finding random closest points,
- **Find random closest point:** in charge of finding N nearest points and returning one randomly
- **Calculate lines between POIs:** in charge of generating a path between each POI pair to enable finding the closest point.

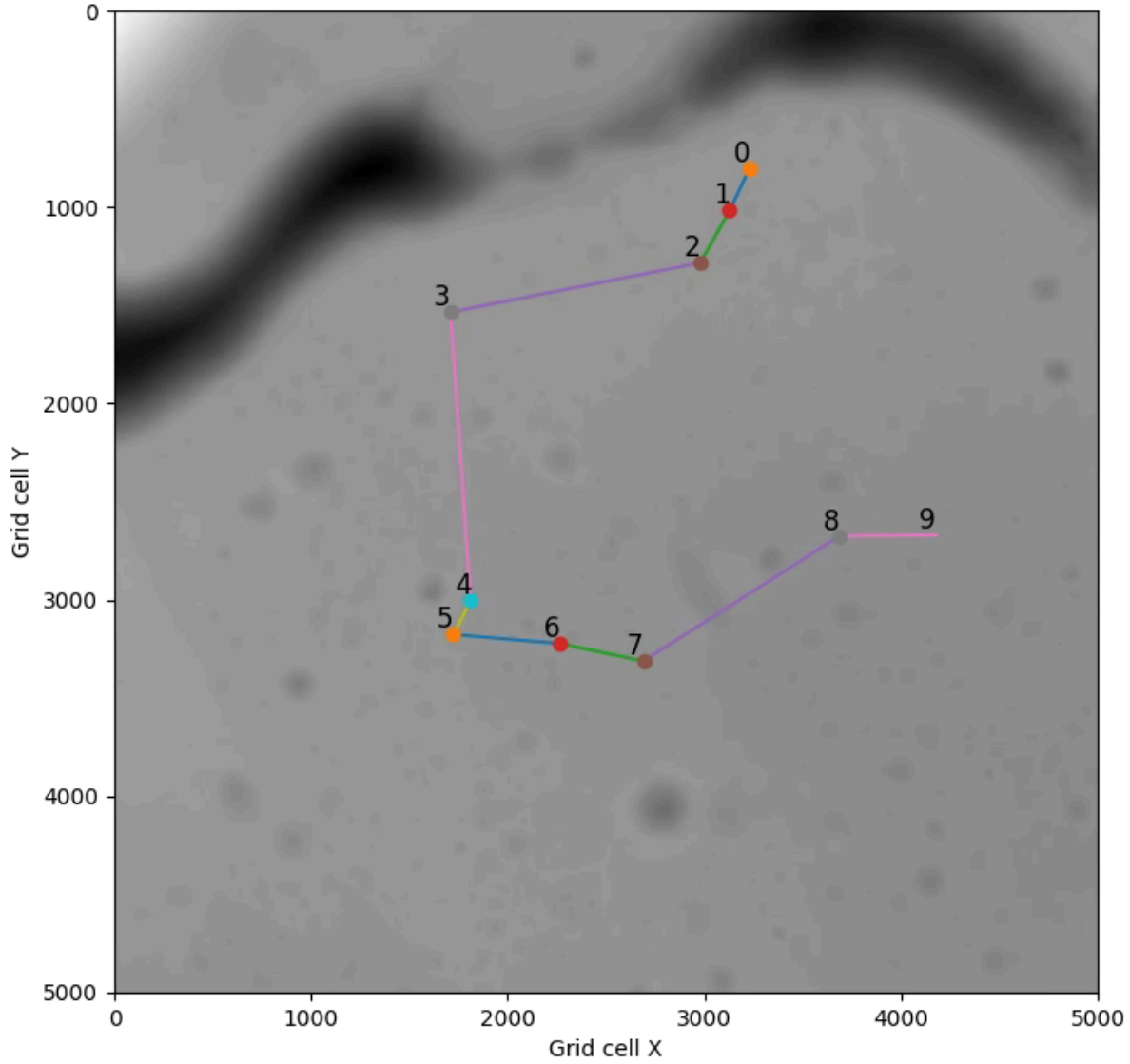


Figure 9. Example calculated route of the algorithm on 10 randomly generated POI-s.

The result depicted on Figure 9 was computed in approximately 168 seconds despite having a comprehensive lookup table of 35 GB loaded into memory on a workstation equipped with an Intel i9-12900k and RTX 3090Ti. It is observed that the algorithm experiences a reduction in processing speed as the number of points increases, or as the spatial separation between these points increases. This expansion increases the number of intermediate points between pairs of POI-s, consequently increasing the time it takes to compute, diminishing the algorithm's efficiency.

The goals of the thesis have been successfully met, the developed VRP solver successfully manages the dynamic shadows that move in time outputting a route that

is optimized for a travel distance on a 5km^2 operational area. Qualitative analysis was conducted through visual assessment by looking at the logical order of the waypoints and if the route was following a sensible path. The performance of the VRP solver is achieved while adhering to the set requirements in the requirements chapter, utilizing less than 64GB of RAM and outputting a route between 10 randomly selected points. The coordinates of the randomly selected points ranged from 750 to 4250 on both the X and Y axes resulting in a 3500 by 3500 operating area which is effectively a 5km^2 operating area when factoring in the pixel resolution of 1,5 meters. Since the task was completed in approximately 168 seconds the time frame used to complete the task can be considered a reasonable amount of time.

7. Discussion

The developed prototype VRP solver effectively manages dynamic shadows that vary over time. It demonstrates potential for preliminary accuracy testing in simulation environments. Furthermore the solver is designed with transferability in mind, it can be directly adapted to other planetary surfaces, provided that the celestial body has a pre-existing DEM and sun altitude and azimuth can be computed with the skyfield API, requiring minimal changes in code.

The primary limitations of the developed VRP solver relate to its computational efficiency and the method of shadow map generation. Currently the VRP algorithm operates on a single thread which constrains its performance. Future optimization can be achieved by running the algorithm across all available processors resulting in an increase in computational efficiency. Further improvements can be made by using a more efficient VRP algorithm, swapping the line algorithm for a pathfinding algorithm or reducing the amount of nodes on the DEM, one of such methods is representing the DEM as a quadtree. An initial prototype was developed to reduce the node count in the DEM but was not used in the final solution due to it being out of scope, which can be read about in appendix 2 and 3.

Initially the VRP solver used lines generated with manhattan restriction to achieve time cost being incremented consistently, but was decided against later and used straight lines instead, more information can be found in appendix 4 on the design choice.

Additionally the shadow map generation could potentially be sped up using CUDA acceleration using similar approaches as employed in video game graphics for shadow rendering. However the binary nature of the shadow map, while adequate for lunar context due to the absence of atmosphere and thus shadows are dark, it may not suffice for other celestial bodies with atmospheres. On such bodies a gradient of shadow intensity could provide valuable risk assessment data for mission planning, indicating the safety of navigating through shadows, which could be an essential consideration.

To further improve the capabilities of the prototype VRP planner the suggested improvements are taking into account terrain features like surface roughness when planning a route. As well as considering the vehicle parameters when traversing a route to optimize for energy efficiency, lastly considering communication windows

into the route, planning a route where there's minimal downtime between communication links.

Future works should involve the replacement of the line algorithm with a pathfinding algorithm and the compression of the DEM to increase the computational efficiency when running the pathfinding algorithm and later the VRP algorithm as well as implementing the inclusion of terrain features into the VRP planner to have a more accurate mission route. Lastly the current VRP planner plans for one vehicle, which results in solving the TSP. To plan for a fleet of vehicles a different VRP algorithm should be used.

8. Conclusion

As a result of this thesis a successful VRP solver prototype was developed, effectively managing dynamic lunar shadows that vary over time while adhering to the requirements stated earlier. The solver's ability to take into account time dependent lunar shadows and its potential for further accuracy testing in simulated environments demonstrate its utility. Furthermore the prototype's design for transferability, dependent on the availability of a celestial body's DEM and solar data using the skyfield API make it desirable for other planetary exploration contexts.

Despite these benefits the solver faces limitations in computational efficiency due to its single threaded nature. Future enhancements should include restructuring the algorithm to run on all available processors or using a more efficient VRP algorithm, which could significantly increase the algorithm's processing speed. Secondly the line algorithm could be replaced with a pathfinding algorithm like A* resulting in a more optimal path between POI pairs.

Additionally the method of shadow map generation, while sufficient for a demonstration and on an atmosphereless environment, may require changes for use on other celestial bodies. Rendering the shadows using CUDA acceleration could also speed up the shadow map generation. Further augmentation to the shadow map generation would prove beneficial for celestial bodies with atmospheres where shadow intensity gradients would provide valuable data for risk assessment on route planning or general mission safety. This would involve no longer using a binary shadow map and borrowing techniques used in video game shadow rendering to generate more accurate shadow maps.

Lastly, since this is an early prototype it is far from an ideal mission planner and there are many features that would need to be added to get the prototype closer to an ideal mission planner. The most important additions would be: Swapping the line algorithm for a pathfinding algorithm, accounting for terrain features, using a more efficient VRP algorithm, accounting for vehicle parameters, reducing the search space without losing important terrain data to increase computational efficiency.

9. Bibliography

[1] Common European Research Classification Scheme (CERCS) Teadusvaldkondade ja- erialade klassifikaator https://www.etis.ee/Portal/Classifiers/Index/26?

[2] Google. (2023, January 16). *Vehicle Routing | OR-Tools*. Google for Developers. Retrieved April 8, 2024, from https://developers.google.com/optimization/routing

[3] Lindsay, J. (2017, July 2). *Geomorphometric analysis - WhiteboxTools User Manual*. Whitebox Geospatial. Retrieved April 10, 2024, from https://www.whiteboxgeo.com/manual/wbt_book/available_tools/geomorphometric_analysis.html#HorizonAngle

[4] Pajusalu, M., et al. (2022). “Large Scale Mobility on the Moon by Transferring Terrestrial Autonomy Capabilities” In 73rd International Astronautical Congress 2022

[5] Lawrence, P. (2022, August 2). *Track moving shadows on the surface of the Moon*. Sky at Night magazine. Retrieved May 8, 2024, from https://www.skyatnightmagazine.com/advice/skills/moving-shadows-moon

[6] Hosseini, S. (2021, August 2). *NASA Study Highlights Importance of Surface Shadows in Moon Water Puzzle*. NASA. Retrieved May 8, 2024, from https://www.nasa.gov/solar-system/moon/nasa-study-highlights-importance-of-surface-shadows-in-moon-water-puzzle/

[7] Hoover, R., & Bowman, A. (2017, July 26). *The Dark Side of the Crater: How Light Looks Different on the Moon and What NASA Is Doing About It*. NASA. Retrieved May 8, 2024, from https://www.nasa.gov/centers-and-facilities/ames/the-dark-side-of-the-crater-how-light-looks-different-on-the-moon-and-what-nasa-is-doing-about-it/

[8] J. Carsten, A. Rankin, D. Ferguson and A. Stentz, "Global Path Planning on Board the Mars Exploration Rovers," 2007 IEEE Aerospace Conference, Big Sky, MT, USA, 2007, pp. 1-11, doi: 10.1109/AERO.2007.352683.

[9] Kuo, M. (2024, January 9). *Solving the Vehicle Routing Problem (2024)*. Routific. Retrieved May 8, 2024, from https://www.routific.com/blog/what-is-the-vehicle-routing-problem

[10] *Vehicle Routing Problem | OR-Tools*. (2024, March 28). Google for Developers. Retrieved May 8, 2024, from https://developers.google.com/optimization/routing/vrp

[11] Matheron, G. (2016). *Designing the path planning algorithm for Mars2020*. Guillaume Matheron. Retrieved May 8, 2024, from https://guillaumematheron.fr/ul/report.pdf

[12] Ferguson, D., Stentz, A. (2007). Field D*: An Interpolation-Based Path Planner and Replanner. In: Thrun, S., Brooks, R., Durrant-Whyte, H. (eds) Robotics Research. Springer Tracts in Advanced Robotics, vol 28. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-48113-3_22

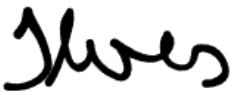
[13] Daftry, S., Abcouwer, N., Del Sesto, T., Venkatraman, S., Song, J., Igel, L., ... & Ono, M. (2022). Mlnav: Learning to safely navigate on martian terrains. IEEE Robotics and Automation Letters, 7(2), 5461-5468.

[14] Hui-Zhong Zhuang, Shu-Xin Du and Tie-Jun Wu, "Real-Time Path Planning for Mobile Robots," 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 2005, pp. 526-531, doi: 10.1109/ICMLC.2005.1527001.

[15] Laporte, G. (2010). A Concise Guide to the Traveling Salesman Problem. *The Journal of the Operational Research Society*, *61*(1), 35–40. http://www.jstor.org/stable/40540226
[16] *LearnOpenGL - Shadow Mapping*. (n.d.). Learn OpenGL. Retrieved May 10, 2024, from https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping
[17] Bowman, A. (2021, February 24). *NASA's Next Lunar Rover Progresses Toward 2023 Launch*. NASA. Retrieved May 10, 2024, from https://www.nasa.gov/centers-and-facilities/ames/nasas-next-lunar-rover-progresses-toward-2023-launch/
[18] Caldwell, S., & Bowman, A. (n.d.). *Space Mission Design Tools*. NASA. Retrieved May 12, 2024, from https://www.nasa.gov/smallsat-institute/space-mission-design-tools/
[19] NASA. (n.d.). *VIPER Lunar Operations*. NASA Science. Retrieved May 12, 2024, from https://science.nasa.gov/mission/viper/lunar-operations/
[20] Balaban, E. (2024, May 6). SHERPA (System Health Enabled Real-time Planning Advisor). Zenodo. https://doi.org/10.5281/zenodo.11118153
[21] Shirley, M., & Balaban, E. (2022). *An Overview of Mission Planning for the VIPER Rover*. Retrieved May 12, 2024, from https://ntrs.nasa.gov/citations/20220008301
[22] Teras, H., et al (2022) “ULYSSES - A State of the Art Sandbox Simulator for Planetary Surfaces” In 73rd International Astronautical Congress 2022
[23] NASA. (n.d.). *Moon Trek*. Solar System Treks. Retrieved May 14, 2024, from https://trek.nasa.gov/moon
[24] USGS Astrogeology Science Center. LRO NAC DEM Apollo 15 26N004E 150cmp, 2011.
[25] *Skyfield — documentation*. (n.d.). Rhodes Mill. Retrieved May 14, 2024, from https://rhodesmill.org/skyfield/
[26] Jain, S. (n.d.). *Traveling Salesman Problem (TSP) Implementation*. GeeksforGeeks. Retrieved May 16, 2024, from https://www.geeksforgeeks.org/traveling-salesman-problem-tsp-implementation/

10. Acknowledgements

I would like to thank my supervisors Rando Avarmaa, Quazi Saimoon Islam and Hans Teras for their guidance throughout my thesis



11. List of Appendices

11.1. Appendix 1: Downsampling the DEM

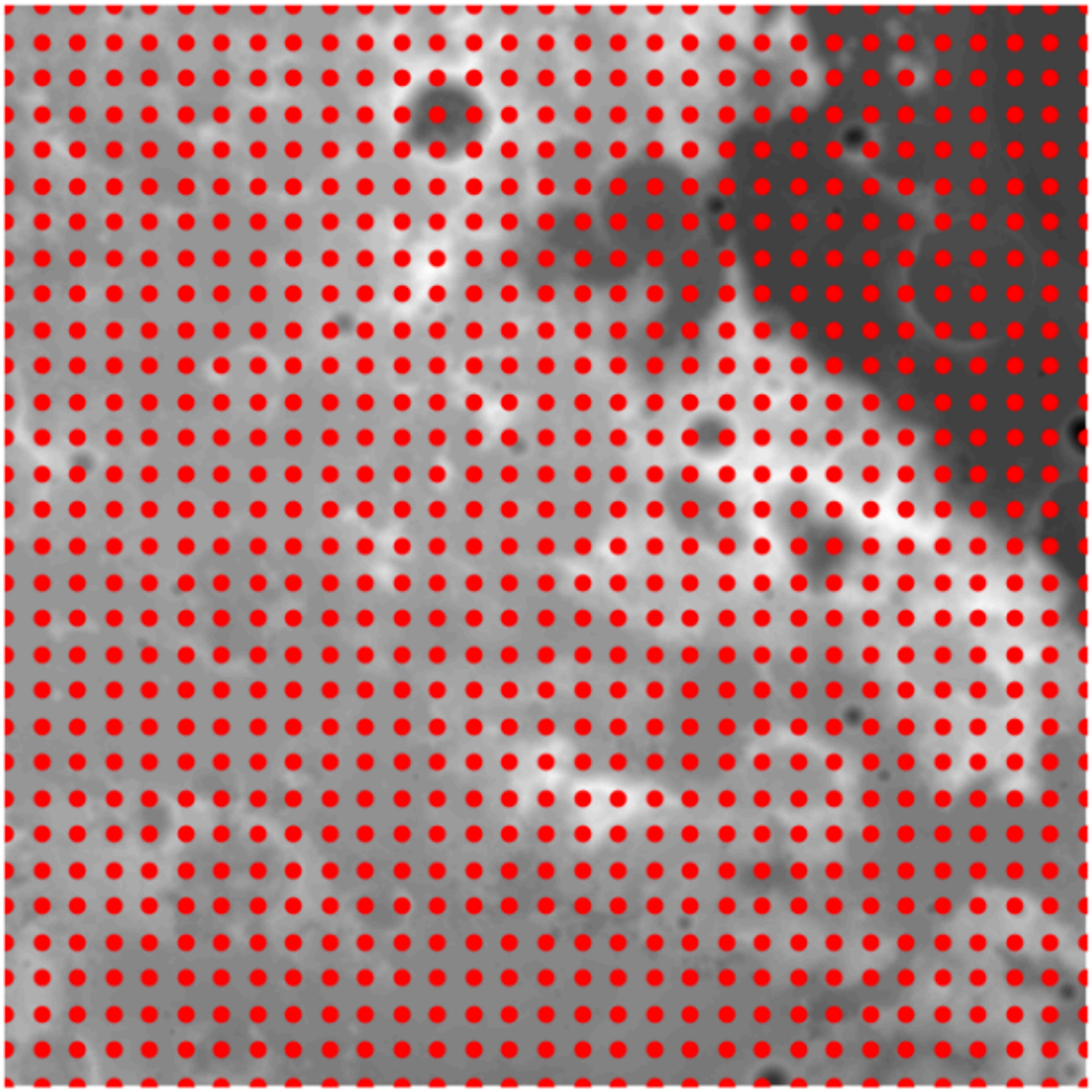


Figure 10. Downsampled nodes (red) overlaid on the 5000 by 5000 DEM.

The initial prototype involved the downsampling of the DEM and creating a graph from the resulting nodes and the reduction of the downsampling iteratively to zero, to analyze the algorithmic efficiency on a vast amount of nodes. An example of the downsampling process is illustrated on Figure 10. The reduction of the downsampling to zero results in the DEM having a node for each pixel. Due to the substantial size of the DEM (5000x5000) this implication would result in 25 million

nodes connected with their nearest neighbors, resulting in an ineffective search space to perform TSP in a timely manner. Due to the ineffective search space this approach did not end up in the final prototype.

11.2. Appendix 2: Optimizing the DEM

Having a large number of nodes in a graph resulted in an ineffective search space as illustrated in appendix 1, to overcome this an alternative approach was considered. To reduce the search space on the DEM it was necessary to compress it in a manner where it retains critical information like sudden elevation changes while losing less important information like similar elevation around nearby grid cells. One method of compressing the DEM as described is to use quadtrees. Each grid cell gets inserted into the quadtree and only the elevation value of the cell is stored in the quadtree structure. The splitting of the quadtree results in higher resolution areas as the splitting condition was chosen to be the threshold of absolute difference between elevations in a given quadrant. This effectively results in large quadrants where elevation values are similar and small quadrants where elevation changes quickly.

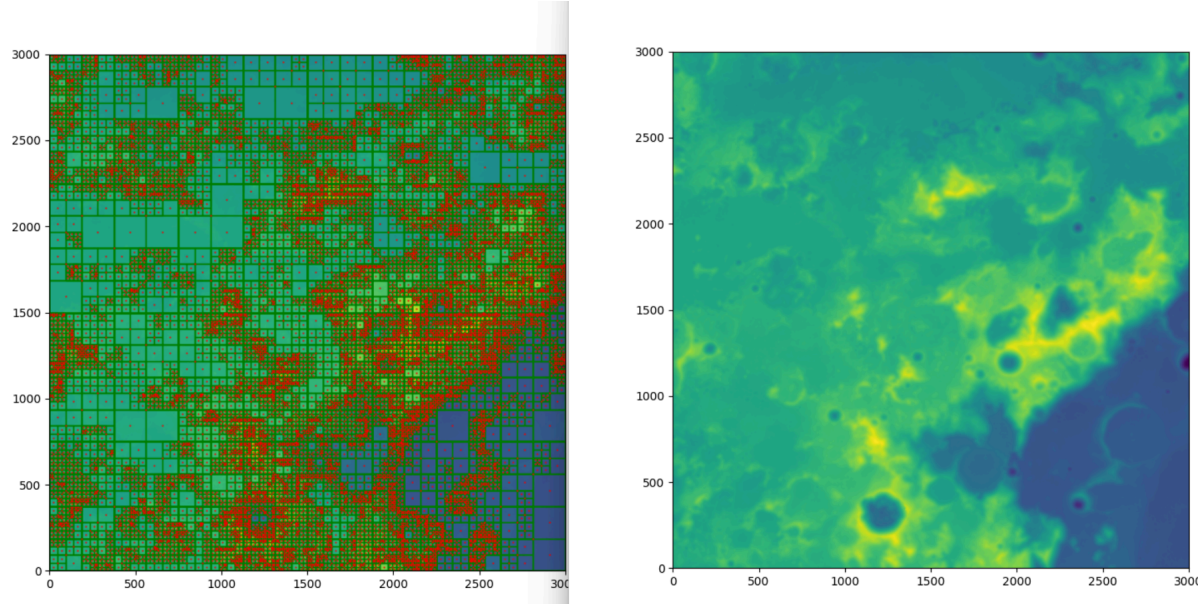


Figure 11. Left - the quadtrees (green squares) and nodes at the center of the quadtrees (red) right - the elevation map for comparison

The above logic is illustrated on Figure 11 where the splitting threshold is 10 meters. From Figure 11 we can conclude that using the quadtree does result in a more sparse DEM while retaining important information, specifically sudden elevation

changes. This results in an optimized search space as the amount of nodes has been reduced.

However this prototype did not end up in the final prototype as a result of simplifying the grid traversal process to straight lines. Using the above optimization with straight lines resulted in difficulties in keeping account of time between traversed points since each point was no longer a fixed distance apart from its neighbors. The straight line simplification however allowed for a less complex solution and an ability to keep track of time more easily since each point could be spaced apart evenly.

11.3. Appendix 3: Graph creation

After the creation of the quadtree in appendix 2, to use it as a search space it was necessary to create a graph from the quadtree. One method of doing so is to use Delaunay triangulation, this was chosen for its ease of implementation and desirable output of the method resulting in mostly connecting only the nearest neighbors over further neighbors which is illustrated in Figure 12.

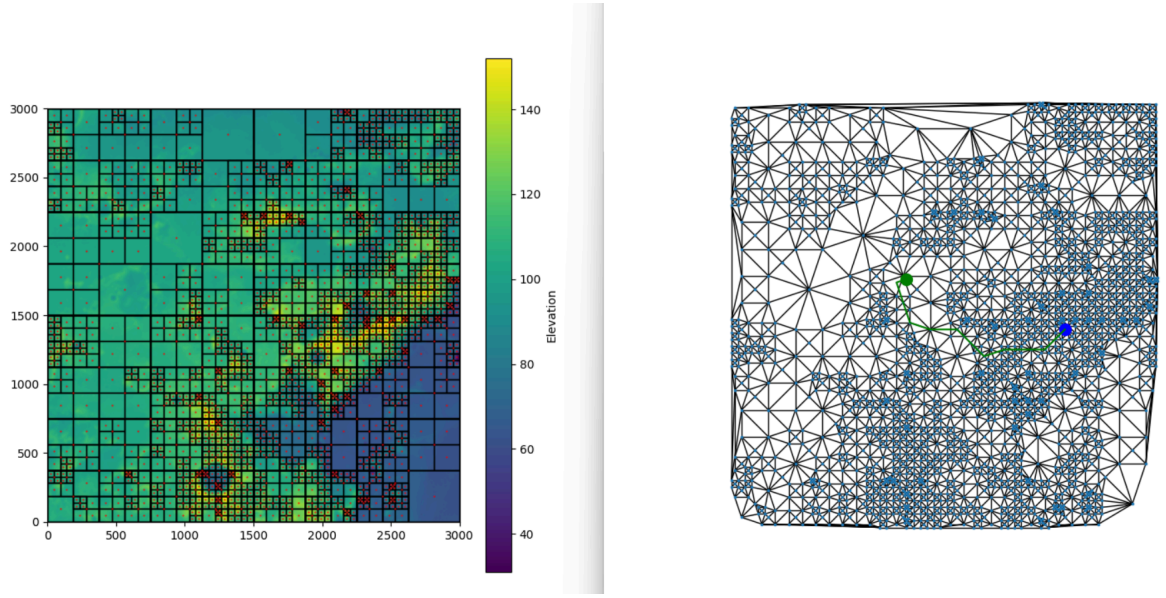


Figure 12. The quadtree (left), the graph created with delaunay triangulation (right)

From Figure 12 it can be concluded that using Delaunay triangulation resulted in the described graph, where most of the nearest neighbors are connected with edges and some with further away nodes.

To use this graph and perform TSP on the graph it was necessary to create a subgraph from the created graph. This is necessary since the graph itself does not

contain edge weight information between nodes that are further away than its nearest neighbors.

The creation of the subgraph enables solving the TSP as the edge between each node would represent the sum of multiple edges on the original graph if the user wanted to plan a route around POI-s that are further apart than one edge.

The creation of the subgraph resulted in the use of the A* pathfinding algorithm provided by NetworkX library to find the sum of edge weights in the original graph between nodes and reduce it to a single edge. The following process is illustrated on Figure 12, where the green and blue points are nodes and the green path between them is the sum of the edge weights which is used as a cost for a single edge in the subgraph.

However this was not used in the final prototype due to the straight line simplification discussed in appendix 2.

11.4. Appendix 4: Line drawing algorithm

Due to the low resolution of the DEM an initial assumption was made - the path calculation between POI pairs will result in a straight line between them. The initial prototype used line drawing with manhattan restriction which results in a line that is composed of strictly vertical and horizontal movements. This was desirable as it did not violate the fixed grid cell traversal time assumption.

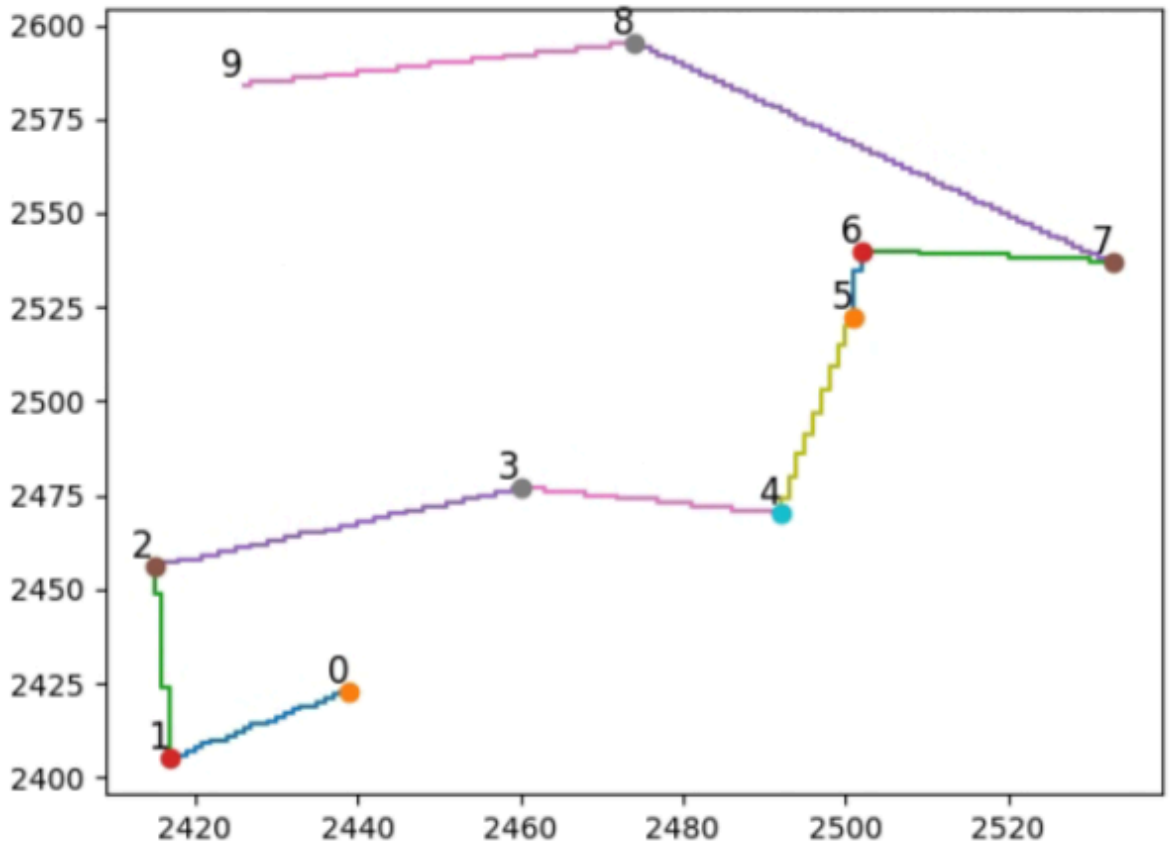


Figure 13. Manhattan restriction lines drawn between POI pairs.

As illustrated on Figure 13 it is seen that the POI pairs have manhattan restriction lines drawn between them. The reason for using the manhattan restriction was to avoid time tracking inconsistencies, if a line without manhattan restriction contained points that are diagonally next to each other it would no longer result in a fixed traversal time, instead traveling the diagonal grid cell would take square root 2 times longer and it would complicate the process of timekeeping, requiring to keep track of when diagonal traversal happened as well. However this implementation didn't end up in the final prototype as the author wanted to have a path between POI-s that deviated less to the endpoint and thus straight lines were used in the final implementation.

11.5. Appendix 5: Shadow Map generation

The initial implementation to create a shadow map involved calculating the rays from the queried point towards the sun and checking if the ray intersected with the terrain along the ray. This implementation was computationally inefficient, with the large shape of the dem it was necessary for the ray to be at least as long to test against all the locations which resulted in a lot of calculations for each location on the dem. The first implementation iteratively increased the ray length by a fixed amount and tested if the ray elevation was higher than the current cell elevation. An example can be seen on Figure 14.

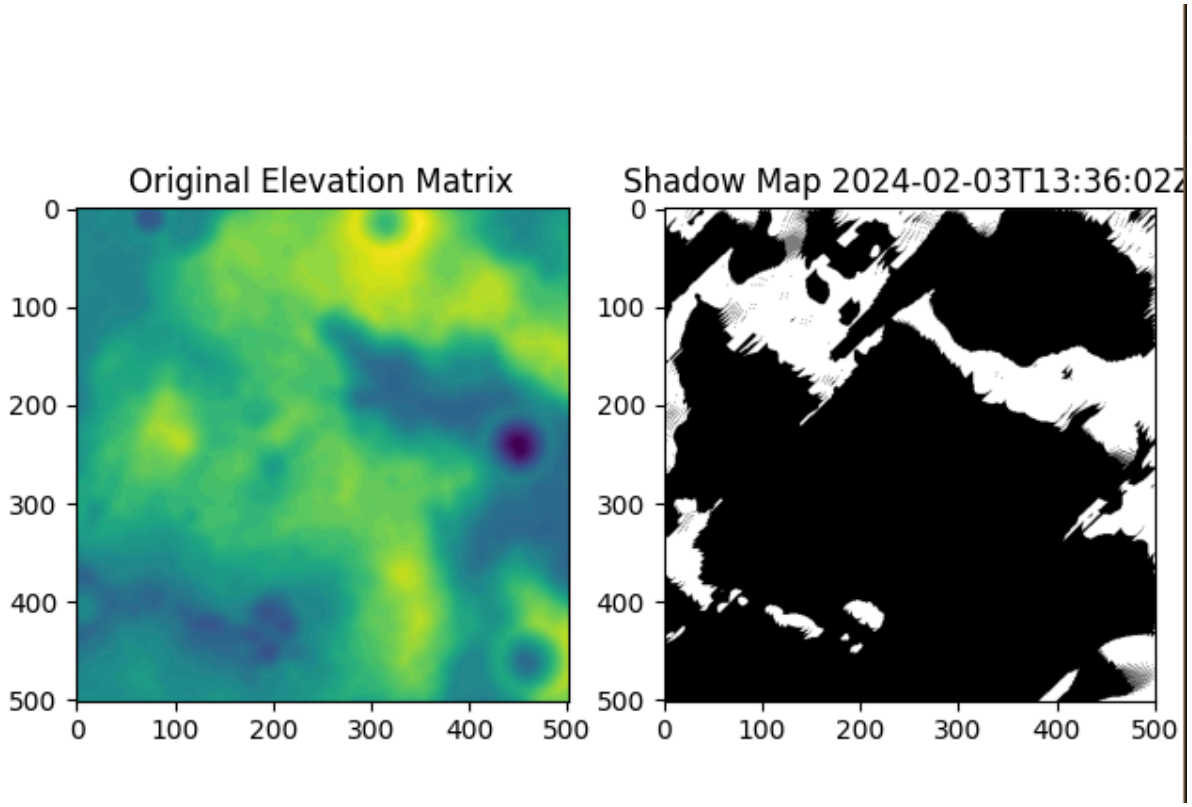


Figure 14. The created shadow map from the elevation matrix (right)

As illustrated on Figure 14 the created shadow map is not correctly created and is limited by its surface area due to the computational load required to create one shadow map. To overcome this issue CUDA acceleration was used with Python. This proved to be highly effective, enabling the creation of higher surface area shadow maps in a timely manner. Despite the performance improvements it resulted in a new problem which is illustrated by the following figure.

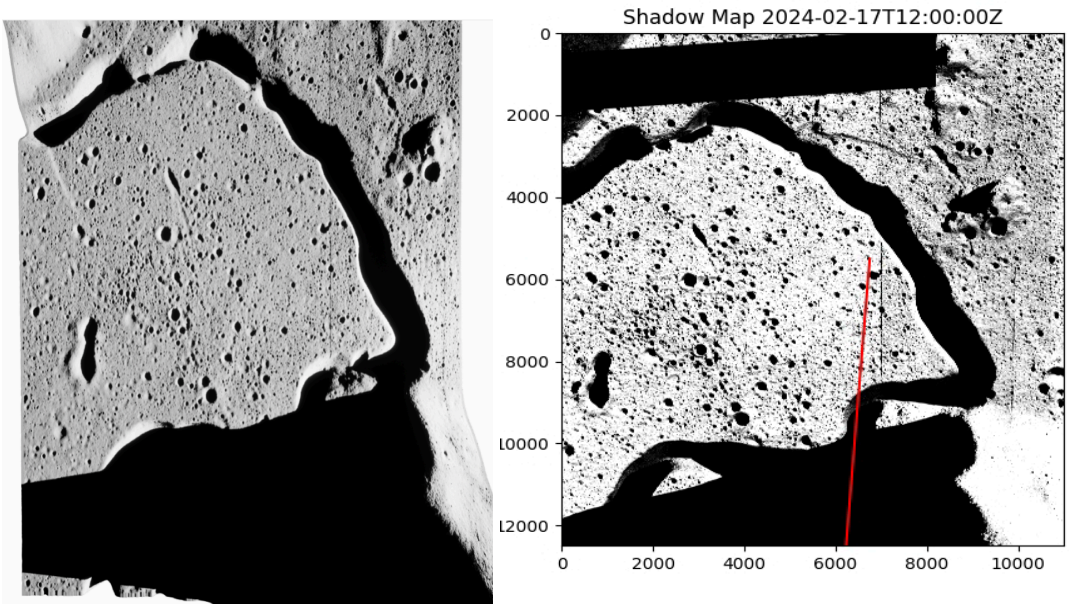


Figure 15. Shadow Map simulation from unreal engine at the same time (left) The created shadow map using CUDA acceleration (right) (Not for scale)

From Figure 15 the long black rectangle and the black top left corner are the described problems. The projected angle of the shadows was slightly off. The rectangle at the top of Figure 15 represents a no data area on the DEM and can be ignored. However the problem really became apparent when the shadow map was calculated further in time which can be seen on Figure 16.

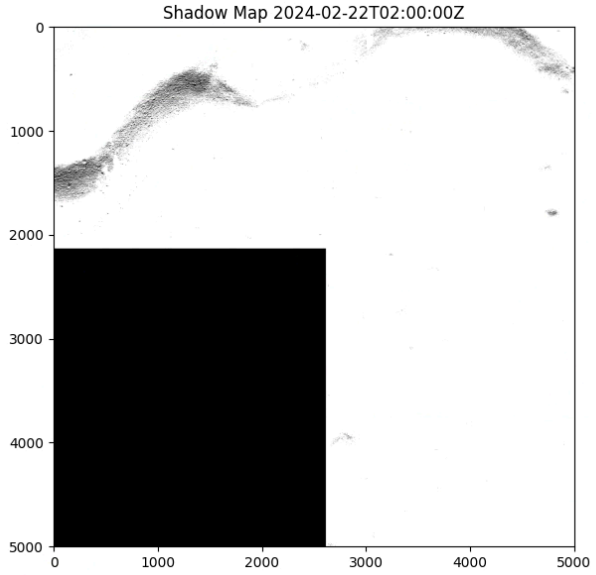


Figure 16. Shadowmap created further in time using CUDA acceleration.

As illustrated on Figure 16 it is seen that the shadow map is calculated incorrectly, represented by a substantial no illumination square during the lunar day, this square moves along the DEM the further in time it is calculated. This was not the expected

result considering that 2024 22nd February is a lunar day, Implying the black square of no illumination should be white.

The implementation was not used in the final prototype due to the complexity of writing code in CUDA and the resulting problems encountered when creating a shadow map, as a result the author later found a utility to create shadow maps in the final prototype from horizon angles using a tool provided by WhiteBoxTools.

12. License

I, Uku Ilves

1. grant the University of Tartu a free permit (non-exclusive license) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, my thesis

Towards Large-Scale Automated Mission Planning for Robotic Lunar Operations - Early Prototyping
supervised by Quazi Saimoon Islam, Rando Avarmaa and Hans Teras

2. I grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons license CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in points 1 and 2.

4. I confirm that granting the non-exclusive license does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation

Uku Ilves
07.05.2024