Computer Vision for Quarry Applications

Gordon A. Christie

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Kevin B. Kochersberger, Chair
A. Lynn Abbott
Devi Parikh
Erik C. Westman

May 6, 2013
Blacksburg, Virginia

Keywords: Computer Vision, Rocks, Image-based 3D Reconstruction
Real-time Image Processing, Image Segmentation
Copyright 2013, Gordon A. Christie
Computer Vision for Quarry Applications

Gordon A. Christie

ABSTRACT

This thesis explores the use of computer vision to facilitate three different processes of a quarry’s operation. The first is the blasting process. This is where operators determine where to drill in order to execute an efficient and safe blast. Having an operator manually determine the drilling angles and positions can lead to inefficient and dangerous blasts. By using two cameras, oriented vertically, and separated by a fixed baseline, Structure from Motion techniques can be used to create a scaled 3D model of a bench. This can then be analyzed to provide operators with borehole locations and drilling angles in relation to fixed reference targets.

The second process explored is the crushing process, where the rocks pass through different crushers that reduce the rocks into smaller sizes. The crushed rocks are then dropped onto a moving conveyor belt. The maximum dimension of the rocks exiting the crushers should not exceed size thresholds that are specific to each crusher. This thesis presents a 2D vision system capable of estimating the size distribution of the rocks by attempting to segment the rocks in each image. The size distribution, based on the maximum dimension of each rock, is estimated by finding the maximum dimension in the image in pixels and converting that to inches.

The third process of the quarry operations explored is where the final product is piled up to form stockpiles. For inventory purposes, operators often carry out a manual estimation of the size of a the stockpile. This thesis presents a vision system capable of providing a more accurate estimate for the size of the stockpile by using Structure from Motion techniques to create a 3D reconstruction. User interaction helps to find the points that are relevant to the stockpile in the resulting point cloud, which are then used to estimate the volume.
Acknowledgements

Here I would like to thank all of those that have made it possible for me to reach this benchmark in my academic journey. I would like to start by thanking my committee members for helping to push me in the right direction and narrow my focus on the ideas that were most important. I would like to especially thank Dr. Kochersberger for providing me with a place to work and a way continue my education. Also, I would like to thank Dr. Abbott for always being willing to listen to my research problems share his understanding of computer vision and pattern recognition with me. Also, thanks to Dr. Parikh for sharing her knowledge of computer vision with me. Thanks to Dr. Westman for helping me to understand more about quarry operations, and providing me with resources that were vital for the credibility and completion of this thesis. I would also like to thank Roanoke Cement, Acco Stone, and Rockydale Quarries for providing the locations to test this work.

Now I would would like to thank all of my fellow and former labmates. I would first like to thank Pete Fanto for all of his help getting me started in the lab, especially considering he was no longer employed by the lab, and was working very hard to finish his thesis when I started working there. Even months after leaving Blacksburg he was still willing to help. I would like to thank Justin Stiltner for setting up the stereo boom’s mount in the back of the truck and helping to streamline the stereo boom’s functionality for its operation at the quarry, which reduced the time necessary for the image taking procedure significantly. He was also helpful for many other random things, such as fixing my bike. Thank you to Donnie Rogers and Kenny Kroeger for constructing the boxes needed to house the camera. Also, thanks to Matt Torok for letting me take over his computer once he left, and also setting up an account for me with Bundler and PMVS ready to go. This saved me a great deal of time. I would also like to thank Brad Clauss for building the mount to hold the stereo boom in a vertical orientation when driving around the quarry.

Finally, I would like to thank my family for all of the support they have given me throughout my life. Thanks to my dad for always being willing to explain things to me,
whether it be something mathematical or another completely random subject for which he somehow still seems to have a deep understanding.
## Nomenclature

### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANFO</td>
<td>Ammonium Nitrate/Fuel Oil</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge-Coupled Device</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>GigE</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>MP</td>
<td>Megapixels</td>
</tr>
<tr>
<td>PCL</td>
<td>Point Cloud Library</td>
</tr>
<tr>
<td>PMVS</td>
<td>Patch Multi-View Stereo</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Squared Error</td>
</tr>
<tr>
<td>SfM</td>
<td>Structure-from-Motion</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>USL</td>
<td>Unmanned Systems Lab</td>
</tr>
</tbody>
</table>
# Contents

1 Introduction
   1.1 Motivation .......................................................... 2
      1.1.1 Bench Face Profiling and Drilling Analysis ............... 2
      1.1.2 Oversize Rock Detection .................................... 3
      1.1.3 Stockpile Volume Measurement .............................. 3
   1.2 Contribution ......................................................... 4
   1.3 Overview of Thesis .................................................. 5

2 Background and Literature Review ........................................ 6
   2.1 3D Reconstruction .................................................... 6
      2.1.1 Feature Points ............................................... 6
      2.1.2 Structure from Motion ...................................... 9
      2.1.3 Dense Reconstruction ...................................... 10
   2.2 Bench Face Profiling, Drilling Analysis, and Stockpile Volume Measurement 10
   2.3 Identifying Oversize Rocks ....................................... 13
      2.3.1 Adaptive Thresholding ..................................... 14
      2.3.2 Preprocessing ................................................ 15
      2.3.3 Watershed Transformation .................................. 18
      2.3.4 Support Vector Machines ................................... 19
   2.4 Stereo Boom .......................................................... 20
      2.4.1 Design .......................................................... 21
   2.5 Modified Bundler ..................................................... 22
   2.6 Point Cloud Operations ............................................ 23
      2.6.1 Outlier Removal ............................................. 23
      2.6.2 Scaling ......................................................... 23
      2.6.3 Plane Fitting ............................................... 26
      2.6.4 3D Rotation .................................................. 26
3 Bench Face Profiling and Drilling Analysis

3.1 System Design ................................................................. 30
3.2 3D Reconstruction ............................................................. 32
3.3 3D Points Selection and Region Extraction ............................. 34
3.4 Borehole Placement .......................................................... 35
3.5 Displaying Boreholes ......................................................... 39
  3.5.1 3D view ................................................................. 39
  3.5.2 Top-Down View ........................................................ 40
  3.5.3 Side View ............................................................... 44
3.6 User Interface ............................................................... 46

4 Identifying Oversize Rocks .................................................... 48

4.1 System Design ............................................................... 48
4.2 Rock Segmentation and Size Analysis .................................. 51
  4.2.1 Image Acquisition .................................................... 53
  4.2.2 Preprocessing .......................................................... 53
  4.2.3 Segmentation ........................................................... 57
  4.2.4 Rock Classification .................................................. 60
  4.2.5 Region Splitting ....................................................... 62
  4.2.6 Estimation and Analysis of the Size Distribution ............... 65
4.3 User Interface ............................................................... 65
4.4 Results ................................................................. 68
  4.4.1 Machine Learning .................................................... 68
  4.4.2 Accuracy ............................................................... 70
  4.4.3 Computation Time .................................................... 74

5 Stockpile Volume Measurement .............................................. 76

5.1 System Design ............................................................... 77
  5.1.1 Image Acquisition .................................................... 77
5.2 Volume Estimation .......................................................... 78
  5.2.1 Stockpile Extraction ................................................ 78

vii
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.2</td>
<td>Mesh Analysis</td>
<td>79</td>
</tr>
<tr>
<td>5.3</td>
<td>Aerial Imagery</td>
<td>82</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Calculating the Ideal Camera Positions</td>
<td>84</td>
</tr>
<tr>
<td>5.4</td>
<td>Accuracy Assessment</td>
<td>86</td>
</tr>
<tr>
<td>5.5</td>
<td>Results</td>
<td>87</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Stockpile Reconstructions</td>
<td>87</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Analytical Models</td>
<td>87</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Accuracy</td>
<td>88</td>
</tr>
<tr>
<td>6</td>
<td>Conclusions and Future Works</td>
<td>92</td>
</tr>
<tr>
<td>6.1</td>
<td>Conclusions</td>
<td>92</td>
</tr>
<tr>
<td>6.2</td>
<td>Future Works</td>
<td>94</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Bench Face Profiling and Drilling Analysis</td>
<td>94</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Identifying Oversize Rocks</td>
<td>95</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Stockpile Volume Measurement</td>
<td>97</td>
</tr>
</tbody>
</table>

Bibliography 98

A Rock Classification Examples 102

B Split Algorithm Examples 104
List of Figures

1.1  Bench face and borehole visualization .................................................. 1

2.1  Building the Gaussian and DoG pyramids. Image courtesy of Lowe [8] [used under fair use guidelines] ................................................................. 7
2.2  SIFT algorithm flow .................................................................................. 8
2.3  Blast Patterns: Parallel pattern (left), Staggered pattern (right) .............. 11
2.4  Original image (left), oversegmentation (right) ....................................... 15
2.5  Structuring element shapes: Box (left), Disk (right) .............................. 16
2.6  Original matrix before morphological operations ................................... 17
2.7  Matrix after an erosion ............................................................................. 17
2.8  Matrix after a dilation ............................................................................. 18
2.9  Linearly separable classes (left) and non-linearly separable classes (right). The circled data points represent the support vectors. $\xi$ in the non-linearly separable example represents the slack (the distance past the margin corresponding to the class with green points). ......................................................... 21
2.10 Stereo boom ........................................................................................... 22
2.11 Building used for generating a 3D point cloud used to test the accuracy of the scale. ......................................................................................... 25
2.12 3D reconstruction of the building used for testing the accuracy (left), and the area being measured both in the point cloud and on the building. ........ 25
2.13 Example plane and normal vector ............................................................ 27
2.14 Rotating points into the Cartesian coordinate system ............................ 28

3.1  Images of the bench faces used to demonstrate image-based 3D reconstructions 30
3.2  Stereo boom setup for a ground-based vehicle (left), traffic barrel used as reference in output(right). ................................................................. 31
3.3  Flow of the bench face profiling and drilling analysis process. .............. 32
3.4  Bundler/PMVS output for the example bench faces. ............................... 32
4.11 Segmentation after the watershed transformation is applied to the gradient (left), segmentation with overlay (right). ................................................................. 58
4.12 Segmentation after eliminating small and background regions, but before eliminating regions bordering edges (left), segmentation with overlay (right). ..... 59
4.13 \( \frac{\text{region area}}{\text{Convex hull area}} \) vs. circularity after the SVM was applied. Green "*" labels represent valid segments, and red "+" labels represent invalid segments. ...... 62
4.14 Region resulting from undersegmentation, where two separate rock regions have been merged together. ................................................................. 63
4.15 Images showing the flow of the split algorithm. ................................................................. 64
4.16 User interface displaying percent passing, and no alarm. ................................................................. 66
4.17 User interface displaying percentage of rocks for different ranges of inch values, and the alarm set. ................................................................. 68
4.18 Final result of the rock segmentation and classification (left) and overlay (right). 74

5.1 RGB images of the example stockpiles. Large pile (left), medium-sized pile (middle), small pile (right) ................................................................. 77
5.2 Flow of the stockpile volume estimation process. ................................................................. 77
5.3 Example of grid used after inlier points of the stockpile projected onto the XY plane ................................................................. 79
5.4 Grid after the Z values have been averaged and cells to be filled have been identified. Blue cells contain averaged Z values, and red cells need to be filled using the k-nearest neighbor algorithm. ................................................................. 80
5.5 Illustration of where the quadrotor should position itself when imaging a stockpile ................................................................. 82
5.6 Setup of the cameras on the tripod, which is used to simulate acquiring images from a UAV in-flight. ................................................................. 83
5.7 Bundler/PMVS 3D reconstructions for the large pile (left), medium-sized pile (middle), and small pile (right) ................................................................. 87
5.8 Analytical models for the large pile (left), medium-sized pile (middle), and small pile (right) ................................................................. 88
5.9 Color coded point cloud for the small stockpile. Red represents inliers, green represents all other points, and blue represents the plane fit through the selected points at the base of the stockpile.

5.10 Poisson surface reconstruction for the inlier points of the small stockpile.

6.1 Felzenszwalb segmentation applied to the example image from Chapter 4, after morphological image processing was applied to the image.
List of Tables

2.1 Blast design formulas ................................................................. 12
4.1 Camera specifications ................................................................. 49
4.2 Final hood dimensions ................................................................. 50
4.3 Features for classification ......................................................... 61
4.4 Features used for each classifier ................................................ 69
4.5 Test accuracy for each classifier ................................................ 70
4.6 Average overall accuracies ......................................................... 71
4.7 Accuracies for capturing the maximum dimension of the rocks at different size ranges. ................................................................. 71
4.8 Accuracies for capturing the contour of the rocks at different size ranges. ................................................................. 72
4.9 Average errors for the invalid regions. ......................................... 72
4.10 Confidence intervals for accuracies ............................................ 73
4.11 Overall Accuracies ................................................................. 74
4.12 Computation times for oversize rock detection ............................. 74
5.1 Parameters for calculating ideal camera positions. ...................... 84
5.2 Results for the stockpile volume measurement of the small stockpile without the Poisson surface reconstruction. ......................... 91
Chapter 1

Introduction

There are many opportunities for computer vision systems in industrial applications. Existing vision systems are capable of providing applications involving accident prevention, while in other cases they can be used to provide statistical analysis. This thesis examines applying computer vision techniques to three different processes of a quarry’s operation. The three processes explored are the blasting process, the crushing process, and the stockpile volume measurement. This thesis will present the different vision systems implemented in order to attempt improving productivity for each process. The following terminology is used throughout this thesis for the context of blast analysis:

Bench \(\rightarrow\) Layer of the rock wall. Different benches are formed, which helps prevent rock from falling a large distance.

Borehole \(\rightarrow\) Hole that is drilled behind the bench face, which is where the explosives are placed.

Figure 1.1 helps to visualize this.

Figure 1.1: Bench face and borehole visualization
1.1 Motivation

1.1.1 Bench Face Profiling and Drilling Analysis

Blasting is one of the first processes in a quarry’s operation. The rock where this process takes place forms terraces, with each terrace referred to as a bench. Behind the bench face, vertical boreholes are drilled at various locations depending on the pattern chosen by the operators. The distance between the first row of boreholes and the bench face is known as the burden. The borehole locations and drilling angles are chosen in an attempt to minimize the error between the desired burden at each point and the actual burden. Once the boreholes are drilled, explosives can be placed inside and detonated to remove the rock.

If the borehole locations and drilling angles are not chosen properly, this can lead to an inefficient blast, where less rock is removed than desired. An even worse situation is if the chosen locations and drilling angles cause a dangerous scenario such as flyrock [1]. Flyrock is where rocks are projected outside of the danger zone, which are capable of causing damage to machines, injury to workers, or even fatalities. If operators rely on manually analyzing the bench face to determine the location and drilling angles for each of the boreholes, then it is possible that these dangerous scenarios occurring are more probable compared to having a software system perform the analysis from a 3D model of the bench face.

An image-based system capable of modeling the bench face provides a cheap way to automatically assign each borehole’s location and angle. Alternative methods such as laser profiling can be expensive, and require an operator to set up the system in the field. An image-based system can allow the operator to select camera poses in advance, drive to the site to take images, and return without ever leaving the vehicle. However, for either system there is still the problem of having to set up targets in the blast zone, so that the operators performing the blast have a reference when determining where to place the boreholes based on the output of the software.


1.1.2 Oversize Rock Detection

After the drilling and blasting, the rocks are transported along conveyor belts to the crushers, which crush the rocks into smaller sizes. The rocks exiting the crushers on a conveyor belt should not be above a size threshold specific to the crusher. If the rocks are too large then they can pose a threat to the next crushers that they are entering, and can lead to a large cost in repair and opportunity cost from the loss of production. For this reason it is important for operators to be aware of situations where oversize rocks are on the conveyor belt from anywhere within the plant. Commercial software exists that is capable of measuring the size distribution of the rocks, such as Wipfrag and Split Online [2, 3], however little information about how the software is operating is provided.

Current methods for determining the size distribution of the rocks on the conveyor belt involve performing what is known as a belt-cut. This is where the conveyor belt is stopped, and the rocks are placed into different buckets through a manual sieving process. Each bucket contains rocks within a certain size range. This method can be time consuming, and only captures the size distribution for a small sample of the rocks.

A computer vision alarm system using 2D imagery provides a way to monitor the number and maximum dimension of these rocks without interrupting the process. What is generally known as the Brazil nut effect [4] describes how larger particles tend to end up on top when the mixture is shaken. This is known to be a source of error for attempting to measure the overall size distribution of the rock pile [5, 6], as most of the smaller rocks have moved to the bottom. For attempting to identify the majority of the larger rocks, however, this effect is actually quite fortunate.

1.1.3 Stockpile Volume Measurement

The last process of a quarry’s operation explored in this thesis is the volume measurement of a stockpile. Once the rocks have been crushed and separated into different sizes, they form stockpiles, which is where the final product is located. These stockpiles can be very large, and measuring the volume often relies on an estimation by someone on-site. An image-based
system capable of modeling the stockpile provides a cheap way to automatically determine the size of the stockpile with minimal user interaction. This provides a more accurate estimate of the stockpile’s volume than estimating the volume using different shapes, such as a cone, as it accounts for variations in the surface of the pile. Manual estimates become significantly more challenging when the stockpiles begin to take on shapes that are difficult to model with simple equations.

For the examples explored in this thesis, the images are taken from two-camera systems. For one of the examples, the cameras were mounted in the back of a pickup truck. This becomes a problem when the stockpiles become very large, or when the stockpiles are close to a wall, which would not allow a truck to travel all the way around the stockpile, and therefore not be able to construct a 3D model to feed to the volume calculation. One solution to this problem would be to deploy a UAV, such as a quadrotor to capture all sides of the stockpile. However, due to current FAA regulations at the time this thesis was written, this was not an option. To prepare for changing regulations, another example pile was used, where the cameras were mounted on a tripod to simulate flight imagery from a UAV.

1.2 Contribution

This thesis makes three contributions to the fields of computer vision and civil engineering. All three of these contributions involve applying computer vision to processes of a quarry’s operation in an effort to improve overall efficiency. The first contribution uses a two-camera system mounted in the back of a pickup truck that, from multiple viewpoints, takes images of the bench in which the drilling and blasting will take place. Open-source image-based 3D reconstruction software is then used to create a 3D reconstruction of the bench face. Algorithms were then developed to analyze the bench face, and suggest to the operator locations and drilling angles for each of the boreholes. The second contribution is a 2D image-based system to analyze the size of rocks exiting the crushers, and report when too many size threshold violations take place. This system keeps a monochrome camera at a fixed distance from the conveyor belt so that pixel distances can be converted to distances in
The system works by taking an image, segmenting each rock in the image, and then eliminating regions that are not believed to be rocks through machine learning techniques. Regions that are eliminated are possibly split into multiple regions if the system believes there may be an undersegmentation. The algorithm incorporates adaptive thresholding, morphological image processing, the watershed transformation, a novel split algorithm, and machine learning. The third contribution is an image-based system capable of generating a 3D reconstruction of a stockpile by circling the pile and taking images from different viewpoints. Using the 3D reconstructions of the stockpiles, algorithms were developed to determine the volume of the stockpile. For bench face profiling and drilling analysis, as well as the oversize rock detection, separate user interfaces were developed in MATLAB to make the debugging process easier, and also for potential future use by operators at the quarry.

1.3 Overview of Thesis

This thesis starts by providing the reader with a high level understanding of three different processes of a quarry’s operation, where the processes are drilling and blasting, crushing, and stockpile volume measurement. It is then proposed how a vision system in each of these processes could help contribute to an overall improvement in efficiency. After this introduction, some of the prior work and existing commercial solutions are explored. The vision systems implemented for each of the three processes will then be presented in three different chapters. The hardware, algorithms, and user interface for each system will be explored in depth in these chapters. The results of each of the systems are then presented, which includes all 3D reconstructions, accuracy assessment, and all other outputs related to each of the three systems. This thesis will then finish with the conclusions each system, including where it performed well, and also where improvements need to be made. Finally, potential future works will be discussed that could be carried out to improve the performance of each system. Other applications will also be discussed that could be made possible using the algorithms developed for this thesis.
Chapter 2

Background and Literature Review

2.1 3D Reconstruction

Image-based 3D reconstruction is an essential part of the bench face profiling and drilling analysis and stockpile volume measurement work presented in this thesis, since the quality of the analysis is dependent upon a 3D reconstruction that accurately represents the scene. This method of reconstruction provides a cheaper alternative to other methods of generating a 3D point cloud, such as laser scanning. Both sparse and dense 3D reconstructions are used to create the 3D point cloud for the scene, where the dense reconstruction builds upon the sparse reconstruction. The Structure from Motion (SfM) [7] method is used in this thesis to perform the sparse reconstruction. The input to SfM is a sequence of images with sufficient overlap of the desired object(s) to be reconstructed in 3D. These images can be taken from a single camera, so long as the camera is rotated and translated through space when taking images. The algorithm operates by identifying feature points across multiple images, and then finding the 3D positions, within a local coordinate frame, for the feature points and the camera positions.

2.1.1 Feature Points

Before examining how SfM and dense reconstructions work, the concept of feature points should first be explained. Feature points are distinct parts of an image, which can potentially be recognized across multiple images. Ideally, these feature points would be invariant to changes in scale, rotation, illumination, etc, so that images can be taken of an object from different viewpoints, and still have many matching feature points identified in most of the
images. A popular feature type is the Scale Invariant Feature Transform (SIFT) [8], which is invariant to scale, translation, rotation, and is partially invariant to changes in illumination. SIFT, introduced by Lowe, was a seminal piece of work, which is used for localization in robotics, object recognition, image stitching, and many more research areas. The primary interest of SIFT with respect to this thesis is that it is at the heart of the SfM algorithm.

The SIFT algorithm starts by creating a scale space. The scale space is where we represent the image at different sizes, and with different amounts of Gaussian blur. The set of images at each level are referred to as octaves, where the images at each octave are of the same size. The next step is to calculate the Difference of Gaussians (DoG), which is used to find the feature points. Using the DoG images with multiple octaves is what provides the scale invariance. Ideally we would use the Laplacian of Gaussian (LoG), which involves taking second order derivatives, However, this is computationally intensive, so DoG is used instead, which is approximately equal to the LoG. This only involves subtracting two Gaussian images, which is easy to compute. Figure 2.1 helps to visualize how the Gaussian and DoG images are represented.

![Building the Gaussian and DoG pyramids. Image courtesy of Lowe [8] [used under fair use guidelines]](image)

Figure 2.1: Building the Gaussian and DoG pyramids. Image courtesy of Lowe [8] [used under fair use guidelines]

Once the DoG images are computed, the algorithm will iterate through each DoG image, locating local maxima and minima in the current and neighboring images in the pyramid, which we use to identify as the initial keypoints. The problem is that this produces a large number of keypoints, some of which are not desired. To eliminate these undesired keypoints,
edge and low contrast keypoints are eliminated. Once these keypoints are identified, magnitudes and orientations are assigned to them. We can then assign a descriptor, which can be thought of as the fingerprint for the keypoint. Figure 2.2 shows the flow of the SIFT algorithm.

Another common feature point found in computer vision applications is the Harris Corner Detector, which is used in the dense reconstruction algorithm presented in this chapter. The Harris Corner Detector builds upon Moravec’s Corner Detector [9], which is noted for being anisotropic, having a noisy response due to the binary and rectangular window, and responding too readily to edges. The details of the Harris Corner Detector algorithm, including the changes made to Moravec’s Corner Detector to improve upon the aforementioned weaknesses, can be found in [10]. Other examples of feature points include those proposed by Shi and Tomasi [11] and SURF [12].
2.1.2 Structure from Motion

Structure from Motion creates a sparse reconstruction, which is generated from a series of images taken of a scene from multiple viewpoints. This sparse reconstruction is created by locating 2D feature points across multiple images and then projecting them into 3D space. An open-source implementation of SfM by Snavely, called Bundler[7], is used for the work presented in this thesis. Bundler was created as part of a photo tourism project, which aimed at using a large collection of photographs of a scene, possibly from an Internet photo-sharing site, in order to generate a 3D point cloud for that scene.

Bundler begins by identifying the SIFT feature points in each image. SIFT points are then matched between each pair of images by identifying the approximate nearest neighbors. RANSAC [13] is then used to compute the fundamental matrix for each pair of cameras. Matches that are outliers to the fundamental matrix are then discarded, and if the number of remaining matches is too low, then all matches are removed. Matches are then organized into what are called tracks, where a track is a set of matches across multiple images. Tracks that contain more than one keypoint in the same image are considered inconsistent.

The next step is to identify 3D locations for each of the tracks, so that the reprojection error is minimized. The reprojection error is the sum of the distances between the projected points from 3D back to 2D and the corresponding image features. This becomes a minimization problem, where it is possible to get stuck in bad local minima. To provide better initial estimates, Bundler uses an incremental approach to estimating the camera parameters. Ideally, the first pair of cameras chosen should have a large number of matches and also a large baseline. After this, another camera is added to the optimization. A camera is chosen that observes the largest number of tracks whose 3D locations have already been estimated. The initial extrinsic parameters are estimated for this camera using a direct linear transform technique inside a RANSAC procedure, which also gives an estimate of the intrinsic parameter matrix. This matrix, along with the focal length estimated from the EXIF tags of the image are used to initialize the focal length of the camera. Once this is complete, more tracks are added, where a track is added if it is observed by at least one of the other recovered cameras.
This procedure is repeated until no remaining camera observes one of the reconstructed 3D points. A more detailed description of this process can be found in [7].

### 2.1.3 Dense Reconstruction

Once a sparse reconstruction is created using SfM, a dense reconstruction is used to fill in some of the missing regions. The dense reconstruction outputs rectangular patches that can be seen in multiple images. The opensource dense reconstruction software used in the work presented in this thesis is Patch-based Multi-view Stereo (PMVS) [14]. This is implemented with a match, expand, and filter procedure starting from the Bundler sparse output. False matches can be removed by examining photometric consistency and visibility constraints. The first part of the algorithm is the matching, where the Harris Corners and Difference of Gaussian operators are matched across multiple images. This provides a sparse set of patches, which correspond to the salient image regions. With these initial set of patches, the expansion and filtering processes are then repeated \( n \) times. Furukawa and Ponce used \( n = 3 \) in their experiments. The expansion step involves spreading the initial patches to nearby pixels in order to form a dense patch, where an expansion is not made to a region where a patch already exists. The last step of the algorithm is the filtering process, where visibility constraints are used to eliminate incorrect patches. One example of this filtering step is where for each patch \( p \), adjacent patches in all of the other images in which \( p \) is visible are collected. If the proportion of patches that do not neighbor \( p \) is too high, then \( p \) is eliminated. More details of PMVS can be found in [14].

### 2.2 Bench Face Profiling, Drilling Analysis, and Stockpile Volume Measurement

SfM and dense reconstruction techniques have been used for applications in many different research areas. However, applying these techniques to the task of profiling the bench face of a quarry to determine the borehole locations and drilling locations, and also measuring the
volume of stockpiles is a fairly novel research area. Commercial software currently exists, BlastMetrix3D by 3GSM [15], which is capable of performing these tasks. Their system for drilling and blasting analysis operates by taking at least two photos with a pre-calibrated SLR camera. Multiple targets are setup, at both the top and bottom of the bench face. This includes a range pole, consisting of two targets separated by a fixed distance to provide scale information. Once the images are taken, a 3D point cloud is created and analyzed in order to suggest borehole placements and drilling angles. BlastMetrix3D also mentions that they provide software capable of calculating the volume of stockpiles, however this seems to involve setting up targets. As expected, however, the documentation does not provide any information about the algorithms used to analyze the bench face or calculate the volume of the stockpiles.

Many factors are taken into account when determining borehole locations and drilling angles with respect to the angle of the bench face. Some of these factors include: bench face height, burden, rock density, and the timing involved with the blast. Examples of typical blast patterns are provided in the production operations section of the SME Mining Engineering Handbook[16]. Two of these patterns are provided in Figure 2.3, where $B$ is the burden (the distance between the rows), and $S$ is the spacing between the boreholes within each row.

![Figure 2.3: Blast Patterns: Parallel pattern (left), Staggered pattern (right)](image)
The SME Mining Reference Handbook [17] provides the formulas necessary to calculate the burden, spacing, subdrilling, and stemming length, which are shown in Table 2.1. Subdrilling is the distance below the floor of the current level that the boreholes are drilled. Stemming refers to the process of packing material on top of the explosive to fill up the rest of the borehole.

Table 2.1: Blast design formulas

<table>
<thead>
<tr>
<th>Burden</th>
<th>( B = K_B )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spacing</td>
<td>( S = K_SB )</td>
</tr>
<tr>
<td>Subdrilling</td>
<td>( J = K_JB )</td>
</tr>
<tr>
<td>Stemming</td>
<td>( T = K_TB )</td>
</tr>
</tbody>
</table>

where the constants \( K_B, K_S, K_J, K_T \) are the constants defined by the operator, which are dependent upon the type of explosive, rock density, and initiation. Defining \( \rho \) as the rock density, if ANFO is being then \( K_B \) is determined by

\[
K_B = \begin{cases} 
22, & \text{if } \rho < 2.7 \text{ g/cm}^3 \\
30, & \text{otherwise}
\end{cases} \tag{2.1}
\]

If higher explosives as used, such as dynamite, then \( K_B \) is determined by

\[
K_B = \begin{cases} 
27, & \text{if } \rho < 2.7 \text{ g/cm}^3 \\
35, & \text{otherwise}
\end{cases} \tag{2.2}
\]

The SME Mining Reference Handbook then defines \( K_S \) to be between 1 and 2 depending on the initiation. It then lists \( K_J = 0.2 \) to 0.5 with an average of 0.3, and \( K_T = 0.5 \) to 1.3 with an average of 0.7.

Other solutions to bench face profiling are provided with the use of laser profiling. Laser Technology Inc provides one such existing solution, which uses an impulse laser combined with software to profile the wall from a single location [18]. The problem with solutions such as this one, however, are that using laser technology is substantially more expensive than
image-based systems. This system also requires the operator to be outside of the vehicle while profiling the bench face, which might mean being near the blast zone.

### 2.3 Identifying Oversize Rocks

Much work has been done in the area of rock size analysis. This ranges from the first image based analysis in 1976 [19] to the use of laser triangulation for capturing 3D surface data [5, 6]. The results by Thurley show that a highly accurate segmentation is possible with the use of laser triangulation [20]. Another algorithm capable of real-time size analysis, also based on laser triangulation, was developed [6]. However, the measurement devices used for laser triangulation are more expensive than most machine vision cameras, which makes support for such a system more challenging. For these reasons, it was determined that the use of 2D imagery was desirable for identifying oversize rocks.

Mkwelo et al. use an adaptive thresholding technique along with a bilateral filter for the preprocessing of the image. They then perform classifier training in order to eliminate erroneous regions [21]. However, at the time Mkwelo found that the average time to process a 236 by 250 image was far too slow for a real-time system [22]. Dunlop explored a way to identify rocks in their natural environment and then classify the type of rock based on various texture features, pixel statistics, and shape measures [23]. Some of these features are explored in this thesis for the purpose of classifying segments as a rock or non-rock.

The more specific task of identifying oversize rocks has also been explored. Cabello et al. use thresholding, erosion, and tracking over multiple images to identify oversize rocks [24]. However, using only thresholding and erosion is not reliable, as this does not preserve the shape of the rock. This technique is also likely to result in undersegmentation if two regions merge. Al Modarresi et al. also explore identifying large rocks [25], but give no information about lighting conditions, computation time, or how the quantity of the larger rocks relates to the overall size distribution of the rocks. A few commercial products for analyzing the size distribution of rocks moving on a conveyor belt also exist. These include Split-OnLine and Wipfrag Momentum [3, 2]. However, they do not give much information about the inner
workings.

2.3.1 Adaptive Thresholding

Adaptive thresholding is the process of finding a threshold that is used to convert a grayscale image to a binary image. This can be used to eliminate background regions, which can facilitate segmentation and classification algorithms. With the rocks on top of the moving conveyor belt, there are many openings that form between neighboring rocks that provide a window to a lower part of the pile or possibly the conveyor belt. In the image, these regions are usually very dark. Without adaptive thresholding it is possible for undersegmentation to result, where two separate rock regions would merge. If not eliminated, these background regions can also form their own segment. The machine learning algorithms would most likely eliminate these segments, but this increases the computation time substantially, as all of these regions then have to be processed. For these reasons, adaptive thresholding becomes an important part of the overall performance of the segmentation.

Otsu’s method of adaptive thresholding [26] was used in this thesis, which attempts to maximize the separability of the gray level classes. This method starts by representing the image with \( L \) gray levels, starting at 1, where \( n_i \) represents the number of pixels at level \( i \). This histogram is then normalized and treated as a probability distribution where

\[
p_i = \frac{n_i}{N} \quad (2.3)
\]

Here \( N = n_1 + n_2 + ... \ n_L \) represents the total number of pixels in the image are then separated into two different classes \( C_0 \) and \( C_1 \), which are separated by a threshold \( k \). \( C_0 \) represents all pixels with a level at or below \( k \), and \( C_1 \) represents all pixels with a level above \( k \). In order to assess how good the threshold value is, measures of class separability are used for discriminant analysis. \( \sigma_W^2 \), \( \sigma_B^2 \), and \( \sigma_T^2 \) represent the within-class variance, between-class variance, and the total variance of levels, respectively, where

\[
\sigma_T^2 = \sigma_W^2 + \sigma_B^2 \quad (2.4)
\]
Otsu’s chosen measure for the separability of the classes is

\[ \eta(k) = \frac{\sigma_B^2(k)}{\sigma_T^2} \]  \hspace{1cm} (2.5)

This measure was chosen for its simplicity, where \( \sigma_B^2 \) is based on class means, and \( \sigma_T^2 \) is independent of \( k \). The optimal threshold \( k^* \) is then found, which maximizes \( \eta \). This also means maximizing \( \sigma_B^2 \), so our optimal threshold is found by

\[ \sigma_B^2(k^*) = \max_{1 \leq k \leq L} \sigma_B^2(k) \]  \hspace{1cm} (2.6)

More details for this method can be found in [26].

### 2.3.2 Preprocessing

The watershed transformation, which is described later in this chapter, was the chosen algorithm used to perform the segmentation. If we are to apply this transformation to the original image, then the result will be extreme oversegmentation. This is due to the varying pixel intensity within each region of the image that represents a rock. To visualize this, an example is provided in Figure 2.4, where the original image and result of the watershed transformation are shown.

![Figure 2.4: Original image (left), oversegmentation (right)](image-url)
In order to solve the oversegmentation problem, an attempt must be made to smooth out the areas within each rock while still preserving their edges. One solution for this is to use morphological image processing techniques. Before some of the specific operations are described, it must first be understood what structuring elements are and how they work. If we are operating on an image $I$, then a structuring element used to perform morphological operations on $I$ can be thought of as another image, which is usually much smaller than $I$. These structuring elements can take on many different shapes, such as a rectangle or disk as shown in Figure 2.5. These structuring elements can take on different sizes, as well, which are dependent on what they are being used for.

![Figure 2.5: Structuring element shapes: Box (left), Disk (right)](image)

The structuring element is passed over the image at each pixel’s location, with the center of the structuring element at the current pixel. Translations of the structuring element can be used at any location in the image, and can be used for the purpose of enlarging a region, or possibly checking to see if it fits inside a region [27]. For all of the following examples, the disk-shaped structuring element from Figure 2.5 will be used. Figure 2.6 shows the original matrix that will be used for providing examples of erosion and dilation.

Grayscale morphological erosion is the operation in which each pixel in the output image takes on the minimum value of the pixels in the surrounding region defined by the structuring element in the input image. Figure 2.7 helps to get an intuitive sense of how morphological erosion operates, which uses the disk structuring element from Figure 2.5 on the original matrix from Figure 2.6.

Assuming the upper left element of the input matrix is located at $(0, 0)$, we can consider the
Background and Literature Review

<table>
<thead>
<tr>
<th>30</th>
<th>39</th>
<th>48</th>
<th>1</th>
<th>10</th>
<th>19</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>47</td>
<td>7</td>
<td>9</td>
<td>18</td>
<td>27</td>
<td>29</td>
</tr>
<tr>
<td>46</td>
<td>6</td>
<td>8</td>
<td>17</td>
<td>26</td>
<td>35</td>
<td>37</td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>16</td>
<td>25</td>
<td>34</td>
<td>36</td>
<td>45</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>24</td>
<td>33</td>
<td>42</td>
<td>44</td>
<td>4</td>
</tr>
<tr>
<td>21</td>
<td>23</td>
<td>32</td>
<td>41</td>
<td>43</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>22</td>
<td>31</td>
<td>40</td>
<td>49</td>
<td>2</td>
<td>11</td>
<td>20</td>
</tr>
</tbody>
</table>

**Figure 2.6:** Original matrix before morphological operations

<table>
<thead>
<tr>
<th>30</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>17</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>8</td>
<td>16</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>8</td>
<td>15</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>13</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2.7:** Matrix after an erosion

Element (2, 2), which has a value of 8. We now consider all pixels in the surrounding region that line up with the structuring element’s elements that take on a value of one. Let $M$ be our output matrix. For this example we have

$$M(2,2) = \min \{48, 47, 7, 9, 46, 6, 8, 17, 26, 14, 16, 25, 24\} = 6 \quad (2.7)$$

Morphological dilation is very similar to erosion, only instead of taking the minimum value, each pixel in the output image takes on the maximum value of the pixels in the surrounding region defined by the structuring element in the input image. This can be visualized in Figure 2.8, which also operates on Figure 2.6 and uses the disk structuring element.

Morphological opening is the process of performing an erosion and then a dilation with the same structuring element. This is used to eliminate small and undesired regions from
the image while preserving the shapes of the larger regions. Morphological closing is the opposite, which performs a dilation followed by erosion. The result of this operation is the removal of small holes in the image.

Image reconstruction is a process that inputs two images, the marker and the mask, and outputs image representing the reconstruction. Dilation operations are repeatedly performed on the marker image until the contour of the marker image fits under the mask [28]. The structuring element can be thought of as a 3x3 square, which is based on the 8-neighbor connectivity that is specified as the default for MATLAB’s imreconstruct function. After dilation is performed on the marker, any values that exceed the corresponding values in mask are set to those values in the mask. If there is no change between this updated marker and the marker before the dilation, then the procedure terminates.

### 2.3.3 Watershed Transformation

The watershed transformation is a method for segmenting an image. The watershed transformation used for the work in this thesis uses the algorithm by Meyer [29]. The gradient of the image is used when applying the transformation. To provide a better understanding of how the watershed transformation works, Meyer discusses how the gradient of the image can be thought of as a topographical relief. The relief is flooded, where sources are placed at regional minimums. As the flooding continues, if two distinct catch basins begin to merge, a
dam is created to prevent the merging. These dams represent what are known as the watershed lines, which separate different regions in the image. The following algorithm describes the details of Meyer’s watershed transformation.

**Algorithm 1** Meyer’s watershed transformation algorithm

1. Locations of where the watershed transformation will begin flooding are chosen, where each location is given a label.
2. All of the pixels in the neighboring region are then inserted into a priority queue, the priority of which is determined by the pixel intensity.
3. Pixels are then removed from the priority queue, and if all of already labeled neighbors of the removed pixel have the same label, then this pixel is also assigned the same label. The non-labeled neighbors that are not already in the priority queue are then inserted.
4. Repeat previous step until the priority queue is empty.

### 2.3.4 Support Vector Machines

The system presented here classifies rocks vs. non-rocks on a conveyor belt is based on analyzing features of the segmentation results. Each part of the input image corresponds to a segment or boundary of a segment, and it is therefore important to be able to tell which segments correspond to a rock, and which do not. A Support Vector Machine (SVM) [30] provides a way of calculating a decision boundary that maximizes the margin between the two classes. For a weight vector \( w \), bias \( b \), and feature vector \( x \), the hyperplanes used in separating the two classes are defined by

\[
\begin{align*}
    w^T x + b = \begin{cases}
    \epsilon^2, & \text{class 1} \\
    -\epsilon^2, & \text{class -1}
    \end{cases}
\end{align*}
\]

We use can then use \( \epsilon = 1 \) for convenience, since \( w, b, \) and \( x \) can all be scaled. The margin is calculated by

\[
\frac{w^T}{\sqrt{w^Tw}}(x_+ - x_-) = \frac{2}{\sqrt{w^Tw}}
\]
where \( x_+ \) is a point from class 1 where \( w^T x_+ = 1 \), and \( x_- \) is a point from class -1 where \( w^T x_- = -1 \). Therefore we minimize the distance \( w^T w \) in order to maximize the margin. For the linearly separable case, the minimization becomes

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} w^T w \\
\text{subject to} & \quad y_n (w^T x_n + b) \geq 1, \quad i = 1, \ldots, N.
\end{align*}
\] (2.10)

However, it is often the case that the training data will not be linearly separable. To handle this, \( \xi \) is defined to be the "slack", which is distance between the correct margin and \( x^n \), where \( \xi \geq 0 \). Considering the slack, the new minimization for the case that it is not linearly separable becomes

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} w^T w + \frac{C}{2} \sum_n (\xi_n)^2 \\
\text{subject to} & \quad y_n (w^T x_n + b) \geq 1 - \xi_n, \quad i = 1, \ldots, N.
\end{align*}
\] (2.11)

Here \( C \) controls the trade off between maximizing the margin and minimizing the slack. It is up to the individual implementing the SVM to determine this value experimentally. Figure 2.9 shows an example for both the linearly separable case and the non-linearly separable case.

By formulating the minimization in 2.11 as a Lagrangian, differentiating and setting the result to zero, \( w \) becomes

\[
w = \sum_n \alpha_n y_n x_n
\] (2.12)

More details can be found in [30].

### 2.4 Stereo Boom

In order to have scale information in the 3D point cloud of the bench face or stockpile, there needs to be a way of relating relative distances in the point cloud to actual distances.
Figure 2.9: Linearly separable classes (left) and non-linearly separable classes (right). The circled data points represent the support vectors. $\xi$ in the non-linearly separable example represents the slack (the distance past the margin corresponding to the class with green points).

One way to solve this problem is to have targets, separated by a known distance, where they are located within view when the images are being taken. Since the camera locations are provided in the 3D point cloud output by Bundler, another way to measure the distances between the cameras based on the odometry of a ground-based vehicle. However, because the image-based systems presented in this thesis operate out of a pickup truck, attempting to use odometry information can result in an inaccurate scaling of the point cloud. For this reason, two cameras separated by a fixed baseline were used, so that the distances between the left and right cameras could be used along with the known distance between the cameras to scale the point cloud.

2.4.1 Design

The two camera system used was the stereo boom designed and built at the USL at Virginia Tech, shown in Figure 2.10, which is normally mounted on an unmanned helicopter to map terrain.
The stereo boom consists of two cameras (Kappa Zelos 655 GV) [31], with a fixed baseline of 1.473 meters. Information for the cameras is provided below:

- 5MP Color Cameras
- CCD progressive scan sensor
- 2448 x 2050 pixel
- High Transfer rate: 1 Gbit/s

The stereo boom can operate in different modes, so that images can be taken automatically at regular intervals or so that images can be taken when a button is pushed on a USB remote connected to the stereo boom. Both cameras are hardware triggered to minimize the time difference between the image capture of the left and right cameras. This is necessary for the assumption made that the left and right images are taken simultaneously at each location so that the appropriate scale can be determined.

## 2.5 Modified Bundler

The 3D points that are output by Bundler and PMVS are all a part of a relative coordinate system. These points include the locations of where the images were taken. For all of the scaling that was performed on the point clouds used for the work presented in this thesis, this was done by scaling the point cloud based on the median distances between the cameras in the initial point cloud, and the known distance between the cameras. In order to accomplish this, Bundler had to be modified so that the relevant information necessary to locate the 3D

![Stereo boom](image)
2.6 Point Cloud Operations

When working with any of the 3D point clouds output by Bundler and PMVS, it is important to understand that these are part of a relative coordinate system. For this reason, an XY plane needs to be defined so that the algorithms for the drilling analysis and stockpile volume estimation can function properly. The points must also be scaled to a known measurement, such as meters. Many outliers can also appear in the resulting point clouds, which can cause problems for the algorithms presented in this thesis, and must therefore be removed.

2.6.1 Outlier Removal

The Point Cloud Library (PCL) [33] provides many libraries for operating on 3D point clouds, which includes a library for removing outliers. This becomes useful for analyzing the bench face, or calculating the volume of a stockpile, because these outlier points can interfere with the algorithms. The outlier removal by the PCL works by computing the mean distance between each point and all of its neighbors. An assumption of normality for the distribution of these distances is made where a mean and standard deviation is calculated for the distribution. Points with mean distances that are outside of the interval defined by the global mean and standard deviation are then eliminated. To run the outlier removal, the number of neighbors to analyze is set, along with the standard deviation used for the elimination. Points with a distance larger than the set standard deviation of the mean distance are removed.

2.6.2 Scaling

For both the stockpile the bench face 3D reconstructions, the point clouds must be scaled for the algorithms to work. The relative distances between all of the left and right cameras
in the point cloud are computed, which is then followed by finding the median value of these
distances. The median is used as it is more robust to outliers in the camera positions. The
actual distance between the left and right camera, in meters, is then divided by this median
value to find

\[
B \equiv \text{Baseline for the left and right camera (meters)}
\]

\[
\rightarrow \text{scale} = \frac{B}{\text{median}(|\text{left}_i - \text{right}_i|_2)}
\]

(2.13)

where \( \text{left}_i \) and \( \text{right}_i \) are the relative 3D coordinates of the left and right cameras respectively.

To assess the accuracy of the Bundler and PMVS outputs after this scaling is applied,
two different tests were performed. The first test was to measure the root-mean-square
errors (RMSE) of the distances between the left and right camera pairs, where the RMSE is
calculated as

\[
\text{RMSE} = \sqrt{\sum_{i=1}^{N} (|\text{left}_i - \text{right}_i|_{\text{scale}} - B)^2}
\]

(2.14)

where \( N \) represents the number of left and right image pairs. The second test involved
measuring a distance in the point cloud an comparing that to an physical measurement
performed in the area where the images were taken.

A 3D reconstruction was generated from images taken of the building shown in Figure 2.11,
which was used to carry out this assessment of the accuracy.

After running Bundler and PMVS, and then scaling the point cloud to meters, the resulting
point cloud in Figure was generated Figure 2.12. The line of red, green, and yellow points in
the point cloud represent the 3D coordinates of the camera positions, as well as their normal
vectors.

The RMSE for the camera positions was calculated to be 5.45cm. The distance \( L \), shown in
the right of Figure 2.12, was measured on the building with a tape measure to be approxi-
mately 242.57cm, while the measured distance in the point cloud was 242.616cm. Assuming
that the measured value is the true value, this gives an error of 0.0188%. There are, of course, errors associated with this value, such as the slight differences in the start and end locations of the measurements. Therefore, this is not meant to prove astounding accuracy, but rather show that the scaling process of the Bundler and PMVS outputs is performing reasonably well.
2.6.3 Plane Fitting

The normal vector to a plane can be used when rotating points so that changes in the Z axis correspond to changes in height. For this reason, a plane needs to be fit through a set of 3D points identified in the point cloud. Assuming that there is a set of pre-selected points to fit to, a plane is fit using Principal Components Analysis (PCA), where the first three principal components represent the basis of the plane, and the normal vector, where

\[
\text{normal} = [a \ b \ c]^T
\]  

(2.15)

We then calculate the final coefficient of the plane’s equation, \(d\), as

\[
d = - (\bar{x} \ \bar{y} \ \bar{z})(\text{normal})
\]  

(2.16)

where \(\bar{x}\), \(\bar{y}\), and \(\bar{z}\) are the mean values for \(x\), \(y\), and \(z\) respectively. The equation for the plane then becomes

\[
a \ x + b \ y + c \ z + d = 0
\]  

(2.17)

The example plane used here is visualized in Figure 2.13.

2.6.4 3D Rotation

With a plane and normal vector defined, we are then able to use this normal vector to rotate all of the 3D points into a known coordinate system. The rotation matrix, \(R\), can be found by first finding

\[
\begin{align*}
\bar{z}_\text{normal} &= [a \ b \ c] \\
\bar{y}_\text{normal} &= [-b \ a \ c] \\
\bar{x}_\text{normal} &= \bar{z}_\text{normal} \times \bar{y}_\text{normal}
\end{align*}
\]  

(2.18)

We then set...
Figure 2.13: Example plane and normal vector

\[ A = \begin{bmatrix} x_{\text{normal}} \\ y_{\text{normal}} \\ z_{\text{normal}} \end{bmatrix} \]  

(2.19)

Now to find \( R \), it is observed that \( AR \) should be equal to the identity. We therefore find that

\[ AR = I_{3,3} \]

\[ \rightarrow R = A^{-1} \]  

(2.20)

By multiplying all of the points by \( R \), we can then rotate these points into a coordinate system where the \( Z \) values represent the height values in a scene. Figure 2.14 helps to visualize this process.
Figure 2.14: Rotating points into the Cartesian coordinate system
Chapter 3

Bench Face Profiling and Drilling Analysis

When performing a blast at a quarry, it is important to identify the optimal locations of where to drill the boreholes. If the locations of the boreholes are not chosen properly, then inefficient blasts can result. An even worse scenario is where boreholes are placed too close to the bench face, possibly resulting in flyrock, which can endanger humans that may be located too close to the blast zone, and possibly cause damage to equipment.

A two-camera system is presented, which uses image-based 3D reconstructions to suggest locations of where, and at what angle to drill the boreholes. The blast design formulas from the SME Mining Reference Handbook provide everything needed to calculate the burden, spacing, subdrilling, and stemming. The work presented in this thesis focuses on providing the 3D reconstruction, suggesting the locations and angles of the boreholes, while leaving the issue of determining what parameters to input to the system up to the operator.

Images of the two bench faces used to demonstrate an image-based 3D reconstruction are shown in Figure 3.1.
3.1 System Design

For capturing the images, the stereo boom was mounted in the back of a pickup truck. The algorithms for the drilling analysis require the stereo boom to be mounted vertically to obtain a reference angle so that the direction of down can be identified. In order to make the direction of the boom invariant to non-level terrain, the stereo boom was mounted on a balancing device built in the USL that maintains a constant direction between the left and right cameras. Figure 3.2 shows the setup of the stereo boom in the back of a pickup truck, as well as one of the two traffic barrels used as a reference in the output.
A traffic barrel was chosen to use as a reference for finding the locations for the tops of each borehole, because they are light enough to carry, but heavy enough to prevent being toppled by gusts of wind. They also reconstruct well when using Bundler and PMVS, as there is varying texture on the surface of the barrel, which makes them easy to identify in the output. These references are needed, because operators need the ability to measure distances from the barrels to the tops of each one of the boreholes. Other methods, such as geo-referencing by using the Helmert Transformation [34], are possible, but can cause problems if there is drift associated with the GPS coordinates used in the georeferencing, or with the measurement device used for identifying the GPS location of the borehole. Figure 3.3 shows the flow of the system’s operation.
3.2 3D Reconstruction

To generate the 3D image-based reconstructions, the images are tagged with either "left" or "right" and then the image number. The images are then passed to Bundler, which generates the sparse reconstruction, and then to PMVS, which generates the dense reconstruction. The dense reconstructions for the example benches are shown in Figure 3.4.

![Bundler/PMVS output for the example bench faces.](image)

For the bench in the left part of Figure 3.4, the cameras were located at an average of around 260m from the middle bench, while for the bench on the right part of the figure, the images...
were taken at an average of around 60m from the bench. This shows the ability of the system to map the bench face even when the images are taken far away from the bench. However, for the example where the images were taken from around 260m away, the traffic barrels were not setup. At this distance, it is possible that larger reference targets may be needed.

Once the reconstructions are finished, then outliers can be removed by using the outlier removal package from PCL. Outliers are removed from the point cloud shown on the right of Figure 3.4, the resulting point cloud of which is shown in Figure 3.5.

![Figure 3.5: Bench face 3D reconstruction with outliers removed.](image)

As the right camera was on top when taking images, and the cameras oriented vertically, the direction of down for each left and right pair of images can be calculated by finding

\[
direction_{Down_i} = left_i - right_i
\]

where \(left_i\) and \(right_i\) are the 3D coordinates of the left and right cameras respectively. The median values for X, Y, and Z in \(direction_{Down_i}\) are then used to provide a more robust estimate for the normal direction of down, where
\[
\begin{align*}
\text{medianX} &= \text{median}(\text{direction}_i.X) \\
\text{medianY} &= \text{median}(\text{direction}_i.Y) \\
\text{medianZ} &= \text{median}(\text{direction}_i.Z) \\
\text{directionDown} &= \frac{[\text{medianX} \text{ medianY} \text{ medianZ}]}{||[\text{medianX} \text{ medianY} \text{ medianZ}]||_2}
\end{align*}
\]

This direction vector is then used to rotate the points by a rotation matrix \( R \) so that

\[
(d\text{irectionDown})(R) = [0 \ 0 \ -1]
\]

### 3.3 3D Points Selection and Region Extraction

Once the reconstruction is finished and the outliers are removed, then the operator selects points on the bench face that are used to extract the region of interest for the blast analysis. A package from the PCL is used to select and store the 3D points. Six points are selected in the point cloud, where the first two of these are on the top of the bench, and the next two are at the base of the bench. Next, the two barrels are identified in the reconstruction and selected, with the left barrel selected first.

Once all of these points are selected, the first four points from the bench face are used to define a region where all of the inliers lie. These inliers are then stored, along with the selected points, and are used in determining the borehole locations and angles. Figure 3.6 shows an example of the locations for the locations of the first four points.
3.4 Borehole Placement

The next step in the process is to identify the borehole locations and angles, where it is assumed that the points have already been rotated so that the down direction is represented by $[0 \ 0 \ -1]$, and that only the points within the selected region exist in the point cloud. Direction vectors are first setup so that there is a reference for movement within rows, to the next row, and to the wall to sample points. The direction vectors are calculated by

$$\text{directionBetween} = \frac{\text{selected}_2 - \text{selected}_1}{\|\text{selected}_2 - \text{selected}_1\|_2} \quad (3.4)$$

$$\text{directionAwayFromWall} = \text{directionBetween} \times \text{directionDown} \quad (3.5)$$

where $\text{selected}_1$ and $\text{selected}_2$ represent the first two selected points. If $\text{selected}_1 + \text{directionAwayFromWall}$ is closer to the camera positions than $\text{selected}_1$, then $\text{directionAwayFromWall}$ is multiplied by -1. The direction to the wall is then calculated by

$$\text{directionToWall} = -\text{directionAwayFromWall} \quad (3.6)$$
With these direction vectors calculated, the next step is to identify the initial tops of all of the boreholes. Using the calculated, or possibly input, values for the burden and spacing, the number of rows desired by the operator, and the desired pattern, all of the tops can be initialized. The two blast patterns generated in this chapter, as examples, are the parallel pattern and staggered pattern, as shown in Figure 2.3. The following algorithm shows the initialization process.

**Algorithm 2 Initialize borehole tops**

```
Input burden, spacing, selected\(_1\), selected\(_2\), blastType, boreholesPerRow

for currentRow = 1 → #rows do
  localPerRow = boreholesPerRow
  if currentRow is even AND blastType = staggered then
    localPerRow = localPerRow - 1
  end if
  for localBorehole = 0 → localPerRow do
    shiftBack = (burden)(directionAwayFromWall)(currentRow)
    tops[currentRow] = selected\(_1\) + shiftBack
    shiftSide = (spacing)(directionBetween)(localBorehole)
    if currentRow is even AND blastType = staggered then
      shiftSide = shiftSide + (\(\frac{\text{spacing}}{2}\))(directionBetween)
    end if
    tops[currentRow] = tops[currentRow] + shiftSide
  end for
end for
```

With all of the tops initialized, the borehole centers and angles are then calculated. This is done by finding sample points from the top of the borehole down to the base of the bench, which is defined by a plane fit through selected\(_3\) and selected\(_4\) (the points selected at the base of the bench face), and additional points created by adding to directionToWall to each of these selected points. Points are sampled at regular intervals by moving further downward at each point with directionDown. At each sample point, an iteration is performed through
a set of 3D points that move toward the wall as the iteration proceeds, which is done using \textit{directionToWall}. At each point, while moving towards the wall, the minimum distance of the distances to each inlier on the bench face is found. The 3D point with the minimum distance of these minimum distances is then identified as $P$. The new sample point, $sample_i$, is then found by

$$sample_i = P + (\text{burden})(\text{directionAwayFromWall})$$

(3.7)

Figure 3.7 shows an example of the resulting sample points used to create one of the boreholes, where ten samples are found, and $B$ represents the burden. This is from a side view, where the circles represent the new sample points.

![Figure 3.7](image)

**Figure 3.7:** Example of the sampling procedure used in creating a borehole.

Once these sample points are found, the center and direction vector of the current borehole is found. The median is used to find both of these, as there can sometimes be outlier points that should be ignored. The borehole center is found by
\[
\text{medianX} = \text{median}(\text{sample}_i.X) \quad (3.8)
\]
\[
\text{medianY} = \text{median}(\text{sample}_i.Y) \quad (3.9)
\]
\[
\text{medianZ} = \text{median}(\text{sample}_i.Z) \quad (3.10)
\]
\[
\text{boreholeCenter}_i = [\text{medianX} \text{ medianY} \text{ medianZ}] \quad (3.11)
\]

The direction of the borehole is then calculated by the following algorithm.

**Algorithm 3** Find borehole direction

- **Input:** sample\(_i\), where \(i = 1, \ldots, \# \text{samples}\)
- Initialize \(\text{overallDirections}\) as an empty array

\[
\text{for } i = 1 \rightarrow \#\text{samplepoints} - 1 \text{ do}
\]
\[
\text{for } j = (i + 1) \rightarrow \#\text{samplepoints} \text{ do}
\]
\[
\text{localDirection} = \text{sample}_j - \text{sample}_i
\]
\[
\text{Append localDirection to overallDirections}
\]

\[
\text{end for}
\]
\[
\text{end for}
\]
\[
\text{medianX} = \text{median}(\text{overallDirections}.X)
\]
\[
\text{medianY} = \text{median}(\text{overallDirections}.Y)
\]
\[
\text{medianZ} = \text{median}(\text{overallDirections}.Z)
\]
\[
\text{boreholeDirection}_i = \frac{[\text{medianX} \text{ medianY} \text{ medianZ}]}{\|\text{medianX} \text{ medianY} \text{ medianZ}\|_2}
\]

The borehole centers and direction vector are then stored and later used to display in 3D, as a top-down view, or from a side view. It is important to note that the process of finding the boreholes’ centers and angles only needs to be carried out for the first one or two rows, depending on the pattern being generated. For the parallel pattern, this only need to be done for the first row, as the boreholes for all subsequent rows will have the same angle as their corresponding borehole in the first row. The borehole centers, however, will be shifted away from the wall, which can easily be calculated using the burden and row number. For the staggered pattern, only the centers and angles of the first two rows need to be calculated. This needs to be done for the second row, as every other row will be shifted by \(\frac{\text{spacing}}{2}\). After the first two rows, each subsequent row will then take on the angles from the corresponding
boreholes in the first or second row, as well as the centers shifted away from the wall by (2)(burden).

The new borehole tops are then calculated by starting at boreholeCenter$_i$, and moving in the direction of $-boreholeDirection_i$ until reaching the plane defined to represent the top of the bench.

### 3.5 Displaying Boreholes

Once the centers and directions of the boreholes have been found for the given pattern, they can be displayed to the operator in 3D, as a top-down view, or as individual side views.

#### 3.5.1 3D view

Generating a point cloud so that the operator can view the boreholes and bench face in 3D is simple. This is done by writing all of the inliers of the bench face to a file, followed by generated points for the boreholes with a distinct color, such as green. The points for each borehole are generated by starting at the center and then for each borehole moving up/down in small increments using the borehole’s direction vector until the top and bottom of the bench, each defined by a plane, are reached. The final output is formatted so that it can be viewed in the operator’s preferred point cloud viewer, such as MeshLab. Figure 3.8 shows two different views of the 3D view generated for one of the example benches, where a parallel pattern was the chosen blast pattern.
3.5.2 Top-Down View

To create a top-down view that the operator can view and use to edit the locations, all of the new borehole tops are projected onto a 2D image plane. Different levels of the bench face are then sampled and the points near that level are projected onto the same 2D image plane in order to view the contours of the bench face at each level. The points at each level are
found by identifying inliers that are located within a defined distance threshold of a plane created to represent the current level.

Once all of the points have been projected onto the plane, an image of a defined size is created, and all of the points are transformed and scaled to fit this image. The borehole tops are displayed as circles with their ID number displayed at the side. To display the contours of the bench face, the points that were projected at each level are grouped into a finite number of points by averaging nearby points. A polynomial is then fit to these points, where the degree chosen is the one that minimizes the leave-one-out cross-validation (LOOCV), calculated by

\[
LOOCV = \sum_{i=1}^{N} (y_i - \hat{y}_{-i})^2
\]

This finds the squared error between the estimate for \( y_i \) and the actual value of \( y_i \), where \( i \) represents the current data point, and \( N \) represents the overall number of data points. \( \hat{y}_{-i} \) is found by fitting a polynomial to all of the data except for the \( i \)th data point, and then estimating the output for \( x_i \). It was found that to display a more accurate contour of the bench face for a specific level, the polynomial degree should be at least 10.

Each of the traffic barrels are then displayed as an 'X' in the image, so that these can be referenced when calculating the distances to each of the boreholes. Figure 3.10 shows an example of a top-down view for a parallel pattern with two rows, with the initial locations of the boreholes displayed. Notice that three of the boreholes are highlighted red, which indicates that they have been selected by the user.
Figure 3.10: Top-down view, with initial borehole locations

Figure 3.11 displays the new locations of the three selected boreholes after the user has shifted them closer to the bench face using keyboard input.

Figure 3.11: The selected boreholes shifted closer to the bench face.

Another option from the top-down view is to identify the distances to each of the boreholes, where one distance is measured along a line perpendicular to the line between the two barrels \( (a) \), and the other is measured along a line parallel to the line between the barrels \( (b) \). Figure 3.12 shows how these distances are calculated, where the red 'X' marks represent
the locations of the traffic barrels.

**Figure 3.12:** How distances are calculated to each borehole using the traffic barrels.

The distances are calculated using a pixels to meters conversion. This can easily be changed to any other preferred distance units. This makes it possible for the operator to identify the locations of the tops of the boreholes on-site. Figure 3.13 shows an example where the distances to two borehole tops are displayed.

**Figure 3.13:** Top-down view with distances to two of the borehole tops displayed.
Figure 3.14 shows an example of the top-down view for the staggered pattern, after all of the borehole locations and angles have been calculated.

Figure 3.14: Example of a top-down for a staggered pattern.

3.5.3 Side View

Another useful view for the operator is the side view for each individual borehole. This allows the operator to view the burden at different sample locations of the borehole. With keyboard input, the locations of the borehole can then be changed by shifting the boreholes closer or further to the bench face. The user is also able to control how far the borehole will shift with each keyboard input. Figure 3.15 shows the initial side view for one of the boreholes.
Figure 3.15: Example of the initial side view of an individual borehole.

Figure 3.16 shows the side view for one of the boreholes after the user has shifted the borehole further from the bench face.

Figure 3.16: Example of the side view of an individual borehole after shifting it further from the bench face.
3.6 User Interface

A user interface was developed so that the user can easily control the input parameters. Figure 3.17 shows this user interface.

![User Interface for the Bench Face Profiling and Drilling Analysis](image)

**Figure 3.17**: User interface for the bench face profiling and drilling analysis.

If the user wants the system to calculate the spacing, subdrilling, and stemming, then the "Calculate" option can be selected. The system will then use the input values for $K_s$, $K_j$, $K_T$. 

46
and $K_T$, along with the formulas from Table 2.1, to calculate these outputs. If "Enter manually" is selected, then the system assumes that the user has input values into the table directly, and will use these values to assign locations and angles for the boreholes. Two different blast patterns can be calculated, as well as whether or not the boreholes should be angled. Once the locations and angles for all of the boreholes have been found, the user can choose to see the top-down view or side view by clicking the corresponding buttons.
Chapter 4

Identifying Oversize Rocks

In this chapter, all of the details of the system created to identify oversize rocks on a conveyor belt are discussed. This starts with the system design, which details all of the components necessary to make the system operate. This includes the hood design, lighting, camera, and the computer. The algorithms and procedures involved with classifying regions of the input image as rocks are then discussed, including the image acquisition, preprocessing, general segmentation with the watershed transformation, region splitting for undersegmented regions, and classification procedure. The calculation and analysis of the size distribution is then discussed, which is used to alarm operators when rocks that are too large are on the conveyor belt, where the size thresholds are specified by the operators. An overview of the user interface is then provided, which allows the operators to tune parameters such as the size thresholds used to trigger the alarm. Finally, the results of the system’s operation are provided, including computation times and performance analysis for each of the algorithms.

4.1 System Design

Most of the design requirements were determined through testing performed at the USL. The design was based around the idea that the system should be able to provide consistent lighting in order to capture clear images. The computer used during testing contained an AMD Phenom II X6 1075T Processor (6 cores, 3 GHz) and 8 GB of RAM. This was determined to be sufficient for the system’s real-time operation. A monochrome camera was used for all of the testing and implementation. One reason is that MATLAB’s watershed function [35], which implements Meyer’s watershed transformation, requires a grayscale image as its input. It was determined that using a color camera was unnecessary, as performing segmentations on
each channel of the image did not seem to provide any potential benefits in the initial stages of the design. Using a color camera also means a higher computation time due to processing each color channel of the image. The camera used for the testing and implementation was the Basler ace acA1300-30gm monochrome camera [36]. The progressive scan CCD sensor prevents the rolling shutter effect. This is necessary for the system’s operation, since the rocks are moving quickly along the conveyor belt, and also the structure holding the camera is constantly vibrating. Details of the camera chosen are provided in Table 4.1.

**Table 4.1:** Camera specifications

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>1296 pixels x 966 pixels</td>
</tr>
<tr>
<td>Interface</td>
<td>Gigabit Ethernet</td>
</tr>
<tr>
<td>Frame Rate</td>
<td>30 fps</td>
</tr>
<tr>
<td>Sensor</td>
<td>Progressive Scan CCD, global shutter</td>
</tr>
<tr>
<td>Power Requirements</td>
<td>12V (DC)</td>
</tr>
<tr>
<td>Sensor Size</td>
<td>1/3 inch</td>
</tr>
</tbody>
</table>

Once the camera and computer were setup, a hood was built to test the distance the camera should be from the conveyor belt and the lighting conditions that would best assist the segmentation algorithm. An ideal model of the hood’s structure is provided in Figure 4.1, where the lights are angled towards the center of the conveyor belt.

![Figure 4.1: Model of hood’s structure](image)

The actual hood built for testing was constructed from two-by-fours, with adjustable legs
to allow testing different heights of the camera to see how it affected the segmentation algorithm. A cart containing a pile of rocks was pushed under the hood quickly during tests, which allowed for an exposure to be set on the camera that did not make the images too dark, while still preventing motion blur. Figure 4.2 shows the hood used for testing.

![Figure 4.2: Hood used for testing in the USL](image)

The final dimensions of the hood used in both the testing, and final implementation are shown in Table 4.2.

**Table 4.2:** Final hood dimensions

<table>
<thead>
<tr>
<th>Height (Top of camera box to conveyor belt)</th>
<th>116cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance between lights</td>
<td>119cm</td>
</tr>
</tbody>
</table>

Using the height $H$ and distance between the lights $D$, the angle of the lights from the side of the hood ($\alpha$) can be calculated by

$$\alpha = 90^\circ - \tan^{-1}\left(\frac{H}{D/2}\right) = 27.3^\circ$$

(4.1)

Four fluorescent light bulbs were used (two per fixture) during the testing. The light fixture contained an electronic ballast, which was used to prevent large changes in brightness due to flickering, which can occur if a magnetic ballast is used. All of the tests performed in the USL were done at night to simulate the use of sun-blocking tarps for the final system.
4.2 Rock Segmentation and Size Analysis

This section describes the algorithms involved in classifying regions of the input image as rocks. A high level view of the rock segmentation process is provided in Figure 4.3.

![Figure 4.3: Rock Segmentation Overview](image)

In this chapter, the majority of the segmentation process will be based on the cropped image in Figure 4.4. The original image, shown on the left of Figure 4.4, is cropped in the early stages of the algorithm to eliminate large portions of the conveyor belt. It was observed that the area where these conveyor belt regions were located within the image remained fairly consistent. The size of the original image is 1296 pixels x 966 pixels, and the resulting cropped image is 1280 pixels x 600 pixels. The cropped original image will be referred to as $I_c$. 
The camera was setup at a cement plant with the same dimensions as the hood used for testing. With the camera fixed 116cm from the conveyor belt, a tape measure was placed on top of the belt so that a pixel to inches conversion could later be made. Since the rocks are transported along the conveyor belt at variable height, the tape measure was slightly elevated to represent an estimated midpoint for the range of possible heights of future piles of rock transported along the belt. An image was then taken, after which two points were selected in the image along the tape measure to get a distance in pixels. This, along with the corresponding distance in inches, was then used to calculate the conversion. The resulting value was $\frac{1}{223}$ inches per pixel. Figure 4.5 shows the final setup for the hood, which is situated around the conveyor belt. Two sun-blocking tarps were used in an attempt to prevent sunlight from altering the lighting conditions.
4.2.1 Image Acquisition

MATLAB’s Image Acquisition Toolbox [35] was used to acquire images, as it supports GigE cameras. During the camera setup, the exposure value is set to roughly 1 millisecond, which was chosen based on tests performed at the USL aimed at finding an exposure value that maximizes the brightness of the images while still preventing motion blur.

4.2.2 Preprocessing

The first process in the segmentation is to preprocess the image so that oversegmentation does not result when the watershed transformation is applied. The first step in accomplishing this is to apply an adaptive thresholding technique to assist the segmentation by eliminating background regions. The adaptive thresholding technique used is Otsu’s method, which is implemented with MATLAB’s graythresh function. This function inputs an image $I$, and returns a level ($l$), where $l \in [0, 255]$. The resulting threshold image $I_T$ is then calculated by

$$I_T(x, y) = \begin{cases} 
1, & \text{if } I(x, y) \geq l \\
0, & \text{otherwise}
\end{cases}$$ (4.2)
Zhang et al. found in [37] that multiplying the threshold $l$ by $\frac{3}{5}$ worked well in their experiments, which also worked well for the images used during the tests performed in this thesis. Once the thresholding is performed, smaller holes are filled, as these holes in $I_T$ often represent areas of a rock surface. Through testing, it was determined that filling in holes connected by 100 or fewer pixels worked well. If these holes are not filled, this can later cause a region of the input image representing a rock to be divided into multiple segments during the segmentation. Figure 4.6 shows both the image as a result of applying Otsu’s method and the post-thresholding image with small holes filled, which will be referred to as $I_{TF}$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image1.png}
\caption{Binary image, post-thresholding (left), binary image with small holes filled $I_{TF}$ (right)}
\end{figure}

A morphological image opening can then be applied, with a 5x5 square-shaped structuring element, to the filled binary image to separate some of the overlapping regions. Image $I_C$ is then updated by masking itself with $I_{TF}$ where

$$I_C(x, y) = \begin{cases} 
I_C(x, y), & \text{if } I_{TF}(x, y) = 1 \\
0, & \text{if } I_{TF}(x, y) = 0
\end{cases}$$  \hspace{1cm} (4.3)

Figure 4.7 shows the updated image.
In order to reduce computation time, $I_C$ is first resized with a scale of 0.5, creating a 640 pixels x 300 pixels image. The scale 0.5 was chosen, as this was one of the smallest scale values tested that still had little to no impact on the performance of the segmentation process. The pixel-to-inches conversion is also updated with this scale for later use. A disk-shaped structuring element $S$ is then created with a radius of 9 pixels. This value was heuristically determined, as a lower value was found to result in oversegmentation, and higher values were more likely to result in undersegmentation. The pseudocode for all of the morphological operations that then take place are shown in the following algorithm.

**Algorithm 4** Morphological operations on the input image

1. Opening of $I_C$ using $S$, and store result in $I_O$
2. Image reconstruction with $I_O$ as the marker, and $I_C$ as the mask. Store the result in $I_{OR}$
3. Image closing of $I_{OR}$ using $S$, and store result in $I_{OC}$
4. Image reconstruction with $I_{OC}$ as the marker, and $I_C$ as the mask. Store the result in $I_{CR}$

The result of performing a opening on $I_C$ with $S$, and the image reconstruction are shown in Figure 4.8.
Figure 4.8: Opening of $I_C$ (left), image reconstruction of the opening with $I_C$ as the mask (right).

The result of performing a closing on the reconstruction in Figure 4.9 with $S$, and the image reconstruction of the result of the closing operation are shown in Figure 4.9.

Figure 4.9: Image closing after an opening/reconstruction (left), image reconstruction of the closing with $I_C$ as the mask (right).

To increase the contrast, an image adjustment is performed to increase the contrast in the image. This is done with MATLAB’s `imadjust` function, which maps values of the input image to the output image so that 1% of the pixel intensities in the image saturate at the low/high end [35]. Since the distribution of the pixels of $I_{CR}$, from Figure 4.9, are centered in the lower range of pixel intensities, this will map the intensities so that they range from 0 to 255.

At this point, a human observer should be able to identify the contours of many of the rocks based on the homogeneous pixel intensities within each region of $I_{CR}$ where the corresponding region in $I_C$ is a rock. However, after the adjustment it can often be observed that within many of the larger regions, there are several smaller regions with different pixel intensity.
This causes problems when applying the watershed transformation to the gradient, because the gradient surrounding these smaller regions will have a high enough magnitude to create a separate region, which can potentially prevent the larger surrounding region from being segmented properly. For this reason, a dilation followed by a reconstruction is performed on $I_{CR}$, the result of which then undergoes a histogram equalization to again enhance the contrast. The final result will be referred to as $J$, which is shown in Figure 4.10.

![Figure 4.10: Final result of the preprocessing ($J$)](image)

### 4.2.3 Segmentation

Once the preprocessing is complete, segmenting the cropped input ($I_C$) can be carried out. The first step of the segmentation is to find the gradient of $J$, which is the final result of the preprocessing. The gradient can then be used as a marker for the watershed segmentation. The gradient is calculated using the sobel operator, which calculates the partial derivatives in a 3x3 neighborhood of pixels [38]. The two kernels used for this operation are

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = S_x^T = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (4.4)$$

The partial derivatives are then found by
\[
I_x(x, y) = -I(x - 1, y - 1) + I(x + 1, y - 1) - 2I(x - 1, y) + 2I(x + 1, y) \\
-I(x - 1, y + 1) + I(x + 1, y + 1)
\]

\[
I_y(x, y) = -I(x - 1, y - 1) - 2I(x, y - 1) - I(x + 1, y - 1) + I(x - 1, y + 1) \\
+2I(x, y + 1) + I(x + 1, y + 1)
\]

(4.5)

It is important to note that \(x\) here is accessing the column, and \(y\) is accessing the row. When at the bounds of \(I\), when calculating \(I_x\) and \(I_y\), and an attempt is made to access a row or column out of bounds, it assumed that these values are equal to the nearest value on the border of the image. Finally, the gradient \((G)\) is calculated by

\[
G(x, y) = \sqrt{I_x(x, y)^2 + I_y(x, y)^2}
\]

(4.6)

During experimentation, it was found useful to threshold \(G\) to prevent problems of oversegmentation, where the threshold used was 1.

The watershed transformation is then applied to \(G\) using MATLAB’s \textit{watershed} function, which is based on Meyer’s flooding algorithm. The output of the watershed is a labeled image, where all of the pixels of the same segment have the same label, and the boundaries of the segments are labeled zero. By using MATLAB’s \textit{label2rgb} function [35], these segments can be visualized in an RGB image. The segmentation is shown in Figure 4.11.

\[\text{Figure 4.11:} \text{ Segmentation after the watershed transformation is applied to the gradient (left), segmentation with overlay (right).}\]

From this segmentation, it can be observed that there are many regions that do not represent rocks. Applying the classification and region splitting algorithms to all of these regions is
unnecessary, and would require much more computation time than desired. The first step in reducing the number of regions is to eliminate all of the regions connected belonging to the background of $I_{TF}$, where $I_{TF}$ is the binary image resulting from applying Otsu's method. The next step is to eliminate regions below a certain size, as very small regions are often the result of oversegmentation, and can be difficult to eliminate during classification. Regions with a size less than 100 connected pixels were eliminated. This threshold was selected by testing different values over multiple images.

The final step is then to eliminate regions connected to the top and bottom borders of the image. This is done to eliminate segments that correspond to regions of the conveyor belt. The width of the rocks on the belt varies, and therefore a cropping was chosen that attempts to capture the maximum width, and therefore regions of the conveyor will frequently be seen in the cropped image. Since regions of the conveyor belt usually do not separate into multiple segments, eliminating regions on the border will eliminate these conveyor belt regions a majority of the time. Figure 4.12 shows the segmentation after eliminating background and small. Initially, only regions bordering top and bottom edges of the image are eliminated. Regions bordering the left and right edges are given a chance to be split, which potentially allows rocks that have been undersegmented to be counted towards the size distribution estimate.

Figure 4.12: Segmentation after eliminating small and background regions, but before eliminating regions bordering edges (left), segmentation with overlay (right).
4.2.4 Rock Classification

Once the segmentation is finished, there are still several invalid segments, due to under-segmentation and oversegmentation. An attempt is made to eliminate these regions by examining features of each region. The final segmentation from Figure 4.12 will be referred to as $L$. The labels of $L$ will be referred to as $l_i$, where $l_i = i$, and $i = 1, \ldots, N$.

The two features from Dunlop [23] found to be most useful for classification were

$$\text{Circularity} = \frac{(\text{Region perimeter})^2}{\text{Region area}}$$

and

$$\frac{1}{\text{Convexity Area}} = \frac{\text{region area}}{\text{Convex hull area}}$$

Another two features that were used, which were also used by Mkwelo [21], were the variance of the pixel intensities inside the region, as well as the gradient ratio for the perimeter and inside region. The gradient values of the original image were used to calculate this ratio, where

$$\text{Gradient Ratio} = \frac{\sum (\text{Gradient perimeter})_i}{\sum (\text{Gradient area})_j}$$

Finally, maximum dimension and region size were added to the set of features to use.

To summarize, the 6 features used for classification, along with their ID number, are shown in Table 4.3.
Table 4.3: Features for classification

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Circularity</td>
</tr>
<tr>
<td>2</td>
<td>$\frac{1}{\text{Convexity Area}}$</td>
</tr>
<tr>
<td>3</td>
<td>Variance of pixel intensities</td>
</tr>
<tr>
<td>4</td>
<td>Gradient ratio</td>
</tr>
<tr>
<td>5</td>
<td>Maximum dimension</td>
</tr>
<tr>
<td>6</td>
<td>Region size</td>
</tr>
</tbody>
</table>

Multiple machine learning algorithms were tested, the accuracies of which are shown in the results section. A Support Vector Machine (SVM) was found to provide the highest test accuracy, and was therefore the classification algorithm of choice. To train the SVM, 25 images were segmented, and then within each segmentation image, segments were selected and labeled as a valid or invalid segment. The training focused on selecting regions that were distinctly good and bad, as there are many segments that correctly capture the maximum dimension of the rock, but contain features that are more likely to classify the region as invalid. The validity of the region was recorded, along with the 6 features, listed above, for the segment. Once the data collection was complete, the SVM was trained. To provide an example of how this would work using only the first two features, the SVM was trained using the two best features, \textit{circularity} and $\frac{1}{\text{Convexity Area}}$. The result of training with only these two features is shown in Figure 4.13, where circled data points represent the support vectors, and the line represents the decision boundary.
Figure 4.13: \( \frac{\text{region area}}{\text{Convex hull area}} \) vs. circularity after the SVM was applied. Green "*" labels represent valid segments, and red "+" labels represent invalid segments.

By applying the SVM with all 6 features, a hyperplane decision boundary is calculated, which is then used to classify new data points.

### 4.2.5 Region Splitting

Undersegmentation sometimes occurs in the segmentation, and these regions will usually be classified as non-rocks due to their irregular shape. In an effort to correct this undersegmentation, an algorithm was developed to attempt splitting these regions into multiple segments where at least one of the resulting segments represents a rock. Once a region is split, the resulting segments are then classified using the average pixel intensity and the SVM used for the initial classification. An example of a region that can be corrected by this splitting algorithm is shown in Figure 4.14, which shows two rocks connected together as a result of undersegmentation, and a failure of the preprocessing and segmentation processes.
Figure 4.14: Region resulting from undersegmentation, where two separate rock regions have been merged together.

In [24] erosion was used to separate connected regions such as these. However, erosion and other morphological operations, such as opening do not work well for this problem, because these methods do not preserve the shape of the region, which is important for calculating the maximum dimension.

The region splitting algorithm starts by performing a morphological opening to smooth the edges, and potentially prevent invalid splits. The structuring element used for each region is based on its size. After, the convex hull of the region is calculated, and areas of the convex hull that are not in the original region are found. The resulting blobs that contain less than 70 connected pixels are then eliminated, as these smaller regions usually cause invalid splits. Generally, valid splits can be performed by splitting the regions between the larger blobs, if they are close enough to one another. Details of the algorithm are shown in Algorithm 5.
Algorithm 5 Region Splitting Algorithm

connectedThresh = 70

toSplit = binary image containing the region to be split

Perform morphological opening with a structuring element proportional to the region size

hullArea = convex hull region of toSplit

hullOnly = hullArea AND ¬toSplit

Remove blobs of hullOnly with number of connected pixels < connectedThresh

if # blobs > 1 then

   for id = 1 → #blobs do

      Find $blob_{min}$, the closest blob to $blob_{id}$

      if no split already made between $blob_{id}$ and $blob_{min}$ then

         $w_{ij} = [x_{i,hull} - x_{j,hull}, y_{i,hull} - y_{j,hull}]$

         Find $i', j'$ in order to find $\min_{i',j'} |w_{i'j'}|$

         Set pixels along a line between $[x_{i'}, y_{i'}]$ and $[x_{j'}, y_{j'}]$ to 0 (split)

      end if

   end for

end if

Figure 4.15 shows the flow of the region splitting algorithm, starting with the original region after undergoing a morphological open operation. The next image in the figure shows the regions belonging only to the convex hull region, after removing smaller blobs. This is followed by the marked image, where the "**" symbols represent the points closest to the edge of the closest blob. The final image shows the result of splitting the original region.

Figure 4.15: Images showing the flow of the split algorithm.
4.2.6 Estimation and Analysis of the Size Distribution

Estimating the size distribution for the rocks on the conveyor belt is a simple procedure. Once the segmentation is complete, the algorithm iterates through each segment, calculating the convex hull points of each segment and then finding the maximum distance between any two of the points. This distance in pixels is then multiplied by the $\frac{1}{22.3}$ inches per pixel conversion to get a maximum dimension in inches.

The percent passing is then calculated by identifying what percentage of the segments have a maximum dimension below each threshold. For all of the results presented in this chapter, these thresholds will be inch values $1, 2, \ldots, 6$. Therefore, if the rocks are not supposed to surpass a maximum dimension of 4 inches, and the percent passing for 4 inches is 70%, then an alarm can be set so that operators can stop the conveyor belt and make the necessary adjustments.

4.3 User Interface

A graphical user interface (GUI) was developed to assist in the debugging and also provide an operator with a way to control system parameters, visualize the percent passing, and record data. Figure 4.16 displays the user interface, when no alarm has been set, and with the percent passing bar graph displayed.
At a cement plant, 100 images were taken, each taken 1.5 seconds apart to ensure no overlap. By clicking the “Run” button (1) in the GUI, the images were then be read from a folder to simulate capturing the images with the conveyor belt running. Each image was processed, for which the percent passing is calculated and displayed (2). The user can also control the number of images to be processed before making an estimate of the percent passing (3), where this estimate is the median percent passing of these images. This is done in an attempt to remove outliers. A bar graph of this percent passing is then displayed (4), which is updated every estimate. The user can control the thresholds for the percent passing by selecting an inch value (5), and setting a percent passing threshold (6). A violation occurs if the percent passing estimate for the corresponding inch value falls below that threshold. The image representing a light signal (7) is gray if the conveyor belt is static, green if moving with no violations having occurred for a user-defined number of images, and red if too many...
violations occur. When too many violations occur, the light (7) will turn from green, or gray, to red. The number of estimates containing violations can be set (8), which will be referred to as $N$, where the number of violation counted by

\[
\# \text{ violations} = \# \text{ images with violations} - \# \text{ images with no violations}
\]  

(4.10)

where "# images with violations” and "# images with no violations” are set to 0 if the following condition is false.

\[-N \leq \# \text{ violations} \leq N\]  

(4.11)

For the example in Figure 4.16 the threshold for the percent passing for 4 inches was set to 70%. The number of images per estimate was set to 3, and the number of violations to be observed before setting the alarm was set to 2. Therefore, a minimum of 6 images need to be processed before setting the status image (7) to green or red. In this example, the percent passing remains above 70% for 4 inches, and so the status image is set to green.

The GUI also allows for the percent passing to be logged by checking a box (9). The graph type can also be toggled between percent passing, and the percentage of rocks within different ranges of inch values (10). To terminate the programs operation, the ”Stop” button can be pushed (11). Figure 4.17 shows an example where the threshold for 4 inches was set to 90, and enough violations were observed in order to set the alarm.
4.4 Results

This section details the results of the machine learning algorithms and segmentation accuracy.

4.4.1 Machine Learning

During the image collection process, 100 images were collected. The data from the first 25 of these images was used for training each of the machine learning algorithms. The data includes the values for each of the 6 features used during training, as well as the output, which identifies whether or not that region was valid or not. The following machine learning
classifiers were trained on this data:

- Neural Network
- k-Nearest Neighbor
- Support Vector Machine
- Naive Bayes
- Decision Tree

In order to determine the accuracy of these classifiers, the dataset was randomly split so that \( \frac{2}{3} \) would be used for training, and \( \frac{1}{3} \) would be used for testing. The accuracy is then recorded as the percentage of times that the classifier correctly identifies the output of in the test data. To increase the robustness of this method, the average accuracy over 20 trials was calculated.

For each classifier, the combination of features to be used during training was determined by iterating through every possible combination of features, and selecting the combination that maximizes the test accuracy. Table 4.3, from the Rock Classification section, contain the features and corresponding ID values. Table 4.4 shows the features used for each classifier.

**Table 4.4**: Features used for each classifier

<table>
<thead>
<tr>
<th>Classifier</th>
<th>IDs of Features Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Network</td>
<td>1, 2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>k-Nearest Neighbor</td>
<td>1, 2, 4, 5</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>1, 2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>1, 2, 3, 4, 5, 6</td>
</tr>
</tbody>
</table>

For the neural network, 1 hidden layer was used, with 10 neurons in this layer. This number was found by values 1 through 20 as the number of neurons at the hidden layer. For the k-nearest neighbor classifier, the best k value was found to be 16, which is the number of nearest neighbors that the classifier identifies before making a classification. The test
accuracy for each of the classifiers is shown in Table 4.5

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Test Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Network</td>
<td>86.79</td>
</tr>
<tr>
<td>k-Nearest Neighbor</td>
<td>86.54</td>
</tr>
<tr>
<td>Support Vector Machine</td>
<td>88.20</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>86.70</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>83.27</td>
</tr>
</tbody>
</table>

As the SVM results in the highest test accuracy, it was the classifier used in the final system.

4.4.2 Accuracy

To estimate the final accuracy of the system, each of the final segments for the next 25 images (26-50) following the training images were classified as one of the following:

- Valid contour
- Invalid contour, but maximum dimension captured
- Invalid
- Undecided

Ultimately, the only feature of the final segments that will be considered when generating an estimate of the size distribution will be the maximum dimension. There are many instances where part of the rock in the ground truth is not included in the segment. However, it is still possible that the maximum dimension is captured by the segment. For this reason, two different accuracies will be considered. One accuracy is presented with valid regions considered to be any segment that captures the maximum dimension but not the full contour, and another accuracy is presented where a segment is valid only if it captures the full contour of the rock, which also means that the maximum dimension is captured, as well. Accuracies with and without the split algorithm will also be presented to assess the split algorithm's
performance.

There are also resulting segments where the validity cannot be determined. These regions have the potential to be valid, but contain too much uncertainty to label it as valid or invalid. They are therefore not included in the accuracies presented in this section. However, the percentage of regions which were undecided is presented. The final accuracies are shown in Table 4.6.

Table 4.6: Average overall accuracies

<table>
<thead>
<tr>
<th></th>
<th>Accuracy Max Dimension(%)</th>
<th>Accuracy Contour(%)</th>
<th># Regions</th>
<th>Undecided(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split Regions Only</td>
<td>71.96</td>
<td>62.62</td>
<td>238</td>
<td>10.08</td>
</tr>
<tr>
<td>Without Split</td>
<td>83.06</td>
<td>75.72</td>
<td>1180</td>
<td>2.97</td>
</tr>
<tr>
<td>All Regions</td>
<td>81.31</td>
<td>73.65</td>
<td>1418</td>
<td>4.16</td>
</tr>
</tbody>
</table>

Table 4.7 shows the accuracies at different size ranges for capturing the maximum dimension of the rocks. The accuracies are shown only for the size ranges that contained at least twenty regions.

Table 4.7: Accuracies for capturing the maximum dimension of the rocks at different size ranges.

<table>
<thead>
<tr>
<th>Range (inches)</th>
<th>Range (pixels)</th>
<th>Overall Accuracy (%)</th>
<th>Split Accuracy (%)</th>
<th>Without Split Accuracy (%)</th>
<th># regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-3</td>
<td>44.7-67.0</td>
<td>82.85</td>
<td>69.33</td>
<td>85.84</td>
<td>447</td>
</tr>
<tr>
<td>3-4</td>
<td>67.0-89.4</td>
<td>85.48</td>
<td>80.28</td>
<td>86.35</td>
<td>515</td>
</tr>
<tr>
<td>4-5</td>
<td>89.4-111.7</td>
<td>78.28</td>
<td>70.73</td>
<td>79.65</td>
<td>272</td>
</tr>
<tr>
<td>5-6</td>
<td>111.7-134.0</td>
<td>76.47</td>
<td>55.56</td>
<td>80.20</td>
<td>119</td>
</tr>
<tr>
<td>6-7</td>
<td>134.0-156.4</td>
<td>67.39</td>
<td>75.00</td>
<td>66.67</td>
<td>46</td>
</tr>
</tbody>
</table>

Notice that the size range with the most number of regions was also the most accurate. This size range of 67 to 89.4 pixels is equivalent to 4.74% to 6.32% of the diagonal dimension of the image respectively, where the diagonal is calculated to be 1413.6 pixels for the 1280 pixels x 600 pixels cropped image.
Table 4.8 shows the accuracies at different size ranges for capturing the entire contour of the rocks. Again, the accuracies are shown only for the size ranges that contained at least twenty regions.

**Table 4.8:** Accuracies for capturing the contour of the rocks at different size ranges.

<table>
<thead>
<tr>
<th>Range (inches)</th>
<th>Range (pixels)</th>
<th>Overall Accuracy (%)</th>
<th>Split Accuracy (%)</th>
<th>Without Split Accuracy (%)</th>
<th># regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-3</td>
<td>44.7-67.0</td>
<td>77.54</td>
<td>60.00</td>
<td>81.42</td>
<td>447</td>
</tr>
<tr>
<td>3-4</td>
<td>67.0-89.4</td>
<td>78.63</td>
<td>74.65</td>
<td>79.29</td>
<td>515</td>
</tr>
<tr>
<td>4-5</td>
<td>89.4-111.7</td>
<td>68.91</td>
<td>60.98</td>
<td>70.35</td>
<td>272</td>
</tr>
<tr>
<td>5-6</td>
<td>111.7-134.0</td>
<td>61.34</td>
<td>38.89</td>
<td>65.35</td>
<td>119</td>
</tr>
<tr>
<td>6-7</td>
<td>134.0-156.4</td>
<td>58.70</td>
<td>50.00</td>
<td>59.52</td>
<td>46</td>
</tr>
</tbody>
</table>

Using the 1 inch to 22.3 pixels conversion, Table 4.9 shows the average errors for the invalid regions. During the accuracy classification, if a region was assigned an invalid label, then the true maximum was manually selected. If there was an undersegmentation, then this maximum dimension was chosen to be the maximum dimension for the largest rock located within the segment. If there was an oversegmentation, then the maximum dimension was chosen to be the rock that takes up the most area of the segment.

**Table 4.9:** Average errors for the invalid regions.

<table>
<thead>
<tr>
<th></th>
<th>Average error (inches)</th>
<th>Average error (pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>All invalid regions</td>
<td>0.9229</td>
<td>20.62</td>
</tr>
<tr>
<td>Invalid split regions</td>
<td>0.816</td>
<td>18.223</td>
</tr>
<tr>
<td>Invalid non-split regions</td>
<td>0.956</td>
<td>21.355</td>
</tr>
</tbody>
</table>

One way to evaluate these errors is to consider allowing a valid estimate of the maximum dimension to be one that estimates within a specific percentage of the true maximum dimension. For example, if a 5% tolerance is allowed, then the overall accuracy is increased from 81.31% to 82.41%.
A 95% confidence interval (CI) for each of these accuracies was created, where the confidence interval is defined as

\[ CI = \hat{p} \pm z_{\alpha/2} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \] (4.12)

The table containing these confidence intervals is shown in Table 4.10.

**Table 4.10:** Confidence intervals for accuracies

<table>
<thead>
<tr>
<th></th>
<th>CI for Accuracy Max Dimension(%)</th>
<th>CI for Accuracy Contour(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split Regions Only</td>
<td>[65.94, 77.98]</td>
<td>[56.13, 69.10]</td>
</tr>
<tr>
<td>Without Split</td>
<td>[80.88, 85.23]</td>
<td>[73.24, 78.20]</td>
</tr>
<tr>
<td>All Regions</td>
<td>[79.24, 83.38]</td>
<td>[71.32, 76.00]</td>
</tr>
</tbody>
</table>

Various factors contribute to the error, including the segmentation and the SVM. There are many different sources of error in the segmentation, which are generally caused by irregular surfaces that can sometimes appear on some of the rocks. These irregular surfaces often lead to oversegmentation. Undersegmentation can also become a problem if an edge cannot be detected between neighboring rock regions if the structuring element used in the morphological image processing is too large. Errors from SVM will also result, as there will always be false negatives in the classification as a result of Bayes error.

As opposed to calculating the false positive rate, the percentage of the image that belongs to one of the segments is calculated. One percentage is calculated as the percentage of overall pixels that belong to the segments, and the other is calculated as the percentage of pixels above the threshold, calculated during the adaptive thresholding step, that belong to one of the segments. These percentages are shown in Table 4.11.
Table 4.11: Overall Accuracies

<table>
<thead>
<tr>
<th></th>
<th>Percentage Unused</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall pixels</td>
<td>82</td>
</tr>
<tr>
<td>Thresholded Regions</td>
<td>73</td>
</tr>
</tbody>
</table>

The final segments of the example input image, shown above in Figure 4.4, that were determined to be valid by the SVM are shown in Figure 4.18.

Figure 4.18: Final result of the rock segmentation and classification (left) and overlay (right).

4.4.3 Computation Time

The computation time will depend on whether or not the split algorithm is being applied. The average computation time over 50 images is shown in Table 4.12.

Table 4.12: Computation times for oversize rock detection

<table>
<thead>
<tr>
<th></th>
<th>Computation Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>With Split</td>
<td>2.45</td>
</tr>
<tr>
<td>Without Split</td>
<td>1.67</td>
</tr>
</tbody>
</table>

In order for the system to operate in real-time, this means having a maximum conveyor belt speed of 117.14 ft/min with the split algorithm running, and 171.85 ft/min without the split algorithm running. Improvements to the computation time could be made by converting all
of the code to C++ instead of MATLAB. Another possibility would be to parallelize the code to have it run on the multi-core architecture of the system. The code could also be written to use a GPU.
Chapter 5

Stockpile Volume Measurement

The final process explored of a quarry’s operation is the volume estimation of the stockpiles where the final product is placed. This is a useful way of providing an accurate estimation of the volume for inventory purposes. Current methods for estimation include using the height, diameter, and shape of the pile to estimate volume. These methods are only accurate if the stockpile closely takes on the shape used to for the estimate, and if the measurements of height and diameter are very precise. As this is generally not the case, a vision based system provides an opportunity for a more reliable estimation of the volume.

Three different stockpiles were used as examples in this chapter. The first is a larger pile containing less variation on the surface than a standard stockpile containing only gravel. This was presented to show the ability of the system to reconstruct different types of piles, and also the ability to eliminate nearby objects, such as the conveyor belt that drops the material onto the stockpile. The second pile was a medium-sized stockpile, which allowed for testing the theoretical positions of the cameras when imaging a pile from a unmanned aerial vehicle (UAV) by mounting two cameras on top of a tripod. The third, and final, pile presented is a smaller pile that provided a ground truth measurement of the volume to use for comparison with the algorithm used to measure the volume of the stockpile. Figure 5.1 shows images of the example stockpiles used in this chapter.
5.1 System Design

The two main parts of the stockpile volume estimation are the image acquisition and the volume estimation. Figure 5.2 shows the flow of the system’s operation.

![Figure 5.2: Flow of the stockpile volume estimation process.](image)

5.1.1 Image Acquisition

By using two cameras, separated by a fixed baseline, the initial 3D reconstruction can be scaled so that a volume estimation of the stockpile can be performed. Torok determined that for measuring the volume of an object using a $360^\circ$ view, pictures should be taken so that shifts of no more than $15^\circ$ should take place, as SIFT features unable to find matches beyond this. This means that a minimum of $\frac{360^\circ}{15^\circ} = 24$ images should be taken. However, Torok found in his tests that 24 images at low resolution (640x480), taken in a circle, was not
always enough to provide a decent reconstruction, but 48 was [39]. This number was used as a minimum number of images to take when circling the stockpile, where 24 pairs of images (from the left and right cameras separated by a fixed baseline) is the suggested minimum. Additional images of the stockpile are unlikely to prevent a decent reconstruction, but will add to the amount of time Bundler and PMVS will take to finish.

The large stockpile was imaged using the stereo boom from designed and built in the USL. Images of the medium-sized and smaller stockpiles were imaged using two Canon Powershot TX1s. The cameras were mounted on a piece of 80/20 so that they would remain rigid when the images were being taken. Color images were captured and downsampled to a resolution of 640x480 to test that the stockpiles could be reconstructed from low resolution imagery. A total of 24 left and right image pairs were captured by circling each of the stockpiles.

5.2 Volume Estimation

There are different options for estimating the volume of the stockpiles. The method used by Torok in [39] was to use voxels. This type of method is necessary to measure the volume of many different objects. However, considering the shape of a stockpile, if starting at the top of the pile and move towards the base, the distance to the center of the pile can be represented by a monotonically increasing function. This allows for a simplification of the volume measurement, as it can now be done by projecting the points onto a plane fit through the base of the stockpile, where the distances of each point to the plane can be used in a mesh procedure detailed later in this section.

5.2.1 Stockpile Extraction

In order to perform the volume estimation, the points that correlate to the stockpile must first be extracted from the generated 3D point cloud. This is currently accomplished by using one of the Point Cloud Library’s (PCL) packages to select n 3D points. The first n − 1 points are selected around the base of the stockpile. A plane is fit through these
points, where a normal vector to the plane is used to rotate the points so that the $Z$ values correspond to the height values of the stockpile. Once all of the points have been rotated, the points with XY positions outside of the area on the XY plane defined by these selected points are eliminated. This area is computed with MATLAB’s `inpolygon` function. The $n^{th}$ point that is selected is at the tip of the pile. This is used to eliminate any points above the stockpile. It can also be used to change the sign of the $Z$ values if the points were rotated below the XY plane.

### 5.2.2 Mesh Analysis

To estimate the volume of the stockpile, all of the inlier points are projected onto the XY plane. A grid is then created, where the 3D points may or may not fall within each one of the cells. Figure 5.3 shows an example of an extremely sparse, and unrealistic, 3D reconstruction of the stockpile, where the grid size is only 8x8. However, this helps to visualize what the algorithm is doing. For this example, it can be assumed that the convex hull points of the projected points are the selected points around the base.

![Figure 5.3: Example of grid used after inlier points of the stockpile projected onto the XY plane](image)

Figure 5.3: Example of grid used after inlier points of the stockpile projected onto the XY plane
Once all of the points have been projected, the cells that are within the convex hull area are marked. The Z values for each of the points that lie within the same cell are then averaged and stored at a location in a matrix that corresponds to that cell. The cells that did not contain any points, that are also located within the convex hull area, are then marked to be filled by finding averaging the Z values of the k-nearest neighbors, where the neighbors are the cells that contain at least one point. All of the tests presented in this section used a k value of 1. Figure 5.4 shows the cells of the above example that contain averaged Z values in blue, and the cells that need to be filled in red.

Figure 5.4: Grid after the Z values have been averaged and cells to be filled have been identified. Blue cells contain averaged Z values, and red cells need to be filled using the k-nearest neighbor algorithm.

Since the grid’s height and width are in pixels, when performing the summation over all of the cells, the pixel to meters conversion must first be calculated. This can be calculated by

\[
\text{pixelsToMeters} = \frac{\| (select_i - select_j)(\text{meters}) \|_2}{\| (select_i - select_j)(\text{pixels coordinates}) \|_2}, \quad i \neq j
\]  

(5.1)

where \( select_i \) and \( select_j \) represent selected points from the base of the stockpile. The algorithm for the volume measurement is shown below.
Algorithm 6 Volume measurement

Initialize grid for volume measurement

Set numberCellsH, numberCellsW

cellWidthMeters = \( \frac{grid\text{-}Width}{number\text{-}CellsW} \) (pixelsToMeters)

cellHeightMeters = \( \frac{grid\text{-}Height}{number\text{-}CellsH} \) (pixelsToMeters)

cellArea = (cellWidthMeters)(cellHeightMeters)

gridPoints = Rotated/transformed stockpile 3D points projected onto grid

stockpileHullRegion = Convex hull region of rotated/transformed 3D selected points from
base of stockpile projected onto grid

Initialize all grid.currCell.averageZ to 0

for all currCell ∈ grid do

  localGridPoints = gridPoints ∈ currCell

  if localGridPoints.length > 0 then

    grid.currCell.averageZ = average(localGridPoints.Z)

  end if

end for

Volume = 0

for all currCell ∈ grid do

  if grid.currCell.averageZ = 0 then

    grid.currCell.averageZ = average(k-nearest non-zero neighbors)

  end if

end for

for all currCell ∈ grid do

  if currCell ∈ stockpileHullRegion then

    Volume = Volume + (grid.currCell.averageZ)(cellArea)

  end if

end for
5.3 Aerial Imagery

One of the goals for this work is to be able to create an image-based 3D reconstruction of a stockpile from flight imagery. Current methods involve having a lidar survey performed, which can be expensive depending on different factors, such as the frequency of the surveys. With a UAV, such as a quadrotor, images can be taken from the air during an autonomous mission, which can then be used to generate an analytical model of the stockpile and determine the volume. This provides a cheaper alternative to lidar surveys, while maintaining a competitive accuracy for the volume measurement.

To develop a basis for future work in this area, formulas were developed to provide the ideal camera positions during the flight, which are based on the height and maximum diameter of the stockpile. Camera positions are chosen that keep the manned or unmanned aircraft as close to the center of the base of the stockpile as possible, while imposing the constraint that two points beyond the top and base of the pile must be kept within view. This ensures that as much detail of the surface as possible can be captured, while keeping the whole stockpile within view. Figure 5.5 illustrates this concept, where the black circles are the points beyond the top and base of the pile that the camera should keep within view, and the green star is the center of the base of the stockpile.

![Figure 5.5: Illustration of where the quadrotor should position itself when imaging a stockpile](image)

However, this method is dependent on the focal length of the cameras, as with a narrower
focal length, the quadrotor will have to position itself further from the pile than with a wider focal length. Another problem is that two cameras are needed in order to provide scale information. For the tests performed in this section, two Canon Powershot TX1 cameras were used. The focal length for this camera is 47.92° for the wider view, and 36.86° for the narrower view. Figure 5.6 shows the setup for the cameras on the 80/20, which is attached to the tripod and elevated to its maximum height.

Figure 5.6: Setup of the cameras on the tripod, which is used to simulate acquiring images from a UAV in-flight.
5.3.1 Calculating the Ideal Camera Positions

To calculate the ideal camera positions, a side view of the stockpile, with the maximum diameter captured, is considered. It is assumed that the center of the base of the stockpile is the origin in a 2D coordinate system. The parameters for this problem are shown in Table 5.1.

Table 5.1: Parameters for calculating ideal camera positions.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FOV_V$</td>
<td>vertical field of view</td>
</tr>
<tr>
<td>$FOV_H$</td>
<td>horizontal field of view</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>angle between the camera’s direction and the gravity vector</td>
</tr>
<tr>
<td>$h$</td>
<td>height of the stockpile</td>
</tr>
<tr>
<td>$r$</td>
<td>maximum radius of the stockpile</td>
</tr>
<tr>
<td>$P_h$</td>
<td>2D coordinate for the tip of the stockpile, $(0, h)$</td>
</tr>
<tr>
<td>$P_r$</td>
<td>2D coordinate for the outer base of the stockpile, $(r, 0)$</td>
</tr>
<tr>
<td>$P_1$</td>
<td>point beyond the tip of the stockpile to be kept within view</td>
</tr>
<tr>
<td>$P_2$</td>
<td>point beyond the outer base of the stockpile to be kept within view</td>
</tr>
<tr>
<td>$P_c$</td>
<td>camera position to be calculated</td>
</tr>
</tbody>
</table>

To calculate $P_1$ and $P_2$, it first needs to be considered how far these points should be from $P_h$ and $P_r$ respectively. In the test performed for this section, the following ratio was arbitrarily chosen for determining this distance:

$$ratio = \frac{||P_1 - P_2||_2}{||P_h - P_r||_2} = 1.4$$  \hspace{1cm} (5.2)

Once an actual UAV is used, this ratio may change depending on various factors, such as the in-flight stability.

Using this ratio, $P_1$ and $P_2$ are then calculated by
\[ \Delta X_r = (P_r.X - P_h.X)(\frac{ratio - 1}{2}) \]  
\[ \Delta Y_r = (P_r.Y - P_h.Y)(\frac{ratio - 1}{2}) \]  
\[ \rightarrow P_1 = P_h - (\Delta X_r, \Delta Y_r) \]  
\[ \rightarrow P_2 = P_r + (\Delta X_r, \Delta Y_r) \]  

Assuming that \( FOV_V \) is less than 90°, the angles of the stockpile and camera can be used to calculate the following:

\[ S = \frac{||P_1 - P_2||_2}{2(\tan \left( \frac{FOV_V}{2} \right))} \]  
\[ \Delta X_c = (S)\sin(\alpha) \]  
\[ \Delta Y_c = (S)\cos(\alpha) \]  
\[ \rightarrow P_c = \frac{P_1 + P_2}{2} + (\Delta X_c, \Delta Y_c) \]

Since two cameras are being used, it is also a requirement that the stockpile remain within view in both images during the image taking process. Assuming that the cameras are in the fronto-parallel position, with the baseline perpendicular to the direction between the center of the tripod and the center of the pile, the maximum baseline can be calculated that will still keep the stockpile within view. This is calculated by

\[ B_{max} = (2) \left( 2||P_3|| \tan \left( \frac{FOV_H}{2} \right) - 2r \right) \]

To test that these camera positions were capable of imaging the medium-sized stockpile, the dimensions of the stockpile were first estimated, where

\[ \text{height} \approx 1.16cm \]  
\[ \text{diameter} \approx 3.32m \]

The Canon Powershot TX1 cameras, which were mounted on 80/20, were elevated on a tripod, which allowed the cameras to reach a maximum height of 274.3cm. With both cameras angled so that \( \alpha = 59.5^\circ \), the following calculation was made:
\[ P_c = (4.5m, 2.74m) \]  

To calculate \( B_{\text{max}} \), the case where the UAV is not able to keep both \( P_1 \) and \( P_2 \) within view, with ratio = 1.4, is considered, where it instead flies closer to the stockpile. For this, the ratio is lowered to 1.035, which will still allow for approximately 1.3 inches past the top and base of the pile. This gives

\[ P_c = (3.56m, 2.18m) \]  
\[ \rightarrow B_{\text{max}} = 0.708m \]

Another option would be to keep a constant distance to the base of the pile. The current method has been developed to have the UAV fly a circular path based on the maximum radius of the stockpile. This could be modified if an approximate shape of the stockpile was known, which would allow the UAV to fly closer to the stockpile when the shape of the stockpile is not conical.

\subsection{5.4 Accuracy Assessment}

For the large, and medium-sized stockpiles, presenting an image-based method to measure the volume of these stockpiles seems more probable to present an accurate estimate compared to calculating the volume of a cone, or other shapes, from estimates for the height and diameter. However, without a ground-truth, one cannot be certain of how accurate this method actually is. For this reason, a smaller stockpile, of known size, was created so that the accuracy of this method could be tested.

A bucket with a volume of 0.0244\( m^3 \), was filled twice with gravel, and then dumped onto a level surface, which resulted in a stockpile of approximately 0.0489\( m^3 \). There is, of course, some uncertainty in this volume, as there is some uncertainty in measurement of the bucket’s volume. Another source of uncertainty is the way in which the gravel piles up. The baseline for the cameras was set at 35.4cm, which was chosen so that the stockpile could be seen within both images.
Different parts of the algorithm for the volume measurement were evaluated to see what impact they had on the accuracy. The granularity of the grid was varied to see how much increasing the number of cells helped. A Poisson surface reconstruction was also applied to see if an interpolation of the surface helped increase the accuracy. An executable by [40] was used, which performs the Poisson surface reconstruction using the normal directions of the points.

5.5 Results

5.5.1 Stockpile Reconstructions

The same procedure for the 3D reconstruction of the bench face is carried out for the reconstruction of the stockpiles. Figure 5.7 shows the Bundler/PMVS 3D reconstructions of the three example stockpiles used in this chapter.

![Figure 5.7: Bundler/PMVS 3D reconstructions for the large pile (left), medium-sized pile (middle), and small pile (right)](image)

These 3D reconstructions demonstrate the ability of the two-camera system to generate dense enough point clouds to analyze and determine the volume of the stockpiles.

5.5.2 Analytical Models

Once the 3D reconstruction is complete, the volume calculation is performed, where each cell within the convex hull region contains a Z value. By using the `surf` function in MATLAB,
an analytical model can be viewed. Figure 5.8 shows the analytical models for each of the example stockpiles presented in this chapter.

Figure 5.8: Analytical models for the large pile (left), medium-sized pile (middle), and small pile (right)

Note that when the models in Figure 5.8 are juxtaposed that they are not representative of the scale. Modifying the output window can skew the shape, and therefore these shapes should not be compared to the Bundler/PMVS outputs, which better represent the shapes of the stockpiles. These models do, however, show the ability to extract the regions of interest, and also perform the necessary rotation so that the Z axis represents the height values for the stockpiles.

5.5.3 Accuracy

To test the accuracy of the system, the small stockpile was used, since the true volume of this stockpile is known. The approximate value of the volume is 0.0489m$^3$. The volume of the stockpile is estimated using the algorithm presented earlier in this chapter. This algorithm was tested both with and without the Poisson surface reconstruction applied.

Once the 3D reconstruction was output by Bundler and PMVS for this stockpile, the point clouds were combined and scaled to meters using the known baseline for the cameras. The user then selected multiple points around the base of the stockpile, and then selected a point at the tip, which was used to remove outliers that might interfere with the volume calculation. A plane is then fit through the points selected at the base of the stockpile, the normal vectors of which are used to rotate the points so that this plane becomes the XY plane. In case the points need to be shifted along the Z-axis, the median Z value for
the rotated selected points is used to shift all of the points up or down so that all of the points at the base of the stockpile are approximately zero. Figure 5.9 shows the result of extracting the region of interest from the point cloud, where the region in red represents the inlier points, or points corresponding to the stockpile, the green points represent all other points, and the blue points represent the plane fit through the selected points at the base of the stockpile.

Figure 5.9: Color coded point cloud for the small stockpile. Red represents inliers, green represents all other points, and blue represents the plane fit through the selected points at the base of the stockpile

It is not important that all of the selected points for the base of the stockpile be exactly at the edge. Points selected further from the base will still work as long as it is a level surface in the region, and nearby regions, between the selected point and the base of the pile. As long as there is a level surface in the area, this should not contribute to the volume, as these points should have zero, or approximately zero, height values.

Once the inlier points are extracted, the Poisson surface reconstruction by [40] can be applied to these points. Figure 5.10 shows the result of applying the Poisson surface reconstruction to the inlier points of the small stockpile.
Figure 5.10: Poisson surface reconstruction for the inlier points of the small stockpile.

The volume calculation was then performed, both with and without the Poisson surface reconstruction. The granularity of the grid was first determined based on the average number of points projected into each cell. For an NxN grid, N was determined by iterating through values between 5 and 300. When a value of N was found that yielded the average number of points per cell to be below 10, the iteration terminated. It was also checked that the difference in volume measurement between an NxN grid and an (N-1)x(N-1) grid was reasonably small (< 0.05%). The value of N for the small pile was calculated to be 203 when using the points output by the Poisson surface reconstruction. The same grid granularity (203x203) was used in the volume estimation without the Poisson surface reconstruction applied. The results with and without the Poisson surface reconstruction are shown in Table 5.2.
Table 5.2: Results for the stockpile volume measurement of the small stockpile without the Poisson surface reconstruction.

<table>
<thead>
<tr>
<th></th>
<th>Without Poisson</th>
<th>With Poisson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume before 1-NN fill</td>
<td>0.04708m³</td>
<td>0.047246m³</td>
</tr>
<tr>
<td>Volume after 1-NN fill</td>
<td>0.04729m³</td>
<td>0.047456m³</td>
</tr>
<tr>
<td>Error</td>
<td>3.22%</td>
<td>2.888%</td>
</tr>
<tr>
<td>Percent filled by 1-NN</td>
<td>0.616%</td>
<td>0.42%</td>
</tr>
<tr>
<td>Average points per cell</td>
<td>6.738</td>
<td>9.928</td>
</tr>
</tbody>
</table>

It was also observed that the difference between a granularity of 202x202 and 203x203 made only a 0.0123% difference. Using the Poisson surface reconstruction dropped the error by 0.332%, while increasing the average number of points per cell by 3.19. Both of the errors presented are acceptable considering the multiple sources of error, such as the scale for the point cloud, uncertainty in the bucket measurement, and other factors.

The volume for the other example stockpiles were calculated, but without the ground truth measurement, the accuracy of the estimates cannot be determined. The volume of the medium-sized stockpile was estimated to be 3.52m³, and the volume of the large pile was estimated to be 273m³. The only assessment of the accuracy that could be done for these estimations was using the fact that the estimations were found to be close to the corresponding volume of a cone given the heights and diameters of the stockpiles.
Chapter 6

Conclusions and Future Works

6.1 Conclusions

In this thesis, three different applications were developed to assist with quarry operations. The first application was for the blasting process, which involved using images to generate a scaled 3D reconstruction of a bench face, and then use this reconstruction to determine the borehole positions and angles based on user inputs. The second application was for the crushing process, where an image-based system was developed to identify when oversize rocks are on the conveyor belt by segmenting the images and estimating the maximum dimension for each region. The third, and final, application was to determine the volume of the stockpiles by generating image-based 3D reconstructions.

For the bench face profiling and drilling analysis, methods were presented to assign borehole locations and angles based on user inputs and the shape of the bench face. An image-based 3D reconstruction is generated by taking images from a two-camera system mounted in the back of a pickup truck. Users of the system can set the boreholes to be angled or vertical. A parallel or staggered blast pattern can also be chosen depending on the preference of the user, as well as the number of rows. The user can have the system calculate values for the burden, spacing, subdrilling, and stemming based on formulas from the SME Mining Engineering handbook, or if preferred, enter values for these parameters directly. The system was setup so that all of the borehole locations could easily be changed. A top-down view was presented, where the user can select multiple boreholes at once and then move them to a new location. Also within the top-view, the two traffic barrels used as reference points are marked. The distances, the directions of which are perpendicular and parallel to the line between the two
barrels, can be viewed to each of the borehole tops. A side view was also presented so that the user can view the burden at multiple sample points along an individual borehole, and then shift the borehole away from or to the bench face.

The second process of the quarry’s operation that was explored was the crushing stage, where algorithms were developed to identify oversize rocks on a conveyor belt. The algorithms to accomplish this started by segmententing the image using adaptive thresholding and morphological image processing. The resulting segments were then classified as rocks or non-rocks by using an SVM trained on the segments of 25 images. Multiple machine learning algorithms were tested, and it was found that SVM yielded the highest test accuracy. Segments that were rejected based on their shape were then passed to a split algorithm that attempted to separate the segment into multiple segments, where at least one would be a valid rock. These split segments were then classified as rocks or non-rocks using the same SVM. The resulting accuracy for this system was 81.31% for capturing the maximum dimension. It was found that the accuracy for the regions passed through the split algorithm and then classified as a rock was lower than the accuracy for the regions that did not pass through the split algorithm. These regions also had a higher percentage of undecided regions compared to the regions that did not pass through the split algorithm. The split algorithm also increased the average computation time by 0.78 seconds. Without the split algorithm, a real-time system is possible provided that the conveyor belt is running at a maximum speed of 171.85 ft/min. With the split algorithm, the conveyor belt must have a maximum speed of 117.14 ft/min, which adds another disadvantage to the split algorithm. However, without the split algorithm, less of the image is being used, and therefore there is a higher false positive rate. For a better assessment of the performance of the split algorithm, belt-cut data should be collected to see if the split algorithm helps provide a better estimate of the size distribution of the rocks.

The final process explored was the stockpile volume measurement. Three different example stockpiles were tested, all of which were successfully able to generate a 3D reconstruction using Bundler and PMVS. The first, and largest, stockpile was imaged from pickup truck using a similar mount to the one used for the bench face profiling, where the cameras were
mounted vertically. The second, medium-sized, stockpile was imaged from two cameras mounted on 80/20, and elevated using a tripod to simulate acquiring flight imagery from a UAV. The third, and smallest, stockpile was imaged with two cameras mounted on 80/20, which was held and carried around the stockpile to acquire images. Since all of the stockpiles imaged were conical, the volume estimate for the large and medium-sized stockpiles were compared to the volume of a cone given the estimates for the height and diameter of the stockpile. While the numbers were similar, this still did not provide an actual performance evaluation of the volume estimation algorithm. For this reason, a small stockpile, with known volume, was constructed to perform this test. The error was found to be 3.22%. By applying a Poisson mesh, this error dropped to 2.888%.

6.2 Future Works

The potential future research for each of the three different topics presented in this thesis ranges from improvements of the current methods to novel applications. The rest of this section will address the needed improvements for each of the applications, as well as presenting potential extensions these applications. One mutual need for both the bench face profiling and stockpile volume measurement is to generate image-based 3D reconstructions more quickly. Using Bundler and PMVS can sometimes cause the 3D reconstruction to take several hours to finish. One potential solution to this would be to use LIBVISO2 [41] to estimate the camera motion, as well as LIBELAS [42] to provide fast stereo maps. This could be used to generate real-time 3D reconstructions, which could then be analyzed in post-processing. This would expedite the process substantially. Another option is to consider using GPS and IMU information, along with stereo maps to generate a faster 3D reconstruction of the scene.

6.2.1 Bench Face Profiling and Drilling Analysis

The primary need of the bench face profiling and drilling analysis application is to have the system used by the individuals performing the blast. This would allow for any necessary
improvements to be pointed out. Another need is to test the system at more locations. Up until now, no difficulties have been encountered when extracting the region of interest. Currently, the first four selected points are used to define the area for region of interest, which may not be enough when imaging other benches. Extensions could also be made to the system by including additional blast patterns. Currently, only two blast patterns have been provided as a way to demonstrate the ability of the current algorithms.

6.2.2 Identifying Oversize Rocks

The problem of segmenting rocks on a conveyor belt with the use of 2D imagery has been around for many years. Some of the most recent papers in the area now suggest that research is shifting towards the use of laser profiling. However, these devices can still be significantly more expensive than most of the decent industrial cameras. 2D image segmentation is a very active research topic, and there is still a chance that a real-time 2D image segmentation could be developed that would work very well for this problem. The watershed transformation provides a reasonable segmentation when preprocessing is carried out on the image, but still occasionally has its shortcomings. Another 2D image segmentation tested briefly during the completion of this thesis was the Felzenszwalb segmentation [43]. The reason this segmentation was not used over the watershed transformation was because the watershed transformation had less computation time. Figure 6.1 shows the Felzenszwalb segmentation applied to the example image from Chapter 4 after morphological image processing was applied to the image in a preprocessing stage.
Figure 6.1: Felzenszwalb segmentation applied to the example image from Chapter 4, after morphological image processing was applied to the image.

One potential application of this work would be to monitor the performance of the crushers. Two imaging stations could be setup, where one is located at the entry to the crusher, and the other located at the exit. A comparison could then be done between the measured differences in size distributions and the expected difference in size distributions based on the parameters of the crusher. Both systems would still be able to monitor for oversize rocks.

Another potential application of this work would be to monitor the size of the rocks leaving the quarry. As the trucks leave the quarry, they enter a weigh station. Two cameras could be setup that would image the rocks in the back of the truck. A stereo map could be created to determine the depth to the rocks so that the appropriate scale could be determined. One of the cameras, or possibly two for increased robustness, could then perform the segmentation to estimate the size of the rocks leaving the quarry.

One way in which the estimation of the size distribution could be improved would be to start building a database with estimations of the size distribution by the program and their corresponding size distribution from the belt-cut. A transformation could then be found that minimizes the error between all of the transformed estimations of the size distribution and the true distribution from the belt-cut.
6.2.3 Stockpile Volume Measurement

The main future work for the stockpile volume measurement is to have the images acquired from a manned or unmanned aircraft. The stockpile volume measurement process, described in Chapter 5, could then be applied to determine the volume of the stockpile, which could then be logged. Other potential applications exist using the algorithms for the stockpile volume measurement. All of the current tests were done for stockpiles made from gravel, but the algorithms could be applied to other types of stockpiles, such as those made by coal.
Bibliography


Appendix A

Rock Classification Examples
Appendix B

Split Algorithm Examples

![Diagram showing the process of splitting shapes through a series of steps.](image-url)