Using Planned View Trajectories to Build Good Models of Planetary Features under Transient Illumination

Heather L. Jones

CMU-RI-TR-16-23

Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

May 2016

Abstract

This research addresses the modeling of substantially 3D planetary terrain features, such as skylights, canyons, craters, rocks, and mesas, by a surface robot. The sun lights planetary features with transient, directional illumination. For concave features like skylight pits, craters, and canyons, this can lead to dark shadows. For all terrain features, the ability to detect interest points and to match them between images is complicated by changing illumination, so seeing planetary features in the best light requires a coordinated dance with the sun as it arcs across the sky.

The research develops a process for planned-view-trajectory model building that converts a coarse model of a terrain feature and knowledge about illumination change and mission parameters into a view trajectory planning problem, plans and executes a view trajectory, and builds a detailed model from the captured images. An understanding of how view and illumination angles affect model quality is reached through controlled lighting laboratory experiments. The planning of view trajectories, (what to image, from where, and at what time), is cast as an OPTWIND, a new vehicle routing problem formulated in the thesis work. Each part of the planned-view-trajectory model building process is examined in detail, with existing tools identified and tested to solve parts of the problem, where appropriate, and new solutions implemented otherwise. The research also demonstrates planned-view-trajectory model building.

Contributions of the research include development, implementation, and demonstration of planned-view-trajectory model building, experimental determination of factors affecting model quality and formulation of view trajectory planning as a new vehicle routing problem. Datasets for planetary analog terrain and for imaging under directional lighting were also collected, totaling over 25,000 images.

Acknowledgements

It would be difficult to name everyone who has helped along the way to this PhD, but I will try to mention a few. Thanks to my parents for many things, including not thinking I was crazy to give up a job and come back for more school.

Many thanks go to my advisor, Red Whittaker, for many years of guidance. David Wettergreen, Michael Kaess, and Jeremy Frank also deserve thanks for graciously agreeing to serve on my committee, and for the advice they provided along the way to this thesis.

Thanks to past and current members of the Lunar team including (but not limited to) Wennie Tabib, Chris Cunningham, Nate Otten, Joe Bartels, Steven Huber, Lauren Schneider, Dean Thompson, Jason Calaiaro, John Thornton, Ethan Minogue, Joe Amato, Eugene Fang, Erika Bannon, Curtis Boirum, Aashish Sheshadri, Steve McGuire, Krzysztof Skonieczny, Tom Carlone and Jon Anderson. Kevin Peterson and Uland Wong deserve special thanks for providing senior grad student perspectives as I was just getting started.

Thanks to all my fellow travelers along this grad-school road. These include my current and former officemates Humphrey Hu, Shaurya Shankar, Mark Desnoyer, Kevin Peterson, Nate Wood, Marynel Vasquez, and Bruno Hexsel and my former housemates Anca Dragan, Chris Dellin, Leland Thorpe, and Shervin Javdani. And I'd like to especially acknowledge the "dinner train", which has at one point or another included Pras Velagapudi, Pyry Matikainen, Brian Becker, Joydeep Biswas, Michael Furlong, Heather Justice, Eleanor Aruvnin, Ada Zhang, Rika Antonova, Nathan Brooks, Ellen Cappo, Brina Goyette, Uma Nagarajan, Peter Anderson-Sprecher, and Rachel Vistein. You helped keep me sane.

Thanks to the Intelligent Robotics Group at NASA Ames for providing a second academic home during the multiple 'visiting technologist experiences' that were part of my fellowship. Thanks to my family up and down the west coast for hosting me when my lodging became suddenly unavailable due to a government shutdown during one of these 'visiting technologist experiences', and for not freaking out (too much) when I disappeared for hours into a lava tube cave with a camera.

I would also like to thank the FINESSE SSERVI project for the chance to participate in a field campaign at Craters of the Moon National Monument and Preserve, Idaho. Craters of the Moon field team members Chuck Whittaker and Ander Solorzano played a critical role in data collection for this work. Uland Wong provided much of the vision for that field campaign, as well as a merged and filtered LIDAR model for King's Bowl pit, used as ground truth in this work. Craters of the Moon park geologist Doug Owen provided valuable advice for successful field operations.

Uland Wong envisioned and implemented a controlled illumination rig in his thesis work that was used for the Controlled Lighting Experiments in this research, and he provided invaluable help in getting it running again for those experiments. Robin Ross, Jake Olkin, and Sachin Kolachana helped with data collection in the Controlled Lighting Experiments.

This work was supported by a NASA Space Technology Research Fellowship. Financial support for this research also came from a NASA Innovative Advanced Concepts (NIAC) grant, a NASA Space Grant Scholarship, and several other NASA grants.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

This thesis addresses the problem of robotically modeling substantially 3D planetary terrain features, such as pits, canyons, craters, rocks, and mesas. Planetary pits are interesting scientific targets, often exposing layers of bare rock in their walls, hinting at past volcanism and other subsurface processes. Skylights are rare pits that access cave entrances. Likely skylights have been identified in recent high-resolution images of the Moon and Mars [7, 8]. These discoveries open a fascinating new class of mission destinations. Skylights could provide glimpses into underground worlds beyond Earth. Crater morphology is interesting for planning rover routes and pursuing ice in permanently shadowed regions. Erosion that causes canyons and mesas exposes bedrock layers. On Mars, the recurring slope lineae (or RSL) that appear on slopes of canyons and mesas were recently identified as seasonal flows of briny liquid water [9]. Modeling these features - from a distance - as the RSLs change seasonally could be key to the search for life on Mars.

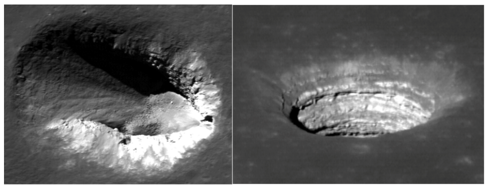


Figure 1.1: Left: Pit viewed from above; Right: Oblique view of skylight, showing layered walls (Image credit NASA/GSFC/ASU)

The sun illuminates planetary features with transient, directional lighting. For concave features like skylight pits, craters, and canyons, this can lead to dark shadows. For all terrain features, the ability to detect interest points and to match them between images is complicated by changing illumination, so seeing planetary features in the best light requires a coordinated dance with the sun as it arcs across the sky.

Quality models of planetary terrain features could provide great scientific and mission planning value, but building these models is challenging due to transient illumination, and there are several important questions to consider. What does a quality model of a planetary terrain feature mean, and what does it take to achieve it? How does a robot plan to model a planetary terrain feature under transient illumination? How can plans to model terrain features ensure model quality?

In this work, it is assumed that a robot has a passive, monocular camera[1] to collect model data. Cameras are good for capturing high-resolution data at long distances and can be used to image faces of a feature, reconstruct terrain feature geometry, and even determine material types if the images have the right set of illumination and view angles, including relative view and illumination angles. In a planetary environment, the illumination angle is determined by the motion of the sun over time, which is well understood. View angles are determined by robot position. The planning problem is then to determine what to view, where to view from, and when, to capture images that meet illumination angle, view angle, relative illumination angle, relative view angle, and viewing distance constraints designed to improve model quality. The ordering of views is important, because of the time of day that images are taken, and also to minimize distance traveled. The research develops planned-view-trajectory model building, which converts a coarse model of a terrain feature and knowledge about illumination change and mission parameters into a view trajectory planning problem, plans and executes a view trajectory, and builds a detailed model from the captured images.

Important considerations in this problem are:

- Illumination Angle: The angles between the surface normal, the incident illumination, and the view vector impact model fidelity, so a planner that seeks to achieve a threshold of model fidelity must plan for illumination angle in each view.

- Time: Under transient illumination from the sun, the absolute time determines the angle of incident light on each surface target, so the period when illumination angles meet constraints for a given face of a terrain feature will be restricted to one or more time windows.

- Viewing Angle: The angle at which terrain feature patches are viewed affects the resolution of the image on the surface and model quality. So among a set of positions from which a rover can observe a target patch, some will be better than others.

- Relative Constraints: Relative view and illumination angles between sets of images will affect whether they can be used for stereo reconstruction.

- Robotic Motion: Viewing all desired faces of a feature requires robot motion. Travel on or near (in the case of a flying robot) planetary terrain is inherently risky and requires use of limited resources, so the travel distance required to complete a view trajectory matters. Maximum robot velocity determines the minimum time to travel between viewpoints, given the distance between them. Robots must also avoid obstacles, making detailed paths between viewpoints potentially expensive to compute.

[1]The assumption of a monocular camera is adopted to simplify the expression of the problem. The work is also expected to be relevant to a number of other sensor configurations.

1.2 Approach

The approach posits that the choice of which images to take is the key to building good models of planetary features under transient illumination. This choice includes what to image, where to image from, and when to image, which in turn set the viewing distance, the view angle, and the illumination angle, as well as the relative view and illumination angles between images. The thesis work develops a process for planned-view-trajectory model building that converts prior data (a coarse terrain model, illumination change information, and robot limitations) into a planning problem that enables decision making on image options based on viewing distance, view angle, illumination angle, and relative view and illumination angles. The planning problem is fed to a planner, the planned view trajectory is executed, and the resulting images are used to build a detailed terrain model.

The thesis develops an understanding of how view and illumination angles affect model quality, primarily through laboratory experiments under controlled-lighting. This understanding facilitates setting limits on these factors to be used in planning.

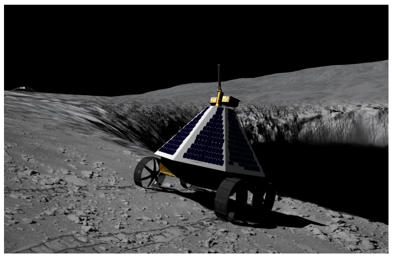


Figure 1.2: Surface rover viewing across a skylight

The approach casts planning of view trajectories for terrain feature modeling as an instance of a new vehicle routing problem, the Orienteering Problem with Time Windows and Inter-Node Dependencies (OPTWIND). The Orienteering Problem (OP) is a long-studied vehicle routing problem that seeks to maximize reward by visiting some number of nodes, each of which has some reward, while minimizing or constraining the cost to travel between nodes [10]. The associated decision problem for the OP is NP-hard [11, 12]. The OP is distinguished from the well-known Traveling Salesman Problem (TSP), in which a route must be found that visits all nodes. An extension of the Orienteering Problem with Time Windows (OPTW) restricts the times in which nodes may be visited to a given time window for each node [10]. The Generalized Orienteering Problem (GOP) modifies the Orienteering Problem by computing the total reward as a nonlinear function of the rewards at each node visited, instead of simply summing up the rewards for each node visited. Nodes have scores in a set of categories, and the score previously accumulated for a given category will determine how much a node with high a score for that category will increase the total value of the route [13, 14]. The OPTWIND extends the OPTW and the GOP by adding *inter-node dependencies*.

In casting view trajectory planning as an OPTWIND, nodes consist of robot sensor positions at a set of discretized times, and each position-time node has some number of visible target faces on the terrain feature. For each target face visible from a position, there are associated view and illumination angles. A view of a target face only counts toward the OPTWIND value if it meets view and illumination angle limits. Relative view and illumination angle limits can be represented in the inter-node dependencies of the OPTWIND. The approach formulates the OPTWIND and develops methods to translate real terrain-feature-modeling cases into OPTWIND instances. The thesis work develops an heuristic algorithm to solve the OPTWIND that starts with an initial feasible trajectory chosen to take advantage of limited time window opportunities. This initial trajectory is improved with local search methods that take advantage of an understanding of categories. However, the dissertation makes no claims about the optimality or efficiency of this algorithm. Further investigation of planning methods is left to future work.

Each part of planned-view-trajectory model building is examined in detail. For some parts, solutions are identified from among existing tools and tested to ensure applicability. For other parts, new solutions are developed. Finally, planned-view-trajectory model building is demonstrated and conclusions are drawn from the work.

1.3 Thesis Statement

This thesis asserts that using planned, illumination-aware view trajectories facilitates building good models of substantially 3D planetary terrain features under transient, directional illumination with monocular imagery acquired by a surface robot.

Planned: All sensor views that will be used to build the model are explicitly planned in advance.

Illumination-aware: Planned sensor views obey illumination constraints, and views may be chosen to achieve specific illumination angles.

View trajectories: Planning determines a sequence of robot positions, sensor view targets, and imaging times.

Good models: 3D model quality meets criteria of coverage, resolution, and interest-point visibility.

Substantially 3D: The feature has elements (i.e. walls) that cannot be traversed by the surface rover, the feature is not planar, and all areas of interest on the feature cannot be captured in a single image.

Terrain features: The thesis scope covers natural terrain features and not man-made objects. Terrain features are assumed to be static at imaging distances and timescales: i.e., seasonal appearance change would be acceptable, but not fast shape change, like windblown tree branches or ocean waves.

Transient, directional illumination: A substantial portion of the illumination comes from a directional source, such as the sun. Illumination angle changes over time in a known way. Change in illumination over mission timescale is significant, but rate of change is slow relative to robot motion.

Monocular imagery: The robot-mounted sensor can be represented by a single camera frame, and it provides a different response depending on scene illumination.

Surface robot: The sensor is carried by a robot that operates on or near the terrain surface, not in orbit. The robot may have thresholds on traversable slope and obstacle size, or approach distance may be limited by contamination concerns.

1.4 Contributions

This thesis makes the following contributions:

- Development of a process for planned-view-trajectory model building
- Investigation of modeling from laboratory and field data to inform selection of view and illumination angle constraints to use in planning
- Demonstration of planned-view-trajectory model building
- Formulation of the Orienteering Problem with Time Windows and Inter-Node Dependencies
- Image datasets for rocks under controlled, directional lighting
- Image datasets for analog terrain features

1.5 Organization

The thesis document is organized as follows. Chapter 2 discusses related work. Chapter 3 introduces the planned-view-trajectory model building process. Chapter 4 explores what makes a good model and how view and illumination angles affect model building, including conclusions from experimental results. Chapter 5 formulates the OPTWIND problem. Chapter 6 details the planned-view-trajectory model building process and discusses how solutions were implemented or existing tools were identified and tested for each part of the process. Chapter 7 discusses demonstrations of planned-view-trajectory model building concepts. Chapter 8 draws conclusions and discusses future work. Appendix A details the campaign of controlled lighting laboratory experiments used to inform the understanding of what makes a good model. Appendix B discusses field experimentation sites. Appendix C includes the PDDL planning files used in Section 6.3.4.

1.6 Publication Note

Portions of this work have appeared in [15] and in [16].

Chapter 2

Background and Related Work

2.1 Introduction

View planning is fundamental to view trajectory planning for modeling large, local terrain features like planetary pits and canyon walls with images taken by surface rovers. This is especially true in the discussion of how views are represented and how potential view-viewpoint pairs are selected. Unlike much view-planning work motivated by factory applications, a rover is constrained to move on the ground and avoid obstacles. Traveling between viewpoints takes time and incurs risk. This motivates the distinction between many view planning methods and view planning for mobile robots, including view trajectory planning. Planning routes with multiple destinations is also critical to view trajectory planning for pit modeling. The Orienteering Problem, and its multiple extensions, provide insight on how to formalize view trajectory planning for pit modeling. Artificial intelligence (AI) planning and scheduling have been used in planetary missions to decide which targets to image, and in what order. AI planning and scheduling also handles time constraints, such as having a period during the day when the illumination angle on a target terrain face is in the desired range for imaging. Section 2.2 presents methods that address view planning, view planning from mobile robots, the Orienteering Problem and its extensions, and AI planning and scheduling.

An understanding of how the model will be built from the planned images is also critical when planning images for model building. Modeling is addressed with a brief treatment in Section 2.3. Current methods used to build models from planetary mission data are discussed, along with techniques for building models from many images, and techniques for taking advantage of illumination.

2.2 Planning

2.2.1 View Planning

Planning a view trajectory requires a representation of view. This is informed by prior work in view planning. What a view sees, (*view target*), where it is seen from, (*viewpoint*), which is camera position and orientation, and how a view is created are all important.

Tarabanis, Allen and Tsai researched view planning for inspecting specific target features on known objects in a factory environment [17]. They make the distinction between generate and test view planning methods, where a set of viewpoint parameters are generated and tested to determine the view target, and synthesis methods, where viewpoint parameters are calculated for each given view target. They identify three components of the view planning problem: the sensor parameters, the sensor and object models, and the feature detectability constraints. Under sensor parameters, they include sensor and illumination position and orientation, as well as more camera-specific parameters such as aperture, focal length, exposure time, gain, and spectral transmittance. Alternately, in planning for pit modeling, only sensor position and orientation and illumination angle are considered. In this research, camera-specific parameters are considered to be fixed. Camera field of view is important for determining what target area can be covered in a single image, but this thesis work assumes that the camera is focused at infinity (such that the distance from the target only affects ground resolution, not focus). A single view can consist of several images in a high dynamic range (HDR) bracket if needed (so exposure time can be ignored).

Sensor and object models may contain models of image formation, geometry of the target object, and photometric information about the target material [17]. The thesis work uses a pinhole camera model and assumes that there is some, perhaps coarse, prior model of the terrain feature. Constraints on illumination and view angle may implicitly encode knowledge of a terrain feature. For example, if for the set of materials from which the feature is expected to be composed, there are known illumination angle ranges that will not produce good images, those ranges could be excluded by absolute illumination angle constraints.

Feature detectability constraints include whether there is line of sight to the feature, and whether the sensor field of view is sufficient to have the feature in frame, focus, resolution, and distortion [17]. In planned-view-trajectory modeling, line of sight, field of view, and resolution are considered in determining what positions can actually view a target. Distortion is handled through view angle constraints.

View planning methods can be broadly categorized as model-based and non-model-based [18]. Model-based methods, like view trajectory planning for pits assume some a priori model of the object or feature to be modeled. In non-model based methods there is no a priori model.

How the view target is represented also matters. This could be based on volume, surface area, or small features of specific interest. Methods that model polyhedral objects often use target vertices and edges in view planning, and assume a constant normal vector for faces. For more irregular targets, such as planetary pits, the volume or surface area that views observe could be discretized finely or coarsely into voxels or planar faces representing surface area.

Sakane and Sato present HEAVEN, a system for planning camera and illumination positions for active photometric stereo [19]. They assume an object with known coarse position, but with uncertainty in position and orientation, and seek to reconstruct its surface normals. They need three light source positions, and these must be sufficiently different for photometric stereo to succeed. With a Gaussian sphere, they predict surface normals and compute a combination of estimated reliability and detectability metrics. These are used to determine light source placement. They use a generate and test method with virtual geodesic domes around the object to provide a set of potential positions for the camera and light sources. Camera and light source positions are then implemented by robotic arms. In the pit modeling problem, photometric stereo would translate to a requirement to take three images of a target, with relative constraints on illumination angle (they must be sufficiently different) and on view angle (should all be the same view angle, or sufficiently close).

Solomon and Ikeuchi also plan sensor and illumination positions for photometric stereo reconstruction of objects in a factory setting [20, 21, 22]. The inputs to their system are a prior model of the object to be modeled, a sensor noise model, and a reflectance model. They assume that light rays are parallel. Although not strictly true due to sun diameter, parallel light rays are also a reasonable assumption for illumination from the sun in the pit modeling case, due to the sun's great distance from the target. Solomon and Ikeuchi determine an exact cover of each face (described by its normal) by a set of light sources placed in an icosahedron centered on the object to be inspected (using a heuristic that seeks to minimize the number of measurements required). Camera position is chosen from a set of possible viewpoints (a generate and test method) to minimize the foreshortening of any faces visible in that view. Additional views are added until all faces to be inspected are covered.

Cowan and Bergman model polygonal objects with a lower bound on resolution by using a synthesis method in which camera positions lie on the intersection of spheres (with radius determined by the maximum distance for a given resolution) centered at vertices on the polygon's convex hull [23]. They compute a region of acceptable light source positions, using limits on minimum and maximum distance to get a goal intensity of reflected light, and subtracting out regions that would result in the camera seeing specular reflections. Specular reflections depend on angle between the surface normal and the incoming light, which they assume is a point source. In planning for pit modeling, the distance between the light source (the sun) and the target is not controllable, but a range of illumination angles are available, depending on time of day, so a "region of acceptable light source placement" is handled by absolute constraints on illumination angle for imaging a particular terrain face, which in turn determines the times when the face can be imaged.

Tarabanis, Allen and Tsai use a synthesis method for determining camera position, orientation and imaging parameters that computes regions in 8-dimensional space in which target features are visible, then optimizes over this space to find the point farthest from the bounds of the region [17]. Choosing such points that are far from the edges that separate acceptable from unacceptable views makes the method more robust to uncertainty. This is important for planned-view-trajectory modeling. It is addressed by the choice of potential robot positions, by ensuring that target terrain faces are smaller than what a camera could cover in one view, and by selecting view and illumination angles that are somewhat conservative. It is, however, not explicitly considered in planning.

Scott, Roth, and Rivest survey view planning methods for 3D object reconstruction using active triangulation range sensors [18]. For model-based methods, they categorize methods as using set theory, graph theory, or computational geometry. Set theory methods use a visibility matrix to encode which surface features on the target are visible from which viewpoints. Graph theory methods use aspect graphs, where aspect is defined as the set of viewpoints that have qualitatively the same view, and arcs connect adjacent aspects in the graph. Computational geometry methods use the art gallery problem, which looks for the minimum number and placement of guards to see all internal walls of an art gallery. Planned-view-trajectory modeling requires a form of visibility matrix to represent which faces of the target terrain feature are visible from which positions.

Non-model-based methods often take a "next-best-view" approach. Given what has been seen so far, the approach generates the next-best-view that provides the most new information. In some cases, overlap with existing data to facilitate model building is also considered [24]. Overlap between views is also important for planned-view-trajectory modeling. This is especially true if one coherent model is desired instead of a set of discrete, 3D faces. Overlap is addressed by setting target terrain face sizes smaller than the camera's view, but it is not explicitly considered in planning.

Scott, Roth, and Rivest divide non-model-based approaches into surface-based, global, and volumetric methods [18]. Surface-based methods include those that look at occlusion edges - these are where (1) the object to be modeled goes beyond the edge of the sensor field of view, or where parts of the object are (2) occluded in the sensor's view or (3) shadowed by other parts of the object. Other surface-based methods are those that keep the sensor close to the object and follow its contours, and those that model the object with parametric surfaces and select views based on the areas with the greatest uncertainty. Global methods include the use of a mass vector chain, where the observed object is segmented into patches with weighted normal vectors and the next-best-view is the negative of the cumulative mass vector chain, and intermediate space representations, which encode a virtual surface between the object and the sensor.

Volumetric methods include voxel occupancy. Kruse, Gutsche and Wahl present a method for planning sensor views to explore a previously unknown 3D space where each voxel in a 3D grid is marked as either occupied, free or unknown [25]. The value of a given view is evaluated by estimating the size of the unknown regions that become known after the measurement and determining the distance between that view and the current position in robot configuration space. Ray tracing estimates the size of unknown regions that can be seen in a given view, using a relatively small number of rays to limit computation time. An early effort at pit modeling by the author of this thesis used a similar voxel-based approach with occupied, free or unknown voxels [26]. Given the large size of planetary pits, canyons, mesas, and craters, any fine voxel discretization will quickly become very expensive in terms of memory and computation, and in the work in [26], voxelizations smaller than about 0.5m/pixel were found to be infeasible.

Octrees, which encode space more efficiently than voxels, are used in other volumetric methods [18]. Space-carving is a volumetric method applied mainly to sensors with limited range. The volumetric method of solid geometry, used in many CAD packages, does well with complex geometry, but it can only subtract, and not add volume.

For most work in planning viewpoints and illumination, the illumination is either out of the planner's control and generally not important to the problem, or completely under the planner's control, meaning that any illumination settings can be selected at any time [23, 17, 20, 21, 22]. This differs from view trajectory planning, where there are a set of potential illumination conditions, but they happen at specific times. This means that the planner can select illumination conditions that are advantageous to the model-building objective, but it cannot necessarily capture views of all target faces under the same illumination condition, since transient illumination conditions do not last indefinitely.

Much early work in view planning centered around factory inspection and modeling tasks. While Kruse, Gutsche and Wahl used a metric of distance traveled between the current sensor configuration and the next view, most classical view planning methods do not. The most optimal views can be selected no matter what positions they are captured from, and the order can be determined based on metrics like information gain. For a mobile robot, choosing the next view from an information-optimal perspective without consideration for distance and navigation contingencies may produce unreasonably long paths. Discussion of view planning applied to mobile robots can be found in Section 2.2.2.

2.2.2 View Planning from Mobile Robots

Next best view has been applied to the robotic exploration of unknown environments [27]. Sawhney, Krishna and Srinathan use the amount of visible (but yet unseen) terrain combined with distance to determine the next best view for individuals in a multi-robot team. They find that the metric computed as (amount of unseen terrain)/distance is the most successful out of several evaluated [28]. This method was tried for pit modeling in the work described in [26], but it produced long path lengths for a single rover.

Moorehead introduces the *Multiple Information Metrics Exploration Planner* [29]. He describes robotic exploration as a Prize-Collecting Traveling Salesman problem. The salesman gets a prize for visiting each city, but incurs a cost for travel between cities; prizes in each city are independent. Prizes in his work are the expected information gain at each location, and thus they change as the robot travels and information is gathered, so they are not independent. This is also the situation in planning for terrain feature modeling. Two positions may have views of multiple terrain faces, with some overlap. Once one of these positions has been visited and the views visible from that position have been taken, the other will not have views to as many unseen faces.

Model-based view planning methods have also been applied to mobile robots. Hollinger et al. use uncertainty to plan sensor views for a ship inspection robot [30, 31]. They use a Gaussian process to model the surface of the ship hull. In their case, the cost of viewing is higher than the cost of moving between viewpoints, which is not the case with pit modeling.

Englot and Hover address the same ship inspection task [32]. They address view planning as a coverage sampling problem and then address the multi-goal planning problem once the set of views are decided. For the coverage sampling problem, they use a generate and test method. Robot view configurations are randomly sampled, and they seek to build a feasible covering set for the modeled target from among the set of sampled views such that each geometric primitive is observed a requisite number of times. For the multi-goal planning problem, they use a Lazy Traveling Salesman Problem (TSP) solver on the assumption that the cost of planning point-to-point routes is high relative to the cost of solving the TSP, since there are relatively few viewpoints (100-200). This may also be the case for view trajectory planning for pit modeling. An algorithm that does not require a full set of path lengths before starting and can instead request that path lengths be computed only as they are needed could be advantageous.

Blaer and Allen present a view planning method for a robot that travels in 2D and models buildings in 3D [33]. Their algorithm has two phases: one model-based using a 2D footprint of a building as a prior model. They select a set of views to cover the building walls from the 2D model, and then use a Traveling Salesman Problem heuristic solver to generate a path between these views. After collecting all the initially planned views with a 3D LIDAR scanner, they then transition to a Next Best View phase, where they seek to improve the model from the first phase.

Wang, Krishnamurti, and Gupta present the Metric Traveling View Planning Problem and demonstrate through L-reduction to the Set Cover Problem that it is O(log m) inapproximable, where m is the number of surface patches [34].

Estlin et al. address the rover mission-activity planning problem: choosing which rocks a robot will view from a set of interesting scientific targets, and in what order [35]. Science targets are selected by a decision layer and then ordered using a TSP heuristic solver. A global path planner provides distance estimates, to get from point to point. These distance estimates are used by the TSP solver. Extensive research addresses the planning of robot paths to go from a start position to a goal. These methods often rely on planning through a graph, where positions in space are the vertices and the costs to go between them (sometimes purely distance) are the edges. Once this graph is constructed, the planner can take advantage of simple graph search techniques, such as Breadth-First Search, Depth-First Search, or Dijkstra's Algorithm, or faster methods that take advantage of heuristics, like A* [36]. Some methods, like D*, seek to facilitate fast re-planning if the robot encounters unseen obstacles or strays from its intended trajectory by planning from the goal to the start and caching multiple paths from the goal [37, 38]. View trajectory planning takes advantage of existing technology for planning point-to-point paths, using knowledge of terrain obstacles, and this thesis does not make fundamental contributions to point-to-point path planning.

Smith also addresses the rover mission activity planning problem [39]. He describes it as an over-subscription planning problem, where there are too many goals for the time or resources available. Planning for terrain feature modeling is somewhat similar, in that there are far more positions with views of terrain than a robot should visit, though this is partly because positions will have overlapping sets of visible targets. Even if the robot covers all target faces, which may not be possible depending on day length and robot resources, it would not have visited every potential position to do so. While Estlin et al. assume that once the planner decides which targets will be viewed (and which will not), the set of activities is fixed and it can be handed over to a TSP solver [35], the order in which the rover chooses to sample the rocks will determine the distance it has to travel to get to each rock, and that determines the cost for each investigation activity. So, as Smith notes, the problem is more properly described as an Orienteering Problem instead of TSP. His method solves the Orienteering Problem as an intermediate step in planning. This enables the determination of path to impact the plan of which rocks should be visited. Pedersen, et al. use the same Orienteering Problem approach as Smith for rover mission activity planning [40].

Methods that combine view and path planning generally do not take time into account, except to minimize the total time required to complete the task. For terrain feature modeling under transient illumination, time is critical, since the time at which images are taken will determine whether or not they are valuable.

2.2.3 The Orienteering Problem and Extensions

The Orienteering Problem (OP), mentioned briefly in the previous section, is similar to the Traveling Salesman Problem, but in this case the agent does not have to visit every goal position, and each goal position provides some reward [41]. The agent seeks to maximize reward while minimizing path length. The OP has also been called the selective traveling salesperson problem, the prize-collecting traveling salesman problem, the maximum collection problem, the bank robber problem, the TSP with profits. It is a special case of the Resource Constrained TSP [10].

Vansteenwegen, Souffriau, and Oudheusden provide a survey of approaches to the OP and several of its extensions [10]. In the formulation discussed by Vansteenwegen et al., the starting and ending nodes for the orienteering problem are fixed. Each node has an associated reward, and each edge has an associated travel time. The goal is to maximize reward within a time budget, T_{max}. Methods for solving the Orienteering Problem often involve finding a feasible path, and then doing a local search to improve the feasible path. The local search often alternates between trying to improve the reward while staying under a maximum distance traveled, and trying to reduce the distance traveled. Common methods to increase reward try to insert nodes already in the path or swap nodes in the path with nodes not in the path. Common methods to reduce distance try to remove edges and replace them with other edges, or exchange nodes to save travel distance. An extension of the OP is the Team Orienteering Problem (TOP), where instead of a single agent there are multiple agents and each one has a route [10].

While the OP is similar to the view trajectory planning, the view trajectory planning problem also has time constraints on when goal positions can be visited to achieve rewards. This is more similar to the Orienteering Problem with Time Windows (OPTW), in which each node can only be visited during a certain time window. Methods to solve the OP do not always translate well to the OPTW [10]. In particular, swapping nodes with other nodes is complicated by the fact that they may not have equivalent time windows. Karbowska-Chilinska and Zabielski use a genetic algorithm to solve the OPTW [42].

Planning view trajectories could be formulated as an OPTW, with nodes being a combination of position and coarse time divisions, and the time window being defined over the coarse time division. The value of each node would then be dependent on whether the illumination was good on the faces visible from a node's position at that node's time. However, the OPTW does not reflect the fact that the value of a node can change based on what nodes were visited previously, which can happen when visits to previous nodes have collected all the images necessary to model a particular terrain face.

The Team Orienteering Problem with Time Windows (TOPTW) combines the TOP with the OPTW. Since this pit modeling work only considers a single rover, the team extensions of OP are not relevant. The TOPTW has also been applied to solve multi-day OPTW problems, assuming that the agent must start and end from pre-determined locations each day. This makes sense when considering the agent as a tourist, salesman, or delivery man. He would have to return to a hotel or to the depot each day. It makes less sense in the case of a planetary rover which will spend the night wherever it happens to be when night falls and continue the next day where it left off.

Mennell introduces the Sequence Dependent Team Orienteering Problem (SDTOP) [43]. In this problem, the value of a route depends on the order in which nodes were visited. For example, if node 5 is visited before node 7, a value of 10 would be added to the score, but if node 7 were visited before node 5, a value of 15 would be added. Planning for pit modeling differs from the SDTOP in that, while absolute time matters, the sequence in which two nodes were visited does not affect the overall value, though it may affect the total distance traveled.

The Target Visitation Problem (TVP) was introduced by Grundel and Jeffcoat, and further analyzed by Arulselvan, Commander, and Pardalos [44, 45]. In the TVP, nodes have different values that is somewhat time dependent. The example given is a surveillance task for an aerial vehicle where intelligence suggests that a terrorist may be in a set of different locations, but there are different probabilities of him being in each location. Thus, the value is greater if the high-probability locations are visited first, but all locations should be checked, and there is a desire to minimize distance traveled. Arulselvan, Commander, and Pardalos use a hybrid genetic algorithm to solve the TVP. Like for the SDTOP, ordering matters in the TVP, where it does not in planning for pit modeling.

In the Multi Constrained Team Orienteering Problem with Time Windows, visits to each node also have specified durations and entry fees. Nodes are also sorted into categories, and once a given number of nodes in a category have been visited, additional nodes in that category are not valuable. Sylejmani, Dorn, and Musliu use the Tabu Search metaheuristic to solve the MCTOPTW [46]. Tabu search uses a number of moves that can potentially improve the path. "Tabu" memory keeps track of moves that cannot be performed for a certain number of iterations, which tends to diversify the search. They also use a technique of fast constraint checking to ensure that constraints (time windows, nodes per category, etc.) are not violated. In view trajectory planning, there is a desired number of views for each face. The problem could potentially be formulated as an MCTOPTW, but this would require that each position-time-view combination be a separate node, which may not be desirable. Not only would this greatly increase the number of nodes in the problem, but if a rover is at a given position, it makes sense to capture all the views associated with that position (or at least all the views of unseen faces or those that meet constraints with prior views of a face), and the MCTOPTW does not capture this connection between faces viewable from the same position. The MCTOPTW also does not handle the idea of relative constraints on angles between nodes.

In the generalized orienteering problem (GOP), the total score is a nonlinear function of the vertices visited [14, 13]. Specifically:

$$Z = \sum_{g=1}^{m} W_g \left[\{ \sum_{i \in P} \left[S_g \left(i \right) \right]^k \}^{\frac{1}{k}} \right] \tag{2.1}$$

This function sums over all nodes in the path P. Each node has a set of scores S_g in different categories. The example given is a tourist planning a trip who wants to see attractions that each have some score for natural beauty, cultural heritage, shopping, etc. Given that the tourist has seen several attractions with high scores in natural beauty, additional attractions with high natural beauty may not be as interesting anymore. Each tourist may have his or her own weightings, W_g for each type of score. This function tends to level off in score for each category as the number of nodes visited increases with high scores in that category increases. Wang, Golden and Wasil use a genetic algorithm to solve the GOP [14].

Planning view trajectories is similar to the GOP. Given position-time nodes with scores determined by the faces visible from that location, if a face has already been viewed many times, it should not increase the node's score. However, there is an additional level of dependency between nodes in planning view trajectories for pit modeling. If the goal is to get stereo depth reconstruction for each face, then there are two views required for each face. After the first view of a face, the second view would be of value if it meets the relative view angle constraints for stereo or no value if it did not. The GOP also does not include time windows. This thesis work thus seeks to extend the GOP and the MCTOPTW and define a new problem, the Orienteering Problem with Time Windows and Inter-Node Dependencies (OPTWIND), that handles inter-node dependencies such as relative constraints on view or illumination angle for a face viewable from multiple nodes.

2.2.4 AI Planning and Scheduling

Research in AI Planning and Scheduling has developed many general-purpose planners. Individual planners are often designed such that they can solve a range of problems across many domains. Some of these planners might be able to solve the OPTWIND.

AI Planning and Scheduling have successfully been used to plan operations for space missions. The Remote Agent Experiment (RAX) on Deep Space 1 pioneered spacecraft autonomy. Its planning and scheduling software was able to access a database of long-term mission goals and plan its activities over a multiple day span [47]. The Planner/Scheduler

(PS) searches over the plan space to determine the most optimal plan given its constraints and goals. It also predicts the state of the craft over the course of the plan, including resource levels [48]. For Deep Space 1, each component of the spacecraft state vector was represented as a timeline, as were the spacecraft resources. Tokens on timelines represent a period when the variable or resource holds a certain value. Constraints were defined between timelines.

Frank and Jónsson present the Constraint-based Attribute and Interval Planning (CAIP) framework, as well as an implementation of this framework, the Extendable Uniform Remote Operations Planning Architecture (EUROPA) [49, 50]. EUROPA is a descendent of the Deep Space 1 planner. It also uses tokens and constraints. In the CAIP framework, constraints are represented as procedures, and arbitrary functions can be expressed as relations between variables of the procedure. If all variables in a procedure's scope are given single values (as opposed to a set of possible values), then it must give a definitive answer. EUROPA is used for planning and constraint enforcement in the Mixed-Initiative Activity Plan Generator (MAPGEN), a mixed-initiative planner-scheduler introduced for the Mars Exploration Rover (MER) mission [51]. Similar systems have been developed for the Phoenix and Mars Science Laboratory (MSL) missions.

The planner-schedulers of Deep Space 1, MER, Phoenix, and MSL consider constraints. Constraint consdieration is important for view trajectory planning in the feature modeling case. The priors handle time, which is also critical for planned-view-trajectory modeling, but they tend to have a large degree of temporal concurrency. This temporal concurrency is not present in view trajectory planning for pit modeling - at least not in the OPTWIND formulation. As far as the OPTWIND is concerned, the rover does only one thing at a time, either viewing or driving. It could be argued that the rover used for pit modeling will have to do planning and scheduling with temporal-concurrency for the same reasons the Mars rovers do. For example, a rover may have a requirement to heat the camera to operating temperature before taking images, and it may do that while driving to a viewpoint position. However, the basic OP is already NP-hard (when cast as a decision problem). As the number of view targets, positions, and time increments grows, planning view trajectories for pit modeling will become quite computationally complex. Trying to solve this problem concurrently with a large set of temporal concurrency constraints is probably not wise. Just as the Mars rovers handle activity planning and path planning separately, detailed activity planning could be done for each drive segment between viewpoints, once the set of viewpoints and times are known.

Many AI planners use the Planning Domain Description Language (PDDL) to represent planning domains and problems. This is the language used by the International Planning Competition, so it is widely used among general-purpose planners. Although frequent users of PDDL often refer to the domain description as a 'model', this thesis will exclusively use the word 'domain' to avoid confusion with 3D terrain models. PDDL has evolved over the years to include more and more capabilities. Originally developed to deal with Booleans, for example (at ?roverA ?locationB) would be true if rover A was currently at location B, and false otherwise, it was expanded in PDDL 2.1 to include numeric expressions and plan metrics [52]. Durative actions - actions that take time - were also added with 2.1.

Durative actions would be important in the terrain feature modeling case, because rover travel between locations takes time. Plan metrics could be used to improve the quality of the planned model, or to minimize distance. The use of numeric expressions could also be useful for representing view and illumination angle constraints. In version 2.2, timed initial literals were added [53]. These enable the representation of variables that change state based on time, rather than solely due to the planning agent's actions. PDDL 3 added soft goals (or preferences) and trajectory constraints [54]. Soft goals could be used to represent that it is preferred that all faces of a terrain feature are viewed, but plans can still achieve partial success if they do not view all faces. PDDL 3.1, the current version, added object fluents, which are not expected to be needed for terrain feature modeling.

Many planner s work with PDDL 3.0, including SGPlan 5 and SGPlan 6 [54, 55]. SGPlan 6 builds from SGPlan 5 and is set up to solve both temporal and non-temporal planning problems [55]. Problems may have soft goals, derived predicates, and ADL features. Both SGPlan 5 and SGPlan 6 generated successful plans for a very simplified version of the terrain feature modeling problem, as described in Section 6.3.3.1.

AI planning and scheduling offer algorithms for general planning problems. Before using these tools, however, a real-world problem (selecting what to image, from where, and at what time to build a good terrain feature model) must be formalized into a problem that the planner/scheduler can understand. If all the key elements of the real-world problem are not represented in what the planner sees, or if the planner is not set up to address these key elements, the planner is not likely to produce a good solution. For example, potential locations and transit times are easy to represent in PDDL, but inter-node dependencies are not.

2.3 Model Building

An important part of planned-view-trajectory model building is understanding how illumination and view angles affect model quality. This work looks at stereo reconstruction as a model-building method.

For stereo reconstruction, two very similar view angles to a target will produce poor-quality depth information, but it may be difficult to match features between images from two very different view angles. So for stereo, there are both minimum and maximum constraints on relative view angle to get a good quality reconstruction. Matching features between images can also fail if the illumination angles in the two images are too different.

To reconstruct the optical properties of terrain, at least 4 distinct measurements are required (given a Torrance Sparrow model for the BRDF [56]), and these measurements should be sufficiently different in the space of possible illumination and view angles. This means that images at different lighting angles would be needed.

Stereo reconstruction has been done from orbit in planetary missions. In this case, the difference in view needed for stereo is achieved by moving and tilting the spacecraft. One stereo reconstruction method for orbital images is the open-source Ames Stereo Pipeline [57, 58]. Ames Stereo Pipeline is used to construct coarse prior models of a pit in Section 6.1.

Stereo from orbit is also similar to the consideration of stereo in this terrain feature modeling work, and can be contrasted against parallel stereo that achieves a difference in view based on camera separation perpendicular to the view direction. The larger the distance is between the cameras and the view target, the wider this separation, or baseline, must be. Stereo between multiple robot positions is used for terrain feature modeling because of the distances involved in viewing across pits or canyons.

The modeling of Victoria Crater by the MER Opportunity is an example of how planetary rovers currently model terrain features [59, 60]. Waypoints for Opportunity were chosen by operators on Earth instead of being planned autonomously. Both MER and MSL carry multiple stereo pairs of cameras, so stereo (over a short baseline) is commonly used. Bundle adjustment is also used for these rovers, to register image sets over the course of a rover trajectory [61].

Depending on the desired spatial resolution of the final model, a terrain feature model could be composed of many images. An effective pipeline for modeling large features with large numbers of images on Earth consists of open source Bundler [62, 63] for bundle adjustment (structure from motion), and CMVS/PMVS multi-view stereo software [64, 65]. Their work has been demonstrated in reconstruction of tourist destinations like the Colosseum in Rome and the Piazza San Marco in Venice using thousands of tourist photos. VisualSFM is another option for structure from motion that can also be used with CMVS/PMVS [66, 67]. Bundler, VisualSFM, and CMVS/PMVS are all used in reconstruction of planetary analog terrain features in this thesis, as discussed in Section Section 6.4. Other modeling software options used or evaluated in the thesis work include OpenMVG and Multi-View Environment (MVE) [68, 69]. Further discussion of these model building methods can be found in 6.4.1.

Another key part of automated model building is how features (here meaning keypoints or interest points as opposed to the "terrain features" in this document) are detected and described for matching between images. This work uses SIFT for feature detection and matching [70]. SIFT tends to be fairly robust to scale variation (changes in viewing distance), some changes in view angle, and changes in overall image brightness. Feature detection and matching methods generally do not consider change in illumination angle from a directional light source. Even for humans, illumination angle variations can cause differences in how well terrain formations can be identified on planetary surfaces [71]. One of the goals of this thesis work is thus to better understand how directional lighting impacts feature detection and matching.

Chapter 3

Planned-View-Trajectory Model Building

Planned-view-trajectory model building takes inputs: terrain data, time information, illumination trajectories, and robot restrictions, converts them into a planning problem, plans a view trajectory that ensures view and illumination angle limits are obeyed, executes the planned view trajectory to capture images, and constructs a detailed model from the captured images. The process is shown in Fig. 3.1.

View trajectories consist of a sequence of view targets (what to image), sensor positions (where to image from), and times (when to image). In order to plan view trajectories, real-world inputs must first be translated into a format that enables a planner make decisions based on parameters, (view angle, illumination angle, viewing distance, and relative view and illumination angles between images), that affect model quality. Inputs include a terrain data (such as orbital imagery or LIDAR elevation models), information about the time (start and end) during which the modeling task must be completed, the trajectory of illumination direction over time for the terrain feature to be modeled, and robot operating restrictions (such as maximum slope, maximum traversable obstacle size, or minimum approach distance due to contamination concerns). Potential robot positions may also be provided as input.

From inputs, problem setup generates a coarse prior model of the terrain, discretizes this coarse model into planar faces, discretizes time, computes the illumination angles on each face at each discretized time, finds a set of robot positions (if this set is not given as input), computes which faces can be viewed from which robot positions as well as associated view angles, and computes path lengths between robot positions. Part of the problem setup also involves setting a value function that gives a value for a plan based on the predicted value of the detailed model built from the planned images.

Planning determines a view trajectory. It seeks to optimize the value function. Execution of the view trajectory happens when a robot on a planetary body captures images for all the views in the view trajectory, though in this thesis work, view trajectory execution was simulated by selecting subsets of images of planetary analog features.

The final part of the planned-view-trajectory model building process is the construction of the detailed model. This is done using structure from motion techniques to build a

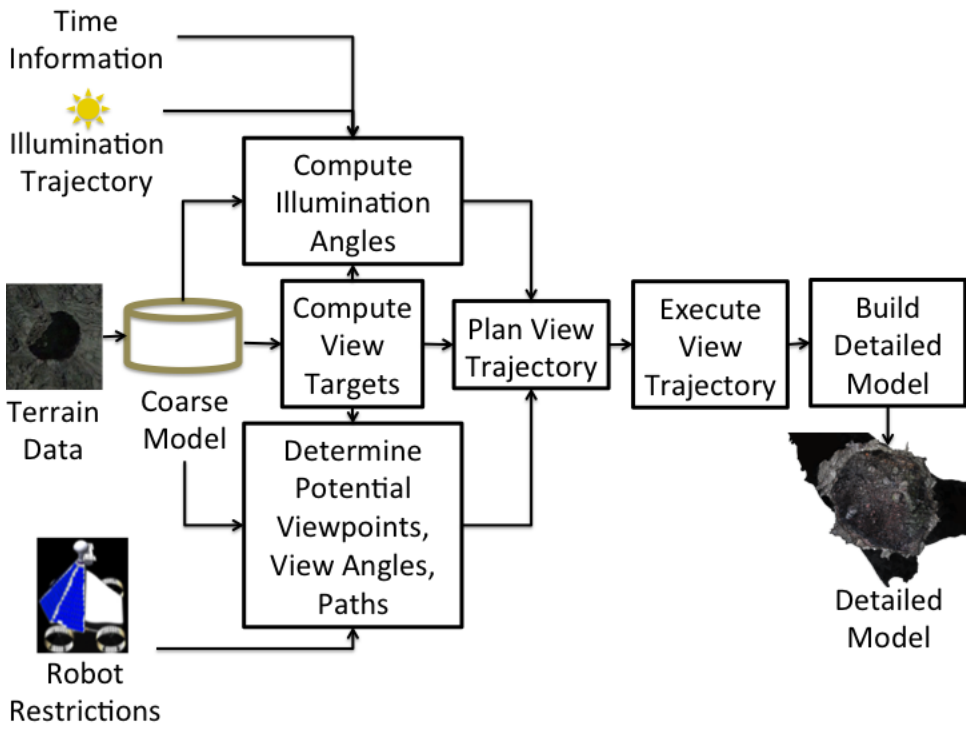


Figure 3.1: The process for planned-view-trajectory model building

sparse reconstruction and then dense reconstruction techniques to build the detailed model. Meshing techniques may optionally be applied to convert a point cloud from the dense reconstruction into a surface model.

In the following sections, a formulation of planned-view-trajectory model building is presented, using a simple model of a skylight pit. More detail for each part of the process is presented in Chapter 6.

3.1 Input

Terrain data in the simple example case consists of a single orbital image of a skylight pit (see Fig. 3.2). The dimensions of the pit can be estimated based on the image resolution, and the depth can be estimated from the shadow position.

Time information consists of the length of daylight - for the Moon this is 14.77 Earth days, or about 354 hours - as well as start and end times for the modeling mission, and a desired number of divisions to consider for the mission period. This last is based on the insight that, while illumination changes continuously with time, over the time scales required to take a single view, which may be less than a second up to a minute or more for high dynamic range imaging, illumination does not change that much. The number of time divisions should be set such that within each time division, the illumination can be considered essentially the same for purposes of planning. This could mean that the illumination angle on a face of terrain changes less than some threshold (5 degrees, 20 degrees, etc.), or that the set of

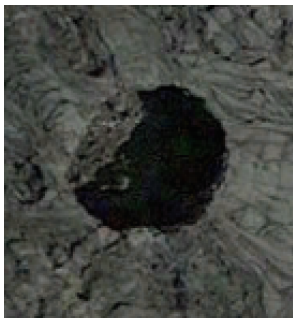


Figure 3.2: An image of a skylight pit [1].

illuminated faces does not change more than some threshold, which in turn depends on face size.

The illumination trajectory is computed from time information and a given latitude. The sun is assumed to travel 360 degrees relative to the pit in one day length, in a plane that is latitude degrees from vertical. One light vector is computed for the center of each time division.

Restrictions on rover travel include a maximum distance traveled in a single time division and a minimum distance that a rover must maintain from the pit edge. The upper bound on maximum distance traveled is set by the maximum rover speed, but the value is set much lower for this work to account for other un-modeled mission considerations.

3.2 Problem Setup

The first step in problem setup is to convert the terrain data into a coarse model of the feature. This coarse model could be as simple as a cylinder projected down to the depth of a feature (for a pit) or its height (for a rock), which can be estimated from shadows. Of course, more accurate prior models enable better prediction of view and illumination angles, so more accurate coarse models are preferred. If elevation models from orbital LIDAR or stereo exist, they can be trimmed down to create the coarse model of the feature. If multiple orbital images of the feature exist, stereo or multi-view stereo models can be specifically created to get the coarse model for planned-view-trajectory model building. Here, the prior coarse model of pit geometry consists of a cylinder with a radius and height (see Fig. 3.3). The cylinder represents pit walls, and pit floor is not considered.

Before planning, the coarse a priori model, illumination, timing, and rover travel restrictions must be converted into a formally-defined view trajectory planning world, $\mathcal{W}$. The view trajectory planning world $\mathcal{W}$ is defined as a tuple $(\mathcal{T}, \mathcal{L}, \mathcal{P})$, where $\mathcal{T} = t_1, t_2, ..., t_N$ is the set of light times in the mission timeframe, $\mathcal{L} = l_1, l_2, ..., l_M$ is the set of locations that a rover can take views from, $\mathcal{P} = p_1, p_2, ..., p_Q$ is the set of target faces. The functions $\alpha\left(l_i, p_j\right)$, $s(t_i, p_j)$, and $d\left(l_i, l_j\right)$ are defined in this world, where $\alpha\left(l_i, p_j\right)$ is the view angle from rover position l_i to face p_j, or ∞ if p_j is not visible from l_i, $s(t_i, p_j)$ is the solar illumination angle of face p_j at time t_i, and $d\left(l_i, l_j\right)$ is the path distance from location l_i to location l_j.

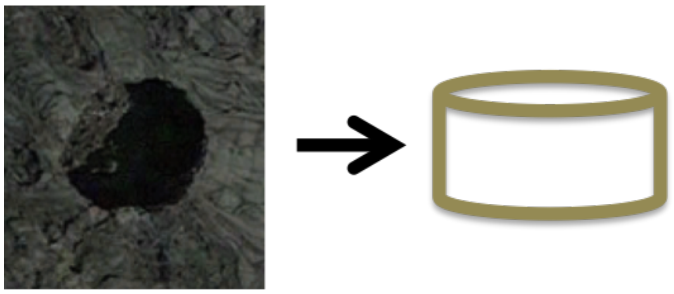


Figure 3.3: A skylight pit is modeled as a cylinder [1].

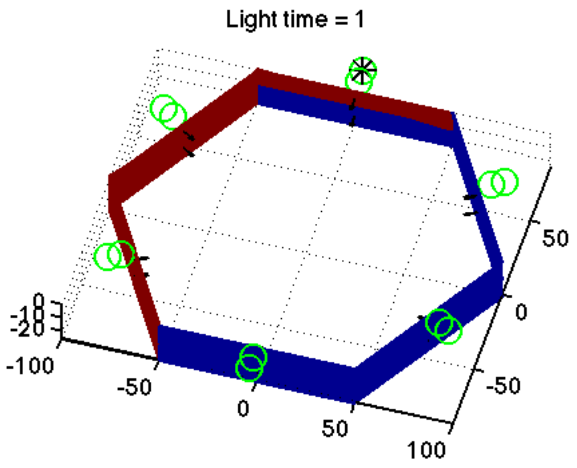


Figure 3.4: Coarse pit model discretization with 2 faces vertically and 6 faces around the cylinder. Potential rover positions are shown as green circles.

The pit surface is discretized into a set of planar faces of uniform width and height, depending on a specified number of divisions vertically and angularly around the cylinder. The face width and height are computed based on the number of divisions. Normal vectors and positions for the center of each face are computed (see Fig. 3.4 for a coarse surface area discretization and Fig. 3.5 for a finer discretization).

For each light vector (one for each time division), a set of potentially lit faces is computed, such that the angle between the light vector and the face normal is less than 90 degrees. Each potentially lit face is then checked against the set of faces that could potentially occlude it to determine if it is shadowed. Only the center of the face is considered for this illumination computation. For faces lit at a given time, information about the illumination angles is stored. An example of lighting for a pit at 45 degrees north latitude that is 100m deep and 87m in radius can be seen in Fig. 3.5, with red indicating lit faces and blue indicating unlit faces.

While the illumination angle relative to a face normal can be represented by a single number, at least two numbers are required in the representation to enable computation of relative illumination angles in 3D. For example, for a given set of illumination vectors $\hat{i}_1$ and $\hat{i}_2$ with the same angle relative to the face normal s, if they point in the same direction,

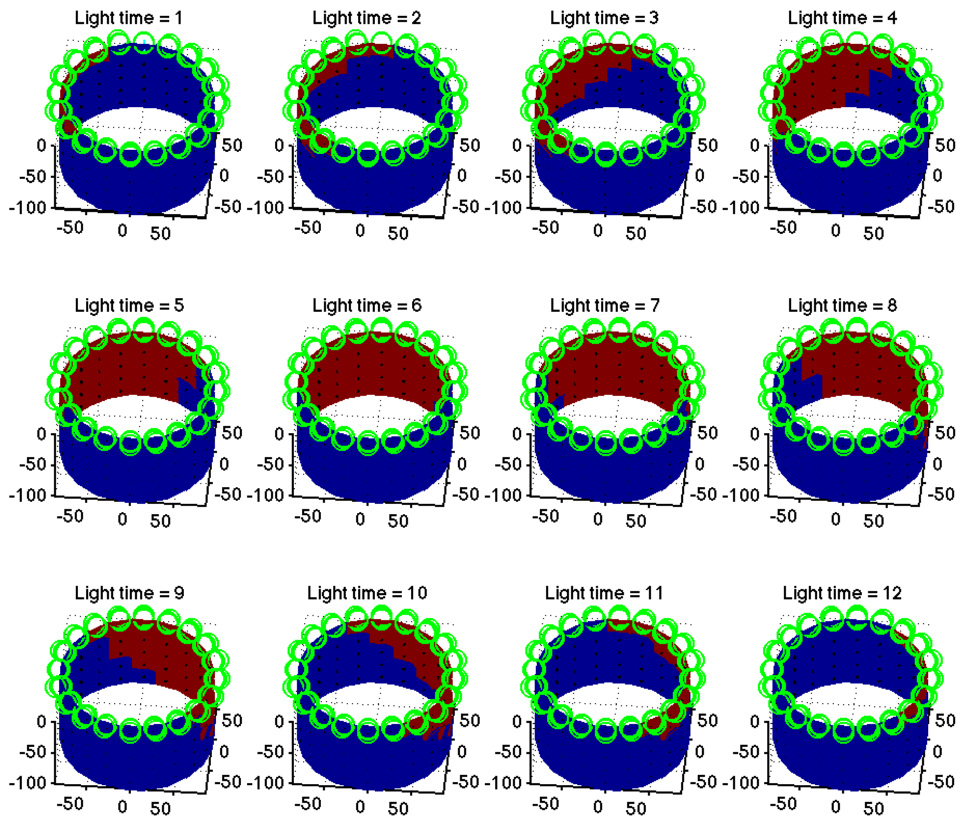


Figure 3.5: Face discretization of a pit with 5 vertical and 20 angular divisions and 12 time divisions, showing a change in lighting over time. Red indicates that a face is lit, and blue indicates that it is not lit.

the relative illumination angle will be zero. If $\hat{i}_1$ is rotated by positive s about x axis from the face normal and $\hat{i}_2$ is rotated by $-s$ about the x axis, the relative illumination angle will be $2s$. If the two vectors are rotated about different axes, the relative illumination angle will be something else. Two values, (which might represent rotation about x and rotation about y, or azimuth and elevation), would be sufficient to reconstruct a unit vector for illumination direction. Illumination angles in x and y could also be compared directly, ensuring that neither angle goes above (or below) the threshold value. This approach avoids doing trigonometric computations during plan time, but thresholds may have to be set more conservatively. For example, the vectors $[0, -0.707, 0.707]$ and $[0.707, 0, 0.707]$ would have angles $(45, 0)$ and $(0, 45)$, respectively. If the threshold on maximum relative angle is 45 degrees in x and y, this pair of illumination conditions would pass. However, the angle between these two vectors is actually 60 degrees. The maximum thresholds on relative angle in x and y would have to be set lower than 45 degrees to ensure a relative angle of 45 degrees or less. For simplicity of presentation, illumination is represented by a single angle in this section, with the understanding that actual implementation would need to compare both x and y angles or directly compute the relative angle between two illumination vectors.

Potential robot positions may be provided as input, generated to cover the set of potential viewpoints in a systematic way (i.e., on a grid or in a ring around a pit), or synthesized from the set of view targets. This example shows synthesis from a set of view targets. Positions are computed based on views that look directly across the pit (see Fig. 3.4). The distance from the pit edge is computed as the maximum distance at which the rover can see a certain number of vertical faces. For example, if the pit had been discretized into 3 divisions vertically, there would be three rover positions directly across from each angular division. The farthest away from the pit edge would see only the top face in the angular division directly across the pit. The middle position would see the top two faces. The position closest to the pit edge would see all three faces vertically. Only rover positions that are farther from the pit edge then the specified minimum approach distance will be added. View angles for each rover position to view each face center are computed (if the face is not visible, the angle is set to infinity). Both horizontal view angle and vertical view angle are computed (equivalent to rotation about x and y in the discussion of illumination angle above).

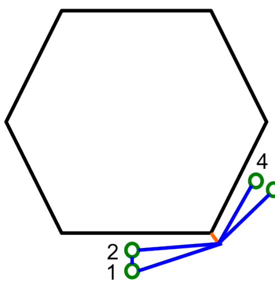


Figure 3.6: Calculating paths between positions in the simple pit model. Paths from positions 1 to 2, 1 to 3, 1 to 4, 2 to 3 and 2 to 4 are computed, and are assumed to be bidirectional. Each path cannot approach the pit edge closer than the minimum approach distance.

Path lengths are computed between rover positions across from the same face, and from the set of positions across from one face to the set of positions across from the adjacent face, wrapping around such that paths between the last and first set of positions considered will also be computed. Paths are straight lines, except that the closest approach to the pit edge cannot be closer than the specified minimum approach distance, so paths between positions across from adjacent faces are made up of two straight lines (see Fig. 3.6). Paths are assumed to be bidirectional. Dijksra's algorithm is run on a graph with position nodes and path-length edges to compute the distance from each position to each other position [72].

3.3 Plan Value

When the end-goal of planning is to create a high-quality model of a pit, it makes sense to evaluate plans based on the quality of the resulting model. The quality of the resulting model cannot be evaluated without first executing the plan, capturing the images, and building the model, so when comparing potential plans, an estimate of model quality is used.

The estimate of model quality is based on illumination and view angle constraints. For an image of a face to add value to the model, it must be lit within the illumination angle thresholds, here set to 0 degrees minimum and 90 degrees maximum. An absolute view angle constraint of 45 degrees is imposed, and images at more oblique angles do not add value. Stereo is the assumed method of 3D reconstruction, so both minimum and maximum constraints on relative view angle are used. In this example, the minimum is 20 degrees and the maximum is 45 degrees. When developing a plan value function for a real terrain feature modeling mission, illumination and view angle thresholds should be set as described in Chapter 4. For each face, a score of 0.5 is given for a single image that meets constraints. A score of 1 is given if there is a pair of views that meet both absolute and relative constraints. The sum of per-face scores is computed, and the model value is the percentage of the maximum possible score that this sum achieves. This takes into account that some faces may never be visible, or may never be sufficiently lit, reducing the maximum possible score in these cases.

Using the definition of the view trajectory planning world described in section 3.2, the view trajectory $\mathcal{V}$ can be defined formally as an ordered set of views, $\{v_1, ..., v_N\}$ such that each view is a tuple $v_n = (t_i, l_j, p_k), t_i \in \mathcal{T}, l_j \in \mathcal{L}, p_k \in \mathcal{P}$. Illumination and view angle constraints are defined with A, the absolute view angle threshold, R_{min} and R_{max}, the relative view angle thresholds, I, the absolute illumination angle threshold, and the following functions:

$$a\left(l_i, p_j\right) = \begin{cases} 1, & \text{if } \left|\alpha\left(l_i, p_j\right)\right| < A \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

The function in eq. 3.1 returns 1 if the view angle from location l_i to face p_j has magnitude less than the absolute view angle threshold, in other words, the rover is not observing the face from too oblique an angle.

$$r\left(l_i, p_i\right) = \begin{cases} 1, & \text{if } \exists l_j, p_j : \left|\alpha\left(l_i, p_i\right) - \alpha\left(l_j, p_j\right)\right| > R_{min}, \\ & \left|\alpha\left(l_i, p_i\right) - \alpha\left(l_j, p_j\right)\right| < R_{max}, j < i \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

The function in eq. 3.2 is used if a face has been viewed before the current view. If it has, and if the difference in view angle meets the relative view angle thresholds (both minimum and maximum), then the function returns 1.

$$b\left(t_i, p_j\right) = \begin{cases} 1, & \text{if } \left|s\left(t_i, p_j\right)\right| < I \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

The function in eq. 3.3 checks whether the illumination angle has magnitude less than the illumination angle threshold. This means the light is not coming in at too much of a glancing angle.

To express the concept of a first view of a face, where only absolute constraints apply, the following function is added.

$$f\left(l_i, p_i\right) = \begin{cases} 1, & \text{if } \nexists l_n, p_n, t_n : a\left(l_n, p_n\right), b\left(t_n, p_n\right), \text{and } n < i \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

In the Orienteering Problem, it is enough to say that a node has been visited one time or zero times. If visited once, the value for that node is added to the total score, and if visited zero times, it is not. Visiting a node more than once does not accrue additional value. For terrain feature modeling, value is gained from viewing faces. Additional value can be gained from viewing a face multiple times with different view and illumination angles, if these angles meet constraints, but once the desired number of views of a face have been collected that do meet relative angle constraints, additional views do not add value. The following function is used to compute a multiplier that is one if a pair of views, (location, time, face triples), meeting angle constraints have not yet been captured and zero if they have been captured.

$$g\left(l_i, p_i\right) = \begin{cases} 1, & \text{if } \nexists l_n, p_n, t_n, l_m, p_m, t_m : \left|\alpha\left(l_n, p_n\right) - \alpha\left(l_m, p_m\right)\right| > R_{min}, \\ & \left|\alpha\left(l_n, p_n\right) - \alpha\left(l_m, p_m\right)\right| < R_{max}, \\ & b\left(t_n, p_n\right), b\left(t_m, p_m\right), n < i, \text{and } m < i \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

The maximum possible score for the model is computed as the sum of the maximum possible scores for each face (here defined as 1), over all faces that are visible within the absolute view angle threshold for some rover position in $\mathcal{L}$ and lit within the absolute illumination threshold at some time in $\mathcal{T}$.

$$J_{max} = \sum_{p_j \in \mathcal{P}} \max_{l_i \in \mathcal{L}} \left(a\left(l_i, p_j\right)\right) * \max_{t_k \in \mathcal{T}} \left(b\left(t_k, p_j\right)\right) \tag{3.6}$$

The score for the model from a given view trajectory is defined as:

$$J\left(\mathcal{V}\right) = \sum_{i=1}^{N} \left\{0.5 * b\left(t_i, p_i\right) * a\left(l_i, p_i\right) * \left[f\left(l_i, p_i\right) + r\left(l_i, p_i\right) * g\left(l_i, p_i\right)\right]\right\} \tag{3.7}$$

And the model value is defined as:

$$V\left(\mathcal{V}\right) = \frac{J\left(\mathcal{V}\right)}{J_{max}} * 100\% \tag{3.8}$$

In addition to the estimated model value, plans might also be evaluated on how efficient they are - for example, how far does the rover have to travel to take all the images in the order specified in the view trajectory plan, and how many images are taken? There is also the concept of risk. If the rover could fail at any point due to the harsh planetary environment, then it makes sense to try to take more images earlier in the mission. If it is more likely to fail while driving, by getting stuck or lost, than while sitting still, then it makes sense to minimize distance traveled.

In this work, a time-based failure heuristic and a distance-based failure heuristic are used. The time-based failure heuristic uses a constant, λ_t, and the distance-based failure heuristic uses a constant, λ_d. These constants are then multiplied by time and distance, respectively, in a negative exponential relationship to get a 'probability' that a view will take place, given the view time and the distance traveled to get to the view position (see eq. 3.9 and eq. 3.10).

$$P\left(v_i|t_i\right) = e^{-\lambda_t t_i} \tag{3.9}$$

$$P\left(v_i|\delta_i\right) = e^{-\lambda_d \delta_i} \tag{3.10}$$

The current distance traveled, δ_i, is computed as $\delta_i = d\left(l_{start}, l_1\right) + \sum_{j=2}^{i} d\left(l_{j-1}, l_j\right)$. Note that for simplicity, the time and distance heuristics are assumed to be independent.

Using these heuristic 'probabilities', the value of a view-trajectory plan can then be computed as the 'expected value' of a model, given the views, the times they are taken, and the distance traveled between views. The expected value of a view trajectory score is:

$$\mathbb{E}\left[J\left(\mathcal{V}\right)\right] = \sum_{i=1}^{N} \left\{P\left(v_i|t_i\right) * P\left(v_i|\delta_i\right) * \\ \left[0.5 * b\left(t_i, p_i\right) * a\left(l_i, p_i\right) * \left(f\left(l_i, p_i\right) + r\left(l_i, p_i\right) * g\left(l_i, p_i\right)\right)\right]\right\} \tag{3.11}$$

Because the total possible model score, J_{max} is not dependent on time or distance traveled, the expected model value can be computed as:

$$\mathbb{E}\left[V\left(\mathcal{V}\right)\right] = \frac{\mathbb{E}\left[J\left(\mathcal{V}\right)\right]}{J_{max}} * 100\% \tag{3.12}$$

The 'expected value' computed using failure rate heuristics is used in Section 6.3.2. This type of heuristic would be hard to represent in the PDDL formulation used in classical AI planning.

3.4 Planning View Trajectories

An example planned view trajectory is given in Fig. 3.7. In this case, the pit has been discretized into 6 faces. Faces and positions are numbered counterclockwise from 1 at the top. All faces are visible from some rover position, but the bottom face, 4, is never lit, so the total possible score, J_{max}, is 5. In the first time division, the rover views face 3 from position 1. The face meets the lighting constraint, $\left(b\left(1,3\right) = 1\right)$, the view angle of 30 degrees meets the absolute view angle constraint, $\left(a\left(1,3\right) = 1\right)$, and there are no other prior views of this face in the view trajectory, $\left(f\left(1,3\right) = 1\right)$, so a score of 0.5 is gained for this face, giving a total model value of $V\left(\mathcal{V}\right) = \frac{0.5}{5} * 100\% = 10\%$. In light time 2, the rover views faces 1, 2, and 3 from position 5. All these faces are sufficiently lit, and all these views meet the absolute view angle constraints. For faces 1 and 2, this is the first view, so they each add a score of 0.5. The difference in view angle between this image of face 3 and the previous image of face 3 is 60 degrees, so it does not meet the relative view angle constraint, $\left(r\left(5,3\right) = 0\right)$, and does not add to the score. The total model value at this point

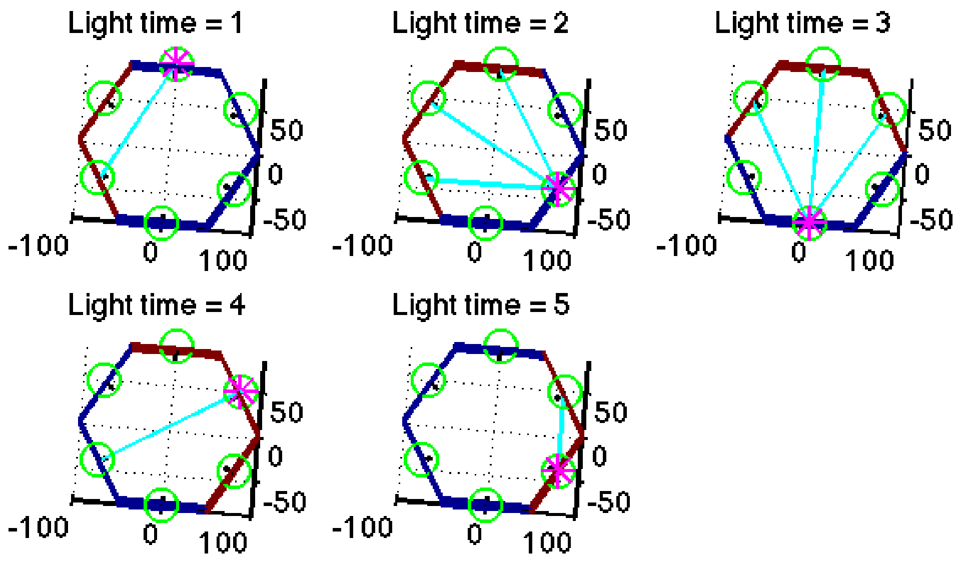


Figure 3.7: Example of a view trajectory. Asterisks indicate rover positions, and cyan lines indicate views.

is $V(\mathcal{V}) = \frac{0.5+0.5+0.5}{5} * 100\% = 30\%$. At light time 3, the rover moves to position 4 and captures 3 images that are sufficiently lit and meet absolute view angle constraints. The views of faces 1 and 2 also meet relative view angle constraints with previous images, so the total model value is now $V(\mathcal{V}) = \frac{1+1+0.5+0.5}{5} * 100\% = 60\%$. At light time 4, the rover moves to position 6 and images face 3. This image meets absolute view angle constraints and relative view angle constraints with the image taken in light time 1, but the face is not lit, so no additional value is gained. At light time 5, the rover moves to position 5 and images face 6. The face is sufficiently lit, but the view angle of 60 degrees does not meet the absolute view angle constraint, so no additional value is gained. In contrast, the view trajectory shown in Fig. 3.8 achieves 100% value for the same pit scenario. The goal of view trajectory planning is to maximize this value score.

Section 6.3 goes into more detail on how view trajectory planning might be done.

3.5 View Trajectory Execution and Detailed Model Construction

Once a view trajectory is planned, it can then be executed. This means a robot travels around and takes images of the planned views. For the simple skylight pit example, images might look like those in Fig. 3.9.

After images are acquired, they are used to build a detailed model of the terrain feature. Fig. 3.10 shows what this model might look like for the skylight pit example.

Each part of the planned-view-trajectory modeling process is discussed in more detail in Chapter 6. This includes identification and testing of existing tools to solve parts of the

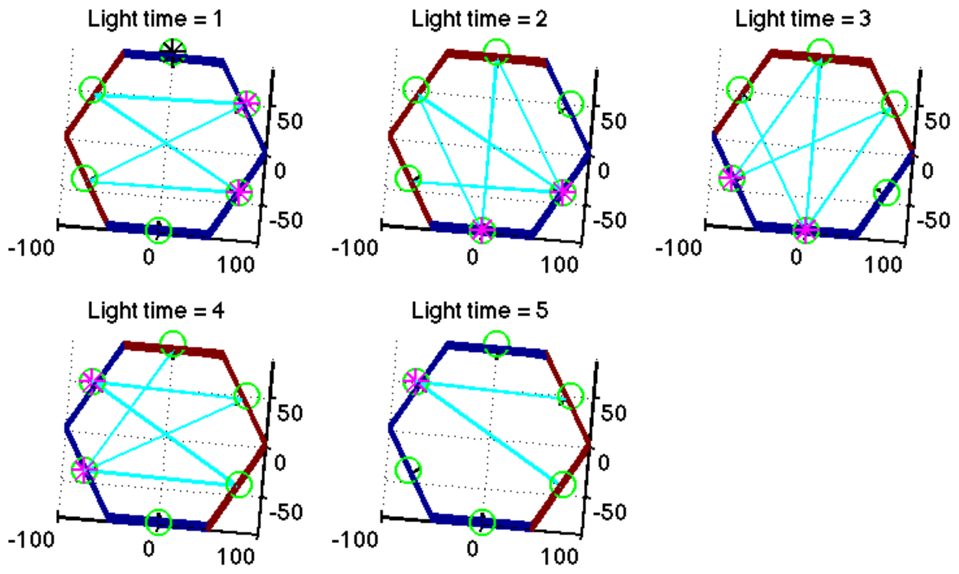


Figure 3.8: Example of a planned view trajectory that achieves 100% value. Black asterisk indicates starting position. Magenta asterisks indicate rover positions, and cyan lines indicate views.

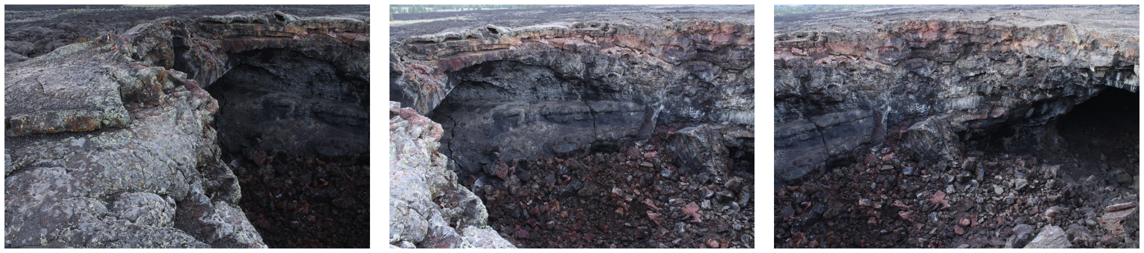


Figure 3.9: Execution of a view trajectory for the simple skylight pit example might capture images like these.

process, and development of new solutions for parts where existing tools are not available or appropriate. Planned-view-trajectory modeling is demonstrated in Chapter 7.

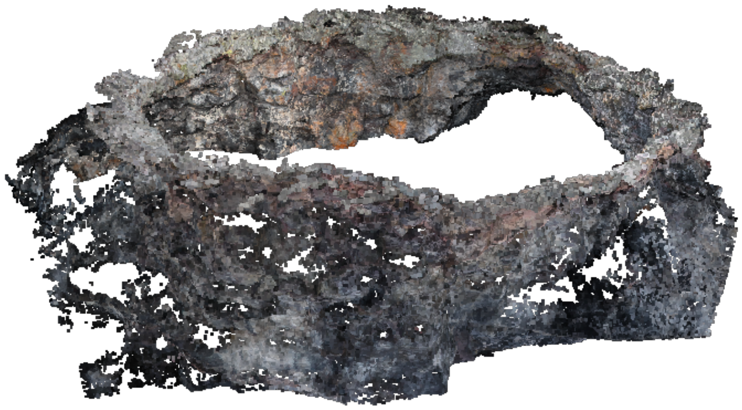


Figure 3.10: The detailed model built after view trajectory execution for the simple skylight pit example might look something like this.

Chapter 4

What Makes a Good Model

Not all models of planetary features are alike. A model of a pit derived from a few satellite flyover images has no chance to achieve the coverage, resolution, or reconstruction accuracy of a model created by days of imaging by a surface rover circling the same pit. This chapter distinguishes metrics that contribute to good models of planetary features. It identifies factors that affect these model quality metrics and explores how limits can be set on these factors to ensure model quality.

4.1 Model Quality Metrics

Model quality metrics include:

- Surface coverage, or the percentage of the terrain feature surface area that is covered by at least one image
- Surface resolution, or the width/height of an area of the terrain feature covered by a single image pixel. This might be measured in meters per pixel.
- Visibility of features, or whether surface features are visible and detectable in images, not too dark to distinguish or too washed out
- 3D coverage, or the percentage of the terrain feature surface that is covered by a 3D reconstruction
- 3D resolution, or the width/height/depth of a terrain feature covered by a single voxel in a 3D reconstruction. This might be measured in meters per voxel.
- Reconstruction accuracy, or how close the 3D reconstruction is to the 3D surface of the feature
- Model continuity, or to what extent are images/3D reconstructed regions isolated versus adjacent

The relative importance of various model quality metrics will depend on the desired use of the model. If the model is to be used for an immersive fly-through, getting high-percentage, uniform, contiguous coverage would be important. For planning robotic rappel into a pit, high-resolution 3D coverage of a narrow region would be important.

4.2 Factors that Affect Model Quality Metrics

For planning views, it is important to understand what factors, among those which can be varied at plan time, affect various model quality metrics. It is assumed that by plan time, a rover that will model the terrain feature has already been designed, so the camera resolution (the number of pixels in the camera sensor) is fixed and the camera field of view is fixed (field of view could be adjusted somewhat with a zoom lens, but it would still be constrained to a limited range).

The amount of a terrain feature's surface that is covered in a set of images depends on the viewing distance, the pointing of the camera for each image, and the total number of images.

The resolution depends on the angular resolution of the camera, but for a fixed field of view and camera pixel resolution, it is determined by viewing distance and the viewing angle. For a given desired resolution with a fixed camera field of view, there is a maximum viewing distance. The set of potential views considered by a view trajectory planner can be restricted to only those within the maximum viewing distance.

The ability to detect terrain surface features in an image depends on the resolution (viewing distance and viewing angle) and on the illumination angle. If the surface is too dark, there may not be enough light to distinguish features. If the lighting is too direct, features may appear washed out. While this is also partly a function of exposure time, with surfaces that have little color variation and get all their surface texture from shape, illumination that is too close to straight-on can make it impossible to see features, (see Figure 4.1), and illumination that is too tangential can cause nearly all of the surface to be shadowed.

Coverage and resolution in 3D depend on the same factors as coverage and resolution in 2D, as well as the ability to do 3D reconstruction. In this work, it is assumed that multi-view geometry is used for 3D reconstruction. In that case, the reconstruction depends partly on the geometry of the problem and partly on the number of features successfully matched between pairs of images. The number of features successfully matched between images depends on the initial number of features detected in each of the images and the similarity in appearance of the two images. If the relative view angle or relative lighting angle between the two images is too different, the images will look too different: the same features will not be detectable in the two images.

To summarize, the factors affecting model quality are:

- Viewing distance
- View angle

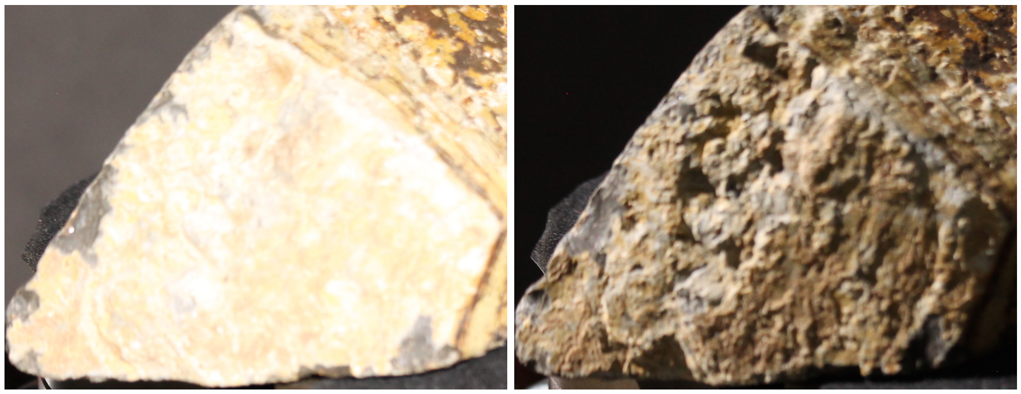


Figure 4.1: Rock face under two different illumination conditions. The illumination vector relative to the face normal is 15 degrees in the left image and 80 degrees in the right image. Using automatic feature detection, 40 features were detected on the face in the left image, and 155 were detected on the face in the right image [2].

- Illumination angle
- Relative view angle
- Relative illumination angle

Each of these factors are discussed in detail in Section 4.4.

4.3 Representing Angles Relative to Terrain and the Planar Face Assumption

For each of the innumerable points on a terrain surface, a normal vector could be computed that would represent the surface orientation. View and illumination angles could then be computed relative to these normal vectors. For planning view trajectories, however, it is not realistic to consider each surface point in the planning process. The location of each point on the surface is not known *a priori*, only a coarse representation of the surface is available, and the ultimate goal is to build a higher-resolution model. Even using only surface points as available in the prior model, though, considering each surface point would result in an infeasibly large planning problem. Instead, this work takes the approach of discretizing the prior model surface into planar 'view target faces' with normal vectors and limited extent.

A local frame can be created for each face, with the z-axis along the normal vector, and the x and y-axes in the plane of the face. In this work, view and illumination angles are represented in these local frames. A rover may move in a global frame, but the global illumination angle is not enough to determine the effect of illumination on local faces. Likewise, the angles of a rover's view vector with respect to to the global frame are not enough to determine how obliquely the rover will view an individual patch (see Fig. 4.2).

All view and illumination angles in this work are given relative to face normals. They may be expressed as a single angle, representing rotation about an arbitrary axis perpendicular

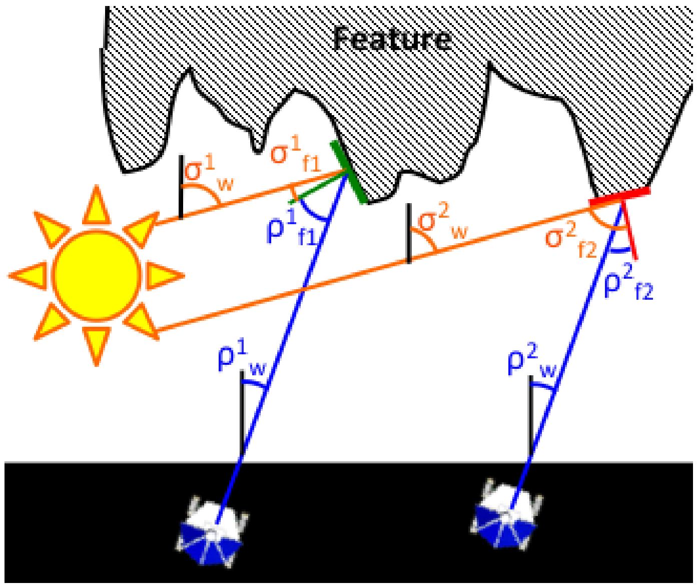


Figure 4.2: View and illumination angles in global and local frames. Note that the view and illumination angles for two views with respect to the world frame are equal, $\rho^1_w = \rho^2_w$ and $\sigma^1_w = \sigma^2_w$, but view and illumination angles with respect to the local frames of two faces, $f1$ and $f2$, are not equal, $\rho^1_{f1} \neq \rho^2_{f2}$ and $\sigma^1_{f1} \neq \sigma^2_{f2}$.

to the face normal, or as a pair of angles, representing rotation about the x and y axes of the face's local frame. The viewing distance will be given from the camera position to the center of a view target face.

The assumption that areas of terrain surface can be discretized into planar faces may be more or less valid depending on the terrain shape and the plane size. One way to quantify at how "planar" a surface is involves looking at the ratio of eigenvalues.

4.4 Understanding the Impact of Factors that Affect Model Quality Metrics

This section examines in more detail how various factors affect model quality metrics. It seeks to develop a geometric intuition for effects and looks at theoretical and/or experimental rationale for setting limits on factors that will facilitate construction of good models. Experimental data comes from the laboratory campaign discussed in Appendix A.

Section 4.4.1 discusses how viewing distance and view angle affect surface resolution and 3D resolution. Section 4.4.2 discusses how view and illumination angle affect detectability of features. Section 4.4.3 discusses how relative view and illumination angles affect 3D coverage. Section 4.4.4 discusses how relative view and illumination angles affect reconstruction accuracy.

4.4.1 Resolution

Surface resolution and 3D resolution are affected by viewing distance and view angle. For a given focal length and camera resolution, the maximum distance d to achieve a target surface resolution (in, e.g., meters per pixel) can be computed geometrically (see Fig. 4.3). For a camera with a fixed focal length and a fixed number of pixels, decreasing the viewing distance makes the surface resolution finer. In many cases where view trajectory planning is applicable, there will be a minimum viewing distance set by the terrain: a rover cannot get closer than distance d because to do so it would have to cross untraversable terrain. If there is a target surface resolution and/or 3D resolution for a model, these can be used to set the maximum viewing distance. For 3D reconstruction done using stereo triangulation from images, (the type of 3D reconstruction considered in this work), the 3D resolution will always be at least as coarse as the surface resolution, and likely coarser. Stereo triangulation could give a depth "answer" for each pixel in an image, but it is unlikely that there is enough texture in the image to unambiguously identify unique features for each pixel; thus, the 3D resolution is likely 3 or more times the surface resolution [73]. So, to get a given 3D resolution, the surface resolution must be at least 3 times finer. Eq. 4.1 gives an equation for the maximum viewing distance, d, given a camera field of view in x and y, a number of pixels across an image in x and y, and a desired 3D resolution, if the view angle is zero. The variable p_{ratio} represents the ratio of image pixels to 3D model points in one dimension (i.e., if a 3x3 square of pixels is needed to create one 3D model point, $p_{ratio} = 3$).

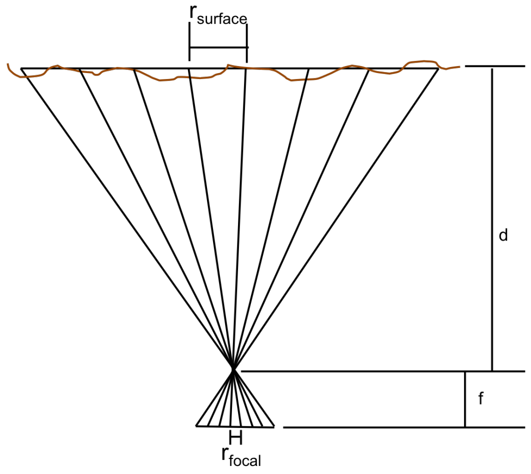


Figure 4.3: (a) A camera with a given focal length f and a given pixel size on its focal plane r_{focal} will have a certain surface resolution $r_{surface}$ at a certain distance, d.

$$d \leq min\left(\frac{\frac{r_{3D}}{p_{ratio}}\,[pixels\ in\ x]}{2tan\left(\frac{fov_x}{2}\right)}, \frac{\frac{r_{3D}}{p_{ratio}}\,[pixels\ in\ y]}{2tan\left(\frac{fov_y}{2}\right)}\right) \tag{4.1}$$

The relationship between resolution and distance is complicated by view angle. The effect of absolute view angle on resolution is visible geometrically. As the view angle increases relative to the normal of the viewed surface, there is increasing variation in surface resolution across the image (see Fig. 4.4). An equation for the maximum viewing distance for a given view angle is given in eq. 4.2. As in eq. 4.1, the minimum distance using either x or y values (for field of view and number of pixels across an image) should be used. An example of this relationship is shown in Fig. 4.5. A maximum view angle can be set based on the expected range of viewing distances. In the example case, if a 3D model of the far wall of a 500-meter-wide canyon is desired at 10cm resolution, then the images used in constructing the 3D model should have view angles of 30 degrees or less.

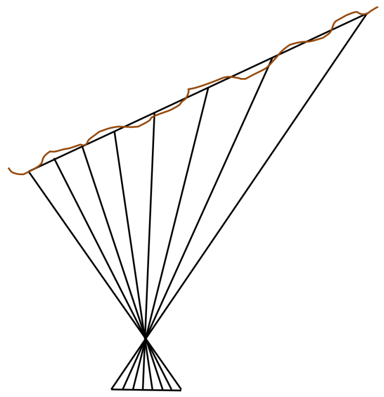


Figure 4.4: When a plane is viewed at a non-zero angle, (relative to its normal vector), the surface resolution varies over the image.

$$
\begin{gathered}
d \leq \frac{\frac{r_{3D}}{p_{ratio}} cos\left(\alpha_{view} + \frac{fov}{2}\right) cos\left(\alpha_{view} + A\right)}{cos\left(\alpha_{view}\right) sin\left(\frac{fov}{2} - A\right)} \\
A = tan^{-1}\left(\left(1 - \frac{2}{pixels}\right) tan\left(\frac{fov}{2}\right)\right)
\end{gathered}
\tag{4.2}
$$

The relationship described in eq. 4.2 is conservative, in that the distance is measured from the center of the camera field of view and the surface resolution is computed for the largest pixel (at the edge of the camera's field of view and thus at a greater distance, given the view angle). If, for a given problem, the size of a face at the expected viewing distance is much smaller than the camera's footprint and the maximum view angle set using eq. 4.2 is too restrictive, it might be advantageous to re-compute a maximum view angle using geometry that includes the concept of face size.

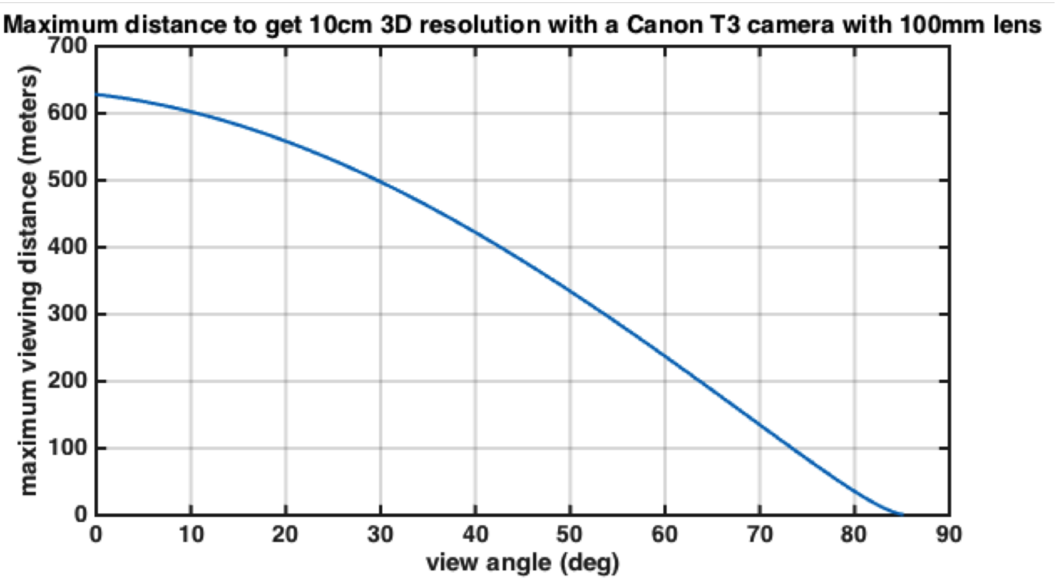


Figure 4.5: Example of the relationship between view angle and maximum viewing distance. This example assumes that a 3D resolution of 10cm is desired with a p_{ratio} of 3. The assumed camera is a Canon EOS Rebel T3 that takes a 4272 x 2848 pixel image. The assumed lens has a focal length of 100mm, giving a field of view of 13 degrees in x and 8.6 degrees in y for the T3 camera.

4.4.2 Visibility of Features

View and illumination angles both affect visibility of features. Aside from the effect of absolute view angle on surface resolution, there are two problems that can occur as the view angle gets larger. The target rock face may start to self-occlude, and part of the camera's field of view may fail to intersect the target face. Both these effects (illustrated in Fig. 4.6) can impact feature visibility.

View angle also affects feature visibility in combination with illumination angle. The phase angle is the angle between the illumination vector and the view vector (see Fig. 4.7). For surfaces that do not have a lot of color (or albedo) variation, most of the visual texture in an image comes from the interaction of illumination, camera, and surface shape. One source of this visual texture is self shadowing, and this becomes less visible at small phase angles (see Fig. 4.8).

Controlled lighting experiments were used to investigate the relationship between feature detection and view and illumination angles (see Appendix A). Features were detected using SIFT [70].

Fig. 4.9 and Fig. 4.10 show results for the lava rock from Appendix A. Fig. 4.9a plots the number of features detected by view angle. Different colors indicate different faces. Note that different faces have different sizes. The maximum number of features detected declines as view angle increases. Fig. 4.9b plots the number of features detected by illumination angle. The maximum number of features detected increases with increasing illumination angle, peaking around 60 degrees, and then decreases toward 90 degrees. Fig. 4.10 plots the number of features detected in 3D with both view and illumination angle. With this plot,

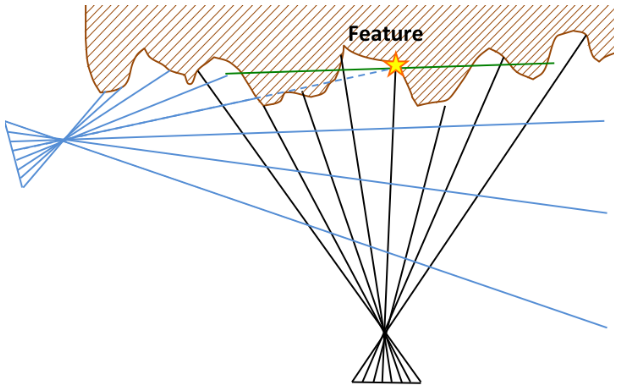


Figure 4.6: As view angles increase, local texture on a rock face can cause self-occlusions, and part of a camera's field of view may not see the target face at all. In the image, two camera views are both planned to observe the same target face (planar approximation is shown in green and the star marks the center), but the camera with the more oblique view angle fails to see the center of the face because its view is obstructed by texture on the rock face. On the lower part of its field of view, it also misses the face completely.

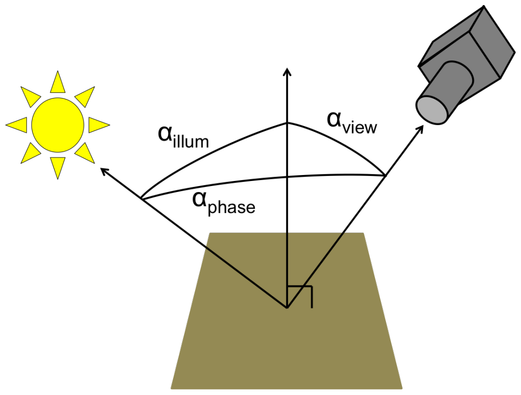


Figure 4.7: The phase angle is the angle between the illumination vector and the view vector. View angle, α_{view}, illumination angle, α_{illum}, and phase angle, α_{phase}, are shown.

it can be seen that the peak in number of features detected at intermediate illumination angles occurs for low view angles, while the number of features detected at high view angles is uniformly low.

Because the size of a rock face affects the number of features detected, plotting the number of features detected for multiple faces is somewhat messy. Different faces can be brought into closer alignment if the number of features is normalized by area. The pixel area in the image that each face takes up was used for normalization, and the results can be seen in Fig. 4.11a, Fig. 4.11b, and Fig. 4.12. (Note that Fig. 4.11a, Fig. 4.11b, and Fig. 4.12 contain data from the same images as Fig. 4.9a, Fig. 4.9b, and Fig. 4.10, plus additional images).

When view angle is examined individually, it is hard to see a relationship with the number of features detected per unit area (Fig. 4.11a). The number of features detected goes down with increasing view angle, but so does the area that the face takes up in the image, so the maximum number of features per unit area stays relatively constant. There is a clear

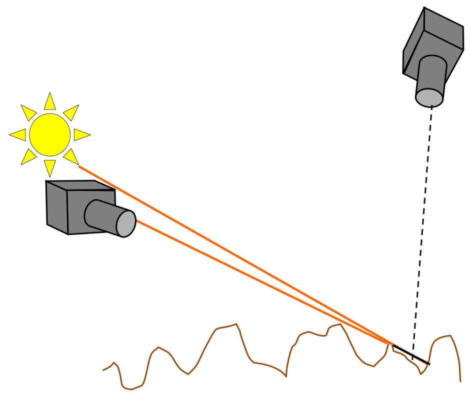
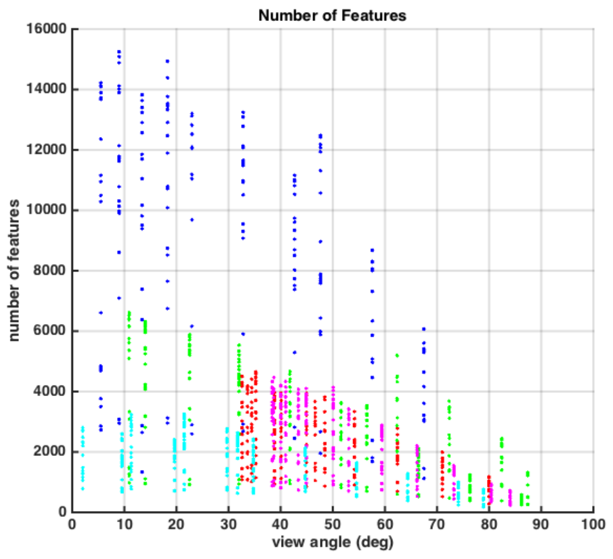


Figure 4.8: For textured rock surfaces, the extent to which self-shadowing is visible to a camera depends on phase angle; it becomes less visible as phase angle decreases.

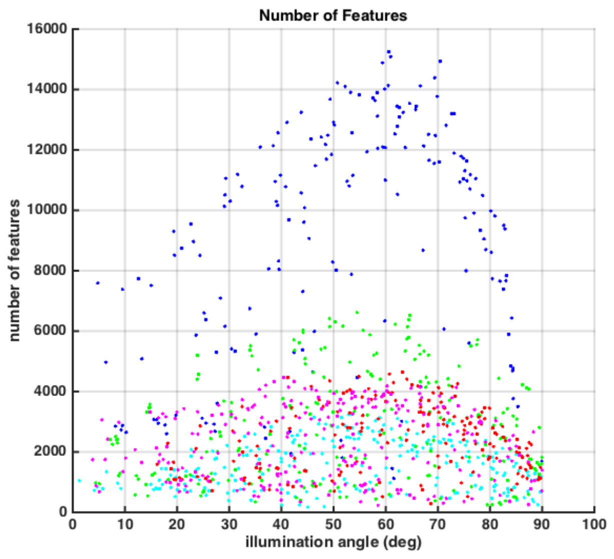
decline in number of features per unit area as illumination angle approaches 90 degrees (Fig. 4.11b). By plotting the number of features per unit area in 3D with both view and illumination angle the relationship becomes clearer (Fig. 4.12), and the low-view-angle peak at around 60 degrees for illumination angle becomes more obvious than in Fig. 4.10. Fitting a surface to these points makes the relationship easier to see (Fig. 4.13). For small view angles, the number of features increases as illumination angle moves away from zero and decreases again as illumination angle approaches 90 degrees. As view angle increases, the data get much noisier in the low to mid illumination angle range, and for low illumination angles, the number of features detected per unit area increases with increasing view angle (partially due to the decreasing area used for normalization).

The variation in number of features detected per unit area over view and illumination angle makes more sense when one thinks in terms of phase angle. Fig. 4.14 and Fig. 4.15 show this relationship visually. The number of features detected at low phase angles is low across all illumination angles. As phase angle increases, the number of features detected tends to increase, but there is still a drop-off as illumination angle approaches 90 degrees. This explains why the view angle plot gets noisier as view angle increases. If both view and illumination angle are small, phase angle must also be small. Likewise if one of view and illumination angle is large and the other is small, phase angle will be large. If both view and illumination angle are intermediate or large, however, phase angle could either be large or small. Hence the large amount of noise in the intermediate view angle, intermediate illumination angle range. Phase angle also explains the peak at high view angle and low illumination angle (this peak is artificially high because large view angles are normalized by small areas, but given high view angle, the number of features detected per unit area is higher at low illumination angles, which would correspond to high phase angles).

While the phase versus illumination angle relationship with number of features detected is cleaner, the view versus illumination angle relationship may be more useful in planning. View angle is set by a combination of rover location and terrain geometry, and it remains constant over all times. Illumination angle is set by time of day (which determines the sun location relative to terrain) and terrain geometry, and it remains constant over different rover



(a)



(b)

Figure 4.9: Plots of number of features detected on a rock face, plotted versus (a) view angle (b) and illumination angle for the image. Different colors indicate different faces.

positions. Phase angle couples rover position, time of day, and terrain geometry. However, if the maximum view angle is restricted due to resolution considerations, as discussed in 4.4.1, the peak at low view angles and intermediate illumination angles is relatively clean.

To set minimum and maximum illumination angles, the surface from Fig. 4.13 is first clipped to remove everything above the maximum view angle set for resolution reasons. The surface is projected onto axes representing illumination angle and number of features detected per unit area. Then the surface is scaled such that its maximum value is set to 100%. A target percentage of the maximum number of features detected per unit area is chosen. The minimum and maximum illumination angles can then be set using the intersection of the surface with a line at the threshold percentage, as shown in Fig. 4.16. When the illumination angle falls between the minimum and maximum thresholds, the number of features detected per unit area (for which the fitted surface provides an estimate) is likely to be at or above the target percentage of the maximum possible number of features detected per unit area.

The number of features detected (both raw and per unit area) does vary across different rocks. (Note that the three rocks tested varied in both texture and color). Results for the lace rock were similar to those presented above for the lava rock, although noisier (see Fig. 4.17). Results for the basalt rock, which has faces with a significantly smoother texture, showed a peak at higher illumination angles (see Fig. 4.18). This is not too surprising, since for faces with smoother geometric texture, the visual texture created by small-scale self-shadowing would only begin to happen at higher illumination angles. While the studies conducted in this thesis hint at how the relationship between number of features detected and illumination angle changes with different surface textures, comprehensive conclusions cannot be drawn

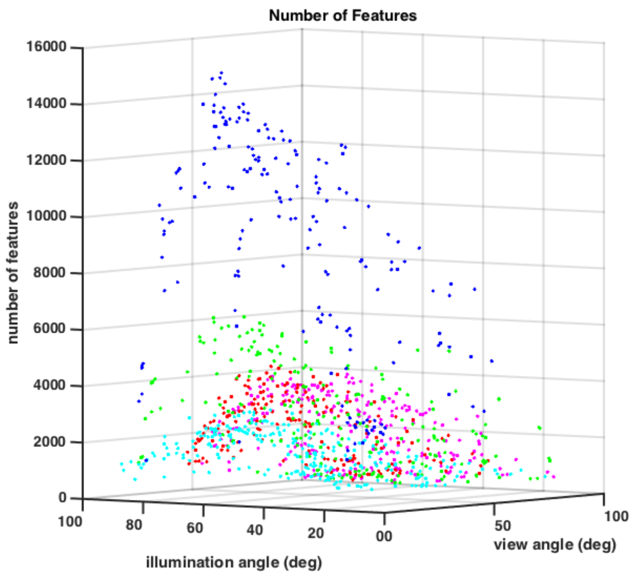
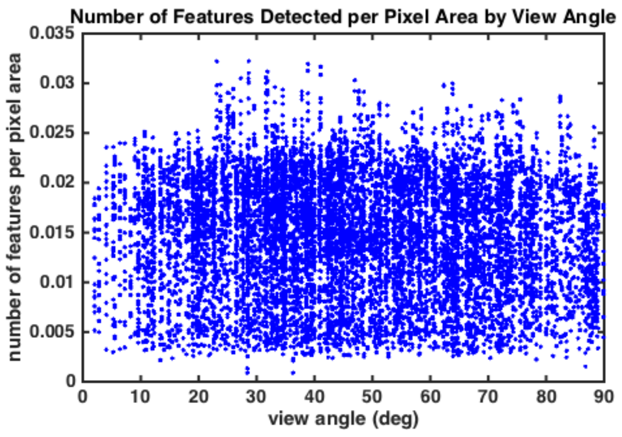
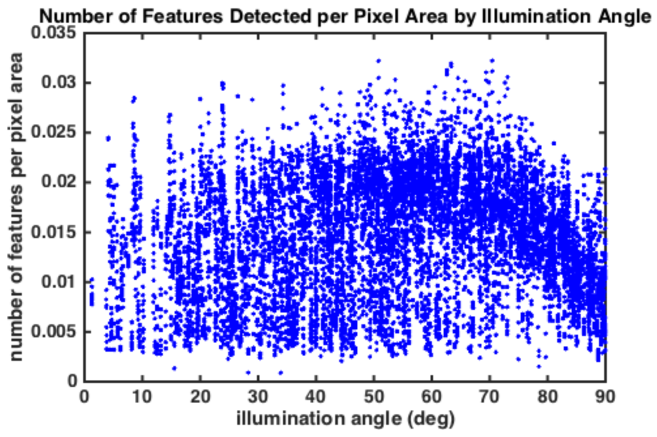


Figure 4.10: Plot of number of features detected on a rock face, plotted versus view angle and illumination angle for the image. Different colors indicate different faces.

from three rock samples. More thorough study of the phenomenon is left to future work.



(a)



(b)

Figure 4.11: Plots of number of features detected on a rock face per unit area (in pixels) that the face covers in an image, plotted versus (a) view angle (b) and illumination angle for the image.

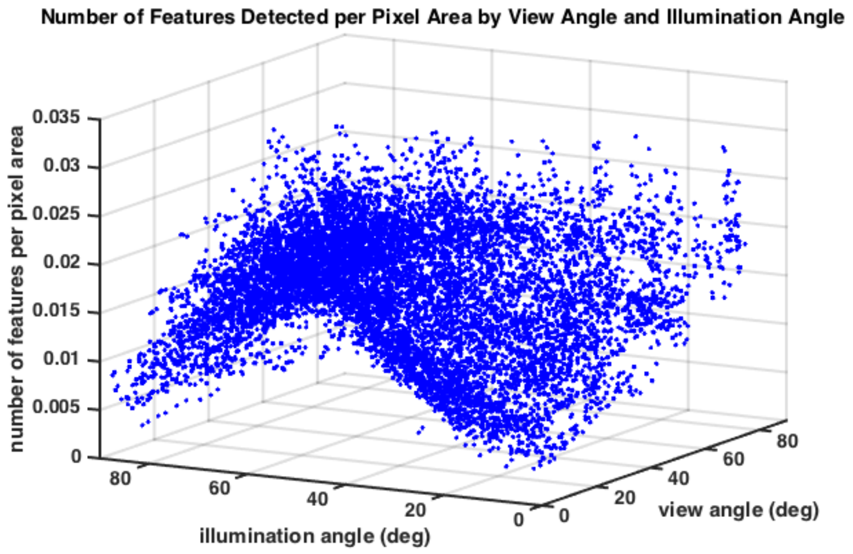


Figure 4.12: The number of features detected on a rock face per unit area (in pixels) that the face covers in an image, plotted versus view angle and illumination angle for the image.

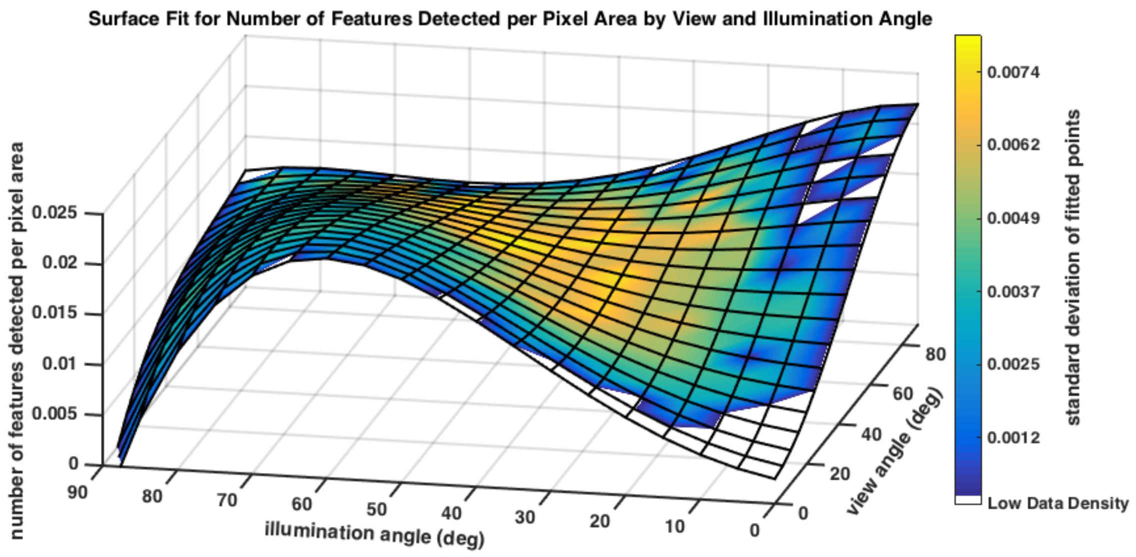


Figure 4.13: Surface fit for the number of features detected on a rock face per unit area (in pixels) that the face covers in an image, plotted versus view angle and illumination angle for the image. The surface is colored by the standard deviation of the number of features per unit area data.

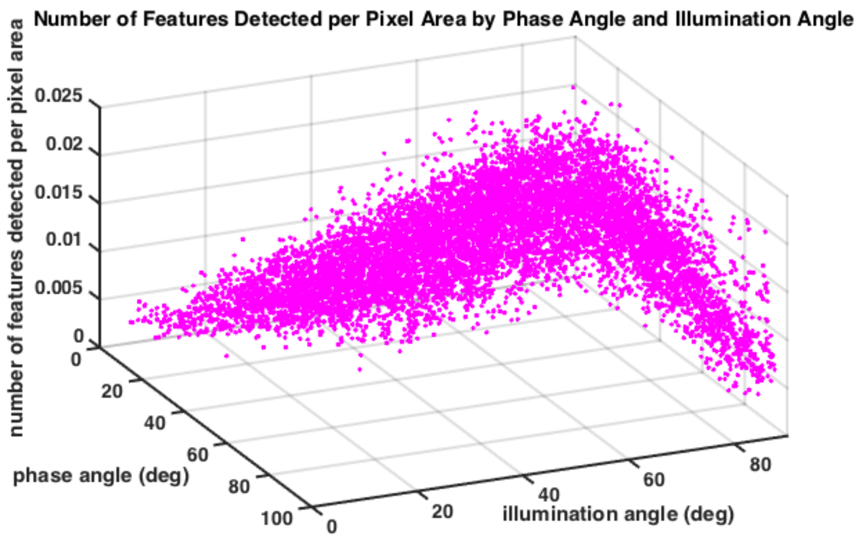


Figure 4.14: The number of features detected on a rock face per unit area (in pixels) that the face covers in an image, plotted versus phase angle and illumination angle for the image.

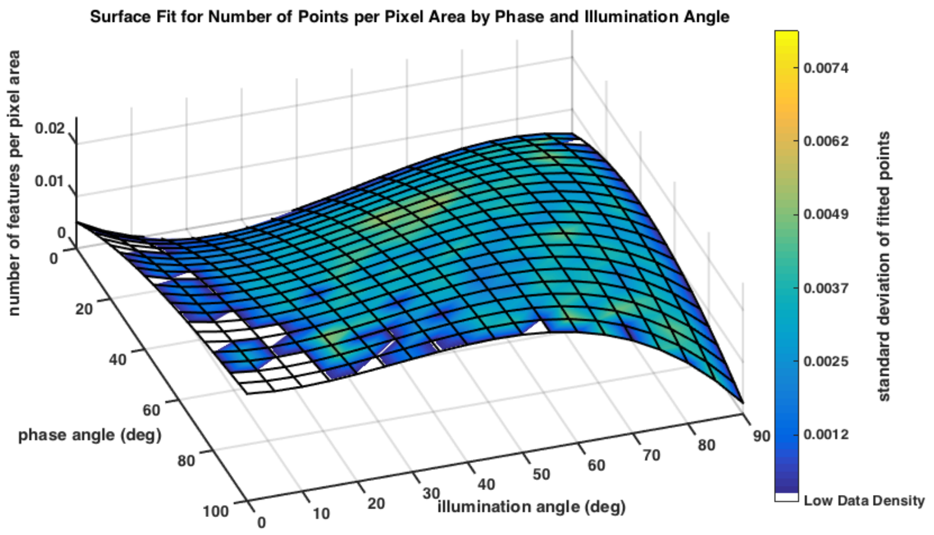


Figure 4.15: Surface fit for the number of features detected on a rock face per unit area (in pixels) that the face covers in an image, plotted versus phase angle and illumination angle for the image. The surface is colored by the standard deviation of the number of features per unit area data. The color scale is the same as in Fig. 4.13

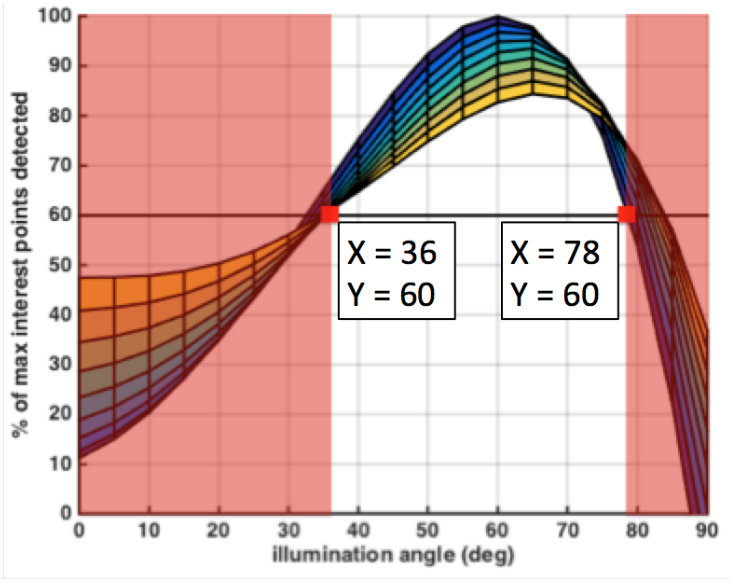
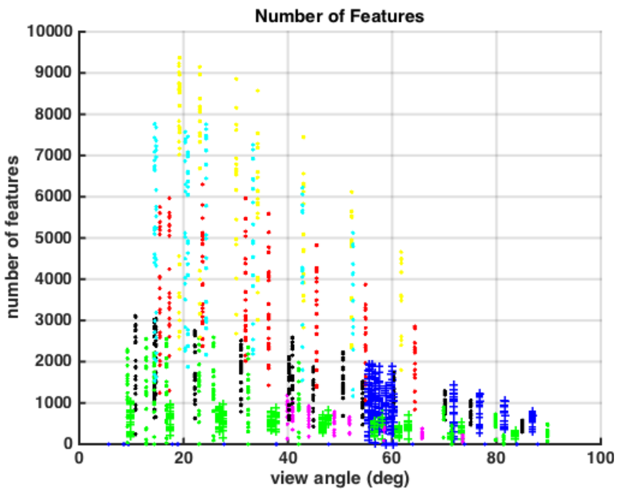
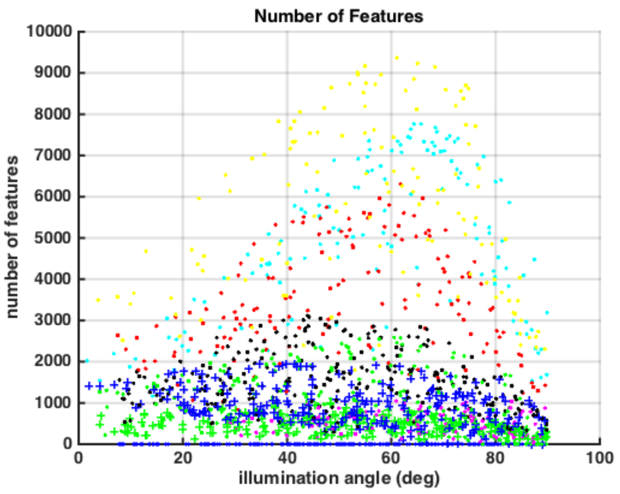


Figure 4.16: Example of picking illumination angle limits. The fitted surface for number of features detected per unit area (from Fig. 4.13) is cropped from 0 to 43 degrees in view angle, projected onto a plane, scaled from 0 to 100%. Targeting at least 60% of the maximum features per unit area detected, minimum and maximum view angle thresholds can be selected. Illumination angles outside these thresholds (red shaded regions) are less likely to produce the target percentage of number of features detected per unit area.

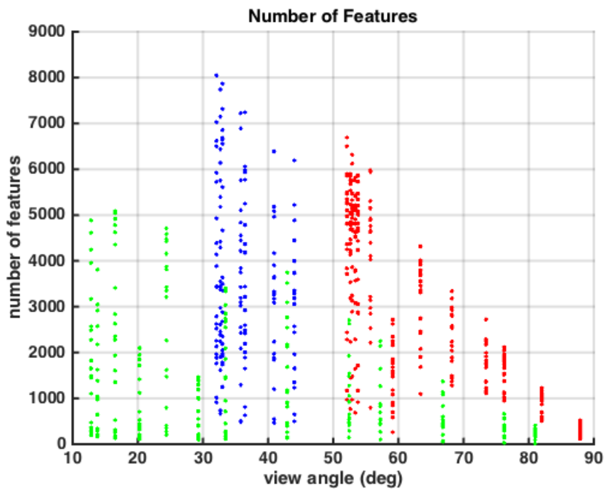


(a)

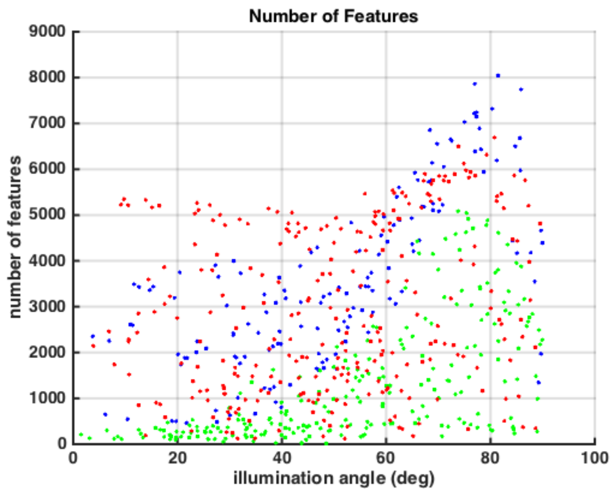


(b)

Figure 4.17: Plots of number of features detected on a rock face for the lace rock, plotted versus (a) view angle (b) and illumination angle for the image. Different colors/symbols indicate different faces.



(a)



(b)

Figure 4.18: Plots of number of features detected on a rock face for the basalt rock, plotted versus (a) view angle (b) and illumination angle for the image. Different colors indicate different faces.

4.4.3 3D Coverage

This work assumes that 3D models are built using stereo reconstruction from camera images, so in order to get 3D coverage of a rock face, it must, at minimum, be covered in 2D by two images. Furthermore, stereo reconstruction requires that the correspondence between images be determined. This can be done by matching features. Feature matching depends on relative view and illumination angles between image pairs. Data for feature matching under directional lighting were collected as part of the experimentation described in Appendix A (see Fig. 4.19). The metric reported is percent of features matched, where the number of features used in computing the percentage is the number of features detected in the image with the smaller number of detected features. Note that this is different than the metrics discussed in the previous section. An exponential surface was then fit to the data, using eq. 4.3. Table 4.1 shows the parameters computed by fitting to data from each of three rocks individually and to all the data from all three rocks.

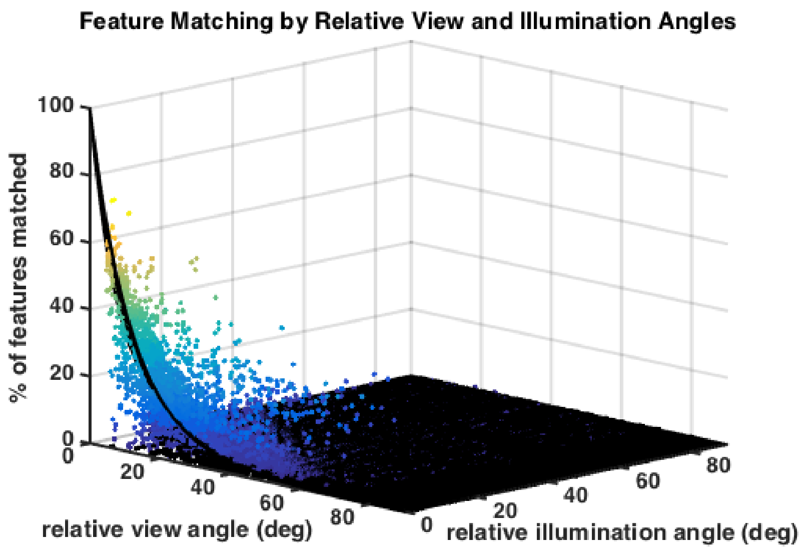


Figure 4.19: Effect of relative view and illumination angle on feature matching. Relative view and illumination angles are measured between view vectors and illumination vectors, respectively. Features matched between image pairs are reported as a percentage of the number of features detected in the image with the smaller number of detected features. Data points come from various faces on three rock samples with varying surface texture and are colored by z-value, with black indicating zero percent of features were matched and lighter colors indicating higher match percentages. An exponential surface fit is also shown.

$$P_{match} = 100e^{-(k_{view}\beta_{view}+k_{illum}\beta_{illum})} \tag{4.3}$$

$$\beta_{illum} = \frac{-k_{view}\beta_{view} - ln\left(\frac{P_{match}}{100}\right)}{k_{illum}} \tag{4.4}$$

Given a desired percentage of features matched, maximum relative view and illumination angles can be selected from along the line described by eq. 4.4, in the region $\beta_{view} > 0$, $\beta_{illum} >$

Table 4.1: Fitted Parameters for Exponential Fit of Percent Features Matched by Relative View and Illumination Angles

Dataset	k_{view}	k_{illum}
basalt, smooth	0.10	0.093
lava, fine texture	0.11	0.13
lace, coarse texture	0.073	0.12
all points	0.086	0.12

0. Using the parameters computed from all points and a target of 5% of features matched, values of $\beta_{view} = 15$ and $\beta_{illum} = 15$ correspond to one point along the line (to the nearest degree).

4.4.4 Reconstruction Accuracy

Understanding how relative view angle can affect 3D reconstruction accuracy requires a basic understanding of how stereo vision works. Features are identified in two images and matched. The difference in image position of a feature between two cameras, called the disparity, increases as the feature gets closer to the camera. Because there is some distance, or baseline, between the cameras, the 3D position of each feature can be triangulated (see Fig. 4.20). As the relative view angle gets smaller, the baseline gets smaller.

For view trajectory planning, a determination of the minimum acceptable view angle is desired. As seen in Fig. 4.21, limited precision in feature localization results in ambiguity in the 3D triangulated position of features in the world frame. As the relative view angle decreases, this ambiguity increases in the depth direction. The example illustrates a precision limit of one pixel. While modern feature detection methods can localize features to sub-pixel precision, they will not be arbitrarily precise, simply due to the discretized nature of digital cameras. To understand how stereo ambiguity varies with changing relative view angles, an simulation experiment was conducted.

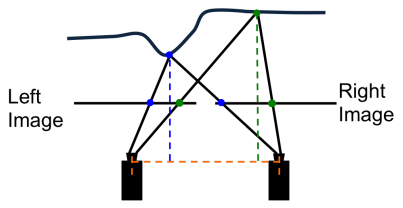


Figure 4.20: The basic concept of stereo vision. Features are identified in two images. Differences in the image positions of features, called the disparity, will be greater for features that are closer to the cameras. Given that the cameras had some separation, or baseline, distance when the images were taken, the 3D positions of features in the world frame can be computed by triangulation.

This simulation views a planar face with two cameras. Camera 1 remains static with view angles of zero in x and y, while the view angles of camera 2 are varied about x and y in the face's local frame. The view vectors from the centers of both cameras point at the center of the face. Both cameras have an even number of pixels in width and height, such that the camera's view vector falls between view vectors for its four central pixels. A feature is assumed to exist at the intersection of the ray from the upper left of the four central pixels for camera 1 and the plane of the face. The "correct" pixel in camera 2 to match is the one whose ray intersects the face plane closest to the feature location. The triangulated position of the feature in 3D is the midpoint of the line connecting the pixel ray

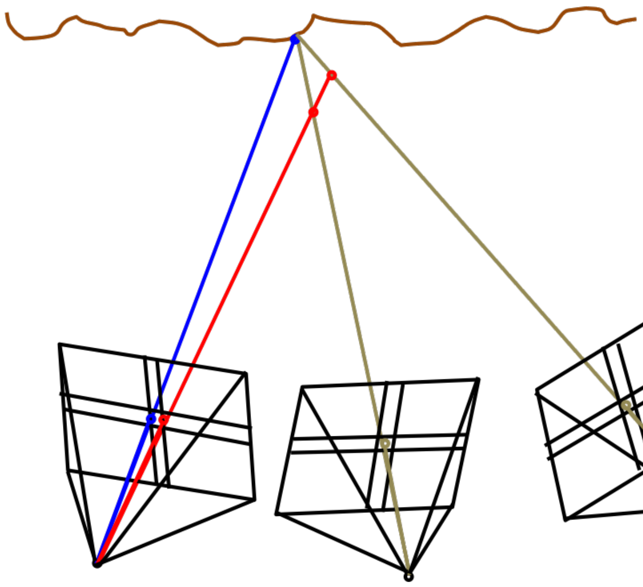


Figure 4.21: Limits on the precision with which a feature can be localized result in ambiguity in the triangulated position of that feature in the world frame. The illustration shows the difference in triangulated position depending on which side of a pixel a feature is located. For smaller relative view angles, the depth ambiguity at the same localization precision is larger.

from camera 1 with the ray from its matching pixel in camera 2, at their closest approach. There will be some error in this triangulated position, even with the correct match, (while modern stereo algorithms rely on identification of features to sub-pixel resolution, there is still some minimum resolution at which such features may be detected, resulting in some amount of inherent measurement error). If the feature location in the image from camera 2 is mis-identified, there will be a greater amount of error. As illustrated in Fig. 4.22, the experiment looks at matching the feature to the correct pixel, to a pixel one row off, to a pixel one column off, and to a pixel one row and one column off (off by one along the diagonal). Fig. 4.23 shows the error over a range of camera 2 angles in each of these cases, given a camera with resolution 4272 x 2848 and focal length 100mm looking at a face 1.35 meters away.

The absolute error for a particular camera configuration is not all that interesting, as the results may not be generalizable. Instead, error can be expressed as a fraction of the surface resolution for camera 2. If the error is greater than the stated resolution of a 3D reconstruction, then clearly that reconstruction will not be valid to that resolution. The best possible resolution for stereo triangulation is the resolution of the images used, in which case, you want the ratio of error over image resolution to be less than 1. However, there is often not enough information in an image to do stereo reconstruction to the base image resolution. One suggested rule-of-thumb is to use at least a 3x3 square of pixels for each point in the 3D reconstruction [73]. Fig. 4.24 shows the computation of error as a fraction of surface resolution for the same configuration as Fig. 4.23. This can be thresholded at 3 times the surface resolution to find the region in which angles are "too small". Minimum relative view angle can then be set as just larger than the maximum angle in the "too small" region.

Table 4.2 gives the values for minimum relative view angle thresholds for several different feature localization precisions. Fig. 4.24 illustrates a 1-pixel feature localization precision to make the phenomena clearer, but modern feature detection methods can achieve sub-pixel precision.

Table 4.2: Minimum relative view angle threshold for ambiguity by feature localization precision, using the laboratory configuration from Appendix A

Feature localization precision (pixels)	Minimum relative view angle for ambiguity < 3x surface resolution (deg)
0.1	2.5
0.2	5
0.3	7
1	22

The experiment was re-run, using a 0.2-pixel feature localization precision, for 50 different camera resolutions varying from 4272x2848 down to 85x56, (1/50th the size), and it produced a result of 5 degrees for the minimum relative view angle in each case. It was tested for focal lengths of 10mm to 500mm in 10mm increments, (corresponding to a field of view of 98x74 degrees for 10mm focal length to 2.6x1.7 degrees for 500mm focal length), and it produced a result of 5 degrees for the minimum relative view angle in each case. Given that surface resolution is proportional to viewing distance when all other camera parameters are held constant and the metric in this experiment is error as a fraction of surface resolution, results should be the same over varying distance as well. Changing the view angles of camera 1 shifts the pattern in Fig. 4.24 to be centered around camera 1's view angles, but it retains essentially the same shape. Thus, it works to set minimum relative view angle depending on feature localization precision.

To support this analysis experimentally, 3D reconstructions were done with pairs of images at different relative view angles in the lava small angles dataset (see AppendixA). The reconstructions were compared to the coarse mesh model, and median and standard deviation of error were computed. Results from this work are shown in Fig. 4.25 and Fig. 4.26. Note that high errors tend to occur at 5 degrees or less of relative view angle and above 15 degrees of relative view angle. The problems for low relative view angles are likely related to the issues discussed above. The problems at higher relative view angles are likely due to the low number of feature matches at these relative view angles (in some cases there is no reconstructed model). Also note that the problems at higher view angles occur in the "light next to camera" cases, which corresponded to a light coming from next to the camera, while the "light overhead" cases correspond to light coming from above the rock. So, when the turntable was rotated, changing the view angle, it was also changing the illumination angle. This illumination angle change was greater in the "light next to camera" cases. Based on the results in Section 4.4.3, it is not surprising to see failure at higher relative view and illumination angles.

While the coarse mesh models are not precise enough to evaluate errors on the scale of surface resolution, it is likely that ambiguity in feature 3D positions at low relative view angles will cause problems in generating the structure-from-motion solution, which result in the high errors seen here.

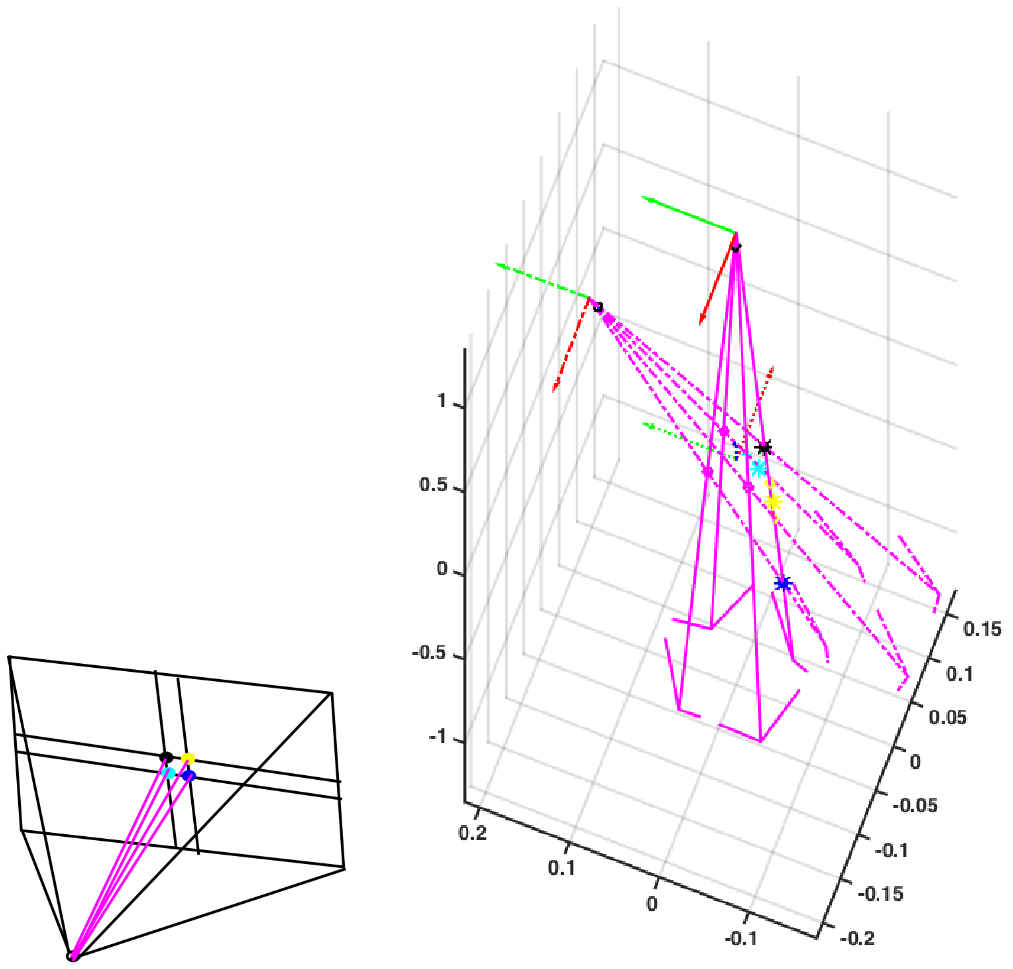


Figure 4.22: Computing ambiguity in feature position as consequence of feature detection precision. The left image shows an illustration of camera 2. The black dot indicates the correct feature location in camera 2's image, the yellow dot represents a location one pixel-width away in x, the cyan dot represents a location one pixel-height away in y, and the blue dot represents a location one pixel-width away diagonally. On the right, magenta arrows represent rays from camera centers to points in 3D space. Camera 1 is shown with solid lines and camera 2 with dotted lines. Colored *'s indicate triangulated positions for each of the four image locations shown on the left. The correct 3D point lies at the spot where one of the rays from camera 1 intersects the xy plane. Using the correct feature location for camera 1 gives a triangulated position very close to the true position. The distance from the true position gets larger when other image locations are used.

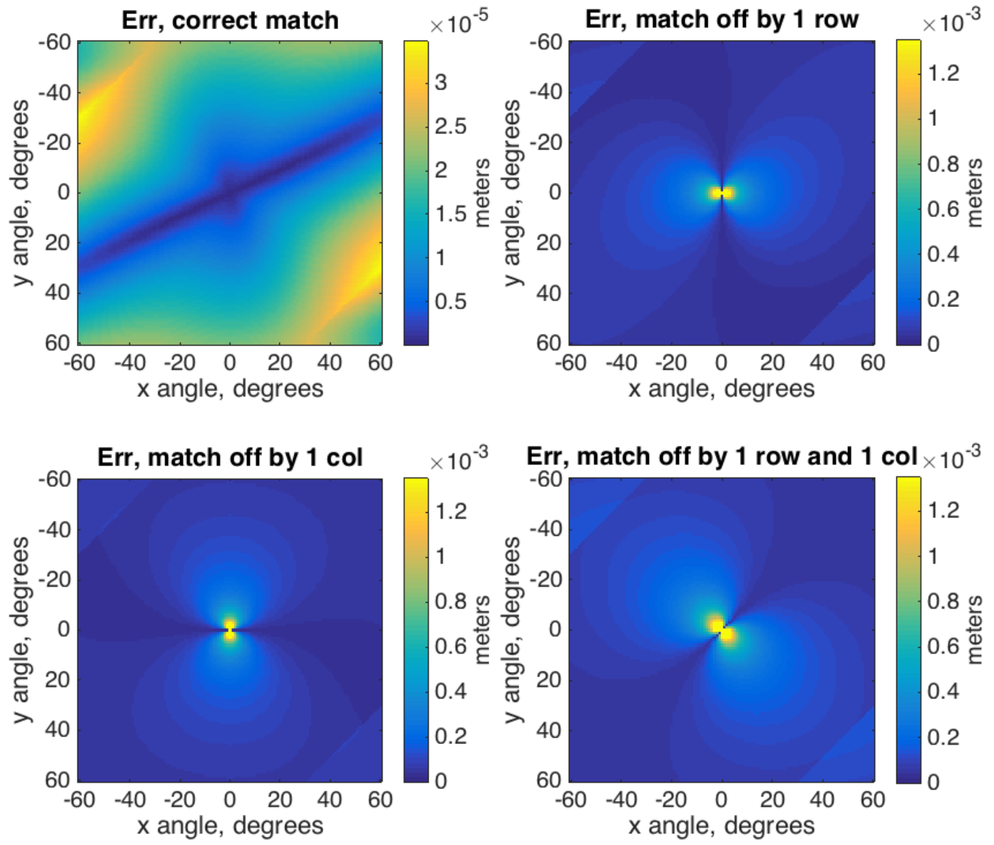


Figure 4.23: Stereo triangulation error when the a feature is matched to the correct pixel in camera 2, and pixels one pixel-width away in row, column, or diagonal. Note the order-of-magnitude difference in error between the correct pixel and the off-by-one matches.

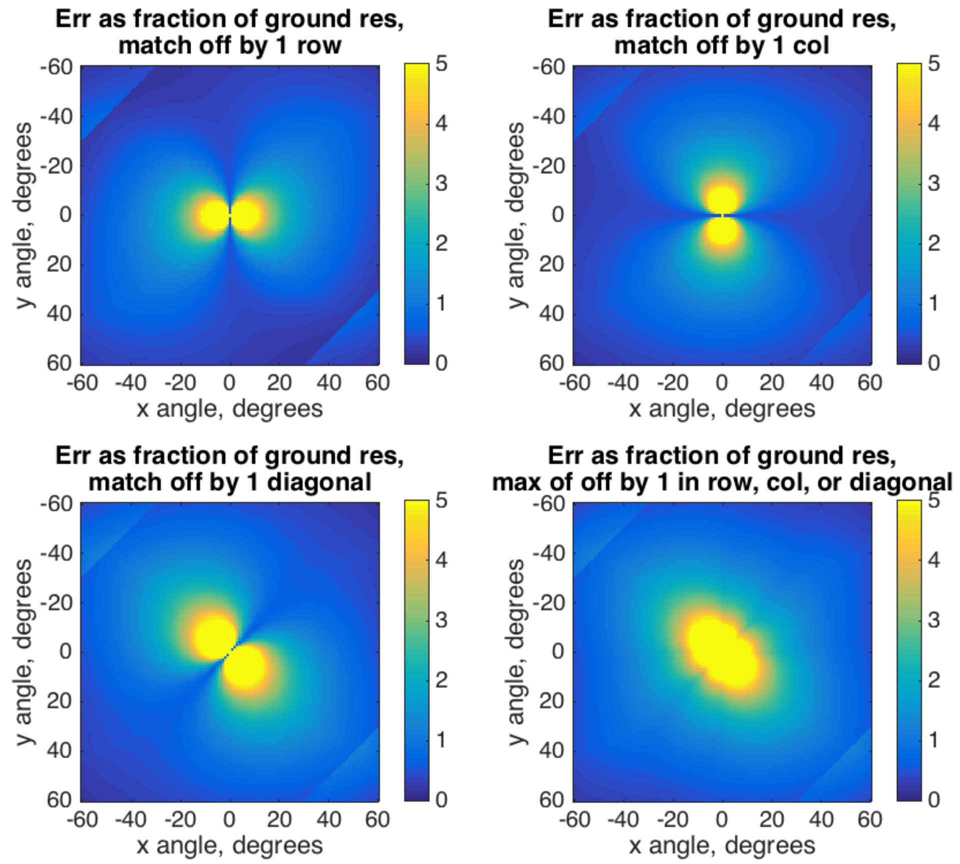


Figure 4.24: Stereo triangulation error as a fraction of the surface resolution of camera 2 when the a feature is matched to pixels one off in row, column, or diagonal. The lower right plot shows the maximum of these three (similar to a logical OR).

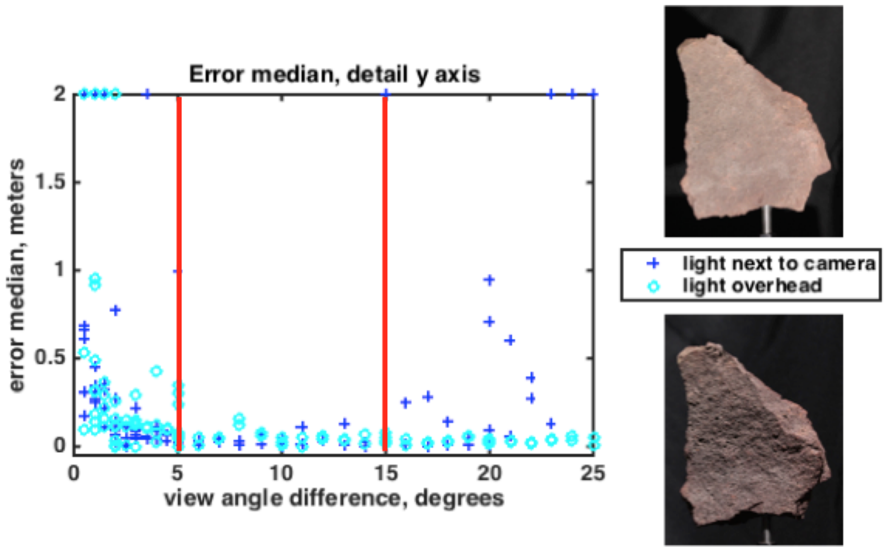


Figure 4.25: Median error for points in reconstructed models from pairs of images at various relative view angles. For reconstructions that failed, the median could be very high (> 100m), so the axes for this plot were capped at a maximum threshold, and all points above that threshold were plotted at that threshold value. Reconstructions with zero points were also plotted at this threshold value. Example images are shown to highlight differences in illumination angle.

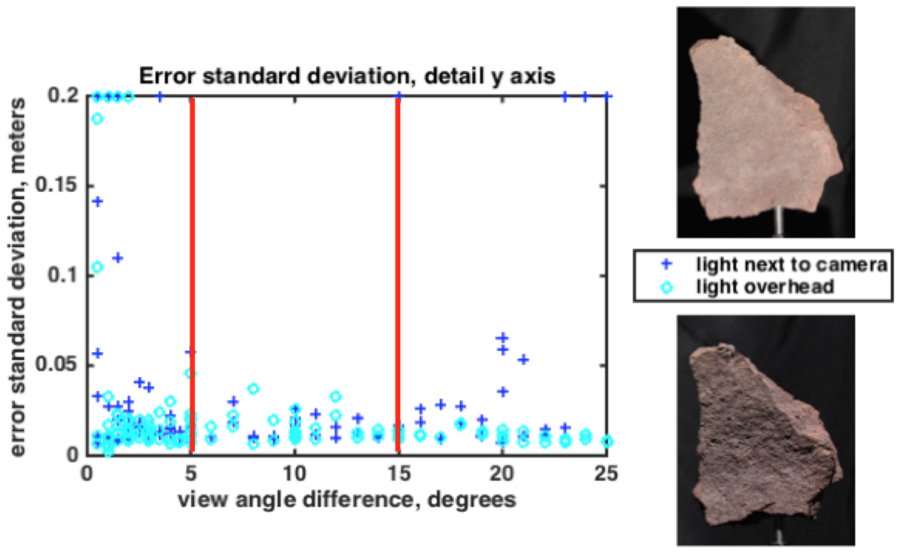


Figure 4.26: Standard deviation of error for points in reconstructed models from pairs of images at various relative view angles. For reconstructions that failed, the standard deviation could be very high (> 100m), so the axes for this plots were capped at a maximum threshold, and all points above that threshold were plotted at that threshold value. Reconstructions with zero points were also plotted at this threshold value. Example images are shown to highlight differences in illumination angle.

Chapter 5

The OPTWIND Problem

5.1 Introduction

This chapter formulates the Orienteering Problem with Time Windows and Inter-Node Dependencies (OPTWIND). The formulation is done with planned-view-trajectory model building in mind. Nodes are assumed to be position-time pairs, where positions are potential robot positions and times are the coarse time divisions used when computing illumination data for each face. Value for each node comes from the faces that are visible from that position at that time that meet angle constraints. However, since faces may be viewed from different positions and at different times, there are *inter-node dependencies*. The value achieved by viewing particular faces from particular nodes is computed similarly to the plan value described in Section 3.3. Some value is achieved for the first view of a face, but after that, inter-node dependencies come into play: additional value is achieved if another view of the face meets relative angle constraints with the first, and once the target number of views for a face (that meet constraints) has been reached, additional views of the face do not add value. Like the Orienteering problem, the value of all nodes in the OPTWIND is maximized subject to a maximum distance constraint.

The OPTWIND is not exclusive to view trajectory planning for terrain feature modeling. Two other scenarios that could be formulated as an OPTWIND are described below.

A geologist (human or robot) wants to build a model of the local geology. To do so, he should get at least one spectrometer reading of each type of rock. To ensure that the rock he chose to sample was not an outlier, the geologist should sample several rocks of each type. Within rocks of the same type, there may be some expected variation. For example, there may be several lava flows in the area that are known from orbital imagery to have occurred at different times. While there are outcrops of basaltic rock from each of these lava flows, the basalt from the older flows may be more weathered, so it would be valuable to take samples from the different flows to understand the class 'basalt' and its variation in appearance and surface composition with weathering. At certain times, the illumination on certain rocks is not sufficient to take a spectrometer reading (for example, on the east side of a hill in the evening), so the geologist must schedule his activities to ensure good illumination. Regions may provide access to multiple types of rocks, but there are also rock types that can only be seen in certain regions. The geologist wants to minimize the distance he has to travel to collect all his measurements. Which regions should the geologist visit, and when?

A student travels to Paris, and has only a few days to visit all the museums in the city. She wants to treat her visit as a lesson in art history, so she wants to see a painting from each of a list of art movements (i.e. Gothic, Impressionist, Expressionist, Baroque, Abstract, Realist, etc.). She wants to see a few paintings from each movement, and she wants those images to be sufficiently different, such that they provide a sense of the range of individual artists' styles within the movement. One gallery in a museum may have several paintings of interest, but paintings may also be located in different galleries of the same museum, and in different museums. Each museum has specific hours when it is open. The student wants to maximize the value of her trip (the increase in her understanding of art history), but she also wants to minimize the distance she has to travel, because walking around museums all day can be quite tiring. Which galleries should the student visit, and when?

The following section formulates OPTWIND more formally.

5.2 Formulation of the Orienteering Problem with Time Windows and Inter-Node Dependencies

The formulation of the Orienteering Problem with Time Windows and Inter-Node Dependencies (OPTWIND) starts from the standard formulation of the Orienteering Problem with Time Windows (OPTW) [10]. There are a set of nodes $(n_1, ..., n_N)$. A set of indicator variables x track whether a given node has been visited and in what order: the variable $x_{ij} = 1$ if the agent visits node n_i immediately before node n_j. The distance and time between nodes n_i and n_j are d_{ij} and τ_{ij}, respectively. For each node n_i, σ_i is the start of the visit to that node. M is a large constant. The time window for n_i is the interval $[O_i, C_i]$.

Constraints in eq. 5.1 ensure that the path starts and ends at predetermined start and end nodes.

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1 \tag{5.1}$$

Constraints in eq. 5.2 ensure that the path is continuous, and each node is visited no more than once.

$$\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \leq 1 \tag{5.2}$$

Eq. 5.3 ensures that the path does not exceed a maximum distance. In some work with the orienteering problem, this is expressed as a maximum time instead of a maximum distance, but time is also constrained by absolute time windows, and if the distance that could be traveled in between the beginning of the first time window and the end of the last is greater than the desired maximum distance, then distance should be constrained independently.

$$\sum_{i=1}^{N-1}\sum_{j=2}^{N} d_{ij}x_{ij} \leq D_{max} \tag{5.3}$$

Eq. 5.4 ensures that the timeline is consistent; in other words, if the agent travels from node n_i to node n_j, then arrival time at n_j cannot be less than the arrival time at n_i plus the travel time between n_i and n_j. The indicator y_i is also added, where $y_i = 1$ if node i is in the path, and zero otherwise.

$$\sigma_i + \tau_{ij} - \sigma_j \leq M\left(1 - x_{ij}\right); \forall i, j = 1, ..., N \tag{5.4}$$

Eq. 5.5 and eq. 5.6 ensure that the visit to a node starts after that node's time window opens and before its time window closes.

$$O_i \leq \sigma_i \tag{5.5}$$

$$\sigma_i \leq C_i \tag{5.6}$$

In OPTW, each node has a fixed score, and the objective is to optimize the sum of the scores over all nodes in the path. Similar to the Generalized Orienteering Problem (GOP), each node has not just one score in the OPTWIND formulation, but scores in several categories, (representing the faces visible from that node), and the total score for the path is a function of per-category scores for each node (these scores correspond to view angles and illumination angles).

In OPTWIND, the score for each node n_i is the sum of the scores across all categories, $\{p_1, ..., p_k\}$, (these categories correspond to faces in terrain feature modeling). In each category, p_k, there are one or more ratings $\{\beta_{ik1}, ..., \beta_{ikz}, ..., \beta_{ikZ}\}$, where each β_{ikz} is a number representing the z'th rating for the i'th node in the k'th category. These ratings correspond to view and illumination angles for terrain feature modeling. The score in a given category for a given node is dependent on whether the ratings for that category meet absolute constraints for ratings and relative constraints with ratings in the same category for nodes previously visited. If constraints are met, the score for the node in that category is S_k, up to a total of $S_{k,max}$ for that category across all nodes.

Eq. 5.7 accounts for the absolute constraints over all ratings (indexed by z). It is formulated as a product, such that if any constraint is violated, the term added to the product for the violated constraint is zero, which makes a equal to zero.

$$\begin{aligned} a\left(n_i, p_k\right) = \prod_{z=1}^{Z} &\left[min\left[max\left(\beta_{ikz} - A_{z,min}, 0\right), 1\right] * \right. \\ &\left. min\left[-min\left(\beta_{ikz} - A_{z,max}, 0\right), 1\right]\right] \end{aligned} \tag{5.7}$$

Eq. 5.8 accounts for the relative constraints over all ratings (indexed by z). It is formulated as a product, such that if any constraint is violated, the term added to the product for the violated constraint is zero, which makes r equal to zero.

$$\begin{gathered} r\left(\{n_1, ..., n_i\}, p_k\right) = max\Bigg\{ \sum_{j=1}^{i-1} \Bigg[y_j * \\ \prod_{z=1}^{Z} \Bigg(min\left[max\left(|\beta_{ikz} - \beta_{jkz}| - R_{z,min}, 0\right), 1\right] * \\ min\left[-min\left(|\beta_{ikz} - \beta_{jkz}| - R_{z,max}, 0\right), 1\right] \Bigg) \Bigg], 1 \Bigg\} \end{gathered} \tag{5.8}$$

The function f indicates whether this is the first node on the path for which p_k meets absolute rating constraints.

$$\begin{gathered} f\left(\{n_1, ..., n_i\}, p_k\right) = \\ min\Bigg\{ -min\left[\sum_{j=1}^{i} \left(a\left(n_i, p_k\right)\right) - 1, 0 \right], 1 \Bigg\} \end{gathered} \tag{5.9}$$

The expression in eq. 5.10 evaluates to a binary value that indicates whether or not the score S_k can be earned. Everything in the expression is multiplied by a: all the absolute constraints must be met to gain value in the category. Given that absolute constraints are met, some value can be gained for the first view of a face (if f is true). If this is not the first view, relative constraints must be met to gain value (r must be true).

$$a\left(n_i, p_k\right)\left\{ f\left(\{n_1, ..., n_i\}, p_k\right) + r\left(\{n_1, ..., n_i\}, p_k\right)\right\} \tag{5.10}$$

The total value for the route that the problem seeks to optimize is described in eq. 5.11. This objective function sums over all categories (or faces), choosing the minimum of the max possible score for that category and the sum of scores earned in that category over all nodes. This minimum ensures that no additional value is gained after the target number of views (which meet constraints) are captured. Each term in the sum across nodes is multiplied by the expression in eq. 5.10 to ensure that views meet constraints before they can add value. The indicator y_i is also included here to ensure that nodes are actually visited before they can add value.

$$\begin{gathered} \sum_{k=1}^{K} min\{ S_{k,max}, \sum_{i=1}^{N} \left[S_k a\left(n_i, p_k\right)\left\{ f\left(\{n_1, ..., n_i\}, p_k\right) + \right.\right. \\ \left.\left. r\left(\{n_1, ..., n_i\}, p_k\right)\right\} y_i \right] \} \end{gathered} \tag{5.11}$$

5.3 View Trajectory Planning for Pit Modeling as an OPTWIND

To formulate view trajectory planning for pit modeling as an OPTWIND, one node is created for each pair of time, t_i, and location, l_i: (t_i, l_i). The time windows are calculated as:

$$O_i = t_i T \text{ and } C_i = (t_i + 1)\, T \tag{5.12}$$

where T is the length of a coarse time division. One category is defined for each target face. The rating β_1 for each patch is the view angle from the rover position l_i in node n_i, and the rating β_2 is the illumination angle at time t_i in node n_i. Additional β values could be added to represent vertical view angles. Distances d_{ij} between nodes n_i and n_j are computed as $d(l_i, l_j)$, and times τ_{ij} are computed from d_{ij} using the rover speed. This application also relaxes the constraint in eq. 5.1 to only specify a start position and not an end position.

Absolute thresholds, $A_{z,min}$ and $A_{z,max}$, and relative thresholds $R_{z,min}$ and $R_{z,min}$ would be specified for each angle of concern. In some cases, the minimum angle threshold will be zero. These parameters should be set according to the procedures described in Chapter 4. Table 5.1 shows example thresholds.

It is also worth noting that this formulation assumes angles and angle limits are stored using x and y components. This will require that maximum limits be set somewhat lower than the single-angle values computed according to Chapter 4. Single-angle values for each angle of interest and each face could also be compared directly with single-angle thresholds, but this would require a more complicated formulation than that presented in eq. 5.7 and eq. 5.8.

Table 5.1: Example thresholds for use in planned-view-trajectory model building

Parameter	Description	Value (deg)	Source
$A_{1,min}$	minimum absolute view angle about x	0	No minimum
$A_{1,max}$	maximum absolute view angle about x	31	Section 4.4.1, reduced from 43 single angle
$A_{2,min}$	minimum absolute view angle about y	0	No minimum
$A_{2,max}$	maximum absolute view angle about y	31	Section 4.4.1, reduced from 43 single angle
$A_{3,min}$	minimum absolute illumination angle about x	36	Section 4.4.2
$A_{3,max}$	minimum absolute illumination angle about x	63	Section 4.4.2, reduced from 78 single angle
$A_{4,min}$	minimum absolute illumination angle about y	36	Section 4.4.2
$A_{4,max}$	maximum absolute illumination angle about y	63	Section 4.4.2, reduced from 78 single angle
$R_{1,min}$	minimum relative view angle about x	5	Section 4.4.4
$R_{1,max}$	minimum relative view angle about x	15	Section 4.4.3
$R_{2,min}$	minimum absolute view angle about y	5	Section 4.4.4
$R_{2,max}$	maximum absolute view angle about y	15	Section 4.4.3
$R_{3,min}$	minimum absolute illumination angle about x	0	No minimum
$R_{3,max}$	minimum absolute illumination angle about x	15	Section 4.4.3
$R_{4,min}$	minimum absolute illumination angle about y	0	No minimum
$R_{4,max}$	maximum absolute illumination angle about y	15	Section 4.4.3

Chapter 6

Planned-View-Trajectory Model Building Implementation

This chapter identifies how each part of the planned-view-trajectory model building process can be accomplished, either identifying and testing existing tools or implementing new solutions. Section 6.1 discusses how coarse prior models can be obtained or constructed to serve as input to the planned-view-trajectory model building process. Section 6.2 discusses how inputs are processed to construct an instance of the OPTWIND problem that can be given to a planner. Section 6.3 discusses view trajectory planning. Section 6.4 discusses how models can be built once a view trajectory plan is executed.

6.1 Coarse Models

Coarse prior models serve as a guide to planned-view-trajectory model building. They are used to determine the planar face discretization of a feature, which in turn is used to evaluate view and illumination angles for robot views. Chapter 3 illustrated how a coarse model of a pit could be generated by measuring the pit diameter and depth, and using a cylinder of the appropriate diameter and height as a coarse model. Many planetary features, however, are not that simple. Many natural pits of ultimate interest here are highly irregular, and there is often no easy geometric shape representation of a canyon or mesa. For these irregular features, the outline of the feature can be traced in satellite imagery and projected vertically to the estimated depth (or height).

If a high-resolution LIDAR digital elevation model of the area around the target feature is available, the coarse model can be created by trimming the digital elevation model down to the target feature. Fig. 6.1 illustrates this for a crater.

Figure 6.1: Coarse model of a crater created by trimming a digital elevation model.

If multiple satellite images (at the right viewing angles) are available, then a stereo model can be constructed from two or more images. This type of model will more accurately capture the 3D nature of a terrain feature. The following two subsections explore stereo modeling from orbit for a pit in the Moon's Lacus Mortis region. This pit is of particular exploration interest, since it has a relatively shallow ramp which could make robotic access easier. Section 6.1.1 builds a model from two orbital images. Section 6.1.2 uses multi-view stereo with five orbital images. (Note: these stereo modeling analyses have been previously presented in [16]).

6.1.1 Stereo Model Building from Two Orbital Images

A model was built with images from the Narrow Angle Camera (NAC) on the Lunar Reconnaissance Orbiter (LRO). Images M1105737674LE and M1105701957RE were used. These are farther apart in emission angle than ideal for stereo (approx. 35°, as compared to a target 20°used by orbital stereo model builders), but because M1105701957RE is highly oblique, the pair may provide better modeling for features not parallel to the ground plane - such as pit walls. A tool from the US Geological Survey, the Integrated System for Imagers and Spectrometers (ISIS) was used for calibration, re-projection, and bundle adjustment of the images [74], and Ames Stereo Pipeline [57, 58] was used to create a digital elevation model (DEM). Fig. 6.2 shows the a segment of this DEM. Fig. 6.3 and Fig. 6.4 show a mesh model created from this DEM. One of the orbital images used for stereo processing is overlaid on the DEM.

Another model was constructed from NAC images M1105737674LE and M1105759104LE. These images were purposely taken as a stereo pair. A section of this model is shown in

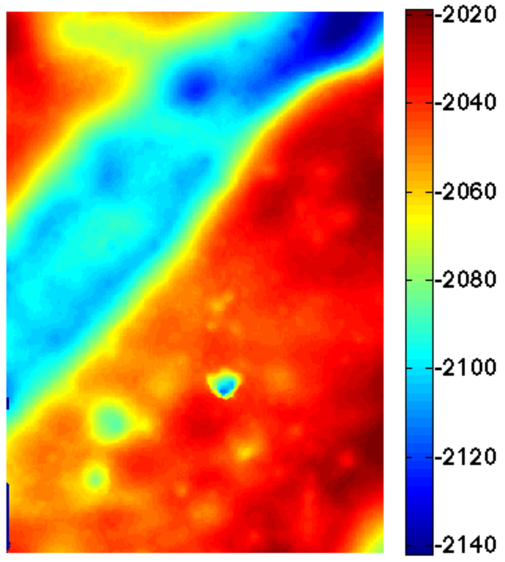


Figure 6.2: DEM created from orbital images of the region surrounding a pit in Lacus Mortis. Elevations are meters above the lunar datum. Notice that elevations are negative. The pit is the low spot located in the center of the image horizontally and level with "-2100" vertically.

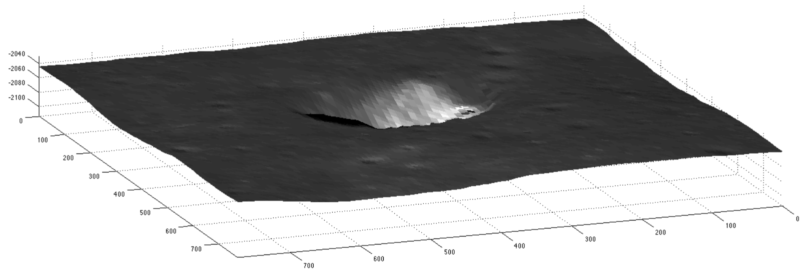


Figure 6.3: Mesh model of Lacus Mortis pit built from DEM, oblique view

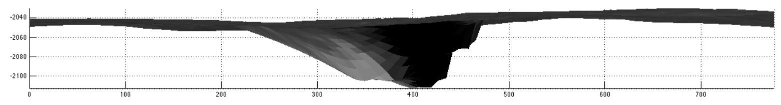


Figure 6.4: Mesh model of Lacus Mortis pit built from DEM, side view

Fig. 6.5. This snapshot focuses much more closely on the pit. Fig. 6.6 shows a comparison of this DEM to a reference DEM created by LRO Camera team members at Arizona State University. Fig. 6.7 and Fig. 6.8 show simulated camera images from a rover perspective. The model viewed has elevation as shown in Fig. 6.5 (from images at approximately 1.2 meters per pixel) overlaid with the highest resolution image available of the pit (approximately 0.5 meters per pixel). This model has several artifacts in areas of high stereo reconstruction error, including areas shadowed in the source images. Note that mesh triangles are clearly visible, despite the high-resolution (for orbital data) images. The lack of detail in this model from orbital data demonstrates how critical a rover perspective will be for creating detailed pit models.

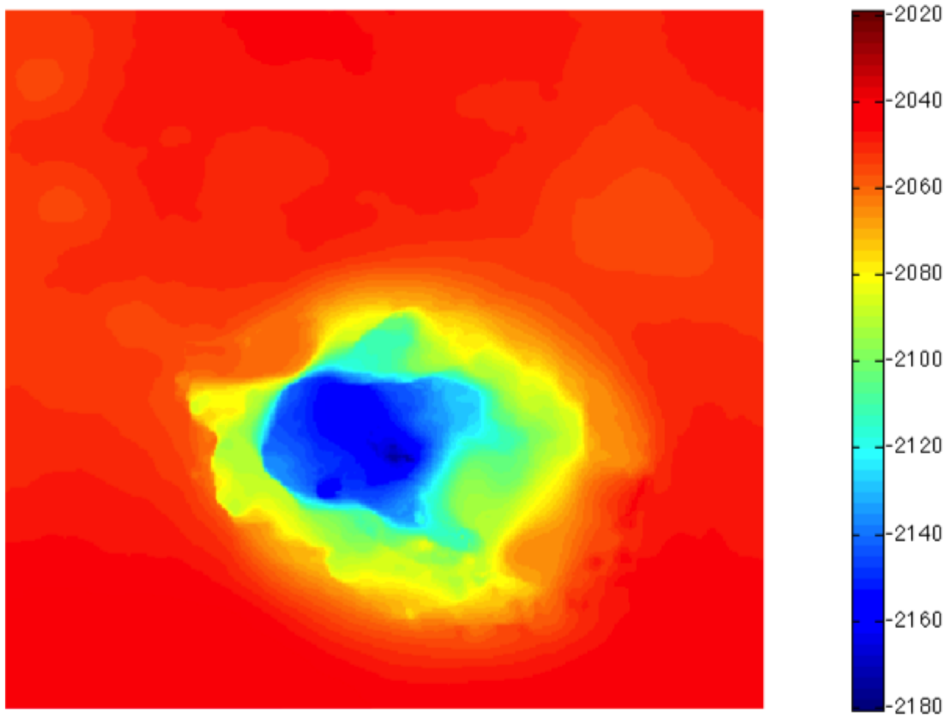


Figure 6.5: Close-up view of pit in DEM created from LRO NAC images M1105737674LE and M1105759104LE. Elevations are meters above the lunar datum. Notice that elevations are negative.

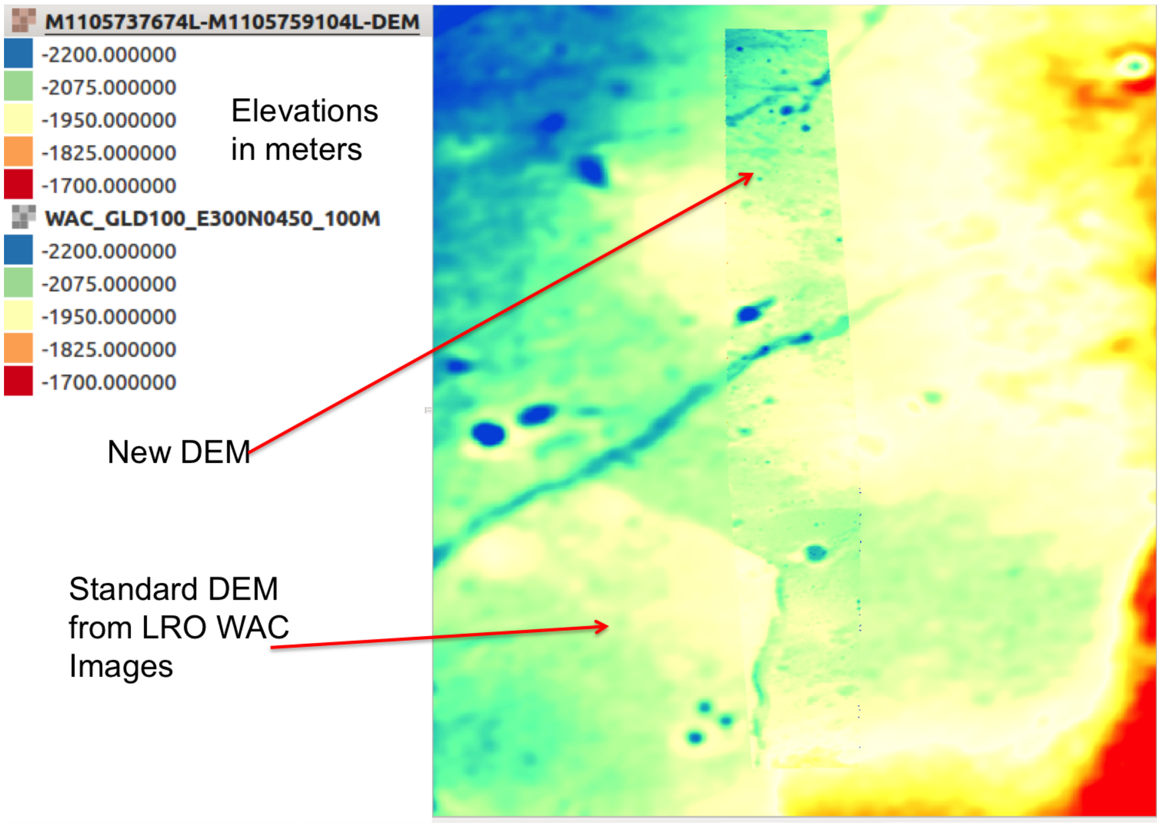


Figure 6.6: The full stereo DEM constructed in this work is overlaid on a lower-resolution reference DEM create from LRO Wide Angle Camera (WAC) images.

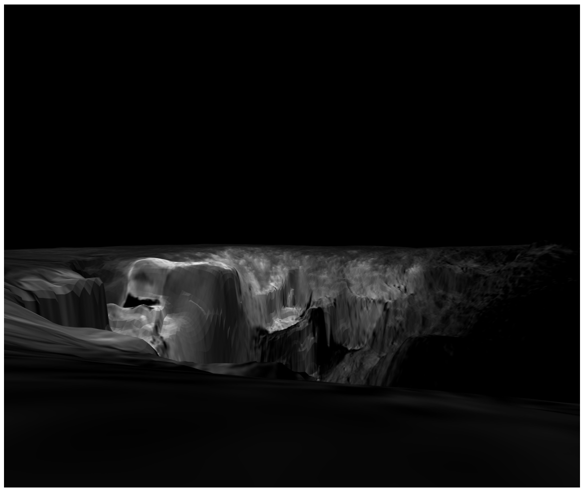


Figure 6.7: A simulated rover's-eye view of a DEM model draped with an orbital image, looking north.

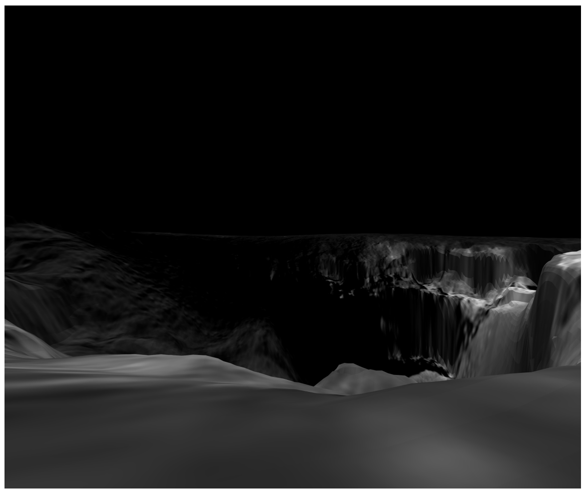


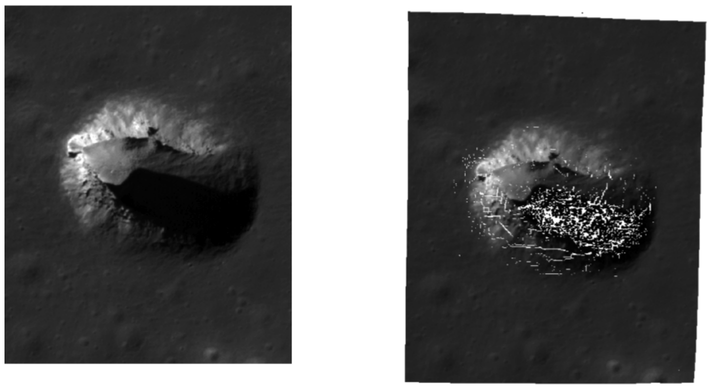
Figure 6.8: A simulated rover's-eye view of a DEM model draped with an orbital image, looking south.

6.1.2 Multiview Stereo Modeling of a Pit from Orbit

Stereo modeling methods designed to operate on pairs of orbital images do not necessarily do a good job handling the steep slopes and vertical cliffs of pit walls, canyons, or mesas. This was observed in the experiment conducted in Section 6.1.1. Incorporating more than two views can greatly improve the model for substantially 3D terrain such as pits. An experiment in multi-view stereo for orbital modeling of pits was conducted using the same pit as discussed in Section 6.1.1.

The experiment used five LROC NAC images with different view angles. These included the stereo pair used in Section 6.1.1, which had emission angles (the angle between the satellite's view and a straight down, or nadir view) of 14° looking east to west (M1105737674L) and 12° looking west to east (M1105759104L), a view 47° looking east to west (M1105701957R), a view close to straight down at 3° looking east to west (M1121075381R), and a view 33° looking south to north (M1136377273R). All of these images had similar illumination angles corresponding to mid-morning local time.

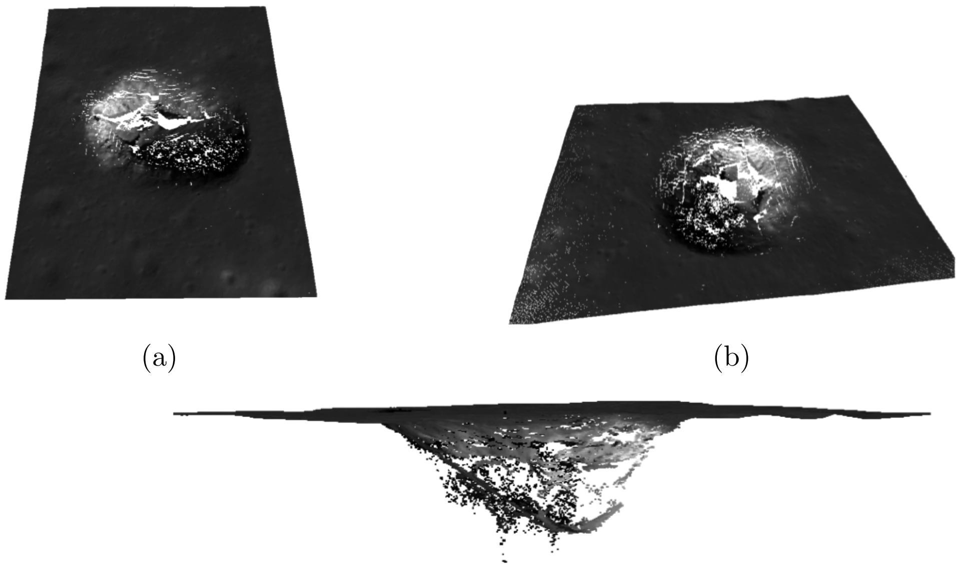
Each image was trimmed to a small region around the pit. Tie points were manually identified in the images and bundle adjustment was performed using USGS's ISIS tool [74]. Ames Stereo Pipeline was used to do the multi-view stereo reconstruction on the bundle-adjusted images [57, 58]. 6.9 shows a view of the pit in one of the images used for the reconstruction and a similar view of the reconstructed model. Fig. 6.11 shows three views of the point cloud model. Note that the region that is shadowed in the images is not well modeled.



(a) (b)

Figure 6.9: Map projection of an orbital image with a straight-down view (M1121075381R) of Lacus Mortis pit 6.9a and top view of 3D point cloud model built using multi-view stereo 6.9b. Note that the shadowed region in the image corresponds to a region with more holes in the point cloud.

There is a discontinuity in the model between the southwest and northwest regions of the floor. This is not consistent with the appearance of the pit in images. Looking at the stereo intersection error computed by the stereo reconstruction software sheds some light on the issue (see Fig. 6.11). Error is much higher in the southwest region. Also, note that the southwest region borders the shadow boundary, which shifts slightly between the different



(a) (b)

(c)

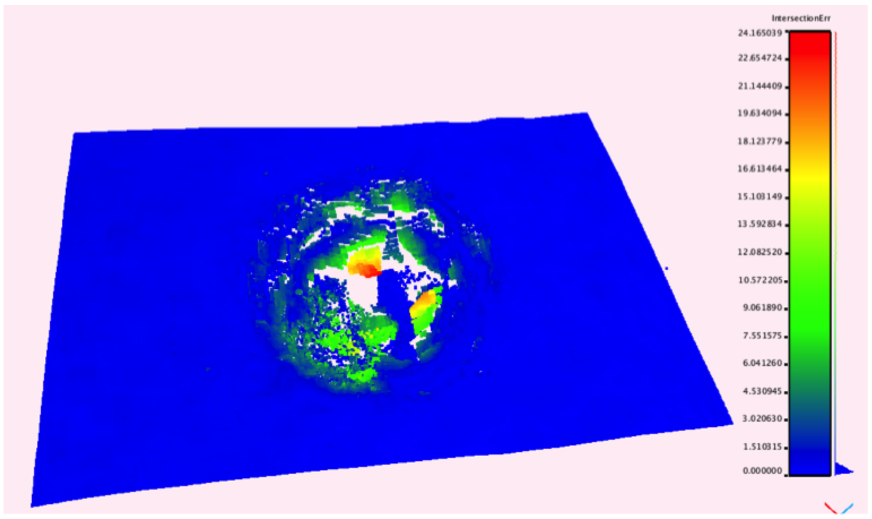
Figure 6.10: Views of a 3D point cloud model of Lacus Mortis pit built using multi-view stereo. The ramp observed in the images can also be seen in the model, at least for the un-shadowed side of the pit. In the shadowed area, because there is little information available to stereo, the model is quite noisy (note noisy black points at the bottom of the pit in the lower image). The top left image shows gaps along the north wall. This could be due to an overhang, or it could be due to a vertical wall that extends to the ground but is not well modeled with only one oblique view that looks north.

images used in this experiment. This sharp shadow boundary may have been incorrectly identified as a feature by the stereo modeling software, causing errors in the model.

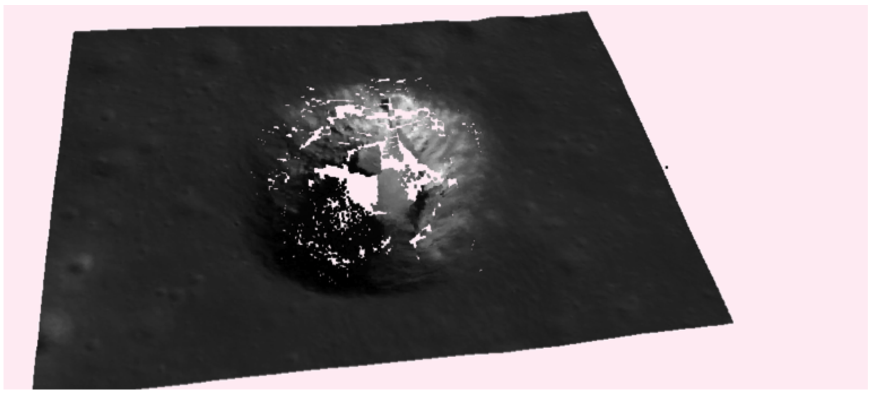
To make the model easier to view, areas of high error were smoothed and the point cloud was converted to a mesh using Poisson Reconstruction [75, 76]. This model can be seen in Fig. 6.12.

This experiment illustrated that multi-view stereo can be successfully used to construct coarse prior models of substantially 3D terrain features from orbital imagery. These are, of course, limited in coverage, incidence angle, and resolution. It also showed that shadows and shadow boundaries are not well handled by this method. Improvements to the model could likely be made with approaches that explicitly consider illumination, such as shape from shadow or shape from shading methods.

The best prior model available should be used for planned-view-trajectory model building, since it will enable the most accurate computation of view and illumination angles, but the prior model will never be perfect. For missions with relatively long duration compared to communication frequency, (and communication delay), partial 3D models might be constructed partway through the mission and compared to the coarse prior model. An understanding of the nature of error in the coarse model could be used to adjust view and illumination angle thresholds to account for uncertainty in the values of those angles, potentially improving performance later in the mission. Alternately, if the mission extends over several local days, the result of a first pass of planned-view-trajectory model building on the



(a)



(b)

Figure 6.11: Stereo intersection error computed by Ames Stereo Pipeline for the pit model (top) and corresponding colorized view (bottom). Note that the southwest floor region that appears disconnected from adjacent patches of floor has very high error. Errors are given in meters.

first mission day could be used as the prior model for a second pass on a later day.

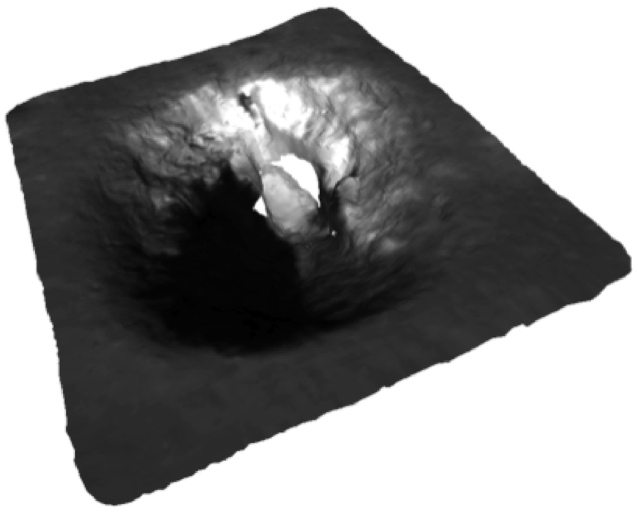


Figure 6.12: Mesh model of Lacus Mortis pit created using multiview stereo

6.2 Constructing the Planning Problem

The coarse prior model must be discretized into planar faces in order to compute view and illumination angles. Chapter 3 showed how this might be done for a simple cylindrical pit model - the circular shape of the cylinder can simply be approximated by a polygon with the desired number of sides. If the prior model is formed from an irregular-shaped outline or from a stereo model, however, the face discretization is not as simple.

Two view target computation algorithms were developed in this work. One picks a starting point, fits a plane to the points within a given radius of that point, picks a new point by moving perpendicular to the face normal direction (or a random point if no uncovered points exist in that direction), and repeats until the number of uncovered points is below some small threshold. The second voxelizes all the points in the coarse prior model and fits a plane to the points in each voxel that contains sufficient points for plane fitting. The voxel-based method ensures that the faces will be evenly spaced, though it is sensitive to the spatial location of the voxel grid (i.e., if a voxel boundary falls close to the feature surface, points that should represent the same surface could be split into two near-parallel surfaces). The point-picking method will tend to 'follow' the surface without regard to arbitrary voxel boundaries, but it is also sensitive to the randomly chosen initial and re-start points. An example output for the voxel-based algorithm is shown in Fig. 6.13. Fig. 6.14 shows a face discretization for a stereo model of Lacus Mortis pit.

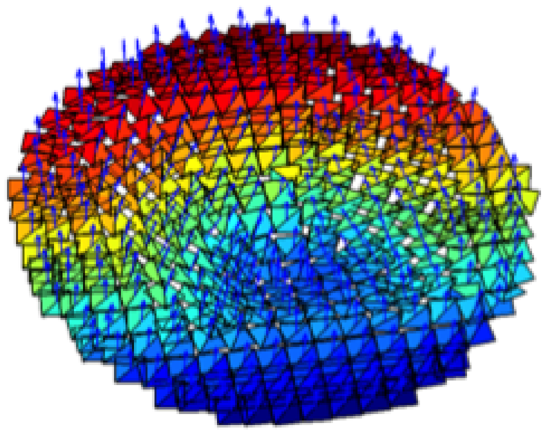


Figure 6.13: Face discretization of the crater shown in Fig. 6.1. Blue arrows indicate face normals.

For any terrain-feature-modeling task, there will be a start and stop date/time. The sun direction can then be computed from ephemeris data. To get more accurate than a simple assumption of 360 degrees of sun angle change corresponds to one local day, the position of the sun in a planet-fixed frame for each time of interest is computed using SPICE [77]. A script written in this research then uses the sun position to create a vector to the sun in a local frame centered on the feature of interest. For simplicity, it is assumed that the sun is a perfect directional light source instead of an area source at a very great distance. This sun direction and the face normals can be used to determine face illumination angles, and

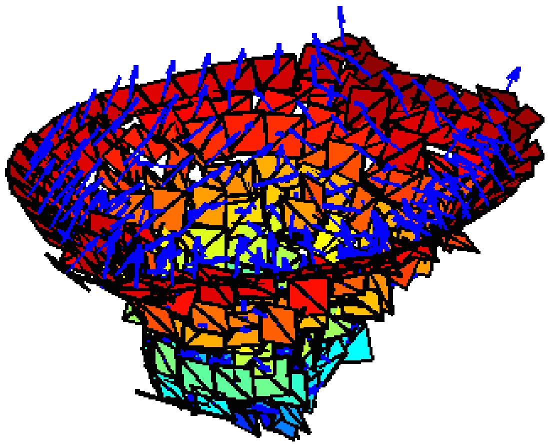


Figure 6.14: Face discretization of the pit modeled in Section 6.1. Blue arrows indicate face normals.

face positions combined with a coarse prior model of the pit can be used to do ray-tracing to determine shadowed faces [78].

To determine the visibility of faces from various rover positions, one can either generate rover positions and views and test which face(s) they can see (generate and test) or determine for each face which rover positions can see it (synthesis). A synthesis approach for visibility computation was used for the simple simulation example discussed in Chapter 3. For all other examples in this work, a generate and test approach was used, (or a receive-as-input and test approach). For pits, positions were generated at regularly-spaced intervals around the pit outline. For the rocks discussed in Section 7.2, positions were received as input.

To determine if faces are visible in a given view from a robot position (the ‘test’ part of generate and test), ray tracing is done from the camera position [78]. The number of rays to use for each camera image is based on the size of a face, (the view-target computation algorithms described above generate faces of uniform size, if faces were not uniform size, the minimum face size should be used), and the spread between adjacent rays at the maximum expected viewing distance. The distance between ray tips at that distance should be somewhat less than the face size to ensure that each visible face is hit by at least one ray. For faces that are intersected by at least one ray, (potentially visible faces), the four vectors between the camera center and the face corners are checked against the camera field of view to ensure that the entire face can be seen. Two dimensional view angles, computed by comparing the direction of vectors from the camera to the face centers with the face normals, are stored for each valid position-face pairing.

Robot path planning is a well studied area, so path lengths between robot positions are computed using existing path-planning algorithms. If the terrain is obstacle-free, (except for the target feature), then a set of straight-line distances can be measured between nearby positions, these can be added to a graph, and Dijkstra’s algorithm can be used to compute an all-pairs shortest path solution to get the distance from each position to each other position [72]. This was the solution implemented in testing for this work. If the terrain is more complex, an algorithm like A* [36] or D* [37, 38] can be used to compute paths that avoid obstacles.

The value function used for planning was essentially the same as that described in Section 3.3. View and illumination angle limits (including relative view and illumination angle limits) were modified on a per-experiment basis. Limits used in simulation experiments were somewhat arbitrary, but for real-world problems, limits should be set according to the procedures outlined in Chapter 4.

6.3 View Trajectory Planning

6.3.1 Developing an Algorithm to Solve the OPTWIND

The Time Opportunity and Inter-Node Target Relationship Planner (TO-INTReP), shown in Algorithm 1, solves OPTWIND and generates view trajectory plans. The algorithm assumes that time has been coarsely divided, based on significant changes in availability or value of positions, similar to the start time, end time, and time divisions, as described in Section 3.1. The algorithm first computes a value for each position at each time, as the sum of potential values for each target type available from that position, weighted by one over the total length of time during which that target type is available from any position. Positions within each coarse time division are then ordered by this value. Starting with the top position for each time division, the algorithm tries to find a feasible trajectory that includes one position per time division and meets distance constraints, proceeding to lower-valued positions for a given time if necessary. From the feasible trajectory, it tries to add positions, ordered by the difference in the set of targets available (the larger the difference, the better), within each coarse time division and remaining within the distance constraints. It does a local search to reduce the trajectory distance. It then determines which targets do not have the desired number of views, and tries to add positions, ordered by the number of target types they meet constraints for, and thus improve value. The insert-by-set-difference, insert-reduce-trajectory-distance, and insert-to-satisfy-constraints are repeated until the change in paths is lower than some specified threshold, or a specified maximum number of iterations is reached.

Eq. 6.1 and eq. 6.2 are copies eq. 5.7 and eq. 3.3, provided for convenience since these are used by Algorithm 1.

$$\begin{aligned} a\left(n_i, p_k\right) = \prod_{z=1}^{Z} &\left[min\left[max\left(\beta_{ikz} - A_{z,min}, 0\right), 1\right] * \right. \\ &\left. min\left[-min\left(\beta_{ikz} - A_{z,max}, 0\right), 1\right]\right] \end{aligned} \tag{6.1}$$

$$b\left(t_i, p_j\right) = \begin{cases} 1, & \text{if } \left|s\left(t_i, p_j\right)\right| < I \\ 0, & \text{otherwise} \end{cases} \tag{6.2}$$

6.3.2 Testing with Partial Implementation of TO-INTReP

The algorithm was implemented to determine if TO-INTReP could successfully plan a set of views for the terrain feature modeling problem. This was seeded with an initial feasible path. The capability to add nodes based on the set difference between faces visible from nodes on the path and those not yet on the path was included. The capability to add nodes that met constraints was partially implemented. The algorithm identifies the set of faces which are visible at some time but do not have full value in the current plan, the "goal faces".

Algorithm 1 Time Opportunity and Inter-Node Target Relationship Planner (TO-INTReP)

```
 1: for $t_i \in \mathcal{T}$ do
 2:     for $l_j \in \mathcal{L}$ do
 3:         $n = (t_i, l_j)$
 4:         $Val\,(n) \Leftarrow \sum_{p_k \in \mathcal{P}} \{ \frac{a(n,p_k)}{\sum_{t_n \in \mathcal{T}} b(t_n,p_k)} \}$    ▷ Here $a$ is used as in eq. 6.1, and $b$ is used as
    in eq. 6.2
 5:     end for
 6:     $PositionTimes\,(t_i, :) \Leftarrow Sort\,(Positions, Val)$
 7: end for
 8: $Count \Leftarrow 1$
 9: $MaxCount \Leftarrow NumPositions$
10: $Path \Leftarrow \emptyset$
11: while $(Path == \emptyset)\, AND\, (Count < MaxCount)$ do
12:     $Graph \Leftarrow BuildGraph\,(StartPosition, PositionTimes\,(t_i, 1 : Count))$
13:     $Path \Leftarrow FindPath\,(Graph)$
14:     $Count \Leftarrow Count + 1$
15: end while
16: if $Path \neq \emptyset$ then
17:     $Count2 \Leftarrow 1$
18:     $MaxCount2 \Leftarrow N$
19:     $ModelVal \Leftarrow 0$
20:     $MaxModelVal \Leftarrow M$
21:     while $(ModelVal < MaxModelVal)\, AND\, (Count2 < MaxCount2)$ do
22:         $PrevPath \Leftarrow Path$
23:         for $t_i \in \mathcal{T}$ do
24:             $Path \Leftarrow AddNodesSetDifference\,(Path)$
25:         end for
26:         $Path \Leftarrow ReduceTravelDistance\,(Path)$
27:         $Path \Leftarrow AddNodesMeetConstraints\,(Path)$
28:         $ModelValue \Leftarrow ComputeModelValue\,(Path)$
29:         $Count2 \Leftarrow Count2 + 1$
30:     end while
         return $Path$
31: else
         return $\emptyset$
32: end if
```

Table 6.1: Results from running a partial implementation of TO-INTReP

Algorithm	Start Pos.	Value	Exp. Value	# Views	Path Length
TO-INTReP	1	100	62.8	14	707
TO-INTReP failurerate	1	90	66	13	505
TO-INTReP	2	100	56.7	14	808
TO-INTReP failurerate	2	90	59.6	13	606
TO-INTReP	3	100	62.8	14	707
TO-INTReP failurerate	3	90	66	13	505
TO-INTReP	4	100	69.4	14	606
TO-INTReP failurerate	4	90	73	13	404
TO-INTReP	5	100	76.8	14	505
TO-INTReP failurerate	5	90	80.7	13	303
TO-INTReP	6	100	69.4	14	606
TO-INTReP failurerate	6	90	73	13	404

Positions that view goal faces and meet constraints are identified. Nodes corresponding to these positions and to times when any of the goal faces are visible are then tested for addition to the path. If the node increases the plan value, it is added. The reduce distance function was not included in this implementation.

As a different heuristic for reducing distance, the failure rate concept from Section 3.3 was used in the add nodes to meet constraints step. Instead of testing against the model value, the expected model value was used.

6.3.3 Experimental Results

Fig. 3.8 shows one test scenario. There were 6 potential rover positions, 6 target faces, and 5 coarse time divisions. Results are shown in Table 6.1 for the TO-INTReP partial implementation, using model value for *AddNodesMeetConstraints*, and using expected model value (TO-INTReP failurate). The algorithms are run starting from each of the potential rover positions. Note that TO-INTReP succeeds in achieving 100% model value for each starting position. TO-INTReP with the failure rate modification did not achieve full model value, but it consistently had a higher 'expected value' and shorter path length when starting from the same starting position. The failure rate heuristic assumes that views that happen sooner are more likely to occur, while views that happen later are less likely to occur because the robot may fail before they can be captured. A lower value result with higher expected value is likely a consequence of this effect.

The partial TO-INTReP implementation was also tested against a pit with 24 faces (12 angular divisions and two vertical), 24 positions, and 12 coarse time divisions. It succeeded in reaching 100% model value from each starting position.

6.3.3.1 AI Planning for the OPTWIND

The AI planning community has created numerous general-purpose planners. Many of these, especially those used for the International Planning Competition. For the International Planning Competition, planning problems and their domains are written in PDDL. These planners can be tested on OPTWIND instances if these instances can be effectively expressed in PDDL.

6.3.4 Representing OPTWIND in PDDL

The knowledge engineering tool itSIMPLE was used to create a partial expression of the OPTWIND for pit modeling in PDDL [79, 80]. The itSIMPLE package enables users to construct planning scenarios in the Unified Modeling Language (UML) and automatically translate them to PDDL. Fig. 6.15 shows the UML class diagram created for this problem. Note that the agent is a rover, of general class "locatable" that can be located at a "LocationTime" node. This corresponds to the nodes in Section 5.3. Each node has an associated Location and Time. Targets may be visible or not from a Location, and lit or not at a Time. The rover has two possible actions: *move* takes the rover from one Location to another and lasts one unit of time, and *view* sets a Target's viewed attribute to true and is instantaneous. The available attribute of a Location can be set to true during times that correspond to its associated coarse time division and false otherwise.

In addition to the UML class diagram, the itSIMPLE user defines a state machine diagram that provides more information on defined actions. From these inputs, a PDDL domain file can be generated.

Before planning, a description of the problem is also needed. Multiple instances of each class can be added to the problem's object repository, and then snapshots defining the initial and goal states are defined. For the OPTWIND problem, the object repository has one RoverObject, 6 Targets, 6 Locations, 5 Times, and 6 x 5 LocationTimes (labeled with the format `loctime_<time>_<location>`).

In the initial state, each LocationTime is linked to its associated location with isat and its associated time with istime. Assuming that the set of locations lie in a circle surrounding a pit, each LocationTime node is connected to its two neighbors in the same time, as well as itself and its location neighbors at the next time. Relations lit and visible are set according to the scenario in Section 6.3.2, although visible is binary on whether it meets the absolute view angle threshold.

In the goal state, the rover is at `loctime_5_6` and faces 1, 2, 3, 5, and 6 are viewed (4 is never lit, so it cannot be viewed). Resulting domain and problem files for PDDL 2.2 can be found in Appendix C.

Several aspects of the OPTWIND are not represented in this domain. These include restrictions on not visiting LocationTime nodes at the wrong time (the available attribute of the LocationTime class is a potential way of implementing this, but setting this value independently of rover actions would require the timed-initial-literals functionality of PDDL, and working with this functionality in the itSIMPLE environment is not well documented).

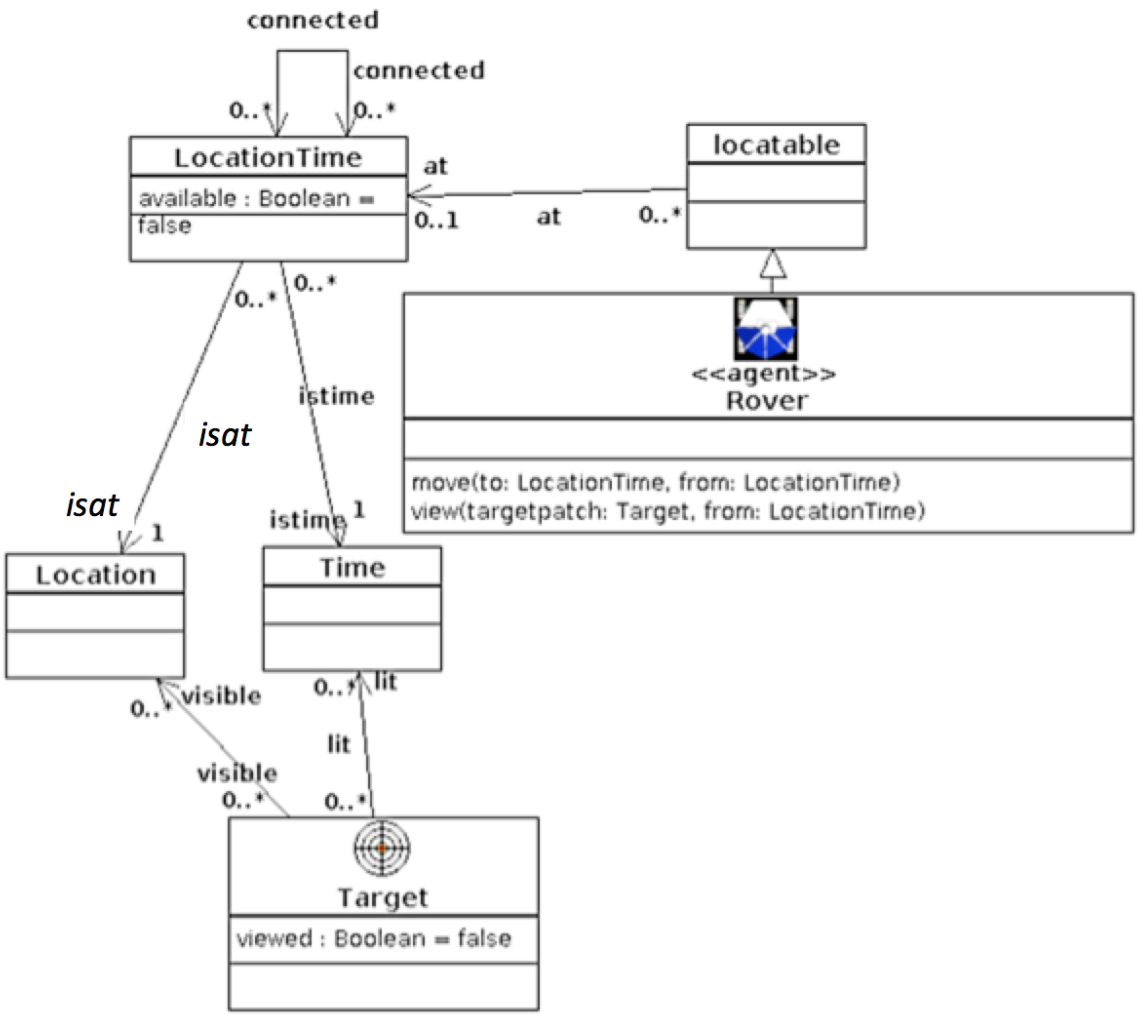


Figure 6.15: UML Class Diagram representation of the OPTWIND for pit modeling

Both lighting and visibility are binary, so enforcement of relative constraints is not possible without a different representation. The rover has a specified end node, which is part of the OPTWIND formulation, but not needed for the pit modeling application. Representation of distance is not yet included in this domain, and the time of the move action is constant, which would not be the case if paths were different lengths. Metrics can be added to evaluate plans in the PDDL/itSIMPLE framework, but none have yet been defined for this domain.

6.3.5 Experimental Results

A set of planners are included with the itSIMPLE distribution. Table 6.2 shows results from all the planners included in itSIMPLE that were judged to be applicable to the problem.

Only three of the tested planners successfully generated plans. The plan generated by LPG-td reached the goal node, but no faces were viewed. SGPlan 5 and SGPlan 6 generated

Table 6.2: Results from running planners on a partial OPTWIND description

Planner	# Actions	# faces Viewed	Plan Found?
SGPlan 5 - Version: 5.2.2 Linux	14	5	yes
MIPS-XXL - Version: 1.0 Linux	0	0	no
LPG-td - Version: 1.0 Linux	7	0	yes
MIPS-XXL - Version: IPC-6 v1.2 Linux	0	0	no
SGPlan 6 - Version: IPC-6 (2008) LInux	14	5	yes
Metric-FF - Version: Linux	0	0	no
FF 2.3 - Version: FFv-2.3 Linux	0	0	no

the same plan and achieved everything in the goal state: the goal node was reached and all 5 target faces were viewed. The plan from SGPlan 6 is shown in Fig. 6.16.

```
0.001: (MOVE ROVEROBJECT LOCTIME_2_2 LOCTIME_1_1) [1.0000]
1.002: (MOVE ROVEROBJECT LOCTIME_2_1 LOCTIME_2_2) [1.0000]
2.003: (VIEW ROVEROBJECT TARGET3 LOCTIME_2_1) [0.0000]
2.004: (MOVE ROVEROBJECT LOCTIME_3_2 LOCTIME_2_1) [1.0000]
3.005: (VIEW ROVEROBJECT TARGET6 LOCTIME_3_2) [0.0000]
3.006: (MOVE ROVEROBJECT LOCTIME_3_3 LOCTIME_3_2) [1.0000]
4.007: (MOVE ROVEROBJECT LOCTIME_3_4 LOCTIME_3_3) [1.0000]
5.008: (VIEW ROVEROBJECT TARGET2 LOCTIME_3_4) [0.0000]
5.009: (MOVE ROVEROBJECT LOCTIME_4_3 LOCTIME_3_4) [1.0000]
6.010: (VIEW ROVEROBJECT TARGET5 LOCTIME_4_3) [0.0000]
6.011: (VIEW ROVEROBJECT TARGET1 LOCTIME_4_3) [0.0000]
6.012: (MOVE ROVEROBJECT LOCTIME_5_4 LOCTIME_4_3) [1.0000]
7.013: (MOVE ROVEROBJECT LOCTIME_5_5 LOCTIME_5_4) [1.0000]
8.014: (MOVE ROVEROBJECT LOCTIME_5_6 LOCTIME_5_5) [1.0000]
```

Figure 6.16: Plan generated by SGPlan 6

6.4 View Trajectory Execution and Detailed Model Construction

Since many great tools for 3D model construction from images exist and are freely available for use, it was decided that model construction from images collected during execution of a view trajectory would be done with existing tools. Several of these tools were evaluated on data from analog planetary features. A brief comparison of model building methods is given in Section 6.4.1. Section 6.4.2 shows modeling of lava tube skylights, a crater, and a canyon without explicit view trajectory planning.

6.4.1 Comparison of Model Building Methods

Model building from images has two steps. First, the relative positions of the cameras are determined in a structure-from-motion step, and then a dense reconstruction is computed given those positions. In this subsection, as presented in [15], several existing modeling methods are evaluated, a combination of structure-from-motion and dense reconstruction algorithms, to determine which is most effective on data from an analog field site.

Camera reconstructions portions of King's Bowl pit, (see Section B.1), were evaluated using a set of images taken at positions E14 at time 17:19 and E13 at time 16:11 (shown in Fig. 6.17). Images were taken with a Canon EOS Rebel T3 DSLR camera with a 50mm lens. Reconstructed models from images were compared to a ground truth LIDAR model of the site [81].

The pipelines evaluated were the Multi-View Environment (MVE) in [69], VisualSFM [66, 67] for structure-from-motion and MVE for dense reconstruction, VisualSFM for structure-from-motion and PMVS/CMVS [64, 65] for dense reconstruction, OpenMVG [68] for structure-from-motion and MVE for dense reconstruction, Bundler [62, 63], Bundler for structure-from-motion and PMVS/CMVS for dense reconstruction, and Bundler for structure-from-motion and MVE for dense reconstruction. The combination of approaches can be seen in Table 6.3.

OpenMVG implemented two methods for computing global rotation from a list of relative estimates. The first method employed Martinec's algorithm detailed in [82] and the second method employed an algorithm developed by Chatterjee, et. al. detailed in [83]. OpenMVG1 in Table 6.3 refers to the first method and OpenMVG2 refers to the second method.

Fuhrman et. al. developed the Multi-View Environment (MVE) in [69] that takes images as input and outputs a dense point cloud. Their algorithm includes structure-from-motion, multi-view stereo reconstruction, and dense point cloud generation from multi-scale data. The structure-from-motion and dense point cloud generation are run together on the input data and compared to other structure-motion-algorithms combined with the MVE dense point cloud generation. The full reconstruction pipeline can be run using the MVE software. Alternatively, MVE can take as input sparse reconstruction from another structure-from-motion software and output a dense reconstruction.

The Clustering Views for Multi-View Stereo (CMVS) [65] and Patch-based Multi-view Stereo [64] algorithms of Furukawa, et. al. were also evaluated to generate dense point

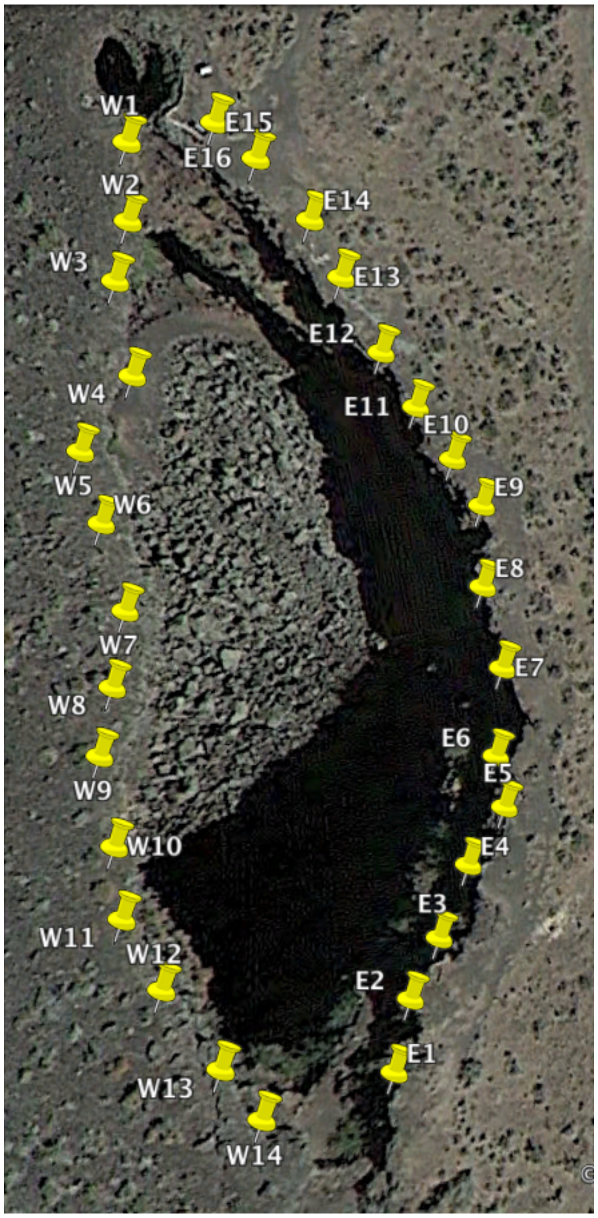


Figure 6.17: Overhead view of Kingsbowl pit. Potential positions are marked by yellow push-pins. (View from Google Earth [3]. Base image from [4])

Table 6.3: Dense Reconstruction Pipelines

Structure from motion	Reconstruction	Coverage	% camera values not found in ground truth
MVE	MVE	5.3	75.7
VisualSFM	MVE	20.8	42.4
VisualSFM	**CMVS/PMVS**	3.8	**23.5**
OpenMVG1	MVE	2.9	92.4
OpenMVG2	MVE	2.9	91.4
Bundler	—	14.5	**24.5**
Bundler	**CMVS/PMVS**	3.6	**22.9**
Bundler	MVE	22.4	45.0

clouds from the camera images.

Each of the methods reconstructed a camera model of the pit of varying quality. The quality of the model was measured by comparing to ground truth collected by a LIDAR. Each camera model was coarsely registered to the LIDAR model using handpicked points. Following the coarse registration, a fine registration was computed using the Iterative Closest Point (ICP) algorithm [84, 85, 86].

In order to accurately compare the LIDAR ground truth model to the camera models generated by the pipelines, the vertices in the models were voxelized using a density of 0.2 meters. The models could then be directly compared to determine the statistics in Table 6.3. The metrics compared are coverage, or the percent of the voxels that are occupied in the ground truth LIDAR model that are also occupied in the camera model, and percent of values in the camera model that are occupied in the the ground truth model. This latter metric gives a measure of how much spurious or 'noisy' data the stereo reconstruction creates.

An analysis of the results concludes that the structure-from-motion component should be Bundler or VisualSFM. When combined with dense reconstruction by CMVS/PMVS, these produce the least noisy models. When combined with MVE, they produce the best coverage, but noisier results. The other methods (i.e. OpenMVG and the structure-from-motion component of MVE) produced significantly poorer results. It is also worth noting that by default, CMVS/PMVS sets a restriction of 10 degrees on minimum angle between views of a point for it to be included in the reconstruction. Relaxing this restriction increases CMVS/PMVS coverage. It must also be noted that these combinations of methods were run on a single dataset with fairly consistent lighting conditions. This investigation was admittedly not comprehensive enough to draw sweeping conclusions about these methods, but it was enough to determine that Bundler or VisualSFM for SFM combined with CMVS/PMVS for dense reconstruction would be sufficient for demonstrating planned-view-trajectory model building.

6.4.2 Model Building for Planetary Analog Features

This section discusses 3D models of planetary analog features. These models demonstrate the feasibility of modeling large-scale planetary terrain features from images. Of particular interest is the demonstration modeling from pit, crater, and canyon rims.

These models were built without explicit view trajectory planning. The effects of transient directional lighting, already less extreme on Earth than on the Moon due to Earth's atmosphere, were mitigated by collecting all images of a sunlit feature in a relatively short period of time so light direction did not change that much and/or by collecting images under cloudy skies. The modeling of Meteor Crater, however, was negatively impacted by not planning for transient lighting.

Section 6.4.2.1 discusses the modeling of a lava tube skylight pit from images and comparison with a ground truth LIDAR model. Section 6.4.2.2 shows modeling of several features of a lava tube cave. Section 6.4.2.3 shows modeling of a rock feature in the Grand Canyon. Section 6.4.2.4 shows modeling of Meteor Crater.

6.4.2.1 Indian Tunnel

A skylight pit near the southern end of the Indian Tunnel lava tube cave was modeled with images (see Section B.2). All images were collected within a 2 hour timespan, and cloudy skies ensured that lighting was mostly global instead of directional. 205 images total were collected from 14 stations around the rim of the skylight. Tripod stations were spaced with a target distance of roughly 4.5 meters. Stations were adjusted as needed to accommodate hazardous terrain, and precise measurements of station positions were not done. At each station, the tripod was manually leveled by eye using an integrated bubble level. The tripod was adjusted such that the camera tilt was zero. The camera was panned to the left-most edge of the pit (in the current view). Additional views were taken by tilting down by 22.5 degrees to 45 degrees and panning by 30 degrees until the right-most edge of the pit was reached.

Two Canon EOS Rebel T3 DSLR cameras with manually adjustable zoom lenses, 18mm focal length, and field of view of approximately 45 degrees vertical and 63 degrees horizontal were used to take images of the skylight. Each image was taken with auto-exposure using center-weighted average metering.

Other camera parameters were fixed for all images, including ISO 100, aperture F8 and white balance of 5200K (daylight).

A 3D model of the pit was constructed using Bundler for bundle adjustment and CMVS/PMVS for dense reconstruction [62, 63, 64, 65]. Fig. 6.18 shows a view of the colorized point cloud created using this method. A mesh model was generated by applying a Poisson reconstruction [75, 76], and the resulting model is shown in Fig. 6.19. (The construction of this model was also reported in [15]).

To evaluate the success of the reconstruction, the point cloud built from camera images was compared against a ground-truth point cloud built using a FARO laser (this error analysis was first reported in [16]). Because structure from motion does not preserve scale, the

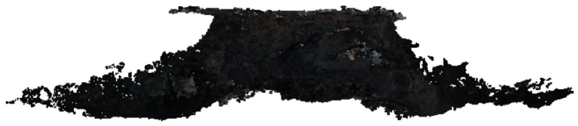


Figure 6.18: Side view of colorized point cloud model created from dense reconstruction from CMVS/PMVS2. The skylight pit is in the center, and the segments of the lava tube cave can be seen extending on either side.

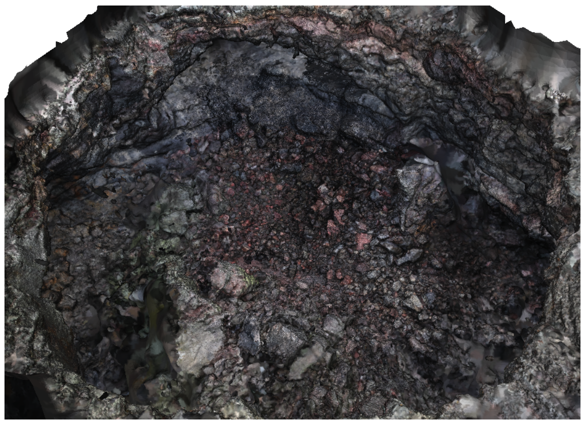


Figure 6.19: Mesh model of a skylight pit in Indian Tunnel cave created with Bundler, CMVS/PMVS, and Poisson Reconstruction

camera reconstruction first had to be scaled. The data also had to be rotated and translated to match the frame of the ground truth model. Initial alignment was done by manually identifying a small set of features in both models and using these to compute scale, rotation, and translation. In a mission scenario, scale could be obtained from knowledge of a robot's relative position between pairs of images, or from comparing a pit model to orbital images.

After initial alignment, the camera reconstruction was cropped to the same bounds as the ground truth model. While the ground truth model has extensive surface coverage, the focus of the camera modeling was on the pit walls and floor (the surface terrain is only covered in a few sparse patches), so surface terrain was cropped out of both models before comparison. After cropping, both models were loaded with CloudCompare software [87]. Fine alignment and scale was done using the iterative closest point method [86], then distance between the two point clouds was calculated. A maximum distance of 8.1m, a mean distance of 0.042m, and a standard deviation of 0.16m were computed. Fig. 6.20 and Fig. 6.21 show a more detailed view of the point cloud distance error. As can be seen in Fig. 6.20, distance errors above 1m (including the 8.1m maximum distance) tend to be outliers, which could easily be removed by filtering. The vast majority of the points have distance errors in the single-digit centimeter range (see histograms in Fig. 6.22 and Fig. 6.23). Fig. 6.24 shows detail for the error evaluation over the range 0cm to 10cm. This indicates that error tends to be lower on the pit walls, and higher at the extremes of the visible tunnel region. This is not surprising given the more oblique viewing angles and greater distance from the camera for the tunnel segments. There also appears to be high error in some areas of the pit floor. While some of this appears to be noise, other high-error patches are due to gaps in the ground-truth data, likely caused by boulders occluding the laser's view.

Fig. 6.25 shows the ground-truth model overlaid on the distance error evaluation to give an indication of model coverage. The lower corner of the lava tube segment on the left and the lower portion of the small tributary lava tube on the bottom, for example, are not included in the camera model. The ground-truth model is also denser than the camera model. Note that in some cases, as illustrated in Fig. 6.26, apparent error in the camera point cloud results from gaps in the ground-truth data.

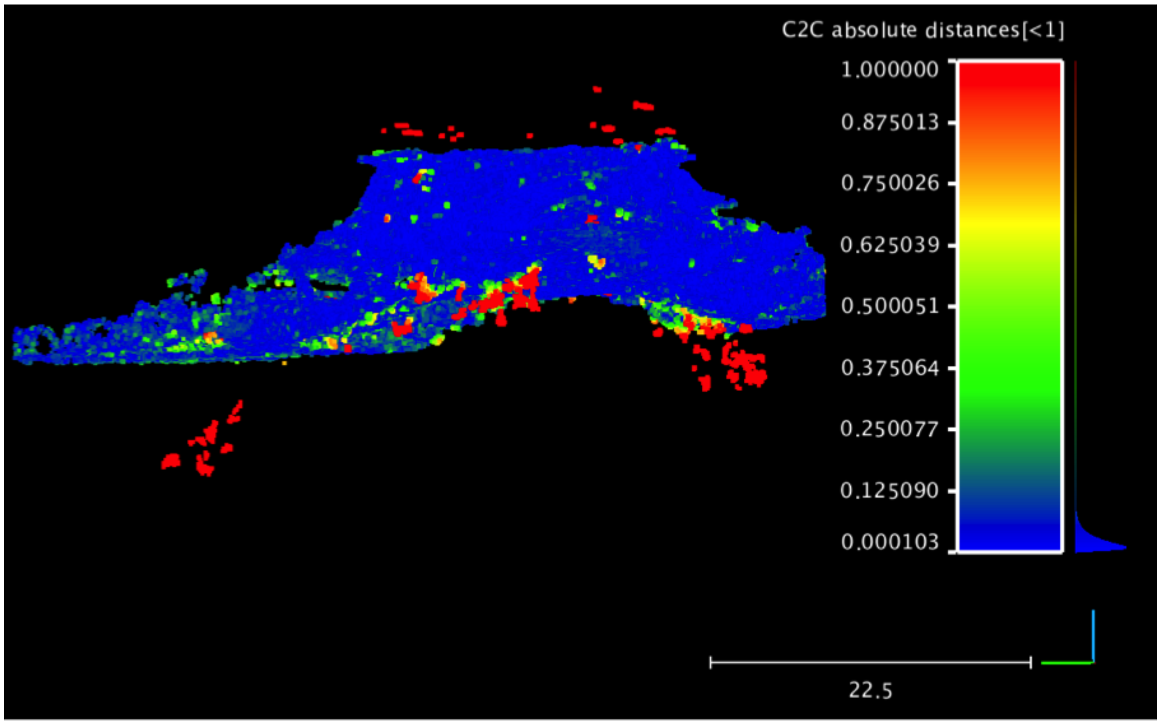


Figure 6.20: Error evaluation for point cloud model, side view. Distances are in meters.

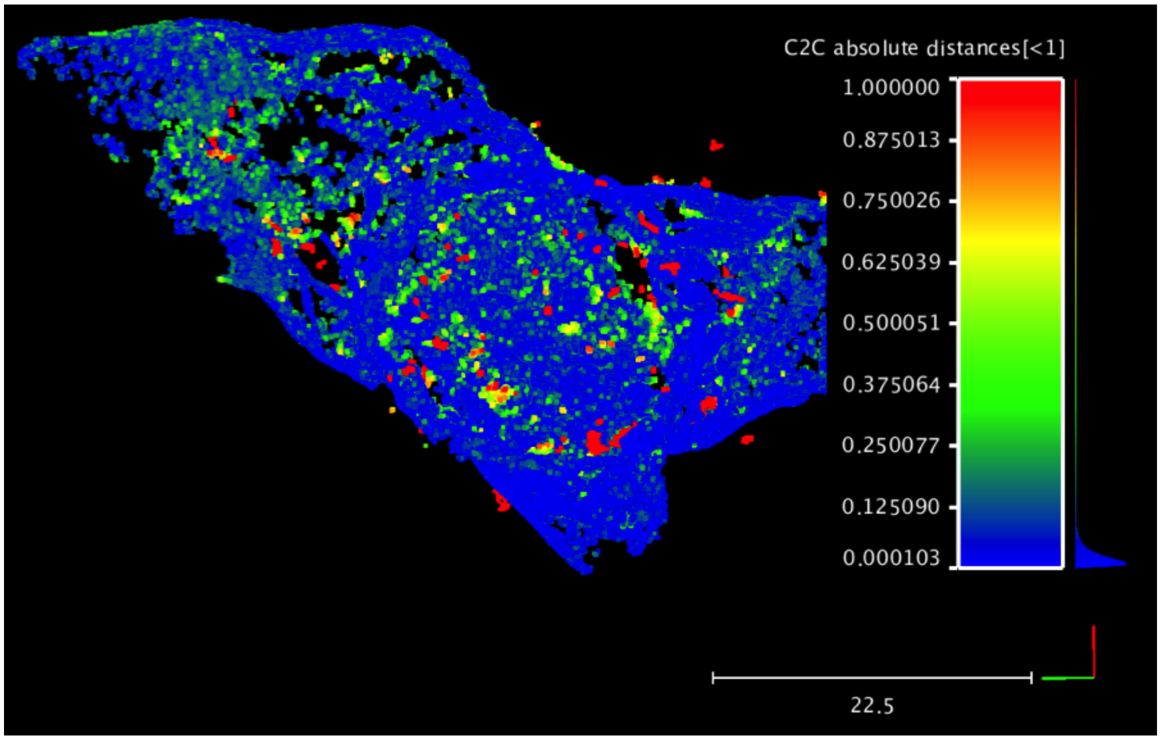


Figure 6.21: Error evaluation for point cloud model, top view. Distances are in meters.

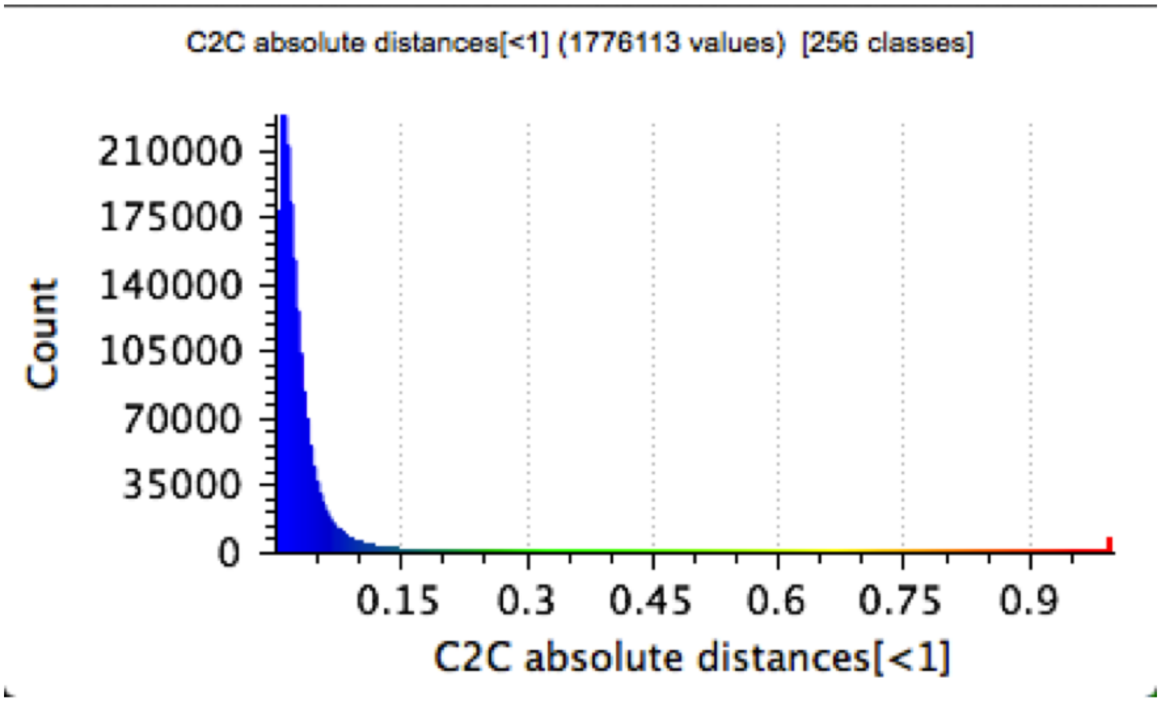


Figure 6.22: Histogram for error distances for camera reconstruction model over the range of 0m to 1m

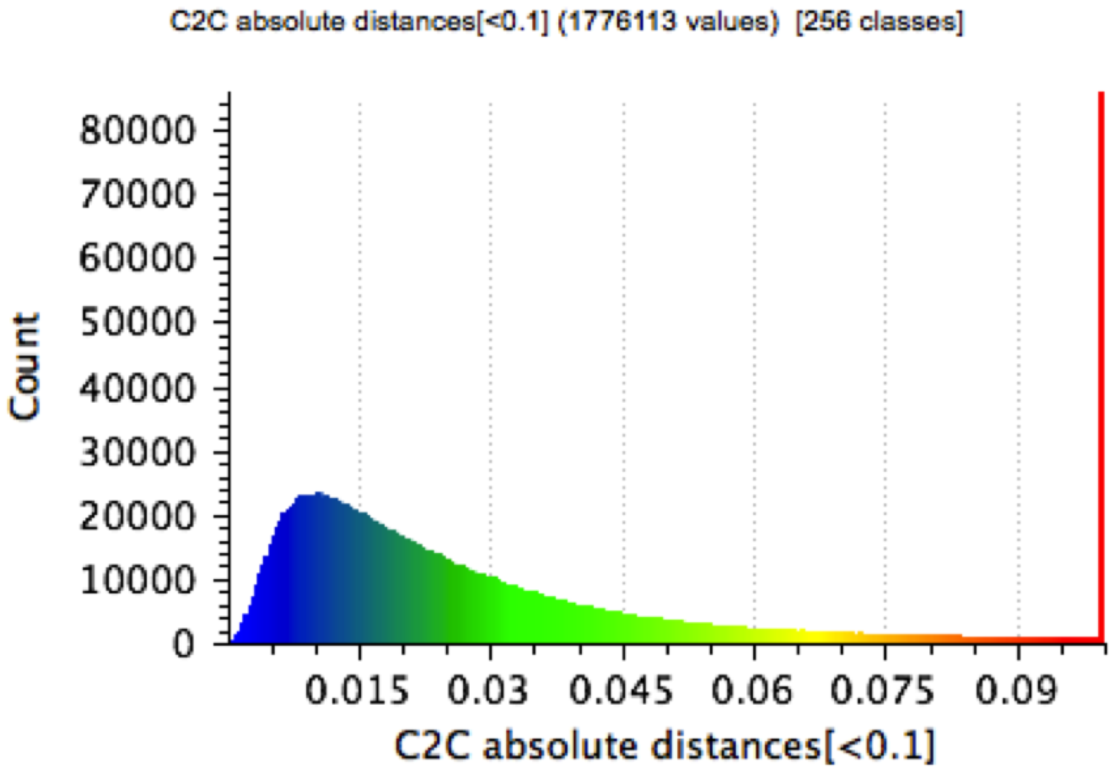


Figure 6.23: Histogram for error distances for camera reconstruction model over the range of 0 to 10cm

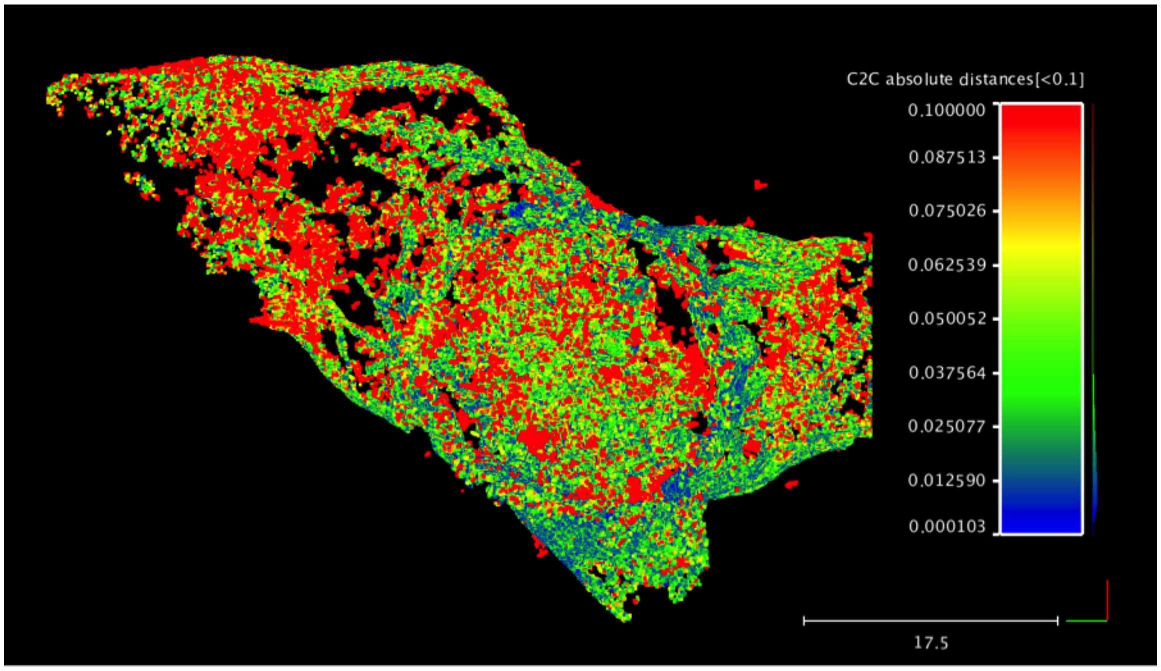


Figure 6.24: Error evaluation for point cloud model, top view, showing error range 0 to 10cm. Distances are in meters.

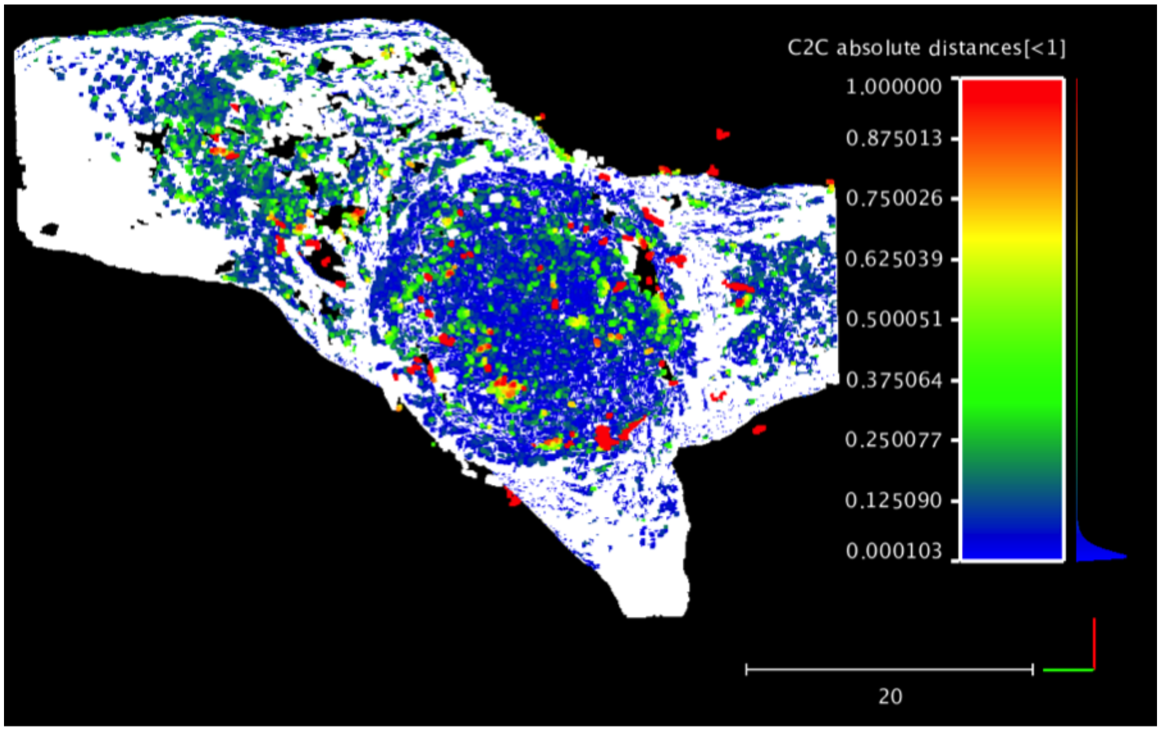


Figure 6.25: Overlay of ground-truth data (white) on error evaluation (same view as Fig. 6.21) to show coverage.

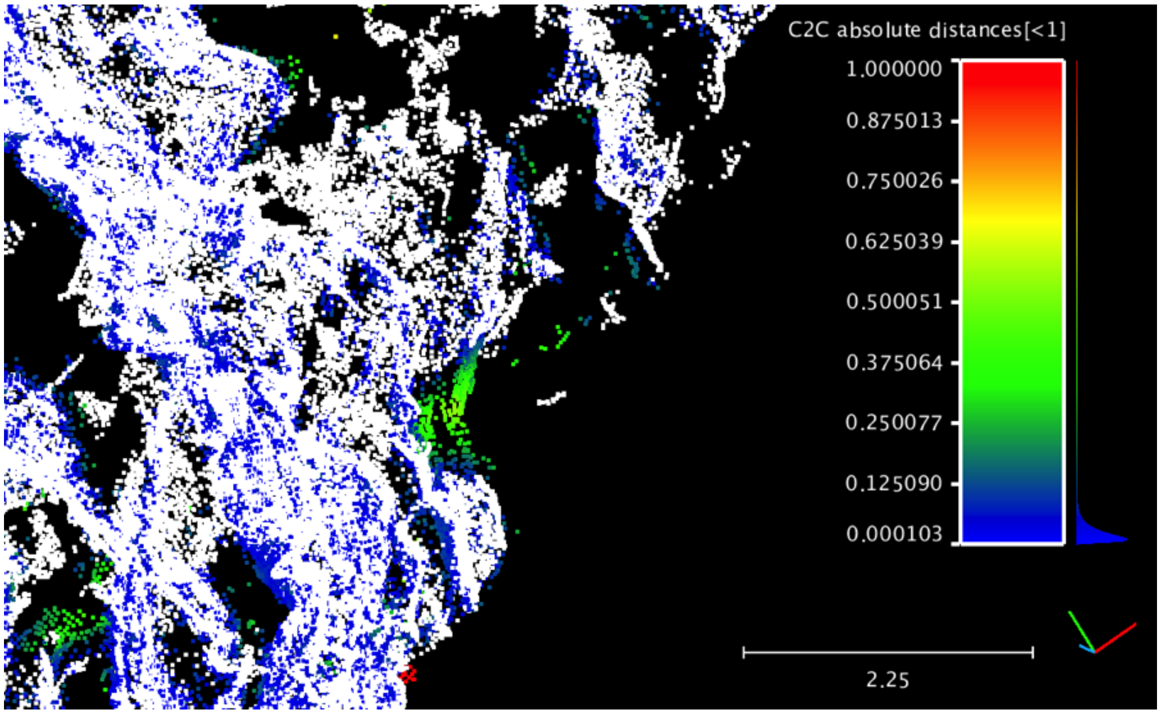


Figure 6.26: A close-up of model with white ground-truth points and error-colored camera model points shows that in some cases, like the green patch in the center of this image, the apparent error is likely due to a gap in the ground-truth data. Distances are in meters.

6.4.2.2 Pluto's Cave

The first attempt in this work at modeling planetary analog features from images was done at Pluto's Cave (see Section B.3). A hand-held Canon EOS Rebel T3 DSLR camera was used to collect a dataset of over 1700 images in the field. Images of features in daylight were taken over short periods. Images inside the cave were taken using a camera-mounted LED array. This setup meant that the phase angle between the view and illumination vectors was always small, except for areas near an entrance or skylight.

Selected images were used to do 3D reconstruction of several segments of terrain. Features were detected and matched across images using SIFT [70], bundle adjustment was done using Bundler [62, 63], and dense reconstruction was done using CMVS/PMVS [64, 65].

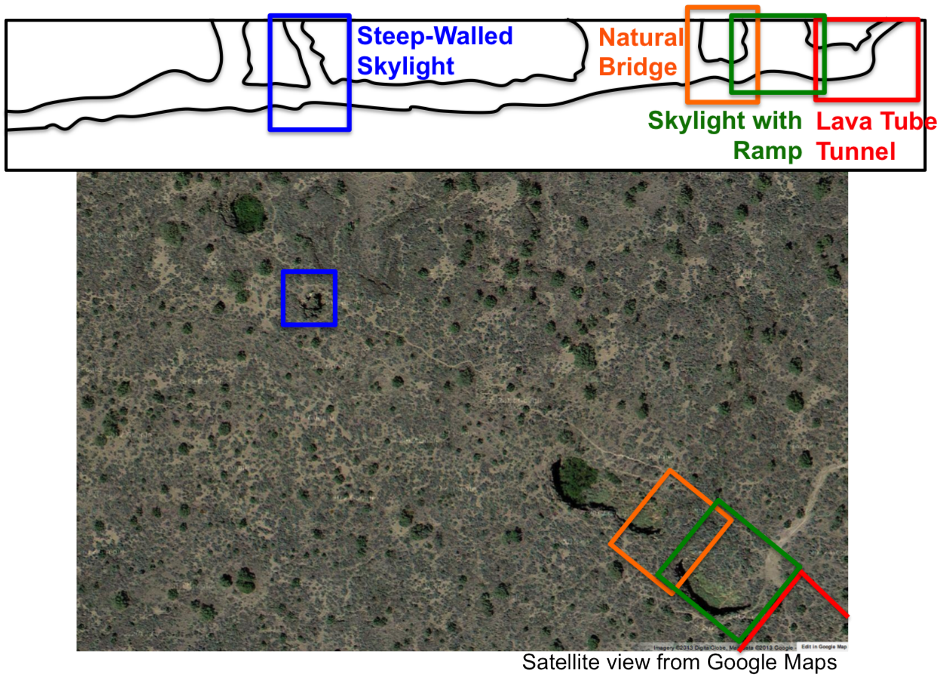


Figure 6.27: Overview of features in Pluto's Cave

Fig. 6.28 shows one of the images of a steep-walled skylight, taken from inside the cave, and an overhead view shows similarities to the Marius Hills Hole on the Moon, a pit believed to be a skylight. Fig. 6.29 shows a reconstructed model of the feature. This model was constructed from 191 images.

There were 280 images of a tunnel segment, 162 of which were used by PMVS. The result is shown in Fig. 6.30. Models were also built of a larger skylight with a ramp entrance (Fig. 6.31), and an arch between two collapsed sections of the cave (Fig. 6.32). Success with the Bundler/CMVS/PMVS pipeline in these early model-building efforts showed that it was a reasonable choice for other feature-modeling experiments in this work, and this conclusion was supported by the investigation in 6.4.1.

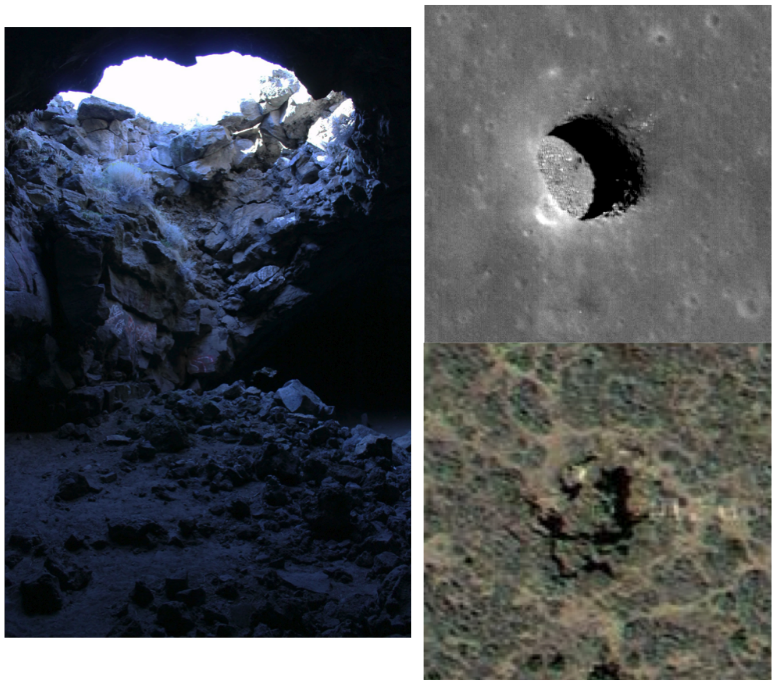


Figure 6.28: Left: Image of steep-walled skylight at Pluto's Cave. Right Top: Marius Hills Hole (Image credit LROC). Right Bottom: Satellite view of steep-walled skylight at Pluto's Cave (Image credit: Google Maps).

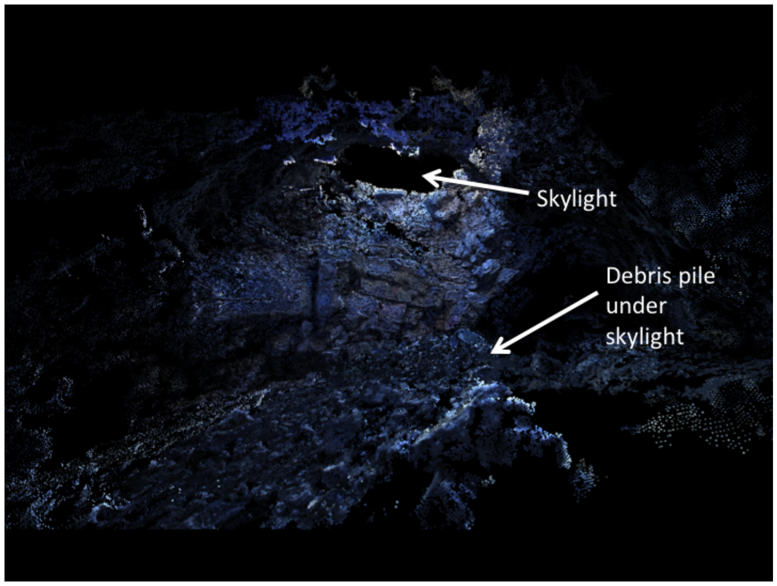


Figure 6.29: Model built of steep-walled skylight in Pluto's cave

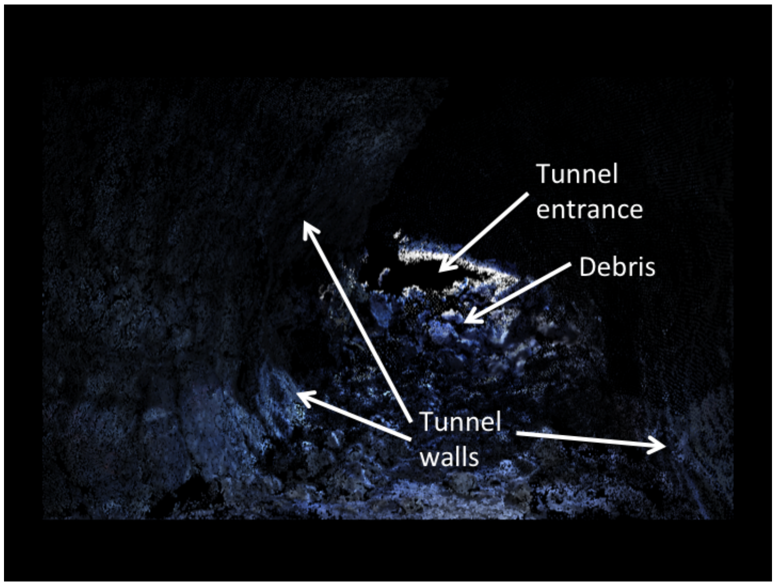


Figure 6.30: Model built of lava tube tunnel in Pluto's Cave

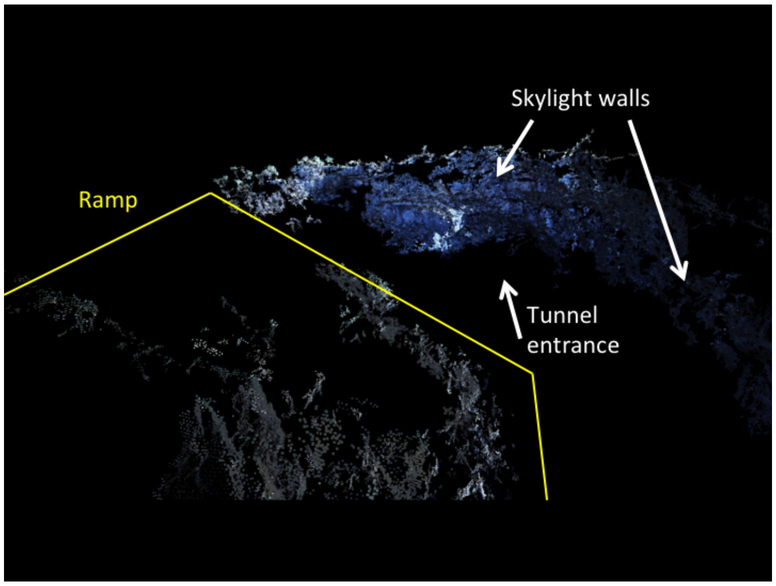


Figure 6.31: Model built of lava tube skylight in Pluto's Cave

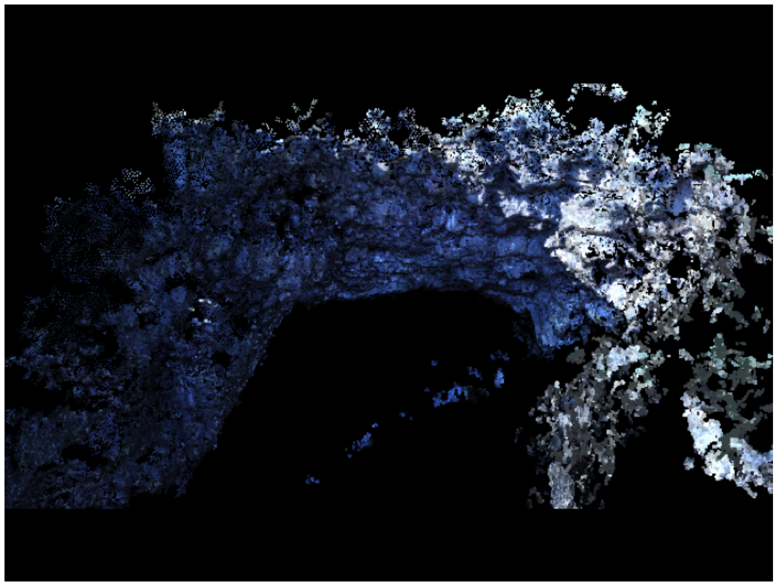


Figure 6.32: Model built of an arch in Pluto's Cave

6.4.2.3 Grand Canyon

A rock formation near Maricopa Point in the Grand Canyon was modeled from images (see Section B.5). At approximately 2km, the Grand Canyon dataset had the largest viewing distance of any of the constructed models. A set of 107 images was collected using a hand-held, wide-field-of-view cell phone camera from multiple viewpoints over about 3km of trail along the canyon rim, spanning a 73-degree range in view angle, as measured in Google Earth [5]. The majority of the images were taken in the span of 1 hour and 15 minutes, which would correspond to less than 18 degrees of sun angle change, although 4 images were taken earlier, making the timespan of the full dataset 2 hours and 45 minutes. The weather was also cloudy with intermittent sun breaks, so some images have more diffuse than directional lighting.

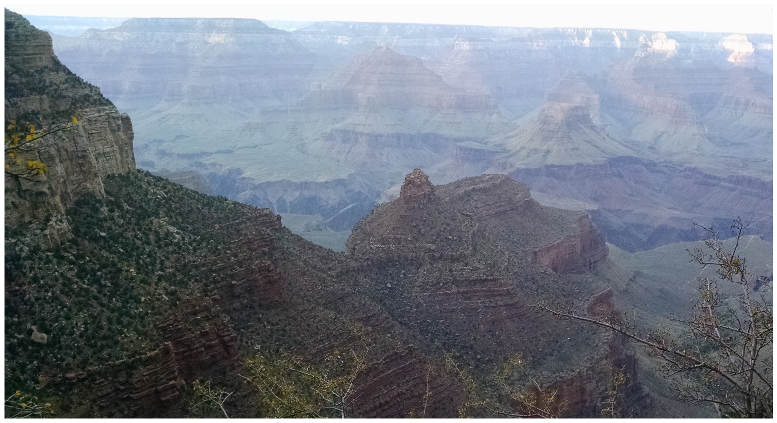


Figure 6.33: Modeled rock formation and surrounding canyon terrain

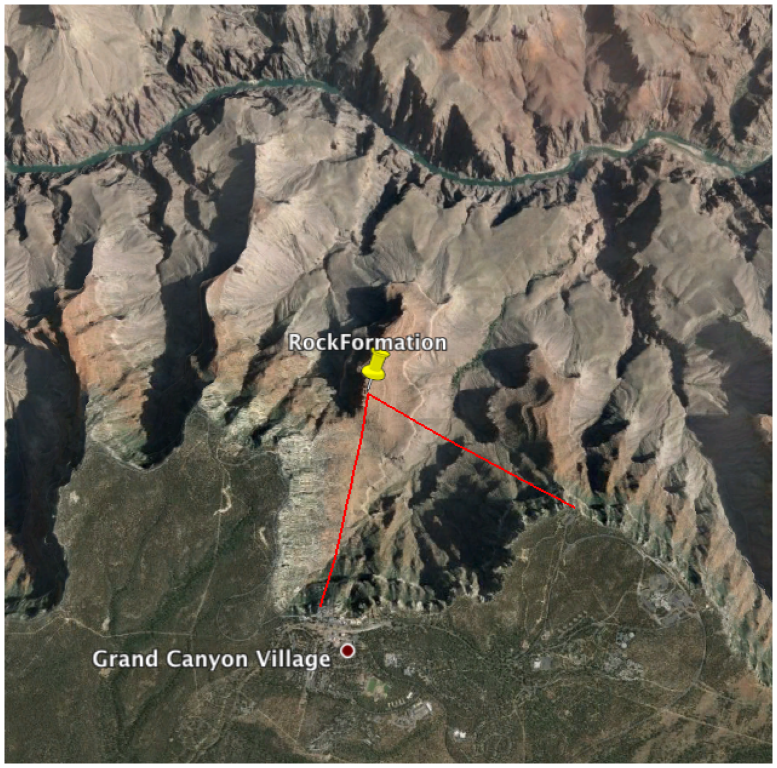


Figure 6.34: Most extreme views for Grand Canyon dataset (Imagery from Google Earth [5])

Bundler [62, 63] was unable to achieve a reasonable structure from motion solution for this image set, most likely because it could not get the camera focal length from the EXIF data attached to the images, so VisualSFM [66, 67] was used for structure from motion. PMVS/CMVS was used for dense reconstruction [64, 65]. Some isolated free-floating point clusters that were clearly erroneous were manually cleaned using MeshLab [88]. Results are shown in Fig. 6.35. Modeling of the rock formation worked well, although reconstruction of background features was noisy.

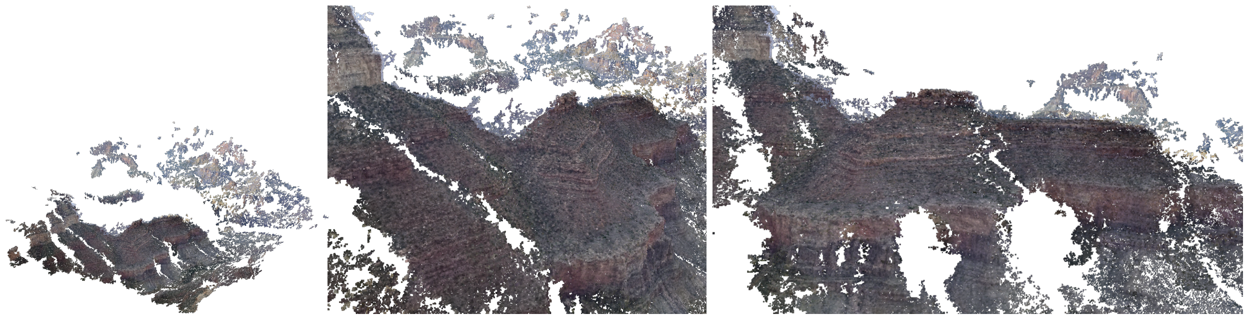


Figure 6.35: Views of a reconstructed point cloud model of a rock formation in the Grand Canyon.

6.4.2.4 Meteor Crater

Meteor Crater was modeled from images (see Section B.4). A set of 240 images was collected using a hand-held, wide-field-of-view cell phone camera from multiple viewpoints in the visitor center area (see Fig. 6.36). The maximum viewing distance in this image set was about 1.2km. Maximum relative view angle difference for the most distant part of the crater to be modeled was about 5.5 degrees, as measured in Google Earth (see Fig. 6.37) [6]. All images were taken in the span of 45 minutes, which would correspond to about an 11-degree change in illumination angle. However, because images were taken near sunset, the illumination angle change resulted in a large change in the area of the crater in shadow. See Fig. 6.38 for an illustration.

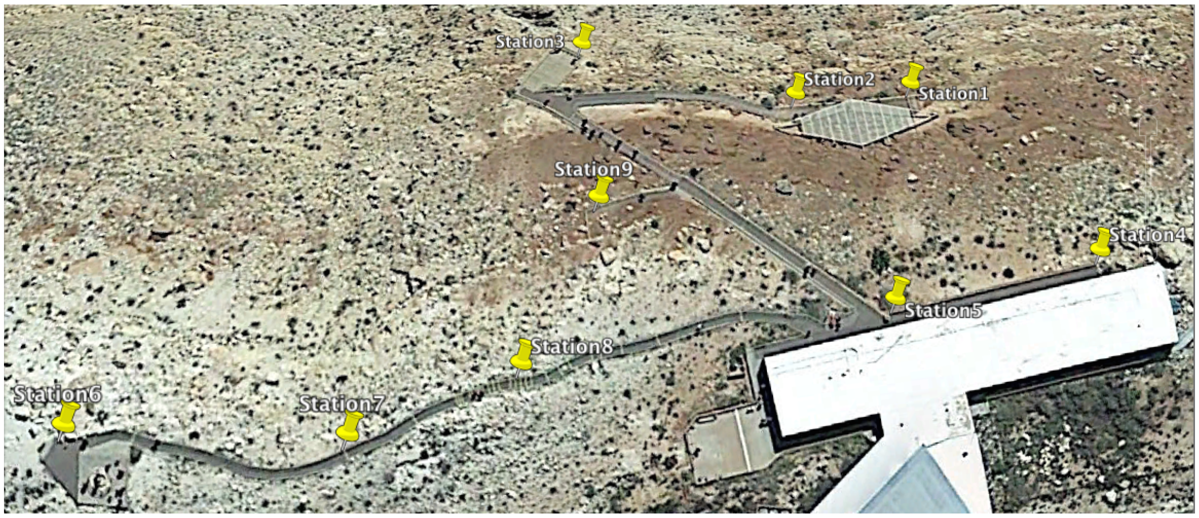


Figure 6.36: Camera positions for Meteor Crater. (Imagery from Google Earth [6])

VisualSFM [66, 67] was used for structure from motion. Dense reconstruction was attempted with PMVS/CMVS, although the dense reconstruction only covered a relatively small portion of the crater, likely due to the small relative view angles to the far side of the crater and the default parameter in PMVS limiting reconstruction to relative views of 10 degrees or higher[64, 65]. Structure from motion results are shown in Fig. 6.39. Note how the reconstruction is fairly good for the continuously lit portion of the crater, but points in the shadowed portion of the crater are more sparse. The shadowed region corresponds to the right part of Fig. 6.39b and the top right portion of Fig. 6.39c. Modeling of this feature would have been more successful if view trajectory planning had been used to determine the time of day when images of various faces of the crater were imaged.

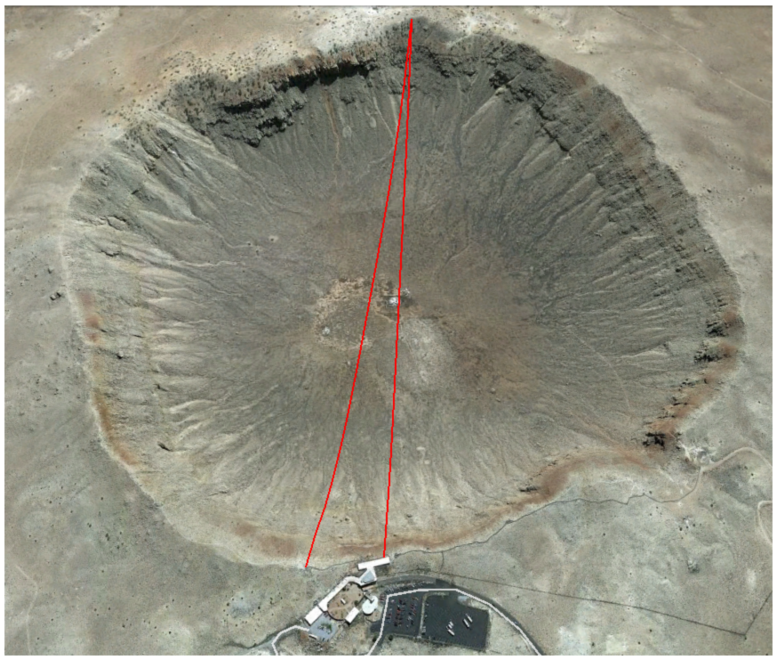


Figure 6.37: Illustration of relative view angle for the most distant camera positions in the Meteor Crater dataset. Views from Station 6 and Station 4 from Fig. 6.36 are illustrated. (Imagery from Google Earth [6])

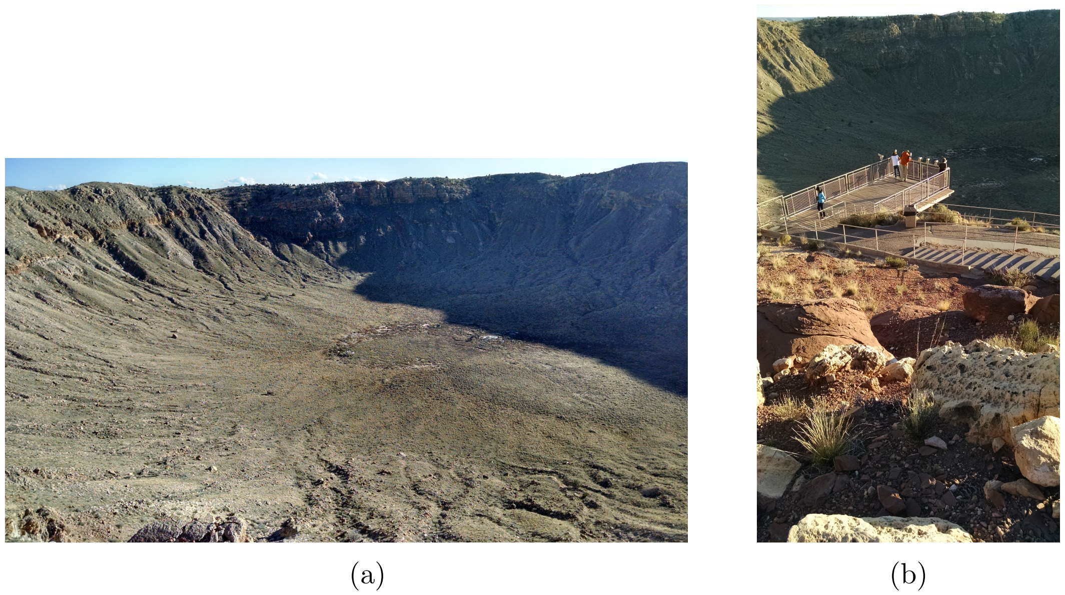
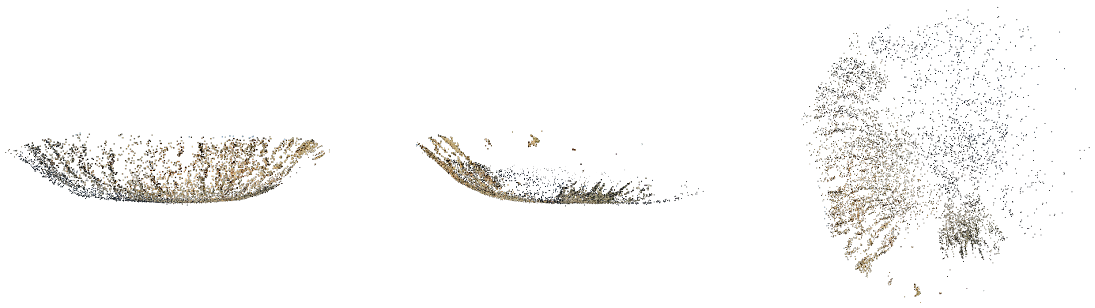


Figure 6.38: Images from the Meteor Crater dataset (a) from the first station, showing a significant portion of the crater floor un-shadowed, and (b) from the last station, showing significantly more shadowing. Note the bright spot near the center of the crater that is just to the left of the shadow in (a) and on the far right of the image in (b).



(a) (b) (c)

Figure 6.39: Sparse reconstruction of meteor crater: (a) side view, (b) view from behind the camera, (c) top view

Chapter 7

Demonstration

This chapter presents two demonstrations of the concepts presented in the thesis. Section 7.1 demonstrates the planned-view-trajectory model building process from end to end on field data for a planetary analog pit. Section 7.2 is the counterpart for modeling a rock with lighting simulating Mars. The rock modeling uses a subset of the controlled lighting laboratory data from Appendix A to simulate illumination directions over the course of a Mars day and views of a rock that are possible from a rover with an arm-mounted camera. Images are then selected from this subset, given limits on view angle, illumination angle, relative view angle, and relative illumination angle set as described in Chapter 4, to model each rock face.

7.1 Field Experiment at King's Bowl

An end-to-end demonstration of the planned-view-trajectory model building process, from inputs and coarse model construction to final detailed model building, was conducted at the King's Bowl planetary analog pit (see Section B.1).

To build the coarse model, the pit was viewed from above using Google Earth, and the outline of the pit was traced and saved as latitude and longitude values in a KML file [3]. The outline was then projected down to the estimated depth of the pit to create a coarse prior model. Fig. 7.1 shows the satellite view and the generated coarse model. Fig. 7.2 shows discretization of the coarse model into patches.

Camera positions were generated at roughly 8 m increments along both the east and west side of the pit (see Fig. 7.3). Because a digital elevation map was not part of the prior model, paths between adjacent positions were manually estimated (avoiding cracks and holes visible in overhead imagery) and measured using Google Earth [3]. Paths are assumed to be bi-directional. Dijkstra's algorithm is run on a graph with position nodes and path-length edges to compute the distance from each position to each other position [72]. To determine if faces are visible in a given view from a rover position, ray tracing was done from the camera position [78]. To distinguish different illumination conditions, time was discretized into two-hour increments.

For the planning part of the planned-view-trajectory model building process, two algo-

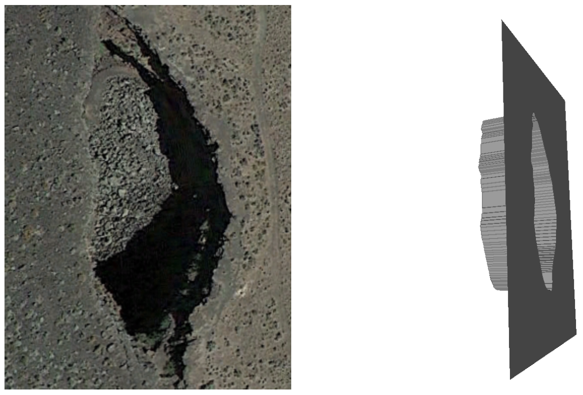

(a) Satellite view (b) Prior model

Figure 7.1: Prior model example for projection of outline, using King's Bowl pit. (Left image from [4], right image created by the authors)

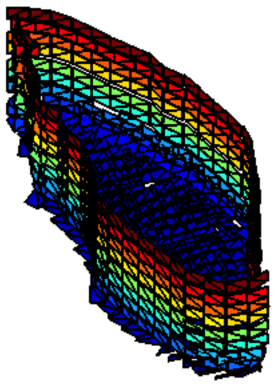


Figure 7.2: A tiling of the coarse model of King's Bowl pit into surface faces

rithms for solving the OPTWIND are used in this experiment. One takes a greedy approach and takes any views that meet absolute and relative view and lighting angle constraints. The other is the timewindow algorithm, a partial implementation of Algorithm 1 from Section 6.3.1. Before this algorithm starts, it is assumed that time has been coarsely divided, based on significant changes in availability or value of positions, similar to the start time, end time, and time divisions. The algorithm first computes a value for each position at each time, as the sum of potential values for each target type available from that position, weighted by one over the total length of time during which that target type is available from any position. Positions within each coarse time division are then ordered by this value. Starting with the top position for each time division, the algorithm tries to find a feasible trajectory that includes one position per time division and meets distance constraints, proceeding to lower-valued positions for a given time if necessary. From the feasible trajectory, it tries to add positions, ordered by the difference in the set of targets available (the larger the difference, the better), within each coarse time division and remaining within the distance constraints. It does a local search to reduce the trajectory distance. It then determines which

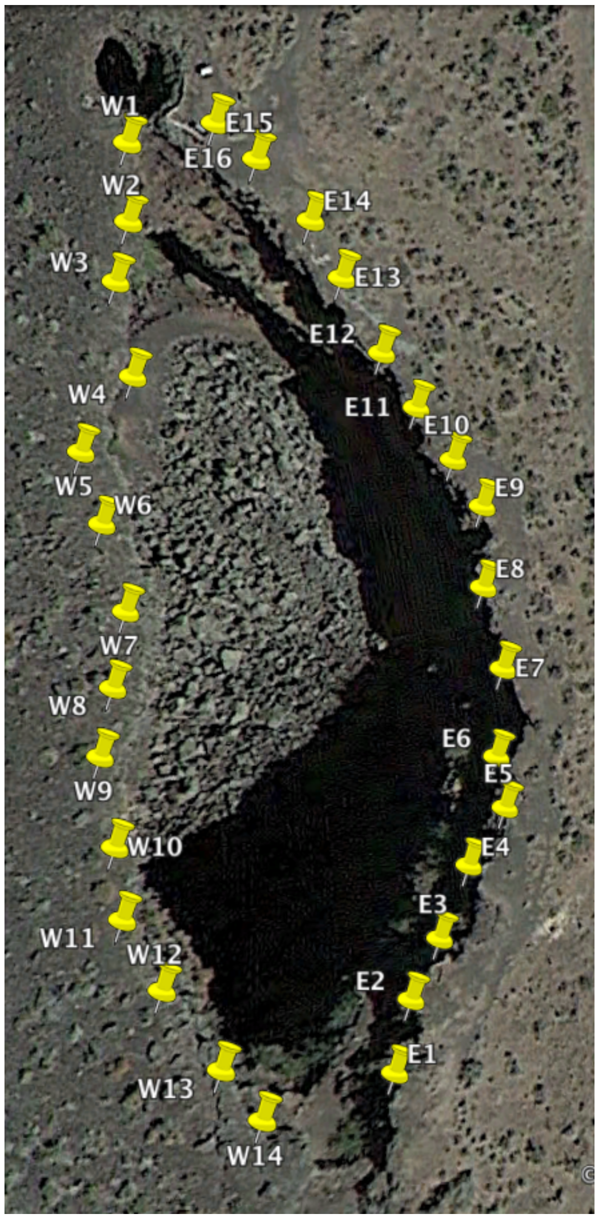


Figure 7.3: Overhead view of Kingsbowl pit. Potential positions are marked by yellow push-pins. (View from Google Earth [3]. Base image from [4])

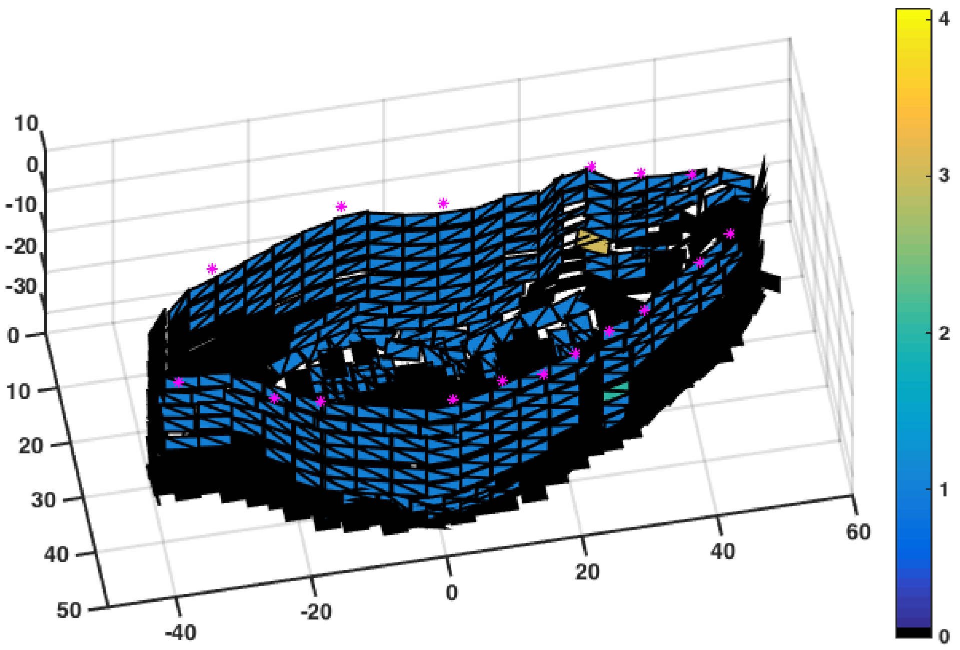


Figure 7.4: Faces viewed under the greedy plan. Faces are colored by number of views. Magenta asterisks indicate the rover positions in this plan.

targets do not have the desired number of views, and tries to add positions, ordered by the number of target types they meet constraints for, and thus improve value. The insert by set difference, reduce trajectory distance, and insert to satisfy constraints steps are repeated until the change in paths is lower than some specified threshold, or a specified maximum number of iterations is reached.

Faces viewed in the view trajectory plans for both the greedy and the timewindow method are shown in Fig. 7.4 and Fig. 7.5. Note that the greedy algorithm appears to get better 2D coverage than the timewindow algorithm, but it has barely any 3D coverage. The timewindow algorithm has a region of good 3D coverage on the west wall (far wall in Fig. 7.5), and limited 3D coverage on the northeast wall. Neither plan has good coverage of the pit floor.

Execution of the view trajectory was done by selecting images from the King's Bowl dataset (see Section B.1). These images were taken from tripod-mounted Canon EOS Rebel T3 DSLR cameras with 50mm fixed focal length lenses. Images were taken over two days and binned into 2 hour time-of-day increments, with some positions being repeated at multiple times. In total, 613 images were available for consideration by the planners. A set of images from a planned view trajectory was then used for model reconstruction. Bundler and CMVS/PMVS were used for reconstruction [62, 63, 64, 65].

Fig. 7.6 shows the reconstruction from a set of 60 images randomly selected from among the 613 total images without regard to illumination or view angles. The reconstructed area

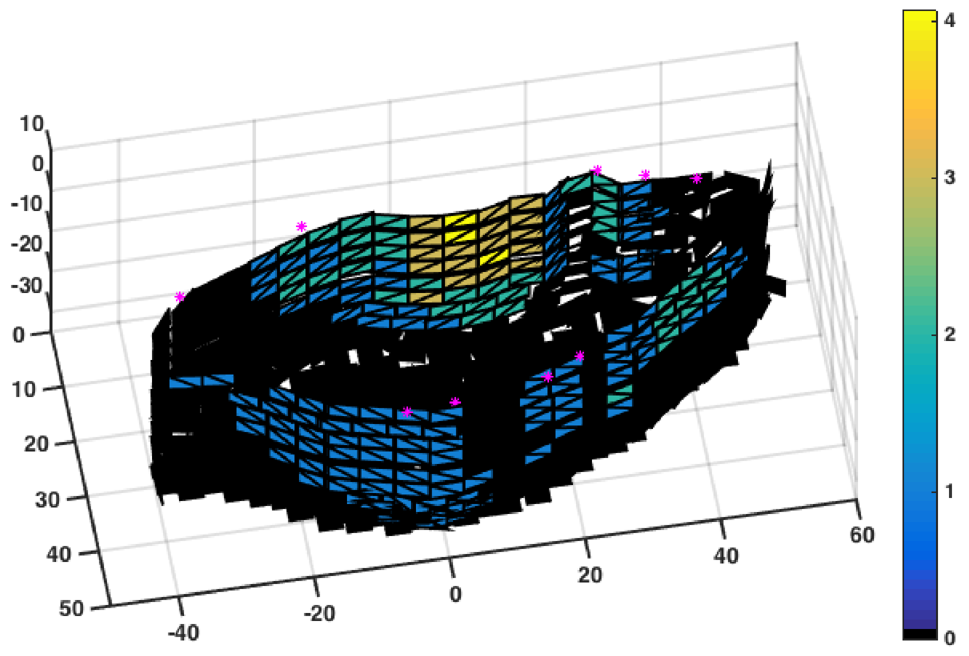
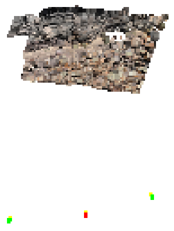


Figure 7.5: Faces viewed under the timewindow plan. Faces are colored by number of views. Magenta asterisks indicate the rover positions in this plan.

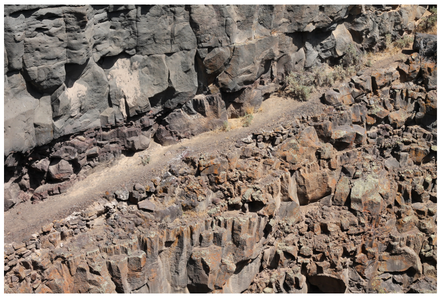
of the pit is in the middle of the east wall and incorporates images from 3 positions.

Fig. 7.7 shows the reconstruction for the greedy algorithm's view trajectory plan, which included 63 images. The reconstructed area lies on the east wall, but is very small, including only 4 images from 2 positions. It likely corresponds to the green face on the near wall in Fig. 7.4.

Fig. 7.8 shows the reconstruction for the timewindow algorithm's view trajectory plan, which included 56 images. The reconstructed area lies on the west wall, likely corresponding to the clump of yellow, orange, and green faces on the far wall in Fig. 7.5. The reconstruction includes images from 4 positions. The timewindow algorithm successfully reconstructs a larger area than the greedy algorithm and the random image set, although none comes close to modeling the entire pit.



(a) A 3d reconstruction from a random set of 60 images of King's Bowl, using the greedy planning method. The reconstruction covers a relatively small patch of terrain on the east wall of the pit. Three images from 3 positions on the west rim of the pit are used in the reconstruction. Red, green, and yellow dots represent camera positions.

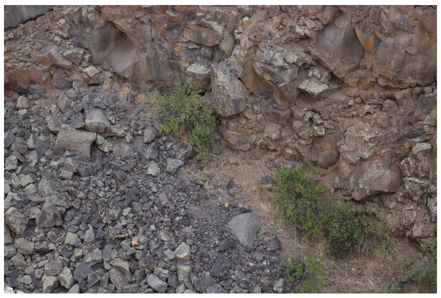


(b) One of the images used to reconstruct the model.

Figure 7.6

All detailed models in this experiment are clearly partial reconstructions with limited feature coverage, and this is expected given the planned view trajectories. The limited coverage is attributable to limitations in data collection. The two day experiment intended

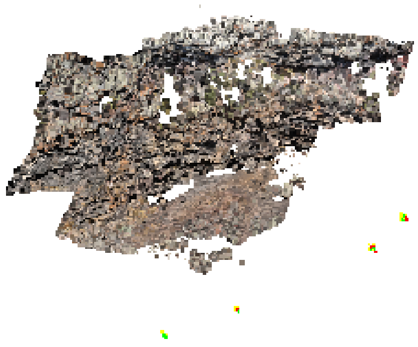
(a) A 3d reconstruction from a planned set of 63 images of King's Bowl, using the greedy planning method. The resulting model covers a very small patch of terrain on the east wall of the pit and only incorporates 4 of the 63 images. The 4 images were taken from the west rim of the pit. Red, green, and yellow dots represent camera positions.



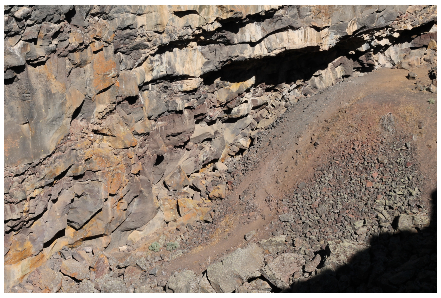
(b) One of the images used to reconstruct the model.

Figure 7.7

to acquire a comprehensive set of imagery from an array of robot locations over a series of illumination angles from which multiple view trajectories could be simulated. Experimental productivity was sub-optimal. Hence, images were not acquired from all viewpoints and under all illumination angles. Limits on relative illumination angle and view angle were set more loosely than ideal to account for the limited dataset. Camera pointing accuracy in the field was also limited, which likely resulted in some faces that the planner expected to be covered by a given image being partially covered or not covered by that image, especially near the edges of the field of view. The prior model was also very coarse. In cases where the coarse model was far from the actual shape of the pit, view and illumination angles could be significantly different than predicted. The model reconstruction pipeline of Bundler and CMVS/PMVS also has limited success when models are not contiguous, which is the case for the planned view trajectories in this demonstration. The resulting models of Fig. 7.7 and Fig. 7.8 reflected the shortcomings of the field campaign. Nonetheless, this experiment



(a) A 3d reconstruction from a planned set of 56 images of King's Bowl, using the timewindow planning method. The reconstruction includes an area of the west wall. 26 of the images from 4 positions on the east rim were used for the reconstruction. Red, green, and yellow dots represent camera positions.



(b) One of the images used to reconstruct the model.

Figure 7.8

succeeds in demonstrating the planned-view-trajectory model building process end-to-end on a real-world planetary analog terrain feature.

7.2 Modeling a Rock in a Simulated Mars Day

To test the success of planned-view-trajectory model building when view and illumination angle limits are set as described in Chapter 4, a simulated Mars rock modeling scenario was set up using images of a turntable-mounted rock from the "lava" dataset, as described in Appendix A. In the scenario, a Mars rover has one day to model a rock with an arm-mounted camera. It can drive around the rock (turntable rotation), or it can raise or lower its arm (tripod angle change) to achieve different view angles. Different illumination angles will be achieved at different times of day.

A subset of the images in the “lava” dataset was selected corresponding to the illumination directions that would occur over a Mars day. For each turntable angle in the dataset, all of the light positions were transformed into a rock-fixed frame, and vectors to the center of the turntable were computed, creating a hemisphere of potential illumination vectors. Illumination vectors were also computed for a surface location on Mars over a single Mars day at approximately 4-minute increments, using SPICE tools and data from the Mars Science Laboratory mission [77]. These Mars illumination vectors were rotated about the local z axis by some angle, θ. Angles between Mars illumination vectors and laboratory illumination vectors were computed, and the Mars illumination vector closest to each laboratory illumination vector was found. The set of laboratory illumination vectors was then reduced to only those for which the nearest Mars illumination vector was less than a threshold angle away (2.9 degrees was used as the threshold angle, corresponding to approximately 12 minutes of sun angle change). A visualization of illumination vector directions can be seen in Fig. 7.9. Only images corresponding to the reduced illumination vector set were included as part of the experiment: the images using the same turntable angle and light number for which a vector was computed, and any of nine tripod angles for each turntable-light combination. Since tripod angle changes correspond to raising and lowering an arm and not moving the robot base, it is assumed that image capture is fast, and several images can be taken during the same 4-minute increment.

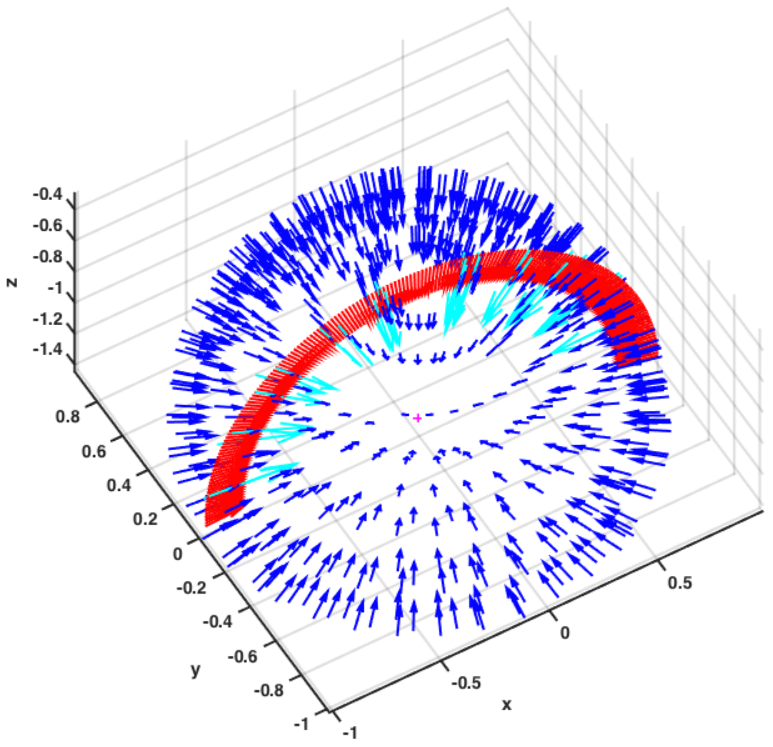


Figure 7.9: Illumination vectors for the Mars rock scenario. Blue vectors indicate potential laboratory illumination directions. Red vectors indicate Mars illumination vectors. Cyan vectors indicate the selected subset of laboratory illumination vectors. The magenta + indicates the center of the turntable on which the rock is mounted.

Since the angle θ used in rotation to align the Mars frame with the rock-fixed frame is arbitrary, angles from 0-360 were tried, and the number of views for each face that met both view and illumination angle thresholds were computed. The angle θ was chosen such that each face had at least 9 potential views that met both thresholds (for the chosen θ, the number of views meeting view and illumination angle thresholds were 51, 14, 9, 62, and 79 for faces 1-5, respectively). The selected θ value was also such that relative view and illumination thresholds could be met by at least one pair of images for each face. View and illumination angle parameters used in this experiment are listed in Table 7.1.

Table 7.1: Parameter Limits

Parameter	Min	Max
View Angle	0	43
View Angle Relative	5	15
Illumination Angle	36	78
Illumination Angle Relative	0	15

The LIDAR model of the lava rock described in Section A.2 was used as a coarse model, and the faces described in Section A.3 for the lava rock were used for a face discretization. Two view trajectories selected to obey all parameter limits were planned, and 10 view trajectories were generated randomly without regard to view and illumination angle limits. Each view trajectory had 10 views, two for each face. It is worth noting that due to the nature of the lava dataset, many images view more than one face, especially for faces 2-5. Only the faces specifically selected to view a face were used for reconstructing that face. Reconstructions were done with Bundler for structure-from-motion (SFM) and PMVS for dense reconstruction [62, 63, 64].

Fig. 7.10 shows results for reconstruction of each face for the planned view trajectories. Note that the majority of each face is successfully reconstructed in both trials. For faces 2-5, parts of other faces are also reconstructed in addition to the target face.

Table 7.2 shows results for the randomly generated view trajectories. For most cases, Bundler failed completely to produce an SFM result. In one case, Bundler produced a result, but it was not valid, and thus PMVS did not produce a result. In one case, PMVS produced a result that was not recognizable. In four cases, PMVS produced a recognizable result, but it was missing all or most of the target face. In four cases, the target face was successfully reconstructed.

With planned-view-trajectory model building, 10 of 10 image pairs are successfully used to build a model of the target face. With randomly selected images, only 4 of 50 image pairs work for modeling the target face. These results clearly show the value of planned-view-trajectory model building.

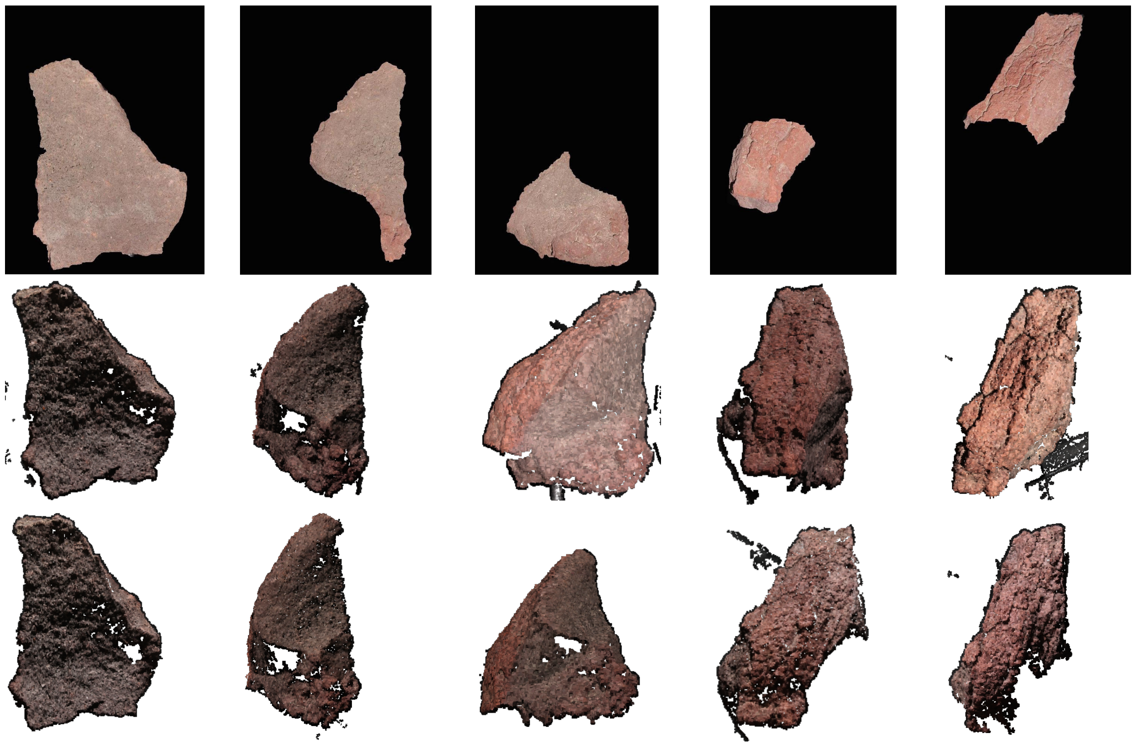


Figure 7.10: Faces 1-5 of lava rock masked in images (top), and reconstructed in trial 1 (middle) and 2 (bottom) of the planned view trajectories.

Table 7.2: Per-Face Results for Random View Trajectories

Trial	No SFM Result	No Dense Reconstruction	Unrecognizable Dense Reconstruction	Misses Face	Successful Reconstruction
1	3	0	0	2	0
2	3	1	0	0	1
3	5	0	0	0	0
4	4	0	0	0	1
5	3	0	0	1	1
6	4	0	1	0	0
7	5	0	0	0	0
8	4	0	0	1	0
9	4	0	0	0	1
10	5	0	0	0	0
Total	**40**	**1**	**1**	**4**	**4**

Chapter 8

Conclusions

8.1 Summary

This research establishes view trajectory as a sequence of robot positions, robot view targets, and viewing times. It asserts that using planned, illumination-aware view trajectories facilitates building good models of substantially 3D planetary terrain features under transient, directional illumination with monocular imagery acquired by a surface robot. Metrics that contribute to good models are distinguished as coverage, resolution, and reconstruction accuracy.

The research develops planned-view-trajectory model building, which takes prior terrain data for a feature of interest, time information, illumination trajectory information, and rover restrictions and converts these into a planning problem that enables limits on viewing distance, view angle, illumination angle, relative view angle, and relative illumination angle to be applied when planning a view trajectory. The view trajectory planning problem is cast as a new vehicle routing problem, the Orienteering Problem with Time Windows and Inter-Node Dependencies (OPTWIND). A view trajectory planning algorithm to solve the OPTWIND is developed and presented.

Through an experimental campaign that captured thousands of rock images under directional lighting, the research investigated the impact of viewing distance, view angle, illumination angle, relative view angle, and relative illumination angle on these model quality metrics. Methods for setting limits on these parameters that minimize negative impacts on model quality were developed.

This work has modeled several substantially 3D terrain features on Earth that are analogs for planetary features. End-to-end demonstration of the planned-view-trajectory model building process was conducted. A demonstration of applying view and illumination angle limits, determined based on experimental results, to planned-view-trajectory model building was also presented.

8.2 Contributions

Contributions of this research include methodology for robotically modeling planetary features, field and laboratory experimentation to show the method in action, and studies to characterize what constitutes a good model. A significant contribution of the research is development and implementation of a methodology for planned-view-trajectory model building. The research demonstrates planned-view-trajectory model building on planetary analog data. The work contributes a characterization of performance for keypoint detection in images of planetary-relevant geologic materials taken under directional lighting over a range of view and illumination angles, based on experimental results. A key contribution is experimental determination of how relative view angles interact with relative illumination angles for keypoint matching. The research formulates view trajectory planning as an innovative vehicle routing problem. The research contributes an archival dataset consisting of nearly 19,000 images of planetary-relevant geologic material under controlled, directional lighting. The research contributes a dataset consisting of over 6,000 images of planetary analog terrain features, including a crater and lava tube skylights.

8.3 Conclusions and Future Work

Good models of substantially 3D planetary terrain features under transient, directional illumination with monocular imagery acquired by a surface robot result from planned, illumination-aware view trajectories. Automatic detection of keypoints varies across different view and illumination angles for directional lighting, with a peak number of keypoints detected around 60 degree illumination angle for small view angles. The number of features detected declines for smaller illumination angles, where the angle between the viewing direction and the illumination direction is too small, and for illumination angles that approach 90 degrees. Keypoint matching between images under directional lighting decreases exponentially as either the relative view angle or the relative illumination angle between image pairs increases. These insights set limits on view angle, illumination angle, relative view angle, and relative illumination angle in view trajectory plans. When these limits are applied, images captured during execution of the resulting view trajectory are useable for successful reconstruction of a given target face on a substantially 3D terrain feature. This is statistically superior to random choice of views of that face.

Avenues for future research include further study of view trajectory planning, more in-depth experimentation to explore the effects of terrain feature surface texture, and implementation on real robots. Future work could include development of a more efficient planner for planning view trajectories. A more comprehensive study of the effect of surface texture on the relationships between view and illumination angles and model quality could be conducted. This research used three diverse rock samples. A broader experimental program would further support the observations and improve statistical significance of the conclusions. A polynomial surface was fitted to data for number of features detected over view and illumination angle change, but the polynomial approximation fails at the edges of the fitted data. More physically based models for feature detection results could be explored. Finally, while this research is motivated by robotic operations, cameras were tripod-deployed. Testing on real robots has yet to be demonstrated.

To date, planetary rovers have explored and modeled benign terrain in a myopic manner. By bringing methodology for deliberate, disciplined quality modeling, this research enables robotic exploration and modeling of planetary pits, canyons, and craters that call to our future.

Appendices

Appendix A

Controlled Lighting Experiments

To understand how view and illumination angles affect model quality, laboratory experiments were conducted with images of rocks under controlled directional lighting. A total of nearly 19,000 images were collected of three rock samples and processed to detect features, match features, and reconstruct 3D shape from images. Each rock sample was also scanned with LIDAR to determine coarse 3D shape. Section A.1 describes the laboratory setup and the datasets collected, Section A.2 describes the construction of coarse mesh models from LIDAR, Section A.3 describes how each rock sample was discretized into faces, Section A.4 describes how masks were computed and used to isolate rock samples and individual rock faces in images for data processing, and Section A.5 describes the data processing.

A.1 Laboratory Setup and Image Datasets

Rock samples were placed on a turntable with 24 individually controllable light sources in an otherwise dark scene (see Fig. A.1). Light sources were positioned along two walls and the top of a cube, and a camera was positioned at one edge of this cube[1]. View angle could be adjusted by tilting the camera or by rotating the turntable. The camera height was adjusted for each tilt angle to keep the camera pointed at the rock. Illumination angle could be adjusted by selecting different light sources. Rotation of the turntable also adjusted the illumination angle relative to the local frames of each rock face.

Three different rock samples were used. View trajectory planning discretizes features into planar faces. The three rock samples differ in the degree to which faces appear planar.

The three rocks were acquired from landscaping supply sources. All are believed to be igneous rocks, but exact petrology is not known. For convenience, each rock was named according to common landscaping terminology: "basalt" is columnar basalt, "lava" is lava rock, and "lace" is volcanic lacerock. Fig. A.2 shows example images for each rock.

Basalt is weathered columnar basalt. It has a fine-grained texture, and most of the visual texture in images actually comes from color variation (from light tan to dark brown) on the

[1]The directional lighting part of the laboratory setup was put together by Uland Wong for his thesis work [89].

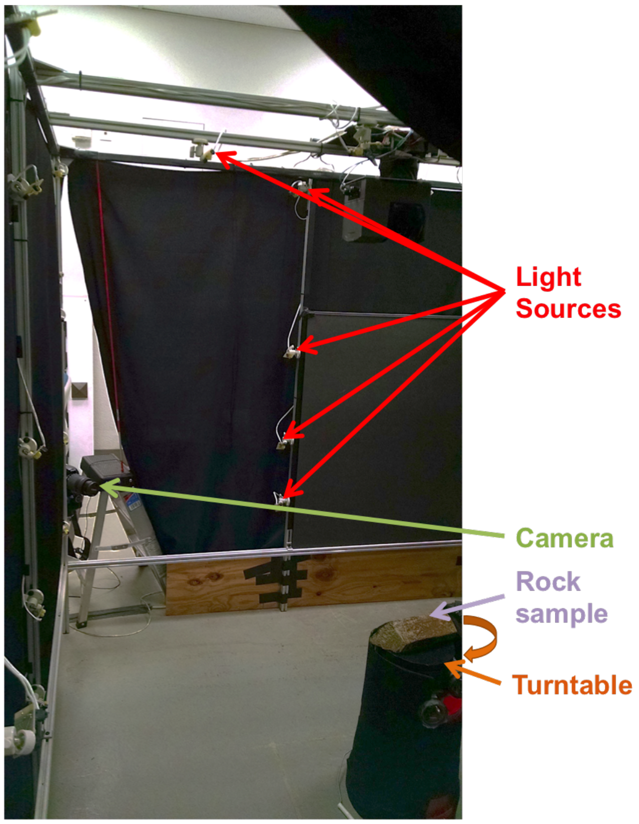


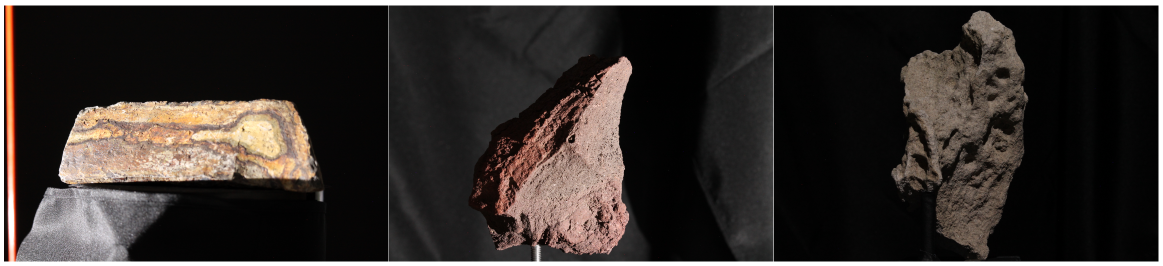
Figure A.1: Experimental Setup

weathered surfaces of the rock. These surfaces are smooth to the touch. Self-shadowing occurs only when faces are illuminated at very oblique angles.

Lava is reddish in color and has a fine vesicular or “frothy” texture. There is some variation in color between different sides of the rock, but most of the visual texture in images comes from shape. Self shadowing occurs on a very fine scale as the illumination moves away from the surface normal, but coarser-scale self-shadowing only happens at very oblique illumination angles.

Lace is gray in color with a very coarse vesicular texture on most faces. The surface is rough to the touch. Most of the visual texture in images comes from shape. There is significant self-shadowing, even at fairly small illumination angles relative to the face normal.

A set of images was collected for each rock, varying the tripod angle (camera height and tilt) at 5 degree increments, the turntable angle at 5 to 10 degree increments, and light source position. Following collection of the main dataset, it was determined that images with smaller relative view angles were needed. Thus, another set of images for additional turntable angles was collected for each rock. These are the “small angles” datasets. Because the rock was not removed from the turntable between the lace and lace small angles sets, these two sets can be used together to get additional relative view angles. Table A.1 gives an overview of the collected datasets. Table A.2 gives detail for the tripod and turntable



(a) “basalt” (b) “lava” (c) “lace”

Figure A.2: Example images for each rock sample

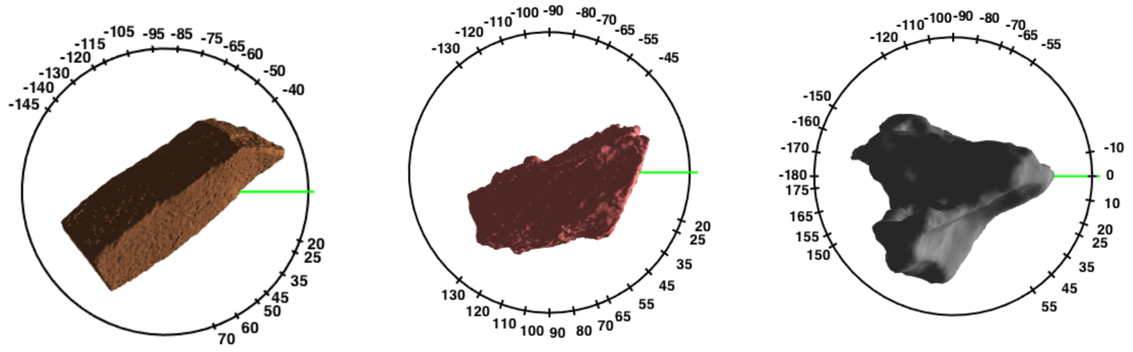


Figure A.3: Visualization of turntable angles for basalt, lava, and lace datasets. The green line indicates the camera direction at turntable angle 0

angles for each dataset. Fig. A.3 shows a visualization of the turntable angles for the basalt, lava, and lace datasets.

Table A.1: Laboratory Dataset Descriptions

Dataset	Tripod Angles	Turntable Angles	Light Positions	Total Images
basalt	8	20	24	3840
basalt small angles	3	24	24	1728
lava	9	23	24	4968
lava small angles	3	24	24	1728
lace	9	24	24	5184
lace small angles	3	20	24	1440

All images were taken with a Canon EOS Rebel T3 camera (two distinct cameras were used, but both were the same model). The lens was a Canon EFS 55-250mm adjustable lens, set to 100mm. All images used a 15 second exposure and “Flash” white balance. The f-number was set to 5 for basalt, 8 for lava and lava small angles, and 16 for lace and lace small angles. The f-number for basalt small angles was 16.

For basalt, (the first rock sample imaged), the camera was mounted on a tripod. A plumb bob was used to position the tripod over the same spot after each height adjustment. For the

Table A.2: Angles for Each Dataset

Dataset	Tripod Angles	Turntable Angles
basalt	0, 5, 10, 15, 20, 25, 30, 35	−145, −140, −130, −120, −115, −105, −95, −85, −75, −65, −60, −50, −40, 20, 25, 35, 45, 50, 60, 70
lava	0, 5, 10 15, 20, 25, 30, 35, 40	−130, −120, −110, −100, −90, −80, −70, −65, −55, −45, 20, 25, 35, 45, 55, 65, 70, 80, 90, 100, 110, 120, 130
lace	0, 5, 10 15, 20, 25, 30, 35, 40	−180, −170, −160, −150, −120, −110, −100, −90, −80, −70, −65, −55, −10, 0, 10, 20, 25, 35, 45, 55, 150, 155, 165, 175
basalt small angles	0, 5, 40	−95, −94.5, −94, −93.5, −93, −92, −91, −90, −85, −80, −75, −70, 25, 30, 35, 40, 45, 46, 47, 48, 48.5, 49, 49.5, 50
lava small angles	0, 5, 40	−135, −130, −125, −120, −115, −114, −113, −112, −111.5, −111, −110.5, −110, 40, 45, 50, 55, 60, 61, 62, 63, 63.5, 64, 64.5, 65
lace small angles	0, 5, 40	25.5, 26, 26.5, 27, 27.5, 28, 28.5, 29, 29.5, 30, 170, 170.5, 171, 171.5, 172, 172.5, 173, 173.5, 174, 174.5

other rocks, (as well as the small angles set for basalt), the tripod head was mounted to a bar attached to the same frame as the lights, to facilitate more repeatable camera positioning.

The procedure followed when collecting data was to cycle through all light positions for each rock-camera relative pose (tripod angle and turntable angle), cycle through all turntable angles for each tripod angle, and cycle through all tripod angles as an outer loop. A few deviations from this procedure occurred. Namely, in the lava set, all turntable angle −120 images were collected first, and then the rest of the data were collected. For basalt, some images had to be re-taken due to errors after the tripod angle had already changed. For most images, the camera was auto-focused on the rock before taking images at each rock-camera relative pose. For higher tripod angles for lace, the aperture and distance from the rock resulted in a greater depth of field, so the camera was only auto-focused once per tripod angle.

Fig. A.4 illustrates the 24 different light directions. Twenty lights were mounted on five posts covering two sides of a cube, and four lights were mounted on the top of the cube. The camera was mounted on the post closest to the bottom-center of the plot. Fig. A.5 shows an example set of images for all these light positions at a single rock-camera relative pose. The four images in the top row are from the lights on the top of the cube. Each column represents one post.

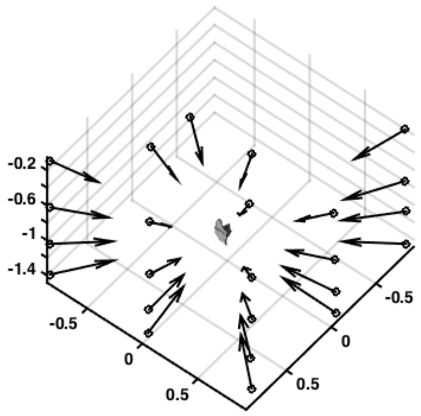


Figure A.4: Visualization of light positions and illumination angles for a single turntable angle. Axes are labeled in meters. Four lights each are mounted on five vertical posts along two sides of a cube and four are mounted on the top of the cube. Camera positions are along the post closest to the bottom corner of this plot.

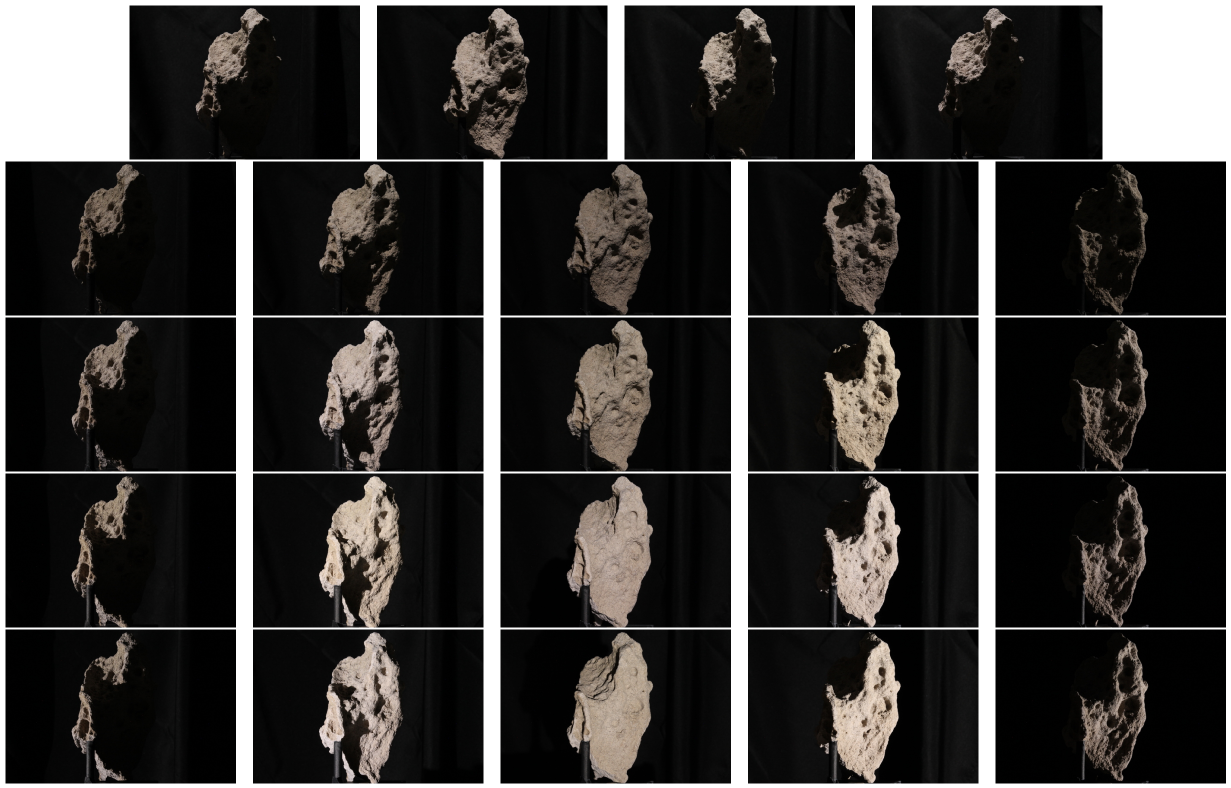


Figure A.5: Example images for lace rock with each light source

A.2 Coarse Rock Models

Geometry of each rock was captured by LIDAR scanning for use in identifying rock faces, for comparison with the models created from images, and to serve as coarse prior models in planned-view-trajectory modeling. Each rock was scanned several times from different directions using a Faro Focus 3D 120 LIDAR scanner. A down-sampled version of each scan was used for alignment between scans. Each scan was manually trimmed to include just the rock and any alignment fiducial markers. Coarse alignment between scans was done manually, either by visually estimating rotations or by identifying common features in two scans and transforming to align these features. Fine alignment was done using the Iterative Closest Point method [84, 85, 86]. Data from the full-resolution scans were manually trimmed to include just the rock, and then these were transformed according to the transformations determined during alignment of the coarse scans. Points from all high-resolution scans were merged into a single point cloud. Normals for the points were then estimated using MeshLab [88]. Color for the coarse model came from the Faro scanner's built-in camera. Because the scanner auto-adjusts the camera white balance, in some cases scan color needed to be histogram adjusted before scans were merged into a single point cloud. Poisson Reconstruction was used to build mesh models from point clouds [75, 76]. These mesh models can be seen in Fig. A.6. For the basalt, lava, lace, lace small angles, and basalt small angles datasets, a LIDAR scan was taken of the rock in the lab to align it relative to the laboratory frame. The transformation between the scan and lab reference frames was computed by identifying light locations in the scan and aligning them (best fit using singular value decomposition) with known light locations. An example LIDAR scan of basalt in the lab can be seen in Fig. A.7.

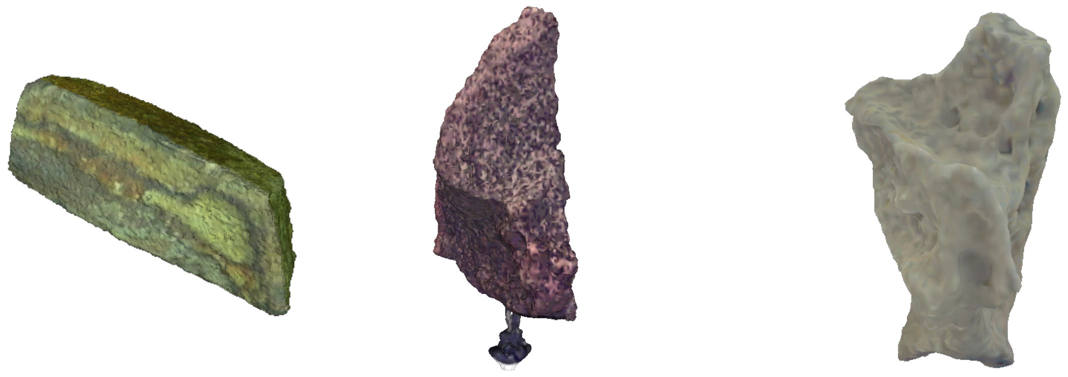


Figure A.6: Coarse mesh models

One of the functions of the LIDAR mesh models is to serve as ground truth for 3D reconstructions, at least for coarse shape. Due to limitations in the sensing used to create the LIDAR models, however, they cannot be used to compute errors on order of the surface resolution of the captured images. The FARO Focus 3D 120 has a systematic range error of $\pm$2mm, one sigma, and a range noise of $\pm$0.6mm for surfaces with 90% reflectance and $\pm$1.2mm for surfaces with 10% reflectance, both one sigma [90]. In comparison, given the

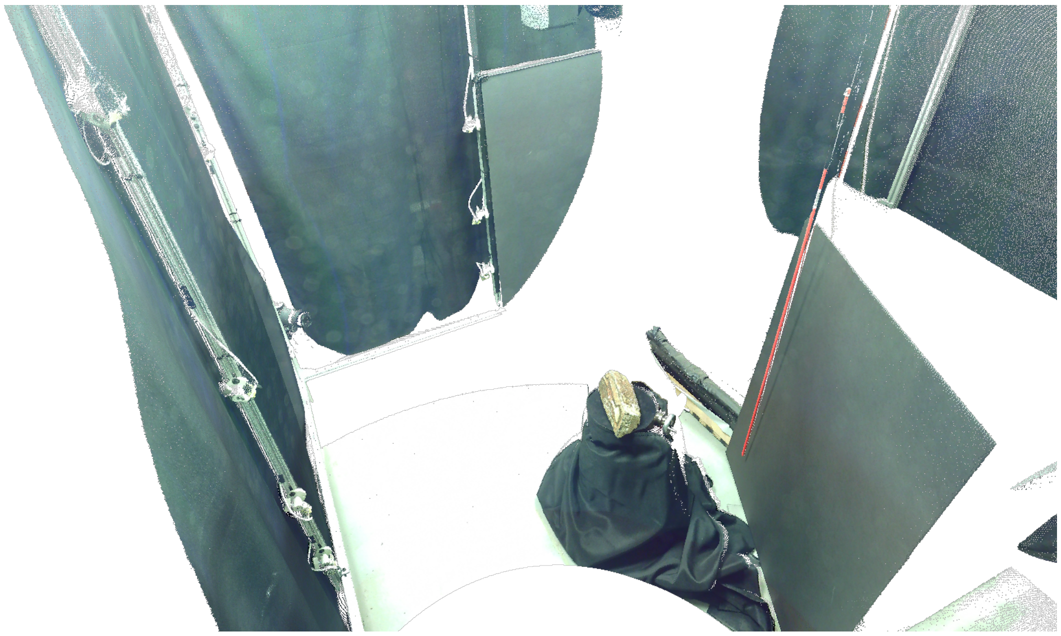


Figure A.7: Colorized LIDAR scan of the basalt rock sample in the lab. The basalt rock is visible on a stand in the center of the image. LED lights are visible mounted to the post on the left side of the image.

laboratory configuration used in these experiments with a viewing distance of 1.485m, the surface resolution is about 0.08mm. If a 3x3 square of pixels is used to reconstruct each 3D point, the 3D resolution of reconstructed models would be 0.24mm, which is order of magnitude smaller than the systematic range error.

Alignment between scans is also a source of error; that is harder to evaluate and is not explicitly evaluated here. Another potential source of error in the LIDAR model is the meshing step. Meshing effectively acts as a low-pass filter on spatial variation, so to the extent that high-frequency spatial variation is due to noise, meshing will reduce error, but to the extent that high-frequency spatial variation is due to rock shape, it will increase error.

Fig. A.8 shows a histogram of point-to-mesh distances, computed using CloudCompare [91] between the point cloud constructed from LIDAR scans and the mesh built from this point cloud for the basalt rock. For these meshes, the mesh normals point outward, so negative point-to-mesh distances indicate that the point is inside the mesh. The mean value for this point-to-mesh distance is -0.45mm, the standard deviation is 1.0mm, and the minimum and maximum values are -7.2mm and 5.3mm respectively. Fig. A.9 shows the histogram of point-to-mesh distances for the lava rock. The mean value for this point-to-mesh distance is -0.16mm, the standard deviation is 0.74mm, and the minimum and maximum values are -4.6mm and 4.5mm respectively. Fig. A.10 shows the histogram of point-to-mesh distances for lace rock. The mean value is -0.11mm, the standard deviation is 0.88mm, and the minimum and maximum values are -5.2mm and 5.9mm respectively. The minimums, maximums, and standard deviations of point-to-mesh distance are within what is reasonable to expect just due to the range error and range noise of the FARO sensor (using 3 standard deviations for both range error and range noise would give a value of

±9.6mm). Thus, evaluations of error between the point clouds and mesh models do not reveal any larger errors introduced in the meshing step than what errors were present in the point cloud due to LIDAR noise and scan alignment errors. However, this amount of error means that the LIDAR models will definitely not be useful in analyzing errors on order of the 3D resolution of the reconstructed model, (expected to be around 0.24mm).

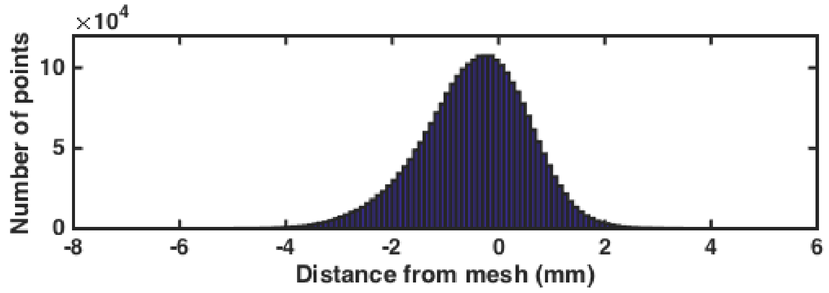


Figure A.8: Histogram of distances between point cloud constructed from LIDAR scans and mesh built from the point cloud for basalt rock

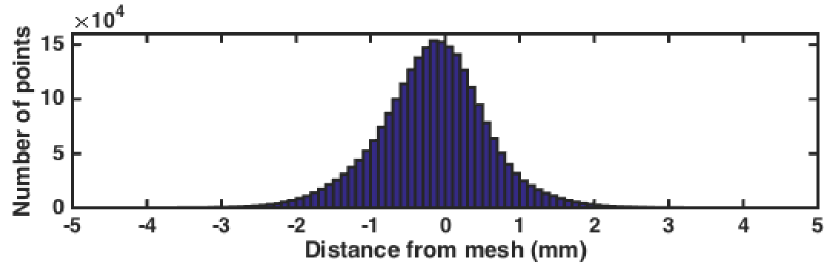


Figure A.9: Histogram of distances between point cloud constructed from LIDAR scans and mesh built from the point cloud for lava rock

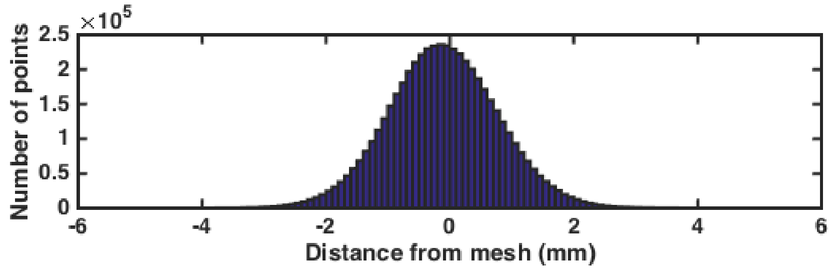


Figure A.10: Histogram of distances between point cloud constructed from LIDAR scans and mesh built from the point cloud for lace rock

A.3 Rock Faces

The planned-view-trajectory modeling process partitions a feature to be modeled into some number of planar faces, so the rock samples in this laboratory experimentation were also divided into planar faces. Division into faces is necessary because view and illumination angles are expressed relative to each face's local frame in view trajectory planning. The faces have varying degrees of fine and coarse texture. Faces were selected manually as subsets of the coarse model. Fig. A.11 shows the faces for basalt, Fig. A.12 shows the faces for lava, and Fig. A.13 shows the faces for lace.

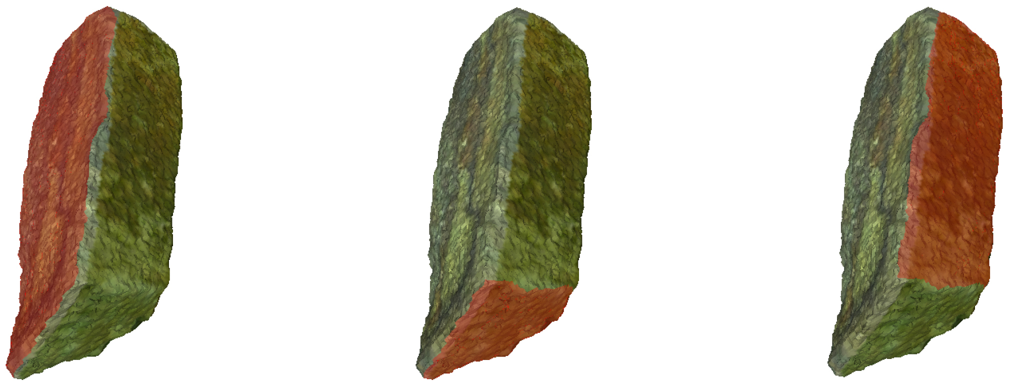


Figure A.11: Faces for basalt rock

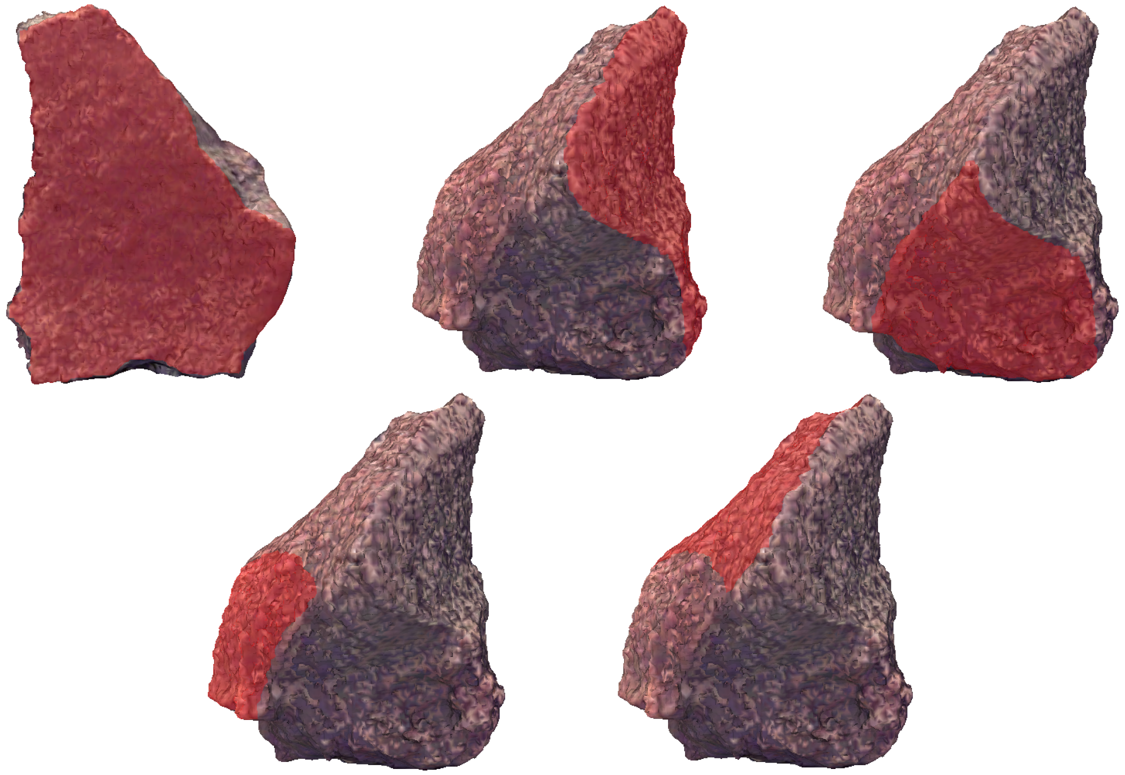


Figure A.12: Faces for lava rock

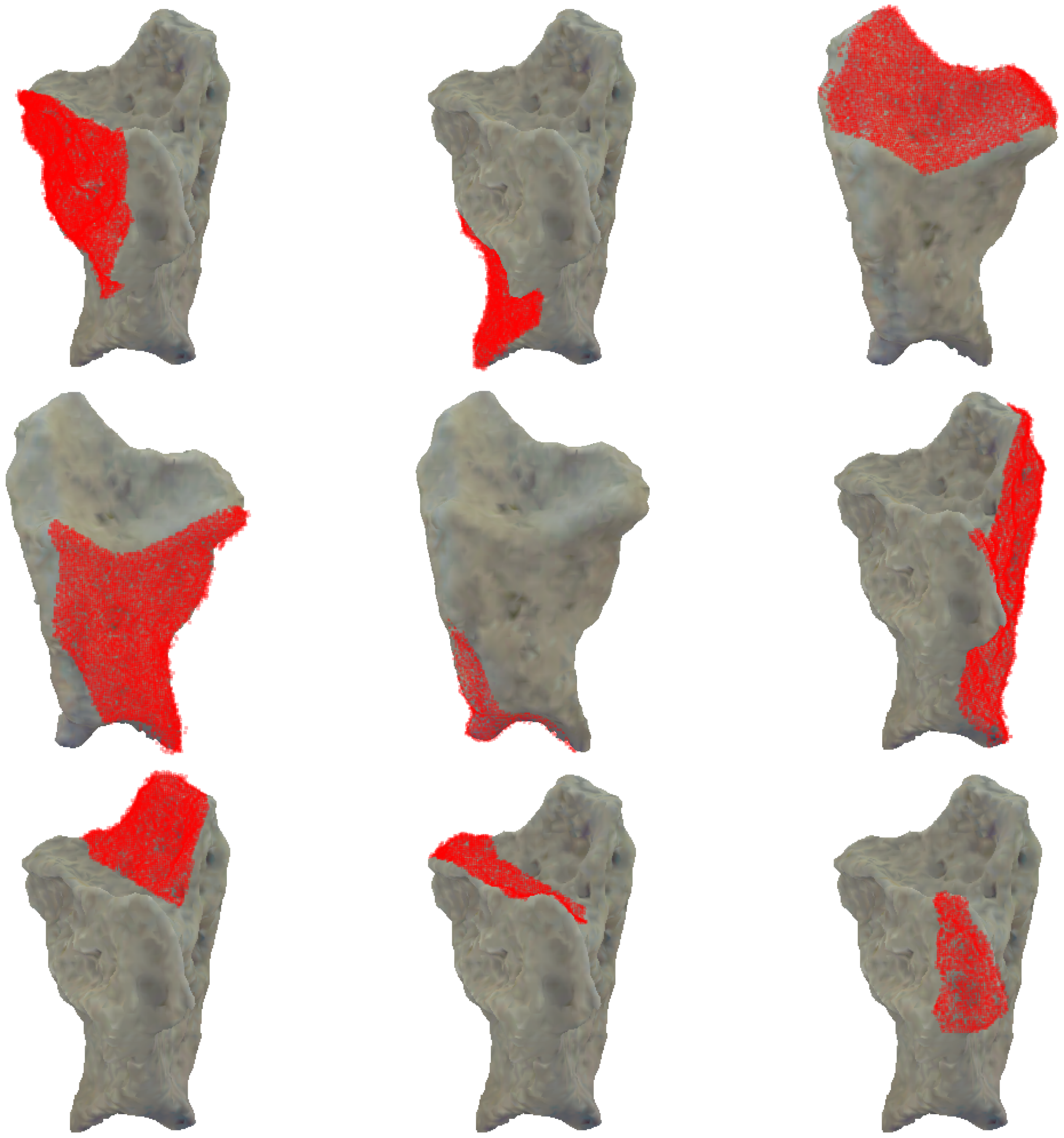


Figure A.13: Faces for lace rock

A.4 Aligning to Coarse Models and Masking Image Segments

In order to study results for individual rock faces when several rock faces and laboratory background objects appear in an image, binary masks were used to identify which pixels were and were not associated with a particular face. These masks were automatically generated. First, a binary mask was generated from the image, segmenting rock from non-rock. This mask from the image was then used to align with the coarse mesh model of the rock discussed in Section A.2. Then, given that the face positions are known relative to the coarse model, masks for each individual face can be generated.

Several variations on the method to generate binary masks from images were used, depending on the dataset, since different approaches worked better for different rocks. A first step either thresholded the image (using the red channel for lava, a combination of thresholded channels for basalt) or clustered the results of Gabor filtering (for lace, lace small angles, lava small angles, basalt small angles). This gave a binary label to each pixel. Because results at the individual pixel level were somewhat noisy, the binary image was dilated and eroded with the goal of getting more solid regions. These regions were segmented and labeled, and the contiguous region that enclosed a sample area that was expected to be on the rock became a binary one, all other regions binary zero. A hole fill operation, which fills all zero-marked regions that cannot be reached from the edge of the image with ones, is also used in some cases. See Fig. A.14 for an illustration of this process [92]. One rock mask is required for each rock-camera relative pose. The light closest to the camera position is used in generating rock masks, to minimize cast shadows.

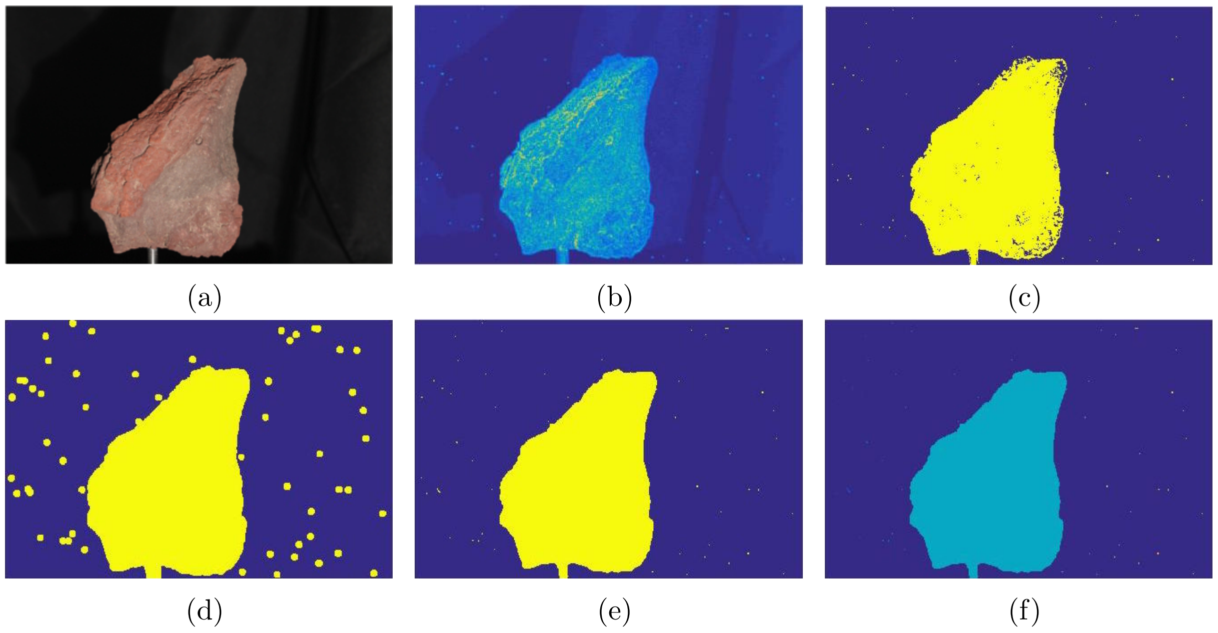


Figure A.14: Process of generating rock masks. (a) original image, (b) visualization of the Gabor filter results, (c) clustering of Gabor filter results, (d) dilate, (e) erode, (f) labeled segment

The process of generating rock masks from images generally worked well, although there were some problems, especially with the colored side of the basalt rock and the higher tripod angles for all rocks. For higher tripod angles, parts of the turntable sometimes get included in the mask (see Fig. A.15). Also, for turntable and tripod configurations where the rock did not take up the center pixels of the image, the rock mask generation process would give the inverse mask (masking out the rock, instead of everything but the rock). For the relatively small number of configurations where this was the case, the rock mask process was re-run with a slightly different window of interest to identify the rock.

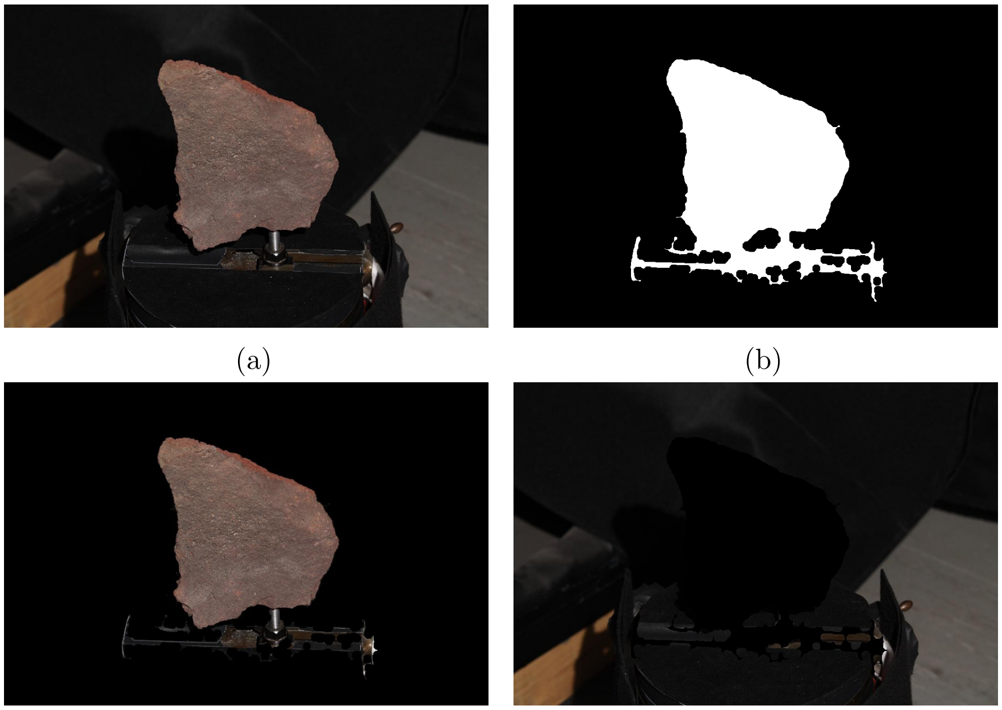
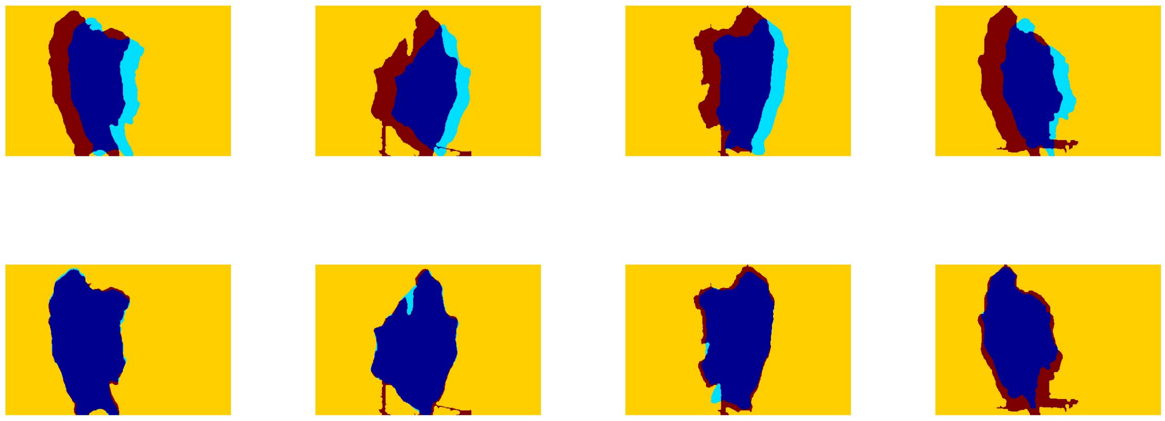

(a) (b) (c) (d)

Figure A.15: Example of a case where the mask generation process encountered challenges. (a) original image, (b) detected mask, (c) “rock” part of original image, (d) “not rock” part of original image

For rock face masks, the rock mask (created as described above) is used to align the coarse model to the image. The coarse mesh model is transformed into the expected pose relative to the camera, given the tripod and turntable angles. This transformed mesh is then used to generate a binary mask. The two masks are then compared, and the sum of the pixels for which they do not agree becomes the error. A new simulated image, and thus a new error value, can be computed for each change in camera position (x, y, z) and orientation (Euler angles about x, y, and z) relative to the initial camera pose. This error is minimized using Nelder-Mead Simplex optimization [93] [2].

Examples of alignment between photo and coarse mesh are shown in Fig. A.16. Fig. A.16a shows a case where alignment worked well. Fig. A.16b and Fig. A.16c show cases where

[2]The fminsearch function from Matlab was used for this operation



(a) (b) (c) (d)

Figure A.16: Example mask alignment. Top images show the masks before alignment. Dark blue means both masks are true, red means only the mask from the image is true, cyan means only the mask from the mesh model is true, and yellow means both are false. Bottom images show the state after alignment.

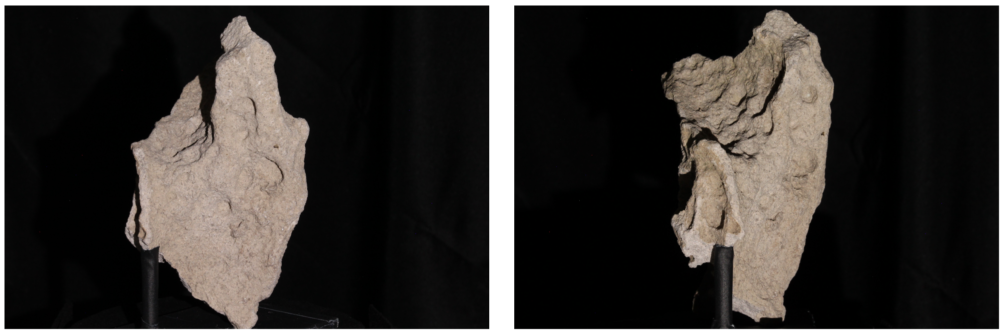


Figure A.17: Cases where alignment worked fairly well despite problems with the photo mask. The left image corresponds to (b) and the right to (c) in Fig. A.16.

alignment worked fairly well, despite shadows in the photograph that obscured part of the rock (see Fig. A.17 for the photographs corresponding to these cases). The alignment method involves non-linear optimization with no guarantee of finding the global minimum, and thus it is prone to getting stuck in local minima. Fig. A.16d shows a case where alignment did not work very well.

Alignment was improved by taking advantage of the fact that, (in most cases), the set of images for one tripod angle was fixed while the turntable was rotated, and the motion of a rock relative to the camera due to turntable rotation is known much more precisely (to 0.5 degrees) than the motion due to camera position and tilt. Alignment results were manually inspected and classified into “good” and “bad”. Then, for the “good” cases, the turntable rotation was taken out of the transformation found by automatic alignment (by multiplying by the inverse). An average of all the transformations for a given tripod angle was computed (each transformation was converted to a twist vector and then the median twist vector was computed). This was then used to get a better starting estimate for rock-camera relative pose. The automatic alignment process was then re-run, producing better results for cases that had been "bad".

Once the rock pose is known, the coarse model can be used to determine where each face would project into the image. A mesh model is created for each face by cropping the coarse mesh model (see Section A.2) to only the triangles that represent the selected face. Another mesh is created for the portions of the coarse mesh not covered by a face. This enables an individual face to be set to white while all other parts of the coarse mesh are set to black such that when rendered against a black background, only the selected face appears white and a binary mask can be created. This method also ensures that faces on the back side of the rock will be properly occluded. A new set of face masks is created for each rock-to-camera relative pose by transforming the meshes based on the aligned pose. A threshold value for the number of one-valued pixels in the mask is used to avoid creating masks where the face is partially or completely occluded. An example is shown in Fig. A.18.

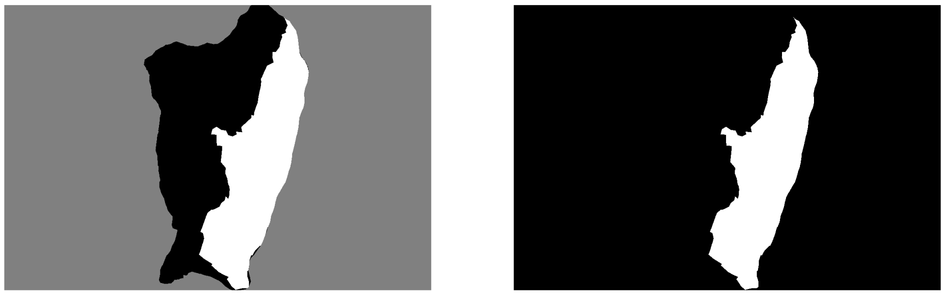


Figure A.18: Example face mask. Rendering of model with everything except the selected face in black (left), and binary mask after setting the background to black (right)

A.5 Data Processing

The following metrics were computed from the laboratory data: number of features detected (for a single image), number of features matched (for pairs of images), and median and standard deviation of error for points in a reconstructed model (for pairs of images).

Features were detected using SIFT [70]. Feature matching for SIFT was done using the keypoint matching executable associated with Bundler [62, 63].Model reconstruction was done using Bundler [62, 63]. Comparison of reconstructed models with coarse was done using point to mesh comparison in CloudCompare [91].

Rock masks or face masks, created as described in Section A.4, are used to filter features before reporting the number of features detected or attempting to match features between images. Fig. A.19 shows an example of filtering detected features using a mask. The alignment done in Section A.4 was used to determine the position and orientation of each face for computing view and illumination angles.

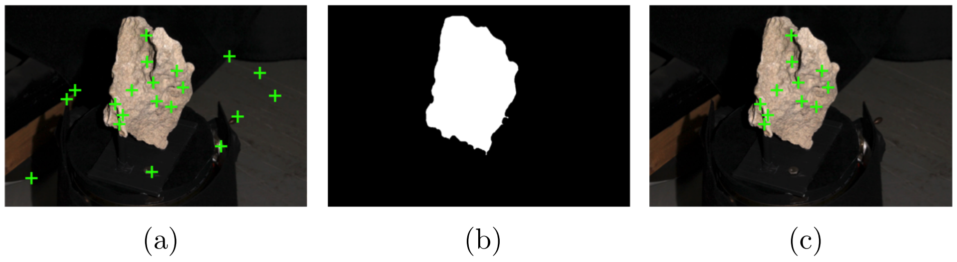


Figure A.19: Example of filtering features using a rock mask. (a) original image with features detected, (b) mask, (c) features remaining after filtering

Different rock faces are different sizes, and the size of a given face changes depending on the view angle and camera distance, so a raw number of features detected is not easily compared across different tripod and turntable angles. Instead, number of features detected is reported as number of features detected per pixel area. (This is computed by dividing by the number of pixels in the mask used).

The number of features matched is, of course, dependent on the number of features detected in each image of the pair on which matching is conducted. The number of features matched can never be greater than the number of features in the image with the greater number of features detected, and if the matches are valid, it should never be greater than the number of features in the image with the smaller number of features detected. Thus, instead of raw number of features matched, the number of features matched is reported as a percentage of the number of features in the image with the smaller number of features detected.

For reconstructed models, the raw structure-from-motion result out of Bundler is only a reconstruction up to scale. Thus, the scale must be adjusted before any comparisons are done. The scale adjustment is computed by comparing the distance between cameras in the Bundler result with the distance between camera positions relative to the rock expressed in a rock-fixed frame. In other words, for turntable rotation, while in reality the camera does not move and the rock is rotated, but if the camera position is expressed in a rock-fixed reference frame, it does change with turntable rotation, and this change is used to adjust the scale of the Bundler result. The result is also transformed using the reference frame of whichever camera is 'camera 1' in Bundler such that all reconstructions can be compared to the coarse mesh model. Even after scale adjustment and transformation into a common frame, however, there is generally some amount of misalignment between the coarse mesh model and the reconstructed model. A point-cloud-to-mesh distance comparison is done between the reconstructed point cloud and the coarse mesh model, which computes an error for each point. The median and standard deviation of this error are reported. See Fig. A.20 for an example of point to mesh comparison. In cases where the reconstruction failed completely, both median and standard deviation of error can be quite large (hundreds or even thousands of meters). Thus, to show detail for more successful reconstructions, plots of median and standard deviation are shown on a limited scale, with all datapoints above the maximum on the limited scale clamped to the maximum value.

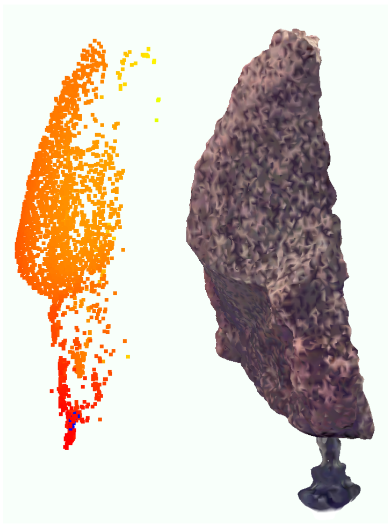


Figure A.20: Comparing a reconstructed point cloud to the coarse mesh model. Points in the reconstructed cloud are colored by distance from the mesh.

Appendix B

Field Sites

B.1 King's Bowl Pit

King's Bowl pit is located along Idaho's Great Rift, in the Craters of the Moon National Monument and Preserve. The pit was formed due to a steam explosion along the rift. It is approximately 76 m long, 30 m wide and 30 m deep. At one end, the pit provides entrance into a deeper cave. While the formation mechanisms of pits on the Moon are likely different from this pit, pits associated with rift systems have been identified on Mars [8]. This pit also provides a good functional analog for planetary pit modeling. Rock layers can be identified in the pit walls, and the identification of shape and material properties for pit wall layers is a useful science objective, facilitating study of the formation of pits and their surrounding geological environments. Data collected at this site include LIDAR scans integrated to build a model of the pit and over 5000 images from tripod-mounted DSLR cameras at surveyed positions [81].

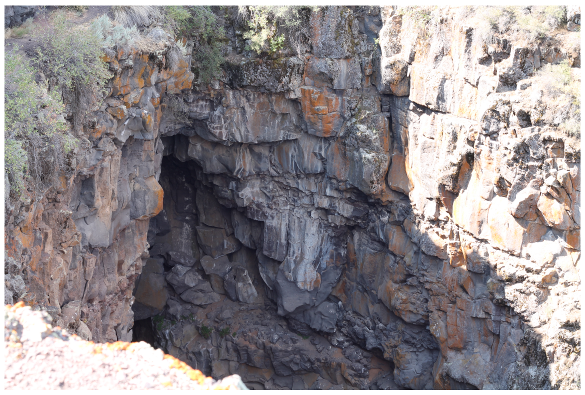


Figure B.1: Sample image from the King's Bowl dataset.

B.2 Indian Tunnel

Indian Tunnel is a lava tube cave in Idaho's Craters of the Moon National Monument and Preserve. It has several skylight pits. Evidence of lava tubes has been identified on the surface of both the Moon and Mars [94, 8]. Data collected at this site include LIDAR scans integrated to build a model of the pit and 612 images from tripod-mounted DSLR cameras [81].

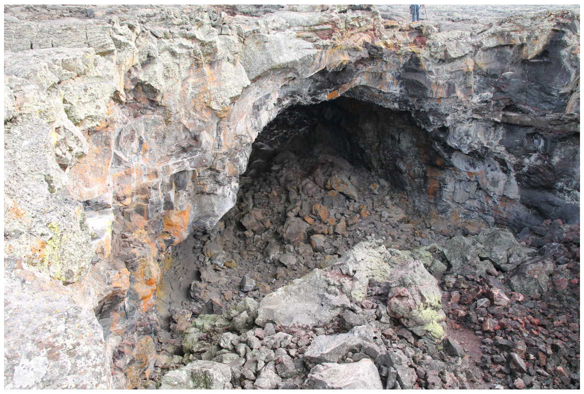


Figure B.2: Sample image from the Indian Tunnel dataset.

B.3 Pluto's Cave

Pluto's Cave is a lava tube cave in Siskiyou, California. It has several skylights. Data collected at this site include over 1700 images from a hand-held DSLR camera.

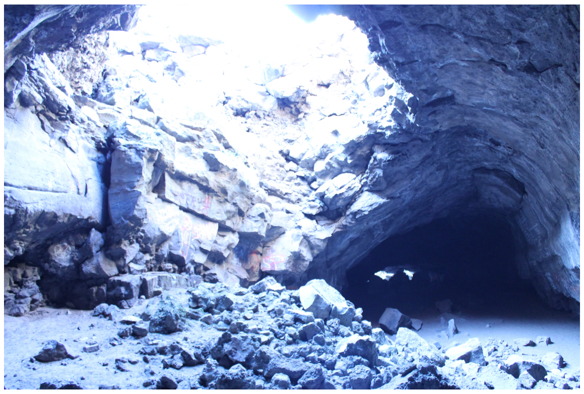


Figure B.3: Sample image from the Pluto's Cave dataset.

B.4 Meteor Crater

Meteor Crater, or the Barringer Meteorite Crater, is a crater in the desert near Flagstaff, Arizona. It is 1.2km in diameter and 180m deep [95]. It was formed by a meteorite impact 50,000 years ago. Craters are found on all rocky planetary bodies, making this a broadly applicable planetary analog site. Data collected at this site include 240 images from a hand-held cell phone camera.

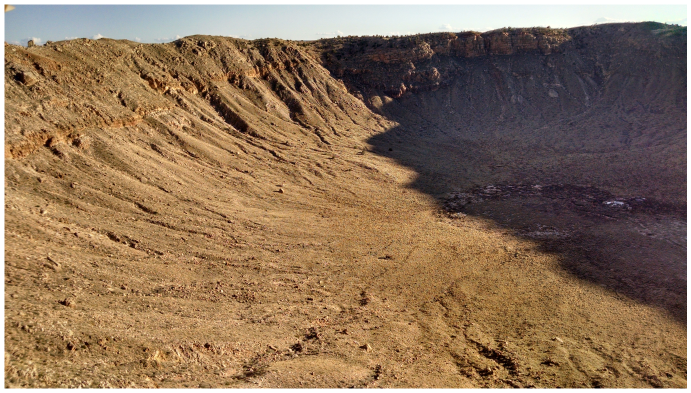


Figure B.4: Sample image from the Meteor Crater dataset.

B.5 Grand Canyon

The Grand Canyon is a nearly 1.5km-deep canyon in Arizona [1]. It is an analog for canyons on Mars. While the entire canyon is about 445 km long, a particular rock formation at Maricopa Point was targeted for imaging (see Fig. B.5). Data collected at this site include 107 images from a hand-held cell phone camera.

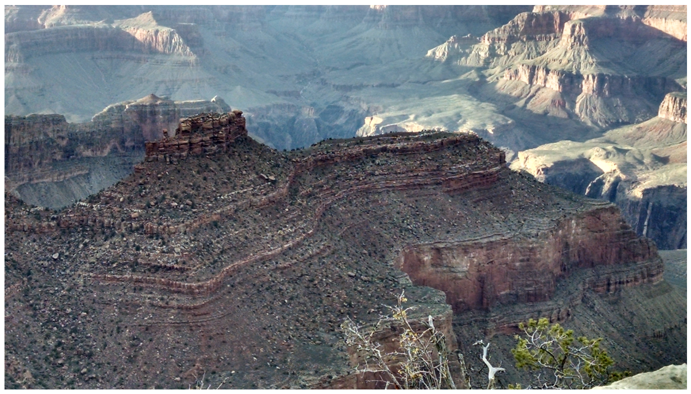


Figure B.5: Sample image from the Grand Canyon dataset.

[1]Dimension data taken from the description of Grand Canyon National Park as a UNESCO World Heritage Site, found at http://whc.unesco.org/en/list/75

Appendix C

PDDL Files

C.1 PDDL 2.2 Domain File for Partial Description of OPTWIND for Pit Modeling

```
(define (domain SimpleRover)
    (:requirements :typing :quantified-preconditions :durative-actions)
    (:types
      LocationTime - object
      locatable - object
      Rover - locatable
      Target - object
      Location - object
      Time - object
    )
    (:predicates
      (at ?loc - locatable ?loc1 - LocationTime)
      (connected ?loc - LocationTime ?loc1 - LocationTime)
      (isat ?loc - LocationTime ?loc1 - Location)
      (visible ?tar - Target ?loc - Location)
      (istime ?loc - LocationTime ?tim - Time)
      (lit ?tar - Target ?tim - Time)
      (available ?loc - LocationTime)
      (viewed ?tar - Target)
    )
    (:durative-action move
     :parameters (?self - Rover ?to - LocationTime ?from - LocationTime)
     :duration (= ?duration 1)
     :condition
       (at start
         (and
          (at ?self ?from)
          (connected ?from ?to)
        )
      )
     :effect
       (at end
         (and
           (at ?self ?to)
           (not (at ?self ?from))
         )
       )
    )

    (:action view
     :parameters (?self - Rover ?targetpatch - Target ?from - LocationTime)
     :precondition
       (and
         (at ?self ?from)
         (exists (?l - Location) (and (visible ?targetpatch ?l) (isat ?from ?l)))
         (exists (?t - Time) (and (lit ?targetpatch ?t) (istime ?from ?t)))
       )
     :effect
       (viewed ?targetpatch)
    )

)
```

C.2 PDDL 2.2 Problem File for Partial Description of OPTWIND for Pit Modeling

```
(define (problem Planning_Problem)
    (:domain SimpleRover)
    (:objects
      RoverObject - Rover
      loctime_1_1 - LocationTime
      loctime_1_2 - LocationTime
      loctime_1_3 - LocationTime
      loctime_1_4 - LocationTime
      loctime_1_5 - LocationTime
      loctime_1_6 - LocationTime
      target1 - Target
```

```
  target3 - Target
  target2 - Target
  target4 - Target
  target5 - Target
  target6 - Target
  loc1 - Location
  loc2 - Location
  loc3 - Location
  loc4 - Location
  loc5 - Location
  loc6 - Location
  loctime_5_3 - LocationTime
  loctime_2_1 - LocationTime
  loctime_2_2 - LocationTime
  loctime_2_3 - LocationTime
  loctime_2_4 - LocationTime
  loctime_2_5 - LocationTime
  loctime_2_6 - LocationTime
  loctime_3_1 - LocationTime
  loctime_3_2 - LocationTime
  loctime_3_4 - LocationTime
  loctime_3_5 - LocationTime
  loctime_3_6 - LocationTime
  loctime_3_3 - LocationTime
  loctime_4_1 - LocationTime
  loctime_4_2 - LocationTime
  loctime_4_3 - LocationTime
  loctime_4_4 - LocationTime
  loctime_4_5 - LocationTime
  loctime_4_6 - LocationTime
  loctime_5_1 - LocationTime
  loctime_5_2 - LocationTime
  loctime_5_4 - LocationTime
  loctime_5_5 - LocationTime
  loctime_5_6 - LocationTime
  time1 - Time
  time2 - Time
  time3 - Time
  time4 - Time
  time5 - Time
)
(:init
(connected loctime_1_1 loctime_1_2)
(connected loctime_1_2 loctime_1_3)
(connected loctime_1_3 loctime_1_4)
(connected loctime_1_4 loctime_1_5)
(connected loctime_1_5 loctime_1_6)
(connected loctime_1_6 loctime_1_1)
(connected loctime_1_2 loctime_1_1)
(connected loctime_1_3 loctime_1_2)
(connected loctime_1_4 loctime_1_3)
(connected loctime_1_5 loctime_1_4)
(connected loctime_1_6 loctime_1_5)
(connected loctime_1_1 loctime_1_6)
(connected loctime_2_1 loctime_2_2)
(connected loctime_2_2 loctime_2_3)
(connected loctime_2_3 loctime_2_4)
(connected loctime_2_4 loctime_2_5)
(connected loctime_2_5 loctime_2_6)
(connected loctime_2_6 loctime_2_1)
(connected loctime_2_2 loctime_2_1)
(connected loctime_2_3 loctime_2_2)
(connected loctime_2_4 loctime_2_3)
(connected loctime_2_5 loctime_2_4)
(connected loctime_2_6 loctime_2_5)
(connected loctime_2_1 loctime_2_6)
(connected loctime_3_1 loctime_3_2)
(connected loctime_3_2 loctime_3_3)
(connected loctime_3_3 loctime_3_4)
(connected loctime_3_4 loctime_3_5)
(connected loctime_3_5 loctime_3_6)
(connected loctime_3_6 loctime_3_1)
(connected loctime_3_2 loctime_3_1)
(connected loctime_3_3 loctime_3_2)
(connected loctime_3_5 loctime_3_4)
(connected loctime_3_6 loctime_3_5)
(connected loctime_3_1 loctime_3_6)
(connected loctime_4_1 loctime_4_2)
(connected loctime_4_2 loctime_4_3)
(connected loctime_4_3 loctime_4_4)
(connected loctime_4_4 loctime_4_5)
(connected loctime_4_5 loctime_4_6)
(connected loctime_4_6 loctime_4_1)
(connected loctime_4_2 loctime_4_1)
```

```
(connected loctime_4_3 loctime_4_2)
(connected loctime_4_4 loctime_4_3)
(connected loctime_4_5 loctime_4_4)
(connected loctime_4_6 loctime_4_5)
(connected loctime_4_1 loctime_4_6)
(connected loctime_5_1 loctime_5_2)
(connected loctime_5_2 loctime_5_3)
(connected loctime_5_3 loctime_5_4)
(connected loctime_5_4 loctime_5_5)
(connected loctime_5_5 loctime_5_6)
(connected loctime_5_6 loctime_5_1)
(connected loctime_5_2 loctime_5_1)
(connected loctime_5_3 loctime_5_2)
(connected loctime_5_4 loctime_5_3)
(connected loctime_5_5 loctime_5_4)
(connected loctime_5_6 loctime_5_5)
(connected loctime_5_1 loctime_5_6)
(connected loctime_3_4 loctime_3_3)
(connected loctime_1_1 loctime_2_1)
(connected loctime_1_1 loctime_2_2)
(connected loctime_1_1 loctime_2_6)
(connected loctime_1_2 loctime_2_1)
(connected loctime_1_2 loctime_2_2)
(connected loctime_1_2 loctime_2_3)
(connected loctime_1_3 loctime_2_2)
(connected loctime_1_3 loctime_2_3)
(connected loctime_1_3 loctime_2_4)
(connected loctime_1_4 loctime_2_3)
(connected loctime_1_4 loctime_2_4)
(connected loctime_1_4 loctime_2_5)
(connected loctime_1_5 loctime_2_4)
(connected loctime_1_5 loctime_2_5)
(connected loctime_1_5 loctime_2_6)
(connected loctime_1_6 loctime_2_5)
(connected loctime_1_6 loctime_2_6)
(connected loctime_1_6 loctime_2_1)
(connected loctime_2_1 loctime_3_1)
(connected loctime_2_1 loctime_3_2)
(connected loctime_2_1 loctime_3_6)
(connected loctime_2_2 loctime_3_1)
(connected loctime_2_2 loctime_3_2)
(connected loctime_2_2 loctime_3_3)
```

```
(connected loctime_2_3 loctime_3_2)
(connected loctime_2_3 loctime_3_3)
(connected loctime_2_3 loctime_3_4)
(connected loctime_2_4 loctime_3_3)
(connected loctime_2_4 loctime_3_4)
(connected loctime_2_4 loctime_3_5)
(connected loctime_2_5 loctime_3_4)
(connected loctime_2_5 loctime_3_5)
(connected loctime_2_5 loctime_3_6)
(connected loctime_2_6 loctime_3_5)
(connected loctime_2_6 loctime_3_6)
(connected loctime_2_6 loctime_3_1)
(connected loctime_3_1 loctime_4_1)
(connected loctime_3_1 loctime_4_2)
(connected loctime_3_1 loctime_4_6)
(connected loctime_3_2 loctime_4_1)
(connected loctime_3_2 loctime_4_2)
(connected loctime_3_2 loctime_4_3)
(connected loctime_3_3 loctime_4_2)
(connected loctime_3_3 loctime_4_3)
(connected loctime_3_3 loctime_4_4)
(connected loctime_3_4 loctime_4_3)
(connected loctime_3_4 loctime_4_4)
(connected loctime_3_4 loctime_4_5)
(connected loctime_3_5 loctime_4_4)
(connected loctime_3_5 loctime_4_5)
(connected loctime_3_5 loctime_4_6)
(connected loctime_3_6 loctime_4_5)
(connected loctime_3_6 loctime_4_6)
(connected loctime_3_6 loctime_4_1)
(connected loctime_4_1 loctime_5_1)
(connected loctime_4_1 loctime_5_2)
(connected loctime_4_1 loctime_5_6)
(connected loctime_4_2 loctime_5_1)
(connected loctime_4_2 loctime_5_2)
(connected loctime_4_2 loctime_5_3)
(connected loctime_4_3 loctime_5_2)
(connected loctime_4_3 loctime_5_3)
(connected loctime_4_3 loctime_5_4)
(connected loctime_4_4 loctime_5_3)
(connected loctime_4_4 loctime_5_4)
(connected loctime_4_4 loctime_5_5)
```

```
(connected loctime_4_5 loctime_5_4)
(connected loctime_4_5 loctime_5_5)
(connected loctime_4_5 loctime_5_6)
(connected loctime_4_6 loctime_5_5)
(connected loctime_4_6 loctime_5_6)
(connected loctime_4_6 loctime_5_1)
(isat loctime_1_1 loc1)
(isat loctime_2_1 loc1)
(isat loctime_3_1 loc1)
(isat loctime_4_1 loc1)
(isat loctime_5_1 loc1)
(isat loctime_1_2 loc2)
(isat loctime_2_2 loc2)
(isat loctime_3_2 loc2)
(isat loctime_4_2 loc2)
(isat loctime_5_2 loc2)
(isat loctime_1_3 loc3)
(isat loctime_2_3 loc3)
(isat loctime_3_3 loc3)
(isat loctime_4_3 loc3)
(isat loctime_5_3 loc3)
(isat loctime_1_4 loc4)
(isat loctime_2_4 loc4)
(isat loctime_3_4 loc4)
(isat loctime_4_4 loc4)
(isat loctime_5_4 loc4)
(isat loctime_1_5 loc5)
(isat loctime_2_5 loc5)
(isat loctime_3_5 loc5)
(isat loctime_4_5 loc5)
(isat loctime_5_5 loc5)
(isat loctime_1_6 loc6)
(isat loctime_2_6 loc6)
(isat loctime_3_6 loc6)
(isat loctime_4_6 loc6)
(isat loctime_5_6 loc6)
(istime loctime_1_1 time1)
(istime loctime_1_2 time1)
(istime loctime_1_3 time1)
(istime loctime_1_4 time1)
(istime loctime_1_5 time1)
(istime loctime_1_6 time1)
```

```
(istime loctime_2_1 time2)
(istime loctime_2_2 time2)
(istime loctime_2_3 time2)
(istime loctime_2_4 time2)
(istime loctime_2_5 time2)
(istime loctime_2_6 time2)
(istime loctime_3_1 time3)
(istime loctime_3_2 time3)
(istime loctime_3_3 time3)
(istime loctime_3_4 time3)
(istime loctime_3_5 time3)
(istime loctime_3_6 time3)
(istime loctime_4_1 time4)
(istime loctime_4_2 time4)
(istime loctime_4_3 time4)
(istime loctime_4_4 time4)
(istime loctime_4_5 time4)
(istime loctime_4_6 time4)
(istime loctime_5_1 time5)
(istime loctime_5_2 time5)
(istime loctime_5_3 time5)
(istime loctime_5_5 time5)
(istime loctime_5_6 time5)
(istime loctime_5_4 time5)
(visible target1 loc3)
(visible target1 loc4)
(visible target1 loc5)
(visible target2 loc4)
(visible target2 loc5)
(visible target2 loc6)
(visible target3 loc1)
(visible target3 loc5)
(visible target3 loc6)
(visible target4 loc1)
(visible target4 loc2)
(visible target4 loc6)
(visible target5 loc1)
(visible target5 loc2)
(visible target5 loc3)
(visible target6 loc2)
(visible target6 loc3)
(visible target6 loc4)
```

```
      (lit target1 time2)
      (lit target1 time3)
      (lit target1 time4)
      (lit target2 time1)
      (lit target2 time2)
      (lit target2 time3)
      (lit target3 time1)
      (lit target3 time2)
      (lit target5 time4)
      (lit target5 time5)
      (lit target6 time3)
      (lit target6 time4)
      (lit target6 time5)
      (at RoverObject loctime_1_1)
    )
    (:goal
      (and
        (at RoverObject loctime_5_6)
        (viewed target3)
        (not (available loctime_5_6))
        (viewed target1)
        (viewed target2)
        (viewed target5)
        (viewed target6)
      )
    )
)
```

Bibliography

[1] Indian Tunnel. 4326'29.32" N and 11331'47.95" W. Google Earth, August 2014. Accessed 11 April 2016.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Surf:speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[3] King's bowl. `https://www.google.com/earth/`, 2014.

[4] King's Bowl. 42.949475 degrees N and 113.216297 degrees W. Google Earth, August 2013. Accessed 12 January 2015.

[5] Grand Canyon. 36 4'32.63" N and 112 8'18.32" W. Google Earth, May 2014. Accessed 9 April 2016.

[6] Meteor Crater. 35 1'55.94" N and 111 1'17.89" W. Google Earth, April 2015. Accessed 8 April 2016.

[7] M. S. Robinson, J. W. Ashley, A. K. Boyd, R. V. Wagner, E. J. Speyerer, B. Ray Hawke, H. Hiesinger, and C. H. van der Bogert. Confirmation of sublunarean voids and thin layering in mare deposits. *Planetary and Space Science*, 69:18–27, 2012.

[8] G. E. Cushing. Candidate cave entrances on mars. *Journal of Cave and Karst Studies*, 74(1):33–47, 2012.

[9] Alfred S. McEwen, Colin M. Dundas, Sarah S. Mattson, Anthony D. Toigo, Lujendra Ojha, James J. Wray, Matthew Chojnacki, Shane Byrne, Scott L. Murchie, and Nicolas Thomas. Recurring slope lineae in equatorial regions of mars. *Nature Geoscience*, 7, January 2014.

[10] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, February 2011.

[11] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2):193 – 207, 1990.

[12] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[13] Zong Woo Geem, Chung-li Tseng, and Yongjin Park. Harmony Search for Generalized Orienteering Problem : Best Touring in China. In *Advances in Natural Computation*, pages 741–750, 2005.

[14] Xia Wang, Bruce L. Golden, and Edward A. Wasil. Using a genetic algorithm to solve the generalized orienteering problem. In Bruce L. Golden, Edward Wasil, and S Raghavan, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 263–273. Springer, 2008.

[15] Heather Jones, Wennie Tabib, and William L. "Red" Whittaker. Planning views to model planetary pits under transient illumination. In *In Proceedings of the IEEE Aerospace Conference*, 2015.

[16] Uland Wong, Heather Jones, Steven Huber, Christopher Cunningham, Warren C. Whittaker, Steve McGuire, Xuesu Xiao, Rick Shanor, Ander Solorzano, Tom Carlone, Wennie Tabib, Christopher Greve, Lauren Schneider, Nathan Otten, and William L. Whittaker. Exploration of planetary skylights and tunnels: NASA Innovative Advanced Concepts phase II report. Technical report, NASA, December 2014.

[17] Konstantinos A Tarabanis, Peter K Allen, and Roger Y Tsai. A Survev of Sensor Planning in Commter Vision. *Robotics and Automation, IEEE Transactions on*, 11(1):86–104, 1995.

[18] William R. Scott, Gerhard Roth, and Jean-Francois Rivest. View Planning for Automated Three-Dimensional Object Reconstruction Inspection. *ACM Computing Surveys*, 35(1):64–96, 2003.

[19] Shigeyuki Sakane and Tomomasa Sato. Automatic Planning of Light Source and Camera Placement for an Active Photometric Stereo System. In *Proceedings 1991 IEEE International Conference on Robotics and Automation*, pages 1080–1087, 1991.

[20] Frederic Solomon and Katsushi Ikeuchi. An illumination planner for Lambertian polyhedral objects. *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, 2:1719–1725, 1995.

[21] Frederic Solomon and Katsushi Ikeuchi. An Illumination Planner for Convex and Concave Lambertian Polyhedral Objects. Technical report, Carnegie Mellon University, Pittsburgh, PA, 1995.

[22] Frederic Solomon. *Illumination planning for photometric measurements.* Phd, Carnegie Mellon University, 1996.

[23] C.K. Cowan and A. Bergman. Determining the camera and light source location for a visual task. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 509–514. IEEE Comput. Soc. Press, 1989.

[24] R Pito. A sensor-based solution to the “next best view” problem. In *Proc. Intl. Conf. on Pattern Recognition*, 1996.

[25] E Kruse, R Gutsche, and F M Wahl. Efficient, iterative, sensor based 3-D map building using rating functions in configuration space. In *ICRA*, Minneapolis, Minnesota, 1996.

[26] Heather Jones, Uland Wong, Kevin Peterson, Jason Koenig, Aashish Sheshadri, and William (Red) L Whittaker. Complementary Flyover and Rover Sensing for Superior Modeling of Planetary Features. In *Proceedings of the 8th International Conference on Field and Service Robotics*. Springer Verlag, July 2012.

[27] K Nagatani, T Matsuzawa, and K Yoshida. Scan-point Planning and 3-D Map Building for a 3-D Laser Range Scanner in an Outdoor Environment. In *FSR*, 2009.

[28] R Sawhney, K M Krishna, and K Srinathan. On fast exploration in 2D and 3D terrains with multiple robots. In *Proc. Intl. Conf. on Autonom. Agents and Multiagent Systems*, 2009.

[29] Stewart John Moorehead. *Autonomous Surface Exploration for Mobile Robots*. Phd, Carnegie Mellon University, 2001.

[30] Geoffrey A Hollinger, Brendan Englot, Franz Hover, Urbashi Mitra, and Gaurav S Sukhatme. Uncertainty-Driven View Planning for Underwater Inspection. In *ICRA*, 2012.

[31] G. a. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *The International Journal of Robotics Research*, 32(1):3–18, November 2013.

[32] B. Englot and F. S. Hover. Three-dimensional coverage planning for an underwater inspection robot. *The International Journal of Robotics Research*, 32(9-10):1048–1073, September 2013.

[33] Paul S. Blaer and Peter K. Allen. View planning and automated data acquisition for three-dimensional modeling of complex sites. *Journal of Field Robotics*, 26:865–891, 2009.

[34] P. Wang, R. Krishnamurti, and K. Gupta. Metric view planning problem with traveling cost and visibility range. In *Proc. IEEE International Conference on Robotics and Automation*, pages 1292–1297, April 2007.

[35] Tara Estlin, Forest Fisher, Daniel Gaines, Caroline Chouinard, Steve Schaffer, and Issa Nesnas. Continuous Planning and Execution for an Autonomous Rover. In *3rd Intl. NASA Workshop on Planning and Scheduling for Space*, 2002.

[36] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[37] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, number May, pages 3310–3317. IEEE Comput. Soc. Press, 1994.

[38] Sven Koenig and Maxim Likhachev. D * Lite. In *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, pages 476–483, 2002.

[39] David E. Smith. Choosing Objectives in Over-Subscription Planning. In *Intl. Conf. on Automated Planning and Scheduling*, 2004.

[40] Liam Pedersen, David E. Smith, Matthew Deans, Randy Sargent, Clay Kunz, David Lees, and Srikanth Rajagopalan. Mission Planning and Target Tracking for Autonomous Instrument Placement. In *Aerospace Conference*, pages 34–51, Big Sky, MT, 2005.

[41] C.Peter Keller. Algorithms to solve the orienteering problem: A comparison. *European Journal of Operational Research*, 41(2):224–231, July 1989.

[42] Joanna Karbowska-Chilinska and Pawel Zabielski. Genetic Algorithm Solving the Orienteering Problem with Time Windows. In Jerzy Switek, Adam Grzech, Pawe Switek, and Jakub M. Tomczak, editors, *Advances in Systems Science*, volume 240 of *Advances in Intelligent Systems and Computing*, pages 609–619. Springer International Publishing, Cham, 2014.

[43] William Kenneth Mennell. *HEURISTICS FOR SOLVING THREE ROUTING PROBLEMS : CLOSE-ENOUGH TRAVELING SALESMAN PROBLEM , CLOSE-ENOUGH VEHICLE ROUTING PROBLEM , SEQUENCE-DEPENDENT TEAM ORIENTEERING PROBLEM.* Phd, University of Maryland, 2009.

[44] Don A. Grundel and David E. Jeffcoat. Formulation and solution of the target visitation problem. In *Proceedings of the AIAA 1st Intelligent Systems Technical Conference*, 2004.

[45] Ashwin Arulselvan, Clayton W Commander, and Panos M Pardalos. A hybrid genetic algorithm for the target visitation problem. Technical report, 2007.

[46] Kadri Sylejmani, Jiirgen Dorn, and Nysret Musliu. A Tabu Search approach for Multi Constrained Team Orienteering Problem and its application in touristic trip planning. In *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, pages 300–305. IEEE, December 2012.

[47] D. Bernard, G. Dorais, E. Gamble, B. Kanefsky, J. Kurien, G. Man, W. Millar, N. Muscettola, P. Nayak, K. Rajan, N. Rouquette, B. Smith, W. Taylor, and Y. Tung. Spacecraft Autonomy Flight Experience: The DS1 Remote Agent Experiment. In *AIAA Conference*, 1999.

[48] N. Muscettola, C. Fry, K. Rajan, B. Smith, S. Chien, G. Rabideau, and D. Yan. Onboard planning for New Millennium Deep Space One autonomy. In *1997 IEEE Aerospace Conference*, volume 1, pages 303–318. Ieee, 1997.

[49] Jeremy Frank and Ari K. Jónsson. Constraint-based Attribute and Interval Planning. *Journal of Constraints*, 8(4), 2003.

[50] Javier Barreiro, Matthew Boyce, Minh Do, Jeremy Frank, Michael Iatauro, Tatiana Kichkaylo, Paul Morris, James Ong, Emilio Remolina, Tristan Smith, and David Smith. EUROPA: A Platform for AI Planning, Scheduling, Constraint Programming, and Optimization. In *Proceedings of the 22nd International Conference on Automated Planning & Scheduling (ICAPS-12) - The 4th International Competition on Knowledge Engineering for Planning and Scheduling*, 2012.

[51] John L Bresina, Ari K Jónsson, Paul H Morris, and Kanna Rajan. Activity Planning for the Mars Exploration Rovers. In *International Conference on Automated Planning and Scheduling*, pages 40–49, 2005.

[52] Maria Fox and Derek Long. pddl2 . 1 : An Extension to pddl for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

[53] Stefan Edelkamp and Jörg Hoffmann. PDDL2 . 2 : The Language for the Classical Part of the 4th International Planning Competition. Technical Report 195, IPC, 2004.

[54] Chih-wei Hsu, Benjamin W Wah, Ruoyun Huang, and Yixin Chen. Constraint Partitioning for Solving Planning Problems with Trajectory Constraints and Goal Preferences. In *Joint Conf. on Articial Intelligence*, pages 1924–1929, 2007.

[55] Chih-wei Hsu and Benjamin W Wah. The SGPlan Planning System in IPC-6. In *6th. Int. Planning Competition Booklet (ICAPS-08)*, 2008.

[56] R. L. Cook and K. E. Torrance. A Reflectance Model for Computer Graphics. *ACM Transactions on Graphics*, 1(1):7–24, January 1982.

[57] M. J. Broxton and L. J. Edwards. The Ames Stereo Pipeline: Automated 3D Surface Reconstruction from Orbital Imagery. In *Lunar and Planetary Science Conference*, 2008.

[58] Z. M. Moratto, M. J. Broxton, R. A. Beyer, M. Lundy, and K. Husmann. Ames Stereo Pipeline, NASA's Open Source Automated Stereogrammetry Software. In *Lunar and Planetary Science Conference*, 2010.

[59] S W Squyres, A H Knoll, R E Arvidson, J W Ashley, J F I I I Bell, W M Calvin, P R Christensen, B C Clark, B A Cohen, P A de Souza, L Edgar, W H Farrand, I Fleischer, R Gellert, M P Golombeck, J Grant, J Grotzinger, A Hayes, H Herkenhoff, J R Johnson, B Jolliff, G Klingelhofer, A Knudson, R Li, T J McCoy, S M McLennan, D W Ming, D W Mittlefehldt, R V Morris, J W Jr Rice, C Schroder, R J Sullivan, A Yen, and R A Yingst. Exploration of Victoria Crater by the Mars rover Opportunity. *Science*, 324(5930):1058–1061, 2009.

[60] R Li, K Di, J W Hwangbo, Y Chen, and Athena Science Team. Rigorous photogrammetric processing of HiRISE stereo images and topographic mapping at Mars Exploration Rover landing sites. In *LPSC*, League City, TX, 2008.

[61] Rongxing Li, Steven W Squyres, Raymond Arvidson, Brent Archinal, Jim Bell, Yang Cheng, Larry Crumpler, David J Des Marais, Kaichang Di, Todd A Ely, Matt Golombek, Eric Graat, John Grant, Joe Guinn, Andrew Johnson, Ron Greeley, Randolph Kirk, Mark Maimone, Larry H Matthies, Mike Malin, Tim Parker, Mike Sims, Larry A Soderblom, Shane Thompson, Jue Wang, Patrick Whelley, and Fengliang Xu. Initial results of rover localization and topographic mapping for the 2003 Mars Exploration Rover mission. *Photogrammetric Engineering & Remote Sensing*, 71(10):1129–1142, 2005.

[62] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo Tourism: Exploring image collections in 3D. In *Proceedings of SIGGRAPH*, 2006.

[63] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the World from Internet Photo Collections. *International Journal of Computer Vision*, 2007.

[64] Yasutaka Furukawa and Jean Ponce. Accurate, Dense, and Robust Multi-View Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[65] Yasutaka Furukawa, Brian Curless, Steven M Seitz, and Richard Szeliski. Towards Internet-scale Multi-view Stereo. In *CVPR*, 2010.

[66] C. Wu. Towards linear-time incremental structure from motion. In *3DV*, 2013.

[67] C. Wu, S. Agarwal, B. Curless, and S. M. Seits. Multicore bundle adjustment. In *CVPR*, 2011.

[68] M. Lionel, P. Moulon, and M. Pascal. Automatic homographic registration of a pair of images with a contrario elimination of outliers. In *IPOL*, 2012.

[69] S. Fuhrmann, F. Langguth, and M. Goesele. Mve - a multi-view reconstruction environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage*, 2014.

[70] D G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2):91–110, 2004.

[71] I. Antonenko, S. J. Robbins, P. L. Gay, C. Lehan, and J. Moore. Effects of incidence angle on crater detection and the lunar isochron system: Preliminary results from the CosmoQuest MoonMappers citizen science project. In *Lunar and Planetary Science Conference*, 2013.

[72] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Single-Source Shortest Paths*, chapter 24, pages 580–619. MIT Press, 2 edition, 2001.

[73] Intelligent Robotics Group. The Ames Stereo Pipeline: NASA's open source automated stereogrammetry software, October 2014.

[74] L. R. Gaddis, J Anderson, K Becker, T Becker, D Cook, K Edwards, E Eliason, T Hare, H Keiffer, E M Lee, J Matthews, Larry A Soderblom, T Sucharski, and J Torson. An Overview of the Integrated Software for Imaging and Spectrometers. In *Lunar and Planetary Science Conference*, 1997.

[75] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, 2006.

[76] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics*, 32(3), June 2013.

[77] C. H. Acton. Ancillary data services of NASA's navigation and ancillary information facility. *Planetary and Space Science*, 44(1):65–70, 1996.

[78] T. Moller and B Trumbore. Fast, minimum storage ray/triangle intersection. *Journal of Graphics Tools*, 1997.

[79] Tiago Vaquero, Rosimarci Tonaco, Gustavo Costa, Flavio Tonidandel, José R Silva, and J Christopher Beck. itSIMPLE4 . 0 : Enhancing the Modeling Experience of Planning Problems. In *Proceedings of the ICAPS 2012*, pages 11–14, 2012.

[80] Tiago S Vaquero, José R Silva, Flavio Tonidandel, and J Christopher Beck. itSIMPLE : Towards an Integrated Design System for Real Planning Applications. *The Knowledge Engineering Review*, 28(2):215–230, 2013.

[81] Uland Wong, Warren Whittaker, Heather Jones, and Red Whittaker. Cmu planetary pits and caves 3d dataset, December 2014.

[82] D. Martinec. Robust multiview reconstruction. 2008.

[83] A. Chatterjee and V. M. Govindu. Efficient and robust large-scale rotation averaging. In *ICCV*, 2013.

[84] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1992.

[85] C. Yang and G. Medioni. Object modelling by registration of multiple range images. In *Image Vision Computing*, 1991.

[86] Zhang Zhengyou. Iterative point matching for registration of free-form curves and surfaces. In *International Journal of Computer Vision*, 1994.

[87] Cloudcompare, 2014. [GPL Software].

[88] MeshLab.

[89] Uland Wong. *Lumenhancement: Exploiting Appearance for Planetary Modeling.* Phd, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2012.

[90] FARO Technologies Inc. *FARO Focus3D: Features, Benefits & Technical Specifications*, 4 2013.

[91] Cloudcompare, 2015. [GPL Software].

[92] M. Haghighat, S. Zonouz, and M. Abdel-Mottaleb. CloudID: Trustworthy cloud-based and cross-enterprise biometric identification. *Expert Systems with Applications*, 42(21):7905–7916, 2015.

[93] J.C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the nelder-mead simplex method in low dimensions. *SIAM Journal of Optimization*, 9(1):112–147, 1998.

[94] V R Oberbeck, W L Quaide, and R Greeley. On the origin of lunar sinuous rilles. *Modern Geology*, pages 75–80, 1969.

[95] E.M. Shoemaker and S.W. Kieffer. *Guidebook to the Geology of Meteor Crater, Arizona*, volume 17 of *Meteorite Studies Publication.* Arizona State University, 1974.