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

Samara Holmes
Northeastern University

Abstract - Mapping the Moon's surface is challenging due to its irregular shapes and features like craters. This paper introduces an efficient method for covering lunar terrains. It uses a neural network to detect craters accurately and combines this with smart algorithms for effective mapping. The result is better coverage with less effort, improving accuracy and efficiency for future lunar missions.

I. Problem Statement

Mapping the lunar surface is a crucial challenge for advancing robotic exploration and future missions. The Moon's irregular terrain, marked by craters of varying sizes, poses significant obstacles to achieving efficient and accurate mapping. Current approaches often struggle with balancing precision and computational effort, limiting their application in real-world scenarios.

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. The goal is to produce outputs that clearly label craters as 'obstacles' and identify the remaining 'areas to cover.' This detection process can leverage image manipulation techniques or neural networks trained specifically for this purpose.
2. Robotic Path Planning for Coverage: Solving the coverage problem by creating efficient algorithms for robotic path planning. This ensures optimal terrain exploration, minimizing effort while maximizing surface area coverage.

II. Data Collection

To obtain data for my project, I will need to grab a lunar TIFF image. Once I load the image, I will randomly grab and save image patches. Using these image patches, I can either annotate and then train a network or directly manipulate them using OpenCV functions.

III. Solution Design

I will be tackling this problem with the following phases:

Phase I: Loading the data (load_image.py)

1. Load a Lunar terrain dataset (in this case, a TIFF image)
2. Get a crop from the TIFF
3. Do some post-processing on the TIFF (gaussian? something for smoothing/noise reduction?)
   4. Save the crop to /images

Phase II: Image Processing and Crater Detection (process_image.py and detect_crater.py)

1. Load the crop from /images
2. Do image processing (sobel filters, binary thresholding)
3. Save those images to /binary_images
4. Detect the craters (from images in /binary_images)
5. Generate DepthAnything images (from images in /random_images)

Phase III: Prepare the data for path planning (generate_obstacles.py)

1. Take the crater detections and turn them into obstacles to be used for the coverage method (combine visuals from detect_crater.py and run_depth_anything.py)

2. Split the image and define 'areas to be covered' vs 'obstacles'

3. Generate a grid map

Phase IV: Conduct coverage path planning for one robot

1. Write the path planning algorithm from [1]
2. Generate the total coverage rate along with the path length for the robot
3. Save path plots with coverage rates and path lengths to /paths/single_robot

IV. Extensions on Design

Phase X: Create a UNet Model

1. Load the data, and generate several randomly cropped images
2. Annotate the data using CVAT
3. Generate a prediction mask 4. Train against the annotations for crater detections 5. Feed this into the coverage algorithm

Phase Y: Make the data more advanced

1. Factor in the depths of the craters - For certain craters, you can go through the middle

Phase Z: Try with other models and compare

1. Try a YOLO model for crater detection
2. Use DeepMoon for crater detection
3. Create crater detection graph comparison

V. Outcome

A good outcome for this project would be a complete application displaying both the computer vision and robotics aspects. The computer vision side should be able to accurately detect Lunar craters and turn those detections into obstacles (whether with a custom or pre-trained model). The robotics aspect should be able to efficiently and effectively develop robotic paths to maximize coverage around those obstacles.

VI. References

1. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10801856
2. https://www.researchgate.net/profile/Lena-Downes/publication/338399036_Deep_Learning_Crater_Detection_for_Lunar_Terrain_Relative_Navigation/links/5efbbce7299bf18816f5ea9d/Deep-Learning-Crater-Detection-for-Lunar-Terrain-Relative-Navigation.pdf
3. https://arxiv.org/pdf/1601.00978
4. https://github.com/wdoppenberg/ellipse-rcnn?tab=readme-ov-file
5. https://github.com/alrzmshy/Mars-Crater-Detection
6. https://developers.arcgis.com/python/latest/samples/lunar-craters-detection-from-dem-using-deep-learning/
7. https://github.com/prathameshbarapatre5/Lunar-Crater-Detection-using-LRO-Dataset/tree/main
8. https://pds-imaging.jpl.nasa.gov/volumes/lro.html
9. https://wms.lroc.asu.edu/lroc/view_rdr/WAC_ROI_NEARSIDE_DAWN
10. https://github.com/silburt/DeepMoon?tab=readme-ov-file