Rover Localization in Sparsely-Featured Environments

Samuel Yim

CMU-RI-TR-16-33

December 2016

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

Abstract

Autonomous outdoor localization is a challenging but important task for rovers. This is especially true in desert-like environments such as those on Mars, where features can be difficult to distinguish and GPS is not available. This work describes a localization system called MeshSLAM, which requires only stereo images as inputs. MeshSLAM uses the spatial geometry of rocks as landmarks in a GraphSLAM algorithm. These landmarks are termed "constellations," and this work will present and compare methods of generating, describing and matching constellations. Motion is estimated through visual odometry.

This work will also present two new methods of detecting rocks in an image — one that uses superpixel clustering and ground plane fitting, and another that uses a convolutional neural network. The analysis of feature descriptors and descriptor matching that follows will show that accurate landmark matching can be achieved by systematically building convex hull boundary descriptors in each image, and rejecting outliers using RANSAC and motion-invariant rock features.

Several thousand stereo images were collected by the rover Zoë from the Atacama Desert in Chile, and these are used to validate the system. On these desert images containing only rocks, MeshSLAM was able to achieve 100% precision in landmark association and less than 2% localization error across a 360 meter path.

Acknowledgments

I would like to thank my advisor Professor David Wettergreen for making this research possible, and for his invaluable guidance and support throughout. I would also like to thank my committee members Professor Michael Kaess and Greydon Foil for their helpful feedback. I am grateful to Michael Wagner, Srinivasan Vijayarangan, and Matthew Deans for their technical guidance and insights, as well as to the NVIDIA Corporation for generously donating the Tesla K40 GPU used in this research. Lastly, I'd like to thank the rest of the RI community for many great memories.

Contents

1 Introduction 1
- 1.1 Motivation 2
- 1.2 Overview 3

2 Rock Detection with Computer Vision-based Methods 4
- 2.1 Related Work 5
- 2.2 Method 5
  - 2.2.1 Keypoint Descriptors 5
  - 2.2.2 TextureCam 7
  - 2.2.3 Relative Height of Clustered Superpixels (RHoCS) 8
- 2.3 Results 11
  - 2.3.1 Ground Truth 11
  - 2.3.2 Recall Results 11
- 2.4 Discussion 11

3 Rock Detection with CNNs 15
- 3.1 Background and Related Work 15
  - 3.1.1 Software background 16
- 3.2 Classifying Terrain Types with Caffe 17
- 3.3 Rock Detection with CN24 18
- 3.4 Rock Detection with MatConvNet 22
  - 3.4.1 Training a CNN for Rocks (RockNet) 22
- 3.5 Results 23
  - 3.5.1 Recall Results 23
- 3.6 Discussion 26

4 Constellations as Landmarks 27
- 4.1 Related Work 27
- 4.2 Method 29
  - 4.2.1 Generating Constellations 29
  - 4.2.2 Describing Constellation Geometry 34
  - 4.2.3 Incorporating Rock Features 34
  - 4.2.4 Matching Constellations 36
- 4.3 Results 41
  - 4.3.1 Comparing Similarity Metrics 41
  - 4.3.2 Monte Carlo Simulations 45
  - 4.3.3 Generating All Constellations 46

4.3.4 Comparing Constellation Vector Descriptors 47
4.3.5 Varying the Sizes of Convex Hull Boundary Descriptors 49
4.3.6 Varying Rock Position Noise 49
4.3.7 Varying the Number of Features Per Rock 50
4.3.8 CHB Descriptor Computation 54
4.4 Discussion 56
5 MeshSLAM 57
5.1 Background and Related Works 57
5.1.1 GraphSLAM 57
5.1.2 Incremental Smoothing and Mapping 58
5.1.3 Powell's Dog-Leg Incremental Optimization 59
5.1.4 Georgia Tech Smoothing and Mapping Library 60
5.2 Method 60
5.3 Results 66
5.3.1 Path A 66
5.3.2 Path B 67
5.3.3 Path C 69
5.3.4 Results Summary 71
5.4 Discussion 72
5.4.1 Limitations 72
5.4.2 Alternative Approaches 73
6 Conclusions 75
6.1 Summary of Results 75
6.2 Contributions 76
6.3 Future Work 77
6.4 Closing Remarks 77

References **79**

List of Figures

1 Scenes from Mars and the Atacama Desert 1
2 MeshSLAM system overview 3
3 Atacama Desert terrain samples 4
4 Keypoint descriptors on rocks 6
5 TextureCam images, training labels and rock probability heat maps. In the training labels, rock pixels are colored blue and ground pixels are colored red. In the heat maps, yellow pixels indicate higher probability of rock classification. 7
6 TextureCam failure cases 8
7 RHoCS algorithm overview 10
8 RHoCS recall results **A** 12
9 RHoCS recall results **B** 12
10 RHoCS recall results **C** 13
11 RHoCS recall results **D** 13
12 Terrain classification samples 17
13 CN24 results on synthetic images 18
14 CN24 results on simple desert scenes 19
15 CN24 results for various input channels 20
16 Hand-labeled 8-class training images 21
17 VGGNet model 22
18 Affine transformations of training images 23
19 RockNet recall results **A** 24
20 RockNet recall results **B** 24
21 RockNet recall results **C** 25
22 RockNet recall results **D** 25
23 All 3-constellations on a desert scene 31
24 Delaunay triangles on a desert scene 31
25 Connected Delaunay triangles 33
26 Convex hull boundaries of connected Delaunay triangles 33
27 Shifted and rotated CHB constellation 37
28 Rock error estimation 38
29 Constellation matching process. 40
30 Comparison between similarity metrics for uniformly distributed variables 43
31 Comparison between similarity metrics for exponentially distributed variables . . 44
32 Atacama Desert distribution of rocks 46

33 False positive rate when generating all 3-constellations 47
34 Comparison between pairwise and polar constellation descriptors 48
35 Varying size of convex hull boundary descriptors 49
36 Precision and recall curves for **pairwise** descriptors at varying thresholds. 51
37 Precision and recall curves for **polar** descriptors at varying thresholds. 51
38 Precision and recall curves for **CHB** descriptors at varying measurement errors. . 51
39 Varying the number of uniformly distributed features per rock 52
40 Distribution of rock surface area and pixel area in the Atacama dataset. 53
41 Varying the number of exponentially distributed features per rock 54
42 Number of point comparisons vs. number of rocks in scene 55
43 RANSAC filtering for odometry 61
44 First sample set of frames showing 5-constellations 64
45 Second sample set of frames showing 5-constellations 65
46 Path **A** ground truth and pure odometry result 66
47 **Path A** MeshSLAM path. 67
48 Path **B** ground truth and pure odometry result 68
49 **Path B** MeshSLAM path. 68
50 Path **C** ground truth and pure odometry result 69
51 **Path C** MeshSLAM path. 70
52 Visualization of matches between all scenes 71
53 Comparison between ground truth, pure odometry, and MeshSLAM paths 72

List of Tables

1 Recall results of RHoCS 11
2 RHoCS parameters 11
3 Recall results of RockNet 26
4 Average number of point comparisons 56
5 MeshSLAM accuracy results 72

1 Introduction

This work presents a method for rover localization in challenging GPS-denied environments where travel distances are large and features are sparse. One such environment is Mars, where it is desirable on science missions to accurately navigate to targets of interest and GPS is unavailable or inadequate. It would also be useful to autonomously keep track of locations of interest and potentially return to them. An accurate vision-based localization and mapping system could provide both of these capabilities.

This thesis describes the design and development of such a system, called MeshSLAM. It uses simultaneous localization and mapping (SLAM) techniques with landmarks formed from rocks, termed *constellations*. This work will discuss the full pipeline of the MeshSLAM system which requires only a set of stereo images to estimate a rover's state.

The primary goal of this work is to demonstrate that localization using collections of rocks as landmarks can be effective in barren environments. The image datasets used to test the system were obtained from the Atacama Desert in Chile in 2015 by a set of stereo cameras on the rover Zoë [7]. For comparison, images from the Atacama desert are shown alongside Mars scenes in Figure 1.

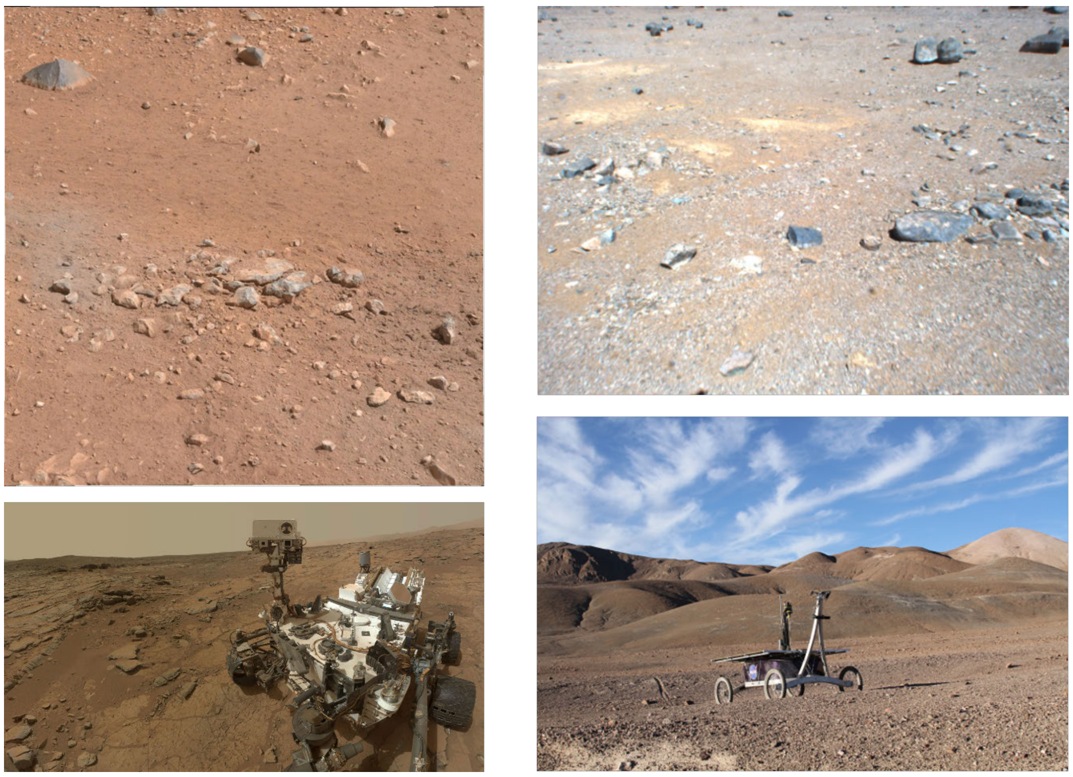

Figure 1: Scenes from Mars [29] (left) and the Atacama Desert [10] (right).

1.1 Motivation

The rovers on Mars have limited bandwidth and data transmission rates; it can take several hours to send 100 megabits of data directly to Earth. Additionally, the rotation of Mars periodically obscures the rover from Earth's view. When Earth is visible, the rover's power and thermal limitations allow direct-to-Earth transmissions for at most three hours a day. An alternative, more efficient means of transmitting data is through orbiters, since they are in view of Earth for most of the day and have access to more solar power. However, the Mars rover Curiosity can only trasmit data to the orbiter for the eight minutes per sol it takes to pass over the rover [1].

Thus, autonomy is an important capability for planetary rovers, allowing them to achieve science and engineering goals without communication from scientists on Earth. Increasing the number of actions a Mars rover can perform autonomously increases the amount of data and useful results that can be returned to scientists. Currently, Mars rovers estimate their position through dead reckoning, fusing data from an IMU, heading sensor, and sun tracker. When error inevitably accumulates on the position estimate, it becomes necessary to transmit stereo images back to Earth and manually correlate stereo points within images. This is a costly and inefficient procedure. Given the limited lifespan of Mars rovers and the cost of a Mars mission, it is crucial to attain maximum efficiency in data collection, which is possible through autonomy.

Rovers have demonstrated several autonomous navigation capabilties on Mars. Visual odometry was first shown to be effective on the rovers Spirit and Opportunity, where it was used to detect slip, improve position knowledge, ensure accurate science imaging, and improve vehicle safety on slopes [46]. Rovers have also had success with local terrain mapping and planning algorithms to reach points of interest while avoiding obstacles [47].

However, no autonomous localization system is currently being used on the Mars rovers, in part because of the difficulty of distinguishing recognizable landmarks in the Mars landscape. A successful SLAM system useable on Mars has implications for further autonomous possibilities. By identifying loop closures, the rover gains an intelligent understanding of the actual topology of its traverse, as opposed to filtering various odometry measurements. This gives it the possibility of autonomously using shortcuts and backtracking. Additionally, autonomous localization leads to more accurate traverses and less time spent correcting rover trajectories and transmitting instructions.

1.2 Overview

This thesis describes the MeshSLAM system and its effectiveness in localizing a rover within the Atacama Desert. A graphical overview of the MeshSLAM system is shown in Figure 2. Emphasis is placed on the front-end SLAM processes outlined in red — rock detection and constellation formation — as these are novel aspects of this work developed for the unique challenges of desert-like environments.

Sections 2 and 3 will focus on rock detection, which provides the basis for forming robust constellations. Section 4 will examine various methods of building up constellations from identified rocks and reliably matching them. In Section 5, the constellations are tested as landmarks in the context of the full MeshSLAM system. In each of these sections, the relevant methods are presented first, followed by an analysis of results. Finally, Section 6 provides a summary of all results and contributions as well as potential areas for future work.

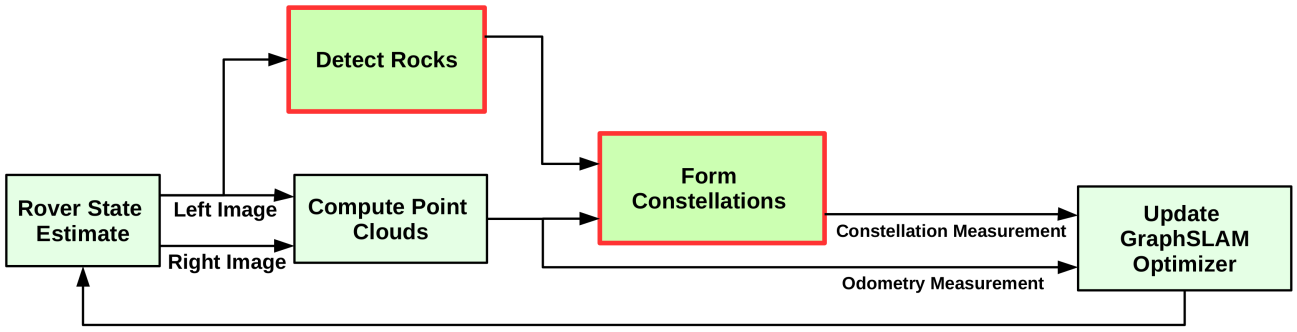

Figure 2: MeshSLAM system overview. The primary contributions of this work are outlined in red.

2 Rock Detection with Computer Vision-based Methods

In most SLAM systems, clearly identifiable and easily described objects are used as landmarks — trees, doors, signs, etc. This is because recognizable landmarks can be accurately matched between scenes, which in turn leads to accurate localization. In the case of deserts, the only objects available with any consistency are rocks. However, following an unmodified SLAM approach with rocks as landmarks will likely fail because individual rocks cannot generally be considered recognizable landmarks. That is, trying to match individual rocks across many frames under changing lighting conditions will almost certainly result in false positive matches and poor localization.

In the MeshSLAM system, we intend to form recognizable landmarks from specific groupings of rocks and the term *constellation* will be used to distinguish this specific type of landmark. The idea and details of forming constellations are discussed in depth in Section 4. For now, we focus on the idea that a constellation can only achieve the same consistency of identification as a typical landmark if each of its consituent rocks are reliably detected. Thus, accurate rock detection is a key component to forming consistent constellation, as well as the first step in the localization pipeline.

Rock detection has been attempted with a variety of computer vision algorithms. It remains a challenging task due to the diversity of shapes, colors, materials, and textures that describe a rock even within the same geographic region. Rocks often occlude each other or are themselves occluded by the ground. They are ill-defined objects, and often the distinction between ground and rock is unclear. Lighting and shadows can significantly affect the appearance of rocks. Additionally, as rocks lack unique identifying characteristics, it is often challenging to track them from one frame to the next, especially if the camera angle changes significantly.

Figure 3 shows a random sample of scenes from the Atacama Desert, and illustrates some of the difficulties involved in rock detection.

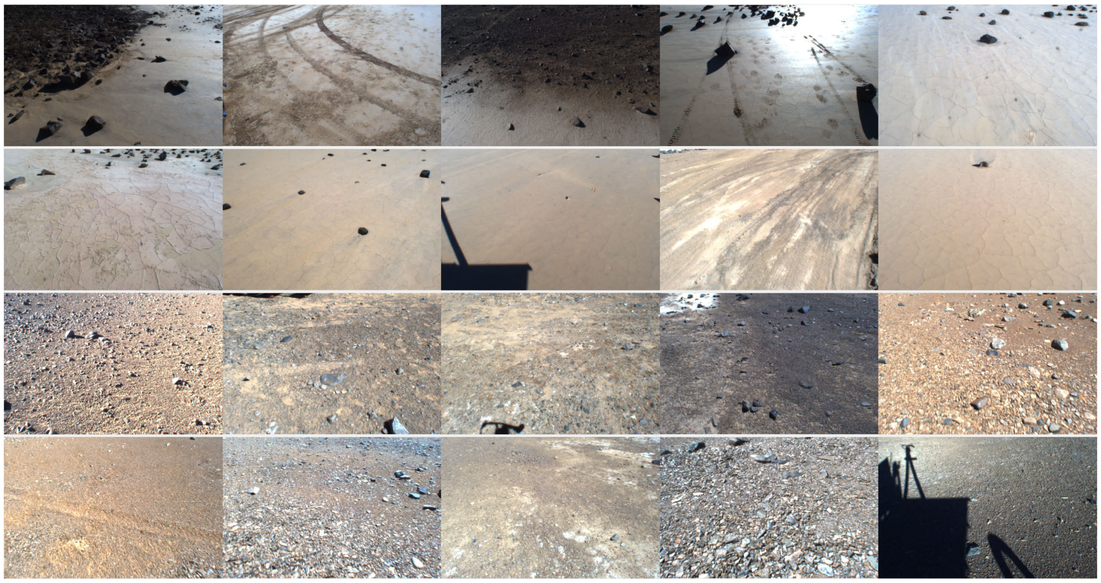

Figure 3: Landscapes and scenes vary significantly even within the same desert.

2.1 Related Work

Rock detection is an important task for planetary rovers, as it is useful for obstacle avoidance, path planning, and sample selection.

Current methods for rock detection vary widely. Thompson and Castaño provide an overview and performance comparison of several earlier rock detection methods [64]. One such method is the Rockfinder algorithm, which uses denoising and smoothing filters with a multi-scale pyramid edge detector [11]. Another that uses stereo camera information fits a ground plane to the image and performs segmentation on the resulting height map [23]. The Marsokhod Shadow Detector searches for candidate points that lie on the edge of illuminated and shadowed rock regions, based on knowledge of the sun angle and a spherical lighting model [25]. The Smoothed Quick Uniform Intensity Detector (SQUID) removes terrain texture with a bilateral filter, and then identifies contiguous regions with similar pixel intensity as rocks [64]. This simple, intuitive algorithm was developed primarily for synthetic images, which generally contain less ambiguous rock features than real images.

Thompson and Castaño also looked at using support vector machines (SVM) to classify each pixel by first characterizing the pixels as feature vectors of local intensity values [64]. Dunlop took this a step further and used an SVM with feature vectors composed of numerous automatically extracted rock characteristics, including roundness, convexity, circularity, compactness, and various colorspace metrics [18].

More recent work by Wagstaff et al. in 2013 classifies rocks based on texture analysis, and can be used for onboard image processing by an instrument called TextureCam [69]. It makes use of a random forest classifier, in which decision trees are trained on subsamples of a labeled dataset. Training involves allowing the decision trees to progressively grow and split until they determine the tests that optimally distinguish between the various provided labels (e.g. sky, ground, rock). The strengths and weaknesses of TextureCam will be explored in more depth in Section 2.2.2.

2.2 Method

Various rock detection strategies are shown below. Keypoint descriptors and the TextureCam method in Sections 2.2.1 and 2.2.2 are used to motivate a novel method for rock detection that is based on superpixel clustering and stereo information, presented in Section 2.2.3.

2.2.1 Keypoint Descriptors

Keypoints decriptors are the simplest types of feature descriptors and are shown here for qualitative comparison to the later methods of rock detection. The three point descriptors tested are Speeded up Robust Features (SURF) [4], Gilles keypoints [22], and Laplacian of Gaussian (LoG) [42].

SURF is a fast and commonly used algorithm for computing point descriptors. The algorithm detects points where the determinant of the Hessian matrix is maximized in some local region,

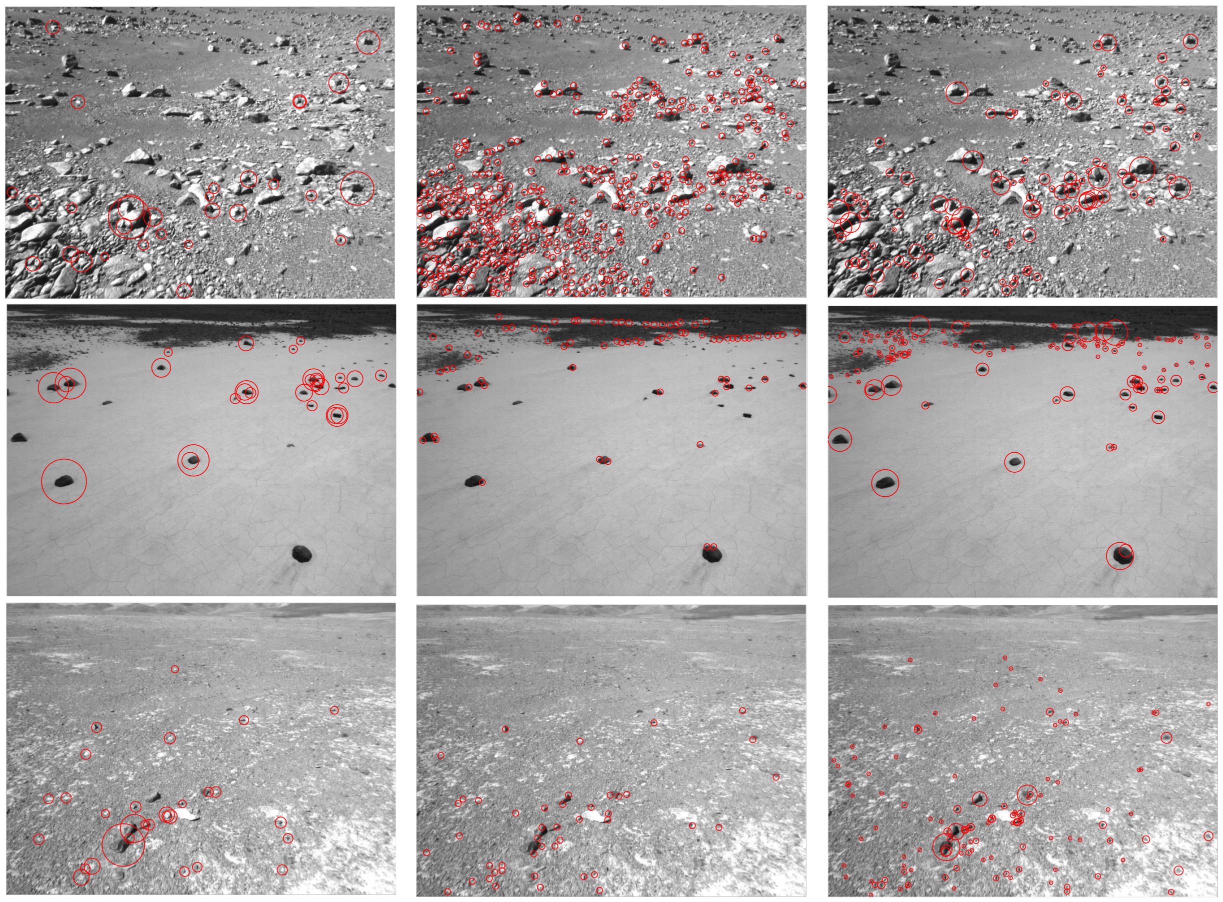

Figure 4: Point descriptors. Left column shows SURF features, middle column shows Gilles features, and right column shows Laplacian of Gaussian features.

indicating large local change. This is done at multiple scales efficiently via a Laplacian pyramid. The next step of SURF is generally to build a descriptor around each keypoint by summing the Haar wavelet responses that describe the intensity distribution in the neighborhood of a point, but only the keypoint extraction step is of interest here.

Gilles keypoints are based on the idea that complexity in natural images is rare and so locations of high complexity are salient features that can be extracted as keypoints. Gilles uses the Shannon entropy of the grayscale values in an image region to measure the complexity of that region. A high-complexity image region in this sense can also be understood as a region with a flatter grayscale histogram.

The Laplacian of Gaussian filter can be used to detect points where rapid intensity changes occur, such as edges. As one might expect, many points of rapid intensity change can be found in a scene of rocks.

2.2.2 TextureCam

TextureCam is software developed at Jet Propulsion Laboratory for the purpose of automatically classifying geologic features. It can be used as a rock detector by training it to distinguish between manually labeled ground and rock class regions on an image. Figure 5 shows examples of training images, their labels, and the output probability heatmaps of TextureCam on test images. In the training images in the middle column, red is drawn over pixels that should be classified as "ground" and blue is drawn over pixels that should be classified as "rock." In the probability heat maps in the right column, yellow pixels have a higher probability of being in the "rock" class.

TextureCam works well in distinguishing high-contrast rocks, especially with uniform backgrounds. It tends to find visually distinct features, such as dark cracks or shadows. Shadows and sparser scenes are more likely to be mislabeled. Some failure cases are shown in Figure 6. Here, the pixels with the top 5% highest probabilities are shown in red. The three causes of failure shown are the rover's own shadow, cracks in the ground, and the rover tracks.

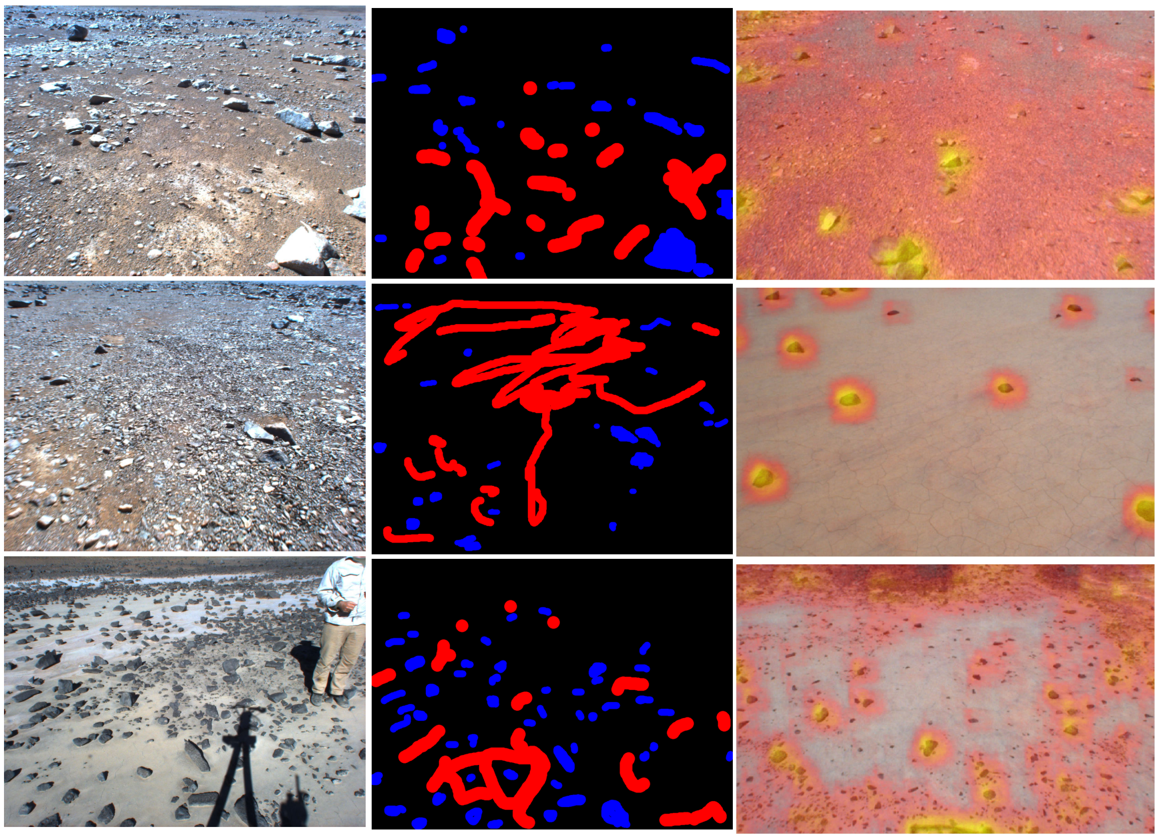

Figure 5: TextureCam images, training labels and rock probability heat maps. In the training labels, rock pixels are colored blue and ground pixels are colored red. In the heat maps, yellow pixels indicate higher probability of rock classification.

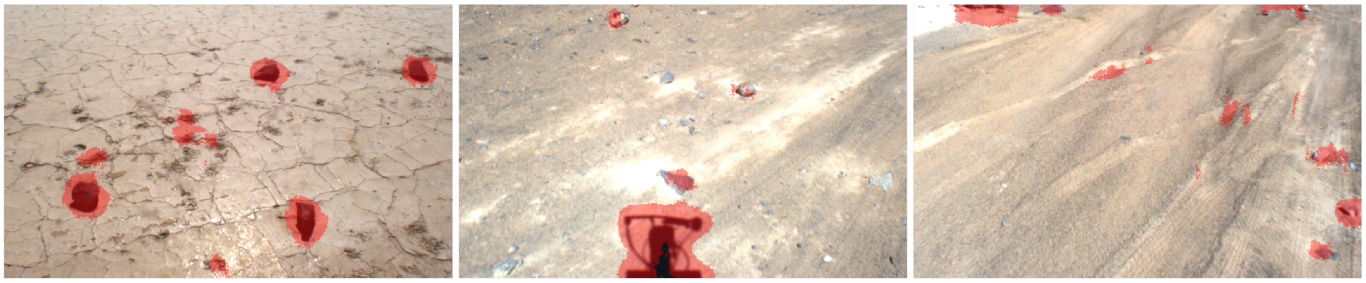

Figure 6: TextureCam results with failure cases in the top 5% most confident rock classifications.

2.2.3 Relative Height of Clustered Superpixels (RHoCS)

Above we showed that keypoint descriptors tend to focus on rock edges without identifying rock boundaries, and TextureCam can identify likely rock regions but fails on darker ground regions. To fix these problems, we developed a method of identifying edge boundaries and avoiding dark regions on the ground by using the Relative Height of Clustered Superpixels (RHoCS). It is based on the idea that the two characteristics that most reliably distinguish rocks are their edge boundaries and their height above their surrounding ground.

RHoCS can be thought of as an expansion of the SQUID algorithm described in Section 2.1. While the SQUID algorithm developed for synthetic images captures the intuitive idea of a rock as a contiguous region of similar pixel intensities, it fails on more ambiguous real-world images. Real-world desert images contain shadows and colored patches of ground, both of which satisfy the continguous pixel intensity criterion. RHoCS can handle these cases by incorporating stereo camera information.

First, superpixels are computed to form boundaries and identify regions of similar color. In this work, supervised linear iterative clustering (SLIC) [2] was chosen for its computational speed as the method to generate superpixels. SLIC is similar to k-means clustering with two main differences: only cluster centers within some local region are searched to maintain semi-regular regions of pixels as clusters, and distance comparisons consider both distance in the Euclidean space and distance in the L*a*b* colorspace.

The superpixels are clustered using density-based spatial clustering of applications with noise (DBSCAN) [19]. DBSCAN was primarily chosen as the clustering method for its robustness to outliers and its reliance on only two intuitive parameters – the minimum points needed to form a cluster and the maximum distance of separation between pixels belonging to the same cluster. DBSCAN combines mutually similar superpixels into larger clusters. A size filter removes trivial clusters or clusters that would represent the sky or ground, leaving only rock candidates outlined.

To find the height of each rock candidate, a ground plane is fit to each rock candidate's local region. Here, Random Sample Consensus (RANSAC) [20] is used on the 3-D point clouds to fit the ground plane, and the region of interest used is the bounding box of the rock expanded to three times its width and length. If the mean height of the 3-D points within a rock candidate region is greater than some threshold τ_{rock} above its local ground plane, the candidate is identified as a rock.

Algorithm 1 provides more details of the method. Figure 7 shows the steps visually for two scenes. Note that spurious rock candidates can be picked up in the second step that isolates superpixels. This is because non-rock objects can be uniformly similar in color. In desert environments, the only such spurious rock candidates that can be expected are shadows or discolorations on the ground. Both of these are flat and thus ideal for being filtered out by a local height map. Setting a higher minimum height threshold increases precision at the cost of recall.

The parameters used by RHoCS are:

m	Relative weighting between spatial proximity and color proximity used by SLIC. Lower values cause superpixels to adhere more strongly to edges but increase computational cost.
k	Approximate number of superpixels to generate in SLIC.
tol	Distance threshold in L*a*b* units that determines which superpixels should be merged in DBSCAN.
τ_{rock}	Difference between the mean height of a rock candidate and its ground plane required to label it a rock.
$inlier\ thresh$	RANSAC threshold used to determine number of inliers at each iterations.

Algorithm 1 Pseudocode for RHoCS

```
/* Get superpixel label image */
$Sp$ = SLIC(image, $m$, $k$)

/* Get clustered label image */
$C$ = DBSCAN($Sp$, $tol$)

/* Check that each cluster meets height threshold */
For cluster $k$ in $C$
  $p_k$ = get_3d_points($k$, left_image, right_image)
  $r_k$ = ransac_fit_plane($p_k$, $inlier\_thresh$)
  $h_k$ = height_above_plane($k, r_k$)

  If $h_k$ > $\tau_{rock}$
    $k$.isRock = TRUE
  EndIf
EndFor
```

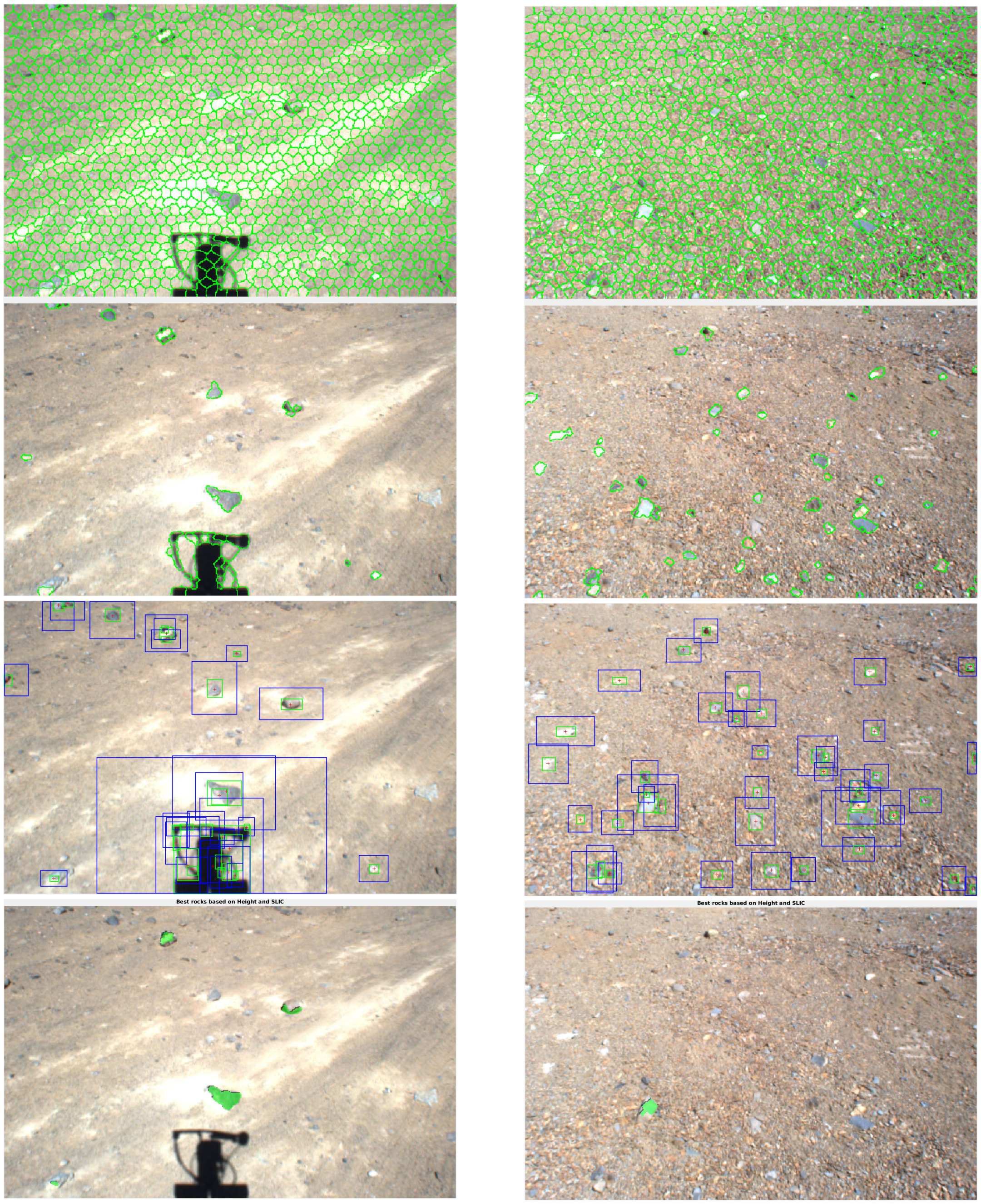
Best rocks based on Height and SLIC
Best rocks based on Height and SLIC

Figure 7: Overview of RHoCS algorithm. The top image shows SLIC superpixels marked. The second image shows the clustering of these superpixels to form the initial rock candidates. The third image show the bounding boxes (expanded three times) around each rock candidate, which are used to fit a ground plane and determine local pixel heights. The bottom image shows the rocks that meet the height threshold set by the user. As desired, shadows do not meet the height threshold.

2.3 Results

To determine the accuracy of RHoCS, the output labels can be compared against a set of ground truth images with rocks manually identified.

2.3.1 Ground Truth

Collecting ground truth labels for rocks, especially at the pixel level, is a difficult task. This is largely due to the ambiguous nature of rocks in most scenes. Whereas rocks of a certain size may be worth noting in some sparse scenes, in other scenes rocks of that same size can be found covering the ground. Compared to many common semantic segmentation tasks, e.g. identifying roads, trees, or cats, the identification of rocks is highly subjective.

For that reason, instead of comparing the output of RHoCS against a set of images with all rocks labeled, it is compared against a set of images with only a few highly distinguishable rocks marked. Specifically, the results here give the recall of the algorithm in identifying the centroids of rocks most humans would identify in a scene. This is useful for two reasons: (1) the centroids of the rocks will be used to form constellations and (2) the highly distinguishable rocks are the most useful in forming recognizable constellations.

Since the terrain varies significantly throughout the Atacama Desert, four sets of representative ground truth images will be considered. Centroids are manually marked. The four image sets are labeled **A**, **B**, **C**, and **D**. Image sets **A** and **B** both contain 30 images. Set **C** contains 11 images and set **D** contains 60 images. All images are of size 640x480 pixels.

2.3.2 Recall Results

Example outputs from the RHoCS algorithm and ground truth centroids are shown in Figures 8 - 11 for the four image sets. The overall recall results from these scenes is given in Table 1. These values are the percentage of the labeled ground truth centroids detected by RHoCS. The specific parameters used to obtain these results are given in Table 2.

Table 1: Recall results of RHoCS

Image Set	RHoCS recall
A	75.0%
B	62.4%
C	68.1%
D	52.3%
Avg	64.5%

Table 2: RHoCS parameters

Parameter	Value
m	10
k	1500
tol	10
τ_{rock}	0.04 (m)
$inlier\_thresh$	0.025 (m)

2.4 Discussion

The results in Figures 8 through 11 illustrate some of the strengths and weaknesses of the RHoCS algorithm. It performs best in image sets **A** and **C** where scenes are uniformly flat, and rocks

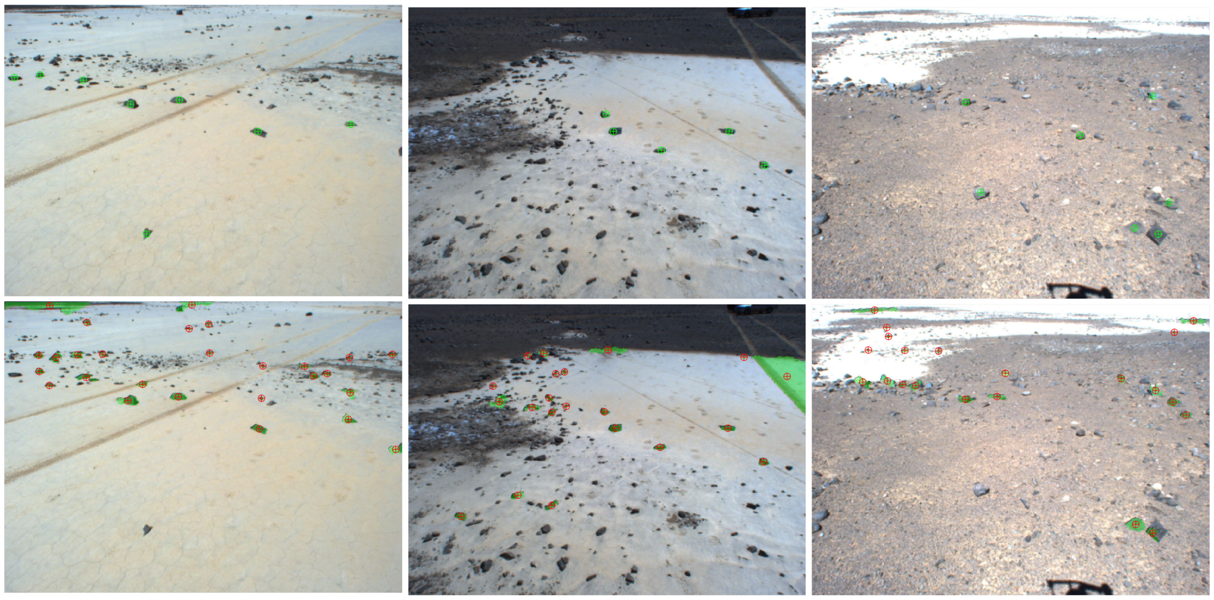

Figure 8: Image set **A**: Top row, ground truth centroids in green, bottom row, detected rocks shaded green with RHoCS-computed centroids in red.

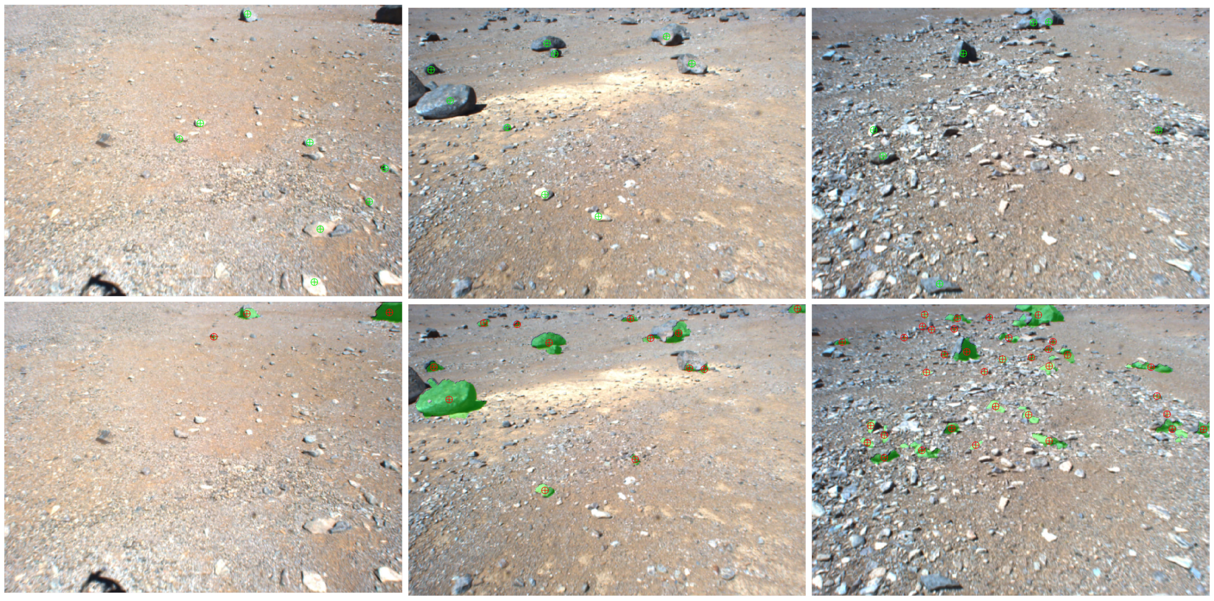

Figure 9: Image set **B**: Top row, ground truth centroids in green, bottom row, detected rocks shaded green with RHoCS-computed centroids in red.

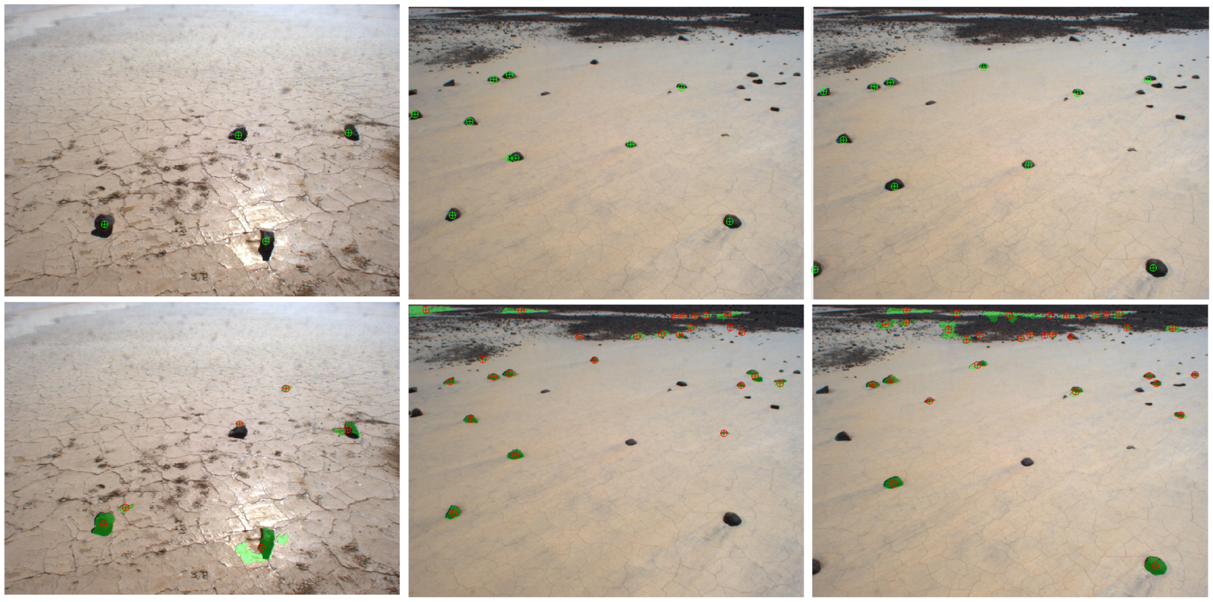
Figure 10: Image set **C**: Top row, ground truth centroids in green, bottom row, detected rocks shaded green with RHoCS-computed centroids in red.

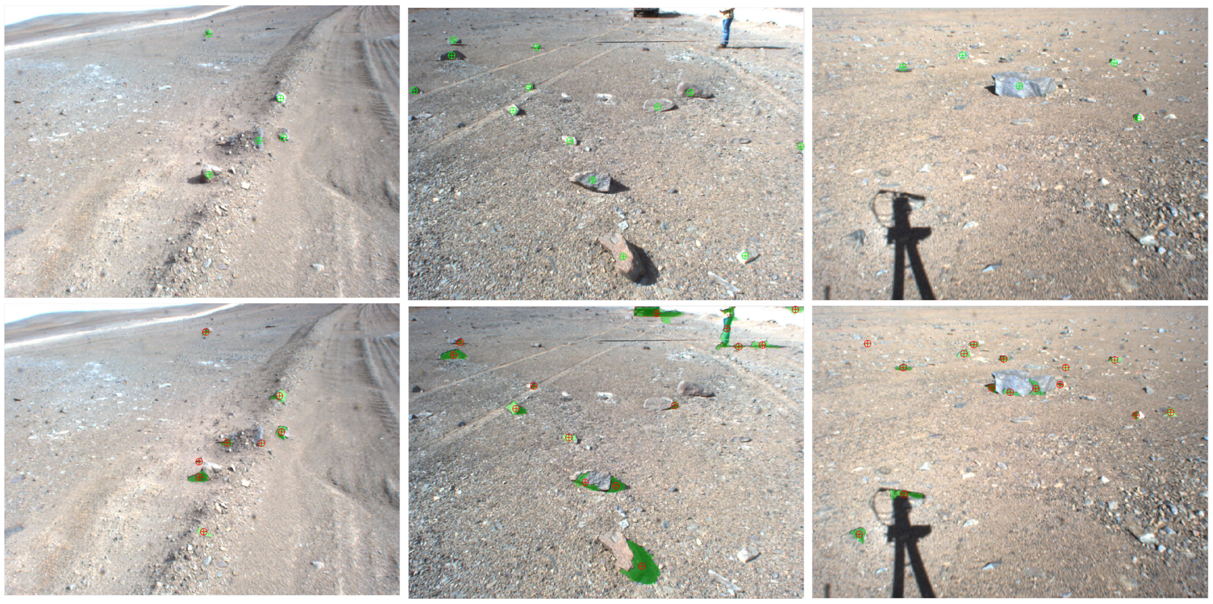
Figure 11: Image set **D**: Top row, ground truth centroids in green, bottom row, detected rocks shaded green with RHoCS-computed centroids in red.

are uniformly colored and topographically prominent. There are four notable scenarios in which RHoCS performs poorly, each of which breaks an assumption of RHoCS:

Uneven surface	If the local ground around a rock varies, then an accurate ground plane may not be fit, and the resulting height measurement of the rock will be incorrect.
Many nearby rocks	If multiple rocks are near enough to be within each others' local region then neither rock may be considered prominent enough to be identified as a rock. More specifically, the ground plane around each rock would include the other rock.
Rocks with varied colors	If a rock is varied enough that multiple regions of the same rock form their own superpixels and DBSCAN clusters, a ground plane will be fit around each of these parts of the rock. The local region around each rock candidate could then include other parts of the same rock.
Non-rock objects	RHoCS assumes the only contiguous regions of similar color above the ground will be rocks. Other non-rock objects such as people and cars can have these characteristics and would be identified as rocks.

Image set **D** (Figure 11) in particular shows some of these difficulties. These images are collected around a road, where clumps of dirt make the terrain uneven. The variance in height of some parts of the ground exceeds the height of rocks of interest. Set **D** also shows that trucks and people can be identified as rocks, as they have similarly colored regions above the ground. Large rocks pose difficulties because of the high chance of containing different-colored regions. Large rocks create shadows or have a shadowed side, and each of these regions will be considered as a separate rock candidate, often with a poorly-fit ground planes.

For the RHoCS outputs shown above, the same height threshold was used for all scenes (4 cm). A possible avenue for future exploration is to adapt the threshold based on the size or number of rocks in each scene.

In terms of computation time, there are three main considerations: SLIC, DBSCAN, and RANSAC. SLIC has linear computational complexity $O(N)$ for N pixels and DBSCAN has complexity $O(k^2)$ for k superpixels. In practice, SLIC is the main bottleneck, taking upwards of 10 seconds in MATLAB. DBSCAN and 100 iterations of RANSAC take approximately 1 second each. The CPU used for timing comparisons is an Intel Xeon E5-1620 v3 at 3.50 GHz.

A compiled language will undoubtedly produce much faster results, and incorporating hardware optimization may provide further speedups. For example, the Oxford Active Vision Library provides a CUDA-based GPU implementation of SLIC called "gSLICr" with a reported 3.4 ms runtime on a single 640x480 image with a GTX Titan X GPU [52]. However, even without computational concerns, the issues with RHoCS discussed above indicate that a more robust approach to rock detection should be considered.

3 Rock Detection with CNNs

In this section we approach rock detection from a machine learning perspective to mitigate some of the failure cases discussed in the previous section.

Convolutional Neural Networks (CNNs) have been under continued study since the 80's [38], but only in recent years with the advent of powerful commercial GPUs and a better understanding of neural network architecture has there been a resurgence in their use. Now, CNNs have been shown to excel, and are often shown to be state-of-the-art, in various image recognition and image understanding tasks. This will be the first work to use them in the context of rock detection.

The advantage of using a CNN for rock detection over other computer vision methods is its generalizability and computational speed. Once the CNN has been trained on a large and varied enough dataset, it could theoretically detect rocks in any lighting or terrain. Instead of a user teaching a system and iteratively testing various image features to determine their effectiveness, a CNN can identify the ideal rock detection parameters on its own from the raw pixels. This can also be potentially disadvantageous as well, since it makes it difficult to determine which features the CNN has learned and why it might be failing in some cases.

With regard to computational speed, the prior RHoCS detection method can produce suitable results but relies heavily on CPU time. The RHoCS algorithm was not tested specifically on mission hardware or in a compiled language but it is reasonable to assume that the highly parallelizable series of matrix multiplications used by a CNN could be better hardware-optimized. For a CNN, the computation time is front-loaded during training but, once complete, the resulting prediction model can be quickly executed on an image.

3.1 Background and Related Work

Artificial neural networks and multilayer perceptrons were inspired by biological neural networks — series of interconnected neurons that fire given various stimuli. In other words, a biological neural network has "learned" to produce a response given a specific stimulus. The same idea applies to artificial neural networks, which are trained to produce a response for specific inputs. The convolutional neural network was developed initially to expand the multilayer perceptron to work efficiently with images, a type of input which has a large number of starting nodes (pixels).

The resurgence of CNN's primarily began in 2012 with Krizhevsky, Sutskeer, and Hinton [36], who used a large CNN to classify 1.2 million images in the ImageNet database into 1000 different classes. The result beat the previous state-of-the-art on ImageNet recognition and relied on two GTX 580 3GB GPUs. The work also introduced a new network architecture and various intermediate layers, some of which are still used today.

In 2014, Simonyan and Zisserman [57] methodically tested various ConvNet configurations and achieved first and second place results in the localization and classification tasks in the ImageNet Challenge 2014. They focused on consistently using small 3x3 convolutional filters and increasing the number of convolutional layers, or network depth. Simonyan and Zisserman also showed that their best configurations, using small 3x3 filters and 16 layers, generalized well to many other classification tasks and databases, including Caltech-101, Caltech-256, PASCAL

VOC-2012 action classification. This 16 layer network called VGG-16 will be used as the final model for rock detection in this work.

CNN architecture and the types of layers used will only briefly be discussed here, as they have been thoroughly described in many other papers [36, 39]. The architecture of a CNN can vary significantly between models, but the layers generally used to reduce an input image to a label or a vector of labels are convolutional layers, fully connected layers, and max pooling layers. A nonlinear activation function is used between layers, and is analagous to the activation energy required for a biological neuron to fire. The most commonly used nonlinear activation function is the rectified linear unit (ReLU), defined as $f(x) = \max(0, x)$. These activation functions are crucial as they allow the CNN to learn a nonlinear function of its inputs.

In the convolutional layer, N kernels of varying sizes $k \times k$ are convolved with each of the M previous layers to produce a new set of layers. In the max pooling step, the max values of a 2x2 or 3x3 window slid across each layer is used to produce a new set of reduced layers. A fully connected layer connects each of the neurons of the previous layer to each neuron of a smaller single-width layer. It can be thought of as a convolutional layer with a kernel of size 1×1 and is similar to a layer of a multilayer perceptron.

3.1.1 Software background

Three different CNN frameworks – CN24 [6], Caffe [30], MatConvNet [67] – were tested. While results should theoretically be similar, in practice difficulties arose. These three CNN frameworks were chosen due to their capability of assigning a label to each pixel that identifies it as either “rock” or “ground.” This is commonly called pixel-wise labeling or semantic segmentation.

The first training set contained approximately 10,000 images. As it was infeasible to label this data by hand, these images were instead labeled by the RHoCS method. While slow, the RHoCS method has enough precision to generate the training labels for the CNN. The CNN can then generalize the results to produce an accurate rock detector. Because a separate detection algorithm was used to label this training set, it is not an ideal ground truth and does produce some false positives and false negatives.

The second training set is composed of 60 images labeled using LabelMe [55] and augmented to a dataset of 1800 images with a series of affine transformations.

Caffe [30] A general purpose deep learning framework. While some have had success with semantic segmentation using Caffe [43], it was used here primarily for scene classification.

CN24 [5] A software framework designed specifically for pixel-wise labeling. It allows the user to design a convolutional neural net architecture, then uses a “convolutional patch network” to produce the semantic segmentation [6]. The convolutional patch network is trained to identify each pixel’s label by the patch of pixels around it.

The convolutional patch network can take advantage of prior spatial knowledge of labeled pixels. For example, a road in a scene tends to show up more commonly in certain parts of the image and that prior probability can improve the CNN results. However, for a typical desert scene, the positions of rocks in a scene are randomly distributed, so prior spatial information cannot be used.

MatConvNet [68] A MATLAB toolbox for implementing CNNs. It provides a VGG-16 model and the primary CNN tools used in the final iterations of this work.

3.2 Classifying Terrain Types with Caffe

One of the provided models in Caffe, CaffeNet[16], was trained on the 10,000 image Atacama dataset. For each scene, CaffeNet produced a 1000 element vector. K-means clustering was used on these 1000 element vectors to produce 20 cluster means, each of which effectively represents a terrain type. This same method can then be used to produce a feature vector to represent any new desert image, and the nearest cluster center to that vector, in terms of cosine similarity, can be used to determine its terrain classification.

While no ground truth terrain classifications were available, the classification qualitatively appears to work well. Figure 12 shows 20 randomly selected images that were classified to four different terrain types. This classifcation is not used in the final results of this work, but it is shown here as a possible venue for future exploration. It indicates that CNNs are capable of parsing out unique features from desert scenes.

One potential use for terrain classification is as a pre-processing check. A rover traveling in the Atacama Desert, for example, would pass through these 20 varied terrain types, and could run its images periodically through the CNN classifier to determine when the terrain type has shifted. This could provide a pose constraint if prior knowledge of the terrain was known, or could allow the rover to switch to a new rock detection algorithm tuned for that terrain type, enabling the use of a CNN ensemble.

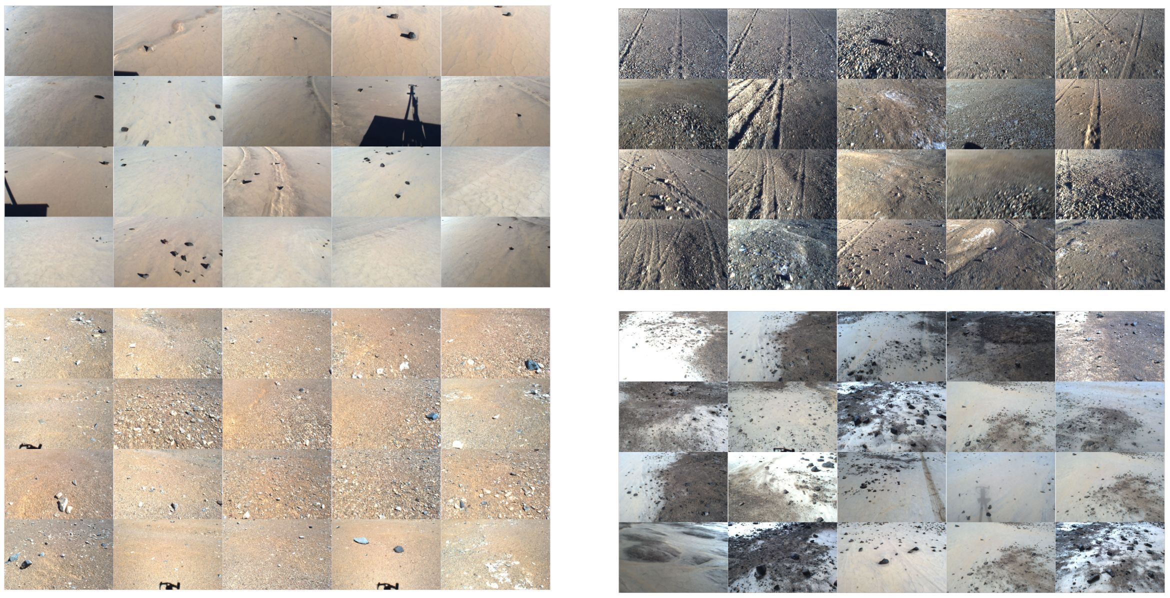

Figure 12: Random images selected from four terrain type clusters.

3.3 Rock Detection with CN24

Rock detection was first attempted with the CN24 framework and a convolutional patch network (CPN), which applies a CNN to patches of pixels at a time. Most of these trials were unsuccessful, but are still provided here for completeness, as the failure cases give some insight into the training data and CNN architecture necessary for successful rock detection.

Figure 13 shows the result of applying various CPN models to a simple synthetic image. One of the original images is shown on the left. The training and test set were produced by applying random affine augmentations to the original synthetic images. Results of the trained model on three test images are shown in the right column.

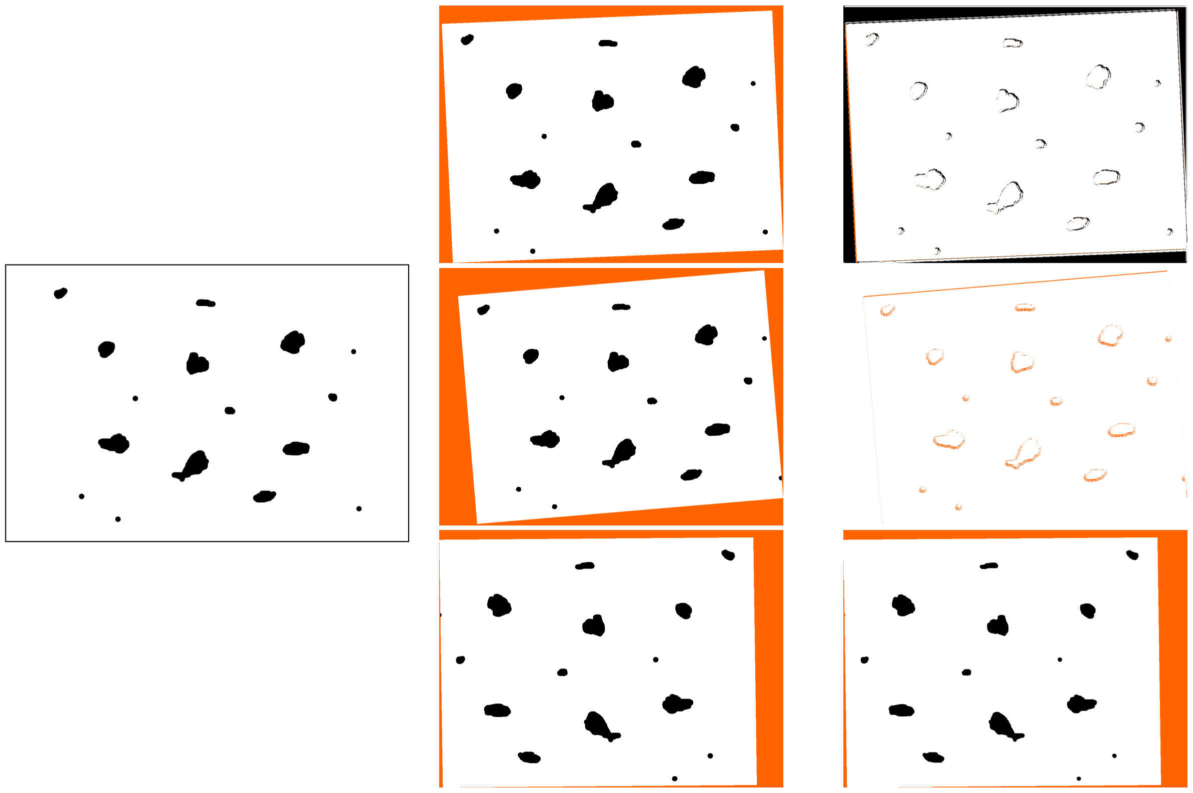

Figure 13: Simple synthetic rock images to test various CPN architectures. Left shows original image, middle shows affine augmentation inputs, and right shows the resulting labels for three different CPN architectures. As demonstrated by the poor results at the top and middle images of the right column, even simple synthetic images require some tuning of the CPN architecture.

The first result uses one convolutional layer with 12 kernels of size 13x13, one fully connected layer with 120 neurons, and a final fully connected layer that will always have 1 neuron for the binary classification of a single pixel. This architecture can be denoted more concisely as $C(12-13\text{x}13) \rightarrow FC(120) \rightarrow FC(1)$. The second result uses the architecture $C(12-7\text{x}7) \rightarrow C(6-5\text{x}5) \rightarrow C(48-5\text{x}5) \rightarrow FC(192) \rightarrow FC(3)$. The third result, which produces the correct labels, uses the architecture $C(3-5\text{x}5) \rightarrow C(3-5\text{x}5) \rightarrow FC(20) \rightarrow FC(3)$. The ReLU activation functions between each layer are left out for brevity. One key factor found to improve labeling on this type of synthetic image is the use of small kernel sizes of about size 5x5 in the convolutional layer.

A simple real desert scene is used to test the system next. It is a high-contrast, low-density scenes, which makes it easy to generate an accurate ground truth label via adaptive thresholding. Since there were not enough of these types of images to train a CNN, random affine transformations were again used to artificially augment the dataset.

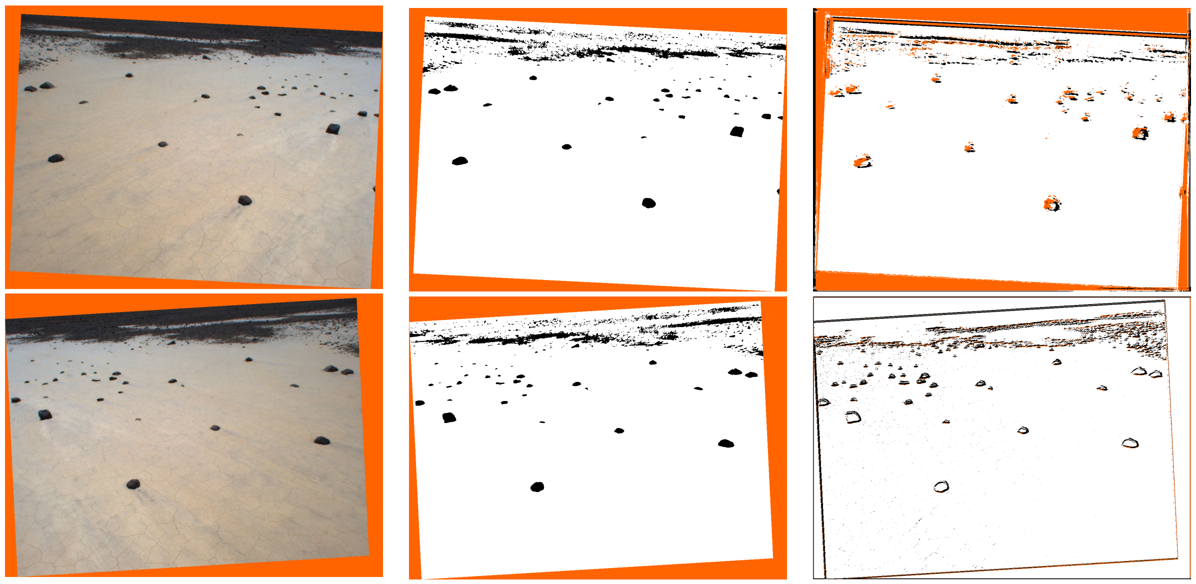

Figure 14: Convolutional patch network results on a simple desert scene. Left shows affine transformed inputs, middle shows ground truth, right shows CPN results.

The results in Figure 14 were produced with the architectures $C(3-5\text{x}5) \rightarrow C(3-5\text{x}5) \rightarrow FC(20) \rightarrow FC(3)$ and $C(12-35\text{x}35) \rightarrow FC(20) \rightarrow FC(3)$, but neither proved successful. Various other architectures were tested, but these also had difficulty with a real desert scene, even using this high-contrast scene. A possible cause is the lack of distinguishable features in the image and the inability to take advantage of spatial priors.

To determine if a different set of inputs might improve results, a CPN was trained on the raw RGB images of the Atacama scenes as well as on variations of the 3-channel input, including every combination of hue, saturation, brightness, disparity, and illumination-invariant images. Some example inputs are shown in Figure 15, with the typical labels produced.

These results indicate that the input channel layer is ineffective at improving rock label accuracy, often introducing noise that decreases accuracy. For example, parts of the image that are more reflective or have increased illumination will become more prominent in the brightness channel, which is not a desirable effect. Disparity can cause large portions of the image to stand out due to the angle or position from which the rover obtained the image.

This lack of improvement is intuitively reasonable as well, since colorspace transformations alone do not introduce new information for the CNN. The transformation required to go from RGB from HSV, for example, could be learned by a CNN if it were truly useful to make that colorspace transformation.

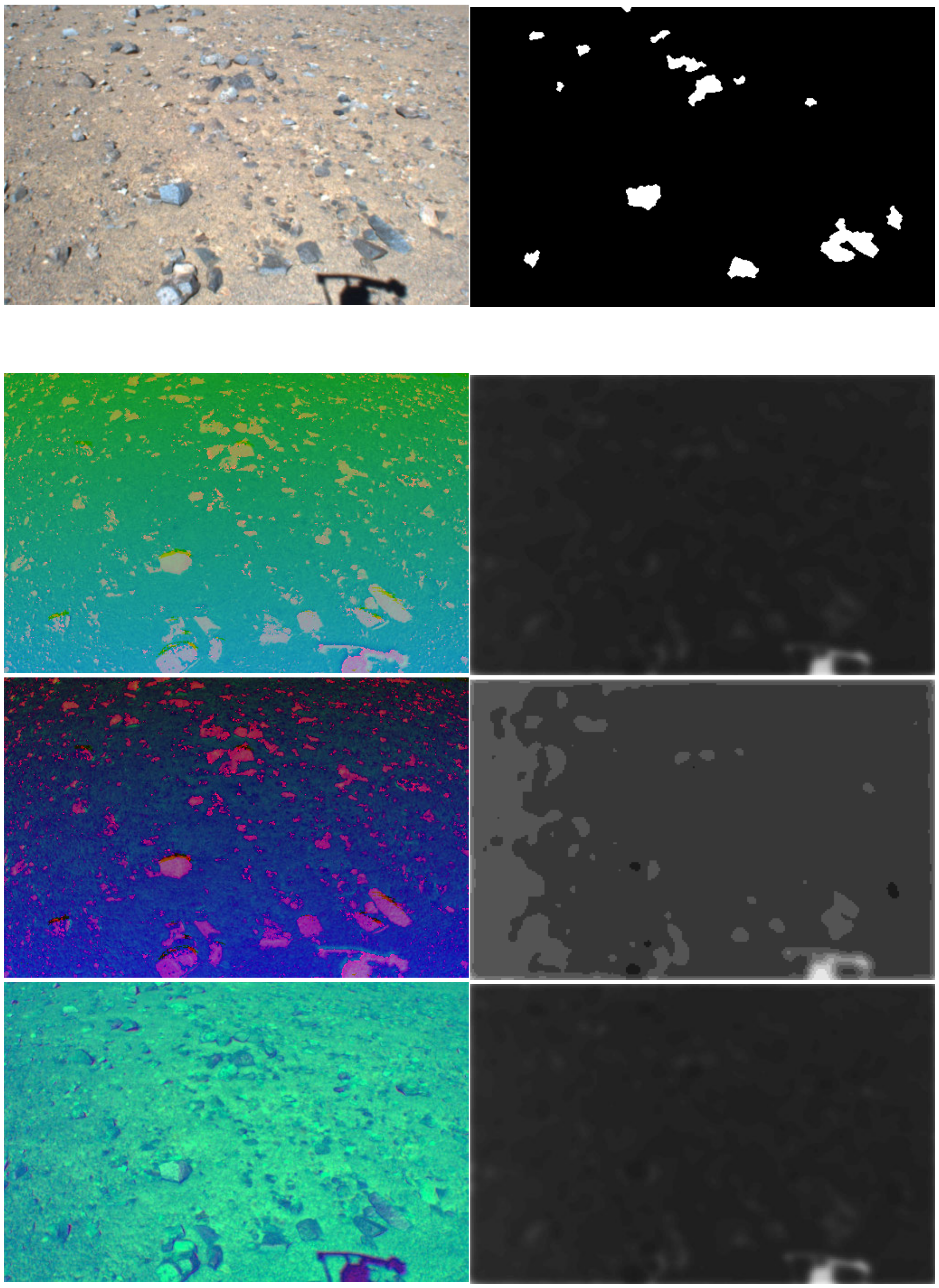

Figure 15: The left column shows, from top to bottom, the original RGB image, hue-illumination-invariant-disparity image, hue-saturation-disparity image, and saturation-value-illumination-invariant image. The right column shows, from top to bottom, the ground truth label from RHoCS, followed by the CPN label outputs.

The training image sets used thus far were automatically labeled by the RHoCS algorithm described in Section 2.2.3. Because of this, the ground truth labels used to train the CNN above were not perfectly accurate. Additionally, the training sets sometimes contained objects or features that were neither rocks nor ground. To eliminate the possibility that inadequate ground truth labels were being used to train the CNN, a hand-labeled, 8-class dataset was created. The LabelMe Toolbox was used to label this set of 60 images[13, 55]. The 8 classes defined were rock, ground, sky, mountain, human, car, shadow, and road. A set of 30 affine transformations were applied to each of the 60 images to produce a 1800 image training set. Figure 16 shows some examples of 8-class ground truth label images.

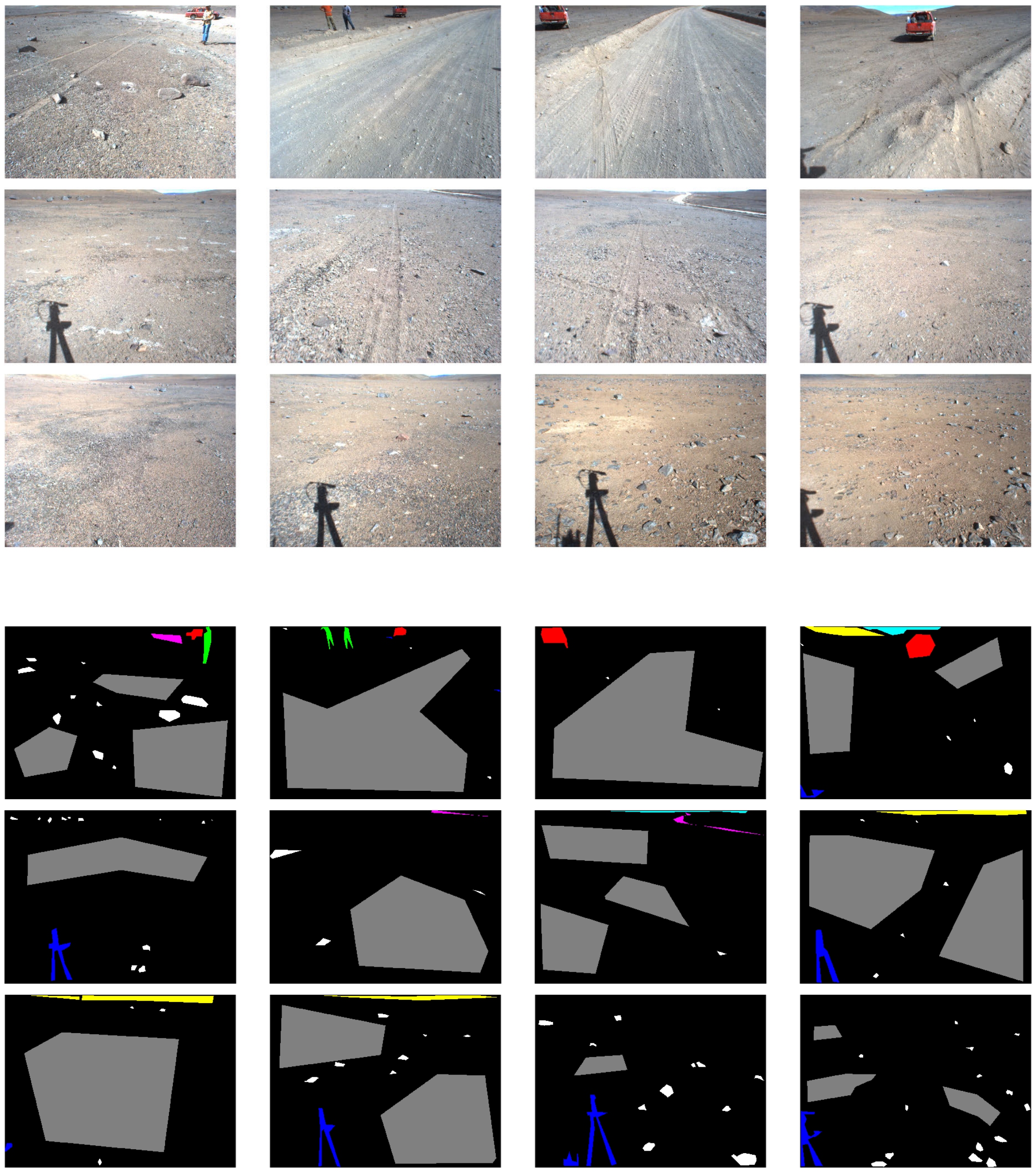

Figure 16: Examples of 8-class training set images and their corresponding hand-labeled ground truth.

CPN results with this near-ideal training set looked similar to earlier results. Though the ground truth was hand-labeled, the distinction between rock and ground in some scenes is nebulous, and it is likely there was some inconsistency in the labels. Additionally, several of the 8 class labels show up only rarely in the images, and may not have been sufficiently learned. However, the most likely cause of failure is that convolutional patch networks are not suited for a rock detection task.

3.4 Rock Detection with MatConvNet

The final rock detection implementation uses MatConvNet and one of the models it provides, VGG-16. Using the notation described earlier, the VGG-16 model architecture is denoted:

$C(64-3\text{x}3) \rightarrow C(64-3\text{x}3) \rightarrow M \rightarrow C(128-3\text{x}3) \rightarrow C(128-3\text{x}3) \rightarrow M \rightarrow C(256-3\text{x}3) \rightarrow C(256-3\text{x}3) \rightarrow C(256-3\text{x}3) \rightarrow M \rightarrow C(512-3\text{x}3) \rightarrow C(512-3\text{x}3) \rightarrow C(512-3\text{x}3) \rightarrow M \rightarrow C(512-3\text{x}3) \rightarrow C(512-3\text{x}3) \rightarrow C(512-3\text{x}3) \rightarrow M \rightarrow FC(4096) \rightarrow FC(4096) \rightarrow FC(1000)$,

where M is a max pooling layer. Note that the model is much larger than the previous models because MatConvNet uses the standard CNN approach, which uses all the pixels of the image as the starting input as opposed to the small patches used by the convolutional patch network.

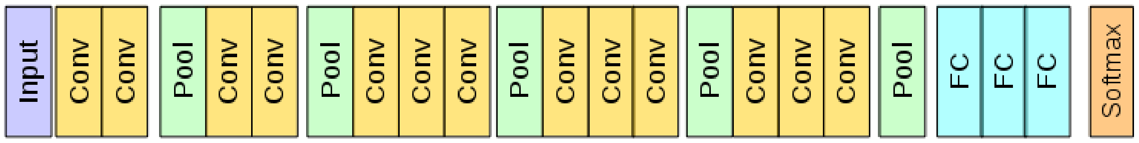

Figure 17: 16-layer VGGNet

3.4.1 Training a CNN for Rocks (RockNet)

A CNN using the 16-layer VGGNet was trained on an image set of 400 images augmented by 20 affine transformations to a set of 8,000 images. These transformations, shown in Figure 18, help to prevent overfitting and improve generalization of the rock detection. They are especially useful in this application where the goal of the rock detector is to detect the same rocks in a scene from any viewing angle and location. The resulting trained model will be referred to as RockNet in the following sections.

The “ground truth” labels were produced by the RHoCS algorithm described in Section 2.2.3, based on the hypothesis that the CNN would be capable of generalizing the high-precision, low-recall rock detection of RHoCS to produce a high-precision, high-recall rock detector that is also significantly faster.

The most important training parameter is the learning rate, which regulates the size of stochastic gradient descent steps when optimizing the CNN weights. If set too high, the stochastic gradient descent may never converge to an optimal value for the weights. A user should consider setting the learning rate low initially, and only raising it based on training time constraints. A learning rate factor of 0.0001 worked well for this training set.

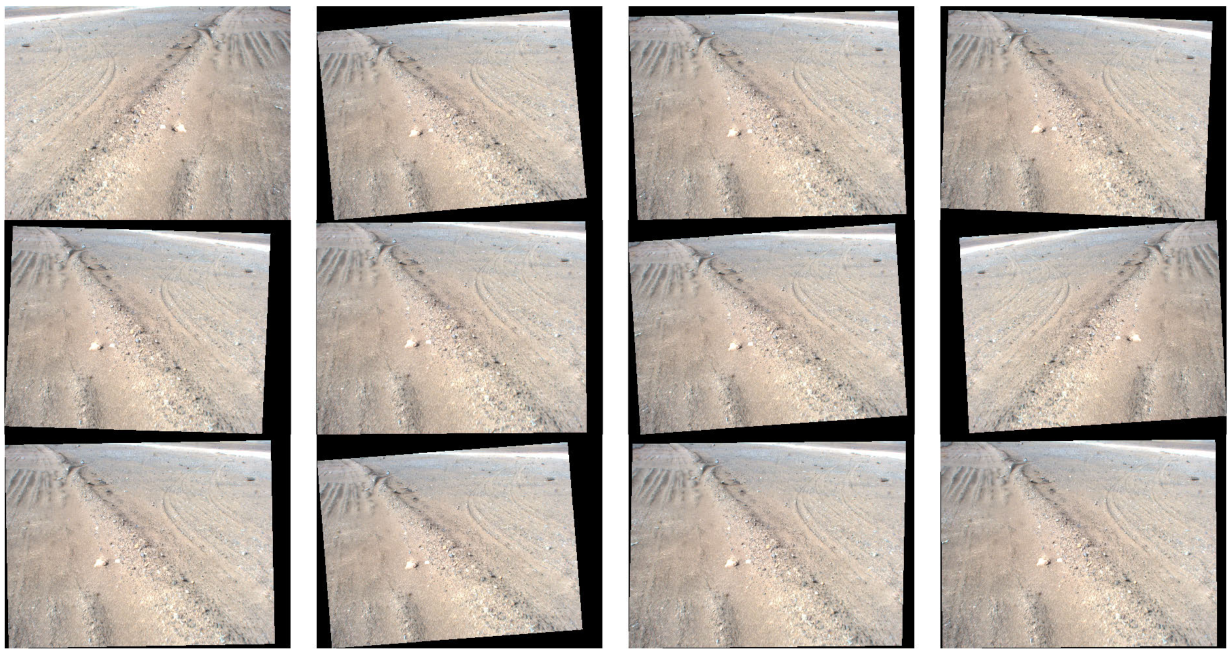
Figure 18: Affine transformations were used to augment the dataset.

3.5 Results

Recall results against a ground truth set of manually labeled images is provided below. The ground truth image sets are described previously in Section 2.3.1. As noted there, only the centroids of highly distinguishable rocks were labeled in the ground truth image sets, so many true positive rocks unlabeled in the ground truth may be correctly labeled in the CNN results.

3.5.1 Recall Results

Example outputs from RockNet and ground truth centroids are shown in Figures 8 - 11 for the four image sets. Table 3 shows the recall of RockNet — the percentage of ground truth centroids detected. The previous RHoCS results are shown again for comparison.

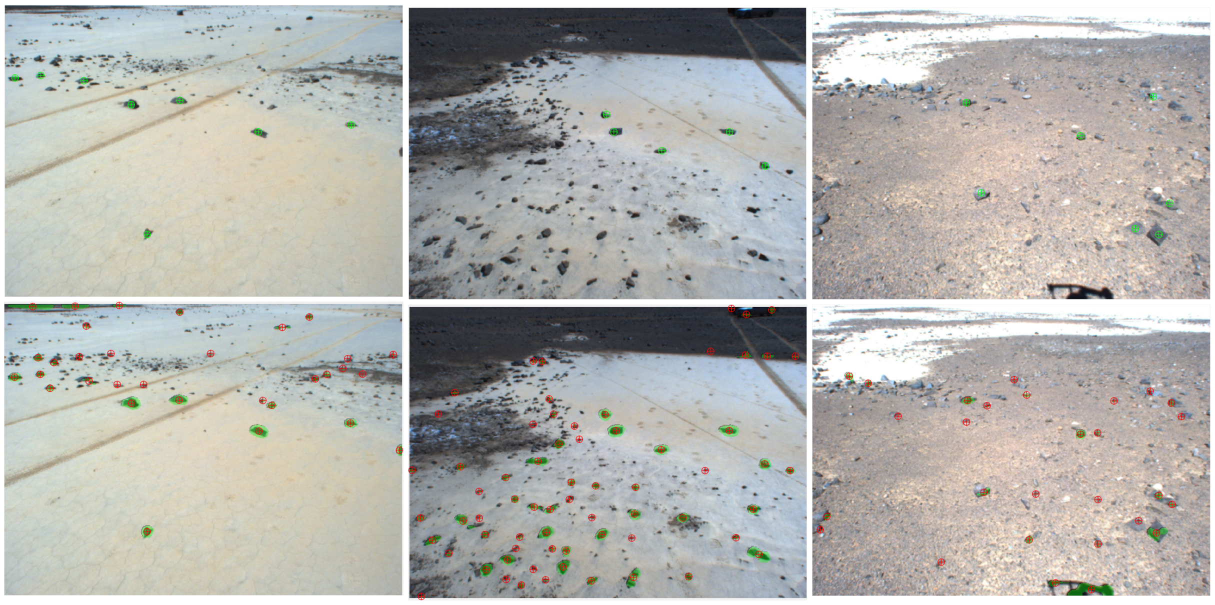

Figure 19: Image set **A**: Top row, ground truth centroids in green, bottom row, detected rocks shaded green with RockNet-computed centroids in red.

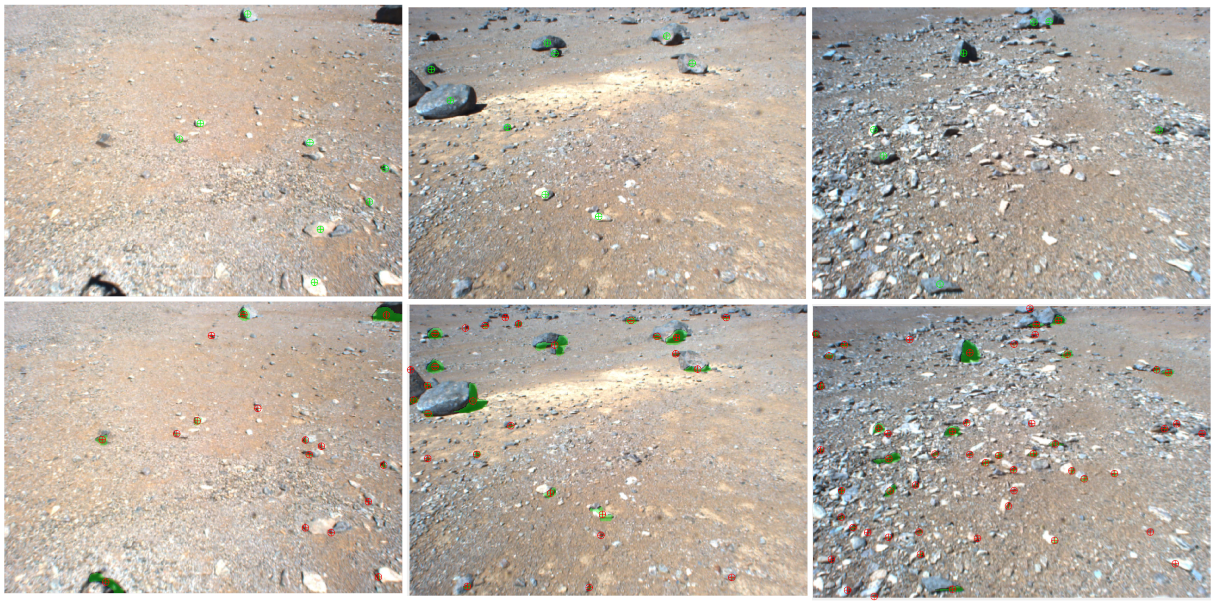

Figure 20: Image set **B**: Top row, ground truth centroids in green, bottom row, detected rocks shaded green with RockNet-computed centroids in red.

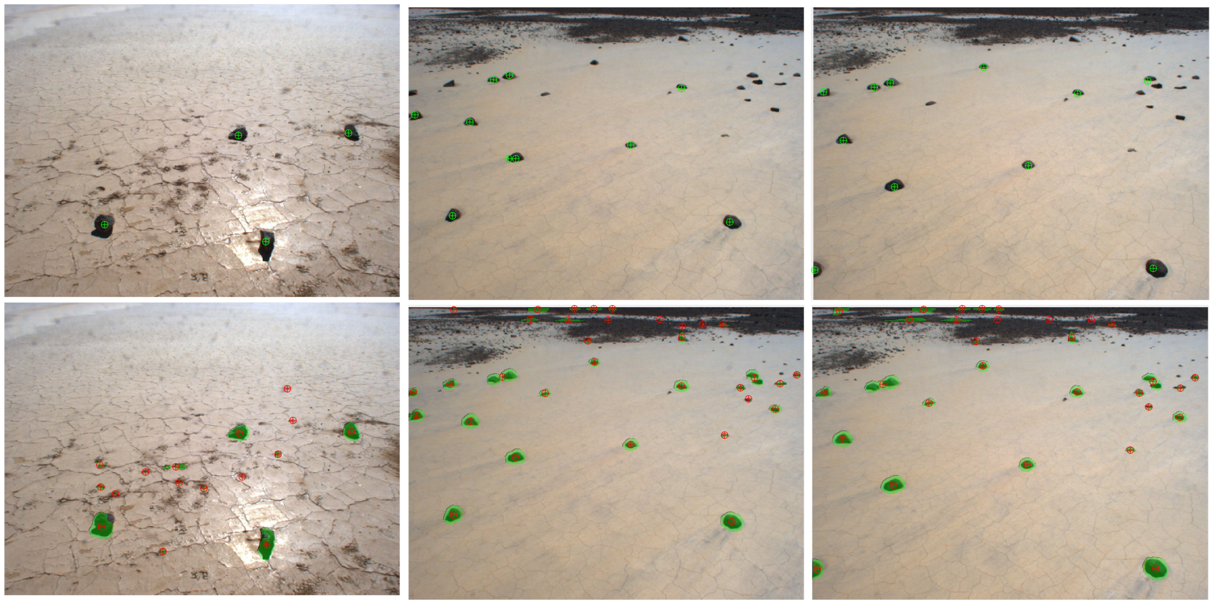
Figure 21: Image set **C**: Top row, ground truth centroids in green, bottom row, detected rocks shaded green with RockNet-computed centroids in red.

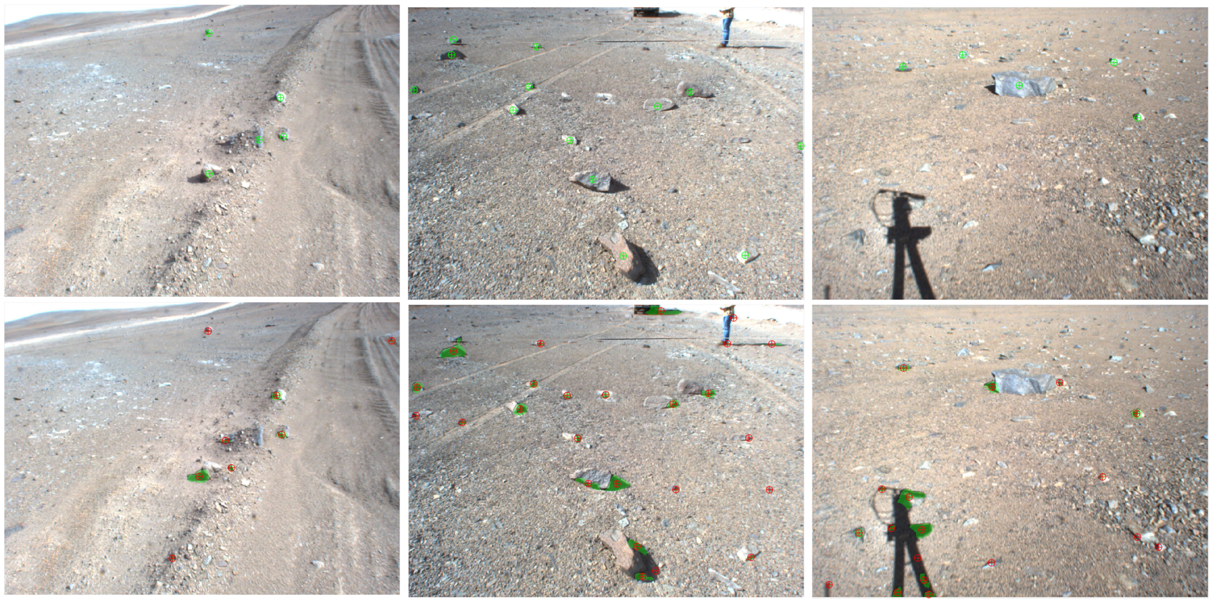
Figure 22: Image set **D**: Top row, ground truth centroids in green, bottom row, detected rocks shaded green with RockNet-computed centroids in red.

Table 3: Recall of RockNet on image sets **A** through **D**, with RHoCS result also shown for comparison**.**

Image Set	**RHoCS recall**	**RockNet recall**
A	75.0%	98.9%
B	62.4%	92.5%
C	68.1%	100%
D	52.3%	69.1%
Avg	64.5%	90.1%

3.6 Discussion

It is clear that in terms of recall, RockNet performs better than the RHoCS algorithm, averaging 90.1% recall across the four image sets. However, it is not an ideal rock detector, as might be expected from using a non-ideal training set. Some of the issues discussed in Section 2.4 can be seen here as well, since the RHoCS algorithm provided the training set. This could explain why some shadows are identified as rocks in image set **D**. Another possibility is that the CNN has over-generalized its training set. The training set consisted of images from the same regions as sets **A** and **B**, which averaged 95.7% recall compared to the 69.1% recall on image set **D**. Image set **C** showed high recall results regardless due to the simplicity of those scenes.

Note that the majority of false positives in the RockNet results consist of very few pixels, e.g. image 1 of Figure 21, and a simple filter on rock size could correct these.

To improve the CNN-based rock detector for future use, the training data should be thoroughly vetted, with images containing people, cars, and other anomalies removed. The training set labels could also be verified against an illumination-invariant set of images with shadows removed [12, 17, 45]. This would ensure that shadows are not passed to the CNN, while removing the need for a shadow-removal preprocessing step during online localization. To account for the possibility of over-generalizing rock features, a larger and more varied training set is also recommended.

With a few filters on rock size, however, the output of RockNet appears to be robust enough for constellation formation, which will be discussed in the following section.

4 Constellations as Landmarks

This section discusses the formation of collections of rocks as useful and identifiable landmarks, which we have termed *constellations*. The hypothesis that a constellation of rocks can be used as a more reliable landmark than an individual rock stems from two main ideas. The first is that individual rocks are difficult to distinguish from each other. This is an inherent property of rocks, as they are low-complexity objects formed from the same materials and by the same geologic processes. Secondly, even rocks that might appear distinct — in color, shape, or size — can be difficult to recognize from different viewpoints or under different illumination conditions. Rock constellations, on the other hand, allow one to take advantage of rock geometry, which is invariant to viewpoint and illumination, and more distinguishable in a desert landscape.

A k-constellation uses the unique geometry of k rocks in 3-D space as a landmark. For example, a 3-constellation would simply be a triangle. The sections below will discuss the methods of describing and comparing constellations. First, related work in star tracking is discussed. Then, in Section 4.2.1, several methods for generating constellations are presented, followed by methods of describing the geometry of those constellations in Section 4.2.2 and incorporating rock features in Section 4.2.3. Once the set of constellations in a scene is adequately described, the constellations can be matched to those of another scene to determine pose graph constraints, and various such matching methods are described in Section 4.2.4.

In the results of Section 4.3 that follow, the methods for generating, describing, and comparing constellations will be compared in a simulation analysis to determine which combination of methods provides high accuracy as well as high practicality. In particular, we will compare similarity metrics (4.3.1) and constellation descriptors (4.3.4), and vary constellation size (4.3.5), sensor noise (4.3.6), and the number of features per rock (4.3.7).

4.1 Related Work

To the best of this author's knowledge, there is no prior literature on using groups of rocks as landmarks. However, a similar problem of reliably matching groups of points occurs in identifying stars. Star identification (Star-ID) is most commonly used for satellite attitude determination [41], but can also be used on a spacecraft to estimate angular velocity [34] or to aid in navigating to a specific landing site [66].

The CCD image sensor used for star tracking varies, but typically has a field of view (FOV) of about 12x12 degrees and resolution of up to 500x500 pixels. A commonly used star database is the NASA Sky2000 Master Sky Catalog, which has position, magnitude, and spectral information for approximately 300,000 stars [48].

Early work in Star-ID in 1986 focused on triplets of stars and compared the inter-star angles of a triplet to a database to find matches [24]. The inter-star angle is the angle between two stars from the camera's perspective. This matching process was made more efficient by first filtering the search space by the area and sum of luminosities of each star triplet [56]. To reduce the need to search different permutations of triplets, the triplets are commonly sorted by luminosity. In 1997, Padgett proposed a new Star-ID algorithm that involved mapping a collection of stars onto the cells of a discrete grid, and searching for matches of the resulting grid pattern against a database [50]. In 2000, a neural network was applied to star matching. The inter-star angles and luminosities were used as inputs into a neural network that identified the stars [27]. This method was very fast but its reliance on more sophisticated parallel computing hardware compared to other methods makes it less practical on spacecraft that have limited power and require radiation-resistant hardware [60]. More modern methods continue to use the luminosities and either inter-star angles or interior angles of small groups of stars, and focus on improving database search efficiency [35, 71]. See [60] for further discussion of these various methods and the improvements in runtime and reliability each makes.

While the problems of matching groups of rocks and matching groups of stars nominally seem similar, several key differences are described below:

Stars can be compared against a known database or star catalog The most important distinction between Star-ID and rock constellation matching in the context of SLAM is that the database of stars is constant and known beforehand, while the database of rock constellations is unknown at the start and updated as the rover moves. The star database can be stored as a binary search tree or other efficient data structure, based on the feature extraction method to be used. The star database is also often filtered to a set of 5,000 - 20,000 stars by removing stars with lower magnitudes, uncertain color indexes, or other indications of unreliability [51, 71].

The ability to filter the database and determine beforehand the most uniquely identifiable groups of stars makes reliable star matching much more feasible. For example, the most reliable star groups can be pre-determined for each subregion of the sky based on a probability analysis of which groups of stars are most unique. Then, in any orientation, a spacecraft could detect one of these reliable star groups and know with high probability that the match is unique.

Star measurements are highly precise This is not to say that star detection itself is perfect — there are numerous sources of possible errors in star detection, including sunlight reflected by the spacecraft, exhaust gas from thrusters, optical device aberrations, and other celestial objects (e.g. planets, comets, satellites). However, the relative positions of stars that are detected in the FOV are known to a high degree of precision. In fact, when performing simulation analysis of Star-ID methods, Zhang [71] considers measurement noise with standard deviation of at most 1 pixel (0.024 degrees). Compare this to rock detection where a measured centroid of the same rock can vary as much as 0.5 meters depending on stereo measurement errors and the specific pixels of the rock detected.

Stars are distant, planar points A star is measured as a single planar point, so inter-star angle measurements of the same two stars from any position and orientation will produce the same result, with no possibility of occlusion. In contrast, rocks are 3-D objects viewed in this work from the elevated camera of a rover. Thus, rocks often occlude each other and measurement errors vary depending on the rocks' locations in the image. Since the exact center of a rock cannot be measured in an image, the size of the rock itself can cause measurement errors.

Stars have few features Using a CCD camera, the only additional feature available for each star is its luminosity. Rocks extracted from RGB images have a larger set of features available.

Star-ID is less prone to partial matches A partial match here refers to a match between two sets of points, in which one of the sets is a subset of the other. This is unlikely for Star-ID algorithms because the database fully describes each star. A partial match would need to be caused by extracting a partial set of features from the FOV, which is avoidable. Consider the following example of a typical Star-ID algorithm: a bright star in the FOV is selected to be matched and its two nearest neighbors will be used to form three inter-star angle features. The database has been set up so that each entry is the inter-star angles between a bright star and its two nearest neighbors. The features in the FOV are matched to the nearest database entry to identify the selected star. A partial match in this scenario could only occur if one of the nearest neighbors of the selected star was outside of the FOV, which could be entirely avoided by selecting a star away from the borders of the FOV.

In the rock constellation problem, there is no database to fully describe each rock so every constellation is potentially a partial constellation. Also, most rocks are viewable for 3-4 frames, so constellations are constantly being obscured as the rover moves.

Star-ID searches for a single match At each star tracker cycle, the star matching algorithm attempts to identify a single star in the FOV, which can be used to compute an attitude estimate. Every star above a desired magnitude will be stored in the database so a match is always guaranteed, provided no star detection errors. Rock constellation matching strives to identify as many landmark matches as possible, as each match improves the localization estimate. No constellations in a given scene are guaranteed to have a match.

With the precision of star detection, current Star-ID methods can achieve 98%-99% identification accuracy in simulations [70]. Star trackers are the primary source of attitude information in spacecraft today [61]. However, several key advantages — a known database, precise measurements, and single points — are lost in transferring the problem to rock constellations. Because of these differences, new approaches for rock constellation matching are investigated in the sections below, with a primary focus on handling partial matches and robustness to measurement and detection errors.

4.2 Method

There are three components to constellation matching between two scenes: (1) the generation of constellations (2) the choice of feature descriptor to describe each constellation and (3) the method of comparing feature descriptors. This section will first describe each of these components and their effects on matching accuracy and precision separately. Then, because they are not entirely independent, their combined effects will also be discussed. All descriptors of constellation shapes assume planar points, which is a reasonable assumption for desert images.

4.2.1 Generating Constellations

For the purpose of defining a rock constellation, each rock is first reduced to a single point. In this work, the rock detector produces a binary image, where rock pixels are labeled white and ground pixels are labeled black. A group of connected white pixels is assumed to be a single rock, and the centroid of those pixels defines the centroid of the rock. Thus, a scene of N rocks produces a set of N points which will be used as the input to a constellation generator. A k-constellation is defined as any set of k centroids in the scene, where $k \leq N$. Any means of producing a set of these k-constellations is a valid constellation builder. However, with the objective of accurate and precise constellation matching in mind, it is possible to judge the quality of a constellation builder based on several factors, namely:

1. The repeatability of constellation generation.
2. The number of constellations generated.
3. The robustness to small changes in the point set.

The first consideration is crucial to obtain any reasonable accuracy. Consider the extreme case of a random constellation builder. Presented with the exact same scene it will produce different constellations, making it unusable for data association.

The second consideration focuses on the computational aspect of comparing constellations. The maximum number of unique k-constellations that can be obtained from N points is $\binom{N}{k}$ which grows factorially to unmanageable numbers fairly quickly. For constellations greater than order 3 it is necessary to find a more efficient set of constellations.

The third consideration is of practical concern, as it is a common occurrence for rocks to be introduced into a scene as the rover moves. For example, a smaller rock can be added to the point set only when the rover moves near enough for it to be identified by the rock detection sensor and algorithm.

The constellation-building methods examined here are full 3-constellation generation, Delaunay triangulation, connected Delaunay triangulation, and convex hull boundaries.

Full Constellation Generation The naïve approach to k-constellation generation is to use every combination of k points. In practice, the Atacama dataset used in this work has scenes with at most $N = 40$ rocks. Generating every possible 4-constellation would produce $91,390$ constellations. But with this rock limit it is still possible to consider all 3-constellations, which will be explored in Section 4.3. The advantage of this method is that it is highly robust to small changes in the point set. If a point is added, the previous constellations will be unchanged. With ideal measurement accuracy, it will achieve perfect recall. However, this is at the cost of precision, as the chance of a false positive match is very high for a complete set of constellations.

Delaunay Triangulation A Delaunay triangulation is a triangulation of all the points P in the scene, such that each triangle's minimum angle is maximized. This gives it favorable features, such as avoiding sliver triangles. Given the same scene and Delaunay algorithm, the same Delaunay triangulation will always be produced. The number of constellations generated will be at most $2N - 5$, making it much more computationally feasible than the $\binom{N}{3}$ triangles of a full 3-constellation generation.

However, small changes to the point set can alter the triangulation. A point that appears within the circumcircle of a triangle will alter that triangle, and a single point can appear in

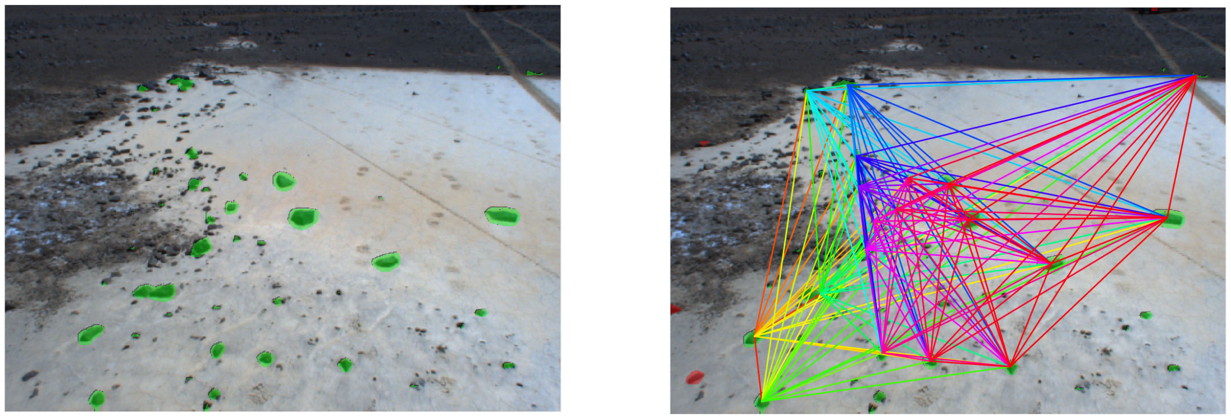

Figure 23: All 3-constellations of an Atacama Desert scene, each shown in a different color.

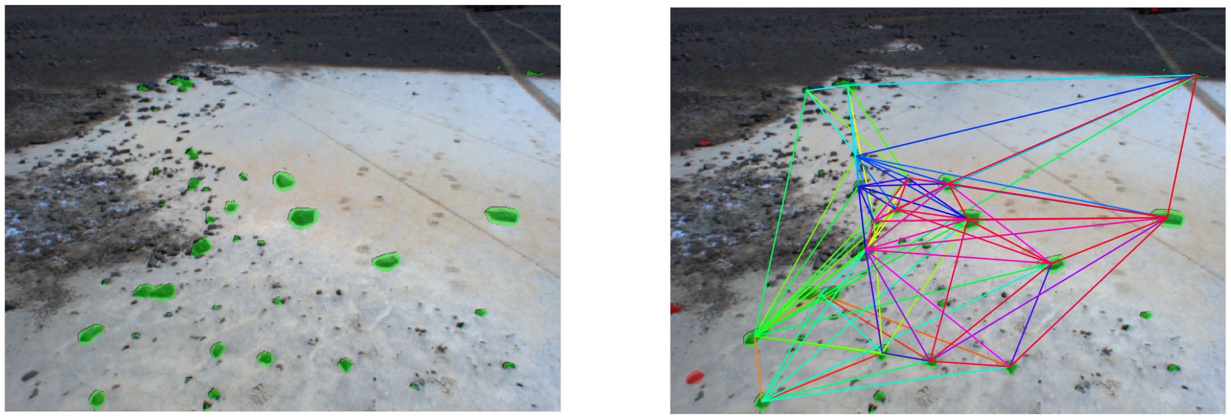

Figure 24: Full Delaunay triangulation of an Atacama Desert scene, with each triangle shown in a different color.

multiple overlapping circumcircles. Additionally, if more than three points lie on a circumcircle, multiple triangulations of those points are valid. While there will never be more than 3 points exactly on a circumcircle in real numerical data, the error in centroid measurements and transient rock issue can potentially cause two views of the same scene to produce different Delaunay triangulations. This makes the use of 3-constellations with Delaunay triangulations non-robust — small changes in the point set alter the set of constellations generated.

Connected Delaunay Triangulation (CDT) An approach to improve the robustness of Delaunay triangles was developed in this work and coined "connected Delaunay triangulation" (CDT). While Delaunay triangles themselves are non-robust, higher-order constellations created by connecting Delaunay triangles that share an edge can be robust, or resistant to small changes in the point set. Additionally, the CDT algorithm is deterministic and for N rocks, produces at most $N * 2^{k-2}$ k-constellations, making it an ideal constellation generator for higher-order constellations.

Algorithm 2 describes how a set of k-constellations can be created from a Delaunay triangulation.

Algorithm 2 Method for connecting Delaunay triangles to from a k-constellation

```
Let a constellation made of $k$ points be denoted as $C_k$.
01.  Compute the Delaunay triangulation to form $T$ triangles $C_3^i$
     for $i=1$ to $T$ .
02.  Compute the $T \times T$ adjacency matrix $A$ between all $C_3$ that
     share an edge $E_{ij}$, where $A_{ij}$ is the point in $C_3^j$ that is not
     in $E_{ij}$.
03.  Let $U_A(C_k^i)$ be an operation that uses $A$ to produce, for
     a list of $n$ edges in $C_k^i$, the set of $n$ new constellations
     $\{S_{k+1}^1, \ldots, S_{k+1}^n\}$.

04.  For $c$ in 3 ... $k$
05.    For $i$ in 1 ... $T$
06.      $S_c^i \longrightarrow U_A(C_c^i)$
07.    EndFor
08.    $C_c \longrightarrow Unique(S_c)$
09.  EndFor
```

An added advantage of this method is that locally defined constellations can be easily enforced by removing triangles with edge lengths above some distance threshold D_E from the original Delaunay triangulation, before constructing the adjacency matrix.

Figure 25 below shows the full set of 3-, 4-, 5- and 6-constellations produced by connecting Delaunay triangles for the set of points on the left.

Convex Hull Boundaries (CHB) The connected Delaunay triangulations can be taken a step further by finding the convex hull of each connected Delaunay triangulation and using the k points on the boundary of a convex hull as a k-constellation. This can reduce the order of a constellation, so it is necessary to re-sort each convex hull boundary into the correct k-constellation size so that the desired sizes of constellations can be matched. Boundary points are sorted in counterclockwise order to increase efficiency when comparing two CHB descriptors.

CHBs have the same advantages as CDTs, but with a reduced set of descriptors and increased robustness to noisy points. A rock that intersects many circumcircles of a Delaunay triangulation can drastically alter that triangulation. However, when taking the convex hull boundaries of the CDT set, those inner noisy points are ignored entirely.

Figure 26 shows an example set of CHB constellations produced by (1) Finding the set of connected Delaunay triangles (2) Computing the convex hull boundaries and (3) Re-sorting the resulting constellations based on size.

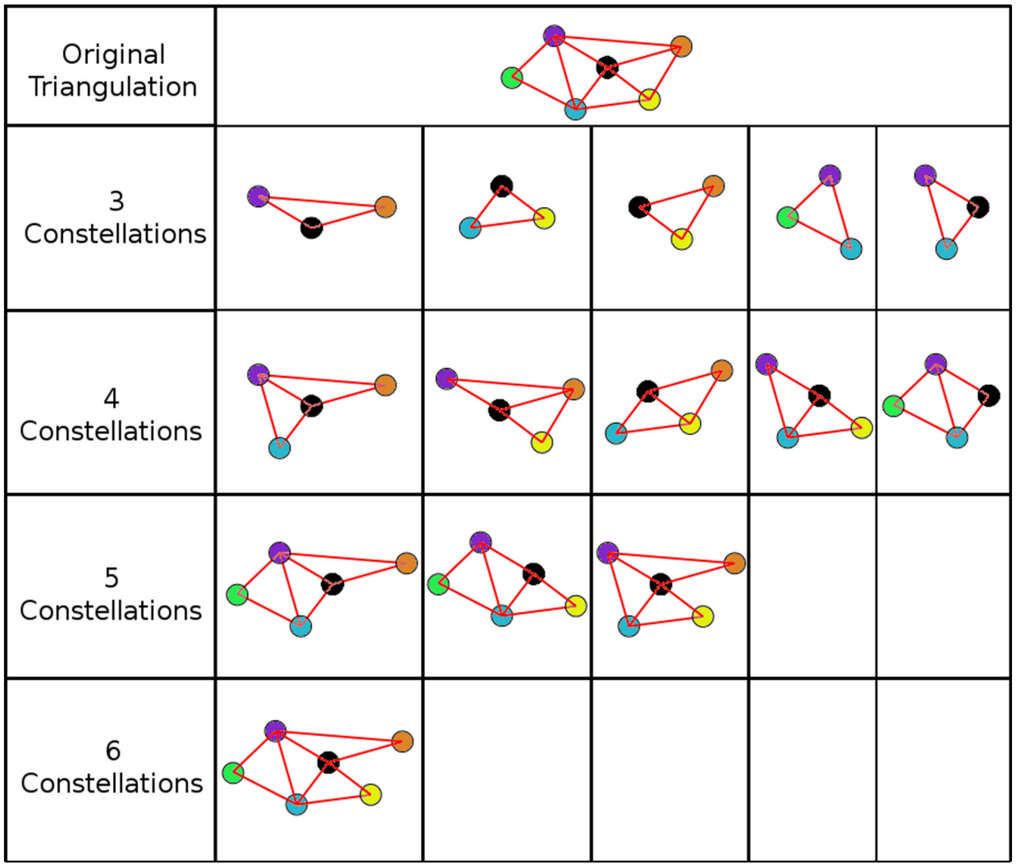

Figure 25: Connected Delaunay triangles

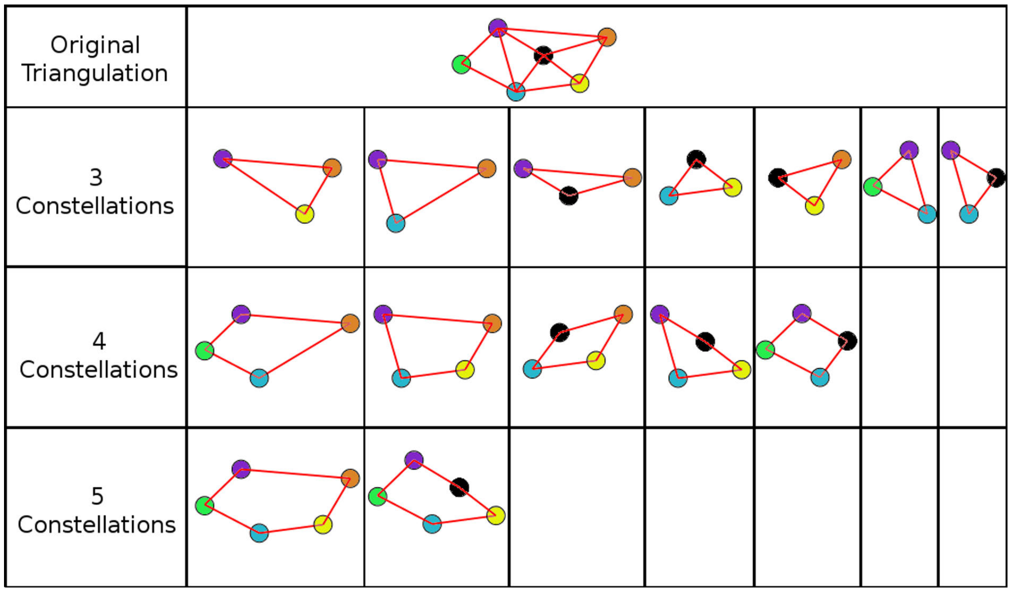

Figure 26: Convex hull boundaries of connected Delaunay triangles

4.2.2 Describing Constellation Geometry

After the constellations have been generated, a feature descriptor must be produced for each one. For a k-constellation, the feature descriptor must

1. Be deterministic.
2. Contain at least $2k - 3$ independent variables.
3. Produce the same descriptor after any planar rotation or translation.

The first condition is a basic requirement to ensure that two congruent constellations can be correctly matched every time. The second condition requires a descriptor to have the minimum number of variables needed to uniquely identify a set of k points. This ensures that only one shape can be described by the descriptor. The third condition ensures that the same constellation on a plane in real-world coordinates can be matched from different viewing locations, which is the basis for data association in the desert.

Pairwise Distances A set of pairwise distances meets the three conditions and produces a descriptor of length $k(k-1)/2$. To be deterministic, the distances are sorted.

Polar Coordinates The polar coordinates of the points produces the minimal feature descriptor of length $2k - 3$. This meets the three conditions if a consistent, rotation invariant frame of reference for computing the polar coordinates can be found. One method that accomplishes this is to (1) Select as the origin P the most isolated point (largest mean pairwise distance) (2) Select the nearest neighbor to this origin Q (3) use the vector PQ as the X-axis with which to compute polar coordinates. The descriptor values are then sorted by increasing angle, with the origin coordinate and angle to Q excluded.

Convex Hull Boundary Points For a convex hull boundary descriptor, the CHB points themselves can be used to produce a feature descriptor of length $2k^2$ — the k pairs of x and y values, all shifted and rotated k times to use each point and its counterclockwise neighbor as the origin and X-axis. This will always produce the same descriptor after any planar motion, but the ordering is not guaranteed to be consistent.

4.2.3 Incorporating Rock Features

So far we have only described the constellation geometries based on the centroids of rocks. However, the pixel-wise labeling outputted by the rock detector also allows us to use information about each rock in the original image. Requiring a match between rock features in addition to constellation geometry further safeguards against false positive constellation matches. Each point added to a k-constellation increases the number of independent variables by two, but each additional rock feature added to the constellation increases the number of independent variables by k. This makes accurate and consistent rock features especially desirable.

An ideal rock feature is one that is invariant to planar rotation and translation, as well as to lighting conditions. Some features that were tested include

Height	The local height of a rock above its surrounding ground. This is found by using RANSAC on the 3-D points around the labeled rock to obtain a ground plane and then taking the mean height above the plane. This is invariant to planar motion, but was often unable to be computed because the stereo point computation failed for most of the featureless desert ground plane.
Isolation	A spatial descriptor for a rock that measures its isolation from all other rocks. Various isolation metrics are possible. For example, the distance to the nth-nearest neighbor or the mean of a rock's distance to all other rocks could be used. This is invariant to planar motion and lighting, but because it relies on knowledge of other rocks' locations and not all rocks are guaranteed to be detected, it is not a robust measurement.
Color	The simplest feature to consider, but due to the lack of variety, general sameness of rock and ground color, and significant changes caused by lighting and shadows, this feature adds little information.
Pixel Area	The number of connected labeled rock pixels for a given rock. This is not invariant to planar motion and depends on consistent rock detection in various lightings. It is not useful for generalized constellation matching but is fairly consistent in consecutive scene matching. Across a single frame (approximately 1 meter of motion) the pixel size does not alter drastically.
Visible Surface Area	The approximate visible surface area in square meters of a detected rock. This is estimated by finding the largest diameter of the rock in pixels, and multiplying this diameter by a value that relates the number of horizontal meters per pixel to the range from the camera to the rock's centroid. Then πr^2 can be used to estimate the surface area. It is invariant to scale changes (moving toward or away from a rock) but not guaranteed to be invariant to viewing a rock from a different angle. However, rocks do tend toward a spherical shape so it can be useful in this application.

None of the rock features described above meet the ideal invariant conditions described, but they still provide a useful approximation that can filter out moderate to severe mismatches.

4.2.4 Matching Constellations

Once a set of constellation descriptors is built for two scenes, the constellations are compared to determine if there are matched landmarks between the scenes. This is the well-known data association problem in SLAM [65]. If a correct match is found, a measurement constraint can be added to the pose graph to improve the estimate of the rover's state. This matching process varies depending on the type of feature descriptor used, so they are categorized below into vector descriptors, where the feature is represented as a vector of values, and point set descriptors, which are all the 2-D points of a constellation.

Vector Descriptors To match vector descriptors, we need some form of similarity metric that assigns a value from 0 to 1 based on how similar two constellations appear. Then, if a user-defined similarity threshold is met, the constellations are considered matched. The vector descriptors should be normalized in some way so that features with different units can be weighted equally.

Cosine distance The cosine distance between two vectors A and B is defined as the cosine of the angle θ between them:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|}$$

Since we deal only with positive values in our feature vector this will give a value from 0 to 1.

Ratio Here, the ratio between two vectors A and B is defined as

$$\text{similarity} = \frac{\sum_{i=1}^{n} \frac{\min(A_i, B_i)}{\max(A_i, B_i)}}{n}$$

which is simply the mean of the smaller values in the feature vector after normalizing by the larger values.

CHB Point Set Descriptors

Describing sets of points requires a different approach because the points are not necessarily in sorted order, so it becomes necessary to find a best-fitting match. However, because it uses all the information about both sets it is also the most flexible method of matching descriptors, allowing two constellations of different sizes to be matched. The problem is defined below for the specific case of convex hull boundary points sorted in counterclockwise order.

Given two sets of points A_{k_1} and B_{k_2}, of sizes k_1 and k_2 respectively, with $k_1 \leq k_2$, determine if A_{k_1} can be translated and rotated such that a point in A_{k_1} is within some error e in Euclidean distance away from exactly one point in B_{k_2}. This is accomplished by computing the coordinates of the points in k_1 different reference frames, using each point and its counterclockwise neighbor as a frame of reference. Call this k_1 length set of transformed point sets S_A. The same is done for the k_2 points of B_{k_2} to produce S_B. Then $k_1 * k_2$ comparisons are made between each of the point sets of S_A and S_B. If any of these comparisons have exactly k_1 points matching, then A_{k_1} and B_{k_2} are considered matches.

If a constellation match is found, care must be taken to keep track of the pairs of points that matched so that rock features can be correctly compared.

For efficiency, the sets of shifted and rotated points S_A and S_B can be computed and stored at the time of constellation generation rather than when matching. This way, the computation is done only once.

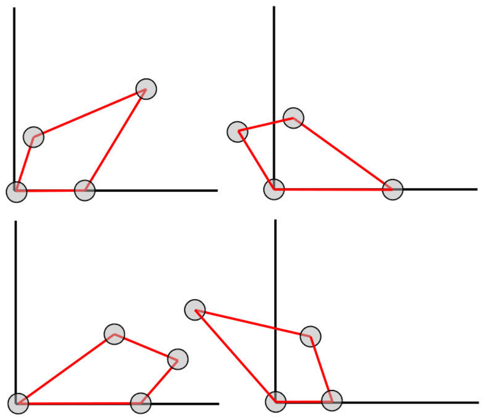

Figure 27: A 4-constellation is shifted and rotated 4 times to produce a 16 point descriptor.

Setting Error Thresholds The error threshold e between two points requires some care in setting. While a single value can be used to determine all point matches, this is not ideal because of the increase in stereo measurement error for points further away. Instead, it is more accurate to assign each rock a measurement error based on its range from the camera. Then, assuming measurement error is approximately normally distributed, the Euclidean distance between two points p_1 and p_2 with associated errors σ_{p_1} and σ_{p_2} should be compared against the threshold given by propagating uncertainty:

$$e_{p_{12}} = \sqrt{(\sigma_{p_1})^2 + (\sigma_{p_2})^2}$$

$$\text{where } \sigma_{p_1} = \varepsilon_m p_{1_{range}} \text{ and } \sigma_{p_2} = \varepsilon_m p_{2_{range}}$$

The value ε_m used in this work was experimentally approximated from the actual variances in the images. In theory, ε_m should be based on both stereo error and rock detection error, but the nature of CNN rock detection makes the latter difficult to determine.

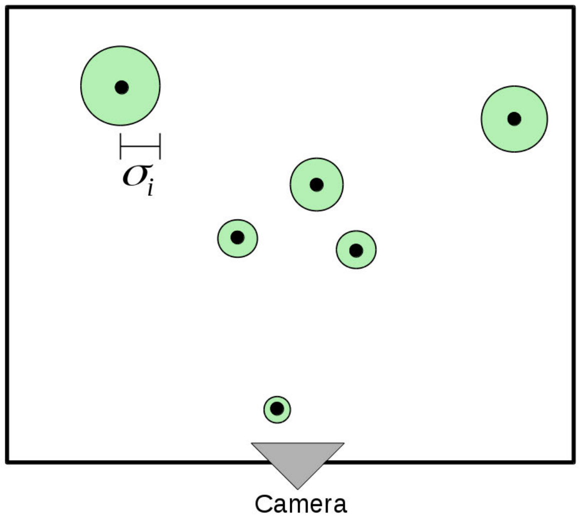

Figure 28: Estimated errors of each rock

Comparing Different Sizes of Constellations When comparing two scenes, each with constellations of varying sizes, the order in which to compare different sizes of constellations is an important consideration. For example, consider the following scenario:

Suppose one chooses to connect up to 5 Delaunay triangles (7 points), and then use 4-, 5-, and 6-constellations (denoted hereafter as 4-5-6-constellations) of convex hull boundary points to describe each scene of rocks. When comparing these 4-5-6-constellations in a scene A to the 4-5-6-constellations in a scene B, there are 9 combinations of comparisons to consider: {(4,4), (4,5), (4,6), (5,4), (5,5), (5,6), (6,4), (6,5), (6,6)}. The sequence of comparisons matters because a 4-constellation could be a subset of a larger constellation that has already been matched. If 4 points in scene A match with 4 points in scene B, and the 4 points in both scenes are subsets of 5-6-constellations, we would find 9 matches of the same 4 rocks.

The subsets of larger constellations cannot simply be removed, since partial matches are a common occurrence. For example, scene A might contain a unique 6-constellation C_6^A and scene B might view the same scene with two of the rocks obscured, mismeasured, or undetected. The 4-constellation C_4^B that is visible in scene B should still be matched to the 4-constellation in scene A that is a subset of C_6^A.

To account for both multiple matches and partial matches, the comparison sequence can be adjusted so that constellations with the least difference in size are compared first, and then as a tie-breaker, consider the largest constellations first. The order in the above example would then be {(6,6), (5,5), (4,4), (6,5), (5,6), (5,4), (4,5), (6,4), (4,6)}. Then, for any constellation that has been matched, remove all subsets of that constellation from consideration in the current scene comparison. With this method, a constellation will not be matched if a superset of that constellation has already been matched, and, if a match exists, only the minimal set of matching points will be found.

The subsets of a constellation can be easily determined during constellation generation since constellations are built sequentially from their subsets.

Algorithm 3 RANSAC on initially matched constellations

```
Given two sets of $N$ constellations $C_1$ and $C_2$, and an inlier
threshold $e_i$

01.  Initialize num_inliers $\rightarrow$ 0

02.  For $i$ ... max_iterations
03.    Randomly select 2 pairs of matching constellations $C_{r_1}$
       and $C_{r_2}$.
04.    Use SVD to determine the least-squares rigid transform
       $(R,t)$ that aligns the points of $C_{r_1}$ to the points of
       $C_{r_2}$[59].
05.    Apply the rigid transform $(R,t)$ to all the points of $C_1$
       to produce a projected set of constellations $C_2'$.
06.    Find the $N$ sums of squared errors between each
       constellation in $C_2'$ and its corresponding match in $C_2$.
07.    Let $I$ be the constellations with SSE < $e_i$ and $M$ be the
       length of $I$.

08.    If $M >$ num_inliers
09.      num_inliers = $M$
10.      best_inliers = $I$
11.    EndIf
12.  EndFor

13.  Return best_inliers
```

RANSAC Outlier Rejection

Random Sample Consensus [20] is a commonly used method for outlier rejection. It is used here to determine the points of initially matched constellations between two scenes that are consistent with each other, as described in Algorithm 3.

In step 3, two constellations are used to fit the rigid transform model because a rigid transform has six degrees of freedom and two constellations guarantee at least six points. In step 8, ties in the number of inliers found can be broken by comparing the mean inlier error. For computational efficiency, the algorithm can also be exited early if some minimum number of constellation inliers is found. The inliers returned by RANSAC can be verified further by comparing the similarity between rock features. This produces the final set of matched constellations that can be defined as landmarks in the GraphSLAM optimizer.

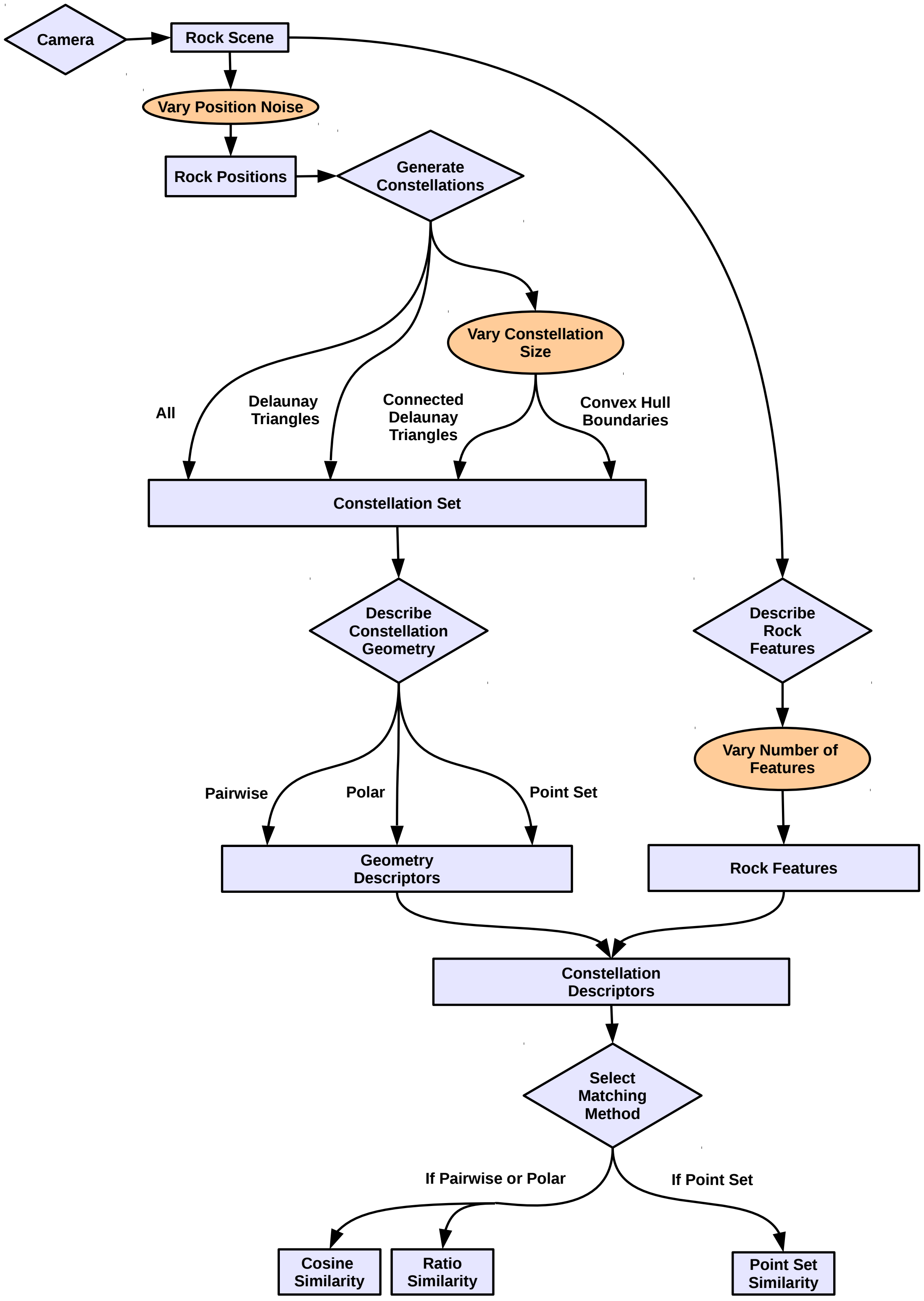

Figure 29: Constellation matching process.

4.3 Results

Determining an optimal constellation matching strategy among the many possible variations discussed above is of prime importance to the success of the overall MeshSLAM algorithm. A visualization of the options and variables at each stage of the matching process is shown in Figure 29. A simulation analysis will be used to compare these options under varying scene-specific conditions (shown in orange ovals in Figure 29). In particular, results are shown for the following:

- Comparing feature vector similarity metrics
  - *cosine similarity* and *ratio similarity* for uniform and exponentially distributed features.
- Generating constellations
  - Generating all constellations vs generating CHB constellations
- Describing constellations
  - Comparing pairwise vs polar vs CHB point descriptors
    - For varying sizes of constellations
  - Varying the number of features per rock
    - For uniform and exponentially distributed features
  - Comparing precision and recall when measurement noise is added
    - For varying measurement errors
- Computation of CHB matching
  - For varying numbers of rocks and constellation sizes

In the results below, a heavy emphasis is placed on robustness to false positives, as this is the primary measure of effectiveness for a constellation landmark. False data associations in the factor graph can easily cause dramatic failures in the optimizer, and preventing them is undoubtedly the main challenge of using rocks as landmarks. However, we intend to show that by selecting appropriate constellation descriptors and matching methods, rocks can indeed by used as landmarks with low false positive rate.

Due to the highly parameterized nature of the constellation matching problem, it will be necessary to make some simplifying assumptions, which we provide the basis for below. The section will close with a discussion of these results and their applicability.

4.3.1 Comparing Similarity Metrics

The two similarity metrics described in Section 4.2.4 are compared in Figures 30 and 31. In Figure 30, a set of 5000 random vectors of length 4, 8, and 12 were generated from a uniform distribution and compared pairwise against each other, using either the cosine similarity metric (blue) or ratio similarity metric (red). These random vectors simulate the features associated with rocks in a constellation. The sizes 4, 8, and 12 correspond to a 4-constellation with 1, 2, or 3 features per rock. Probability density functions are shown on the left and the corresponding cumulative density functions are shown on the right.

An ideal similarity metric provides a large separation in score between a true and false positive match. This means that, on average, two randomly selected feature vectors should have a low similarity score. Highly unique similarity metrics correspond to a lower likelihood of a constellation's set of features incorrectly matching with another constellation's set of features, thus decreasing the false positive rate. In the plots below, a unique similarity metric has a longer right-sided tail in the probability density function, indicating that a smaller proportion of feature vectors match strongly. Equivalently, the cumulative density function of a more robust similarity metric should have higher value of $cdf(x)$ for the same value of x. That is, If $cdf_{metric_1}(x) > cdf_{metric_2}(x),\ metric_1$is the more robust similarity metric.

In Figure 30, the cosine similiarity metric performs better for all numbers of features and for all similarity thresholds. Figure 31 shows similar plots but samples the random features from an exponential distribution. The result is that the ratio similarity metric performs better in this case for similarity thresholds greater than about 0.4, though this value varies depending on the number of features used.

We can conclude that features distributed uniformly should be compared using the cosine similarity metric while features distributed exponentially should be compared with the ratio similarity metric when higher similarity score thresholds are desired. In practice, most features are distributed exponentially and the similarity threshold will almost always be set higher than 0.4, so the ratio metric is preferred and will be used where relevant in the rest of the results.

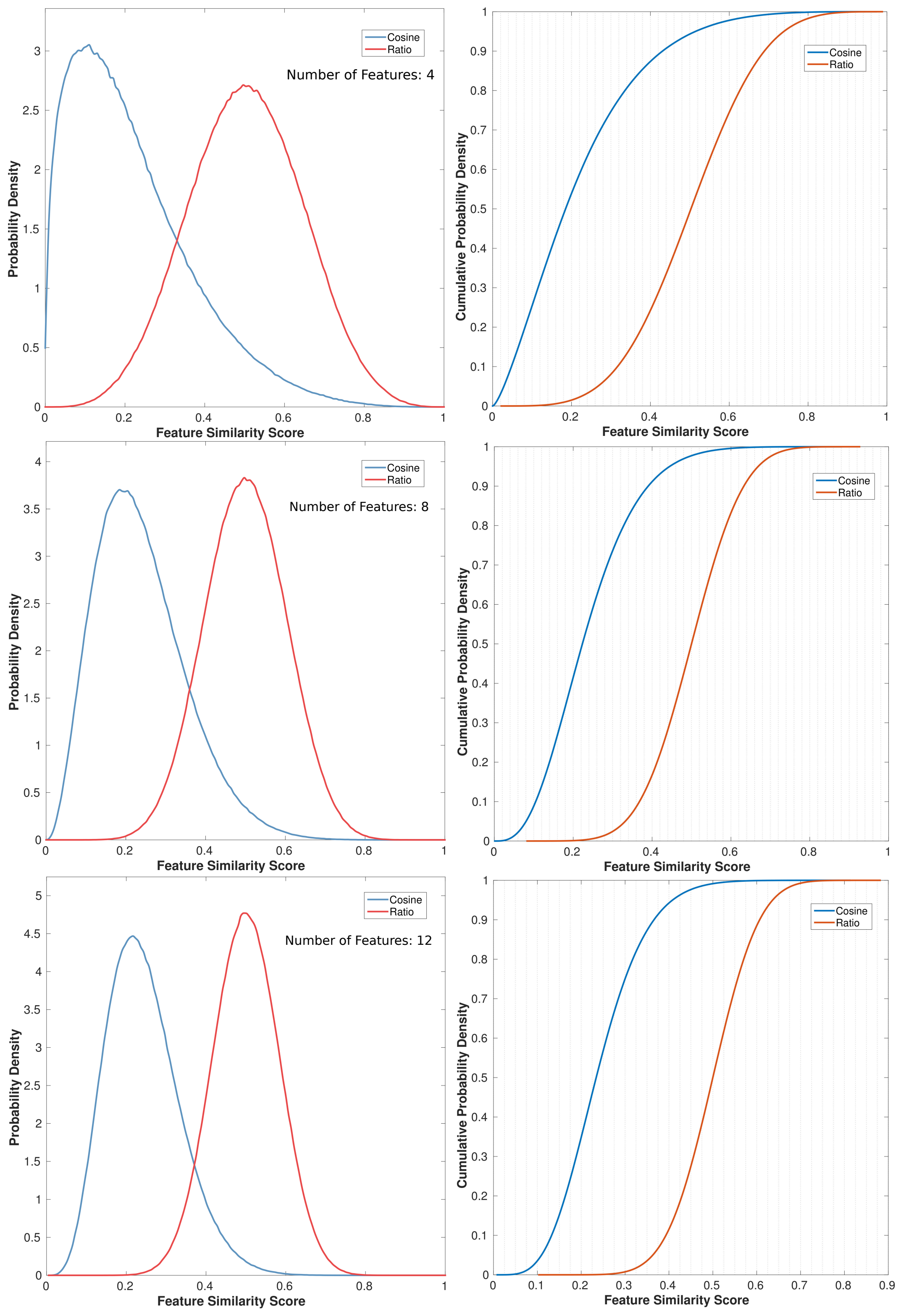

Figure 30: Probability density functions (left) and cumulative density functions (right) when using cosine and ratio similarity metrics to compare **uniformly** distributed random feature vectors of size 4, 8, and 12 from top to bottom.

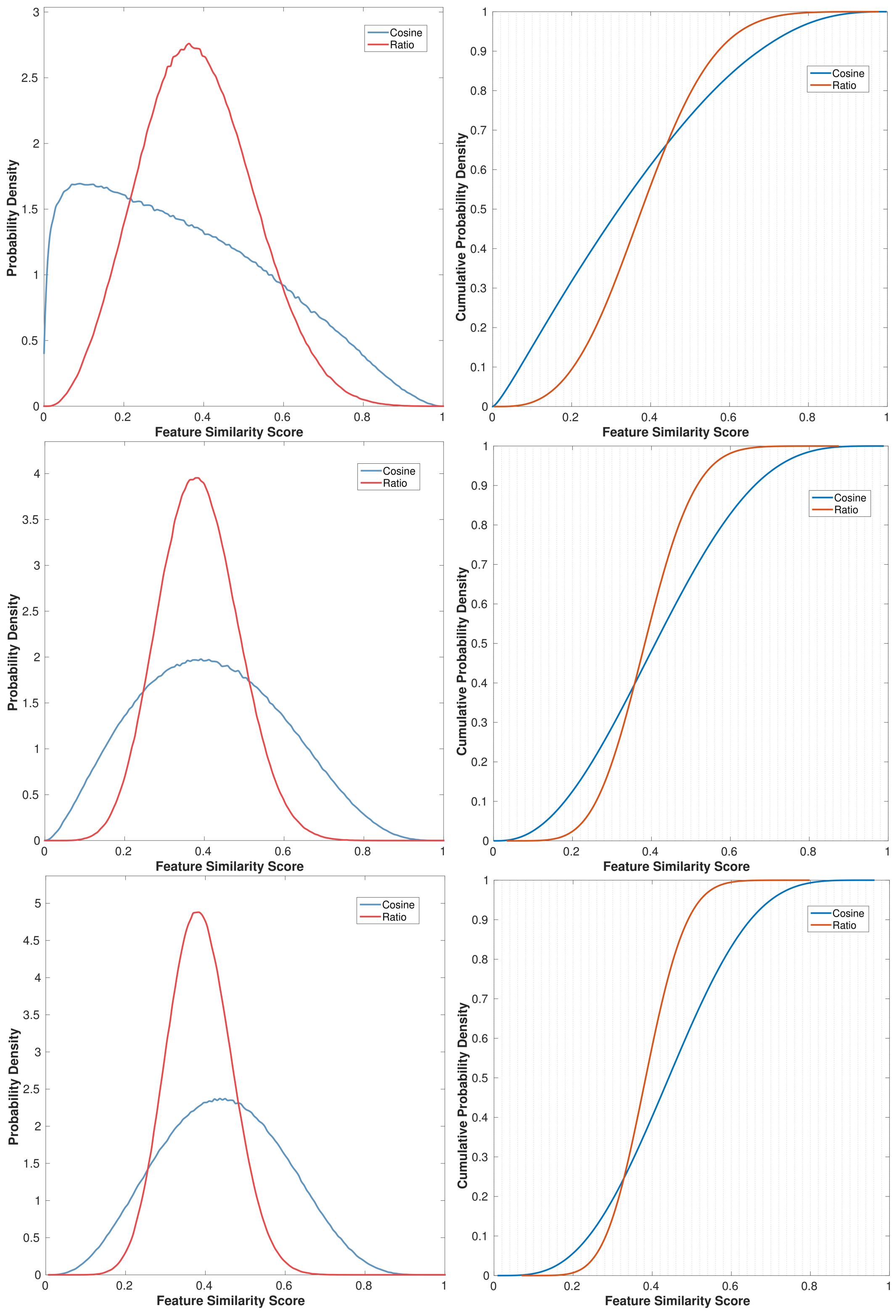

Figure 31: Probability density functions (left) and cumulative density functions (right) when using cosine and ratio similarity metrics to compare **exponentially** distributed random feature vectors of size 4, 8, and 12 from top to bottom.

4.3.2 Monte Carlo Simulations

In the following results, a Monte Carlo simulation is used to model constellation matching. The specific procedure varies depending on the values being analyzed but the general procedure is as follows:

1. Generate 5,000 pairs of scenes with a field of view of 20x20 meters, each containing 20 rocks uniformly and randomly distributed.
2. Build constellations up to a max order of 8, and form constellation descriptors.
3. Make 5,000 comparisons between the constellations in each pair of scenes.
4. Model the number of constellation matches that occur between a pair of scenes as a Poisson distribution $p(k)$, where the λ to be estimated is the average number of constellations that match between two randomly selected scenes.
5. Compute the probability of at least one match between two random selected scenes as $1 - p(0)$.

The above process makes several assumptions based on the specific Atacama dataset under study. The size of the field of view is approximately 20x20 meters for the stereo camera and viewing angle used by Zoë. It's also assumed that 20 rocks can be seen in each frame. This is an upper bound assumption since most scenes contain far fewer. The boxplots in Figure 32 show the distribution of the number of rocks in each frame for 6 sample datasets from the Atacama Desert of varying terrain types. Nearly all scenes contain fewer than 20 rocks as can be seen by the dashed green line above 5 of the 6 boxplot whiskers. Atacama Desert scenes contain on average 7 rocks, so the 20 rock assumption can be safely made without overestimating the robustness of constellation matching.

The Poisson distribution is a reasonable approximation of the distribution of constellation matches. Each pair of scenes compares a large number of constellations and each constellation comparison can be considered a Bernoulli trial with very low probability. Thus, by the *law of rare events,* the Poisson distribution can be safely applied to constellation matches [9].

All error bars shown are 95% confidence intervals based on the approximation of the value λ, unless otherwise noted.

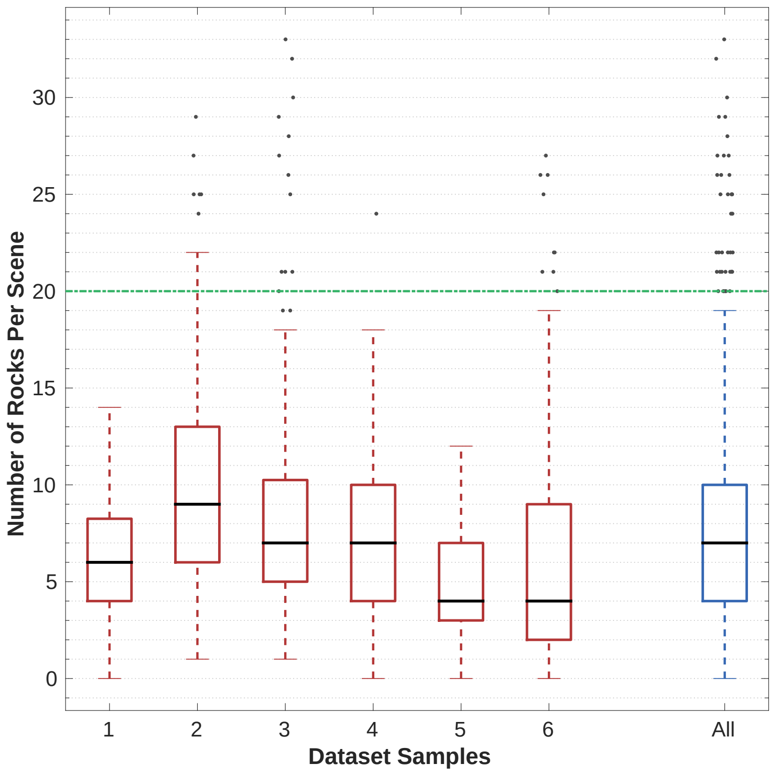

Figure 32: Variation in number of rocks between scenes in 6 different sample datasets from the Atacama Desert.

4.3.3 Generating All Constellations

The method of generating all possible combinations of 3-constellations in a scene is used as a baseline metric for analyzing false positive rates. Intuition suggests that the sheer number of constellations generated by this method makes it susceptible to large numbers of false positives and the results appear to agree. Figure 33 shows the probability of at least one false positive match between two randomly selected scenes with 20 rocks each when every possible 3-constellation is generated. The CHB point set method is used to match the constellations (Section 4.2.4). The probability of a false positive match is shown as a function of measurement error, which is a factor multiplied by the range of a rock to determine that rock's position error, as described in Section 4.2.4. For example, a rock 15 meters away with 1% measurement error would have position error of 0.15 meters.

The plot shows that even for measurement errors as low as 0.3%, the chance of a false positive match between two random scenes is 95%.

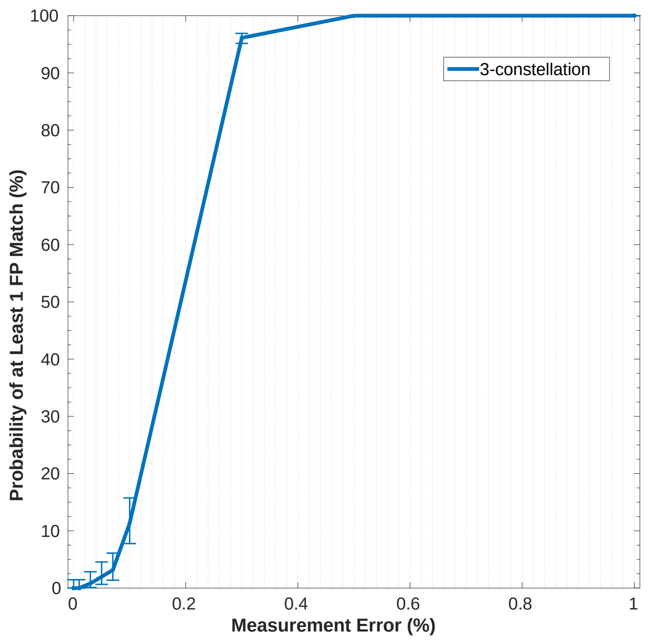

Figure 33: Probability of at least one false positive match vs. the measurement error as a percent of rock range when generating all possible 3-constellations.

4.3.4 Comparing Constellation Vector Descriptors

The two vector descriptors — *pairwise* and *polar* descriptors (Section 4.2.2) — are compared in Figure 34. The top images show the unmagnified and magnified results of constellation matching using pairwise descriptors. The bottom images show the unmagnified and magnified results of using polar descriptors. The probability of at least one false positive match is plotted against the descriptor similarity threshold for varying sizes of constellations. The ratio similarity metric, described in Section 4.2.4, was used to measure constellation similarity. The similarity threshold is the percentage of similarity from 0% to 100% between two constellations for them to be considered matches. In general, a high similarity threshold decreases the likelihood of a constellation match.

An ideal constellation vector descriptor would achieve low false positive rates at low similarity thresholds. Lowering similarity thresholds increases recall, which is desired as long as false positive rate remains low. Similarity thresholds set very high will guarantee no false positives but may not accept matches at critical loop closure points. As always, the primary tradeoff is between precision and recall.

For the pairwise descriptor, the figure shows an interesting crossover for the plots of different-sized constellations. This is caused by the many redundant variables used by the pairwise descriptor. For large k-constellations, the number of pairwise descriptors given by $k(k-1)/2$ is much larger than the number of independent variables needed to describe the constellation $2k-3$, and the distribution of ratio similarity scores from matching a large number of variables tends toward a normal distribution, as can be seen in Figure 30. Thus, at lower thresholds, larger constellations with their many features become less unique than smaller onstellations, which are described by fewer variables.

In the bottom plots showing the same results for a polar constellation descriptor, we note a simple trend — larger constellations can achieve the same false positive probability rate as smaller constellations but at lower thresholds. As might be expected, larger constellations are more unique than smaller ones. However, the plot also shows diminishing returns on the reduction of false positives, so the benefit of large constellations should be balanced against the extra computational cost.

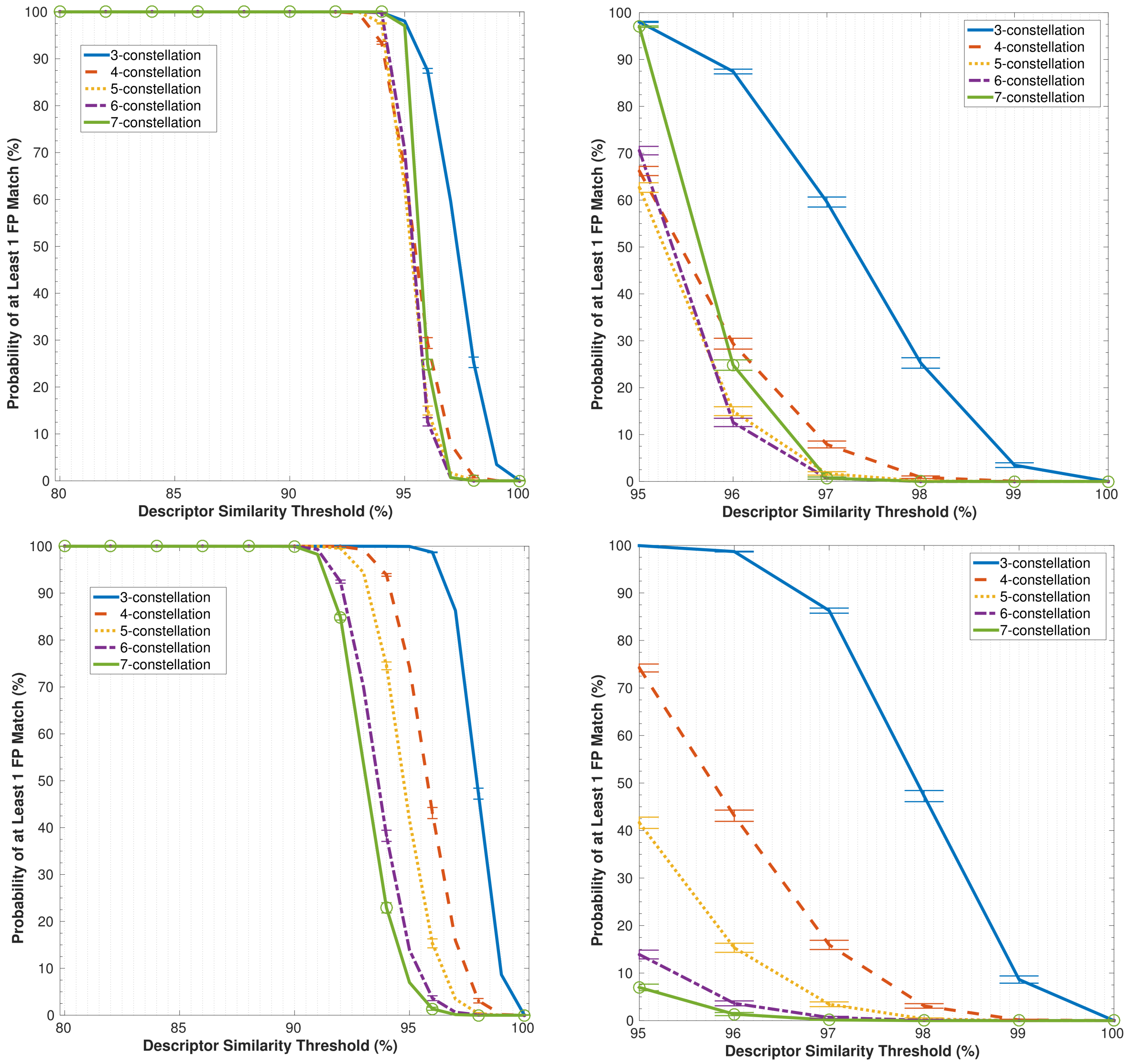

Figure 34: Comparison between **pairwise** (top) and **polar** (bottom) descriptors, showing the probability of at least one false positive match vs. the measurement error as a percentage of rock range for **different sizes of constellations**. The right column only shows similarity thresholds of 95% or greater.

4.3.5 Varying the Sizes of Convex Hull Boundary Descriptors

Results in Figure 35 show the probability of a false positive match plotted against the measurement error for varying sizes of convex hull boundary descriptors. The results are plotted only for measurement errors up to 5% of rock range, as beyond this is an unrealistic amount of error in the measurement of the rock position. A rock at 20 meters with 5% measurement error would have 1 meter in position error, which is a large tolerance for stereo images, even considering the addition of error from the rock detector.

We note again that for the same measurement error, larger constellations achieve lower false positive rates, although here there are no diminishing returns.

It would seem robustness can be achieved simply by using very large constellations. However, larger constellations have the significant disadvantage of low recall in real desert scenes, as they require more rocks to form and are more likely to be partly occluded. Recall that an average scene in the Atacama Desert contains 7-8 rocks. Thus, for the remainder of the results we focus on analyzing the effectiveness of smaller constellations of size 4 and 5.

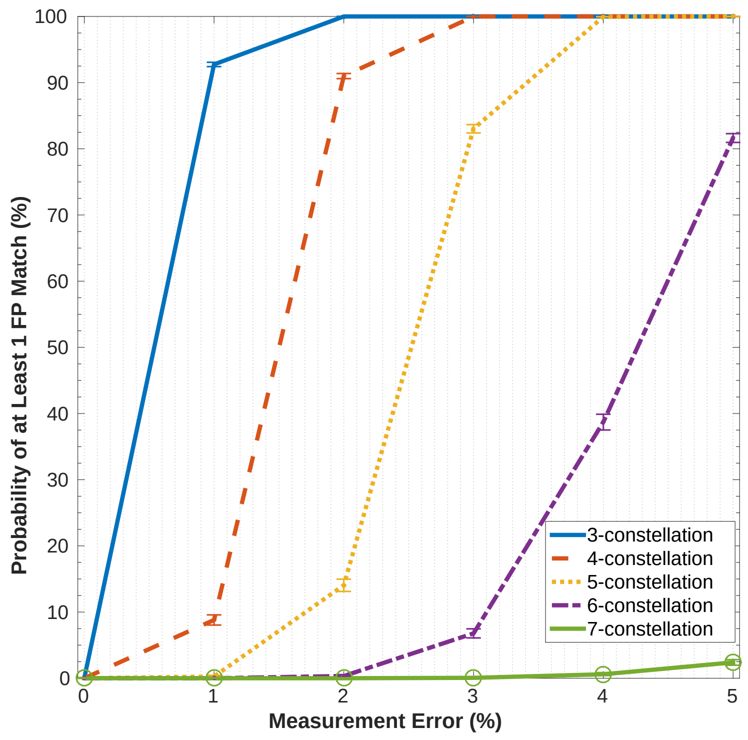

Figure 35: Probability of at least one false positive match vs. the measurement error as a percentage of rock range for different sizes of CHB constellations.

4.3.6 Varying Rock Position Noise

The vector descriptors cannot be compared directly with the CHB descriptor since they use different parameters. However, their performance in recall and precision can be directly compared when noise is added to the rock positions.

The Monte Carlo simulation used here is similar to the method described above, except only 5,000 scenes of constellations are generated, which are then matched with noisy versions of themselves. Gaussian noise with zero mean and varying standard deviations was added to each rock's position, and then constellations were built from the noisy set of points. The constellation size was set to 4. The 5,000 pairs of scenes were matched against each other and the number of true positive, false positive and false negative matches were determined to provide an estimate on the recall and precision, given by

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Figure 36 shows the precision and recall curves for pairwise descriptors and Figure 37 shows the precision and recall curves for polar descriptors. The similarity threshold is also varied in both cases. Figure 38 shows the precision and recall curves for CHB descriptors with varying measurement errors. To make the plots comparable to each other, similarity thresholds and measurement errors were chosen to correspond to the same false positive probabilities. For example, the 97% similarity threshold used in the pairwise descriptor plots corresponds to a false positive probability of about 9% in Figure 34. The measurement error of 1% also corresponds roughly to a false positive probability of 9% in Figure 35, so the recall and precision for these two curves can be directly compared.

All descriptors perform comparably well in recall, but the CHB descriptor has generally lower recall for all amounts of rock noise. More important, though, are the precision curves which determine the descriptor's susceptibility to incorrect data associations. There we can see that the CHB descriptor is outperformed by the polar descriptor only in the case of very high rock noise and low measurement error/high similarity threshold, which is the most unrealistic case. In the cases where recall is greater than 10% (below 0.3 meters rock position noise) the CHB descriptor is a more precise constellation descriptor at all tested measurement errors.

4.3.7 Varying the Number of Features Per Rock

Results so far have examined methods of constellation generation and geometric descriptors. CHB descriptors have been shown to be the most robust descriptors, but the likelihood of at least one false positive match is still unmanageably high (Figure 35) for smaller constellations at measurement errors greater than 1% of rock range. Comparing the 4-constellations of two random scenes of 20 rocks at 2% measurement error produces a false positive probability of 90%. For 5-constellations, this value reduces to 14%.

However, incorporating rock features into constellation matching can greatly reduce this risk. This can be shown by performing the same constellation matching procedure as above, but also assigning each rock a random feature vector to simulate the features of real rocks (e.g. pixel area,

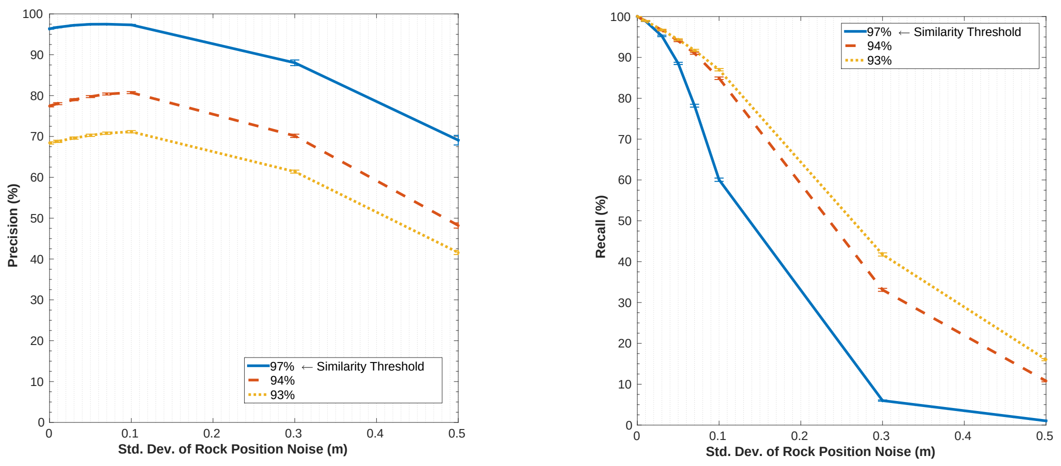

Figure 36: Precision and recall curves for **pairwise** descriptors at varying thresholds.

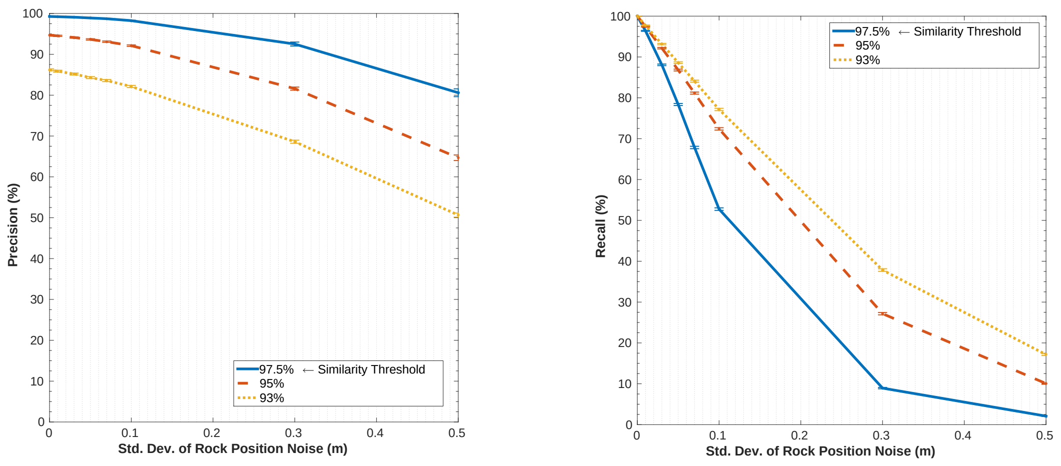

Figure 37: Precision and recall curves for **polar** descriptors at varying thresholds.

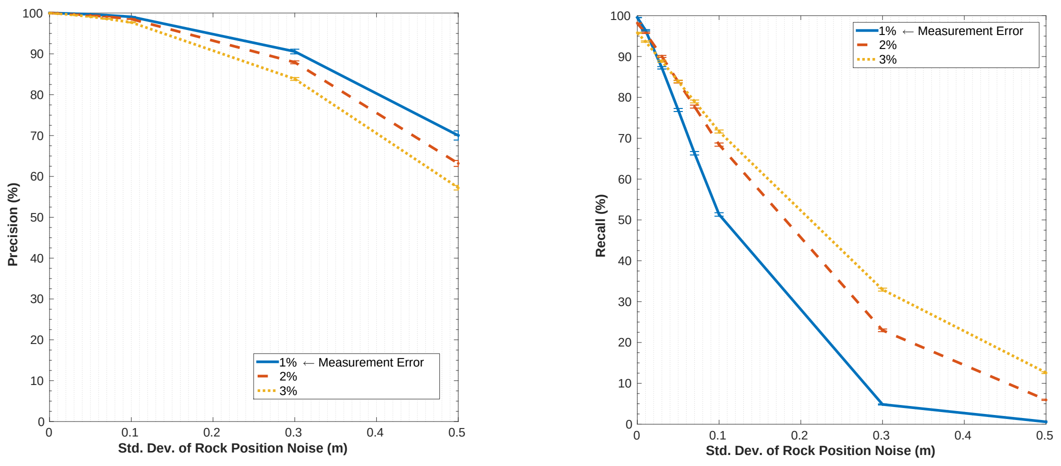

Figure 38: Precision and recall curves for **CHB** descriptors at varying measurement errors.

rock size, color, etc.). This allows us to determine how the false positive rate is affected by these extra random features.

The result of this analysis is shown in Figure 39. The random features for each rock are uniformly and randomly generated. Measurement error is set constant at 2%, and the analysis is repeated for both 4-constellations (top) and 5-constellations (bottom).

Overall, these results indicate the usefulness of extra rock features in reducing false positive rate. For feature similarity thresholds greater than 60%, the likelihood of false positive matches is greatly reduced, though we note diminishing returns on these improvements.

Uniformly distributed random features

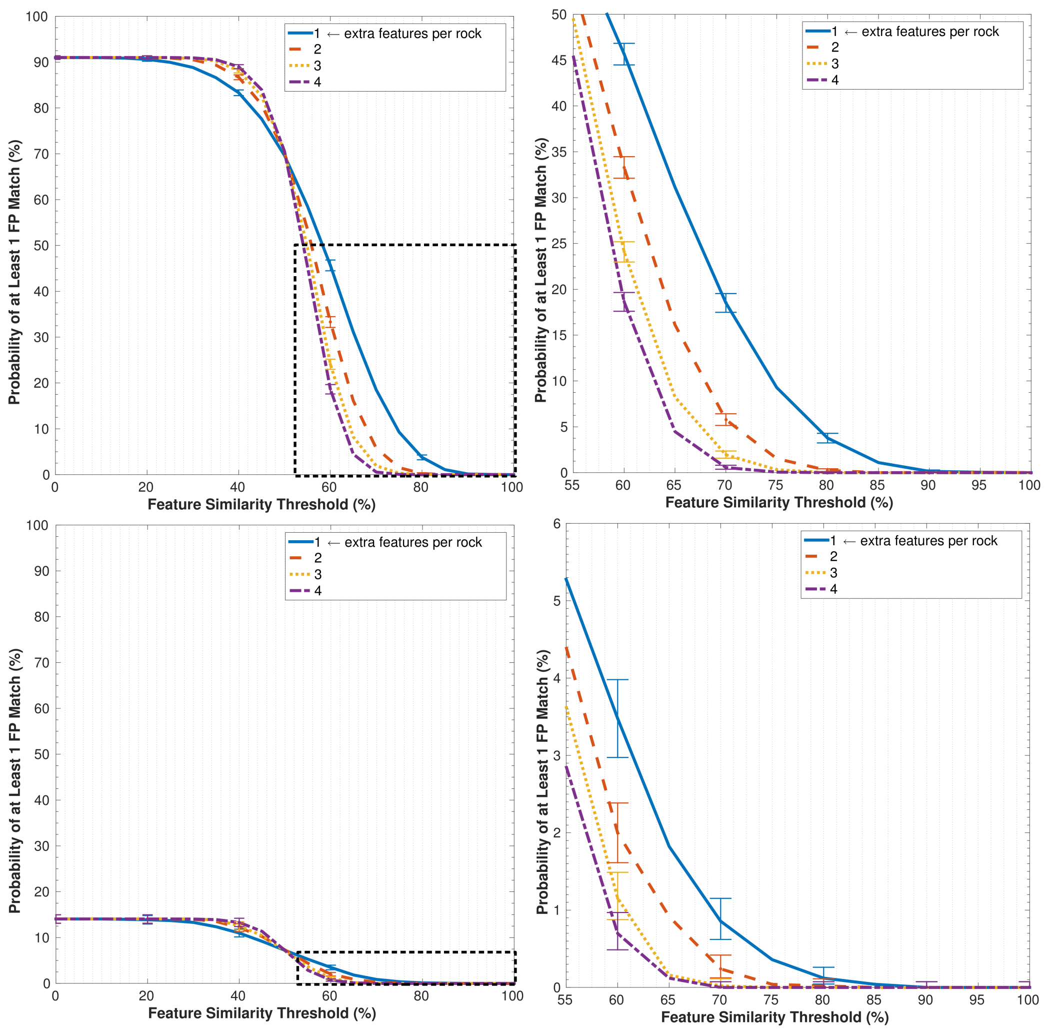

Figure 39: Probability of at least one false positive match vs. feature similarity threshold for varying numbers of **uniformly** distributed random features. The top images show the unmagnified and magnified results for 4-constellations and the bottom images show the unmagnified and magnified results for 5-constellations.

Exponentially distributed random features Most distributions formed by natural processes tend to be exponential. As an example, distributions of the surface areas and pixel areas of all rocks in the Atacama dataset are shown in Figure 40. Thus, we can repeat the same analysis as above but sample random features from an exponential distribution.

The result, in Figure 41 below, shows that this realistic assumption further improves results. The curves are moved leftward, indicating that the same false positive probability can be achieved at a lower similarity threshold.

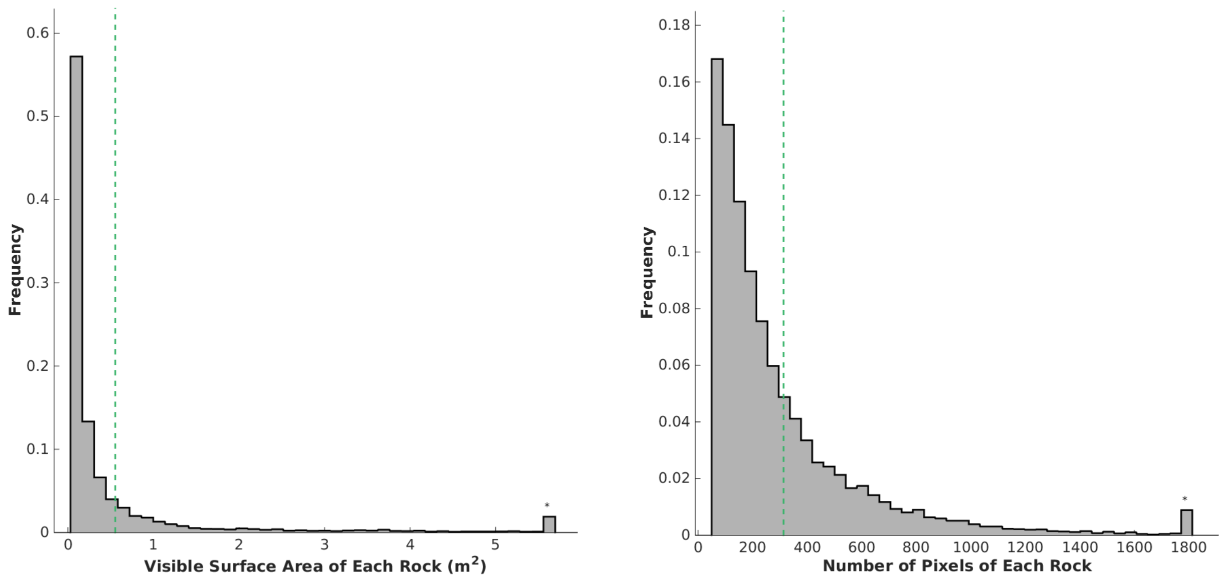

Figure 40: Distribution of rock surface area and pixel area in the Atacama dataset.

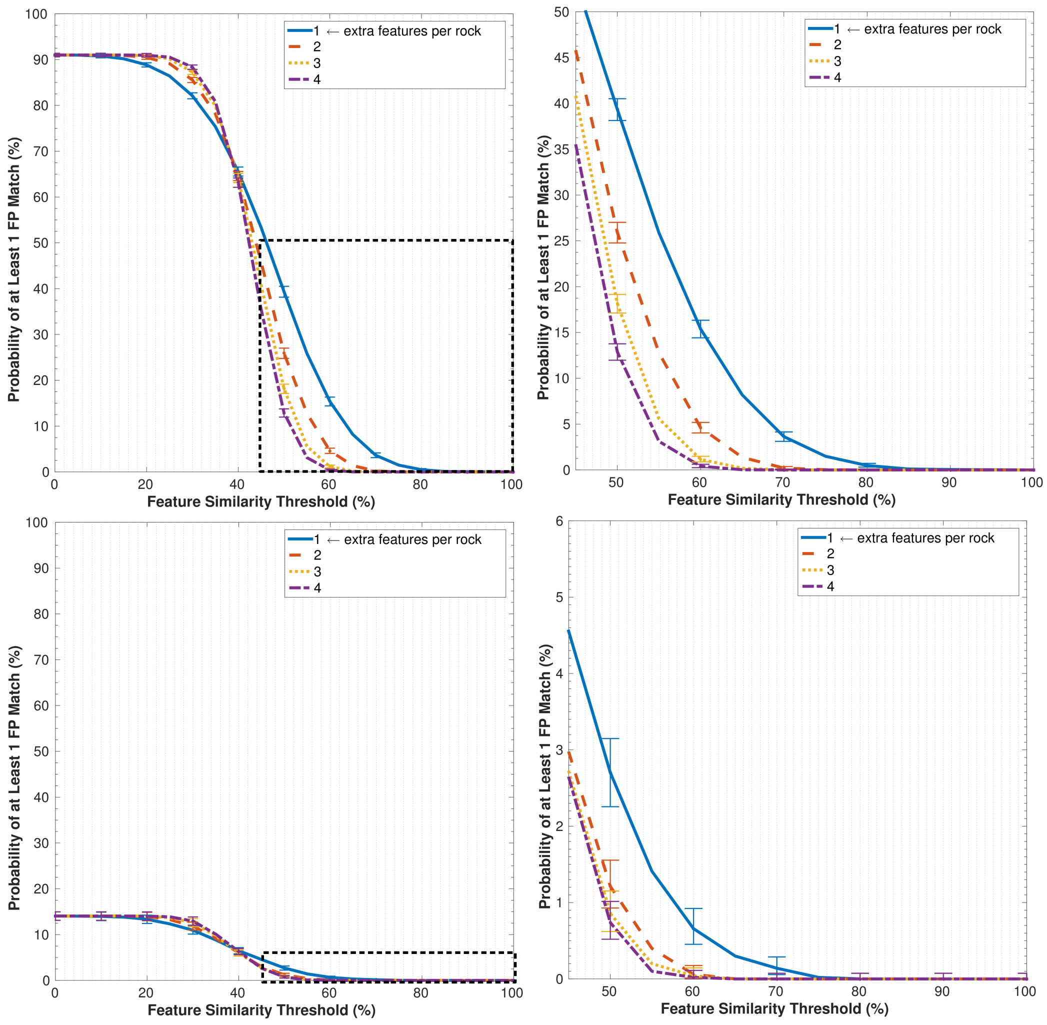

Figure 41: Probability of at least one false positive match vs. feature similarity threshold for varying numbers of **exponentially** distributed random features. The top images show the unmagnified and magnified results for 4-constellations and the bottom images show the unmagnified and magnified results for 5-constellations.

4.3.8 CHB Descriptor Computation

Computation is a reasonable concern for the CHB descriptors. Generating convex hull boundaries from Delaunay triangulations can be done very efficiently. The main computational bottleneck occurs in matching all the shifted and rotated points of one set of constellations to all the points of another. Below we analyze how this computation is affected by the size of constellations generated and the number of rocks in the scene. As it is nontrivial to analytically determine the average number of constellations generated by connecting Delaunay triangulations and extracting their convex hull boundaries, a simulation approach is used.

A set of N_r rocks is randomly generated in an FOV that matches that of Zoë. By forming the set of CHB constellations for those simulated rocks, we are able to determine how the number of constellations increases with respect to the number of rocks visible in the scene. The leftmost image in Figure 42 shows the result of this simulation. The number of constellations increases close to linearly for the range of 0 to 50 rocks. We note that for a dataset in which we expect to see less than 10 rocks in many scenes, it would be unhelpful to search for 6- and 7-constellations, as these are much less likely to be found.

From the number of constellations found, we can also estimate how computational time is affected by the number of visible rocks, which will be useful in designing future improvements to constellation matching. Consider two scenes that use the same size k-constellations and same number of constellations N_C. For the brute force method used in this work in which all constellations of one scene are compared against all constellations of a second scene, this comparison simplifies to the inner product between two large matrices for a total number of point comparisons $N_c^2 \times k^4$. We find that the average number of point comparisons made between two scenes increases quadratically as the number of visible rocks increases, as shown in Figure 42.

While the quadratic increase in computation with respect to the number of rocks in the scene is certainly undesirable, in the context of rocks in the desert the actual numbers of point comparisons made are manageable. As shown in Figure 32, the number of rocks in an Atacama scene is on average 7.4 rocks, with 78% of scenes containing less than 10 rocks and 97.5% of scenes containing at most 20 rocks. The average number of point comparisons is on the order of 10,000 between scenes of 10 rocks and on the order of 100,000 between scenes of 20 rocks, as shown in Table 4. These values are easily manageable by modern CPUs.

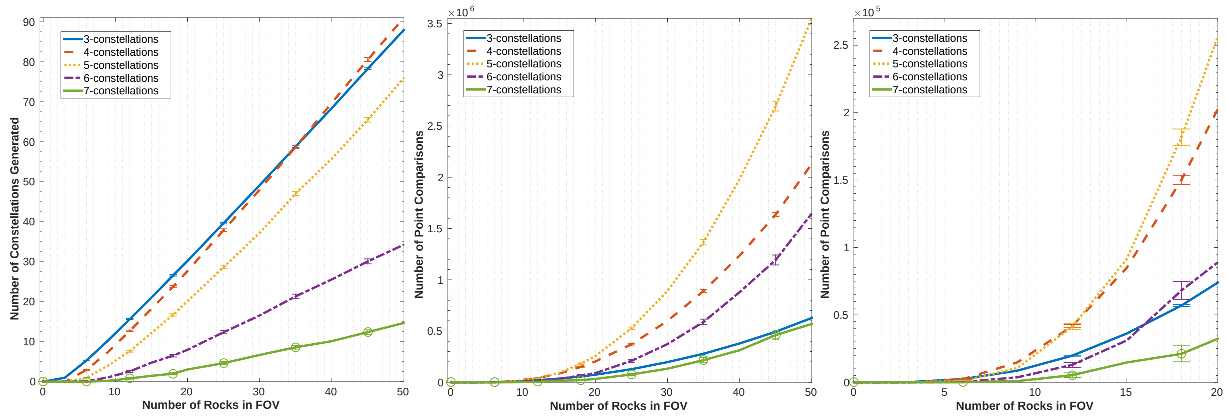

Figure 42: Number of constellations generated (left) and number of CHB point comparisons (middle) vs the number of rocks in the field of view. The figure on the right shows a magnified version of the middle plot, focusing on the range of rocks relevant to desert scenes.

Table 4: For the brute force approach to constellation matching, the average number of point comparisons is compared for various constellations sizes.

Constellation Size	Number of Point Comparisons (10 rocks)	Number of Point Comparisons (20 rocks)
3	8700	74,000
4	15,000	203,000
5	10,700	260,000
6	3700	89,000
7	900	32,000

4.4 Discussion

The analysis of constellation matching above provides useful guidelines for setting parameters to avoid false positive matches. For example, we have shown that both 4- and 5- constellations can achieve close to 0% false positive probability using

- CHB descriptors
- 2% measurement error
- Two additional features per rock at 70% similarity threshold

The measurement error and similarity threshold can be increased to further reduce false positive rate so long as sufficient recall is maintained. The value for "sufficient recall" is dependent on the dataset, odometry error, and difficulty of loop closure matches. False positive matches unaccounted for by the above process can be safely guarded against by a final RANSAC outlier rejection.

Numerous variations of the above parameters can be used to achieve similar results, and this flexibility is useful in adapting constellations to other datasets or applications. This is especially true for applications where prior information about the terrain or features is available. For example, several hundred thousand images of Mars rocks are available online, and analysis of these images could be used to train a Mars rock detector, determine the distribution of rock sizes, rock positions, and rock cluster density, or identify rock features best suited for Mars rocks. One might find that the color of Mars rocks, for example, is an unhelpful feature.

We have shown that the computational complexity is manageable for the number of rocks in typical desert scene when using a brute force computation approach. Future implementations might include a smart adaptive filter to adjust the set of rocks detected depending on the scene. For example, a scene with a large number of rocks could simply be ignored, or the rock size requirement could be increased to only allow the 10 most distinct rocks to be detected. This would constrain computational time.

For a truly scalable constellation approach, however, a more efficient database search could be implemented in the same vein as star identification methods. Intuitively, it is clear that not every pair of constellations between two scenes should necessarily be compared. The constellations could be stored in a binary search tree sorted by area to greatly reduce the number of comparisons made.

5 MeshSLAM

This section discusses the complete GraphSLAM-based localization system using constellations as landmarks, named MeshSLAM. The constellations are obtained as previously discussed in Sections 3 and 4. The change in robot poses are obtained by (1) matching features between consecutive scenes, (2) rejecting outliers using RANSAC, (3) using the absolute orientation method [28] to calculate the bearing and translation change between two sets of 3-D points.

A detailed breakdown of the MeshSLAM pipeline and its effectiveness on real datasets are shown below.

5.1 Background and Related Works

The fundamental problem of localizing a robot in an unknown environment is known as the simultaneous localization and mapping problem, or SLAM [65]. As its name implies, SLAM involves building up a map of the environment and localizing with respect to that map. More formally, given a series of sensor measurements $z_{1:t}$ and odometry measurements $u_{1:t}$, the full SLAM problem computes an estimate of the trajectory $x_{1:t}$ and map m, expressed as $p(x_{1:t}, m|z_{1:t}, u_{1:t})$. In visual SLAM, the map is a set of discrete landmarks. Thus far, discussion has focused on the front-end of the SLAM problem — feature extraction and data assocation — but below, some context is given for the back-end of SLAM optimization.

The SLAM problem has been around for nearly three decades now [40, 58]. Most early work focused on solving SLAM with extended kalman filters (EKF SLAM). However, these algorithms only solve the online SLAM problem $p(x_t, m|z_{1:t}, u_{1:t})$, and large uncertainties in the posterior or highly non-linear measurements can lead to significant inconsistencies in pose estimates [31].

5.1.1 GraphSLAM

GraphSLAM (also called *factor graph optimization* or *smoothing and mapping*) is a term used to describe SLAM algorithms that solve the full SLAM problem [65]. It is so named because the SLAM problem can be expressed naturally as a graph of relationships between odometry and landmark measurements, robot poses, and the map. The nodes of the graph are the poses or landmarks. Every edge in the graph, i.e. every measurement, is a soft nonlinear constraint, and the objective function to be minimized by GraphSLAM is the sum of all these constraints, resulting in a nonlinear least squares problem:

$$J = \underbrace{x_o^T \Omega_o x_o}_{\text{anchoring constraint}} + \underbrace{\sum_t \left[x_t - g\left(u_t, x_{t-1}\right)\right]^T R_t^{-1} \left[x_t - g\left(u_t, x_{t-1}\right)\right]}_{\text{odometry measurements}} + \underbrace{\sum_t \sum_i \left[z_t^i - h\left(x_t, m, c_t^i\right)\right]^T Q_t^{-1} \left[z_t^i - h\left(x_t, m, c_t^i\right)\right]}_{\text{landmark measurements}}$$

In the above expression, g is the nonlinear motion model, R is the covariance of motion noise, h is the nonlinear measurement model, Q is the measurement covariance, and c is a discrete correspondence variable that relates landmarks. The first term with the initial pose estimate is used to anchor the solution to absolute map coordinates. Each measurement, odometry or landmark, is a quadratic constraint, equivalent to the negative log-likelihood of that measurement given an initial estimate of the pose or landmark being measured and the measurement covariance.

EKF SLAM [40] and GraphSLAM both represent state beliefs by Gaussians. However, whereas EKF SLAM directly tracks the means and covariances μ and Σ, GraphSLAM keeps track of a different parameterization of the Gaussians, the information matrix and information filter Ω and ξ, respectively. They are related by

$$\Omega = \Sigma^{-1}$$

$$\xi = \Sigma^{-1}\mu$$

This formulation has the advantages of being (1) more computationally efficient (2) more numerically stable and (3) able to keep track of all previous trajectory and landmark information [65]. The third advantage in particular gives GraphSLAM more flexibility and robustness in handling complex trajectories and data associations.

To build up the information matrix Ω and information vector ξ, the Jacobians of the measurement models G_t and H_t are used to linearize each measurement update around its initial estimate via Taylor expansion. These linear approximations are used to update the information matrix and vector. In this way, the nonlinear least squares problem becomes a linear least squares problem of the form $Ax = b$:

$$\Omega\mu = \xi$$

The most straightforward way to extract the optimal state estimates would be through inversion:

$$\mu = \Omega^{-1}\xi$$

$$\Sigma = \Omega^{-1}$$

However, since each pose is connected only to consecutive poses and landmarks are only seen locally, the information matrix Ω is sparse, and much more efficient sparse linear algebra techniques can be used to solve the problem.

The current standard for batch optimizations of nonlinear least squares problems is the Levenberg-Marquadt algorithm. This can be computationally expensive, though, and a real-time SLAM system would require an incremental optimizer.

5.1.2 Incremental Smoothing and Mapping

Incremental Smoothing and Mapping is an efficient method of solving the GraphSLAM problem incrementally, maintaining a best estimate of the current state as new measurements arrive [32, 33]. For the online version of GraphSLAM, the nonlinear least squares problem becomes a series of linear least squares problems. In the smoothing step, the linear least squares problem is solved by using a QR or Cholesky decomposition to factorize the information matrix. A Cholesky decomposition, for example, decomposes the information matrix into

$$\Omega = R^T R$$

Then forward and back substitutions can be used to solve for the state estimate μ:

$$R^T y = \xi$$

$$R\mu = y$$

The forward and back substitutions are efficient if R is sparse, but in general this is not true. However, the variables of Ω can be reordered, e.g. using the column approximate minimum degree ordering algorithm (COLAMD), in such a way that the factorization R is sparse [14]. The original iSAM algorithm described a method of incrementally adding measurement updates directly to the factorized R matrix using Givens rotations [33]. Expensive periodic batch steps were still necessary to relinearize and reorder the variables in the information matrix to improve convergence and maintain sparsity in R.

An improved version of the algorithm, iSAM2, uses a novel data structure called a Bayes tree to represent R [32]. The factor graph can be converted to a Bayes net, and the cliques in the Bayes net can be naturally converted to a tree structure. The Bayes tree is shown to have several advantages over previous data structures. With the Bayes tree, cliques can be rearranged to produce similar results as reordering variables of the information matrix, but it can be done much more cheaply and incrementally. Additionally, the Bayes tree introduces *fluid relinearization* — by keeping track of variables that need to be relinearized, relinearization can be performed efficiently at every iteration by only affecting those variables and the cliques associated with them.

5.1.3 Powell's Dog-Leg Incremental Optimization

The original iSAM algorithm uses an incrementalized version of the iterative Gauss-Newton method to find the minimum of each new nonlinear least squares problem. However, the Gauss-Newton method, while fast, is not robust to highly nonlinear objective functions since it assumes local linearity at each iteration. A more robust method would be a steepest descent method, which iteratively converges to a solution by stepping in the direction of the largest gradient. These methods, however, have slow convergence time.

The Powell's Dog-Leg trust region method combines the advantages of the Gauss-Newton method and steepest descent method [53]. A region of trust of radius Δ is maintained around each linearization point. Optimization steps are determined by interpolating between the fast Gauss-Newton method and the more reliable steepest descent method based on the size of the trust region. At each iteration, the radius Δ of the trust region is updated based on the ratio between the actual reduction in the objective function and the predicted reduction. When the linear approximation performs well, the radius increases to allow for faster convergence.

When the Powell's Dog-Leg trust region is used with iSAM2, the algorithm is known as Robust Incremental least-Squares Estimation (RISE2) [54]. This method is used by MeshSLAM for its combination of speed and robustness to nonlinearities.

5.1.4 Georgia Tech Smoothing and Mapping Library

The Georgia Tech Smoothing and Mapping Library (GTSAM) is a library of methods and tools for graph optimization based on factor graphs and Bayes networks [15, 21]. GTSAM includes a MATLAB toolbox that was used to implement the MeshSLAM back-end in the following sections. The specific incremental solver used is iSAM2 with a Powell's Dog-Leg optimizer using default parameters.

5.2 Method

The pipeline from stereo images to SLAM update is described here. These steps are repeated at each time step.

Step 1. Compute point cloud

Th standard method is used to estimate a point cloud from stereo images: (1) determine corresponding points between the left and right images (2) rectify the images, and (3) compute the point cloud using known camera parameters.

Step 2. Estimate pose change

Pose changes are estimated from the point clouds of consecutive frames. Here, the Absolute Orientation method (AO) is used for its simplicity and effectiveness on outdoor datasets [3]. Given two sets of corresponding 3-D points, AO estimates a rotation R and translation T between the points. Lorusso [44] showed that the singular value decomposition (SVD) technique for this alignment provides the best overall accuracy and stability. The steps for the unweighted version are reproduced below [59]:

Given N 3-D points X and their corresponding 3-D points Y, with respective centroids $\bar{X}$ and $\bar{Y}$:

$$\text{Let } H = \frac{1}{N}\sum_{i}^{N}(Y_i - \bar{Y})(X_i - \bar{X})$$

Compute SVD of H:

$$H = USV^T$$

Solve for the optimal rotation R:

$$R = V^T \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & \det(VU^T) \end{pmatrix} U$$

Solve for the optimal translation T:

$$T = \bar{X} - R\bar{Y}$$

To obtain the 3-D points required for AO, features are matched between frames. In this work SURF keypoint features were used, but any type of feature that matches well between consecutive scenes could be used. Even centroids of constellations could potentially be used as keypoints, though one difficulty that might arise is that, whereas nearly all scenes contain at least three SURF keypoints, it is not as common for a desert scene to contain at least three constellations. This can lead to inaccurate odometry estimates.

RANSAC is used for outlier rejection on the matched keypoints. In each RANSAC iteration, three random matched points are selected and the SVD technique above is used to estimate a rigid body transform (RBT) to apply to all the points of the first scene [59]. This is repeated until an RBT produces an adequate number of inliers. The translation and yaw updates (x_t, y_t, θ_t) are extracted directly from the best-performing rigid body transform $(R*, T*)$.

This process is shown in Algorithm 4. In lines 1-2, the 3-D locations of features in the the image are computed. In lines 3-7, matched points that are beyond some distance threshold τ_{far} are removed, as these are more likely to generate an inconsistent RBT. In lines 8-10, RANSAC is used to fit an RBT to the matched points, and the yaw θ and translation t are computed. In lines 11-19, a check is performed to ensure enough inliers were found and that the yaw and translation between the frames are consistent with the rover's physical limitations. If they are not, re-use the previous odometry motion.

Figure 43 shows RANSAC being used on matched points in the desert to produce a consistent pose estimate.

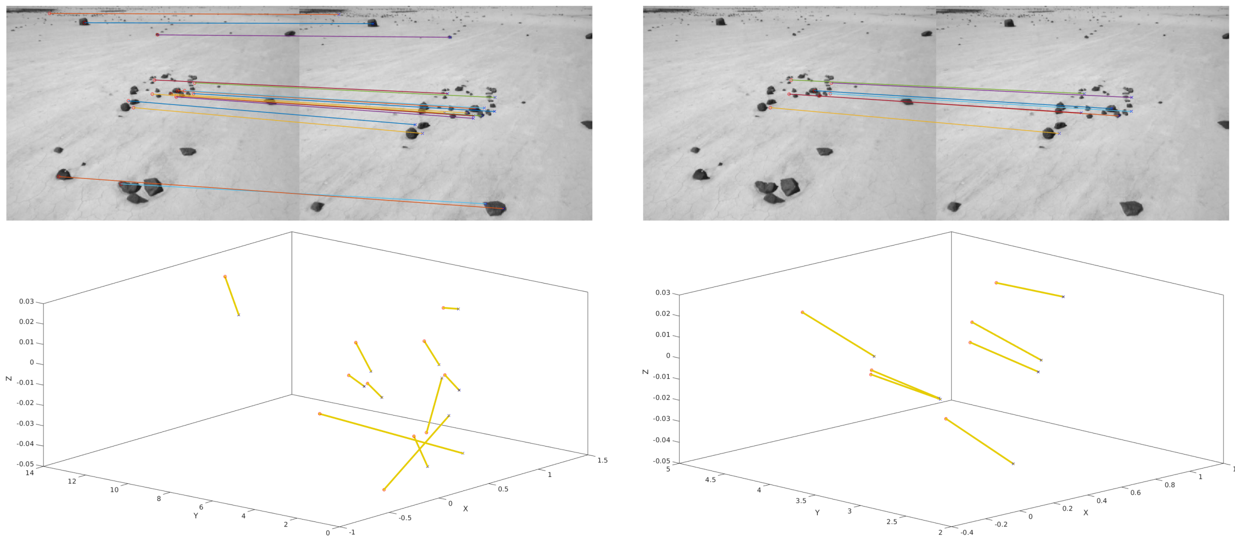

Figure 43: Matched features between frames, before RANSAC (left) and after (right). SURF features are shown on top and the corresponding matches are shown in 3-D below.

Algorithm 4 Pose update

```
Given two consecutive scenes $I_1$ and $I_2$

01.  Compute feature descriptors for $I_1$ and $I_2$.
02.  Find $n$ corresponding 3-D points $s_1$ and $s_2$ at matched
     feature locations.
03.  For $i$ ... $n$
04.    If $s_1^i > \tau_{far}$ or $s_2^i > \tau_{far}$
05.      Remove $s_1^i$ and $s_2^i$
06.    EndIf
07.  EndFor

08.  /* Use RANSAC to fit RBT between $s_1$ and $s_2$*/
09.  $inliers$, $(R,t)$ = ransac_fit_rbt( $s_1$, $s_2$, $\tau_{inlier}$ )
10.  $\theta$ = compute_yaw( $R$ )
11.  If $inliers$.length $> 3$ and $\theta < \tau_{yaw}$ and $t < \tau_{translation}$
12.    $x_t = t.x$
13.    $y_t = t.y$
14.    $\theta_t = \theta$
15.  Else
16.    $x_t = x_{t-1}$
17.    $y_t = y_{t-1}$
18.    $\theta_t = \theta_{t-1}$
19.  EndIf
```

Step 3. Update pose factors

Add the computed pose update (x_t, y_t, θ_t) as a BetweenFactorPose in GTSAM, with a Gaussian noise model based on estimated stereo error. Iterate the graph optimizer and obtain the current rover world pose X_t^W.

Step 4. Detect rock pixels

Use the trained CNN model to label rock pixels in one of the stereo images. Identify each set of connected rock pixels as a rock, and store the centroids of each rock.

Step 5. Compute rock features

For each identified rock in the label image, compute desired rock features as described in Section 4.2.3. This includes height, isolation, pixel area, and surface area.

Step 6. Build constellation descriptors

Build constellation descriptors based on convex hull boundaries from the set of rock centroids, as discussed in Sections 4.2.1 and 4.2.2. This involves (1) connecting the Delaunay triangles (2) computing the convex hull boundaries of the connected triangles and (3) storing the set of convex hull boundary points shifted and rotated to the reference frame of each point.

Two example sets of convex hull boundary constellations are shown in Figures 44 and 45 for two paths in the Atacama Desert.

Step 7. Match constellations to nearby scenes

Identify poses within some distance threshold D to the current pose X_t^W, and compare the set of constellation descriptors in those scenes with the constellation descriptors in the current scene using the matching methods discussed in Section 4.2.4. Use RANSAC and rock feature thresholds to reject outlying matches, and return the final set of verified matches.

Step 8. Update landmark factors

Compute the bearings and ranges to each of the centroids of the N_L matched landmarks, and add this set of bearings and ranges $\{(b_i, r_i), \ldots, (b_{N_L}, r_{N_L})\}$ as BearingRangeFactors in GTSAM. Iterate the graph optimizer.

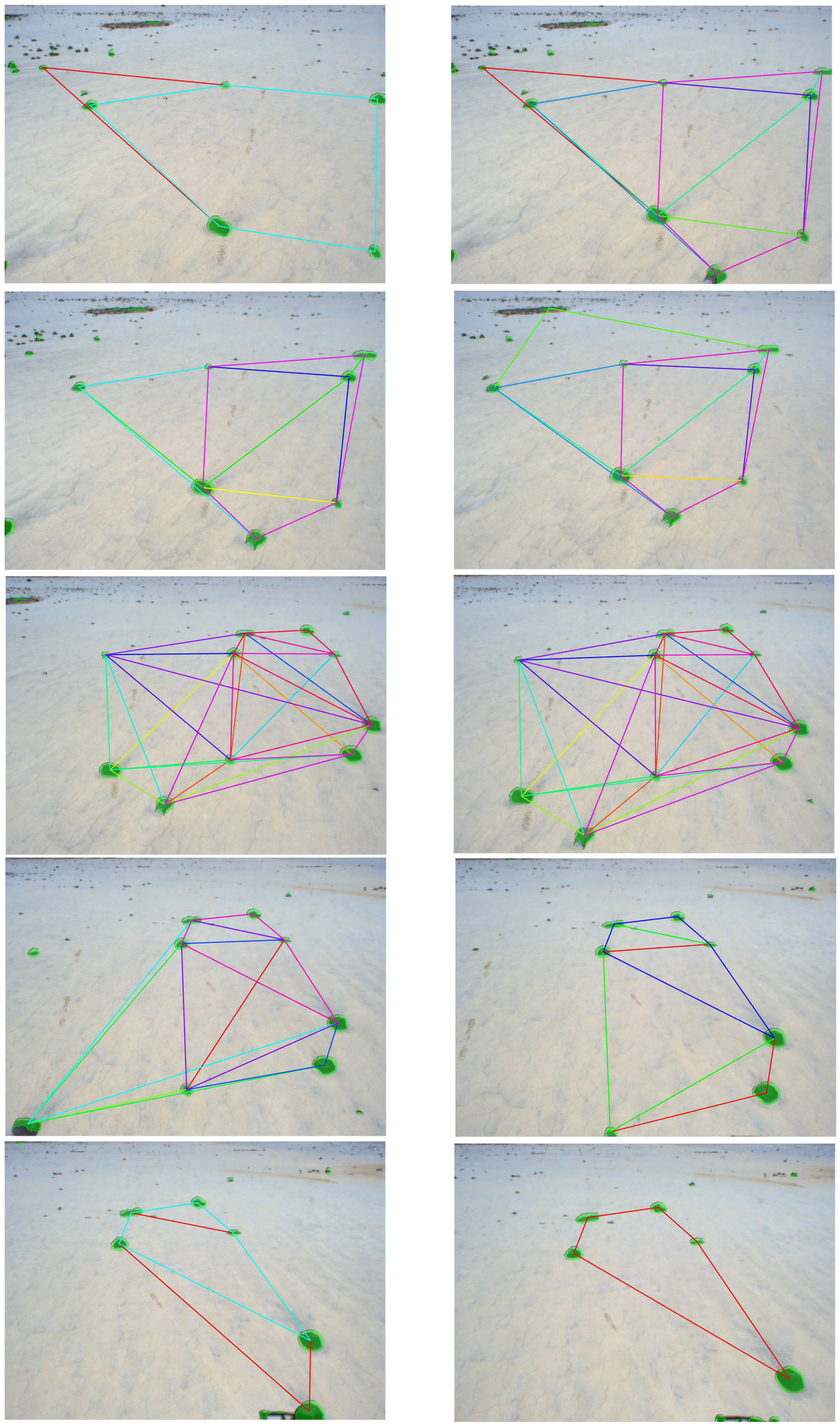

Figure 44: Sample set of consecutive frames showing detected rocks and 5-constellations built during MeshSLAM (left to right, top to bottom). Each constellation is displayed with a random color.

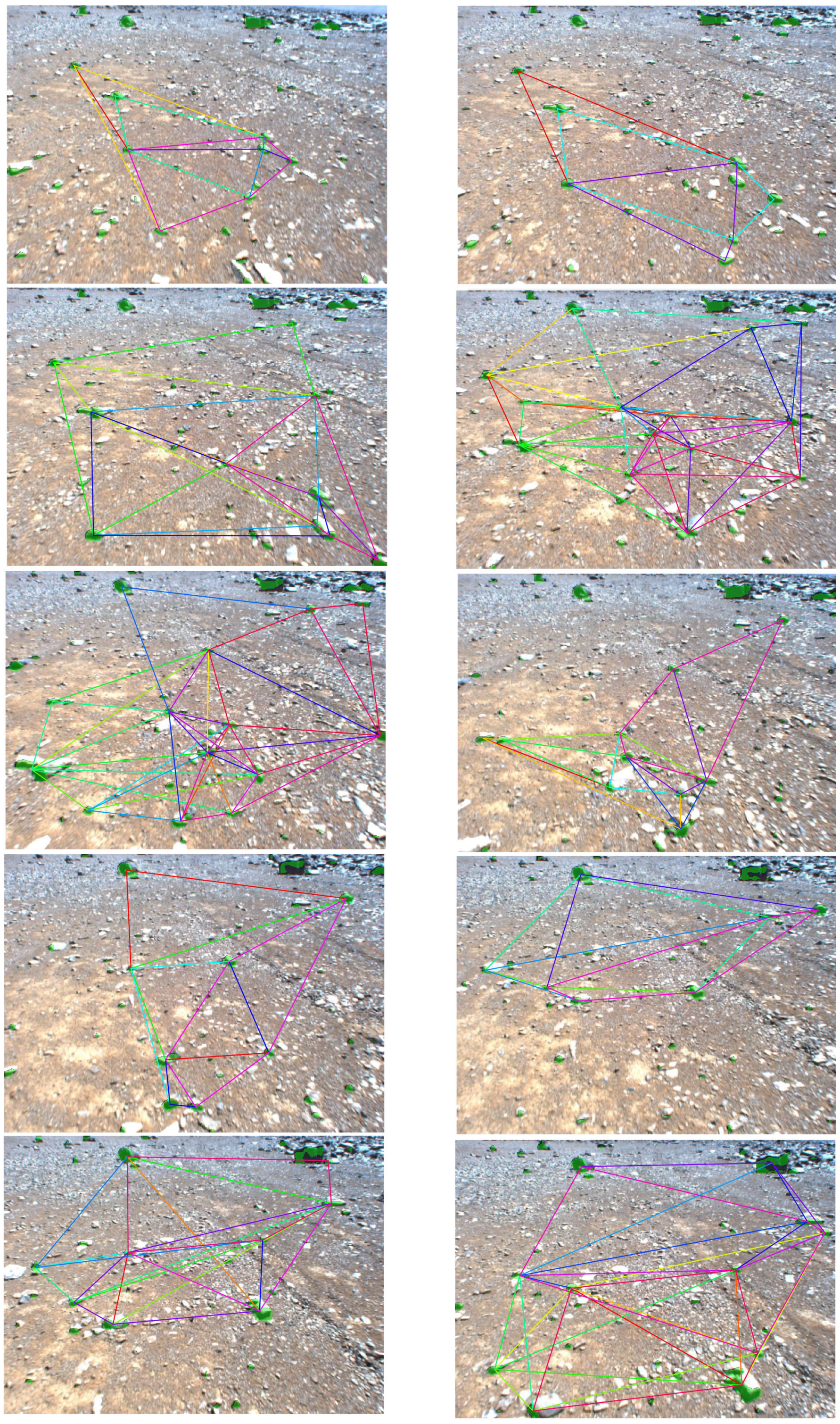

Figure 45: Sample set of consecutive frames showing detected rocks and 5-constellations built during MeshSLAM (left to right, top to bottom). Each constellation is displayed with a random color.

5.3 Results

The MeshSLAM system was tested on three paths from the Atacama dataset that contained loop closure points. They are labeled **A**, **B**, and **C** in the following figures. The results below show the ground truth data provided by the position estimator onboard Zoë, the result from relying only on the pose estimation method described above, and the result from using MeshSLAM to detect loop closure points from constellation landmarks.

5.3.1 Path A

The ground truth and pure odometry result are shown in Figure 46. Odometry works well on this strongly textured path. Figure 47 shows the progression of the MeshSLAM path. The first plot (top left) shows the path just before a loop closure point and the second plot (top right) shows the path just after the loop closure. We see that the entire trajectory is adjusted and better matches the ground truth after the loop closure correction, and covariance error around the current rover state is reduced. The third plots shows the path just before a second loop closure point and the final plot shows the result after. Again the trajectory is corrected and covariance is reduced.

Although the final path shows only small improvement over the pure odometry path in terms of accuracy, it is important to note that the confidence in its path has increased, as indicated by the overall lower covariances, and additionally the rover has gained an understanding of the topology of the map.

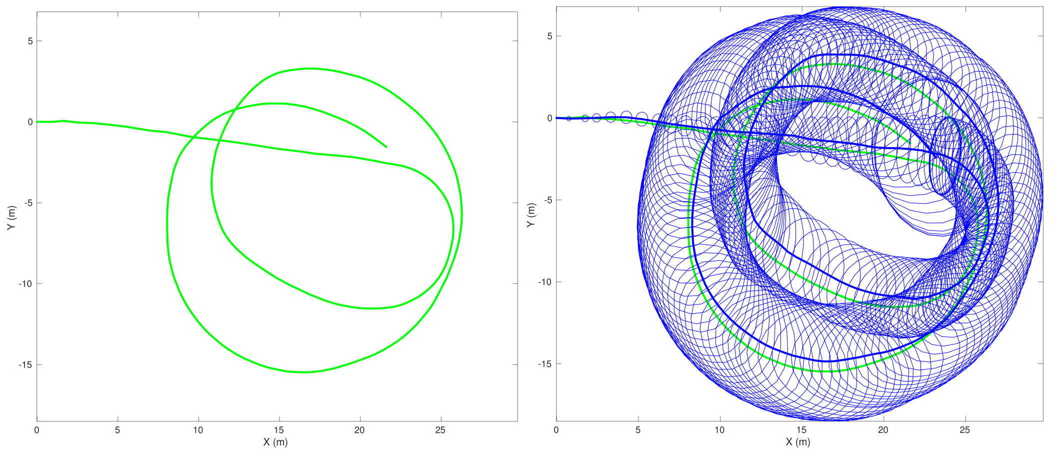

Figure 46: Path A Position estimator ground truth (left) and pure odometry (right).

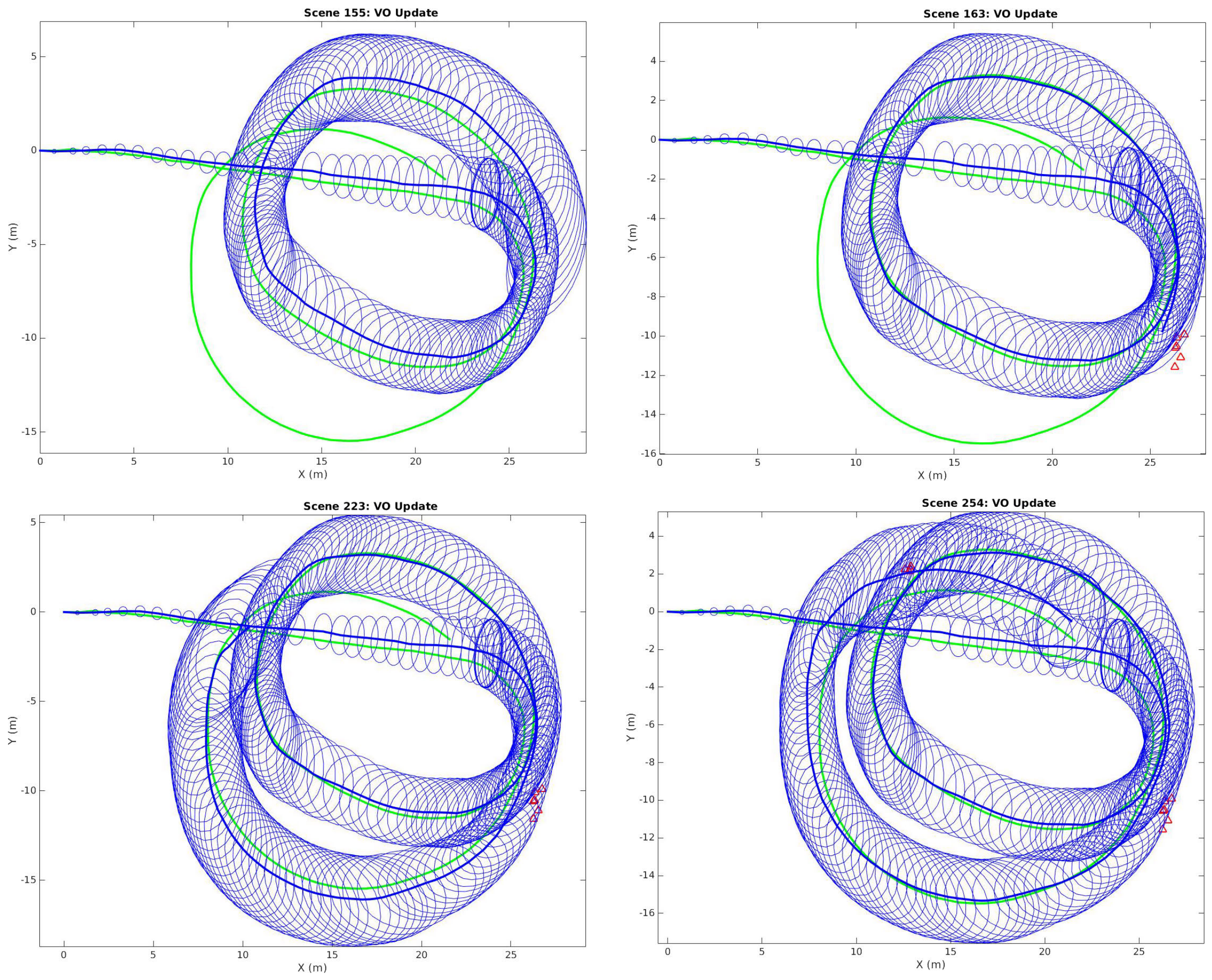

Figure 47: Path A MeshSLAM path.

5.3.2 Path B

Figure 48 shows the ground truth path and the path resulting from only odometry updates. The terrain in path **B** is sparsely-featured at times and contains bare landscape for long stretches. This makes it difficult to detect keypoint features, which is necessary the pose estimation. The result is a highly inaccurate path from only using odometry. Correction of this path is also impossible without some knowledge of the map.

Figure 49 demonstrates that even poor odometry estimates can be corrected with a few loop closure detections. The second plot shows the path just before loop closure and the third plot shows the corrected path after loop closure. The fourth plot shows the final MeshSLAM path.

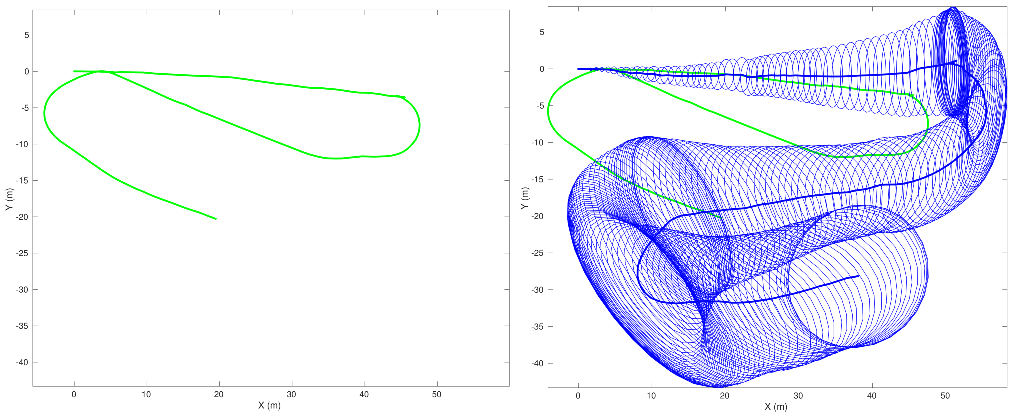

Figure 48: Path B Position estimator ground truth (left) and pure odometry (right).

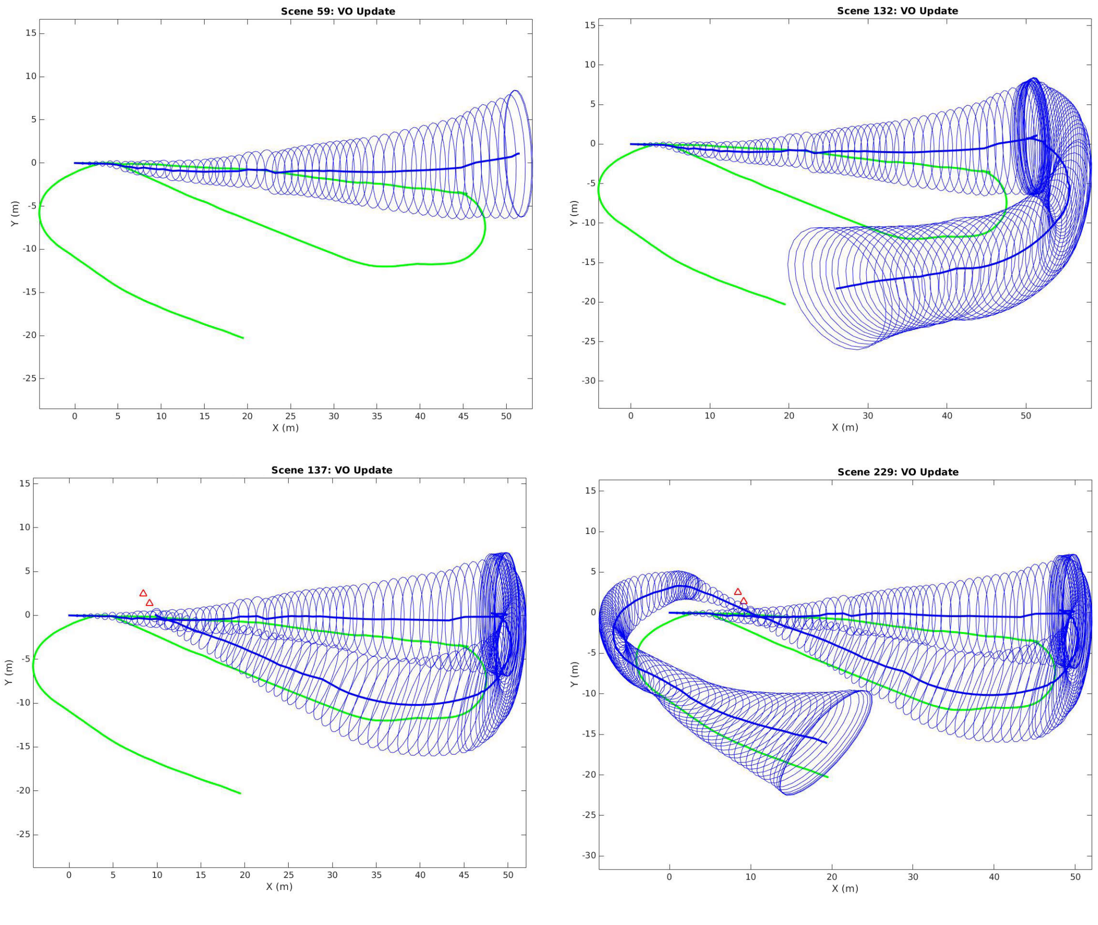

Figure 49: Path B MeshSLAM path.

5.3.3 Path C

In addition to being inaccurate, odometry-only systems maintain path errors that grow unconstrained in GPS-denied environments, as can be seen in Figure 50. Path **C** is more than twice as long as the previous two paths so the growth in covariance can be seen clearly here. Path **C** is also a well-featured environment with strong loop closure points across multiple frames, so landmark updates are obtained more frequently to produce the result in Figure 51. Landmark updates along the path that are not loop closure points generally do not cause drastic changes in the path but do help to constrain the covariance errors.

The fourth plot in Figure 51 below shows the MeshSLAM path before loop closure and the fifth plot shows the correction after loop closure. A final loop closure point is detected between plots 6 and 7, though it lasts for a single frame and provides only a small correction.

Path **C** contains some scenes with non-rock objects, people and cars, but constellations were still matched in these scenes by simply filtering out detected "rocks" above some height threshold, here 0.8 meters.

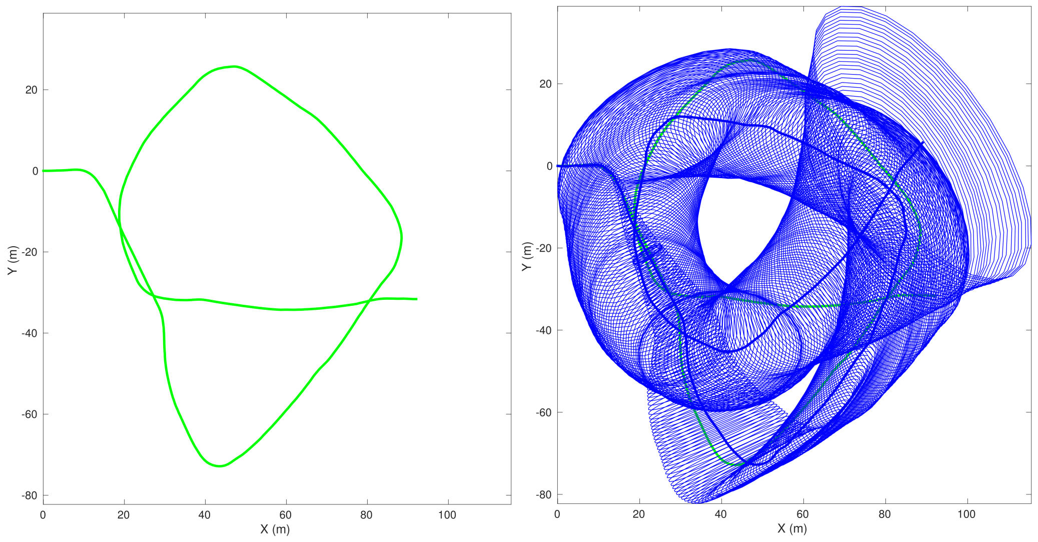

Figure 50: Path C Position estimator ground truth (left) and pure odometry (right).

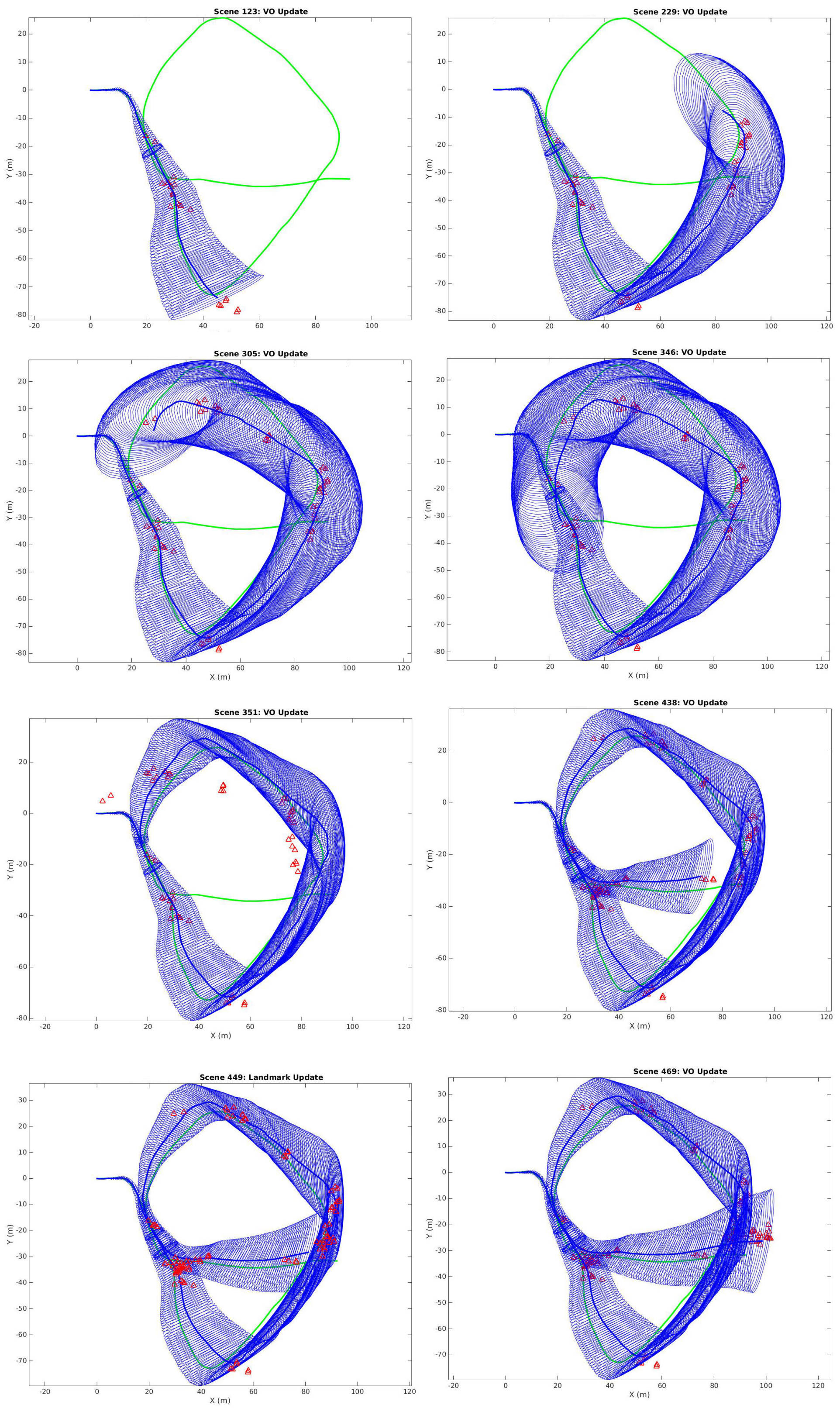

Figure 51: Path C MeshSLAM path.

Figure 52 shows a visualization of the constellation matches that occur between each scene. The left match matrix shows an approximate ground truth, obtained by setting very low thresholds to increase recall to nearly 100%, then manually checking that the resulting matches were true positive matches. The middle image shows the initial matches made by the constellation matching methods described in Section 4. Some false positives are found here, which is inevitable in some datasets as true positive loop closure matches and false positive matches are not guaranteed to be separable. However, using larger constellations or more features can increase the likelihood of separability.

The right match matrix shows the constellation matches that remain after performing RANSAC outlier rejection. Constellation matching precision of 100% is obtained indicating that no false data associations were made.

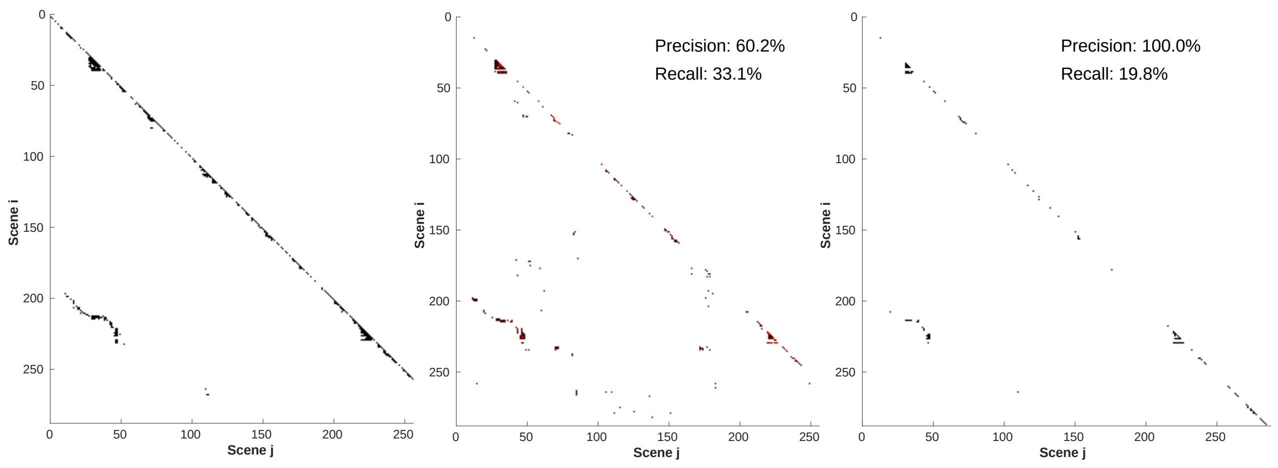

Figure 52: Matrix showing constellation matches detected between every scene. The ground truth is shown on the left, the initial set of matches is shown in the middle, and the final detected matches after RANSAC outlier rejection is shown on the right.

5.3.4 Results Summary

Figure 53 shows the ground truth, pure odometry, and MeshSLAM paths for loops **A**, **B**, and **C.** Table 5 summarizes the accuracy results in terms of mean and max position error along the entire path.

Table 5: MeshSLAM accuracy results

Path	Total Distance	Pure Odometry		MeshSLAM	
		Mean Error	Max Error	Mean Error	Max Error
A	123 m	0.9%	2.0%	0.48%	1.3%
B	140 m	10%	19%	2.4%	4.0%
C	365 m	3.5%	10%	1.4%	2.8%

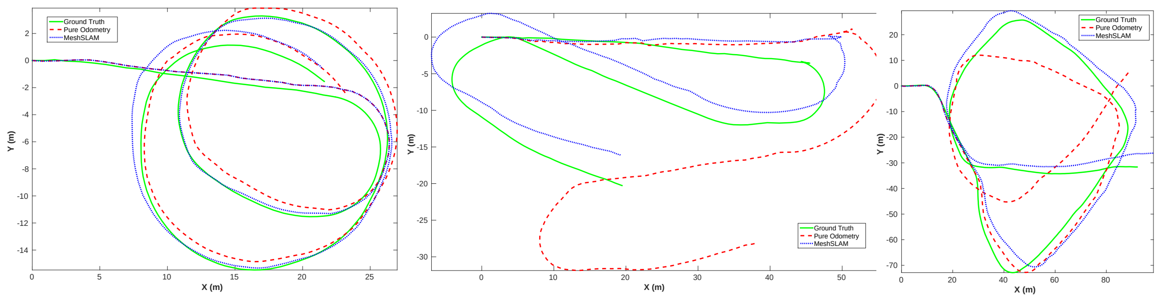

Figure 53: Comparison between ground truth (solid), pure odometry (dashed), and MeshSLAM paths (dotted).

5.4 Discussion

The results above demonstrate the potential of MeshSLAM as a localization system and the effectiveness of using constellations as landmarks in a real desert setting. The MeshSLAM system achieved an mean error of 1.4% across the three paths in the desert using only stereo images. This accuracy could be improved further by fusing other sensing modalities — IMU, wheel encoders, gyro, etc — into the odometry estimate.

We also showed that on real images, the convex hull boundary descriptors can be used with rock features and RANSAC to achieve 100% precision. On each of the paths above, the features used were sufficient to fully separate true and false positive matches.

5.4.1 Limitations

Not all of the Atacama Desert datasets produced successful results. In 6 of the 11 loops tested, the loop closure point could not be detected. In these cases, two main factors led to failure: (1) the loop closure point was only visible in one or two frames and (2) the rocks at the loop closure point were too few or too far. Without a sufficient number of non-occluded rocks, no constellations can be matched. Some other limitations of the MeshSLAM approach are described below.

Motion blur Datasets with significant motion blur are difficult to deal with when using only stereo vision. In these cases an IMU would be especially helpful.

Too few features While the MeshSLAM is designed to be used in environments with a sparse set of features, some features must still be present to form constellations. In some datasets, such as those from an expanse of dry lakebed in the Atacama, long stretches of scenes were devoid of rocks entirely and no loop closure points could be detected.

New terrain The MeshSLAM rock detector was trained on a set of images from the Atacama Desert, and so will be most accurate on similar images. Introducing the system into new terrain "out of the box" would likely not work until the rock detector CNN has been generalized to more terrains and rocks.

Parameter selection One of the open problems in SLAM is to develop automatic parameter tuning methods so that SLAM systems can be used in arbitrary scenarios [8]. Unfortunately, MeshSLAM does not have this capability and requires some parameter tuning to work well on the datasets above. Specifically, the main parameters to be adjusted are the measurement error ε_m, the RANSAC inlier threshold, and rock feature similarity thresholds.

5.4.2 Alternative Approaches

The system proposed in this work is just one specific implementation of a GraphSLAM framework that could be used with constellations as landmarks. Any of the numerous variations of GraphSLAM and visual odometry methods developed recently could be used in the back-end of MeshSLAM to further improve localization and robustness. Some examples are given below.

Robust Cost Function Most GraphSLAM algorithms are derived from the assumption of measurements with Gaussian noise, which leads to a quadratic cost function to be minimized. Outliers for a quadratic cost function can produce intolerable errors and prevent the optimizer from converging. Instead, a robust estimator such as the Pseudo-Huber loss function could be used [26]:

$$f(x) = b^2 \left(\sqrt{1 + (x/b)^2} - 1 \right)$$

The Pseudo-Huber loss function approximates the quadratic cost function for measurement errors below some threshold b, and approximates a linear cost function above b**.** Thus, b can be used to separate inliers and outliers, and reduce the effect of data association errors.

Pose Estimation An alternative pose estimation technique is the Pespective-n-Points method (PnP). This class of methods estimates the camera's pose based on n 3-D points in a scene and their corresponding 2-D projections in the camera frame. Alismail [3] showed that the PnP method (particularly P3P) produces superior accuracy to AO in a visual odometry (VO) framework. However, visual odometry accuracy is highly dependent on the environment and movement speed so the outdoor VO results provided in the work are not directly comparable to pose estimation in the desert. Regardless, the PnP method and perhaps a full VO approach using Kalman filters or Structure from Motion (SfM) is certainly worth exploring further.

Robust Back-End One of the main challenges of many SLAM systems, MeshSLAM included, is in achieving 100% precision in data association. This is necessary because data association errors are highly detrimental to graph optimization, and a major mode of failure in most modern SLAM back-ends [8, 62]. Iteratively minimizing a nonconvex cost function requires a good initial estimate. Even a single bad estimate or false data association can change the topology of the graph, resulting in further bad initial estimates and making a defective solution very likely.

Fortunately, promising work is being done in the area of robust loop closures [37, 49, 63]. By moving some of the responsibility of loop closure verification from the front-end of SLAM to the back-end, data association errors can be mitigated. One method proposed by Sünderhauf and Protzel uses *switchable constraints* [63]. A variable would be added to each loop closure constraint in the graph that could represent an outlier. Then optimization is performed over the original constraints as well as these variables, which extends the graph optimization problem to a search for the optimal graph topology.

6 Conclusions

In this thesis, we designed, implemented, and tested a method of localizing a rover in a desert environment that might generally be considered barren. We hypothesized that, while a single rock may be non-distinct, a collection of rocks can be used as a recognizable landmark for localization, and tested this hypothesis with simulations and in the context of SLAM for real desert images. The localization results using this method suggest that constellation landmarks can be reliably matched in a SLAM setting and are worth exploring further.

A summary of results as well as contributions and recommendations for future work is provided below.

6.1 Summary of Results

Rock Detection We applied various computer vision techniques to rock detection and determined that a stereo vision-based method could be used to improve current rock detection techniques. The resulting algorithm developed, called RHoCS, is based on the intuitive notion of a rock as a region of contiguous color at some height above the ground and enclosed by a boundary. The method was shown to achieve 64.5% recall on a set of ground truth labels, but was determined to be too slow for real-time use.

A convolutional neural network was trained for fast rock detection, and the pixel-level labeling of its ground truth training set was produced by the RHoCS algorithm. The CNN rock detector averaged 90.1% recall on a set of ground truth labels.

Constellation Matching Constellations are formed from groups of rocks, and various methods of generating, describing and matching constellations were discussed. The primary goal of constellation matching is to determine landmark associations in a SLAM system, which requires high precision to prevent false data associations. Monte Carlo simulations were used to test the various constellation matching methods for robustness to false positives. We showed that:

- The cosine similarity metric should be used when comparing features from a uniform random distribution, and the ratio similarity metric should be used for exponentially distributed features.
- Generating all possible constellations leads to high false positive rate even for very low measurement error.
- Constellation vector descriptors, pairwise and polar, only achieve false positive probability rates of less than 5% for similarity thresholds greater than 95%.
- Larger constellations increase robustness to false positives for all descriptors.
- Convex hull boundary descriptors out-perform pairwise and polar descriptors in precision and recall when noise is added to rocks.
- Extra features can be added to each rock to greatly reduce false positive rate, if the feature similarity threshold is set sufficiently high (> 60%).

- Rock features derived from exponential distributions further reduce the false positive rate.

SLAM Localization The full SLAM system using constellations as landmarks, called Mesh-SLAM, was tested on stereo images collected from the Atacama Desert. We showed that it could reliably detect loop closure points and avoid incorrect data associations. Without constellation landmarks, position estimates averaged 4.8% error across the three datasets tested. Using constellation landmarks reduced this error to 1.4% and, additionally, resulted in a map of the environment and reduced error along the estimated path.

6.2 Contributions

The main contributions of this work are summarized below.

The RHoCS algorithm for rock detection The RHoCS algorithm is a novel stereo vision method developed for rock detection that can describe rocks at the pixel level. It uses superpixel clustering and ground plane-fitting to identify rocks.

An application of deep learning to rock detection Various convolutional neural network architectures and CNN inputs were tested for effectiveness in detecting rocks. Obtaining a pixel-labeled training dataset is a difficult aspect of applying CNNs, but a method of automatic ground truth labeling using RHoCS was shown to be an effective way to train a CNN-based rock detector.

A method of systematically generating robust k-constellations A novel method was presented for describing a set of points by convex hull boundaries formed from connected Delaunay triangles. For a given scene of rocks, the method can be used to deterministically generate a set of k-constellations that is (1) of manageable size and (2) robust to noisy points. The constellation generation method described can be used to form landmarks in any planar environment with sparse features.

An analysis of constellation matching Most SLAM systems are susceptible to false data associations, and this problem is magnified in a desert setting. Thus, building distinguishable and reliable landmarks from a set of mostly uninteresting and similar rocks was a primary goal of this work. A Monte Carlo simulation analysis was used to compare various methods of generating, describing, and matching constellations. The analysis shows various ways of mitigating the likelihood of false positive matches.

A SLAM system that uses rock constellations as landmarks A complete SLAM system that uses constellations as landmarks was developed and tested on real desert scenes. It was shown to be capable of precise data assocations and accurate localization.

6.3 Future Work

The MeshSLAM system described in this work is a first prototype, designed to show the potential of using constellations as landmarks. SLAM in the desert remains a challenging problem and various improvements can be made at all stages of the SLAM pipeline to increase its robustness and generalizability.

The CNN model used for rock detection was trained only on Atacama Desert scenes. For a more generalized model, scenes from other deserts or rock environments should be used to train the CNN. For the Mars rover in particular, many images of Mars terrain have been collected and could be used to train a robust CNN model specifically for rock detection on Mars.

The current system relies solely on camera data, both for data association and between-pose motion. Incorporating other sensor modalities, such as an IMU or wheel encoders, would be a reasonable next step to improve pose estimation. The simple approach used here for pose estimation is sufficient for slow-moving rovers, but for generalizability a more modern visual odometry method could be applied, such as one that uses structure from motion.

Modern odometry systems have shown significant advancements in recent years, with some systems capable of errors less than 0.5% of trajectory length. However, the distinct advantage of a SLAM system is its ability to close the loop, which can dramatically correct paths and constrain errors. To take full advantage of the localization improvements provided by SLAM, it will be necessary to design intelligent trajectories that optimize the spacing or number of loop closure points. This is especially true in a desert scenario where a rover may have no reason to travel in loops. For long-term autonomy, regular path corrections are crucial to maintaining an accurate pose estimate.

As previously discussed in Section 5.4.2, using a robust cost function and a robust loop closure verification method would mitigate the effect of data association errors, reducing the burden of 100% precision in the SLAM front-end. This would have the added effect of reducing the amount of parameter tuning necessary for precise data association.

The brute force method used to compare constellation descriptors between scenes is sufficient for the Atacama Desert, but would not scale well to applications with a denser set of points. An efficient data structure could be used to store the constellations and drastically reduce the number of constellation comparisons made.

Finally, the current system has only been tested offline on series of images. Optimizing the system to work in real-time on rover-grade hardware would be a necessary step toward making the MeshSLAM system mission-ready.

6.4 Closing Remarks

Results have focused primarily on localization in the desert, but the idea of constellations as landmarks has a wide range of applicability. Any planar environment with identifiable, but non-distinct, features could be used. For example, an aerial view of trees or buildings, or an AUV's view of the ocean floor can be modelled in an analagous way to rocks in the desert. Possible applications exist at the micro-level as well, such as in cell image analysis. This is a rich area of research only in its beginning stages, and it will be interesting to see future developments to the constellation idea.

References

[1] Data rates/returns. http://mars.nasa.gov/msl/mission/communicationwithearth/data/. [Online; accessed October 30, 2016].

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[3] H. Alismail, B. Browning, and M. B. Dias. Evaluating pose estimation methods for stereo visual odometry on robots. 2010.

[4] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359, 2008.

[5] C.-A. Brust. CN24. `https://github.com/cvjena/cn24`, 2016.

[6] C.-A. Brust, S. Sickert, M. Simon, E. Rodner, and J. Denzler. Convolutional patch networks with spatial prior for road detection and urban scene understanding. *arXiv preprint arXiv:1502.06344*, 2015.

[7] N. Cabrol, D. Wettergreen, R. Whittaker, E. Grin, J. Moersch, G. Chong-Diaz, C. Cockell, P. Coppin, J. Dohm, G. Fisher, et al. Searching for life with rovers: exploration methods and science results from the 2004 field campaign of the "Life in the Atacama" project and applications to future Mars missions. 2005.

[8] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. Simultaneous localization and mapping: Present, future, and the robust-perception age. *arXiv preprint arXiv:1606.05830*, 2016.

[9] A. C. Cameron and P. K. Trivedi. *Regression analysis of count data*, volume 53. Cambridge university press, 2013.

[10] Carnegie Mellon University. Life in the Atacama. `http://frc.ri.cmu.edu/atacama/images.php`, 2013. [Online; accessed July 20, 2016].

[11] R. Castaño, M. Judd, T. Estlin, R. C. Anderson, D. Gaines, A. Castaño, B. Bornstein, T. Stough, and K. Wagstaff. Current results from a rover science data analysis system. In *Aerospace Conference, 2005 IEEE*, pages 356–365. IEEE, 2005.

[12] P. Corke, R. Paul, W. Churchill, and P. Newman. Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *Intelligent Robots and Systems*

(IROS), 2013 IEEE/RSJ International Conference on, pages 2085–2092. IEEE, 2013.

[13] CSAILVision. LabelMe toolbox. `https://github.com/CSAILVision/LabelMeToolbox`, 2014.

[14] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. A column approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 30(3):353–376, 2004.

[15] F. Dellaert. Factor graphs and GTSAM: A hands-on introduction. 2012.

[16] J. Donahue. Caffe models. `https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet`, 2015.

[17] M. S. Drew, M. Salahuddin, and A. Fathi. A standardized workflow for illumination-invariant image extraction. In *Color and Imaging Conference*, volume 2007, pages 36–41. Society for Imaging Science and Technology, 2007.

[18] H. Dunlop. Automatic rock detection and classification in natural scenes. PhD thesis, Carnegie Mellon University, 2006.

[19] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[20] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[21] Georgia Tech Borg Lab. Georgia tech smoothing and mapping library. `https://bitbucket.org/gtborg/gtsam`, 2016.

[22] S. Gilles. Robust description and matching of images. PhD thesis, University of Oxford, 1998.

[23] V. Gor, R. Castaño, R. Manduchi, R. Anderson, and E. Mjolsness. Autonomous rock detection for Mars terrain. *Space*, pages 1–14, 2001.

[24] E. J. Groth. A pattern-matching algorithm for two-dimensional coordinate lists. *The astronomical journal*, 91:1244–1248, 1986.

[25] V. C. Gulick, R. L. Morris, M. A. Ruzon, and T. L. Roush. Autonomous image analyses during the 1999 Marsokhod rover field test. *Journal of Geophysical Research: Planets*, 106(E4):7745–7763, 2001.

[26] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[27] J. Hong and J. A. Dickerson. Neural-network-based autonomous star identification algorithm. *Journal of Guidance, Control, and Dynamics*, 23(4):728–735, 2000.

[28] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[29] Jet Propulsion Laboratory, California Institute of Technology. Mars rover.

http://www.jpl.nasa.gov/spaceimages, 2015. [Online; accessed July 20, 2016].

[30] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[31] S. J. Julier and J. K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pages 4238–4243. IEEE, 2001.

[32] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, page 0278364911430419, 2011.

[33] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[34] A. B. Katake, J. Ochoa, J. Zbranek, B. Day, C. Bruccoleri, and D. Goodsell. Development and testing of the StarCam SG100: A stellar gyroscope. In *AIAA Guidance and Control Conference Exhibit*, 2008.

[35] M. Kolomenkin, S. Pollak, I. Shimshoni, and M. Lindenbaum. Geometric voting algorithm for star trackers. *IEEE Transactions on Aerospace and Electronic Systems*, 44(2):441–456, 2008.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[37] Y. Latif, C. Cadena, and J. Neira. Robust loop closing over time for pose graph SLAM. *The International Journal of Robotics Research*, page 0278364913498910, 2013.

[38] Y. Le Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jacket, and H. Baird. Handwritten zip code recognition with multilayer networks. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 2, pages 35–40. IEEE, 1990.

[39] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[40] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on*, pages 1442–1447. IEEE, 1991.

[41] C. C. Liebe. Star trackers for attitude determination. *Aerospace and Electronic Systems Magazine, IEEE*, 10(6):10–16, 1995.

[42] T. Lindeberg. Scale selection properties of generalized scale-space interest point detectors. *Journal of Mathematical Imaging and Vision*, 46(2):177–210, 2013.

[43] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[44] A. Lorusso, D. W. Eggert, and R. B. Fisher. *A comparison of four algorithms for estimating 3-D rigid transformations*. University of Edinburgh, Department of Artificial Intelligence, 1995.

[45] W. Maddern, A. Stewart, C. McManus, B. Upcroft, W. Churchill, and P. Newman. Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In *Proceedings of the Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China*, 2014.

[46] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the Mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[47] M. Maimone, A. Johnson, Y. Cheng, R. Willson, and L. Matthies. Autonomous navigation results from the Mars Exploration Rover (MER) mission. In *Experimental robotics IX*, pages 3–13. Springer, 2006.

[48] J. R. Myers, C. B. Sande, A. C. Miller, and W. H. Warren. The SKY 2000 master star catalog. *Spaceflight mechanics 1997*, pages 767–782, 1997.

[49] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. *The International Journal of Robotics Research*, 32(7):826–840, 2013.

[50] C. Padgett and K. Kreutz-Delgado. A grid algorithm for autonomous star identification. *IEEE Transactions on Aerospace and Electronic Systems*, 33(1):202–213, 1997.

[51] B. Quine and H. F. Durrant-Whyte. A fast autonomous star-acquisition algorithm for spacecraft. *Control Engineering Practice*, 4(12):1735–1740, 1996.

[52] C. Y. Ren, V. A. Prisacariu, and I. D. Reid. gSLICr: SLIC superpixels at over 250Hz. *ArXiv e-prints*, 2015.

[53] D. M. Rosen, M. Kaess, and J. J. Leonard. An incremental trust-region method for robust online sparse least-squares estimation. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1262–1269. IEEE, 2012.

[54] D. M. Rosen, M. Kaess, and J. J. Leonard. RISE: An incremental trust-region method for robust online sparse least-squares estimation. *IEEE Transactions on Robotics*, 30(5):1091–1108, 2014.

[55] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[56] T. Sasaki and M. Kosaka. A star identification method for satellite attitude determination using star sensors. In *15th International Symposium on Space Technology and Science*,

volume 1, pages 1125–1130, 1986.

[57] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[58] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*, pages 167–193. Springer, 1990.

[59] O. Sorkine. Least-squares rigid motion using SVD. *Technical notes*, 120(3), 2009.

[60] B. B. Spratling and D. Mortari. A survey on star identification algorithms. *Algorithms*, 2(1):93–107, 2009.

[61] T. Sun, F. Xing, X. Wang, Z. You, and D. Chu. An accuracy measurement method for star trackers based on direct astronomic observation. *Scientific reports*, 6:22593, 2016.

[62] N. Sünderhauf. Robust optimization for simultaneous localization and mapping. PhD thesis, Chemnitz University of Technology, 2012.

[63] N. Sünderhauf and P. Protzel. Towards a robust back-end for pose graph SLAM. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1254–1261. IEEE, 2012.

[64] D. R. Thompson and R. Castaño. Performance comparison of rock detection algorithms for autonomous planetary geology. In *Aerospace Conference, 2007 IEEE*, pages 1–9. IEEE, 2007.

[65] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[66] B. Van Pham, S. Lacroix, M. Devy, M. Drieux, and C. Philippe. Visual landmark constellation matching for spacecraft pinpoint landing. *AIAA Guid. Navigat. Control*, 2009.

[67] A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for MATLAB. 2015.

[68] VLFeat. MatConvNet: CNNs for MATLAB. `https://github.com/vlfeat/matconvnet`, 2016.

[69] K. Wagstaff, D. Thompson, W. Abbey, A. Allwood, D. Bekker, N. Cabrol, T. Fuchs, and K. Ortega. Smart, texture-sensitive instrument classification for in situ rock and layer analysis. *Geophysical Research Letters*, 40(16):4188–4193, 2013.

[70] X. Wei, G. Zhang, and J. Jiang. Star identification algorithm based on log-polar transform. *Journal of Aerospace Computing, Information, and Communication*, 6(8):483–490, 2009.

[71] G. Zhang, X. Wei, and J. Jiang. Full-sky autonomous star identification based on radial and cyclic features of star pattern. *Image and Vision computing*, 26(7):891–897, 2008.