Implementation of An Efficient Coverage Method for Irregularly Shaped Terrains with Lunar Terrain

Samara Holmes
Institute for Experiential Robotics, Khoury College of Computer Science
Northeastern University
Boston, Massachusetts
holmes.sam@northeastern.edu

***Abstract*—Navigating the Moon's surface is challenging due to its irregular shapes and features such as craters. This paper introduces a method to conduct coverage path planning on lunar terrains. It proposes an effective method to detect craters and combines this with a path planning strategy for complete coverage. The result is a low-redundancy path planner for robots traversing between craters on the Moon.**

I. Introduction

Coverage path planning (CPP) presents numerous challenges, especially when applied to environments with complex conditions. Irregular terrain, varying surface features, and the presence of obstacles often make it difficult to create paths that ensure complete and efficient coverage. Additionally, the trade-off between total coverage and efficiency must be managed to avoid excessive redundancy while still guaranteeing comprehensive coverage.

Developing robust detection models that leverage image processing techniques or neural networks can enable high-resolution identification of lunar craters from satellite images. These models not only improve obstacle detection, but also help define navigable areas for exploration.

Complementing this, path planning algorithms aim to address the coverage problem by creating systematic and efficient routes for robots to drive.

II. Problem Description

Navigating lunar surfaces is a key challenge for advancing robotic exploration. The Moon's irregular terrain, marked by craters of varying sizes, poses significant obstacles to achieving efficient and accurate coverage.

This paper addresses two primary challenges:

1) Lunar Crater Detection: Developing a reliable method to detect and identify craters on the Moon's surface using high-resolution TIFF images [1].
2) Robotic Path Planning for Coverage: "An ideal coverage path should enable a robot to thoroughly cover the area of interest while minimizing revisits to previously covered zones" [2]. This system aims to implement the methods seen in [2] to create efficient paths for robots while ensuring complete coverage.

III. Related Work

A. Crater Detection

Crater detection is a pivotal task in planetary science with applications ranging from terrain relative navigation (TRN), hazard detection, and spacecraft navigation. Traditional approaches to crater detection relied on manual annotation and image processing techniques such as edge detection. Recent advances in deep learning have improved the field of crater detection. These advancements include using convolutional neural networks (CNN), You-Only-Look-Once (YOLO) object detection [3], and U-Net models [4]. Hybrid approaches compare images with the Moon's digital elevation map (DEM) data [5]-[6].

In [7], they created a TensorFlow-based pipeline, DeepMoon, to train a CNN to detect craters. This was a method primarily used for post-processing lunar images and identifying craters.

Similarly, LunaNet has been proposed as a system that uses CNN to detect craters [8]. It differs from [7] in that LunaNet can run in real-time and compares the identifications with a crater database. In LunaNet, once the craters are identified, crater identifications are compared with a database of known crater locations, which can then be used to generate landmarks for TRN. Fig. 1 shows the image processing that LunaNet uses to predict craters. It's input is a grayscale image, then a prediction mask is outputted by the neural network, a binary threshold is applied to the prediction, the contours are fitted with ellipses to form circular crater detections, then the contours are applied to the original grayscale image.

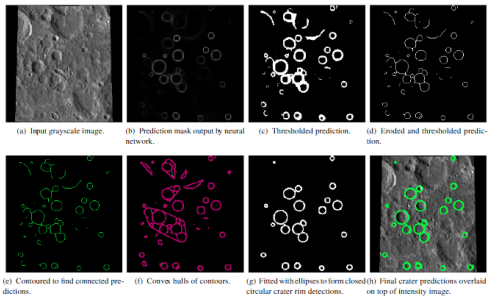
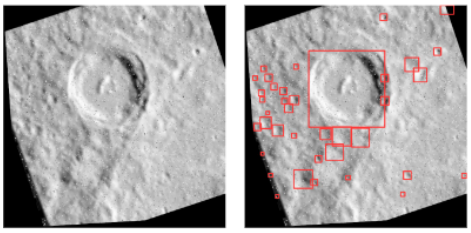


Fig. 1: Processing operations performed on neural network prediction image to obtain final discrete crater detections output by LunaNet [8]

In [6], a workflow is proposed to perform automated crater detection. For the workflow, the lunar DEM data is divided into smaller regions to speed up the CNN crater detection process. For post-processing, regions are merged back together while removing duplicate detections using a non-maximal suppression (NMS) algorithm. In this workflow, several object detection architectures are compared using varying CNN models.

YOLOv8 was trained using high-resolution lunar surface images [3]. The model was fed several surface images with locations of craters and non-craters labeled.



(a) Actual Image (b) Predicted Image

Fig. 2: YOLOv8 example prediction from [3]

[9] uses the Segment Anything Model (SAM) from META AI to obtain segmentation masks from lunar surface images. The presented crater detection algorithm (CDA) does a similar process to LunaNet, where a mask is fitted from the input image and the craters are then drawn and applied (see Fig. 3. This CDA works well on top-down lunar images and lunar images at different angles. It uses a Canny filter for the edge detection on the masks.

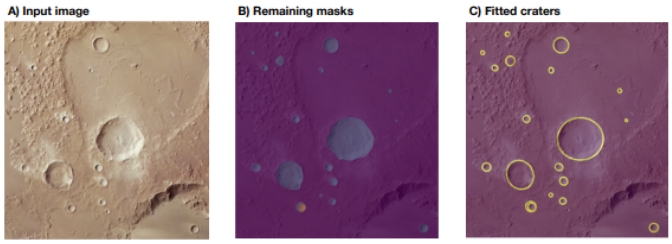


Fig. 3: Applying the proposed CDA to the top left area of the Mars Express HRSC natural colour image of Orcus Patera shown in Figure 1. A) column is the input image, B) are the remaining masks, and C) are the final fitted craters [9]

Ultimately, plenty of research has been conducted relating to lunar crater detection, however, it has not been applied to the CPP problem. In this implementation, we plan to use a similar workflow to [6], where we take the DEM data and generate random crops for detection. From there, we will use edge detection, DepthAnythingV2 [10], and a YOLO model to conduct crater detection.

B. Coverage Path Planning

Path planning for coverage is a critical robotics problem focusing on finding a path for a robot to follow while ensuring complete coverage. The problem has several applications in diverse fields including, search-and-rescue operations, agricultural monitoring [6], and robot vacuum cleaning. Traditional methods use grid-based formatting and heuristic algorithms. Recent advancements utilize divide-and-conquer strategies, which allow multiple robots to coordinate coverage tasks. For our problem, we will be looking at single robot coverage path planning, however, extensions will be discussed later.

In [11], a grid-based system (see Fig. 4) is used to solve the multi-robot-persistent-coverage problem (MRPCP) defined in [12], while factoring in stochastic failures. In 4, we can see multiple different paths for each of the eight robots, ensuring that there is persistent coverage between all the robots.

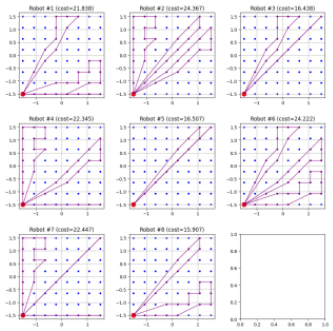


Fig. 4: Grid-based solver example for a mutli-robot system from [11]

In [13], the optimal coverage problem is solved in two parts: first using an offline solver for the construction of the Boustrophedon Cellular Decomposition (BCD) and the Reeb Graph (RB); second using an online coverage solver that uses the sequence of edges to guide the coverage from one cell to the next. Fig. 5 illustrates the algorithm from [13] being applied to different test area environments with varying obstacles.

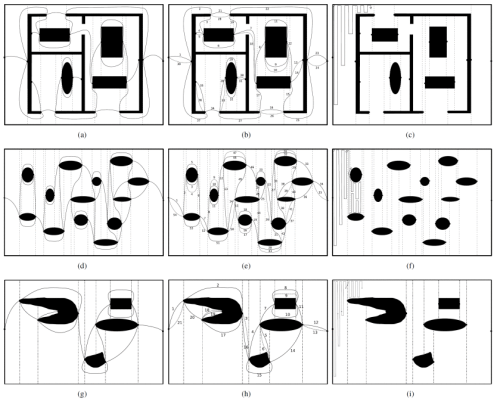


Fig. 5: Coverage comparison from [13]. (a) An environment representative of an office space, with the Reeb graph (RG, solid lines) and the Boustrophedon Cellular Decomposition (BCD, dashed lines). (b) the Euler tour for (a) with the order in which the cells are going to be visited marked. Doubled RG edges marked as dense dashed lines. (c) A snapshot of coverage. (d) An open space with sparse obstacles in line, representing an open minefield with RG and BCD. (e) The Euler tour for (d). (f) Coverage of (d). (g) An arbitrary environment populated with convex and concave obstacles with RG and BCD. (h) The Euler tour for (g). (i) Coverage of (g).

[14] presents a comparison of different coverage methods and a review of the differences between conducting CPP for Unmanned Aerial Vehicles (UAV) and Unmanned Ground Vehicles (UGV). It also presents strategies for energy saving in CPP algorithms.

In summary, current methods in robotic path planning struggle to balance efficiency, complete coverage, and minimal redundancy. For this paper, we plan to take characteristics from each paper, including implementing a grid-based system and a two-part solver to minimize redundancy and path length. While current methods in crater detection are reliable, they haven't been used in combination with a robotic path planning scenario.

IV. Methodology

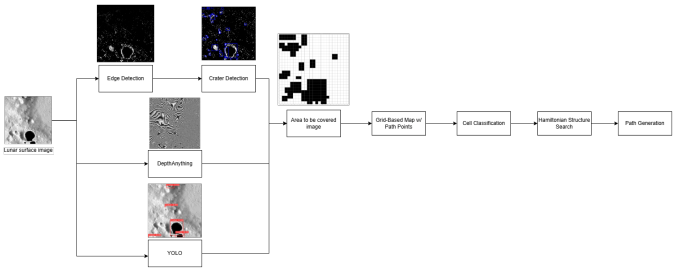


Fig. 6: Process Diagram

A. Crater Detection

We begin by loading a TIFF image containing lunar crater data [1]. From there, we can randomly select crops from this image and conduct processing to detect the craters. We propose three methods to conduct crater detection processing.

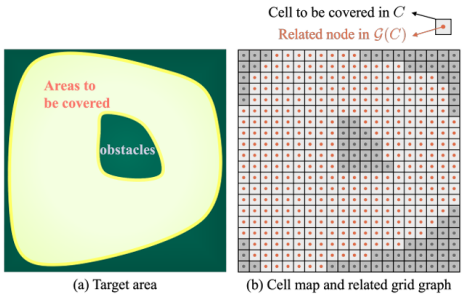
1) Edge Detection: For edge detection (Fig. 11-12), we perform the following steps:

i Apply a Sobel filter along the X and Y axes
ii Use the gradient magnitude to normalize the image
iii Apply a binary threshold
iv Use a morphological filter to clean the image up
v Use skimage.measure.regionprops to identify crater regions

2) Relative Depth: Using DepthAnythingV2, we can obtain an image that looks similar to a topographical map to highlight detections (Fig. 13). DepthAnythingV2 only provides relative depth information, so the detections may not be as accurate as edge detection.

3) YOLO Detection: We take a pre-trained YOLO model from [15]. From the YOLO model, we receive detections and their locations/size on the image (Fig. 15). We can apply these detections to grid mapping.

All of these methods will result in an image that highlights regions where the craters are likely to be located. The goal is to produce outputs that clearly label craters as 'obstacles' and identify the remaining 'areas to cover.' This image can then be fed into the path planning algorithm.



(a) Target area (b) Cell map and related grid graph

Fig. 7: Example from [2] displaying image-defined coverage map (left) and its grid-based coverage map

B. Coverage Path Planning

For coverage path planning, we will mainly refer to a grid-based coverage algorithm [2], which proposes a model to minimize duplication of the paths while ensuring complete coverage. This model is based on approximate cell decomposition (ACD).

This algorithm identifies Hamiltonian cycles in which each node in a graph is visited once without repetition. The authors in [2] identify that their algorithm is inspired by Spanning Tree Coverage (STC), however, they modify this algorithm because STC is known for redundancy issues.

We begin by creating a classification of cells that need to be covered. This is based on our initial image created from the crater detection. There are four types of cells:

- Cells exempt from coverage (craters)
- Subsidiary cells - all occupied cells and cells with multiple paths
- Branch cells - the cells left over after excluding subsidiary and trunk cells
- Trunk cells - cells capable of forming complete mega-cells

To achieve complete coverage and minimal repetition simultaneously, it is necessary to find a sequence of path point $\{r_0, r_1, \ldots, r_N\}$ such that,

$$\begin{aligned} &\min \|\{r_0, r_1, \ldots, r_N\}\|, \\ &\text{s.t. } r_i \in C,\ \forall i = 0, 1, \ldots, N \end{aligned} \tag{1}$$

$$\{r_0, r_1, \ldots, r_N\} \supseteq \rho \tag{2}$$

where $\|r_0, r_1, \ldots, r_N\|$ represents the length of the path formed by sequentially connecting each path point. (1) represents the goal of minimizing the total length of the coverage path. (2) specifies that the path should pass through every path point.

C. Algorithm

Input: $\mathcal{T}, \mathcal{S}, \mathcal{H}, C$
Output: A low-repetitive path that traverses all path points
$k \leftarrow$ number of Hamiltonian structures in $\mathcal{H}$
$\mathcal{E} \leftarrow \emptyset$; // rules of tree formation
do
  for $j \leftarrow 1$ **to** k **do**
    if $\mathcal{B}'_j \notin \mathcal{T}$ ***and*** $l(\mathcal{B}'_j)$ *fits (9)* **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{B}'_j$
      $E_j \leftarrow$ extra rules due to the link
while $\mathcal{T}$ *has been augmented with new parts*;
$ST \leftarrow$ an MST formed by PRIM in the mega-cell grid graph, prioritizing all rules in $\mathcal{E}$;
while *any* E_x *in* $\mathcal{E}$ *mandates using cells that are already utilized in the MST.* **do**
  $\mathcal{T} \leftarrow \mathcal{T} - \mathcal{B}'_x$
  $\mathcal{S} \leftarrow$ cells in $\mathcal{B}'_x$
$HC_t \leftarrow$ circular path in $\mathcal{T}$ obtained through ST
$HC_t \leftarrow$ connect all Hamiltonian paths complying with PRIM to HC_t according to (8)
$L \leftarrow \mathcal{L}(HC_t)$
$m \leftarrow$ number of remained path points
for $n \leftarrow 1$ **to** m **do**
  $\Delta(L, r_{s_n}) \leftarrow$ argmin $(\|\Delta(L, r_{s_n})\| - \|L\|)$
  $L \leftarrow \Delta(L, r_{s_n})$
return L

Fig. 8: Algorithm used in [2]

Using the above algorithm presented in [2], we identify the following steps:

1) Create a grid mapping of the target area using square cells
2) Conduct cell classification, which will be discussed more below
3) Identify Hamiltonian cycles in the trunk graph using the STC algorithm
4) Identify Hamiltonian paths in the branch graph
5) Combine the cycles and the paths

For the cell classification, we begin with our grid map. We can mark the craters detected as 'cells exempt from coverage'. The subsidiary cells are cells that contain multiple path points and are marked light-blue. We look for cells that can generate mega-cells (a 2 x 2 group of cells needing to be covered), and classify those as trunk cells. The remaining cells that aren't trunk or subsidiary cells are branch cells.

Fig. 9 is a representation of the navigation process. To identify Hamiltonian cycles, the STC algorithm uses mega-cells as nodes to construct the minimum spanning tree (MST). Then, a path can be generated to navigate around the MST. Depth first search is used to form the spanning tree.

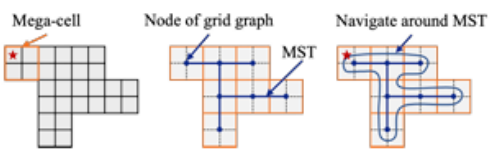


Fig. 9: Graphic representation of the coverage path generated by the STC algorithm from [2]

The primary task is to sequentially identify several types of graph structures within the branch graph, in the order of rings, 2 x n $(n \leq 2)$ grid graphs, and vertex pairs, ensuring no vertex is reused.

Fig. 10 shows how the Hamiltonian cycle and Hamiltonian path can be combined, thus extending the original Hamiltonian cycle. It's important to note that according to [2], "In cases where a Hamiltonian cycle is not possible, we resort to the most basic Hamiltonian path structure, which comprises pairs of adjacent vertices."

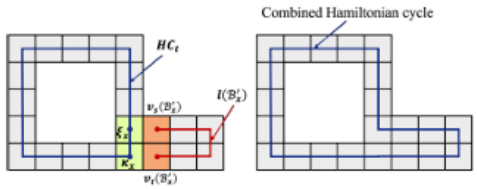


Fig. 10: Illustration of how to extend a Hamiltonian cycle in a grid graph from [2]

To validate the path planning algorithm, we will run numerous Python simulations.

V. Results

A. Crater Detection

In Fig. 11, we present the randomly generated crop taken from [1]. The edge detection and binary thresholding resulted in the image on the right in the figure.

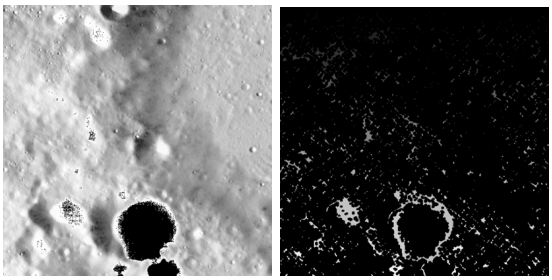


Fig. 11: Original TIFF crop (left) and binary threshold (right)

Using the edge detection image, we were able to detect sufficiently large regions and mark those as craters (Fig. 12). Using the crater detections, we convert this to a grid map for later use in cell classification.

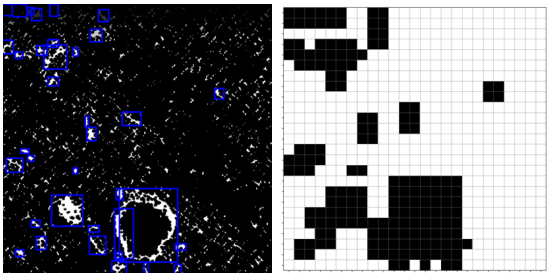


Fig. 12: Crater detection (left) and grid map (right)

We also performed other types of crater detection experimentation, including generating a DepthAnythingV2 result (Fig. 13), and generating a result from a YOLO model (Fig. 15).

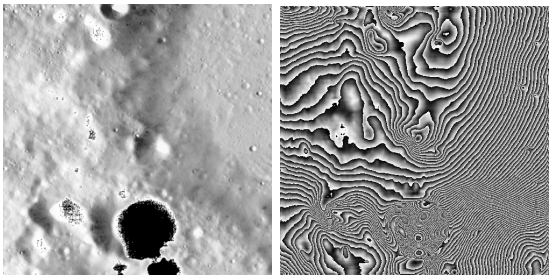


Fig. 13: Original TIFF crop (left) and DepthAnythingV2 result (right)

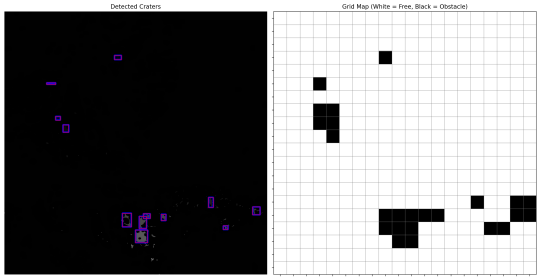


Fig. 14: DepthAnythingV2 grid result

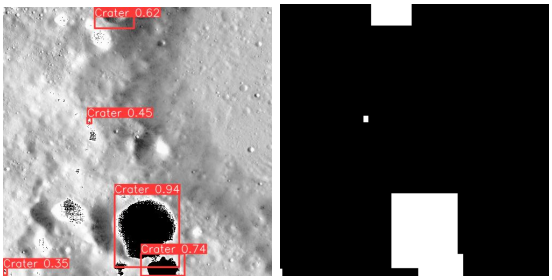


Fig. 15: Crater detection with YOLO model

We can move to coverage path planning using the grid generated in Fig. 12.

B. Coverage Path Planning

To begin the coverage path planning workflow, we begin by doing cell classification. Fig. 16 is an example of cell classification run on the pre-generated grid map described in [2]. Fig. 17 shows our results for cell classification.

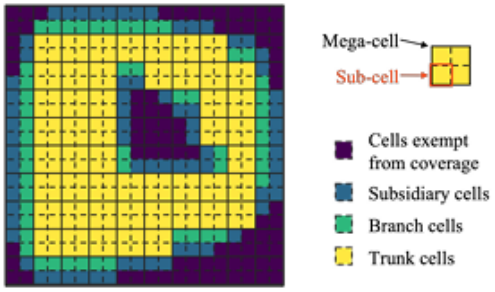


Fig. 16: Cell classification from [2]

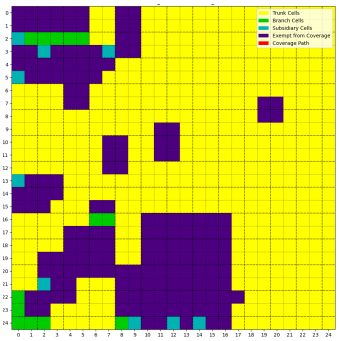


Fig. 17: Cell classification on crater detection grid map

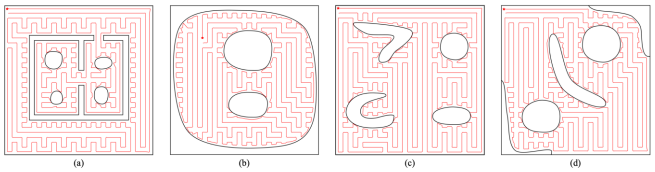


Fig. 18: Example coverage paths generated by the planner in [2]

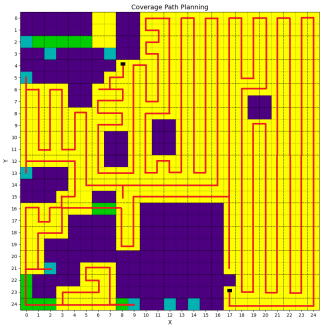


Fig. 19: Coverage path planning on lunar crater grid map

In our simulation, we create a 25m x 25m grid. The number of cells exempt from coverage is 170 (27.2%). The number of cells that are inaccessible to the robot is 16. The number of cells covered is 439. We determine that the maximum amount of coverage that the robot can do is 96.48%, which is 439 out of the 455 cells. The path length is 460 meters for a 625m^2 grid.

In Fig. 18, each test area is 6m x 6m. For test areas a-d, the planner's path length is 158.83, 127.17, 160.31, and 145.63 meters, respectively for a 36m^2 area.

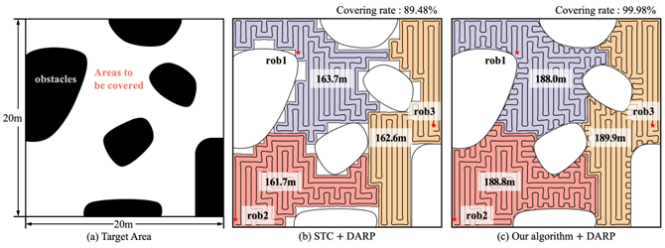


Fig. 20: Multi-robot coverage paths from [2]

VI. Discussion

A. Key Findings

We found that using the 3 different crater detection methods (edge detection, relative depth, and YOLO) generated 3 significantly different grid maps. We found that the grid map created using Sobel edge detection provided the easiest lunar crater coverage path planning. The DepthAnythingV2 grid result provided too many sparse crater detections, while the YOLO model provided less detections.

With the coverage path planner, we were able to generate a path with 96.48% coverage. There are cells where there is repetition, but it is limited. Something that we observed that will be discussed in limitations, is that there were cells that were inaccessible to the robot for coverage.

In the Appendix below, we display other crater detection results. Visually, we can see that the smaller craters are detected in 23. We can also see that in images such as 27, too many craters were detected.

B. Limitations

It is difficult to compare the resulting paths (see Fig. 18 used in [2] and our implementation because the test areas are generated differently. Not only are the test areas different, but they are also of different sizes. In [2], these use four 6m x 6m test areas and one 20m x 20m that is used for a multi-robot system test.

It would be interesting to expand this to higher resolution paths. The resolution of the grid that we test on is 25 x 25, but if this were increased, the obstacles would be smoother and the paths would be longer.

There are also different metrics used for each test area. In our implementation, we only generated one coverage map, where we obtained the path length and the coverage percentage. However, in [2], path length is used only for the 6m x 6m test areas. While, coverage rate and path length are used for the multi-robot system test. The percentage of cells exempt from coverage are also omitted in [2], making it difficult to determine whether our path lengths are sufficiently good.

Another limitation that we saw in this is that, because of the borders of the grid, there were areas that needed to be covered, but were inaccessible to the robot. In [2], the authors were able to avoid this issue because their testing environments didn't have areas to be covered that were inaccessible to the robot (see Fig. 18). This caused our resulting coverage to be lower than what it could be.

C. Future Work

In the future, we plan to improve our crater detection algorithm and develop a more general coverage planning framework for multiple image datasets. All code for the current workflow can be found at: https://github.com/holmes1000/Lunar-Path-Planning

If we had more time, we would've liked to compare our results to DeepMoon [7]. It would also have been nice to take the lunar image crops, manually annotate them, feed them into a U-net model, and detect the craters.

We would also like to expand this to work with fuel-constrained multi-robot systems. In [2], multi-robot path planning is briefly covered and compared using three robots. However, it is only tested using three robots and one coverage environment.

Using multi-robot systems would require more computational power. This project was run using a Lenovo Thinkpad with an Intel Core Ultra 7 155H 16-core processor and an Nvidia RTX 500 Ada Generation Laptop GPU.

Lastly, the workflow proposed in this paper is an offline image processor and path planner. It would be beneficial to expand this to online applications, enabling real-time capabilities for lunar navigation.

VII. Conclusion

We propose a system for detecting craters and for planning efficient complete coverage paths on lunar terrains. Using computer vision techniques, we develop a crater detection algorithm. Based on these resulting detections, we derive a grid-based map; defining areas that need to be covered. A Hamiltonian cycle expansion strategy and a low-repetitive coverage path planner are then used to generate paths.

References

[1] E. J. Speyerer, R. V. Wagner, M. S. Robinson, *et al.*, "Pre-flight and On-orbit Geometric Calibration of the Lunar Reconnaissance Orbiter Camera," en, *Space Science Reviews*, vol. 200, no. 1-4, pp. 357–392, Apr. 2016, ISSN: 0038-6308, 1572-9672. DOI: 10.1007/s11214-014-0073-3. [Online]. Available: http://link.springer.com/10.1007/s11214-014-0073-3 (visited on 04/05/2025).

[2] Y. Tang, Q. Wu, C. Zhu, and L. Chen, "An Efficient Coverage Method for Irregularly Shaped Terrains," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Oct. 2024, pp. 7641–7647. DOI: 10.1109/IROS58592.2024.10801856. [Online]. Available: https://ieeexplore.ieee.org/document/10801856 (visited on 04/05/2025).

[3] Y. Bandyopadhyay, "Lunar Crater Detection Using YOLOv8 Deep Learning," en, Mar. 2024. [Online]. Available: https://eartharxiv.org/repository/view/6828/ (visited on 04/05/2025).

[4] M. Sinha, S. Paul, M. Ghosh, S. N. Mohanty, and R. M. Pattanayak, "Automated Lunar Crater Identification with Chandrayaan-2 TMC-2 Images using Deep Convolutional Neural Networks," en, *Scientific Reports*, vol. 14, no. 1, p. 8231, Apr. 2024, ISSN: 2045-2322. DOI: 10.1038/s41598-024-58438-4. [Online]. Available: https://www.nature.com/articles/s41598-024-58438-4 (visited on 04/05/2025).

[5] M. Chen, D. Liu, K. Qian, J. Li, M. Lei, and Y. Zhou, "Lunar Crater Detection Based on Terrain Analysis and Mathematical Morphology Methods Using Digital Elevation Models," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 7, pp. 3681–3692, Jul. 2018, ISSN: 1558-0644. DOI: 10.1109/TGRS.2018.2806371. [Online]. Available: https://ieeexplore.ieee.org/document/8310940 (visited on 04/05/2025).

[6] X. Lin, Z. Zhu, X. Yu, *et al.*, "Lunar Crater Detection on Digital Elevation Model: A Complete Workflow Using Deep Learning and Its Application," en, *Remote Sensing*, vol. 14, no. 3, p. 621, Jan. 2022, ISSN: 2072-4292. DOI: 10.3390/rs14030621. [Online]. Available: https://www.mdpi.com/2072-4292/14/3/621 (visited on 04/05/2025).

[7] A. Silburt, M. Ali-Dib, C. Zhu, *et al.*, *Lunar Crater Identification via Deep Learning*, arXiv:1803.02192, Nov. 2018. DOI: 10.48550/arXiv.1803.02192. [Online]. Available: http://arxiv.org/abs/1803.02192 (visited on 04/08/2025).

[8] L. M. Downes, T. J. Steiner, and J. P. How, "Deep Learning Crater Detection for Lunar Terrain Relative Navigation," en, *Other repository*, Jan. 2020. [Online]. Available: https://dspace.mit.edu/handle/1721.1/137175.2 (visited on 04/05/2025).

[9] I. Giannakis, A. Bhardwaj, L. Sam, and G. Leontidis, *Deep learning universal crater detection using Segment Anything Model (SAM)*, 2023. DOI: 10.48550/ARXIV.2304.07764. [Online]. Available: https://arxiv.org/abs/2304.07764 (visited on 04/08/2025).

[10] L. Yang, B. Kang, Z. Huang, *et al.*, *Depth Anything V2*, en, Jun. 2024. [Online]. Available: https://arxiv.org/abs/2406.09414v2 (visited on 04/20/2025).

[11] Y. Idikut, C. Cummings, and S. Holmes, "Multi-Robot Persistent Coverage Under Fuel and Stochastic Failure Constraints," Undergraduate Major Qualifying Project, Worcester Polytechnic Institute, 2024.

[12] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, ISSN: 1050-4729, May 2015, pp. 1093–1099. DOI: 10.1109/ICRA.2015.7139312. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7139312 (visited on 04/05/2025).

[13] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in *2010 IEEE International Conference on Robotics and Automation*, ISSN: 1050-4729, May 2010, pp. 5525–5530. DOI: 10.1109/ROBOT.2010.5509860. [Online]. Available: https://ieeexplore.ieee.org/document/5509860 (visited on 04/05/2025).

[14] G. Fevgas, T. Lagkas, V. Argyriou, and P. Sarigiannidis, "Coverage Path Planning Methods Focusing on Energy Efficient and Cooperative Strategies for Unmanned Aerial Vehicles," en, *Sensors*, vol. 22, no. 3, p. 1235, Feb. 2022, ISSN: 1424-8220. DOI: 10.3390/s22031235. [Online]. Available: https://www.mdpi.com/1424-8220/22/3/1235 (visited on 04/07/2025).

[15] *Martian/Lunar Crater Detection Dataset*, en. [Online]. Available: https://www.kaggle.com/datasets/lincolnzh/martianlunar-crater-detection-dataset (visited on 04/20/2025).

VIII. Appendix

The results section presented one set of crater detection tests; however, it is important to note that the workflow allows multiple random image crops to be generated and processed. See the figures below for examples.

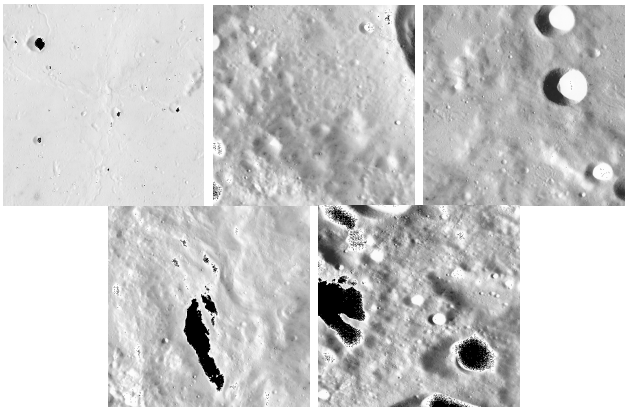


Fig. 21: Lunar random crop samples

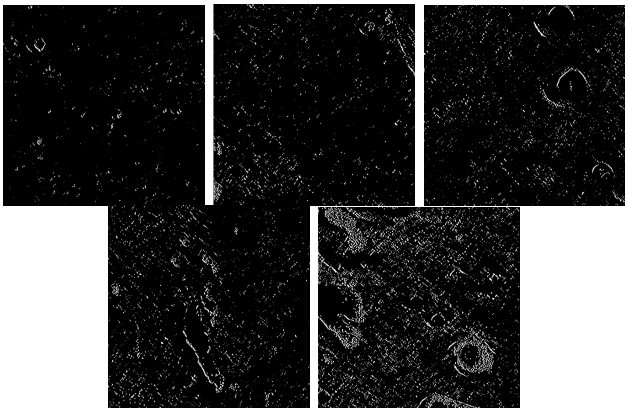


Fig. 22: Edge detection samples

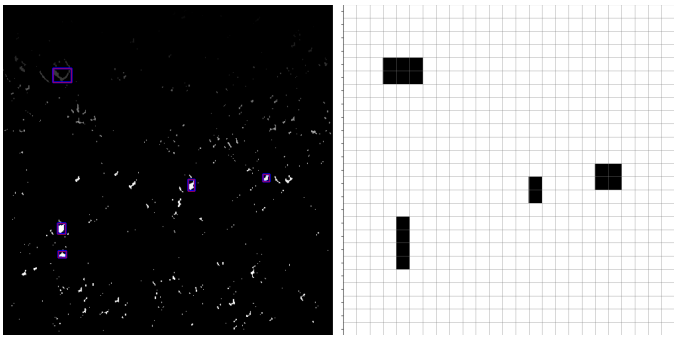


Fig. 23: Crater detection and grid result - test 1

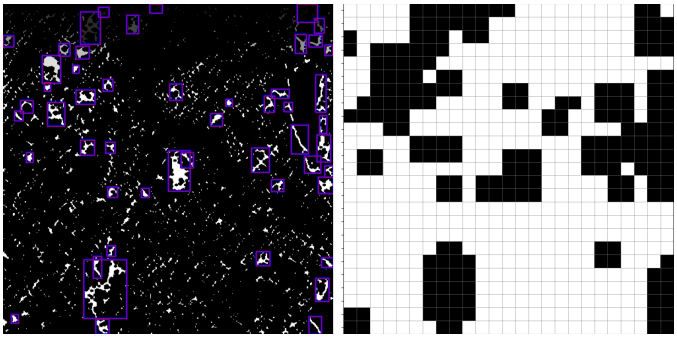


Fig. 24: Crater detection and grid result - test 2

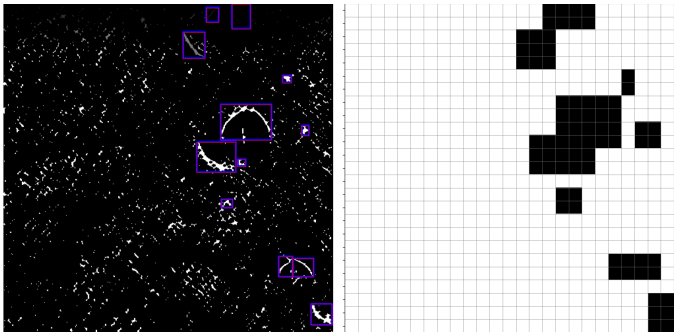


Fig. 25: Crater detection and grid result - test 3

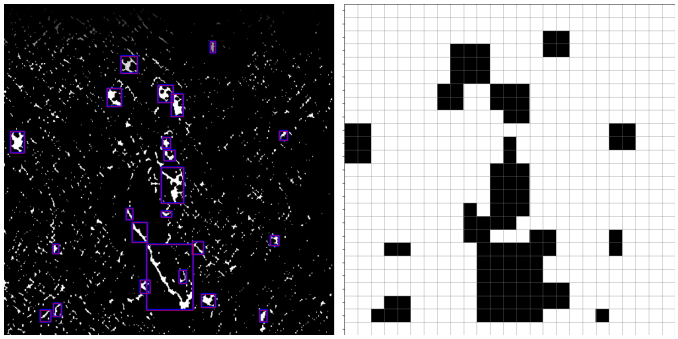


Fig. 26: Crater detection and grid result - test 4

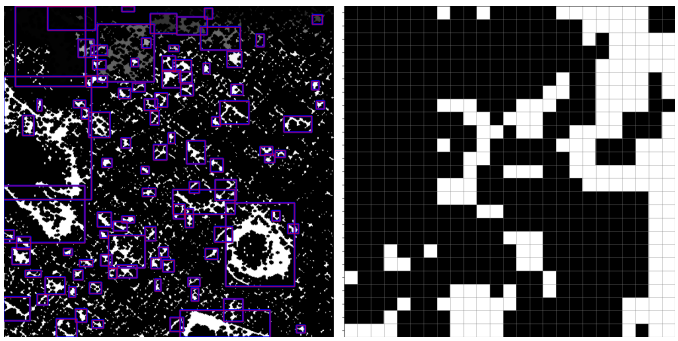


Fig. 27: Crater detection and grid result - test 5