Autonomous Sample Collection Using Image-Based 3D Reconstructions

Matthew M. Torok

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
Mechanical Engineering

Kevin Kochersberger, Co-Chair
Mani Golparvar-Fard, Co-Chair
Corina Sandu

April 26, 2012
Blacksburg, Virginia

Keywords: Robotics, Scoops, Image-based 3D Reconstruction,
Autonomous Sample Collection, Building Crack Detection

Autonomous Sample Collection Using Image-Based 3D Reconstructions

Matthew M. Torok

ABSTRACT

Sample collection is a common task for mobile robots and there are a variety of manipulators available to perform this operation. This thesis presents a novel scoop sample collection system design which is able to both collect and contain a sample using the same hardware. To ease the operator burden during sampling the scoop system is paired with new semi-autonomous and fully autonomous collection techniques. These are derived from data provided by colored 3D point clouds produced via image-based 3D reconstructions. A custom robotic mobility platform, the Scoopbot, is introduced to perform completely automated imaging of the sampling area and also to pick up the desired sample. The Scoopbot is wirelessly controlled by a base station computer which runs software to create and analyze the 3D point cloud models. Relevant sample parameters, such as dimensions and volume, are calculated from the reconstruction and reported to the operator. During tests of the system in full (48 images) and fast (6-8 images) modes the Scoopbot was able to identify and retrieve a sample without any human intervention. Finally, a new building crack detection algorithm (CDA) is created to use the 3D point cloud outputs from image sets gathered by a mobile robot. The CDA was shown to successfully identify and color-code several cracks in a full-scale concrete building element.

Acknowledgments

I would like to take a moment to thank all of the individuals who have helped me with the work presented in this thesis and throughout my engineering education. First and foremost, my advisor Dr. Kochersberger got me to come to Virginia Tech and offered me a Research Assistantship at the Unmanned Systems Laboratory. He promised that the lab did cool stuff, and despite some trials and tribulations along the way, this has been borne out. I would also like to thank Dr. Golparvar-Fard for his assistance with all of the image processing work I did. His class inspired me to think of new ways to apply 3D reconstructions to robotics that formed the core of my thesis work.

I certainly must acknowledge the tremendous support I received from all the student members of the Unmanned Systems Lab. Kenny Kroeger, Bryan Krawiec, Pete Fanto, Bob Collins, Justin Stiltner, Michael Bromley, Troy Probst, and Donnie Rodgers, you guys have been an invaluable resource for all kinds of engineering knowledge. Without your everyday assistance this project would've been impossible. Also, among graduates, I thank Jerry Towler, Jason Gassaway, Shajan Thomas, and Eric Gustafson, who overlapped with my time here.

Finally, I would like to thank my parents and siblings for their love, support, and birthday cake throughout my college career. I'd also like to thank Jaret Matthews, my mentor at the Jet Propulsion Lab in Pasadena, CA, where I first learned about the awesomeness of robotics, and all others who have helped me develop my God-given talents during my years in school.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Acronyms

BPC	Bundler Point Cloud
CDA	Crack Detection Algorithm
EXIF	EXchangable Image Format (photo properties tag)
IMU	Inertial Measurement Unit
PMVS	Patch Multi-View Stereo
PPC	PMVS Point Cloud
RANSAC	RANdom Sample Consensus
RGB	Red Green Blue color triplet
SfM	Structure-from-Motion
USL	Unmanned Systems Laboratory

Chapter 1

Introduction

This thesis presents work conducted at the Unmanned Systems Laboratory (USL) at Virginia Tech relating to the development of an autonomous sample collection system and associated collection mechanisms. The sample collection system is part of the USL's overarching mission of developing an autonomous robotic response system to be deployed following a hazardous event. One scenario closely examined as a type of hazardous event would be environmental contamination following a release of radioactive material (e.g. the disaster at the Fukushima Daiichi nuclear power plant in Japan). After such an occurrence the robotic system, consisting of a helicopter and ground robot, could be remotely deployed to gather valuable information (via visual inspection, radioactivity mapping, etc.) and collect samples from the affected site, keeping emergency personnel out of the danger zone. Apart from its incorporation into the overall USL system, the sample collection efforts described herein can be performed by, and were tested on, an independent robotic platform. An image-based three dimensional reconstruction method is used to generate a map of the terrain in the target sampling area as well. This technique, combined with post-processing and a scoop sample collector, represents a novel approach to autonomous sample collection. Furthermore, the 3D reconstruction method, along with other algorithms, is applied to demonstrate crack detection in damaged concrete building elements.

1.1 Motivation

The primary motivation behind the development of the USL's robotic system is to keep first responders out of harm's way following a hazardous event. This separation will help ensure their safety while enabling them to carry out mission-critical surveying and analysis of the affected site. Therefore, the helicopter and ground robot can be remotely operated to send critical data back to the operator's ground station. Helicopter payloads include a stereo boom (for large-scale 3D terrain reconstruction), radiation pod (for measuring radiation levels), and gimbaled camera (for visual inspection of a site). The helicopter can also deploy the small ground robot via a winched tether for sample collection.

Figure 1.1: Virginia Tech USL Yamaha RMAX autonomous helicopter

Because sample collection is one of the mission goals, it is sensible to develop methods to assist an operator in acquiring the sample(s) of interest. A mobile ground robot platform capable of navigating post-disaster terrain had been previously created by the USL [1]. This platform accommodates a variety of sample collection systems via an interchangeable payload tray mounted in the front of the robot. Prior to this research, two sample collection systems which fit into the payload tray had been developed.

The first collection system uses jets of pressurized air to collect particulate samples [2]. The

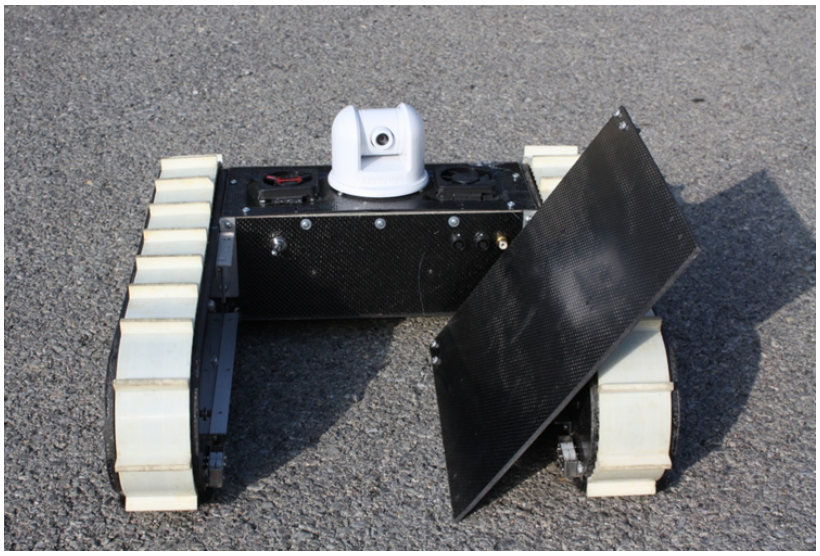


Figure 1.2: Ground sampling robot platform with interchangeable payload tray

air flows across an exposed section of the ground's surface, entraining small particles. These particles travel up a ramp and are deposited into a collection box incorporated into the device. The second collect system is designed to operate at the opposite end of the particle size spectrum. It is a 3 degrees of freedom (DOF) robotic arm with a gripper end effector to grasp large "chunk" samples up to 250g [3]. Since these two systems already existed at the USL, effort was spent to examine alternative means of collection. Work would focus on developing a system which was lighter and/or easier to use from an operator's standpoint.

Remote operation of robotic systems becomes much more difficult if there is a time lag in the system [4]. This delay could be caused by slow processing or long transmission distances, and would be particularly pronounced if a sampling system were to be deployed for extraterrestrial planetary exploration. For example, communication to and from Mars takes on average 20 minutes depending on its position in orbit [5]. These circumstances render teleoperation virtually impossible.

Another operator burden which could be eliminated are the minute adjustments required to align a collection system (such as a multi-DOF robotic arm) with a sample. If the object and robot's positions relative to each other were known then some sort of automation or

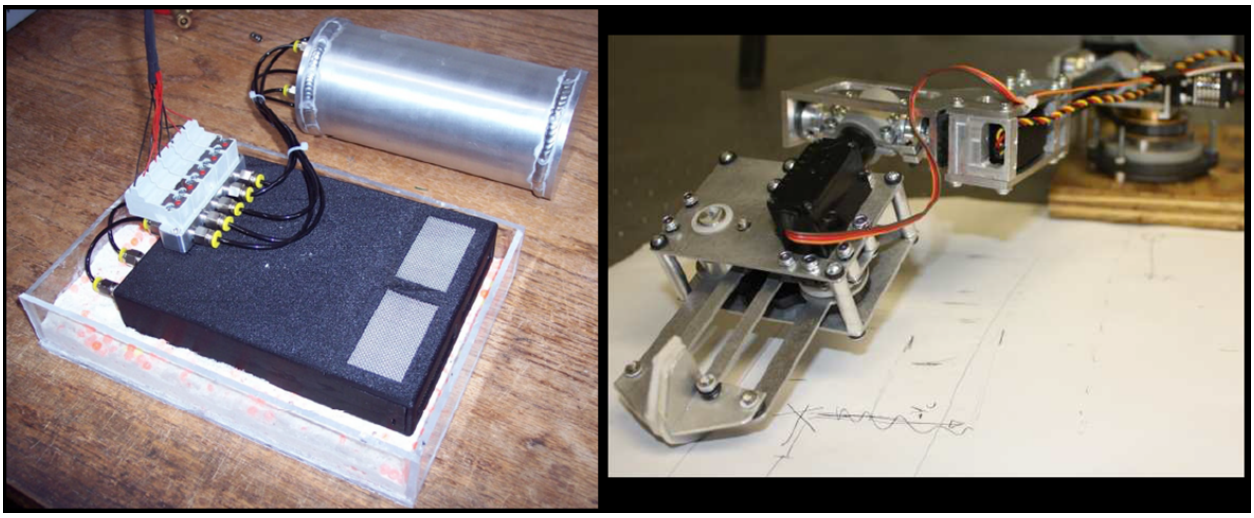


Figure 1.3: Existing collection systems particulate (left), and 3 DOF arm (right)

autonomy could be used to guide the robot to the sample and possibly pick it up without intervention. The method derived from image-based 3D reconstructions presented here is intended to address both issues, as discussed in more detail later. The following introductory sections describe in greater depth the USL's robotic system architecture and the scope of the work discussed in this thesis.

1.2 System Architecture

Two robotic platforms comprise the core of the USL's autonomous robotic response system. The first is the aerial (autonomous helicopter) platform, which deploys the ground robot and provides situational awareness as well as hosting a suite of sensors. The second is the ground robot platform (referred to as "the robot"), which houses a variety of sample collection devices fitted to the interchangeable payload tray. Both systems are controlled by the ground control station, as shown in the figure below. The solid lines in the figure represent communication links between devices. In the robot's case, the helicopter acts as a repeater, receiving commands from the ground station and relaying them to the robot.

However, the robot can also be operated as an independent system. In that case it would be

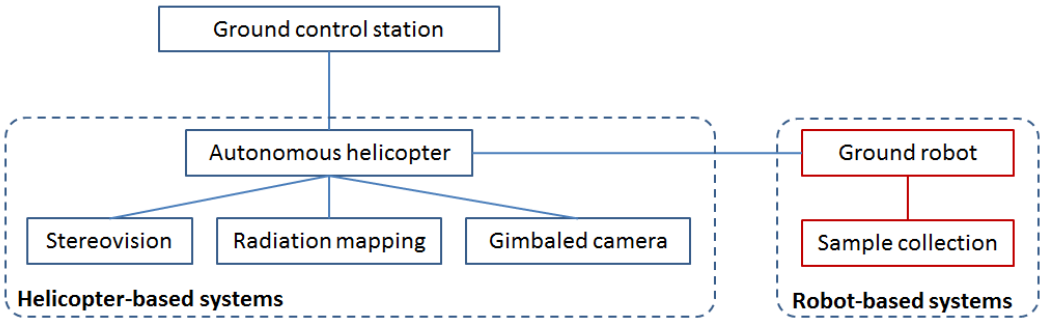


Figure 1.4: Operational platforms for robotic mission

driven directly to the site of interest rather than deployed from beneath the helicopter. This also means that an algorithm developed using the ground robot and its payload(s) could be applied independently or in conjunction with the overall mission; one is not a prerequisite for the other. Practically, the algorithms and collection system developed for autonomous sampling could be used on any robotic platform with the requisite hardware.

1.3 Description of Specific Work

The research presented in this thesis describes three related developments capitalizing upon the capabilities of a ground robotic platform. They are:

1. Development of a scoop sample collection system
2. Creation of an autonomous sample collection algorithm and its use of the scoop collection system from (1)
3. Creation of a crack detection algorithm for assessing building elements post-disaster using the robotic platform from (1) and (2)

All three topics involve the same basic hardware, which is the Unmanned System Laboratory's ground robot. It is used as a host for the scoop collection system, which functions along with a camera mounted to the robot to perform autonomous sample collection. The

same imaging concept used to create the 3D reconstructions for sample collection is applied at a larger scale for the crack detection algorithm.

These topics, in combination, would allow first responders to assess a site, quickly collect samples for analysis, and determine the structural stability of buildings following a disaster. Overall, they address the goal of ensuring the safety of emergency personnel after a hazardous event. However, that is by no means a limitation on their potential usage. The autonomous sampling algorithm is ideally suited for sample collection in planetary exploration, since it is not adversely affected by the time lag involved. The crack detection algorithm could also be very useful outside of disaster scenarios. For example, it could be used to assess the quality of the finish on concrete elements of buildings under construction. Each topic is addressed in its own chapter of the thesis that follows.

Chapter 2

Literature Review

Isaac Newton once wrote, "If I have seen further it is by standing on the shoulders of giants." Likewise, this thesis draws heavily on a large body of prior work, and it would be inconceivable to begin to discuss its content without an introduction to other research on the relevant subjects. Consequently, background information on robotic sample collection, image-based 3D reconstruction, and crack detection which was exceptionally useful as a foundation for this research is presented here. This chapter consists of a literature review and some useful discussion of the basic concepts essential for full comprehension of later chapters. The literature review is divided into three sections, each addressing one of the topics mentioned above.

2.1 Robotic Sample Collection

2.1.1 Sample Collection Devices

Robotic sampling mechanisms have long been used in environments too deep, dangerous, or remote for humans to access directly. Oceanic sea floor and waterway sample collection was an early application of such devices, and an article entitled "A Survey of Marine Bottom

Samplers" from 1964 lists 17 different designs dating from the 1890's to the 1960's [6]. These purely mechanical devices include a variety of scoops and containers and are usually actuated by pulling on a chain or cable from the surface. Designs like the schematics in Figure 2.1 served as an inspiration for the development of the scoop system presented in Chapter 3.

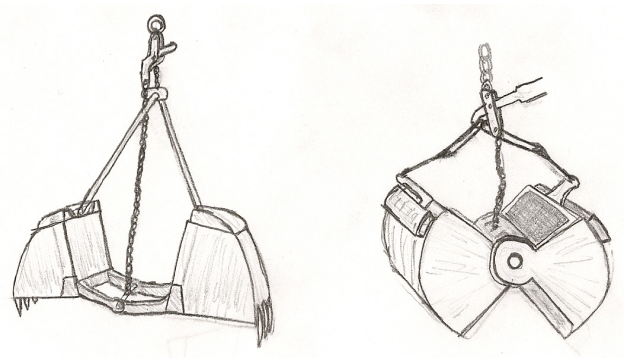


Figure 2.1: Two marine samplers, the Petersen grab (right) and Seki bottom grab (left).

In more recent times, commercially available robots can be equipped with a variety of grippers, grabbers, drills, and other mechanisms. For instance, iRobot's PackBot accessories include several styles of grippers, cable cutter, glass breaker, and a forklift-type attachment [7]. The robotic arm developed by Thomas [3] incorporated a 4 bar parallel motion gripper with two movable plates. In another work, an off-the-shelf robotic platform was given a soil cutter and collection tube to analyze explosives residues at decommissioned firing ranges [8].

Space exploration, as one might imagine, is also an ideal application for robotic sample collection. Planetary surface exploration is a critical component of many missions, particularly to Mars. Spacecraft such as the Mars Phoenix Lander (MPL) land on the Martian surface to perform detailed soil analysis and must gather materials in situ for proper examination. Therefore, it is important to create a reliable mechanism for collecting and containing those materials. Several scooping-based approaches have been used by NASA with MPL [9] and tested on Earth for future missions [10].

2.1.2 Autonomy in Sample Collection

Incorporating autonomy into sample collection operations offers several potential benefits including relieving humans of tedious, repetitive tasks. Additionally, the long time delays incurred by the speed of light limit on signal transmission may render conventional teleoperation impossible during space missions [4]. As a result a variety of approaches have been developed to address those concerns. Several of these existing autonomous sample collection methodologies are discussed in this section.

For the previously mentioned explosives residues collection, human operators used to gather a large number of samples in a grid pattern around a contaminated area. Because this task was relatively straightforward it could be fairly easily automated. Therefore, a robotic platform was equipped with the sample collector and commanded with an open-loop control scheme to drive the grid pattern and pick up soil samples just like a human would. In laboratory tests the robot was reported to perform the task well [8]. A more civilian application for autonomous collection has been under development at Stanford University, where a robotic arm performs simulated grocery store checkouts [11]. The system incorporates a snapshot 3D model of a scene containing common supermarket items and a software algorithm searches planar slices of the model for the best position for grasping with the robot's end effector. A success rate of 91.6% has been reported for grasping objects such as a mug, cup, or bowl automatically.

Autonomy for sample collection has also been used at a larger scale. For instance, a 2003 Japanese sample collection and return mission to an asteroid required autonomous algorithms for guidance and navigation. Since the asteroid's precise shape and surface terrain were not known beforehand they needed to be mapped by the spacecraft once it arrived on site. This was accomplished by acquiring images as the asteroid rotated and using a stereo image processing algorithm to construct a 3D model (the images were transmitted back to Earth for this task). Some additional spatial constraints were added to assist this algorithm in accurately locating 3D points, and for the descent stage a "target marker" was launched to

provide a feature point for navigation. Finally, a small projectile was fired at the asteroid from close range and debris thrown out by the impact was collected for return to Earth [12].

A more recent publication by scientists at the NASA/Caltech Jet Propulsion Laboratory also presented several topics relevant to this thesis. It described the creation and implementation of an automated system to identify high-value science targets from the Mars Exploration Rover (MER) wide field-of-view images. Images acquired at the end of each sol's (day's) drive were processed using filters and edge detection to look for rocks with certain pre-specified characteristics. For instance, the algorithm could find rocks with a given shape (i.e. roundness) and reflectivity and automatically designate them for closer inspection. Once high-value targets were identified they were automatically examined with the MER camera suite and the resulting images were transmitted back to Earth for detailed analysis. This process meant that scientists would receive much more frequent (and relevant) observational data since no time was wasted waiting for operators on Earth to manually specify objects of interest (communication with the rovers occurs only on a limited basis). The authors remarked that the system represented "the first deployment of autonomous rover geology to a planetary rover mission" [13]. These and other approaches provided a background for the development of the autonomous sampling technology explained in later chapters.

2.2 Image-based 3D Reconstruction

Image-based 3D reconstructions are a cornerstone of the autonomous sample collection algorithm since they provide the source 3D colored point clouds for processing and analysis. There are a variety of 3D reconstruction methods, including structured light [14], space carving [15], and Structure-from-Motion [16]. Structure-from-motion (SfM) techniques are a good candidate for robotic applications because they require only a single camera and movement to create a reconstruction. Other methods may require a specific setup, more equipment, or be less robust in real-world applications. Therefore, SfM offers a cheap, low

power requirement, and simple alternative to other 3D reconstruction methods (including non image-based techniques such as laser scanners). One significant drawback to SfM is the scaling ambiguity inherent in images (i.e. without knowing the absolute positions of at least 2 points in space, the 3D point positions can only be determined relative to each other). This complication and a solution to it are discussed in greater detail in Chapter 4. Fundamentally, SfM techniques work by first identifying "feature points" that are common across images and then performing calculations to determine how the camera was moved relative to these points in 3D space. This concept is illustrated by the graphic below.

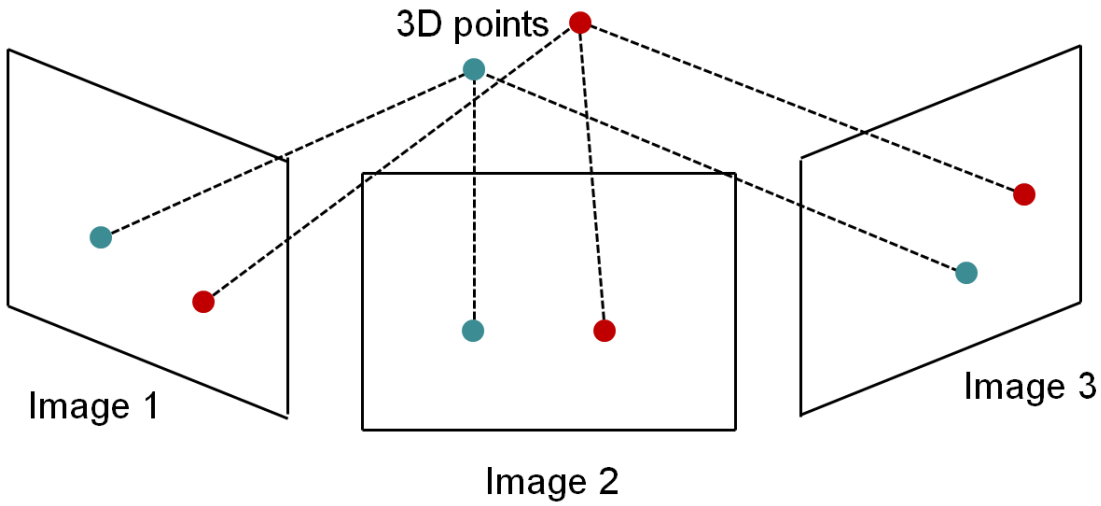


Figure 2.2: Identification and projection of feature points for Structure-from-Motion.

In this figure the same two feature points (red and blue) can be seen across three different images. The images could be from multiple cameras, or also from the same camera shifted to three different points in space (to the reconstruction both scenarios are equivalent). Computationally, the rotation and translation of the camera between each image is determined and used to find the location of the points in space, projected into 3D from the 2D image sequence.

2.2.1 Feature Points and Descriptors

In order to identify points or visible similarities across multiple images there must be a method of first locating and then describing a "feature". Over the years several algorithms have been developed to do this, including KLT [17], Harris corner detection [18], and SIFT. These image processing techniques operate on greyscale images and use pixel intensities for computations. The structure-from-motion package used in this thesis is designed to work with SIFT (the Scale Invariant Feature Transform) developed by Lowe [19]. As the name implies, it is specifically formulated to be invariant to changes in scale, but it is (relatively) invariant to changes in orientation as well. These two properties mean that common points can be identified even if a camera rotates and translates closer to or farther from an object.

The first step in SIFT (and what provides its scale invariance) is to create Difference of Gaussian (DoG) representations of the initial image at its original size at down-sampled sizes (scales) as well. Next every pixel in each DoG image is compared to its neighboring pixels at the same scale and the scales directly above and below; this identifies local minima and maxima as keypoint candidates. Other procedures are performed to discard points with low contrast or along edges (which will produce a strong response to certain filters).

The second main step in SIFT is to generate keypoint descriptors for each of the previously identified feature points. This is done by computing the image gradients around the feature point as seen in Figure 2.3.

The descriptor is composed of a 4x4 grid (like the 2x2 one shown on the right in the figure) with 8 orientation "bins" (see the arrows) for each section whose gradients' magnitudes stored in a histogram descriptor (128 elements total). Afterwards, the histograms are normalized to produce invariance to changes in illumination. Once a series of SIFT keypoint descriptors are generated they can be compared to determine if two keypoints identified in different images are a match.

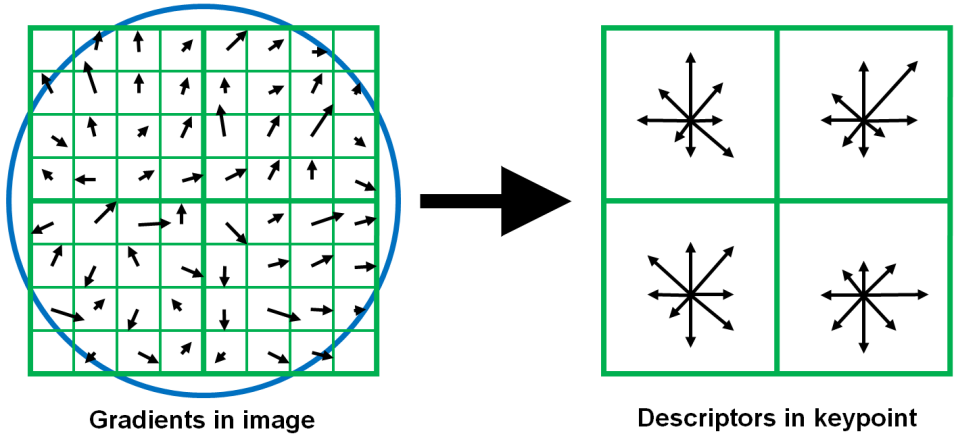


Figure 2.3: The SIFT keypoint descriptor.

2.2.2 Structure from Motion

Structure-from-Motion (SfM) reconstructions are created from a series of images in which the camera position has been rotated and translated through space. They produce "sparse" reconstructions that are solely composed of 2D feature points projected into 3D space (dense reconstructions, also used for this thesis, are discussed in the next section). In particular, the open source SfM package Bundler [16] was used, but the principles of SfM discussed here are largely generic.

Figure 2.4: Generalized structure-from-motion pipeline.

The SfM pipeline begins by taking a series of images (from a few to hundreds) and running some type of feature detection/keypoint identification on them. SIFT is excellent for this purpose since it accommodates changes in scale, illumination, pose, etc. It generates a set of keypoints with descriptors and their locations for each image. The next step is matching keypoints between each pair of images (descriptors are compared using approximate nearest neighbors) [16]. Then the matched keypoints between each image pair are used to estimate the fundamental matrix (which relates corresponding points between stereo images) for it inside of a RANSAC (RANdom SAmple Consensus) loop. Afterwards matched keypoints which are outliers to the calculated fundamental matrix are discarded. Finally, sets of matches are combined into tracks, which are composed of keypoints matched across multiple images.

Once a series of tracks has been created the SfM algorithm finds camera parameters and 3D positions for each track. Then the sum of the reprojection errors (from 3D back to the 2D images) is minimized.

Cameras are then added incrementally to the reconstruction. The initial image pair is chosen by finding the set of images which are best modeled by the fundamental matrix (F) versus a homography (H). Therefore, the starting pair will have a large baseline (better for structure *from motion*), which means that F was good fit, instead of a small baseline, which means that H was a good fit. Each new camera's extrinsic (rotation and translation) and intrinsic (focal length, distortion) properties are estimated using the direct linear transform inside of another RANSAC loop. A focal length estimate read from EXIF (JPEG header) tags is also incorporated into this procedure.

For more details of the process the reader is referred to the works cited in [16]. Essentially, as each camera is added a sparse bundle adjustment is performed on the overall scene to minimize an overall error function. The final output from this structure-from-motion pipeline is a colored 3D point cloud containing feature points and camera locations and view directions. One last note: Snavely, et. al. also developed a procedure for geo-registering (and scaling)

their reconstructions; however, it is a manual process which requires that users specifically identify matching points. One of the goals of this thesis will be to automate that portion of the 3D reconstruction.

2.2.3 Dense Reconstruction

The dense reconstruction method used in Patch Multi-View Stereo (PMVS) starts with the Bundler SfM outputs to help initialize its reconstruction. It is described by the authors as using "a simple *match, expand, and filter procedure*" [20]. Those steps are performed in order: first, finding and matching features; second, expanding matches to create dense patches; third, filtering using visibility constraints to get rid of bad matches.

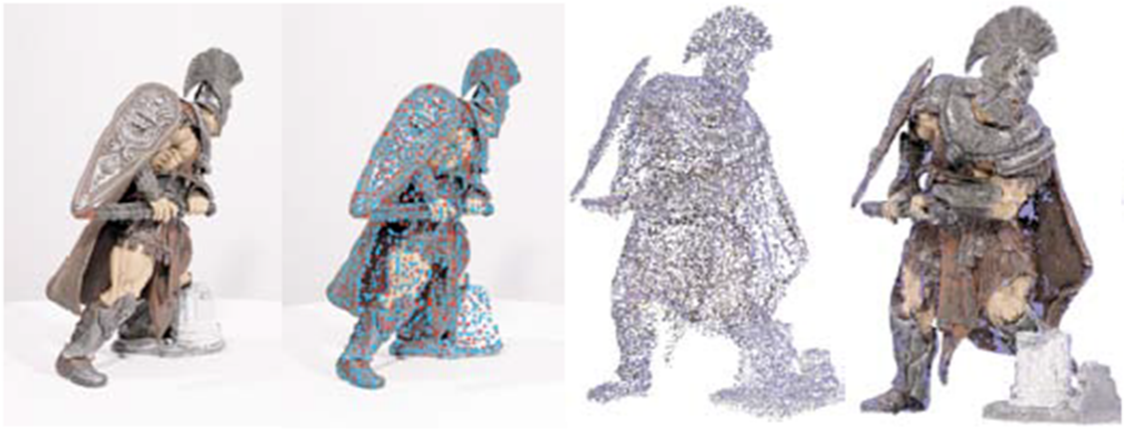


Figure 2.5: Progression of PMVS (from left to right): image, features, patches, expansion [20]. © 2007 IEEE. Used with permission.

The first step (matching) relies upon detecting corner and blob features in each image. This is done by dividing the image into coarse "cells" and identifying the strongest responses to the feature detectors in each one. These features are matched between images by looking for the same type of feature near the epipolar lines in other images. Subsequently, these candidate patches are reconstructed into a sparse 3D model (a patch must be visible in a minimum number of images to qualify).

Once an initial reconstruction is created, patches are expanded by adding neighbors until all

visible surfaces are covered. Since some matches from the previous two steps may be invalid, the third step of filtering is used to enforce visibility constraints. For example, if a patch is reconstructed "inside" of an outer surface of other patches, it will be removed. The PMVS algorithm iterates between the expansion and filtering steps three times before finishing.

The final product of PMVS is a dense colored 3D point cloud. The authors, Furukawa and Ponce describe applying a surface reconstruction to this point cloud; however, that was not necessary for the work presented here, so the PMVS output employed in the following chapters is a collection of a large number of discrete 3D points representing a realistic reconstruction of a scene.

2.2.4 Practical Considerations

The output all of these image-based 3D reconstruction algorithms is almost entirely a function of the quality of the input images. More feature points will be detected in higher-resolution, higher quality images. Likewise, having more pixels to work with will create a more dense point cloud from multi-view stereo techniques too. Constraints are also imposed based upon the feature detector used. For example, SIFT has a maximum effective angle between photos of 12-15°. Shifts through angles larger than this between images can cause feature points not to be properly matched and reconstructions to fail. Therefore, using SIFT to reconstruct a complete 360° around an object requires a minimum number of photos be taken. Also, image-based reconstructions have difficulty with non-Lambertian surfaces (i.e. those that don't reflect light in a predictable way). Because these algorithms continue to evolve some of these limitations may be overcome, but for now they are still significant considerations when applying the techniques.

2.3 Crack Detection

The point cloud generated from image-based 3D reconstructions has many potential applications beyond the main focus of this work (in autonomous sample collection). One excellent application of the overall system draws on the ability of a robot to enter dangerous environments and perform the automated image acquisition routine. Since the 3D point cloud model is not scale-dependent, entire buildings or building elements can be easily reconstructed using the same software pipeline. With proper processing this model can be used to automatically detect and analyze cracks in building elements.

There are a variety of different crack detection techniques that already exist for various materials. However, almost all of the current techniques do not take advantage of any 3D data. They rely instead on processing individual 2D photos and looking for tell-tale features of surface cracks. Examples of these efforts include 2D crack detection for asphalt roads [21][22], or building surfaces [23][24]. A typical approach, analyzed in [25], is template matching. This involves passing a template image (much smaller) over the detection image at various angles and subtracting the two. If that result at a given position is a small enough value the target feature from the template (i.e. a crack) has been located. This process is illustrated in Figure 2.6.

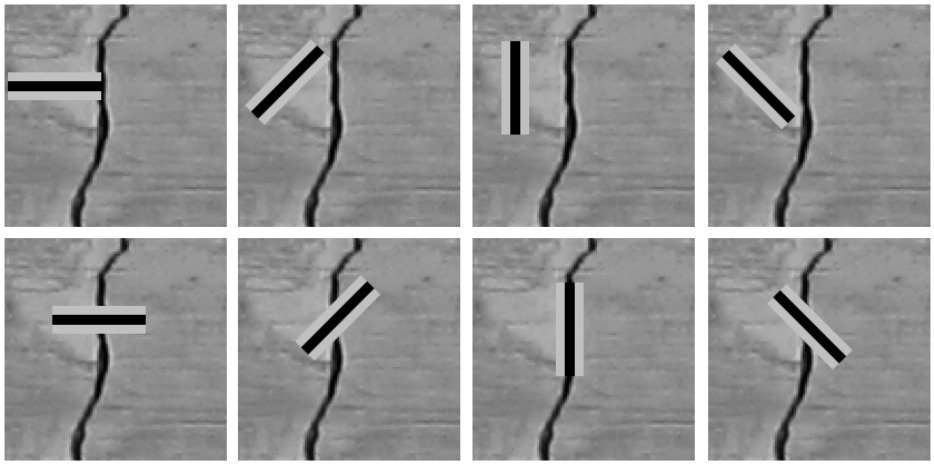


Figure 2.6: An example of template matching for crack detection.

One more recent work starts toward incorporating SfM data into crack detection [26]. In it the estimated camera properties from the SfM algorithm are used to assist in equating the 2D image pixels with real-world distances in millimeters. While this approach currently still requires some manual measurements, it may eventually be able to automatically provide scaled length and width measurements for cracks (something which has already been accomplished in this thesis anyway). Despite their benefits, all of these essentially 2D approaches do not provide 3D depth information and as a result do not enable measurement of severity of the cracks.

Chapter 3

Scoop Sample Collection System

This chapter describes the design, construction, and testing of a sample collection device for the helicopter-deployed ground robot. The goal of the device is to be able to collect and contain a single object weighing up to 250 grams with a maximum volume of 600 cm^3. A dual scoop design is presented to meet all weight and size constraints, as well as demonstrating the ability to collect and contain a sample utilizing the same components.

3.1 Requirements and Criteria

3.1.1 Design Requirements

The fundamental constraint upon the sampling system design was the ability to pick up large "chunk" samples (versus small particulate samples) weighing up to 250 grams with a maximum volume of approx. 600 cm^3 (equivalent to that of a soda can). Several other key constraints were imposed due to the nature of the mission and the ground robot platform. First, the overall mission architecture requires that the robot be lowered from the Yamaha RMAX helicopter (Figure 1.1), which limits the maximum weight of the system to less than 6.8 kg. Second, the ground robot platform (Figure 1.2) is designed with a payload tray of

26.5 cm x 26.5 cm, so any sample collection device must have a footprint no greater than this area.

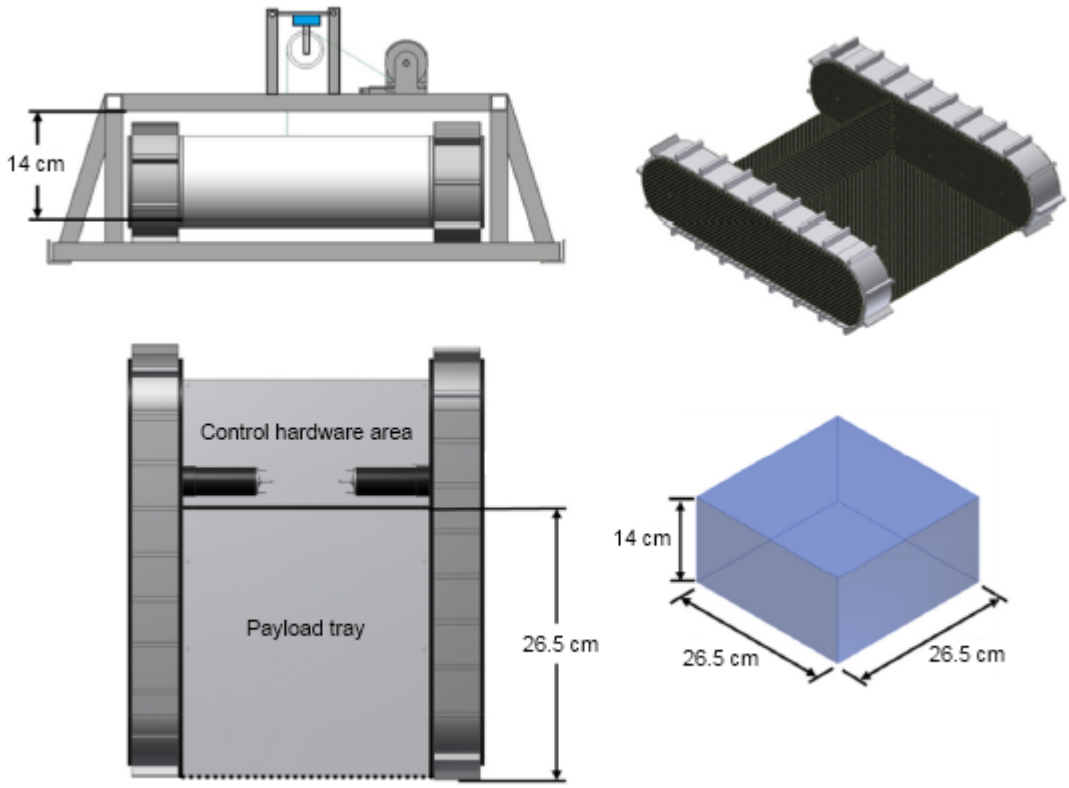


Figure 3.1: Dimensional constraints on sample collection system. Adapted from [2]

Additionally, the height of any device is also limited by two factors: the maximum height of the robot in the robot deployment pod and the field of view of the drive camera. Functionally, this meant the height available was approximately 13 cm. The physical constraints on a chunk sample collection system are summarized in Table 3.1 below.

Several other factors are also important considerations for the sampling system. For instance, safely transporting the material after collection is a significant concern. Since the overall system is designed to be deployed into hazardous environments and could be acquiring radioactive material, the sample(s) must be completely contained. This would limit exposure

Table 3.1: Constraints on sample collection system design

Height	13 cm
Width	26.5 cm
Depth	26.5 cm
Overall weight	6.8 kg
Robot weight	5.5 kg
Collection system weight	1.3 kg

of personnel to dangerous material, and also prevent the material from being contaminated by the environment.

Operator skill and mechanism complexity were also considerations. More complex collection systems would require either a highly skilled operator or a complicated control system. Simpler systems would require less input and control but might be more limited in application. This trade-off was one of the central criteria examined during design selection.

3.1.2 Selection Criteria

Initial designs focused on techniques that would sweep a wide enough area to collect the object of interest with minimal operator input. This led to several scraping-based approaches, in which a sample is lifted or dragged into an integrated container (Figure 3.2).

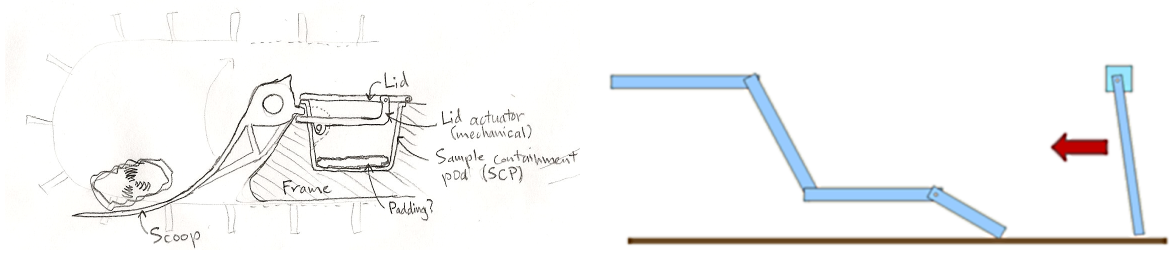


Figure 3.2: Scraping-based sampling approaches (left - scoop, and right - drag scraper)

The scraping-based approaches turned the entire front of the robot into a giant sample collection device, providing a wide collection area. They also relied on the robot itself to provide three degrees of freedom (DOF), i.e. x and y translation and rotation about a vertical

axis. These designs were essentially 1 DOF mechanisms which dragged the sample into a container which was opened and closed automatically by the mechanism.

The second class of non-traditional collection mechanisms that was examined were scooping-based. In these, a scoop would collect a sample and contain it after picking it up. As previously mentioned in the literature review, this is commonly done for oceanic and waterway sampling [6]. The scooping designs are shown below in Figure 3.3.

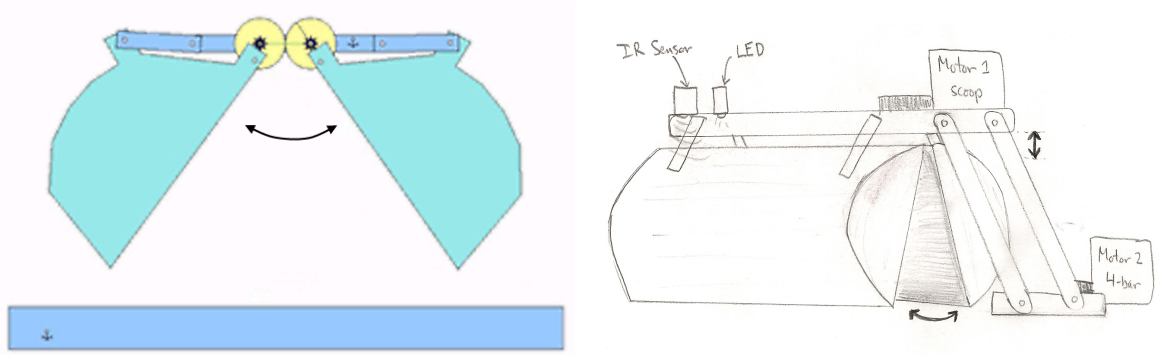


Figure 3.3: Scooping-based sampling approaches (left - 1 DOF, and right - 2 DOF scoops)

The 1 DOF scoop operates by attaching the inner scoop pivots to two interlocking gears and the outer pivots to linkages. When actuated, the scoops make full contact with the ground plane at approx. $^2/_3$rds closed position, and then lift up slightly when fully closed. This configuration would raise the scoops slightly, allowing the robot to drive around after collecting a sample. The 2 DOF scoop operates in a similar manner, except that it separates the vertical translation and closing motions. The 2 DOF scoop would incorporate two independent actuators, one to control the height of the scoops, and one to open and close them.

Given the variety of approaches, a decision matrix was created to summarize the strengths and weaknesses of each. For comparison purposes, a traditional robotic arm is also listed. The decision matrix appears in Table 3.2 below.

In the decision matrix, each column presents a different sample collection method and each row lists a different performance factor. All performance factors (Range of motion, Area

Table 3.2: Decision matrix for comparison of collection systems

	Robot Arm	Scooping Scraper	Dragging Scraper	1 DOF Scoop	2 DOF Scoop
Range of motion	5	2	1	3	4
Area covered	1	5	5	2	3
Complexity of control	1	5	5	4	4
Terrain flexibility	5	2	1	2	4
Weight	1	4	4	5	5
Repeatability	5	5	1	5	5
Totals	18	23	17	21	25

covered, etc.) are rated on a scale of 1 to 5, with 1 being worst and 5 being best. With the exception of the robotic arm (which already existed), scores were assigned based upon the anticipated performance of each technique. For instance, the dragging scraper concept would only operate at one height (giving it the lowest possible terrain flexibility with a score of 1), but would merely need a simple "start" command to begin operation (therefore, extremely easy to use, score of 5). Primary comparison was made by summing each method's scores for every factor, and the collection methods with the highest total scores do the best overall job performing sample collection.

From Table 3.2, it is apparent that there is no perfect solution to the chunk sample collection problem because any system will have some limitations. Ultimately, the 2 DOF scoop was chosen for development, because it offered a good trade-off between complexity, weight, and functionality. It could accommodate varied terrain (to some degree) because the scooping height was adjustable. It also allowed for multiple scooping attempts in the same location since the scooping motion was independent from the rest of the actuation. The 2 DOF scoop was lightweight, because it could enclose a large area with minimal containment material. Additional weight savings were gained by its dual functionality: having the scoop be both the sample collection device and container eliminated the need to create separate subsystems for these tasks.

3.2 Scoop Sampling System Design

3.2.1 Proof of Concept

The first step in the examination of the scoop design was to validate its use on a smaller scale. To accomplish this a basic mini scoop was developed in CAD (see detail drawing in the Appendix) with dimensions of approx. 6.4 cm x 6.4 cm x 3.8 cm (Figure 3.4). Two mini scoops (each one being one side of a full scoop) were fabricated out of ABS plastic using rapid prototyping.

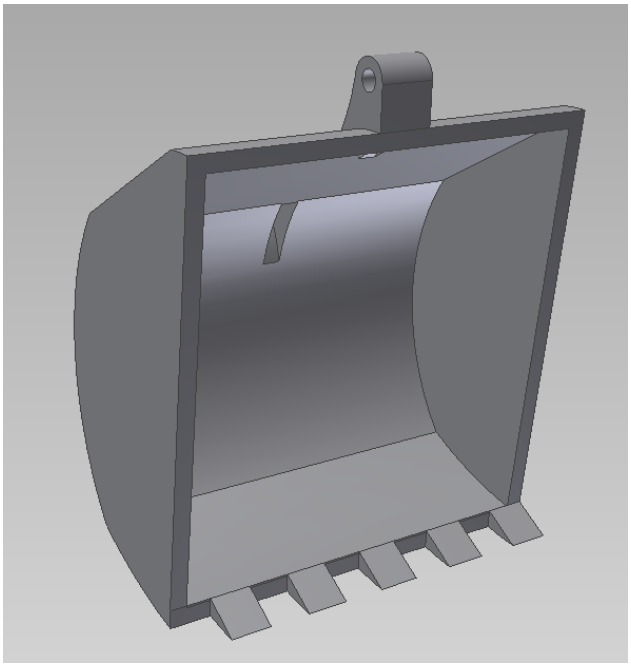


Figure 3.4: Mini scoop CAD rendering

The mini scoops were attached to a commercially available robotic arm (Figure 3.5). The arm was controlled through a game-controller interface attached to a computer for a series of preliminary tests. These basic tests consisted of picking up a rock from a small pile on a flat surface and depositing it into a nearby container.

The mini scoops were quite adept at acquiring and manipulating small rock samples. The rounded shape of the scoops assisted in enclosing the target objects even if the arm was

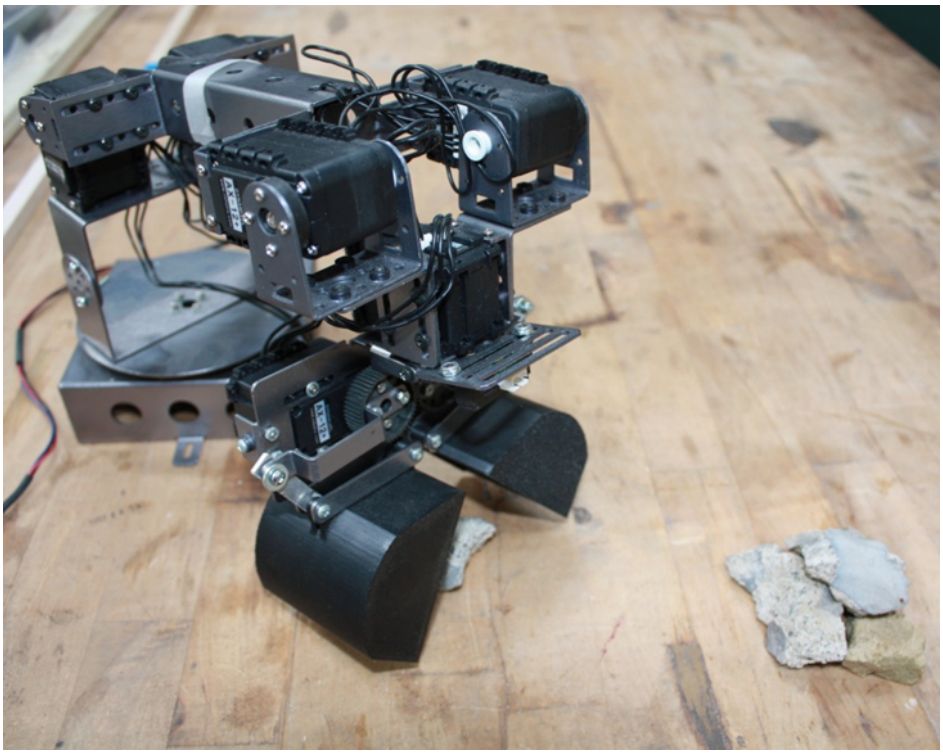


Figure 3.5: Mini scoops on robotic arm

not precisely aligned with the samples. Because of this success, development of a full-scale scooping system for the ground robot was undertaken.

3.2.2 Full Scale System

The full-scale scoop collection system is composed of two large scoops, linkages to control their vertical motion, servo motors for actuation, and associated other hardware (e.g. the mounting tray). It is shown in Figure 3.6 below.

System Architecture

The vertical degree of freedom of the scoop sampling system is principally composed of a four-bar mechanism which enables vertical movement and holds the scoops. One of the two servo motors is directly attached to the vertical links that form the mechanism. When it moves, both scoops are either raised or lowered together (with respect to the mounting tray).

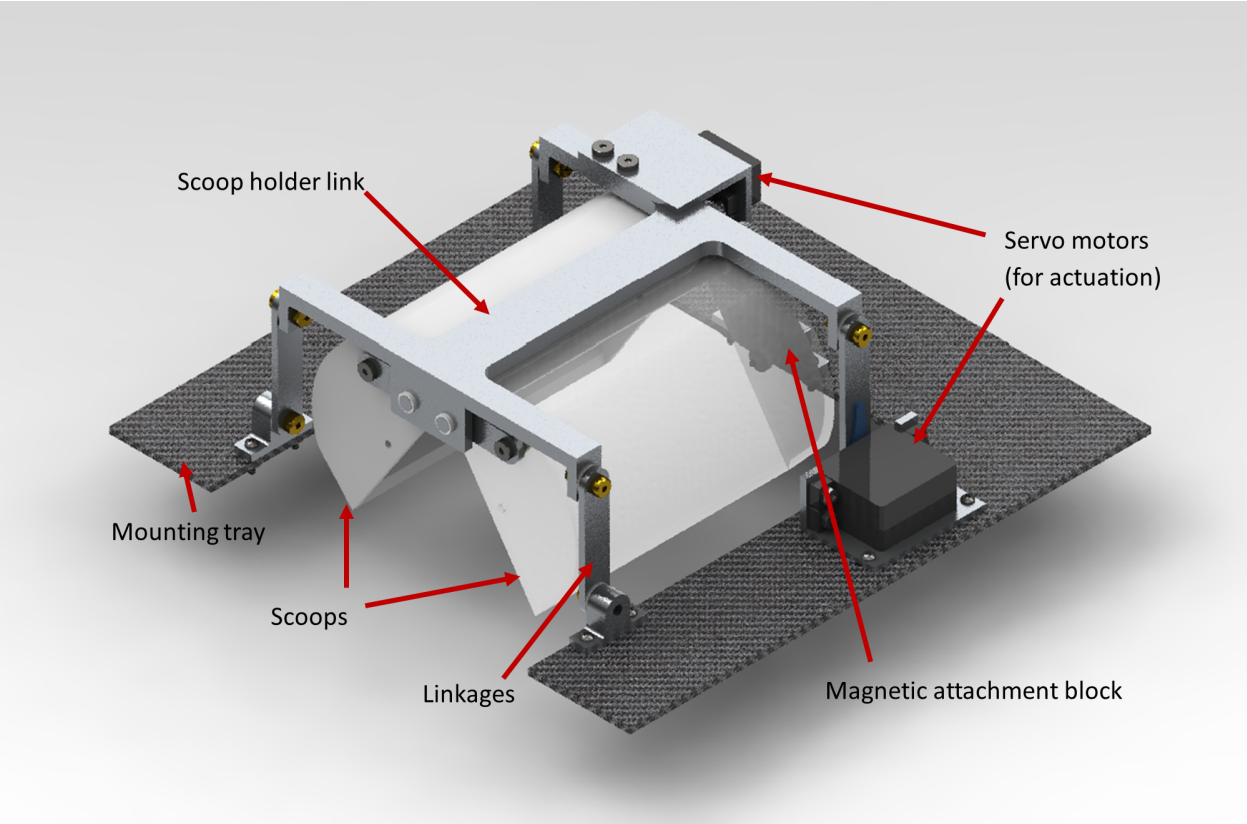


Figure 3.6: CAD rendering of full-scale scoop system with principal components highlighted

This provides two advantages: first, the scoops can accommodate varying ground heights, and second, the scoops can be retracted while driving and during transport. The four-bar linkage can be seen in Figure 3.7.

Several additional advantages are also gained by implementing the four-bar linkage. For instance, one of the defining characteristics of a parallelogram four bar mechanism is that the sides remain parallel to each other throughout its range of motion. Therefore, the scoops would always be oriented parallel to the ground, as desired. The linkage mechanism also distributes the weight of the scoops and anything they collect across the four mounting points. The vertical links were 5 cm in length, providing the scoops with up to 10 cm of vertical motion (practically, their movement is limited by the range of motion of the link servo). In practice, the most common scooping height is at the robot's ground plane, approx. 2.5 cm below the mounting tray.

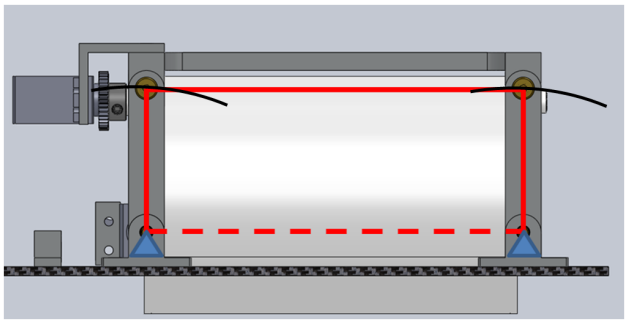


Figure 3.7: Four-bar linkage in scoop system. The red lines represent links (dotted line is virtual ground link) and the blue triangles represent the fixed pin joints around which the vertical links rotate. Black arcs represent the motion of the scoop holder link.

Scoops

The full-scale scoops have dimensions of approx. 13 cm x 5 cm x 9 cm. Combined, when shut, the two scoops can enclose a volume of 655 cm^3 (a cylinder of radius 4 cm and length 12 cm in fits inside). These dimensions were chosen based upon the maximum expected size of the samples.

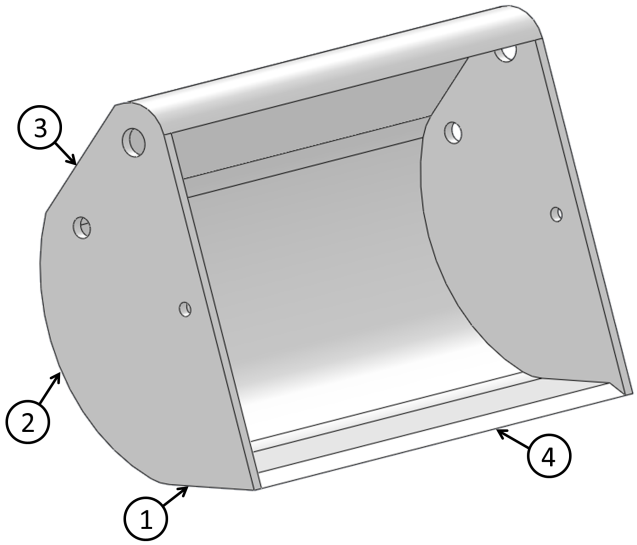


Figure 3.8: Full-scale scoop. Holes in scoop are for attachments explained in *Details*.

A scoop profile similar to that of the mini scoops is maintained, since that had already proven to be effective for sample collection. Essentially, the bottom of the scoop is slightly angled (1) so that it could easily reach underneath ground-level objects, which leads into

a rounded section (2) to pick up and contain the sample(s), and the top is sized to fit the torque transmission link (3, see Details section). A small bevel (4) is added to the leading edge to enable easier grabbing as well.

Actuation and Design Details

The opening and closing of the scoops is actuated by a mini servo motor mounted horizontally behind them (see Figure 3.9). This servo has a gear attached to its spline, which turns two gears directly attached to shafts running through the scoops. The two scoop gears (36 teeth, 48 DP) are chosen to be identical, which produces an inverted 1:1 ratio of movement for opening and closing. The gear mounted to the servo spline (18 teeth, 48 DP) has half the diameter of the scoop gears, which doubles the effective torque of the servo. The shafts are keyed so that they can transmit torque to two torque transmission links on the opposite side of the scoops. These links, in turn, are rigidly connected to the scoops with a screw, allowing them to control the scoops' position.

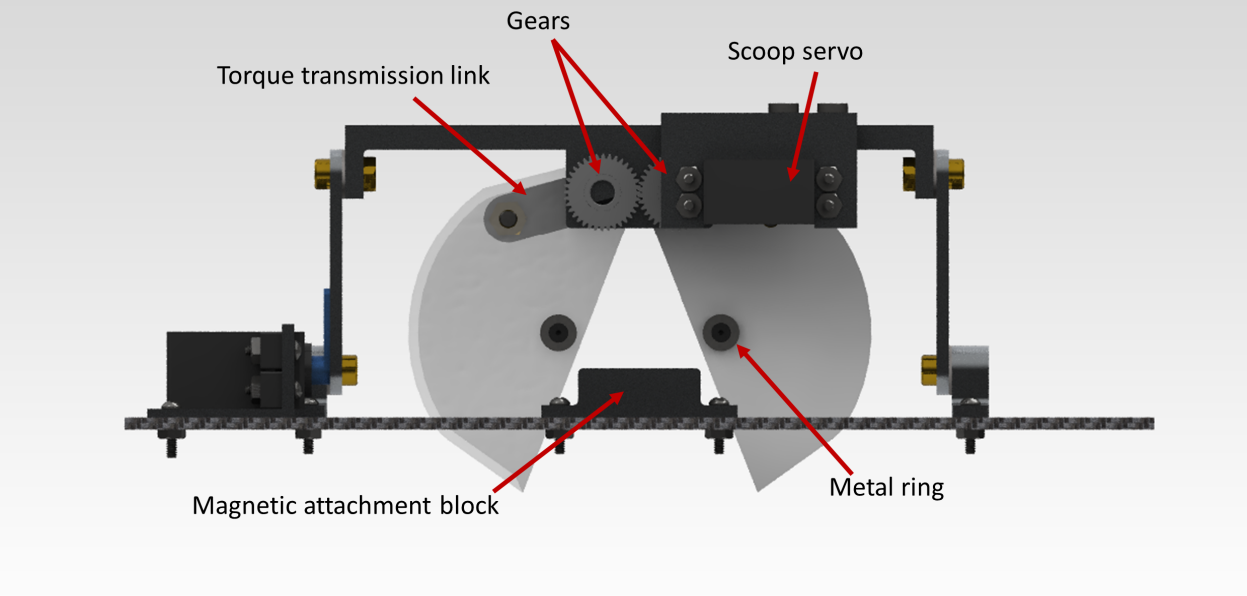


Figure 3.9: Rear detail view of scooping system.

Many other factors were considered when designing the scoop sampling system. One concern was keeping the sample contained during flight. Once a sample was acquired it would be

enclosed by the scoops, but to keep the scoops shut the servo motors would have to be continuously powered. Additionally, oscillations or other flight disturbances needed to be minimized so that a potentially harmful sample would not be unintentionally released. Thus, a magnetic latching system was created to address both of these issues. It operates as follows:

A small metal ring is attached to the back of each scoop in a position so that when the scoops close, both rings are close together (the ring is attached at the small hole seen in the center of Figure 3.8). Opposite the metal rings on the mounting tray is the "magnetic attachment block" seen in Figure 3.9. It contains two small, powerful magnets which attract the metal rings on the scoops. When the scoops are moved all the way back to the "stowed" position (which happened prior to robot deployment and after sample collection) the magnets engage, securely fastening them. The high-torque servo motor actuating the links produces enough force to separate the two, but otherwise the scoops remain stationary. This magnetic system means that the scoops will be rigidly attached during flight, so the servo motors can be powered off, conserving battery life.

The scoop components were created using a variety of manufacturing techniques. Rapid prototyping created the scoops themselves and the scoop holder link. A waterjet was used to cut out the four vertical links. Aluminum 90° extrusions were machined on the CNC mill to create the servo brackets, as was bulk aluminum for the lower link attachment points. Finally, the mounting tray was formed from a wet lay of carbon fiber composite sheets over a balsa wood core. A picture of the completed scoop sample collection system appears below.

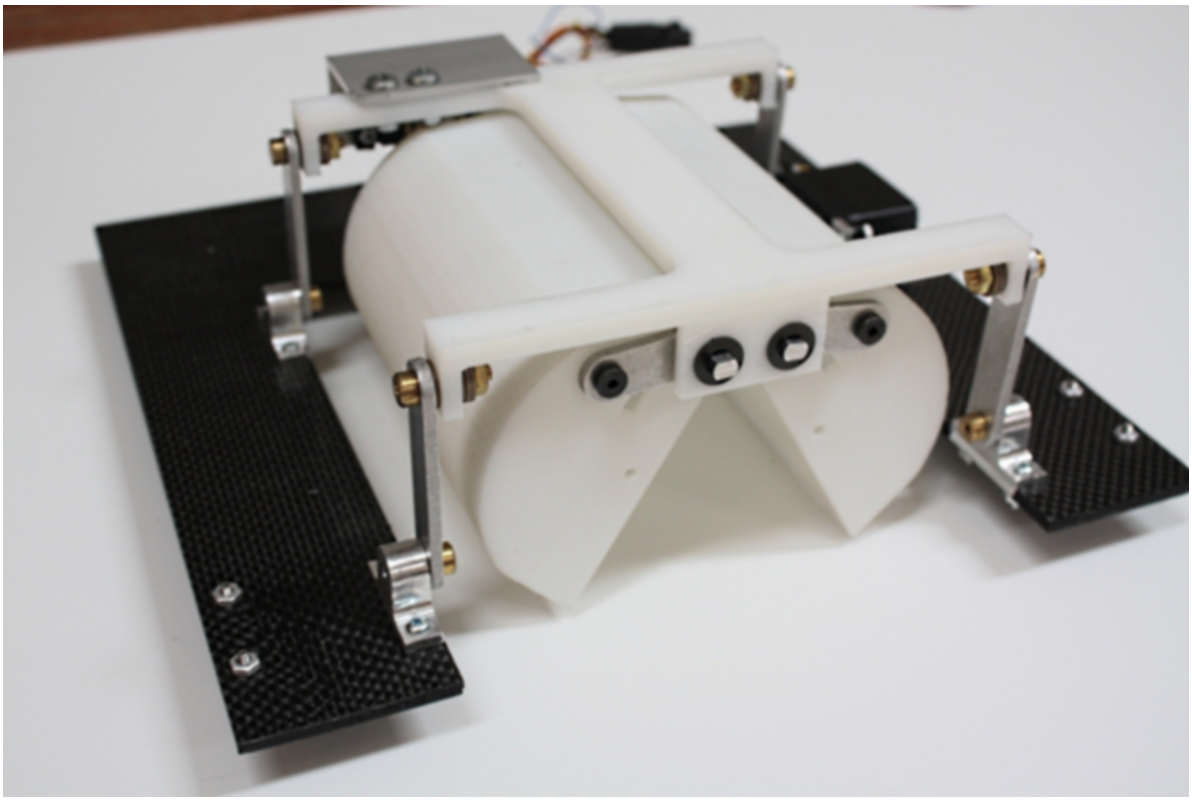


Figure 3.10: Assembled scoop system mounted on interchangeable tray

3.3 Testing and Robot Integration

3.3.1 Auto-scooping algorithm

Prior to attaching the scoop assembly to the robot, a series of bench tests were performed to validate its performance and create an auto-scooping algorithm. An Arduino microcontroller, 2 axis joystick and two pushbuttons were assembled on a breadboard to interact with the servo motors. The joystick provided manual adjustment of the scoop position (one axis controlling vertical position and the other axis open/close), one button opened the scoop (return to "home" position), and the other performed the auto-scoop algorithm.

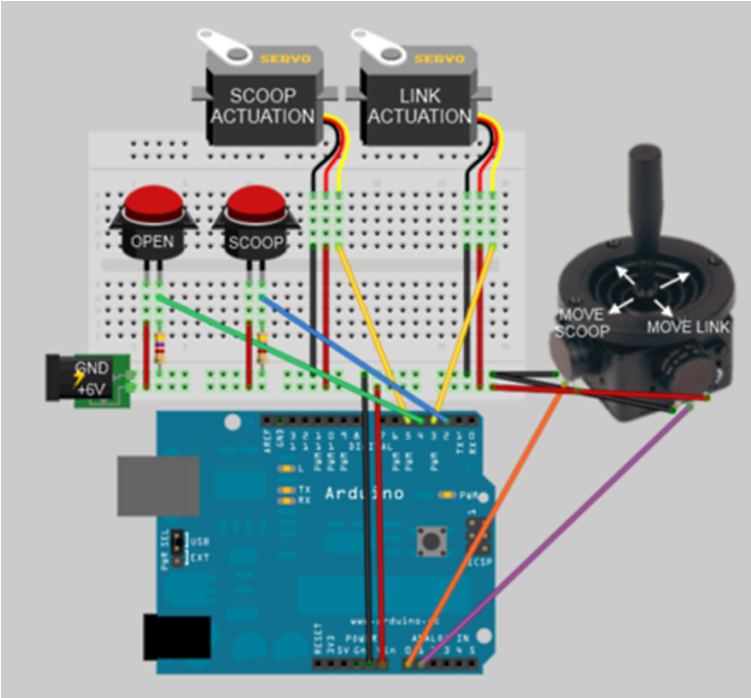


Figure 3.11: Circuit layout for bench tests and determination of auto-scooping algorithm.

The purpose of an auto-scooping algorithm was to speed the process of sample collection, so that an operator would not have to manually perform the scoop movement each time. It provided a reliable and repeatable motion which could be easily engaged with the push of a button. The auto-scoop algorithm was established by doing a manual scoop with the joystick controls and recording the servo PWM signals during the procedure. An example of the auto-scooping algorithm (which operates assuming a predetermined standard ground height) is written in detail below.

Nomenclature for algorithm:

1. $srvposV$ is the PWM signal (in μs) to the vertical (link height adjustment) servo
2. $srvposS$ is the PWM signal (in μs) to the scoop (open/close) servo

Algorithm 3.1 Automatic scooping

1: Reset to Home position (srvposS = 1240 (open), srvposV = 1500 (up))
2: **while** $srvposV \leq 1920$ **do**
3: $srvposV = srvposV + 20$ {lower}
4: **end while**
5: **while** $srvposS \leq 2000$ **do**
6: $srvposS = srvposS + 20$ {close}
7: $srvposV = srvposV - 1$ {raise slightly}
8: **end while**
9: **while** $srvposV \geq 1500$ **do**
10: $srvposV = srvposV - 10$ {raise}
11: **end while**

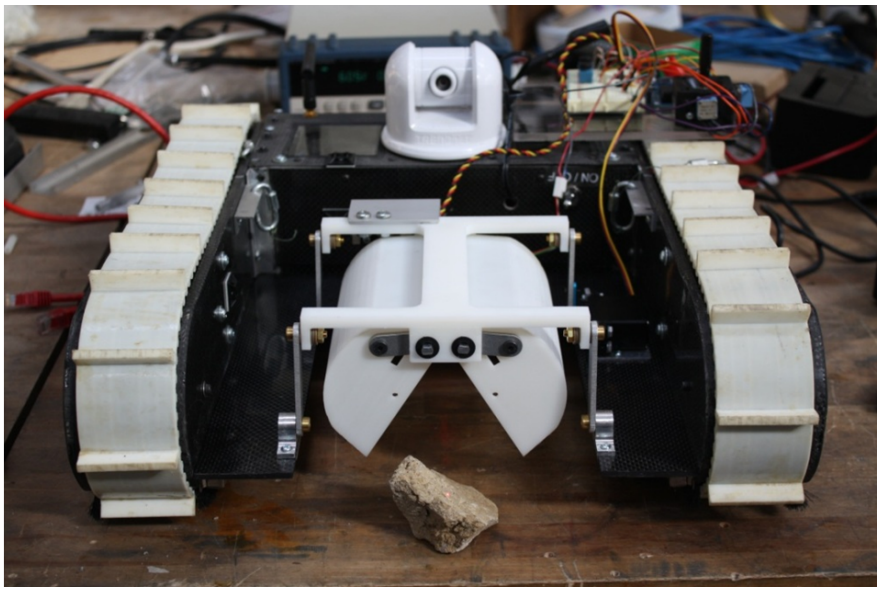


Figure 3.12: Scoop mounted on robot for bench tests (manual operator controls can be seen sitting on the rear of the robot next to the drive camera).

3.3.2 Additional Sensors

After the initial bench testing and auto-scooping algorithm development were completed a suite of additional sensors was added to the scoop to facilitate operator situational awareness. In order to locate these additions, the magnetic mounting block was removed and replaced with a video camera, laser indicator, and IR distance sensor. The video camera provides

the operator a secondary view of the inside of the scoop, which greatly eases locating an object to collect. Rather than relying solely upon the view from the drive camera (which is obscured once the object is very close to the front of the robot), the operator can toggle cameras and directly see the sample as it was enclosed by the scoop. Likewise, the IR sensor indicates the distance to the sample, providing a simple "YES/NO" feedback to show if it was in the correct area to scoop. Additionally, the laser indicator shines a beam of light out forward from the robot, which is used to mark a sample and can be viewed in both the drive and scoop cameras.

3.3.3 Outdoor Tests

Once the additional sensors were added, the ground robot with the scoop system was run through a series of outdoor tests. These tests were designed to establish the effectiveness of the scoop on different terrain types. Two rocks were used as the stereotypical samples to collect; each rock weighed approx. 85g and had a diameter of 4-5 cm.

The scoop was tested on concrete, asphalt, grass, dirt, gravel, and sand. In each case, the robot was teleoperated and maneuvered until the scoop was over the target sample, and then auto-scooping was activated. Afterwards, the robot was driven a short distance to assess sample containment prior to opening the scoop and releasing the sample. From these tests it was determined that the scoop was able to acquire the two rocks individually on most terrain types. Overall results are summarized in the table below and rated from 1 (worst performance) to 5 (best performance). A score of 5 indicates that the operation succeeded in virtually all cases, and a score of 1 indicates that it always failed. Intermediate scores are indicators of varying degrees of success.

Table 3.3: Scoop performance on various terrain types

Terrain Type	Concrete	Asphalt	Grass	Dirt	Gravel	Sand
Ability to Collect Sample	5	5	1	4	3	5
Ability to Contain Sample	5	5	N/A	5	5	5

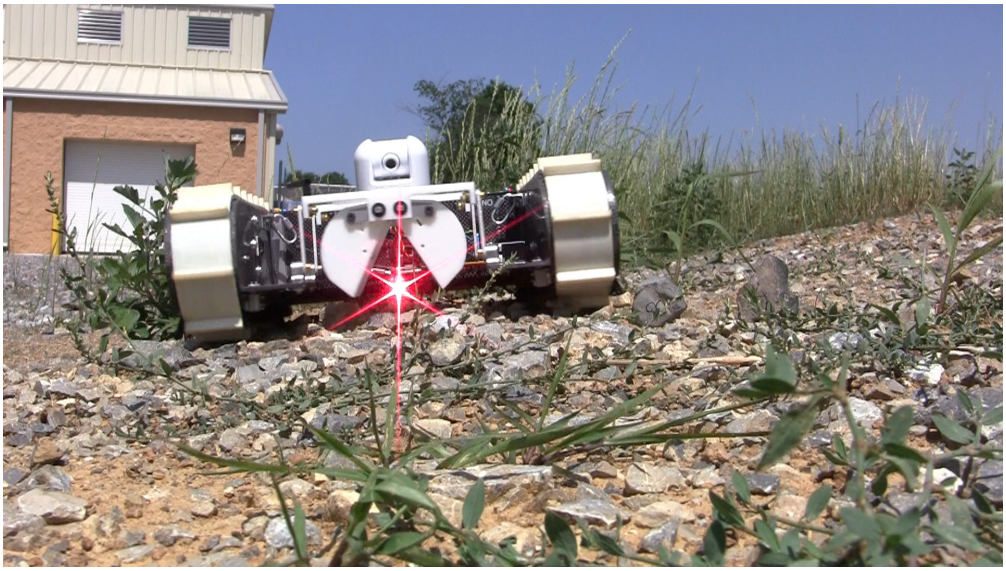


Figure 3.13: Outdoor scoop tests with the robot on rough terrain. A red light is visible from the laser indicator.

3.4 Results

The scoop sample collection system was able to meet all of the physical design requirements as specified in Table 3.1, and the final parameter values are listed in Table 3.4. Please note that the collection system weight shown in the table does not include an independent power system because power is provided by the robot itself.

Furthermore, as can be seen in Table 3.3, the scoop performed well on four out of the six terrain types, and moderately well on a fifth. Any surface on which the scoop could make full contact with the ground and close around the sample worked well. However, if there were fragments of rock or blades of grass which interfered with the scoop closing completely, then

Table 3.4: Comparison of constraints versus final scoop design.

Parameter	**Constraint**	**Actual Value**
Height	13 cm	9.7 cm
Width	26.5 cm	26.5 cm
Depth	26.5 cm	26.5 cm
Collection System Weight	1.3 kg	0.57 kg

the sample could not be properly collected in most cases. Thus, performance was almost completely impaired by grass, and somewhat impaired by gravel (at this point it is worth noting that grass was not actually an original design scenario, since it was not anticipated to be likely after a nuclear disaster).

Containment was always successful if the sample could be collected. Once the scoop was fully closed, even traversing rough rocky terrain with the robot did not dislodge the sample. In some cases, if the scoop was not fully closed around a sample, the shaking induced by rough terrain allowed it to close completely. This observation led to the development of a "gulping" modification to the auto-scooping algorithm, in which the sample was retracted halfway, shaken slightly by the scoop, and then retracted fully. Ultimately, the gulping motion was found to not be very reliable and was not used in the final auto-scooping algorithm.

Overall, the scoop sample collection system proved to be a successful platform for collecting large, discrete samples. It was also a fine containment system, meeting the goal of combining both of those operations using the same hardware. Real-world outdoor tests demonstrated that while certain terrain could impair sample collection, good performance was achieved on most surfaces.

Chapter 4

Autonomous Sample Collection

This chapter describes in detail the work done to perform fully autonomous sample collection. It begins with a conceptual overview of how the entire system operates, discusses how images are collected, and explains the processing done to them, including custom-developed software code and algorithms. The chapter also covers some simulations created to analyze the performance of robotic hardware and its effect upon the success rate of sample collection. Then it describes the design, construction, and testing of a custom robotic mobility platform (Scoopbot) for the scoop system described in Chapter 3. Finally, the Scoopbot's performance in real-world testing for autonomous sample collection is examined along with an explanation of the "full" and "fast" operational modes for the system.

4.1 Conceptual Overview

4.1.1 Concept of Operation

Sample collection may require minute, detailed operations and precise movement and control of a ground robot or manipulator. This places a burden on the operator and can necessitate a lot of training and practice to perform the task well, and even then, the knowledge may

only relate to one particular system. If sample collection is difficult in real-time operations, where continuous feedback is possible, the problem is compounded when significant time delays are introduced into the system. For example, as mentioned earlier, the time delay in communication during planetary exploration on Mars (solely based upon distance) is an average of 20 minutes depending on the relative position of it to Earth. Under these circumstances teleoperation is essentially impossible, because even with the shortest possible delay the response to any movement will take at least 16 minutes to be communicated back to the operator. It would be preferable, therefore, to develop a system which could autonomously or semi-autonomously perform sample collection, and which would require a minimum amount of additional hardware to be placed on a robot.

These factors were the motivation behind the development of the autonomous and semi-autonomous sample collection techniques discussed here. First, the system must be able to operate asynchronously: i.e., no real-time control inputs are required of the operator. Secondly, the system should use hardware available on most robots, including those designed for planetary exploration: a camera. Structure-from-Motion (SfM) and other image-based 3D reconstruction techniques (as explained in the literature review) provide the ability to create colored 3D point clouds from a series of images. These point clouds give an accurate representation of the in situ environment. Furthermore, rather than requiring the transmission of large point cloud datasets between robot and operator, only images need to be sent from the robot, lessening the demand placed on communication bandwidth under circumstances where it may be very limited. All processing can be performed at the base station and a simple set of movement commands can be transmitted back to the robot.

4.1.2 System Overview

One of the advantages of this image-based reconstruction system is that it is platform-independent. It requires a robot (or mobile manipulator) with a camera and accurate position and orientation feedback (which can be a challenge to obtain), but otherwise no additional

hardware is needed. This research was originally intended to be implemented using the USL's ground robot; however, difficulties with its electronic systems led to the creation of a separate small robotic platform for the scoop (to be discussed in more detail at the end of the chapter). Fortunately, this change simply highlights the platform-independent nature of the system, and did not hinder research efforts. In the following paragraph, the term "robot" refers to any platform or system which has all of the necessary hardware to complete the sample collection task.

A typical operation for (semi-)autonomous sample collection runs as follows: First, the robot is driven to an area of interest that is likely to contain a "chunk" sample to collect. Once the robot is in the general area, the operator sends a command to begin a picture-taking routine. As developed, this routine drives the robot in an approx. 1.2 m diameter circle and takes 48 equally-spaced images around its perimeter, which ensures that a complete 360° model can be reconstructed (the rationale for this is explained later, but the size of the area circumscribed could be scaled to adjust for other robots and anticipated sample dimensions). These images are transmitted back to the operator's base station, which then performs the computations to create the 3D reconstruction. Now the two modes of sample selection come into play: if the characteristics of the desired sample are known ahead of time an autonomous algorithm can be used to automatically identify the 3D points associated with the sample in the point cloud. If the sample's characteristics are not known, or the operator wishes to manually select the sample, this can be done through an intuitive user interface which allows him or her to draw a bounding box around the sample's 3D points in the point cloud. Next, some processing is done to determine the sample's location and properties. Then the base station computes a path for the robot from its current position to the sample. This command is sent to the robot and it drives to the sample's location and collects it autonomously. Afterwards manual control is returned to the operator to assign the next task to the robot. The overall system operations concept is shown schematically in Figure 4.1 below.

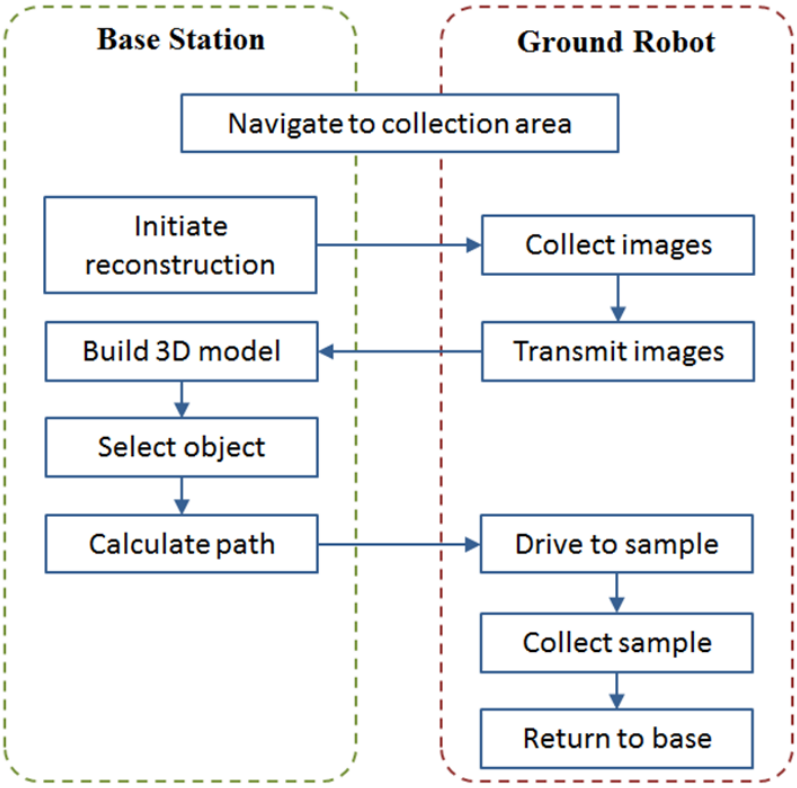


Figure 4.1: Robot-base station interaction for autonomous sample collection.

4.2 Image-based 3D Reconstructions

4.2.1 Image Collection Technique

Because the goal was to provide an operator with a colored 3D reconstruction of an entire area, the best method of collecting the images was to command the robot to drive in a circle completely enclosing the area(s) of interest. Thus, if a sufficient number of images were collected, a complete 360° model would be created. This model would allow the operator to identify the sample to collect anywhere in the area, rotating and scaling it as needed to get the best view. Also, in either the semi-autonomous or the autonomous case, a 360° model would provide enough information for an accurate assessment of the target sample's mass

and dimensions (an incomplete model would not show all sides of the object). For all tests and experiments, based upon the robot and scoop sizes, a nominally 120 cm diameter circle was used.

As stated in the "Practical Considerations" section of the literature review, many factors affect the output model quality (for example: image resolution, angle, etc.). To answer the question of the best method for collecting photos for the reconstructions, a series of tests were performed. These tests are described in the following section.

4.2.2 Analysis of Reconstruction Parameters

Number of images

Many factors influence the quality, density, and time to create the image-based 3D reconstructions. Even prior to examining the effects of individual parameters on the reconstructions, the most important consideration was the number of images required to build a 360° view. The SIFT feature descriptor can accommodate shifts in angle up to approx. 15° before it can no longer match a point between images. Using this criterion as a starting point, the 120 cm circle was divided into sections using Equation 4.1:

$$\frac{360^\circ}{n} = \alpha \leq 15^\circ \tag{4.1}$$

Where:

n = number of camera locations around circle perimeter

α = angle between subsequent camera locations

Therefore, with $\alpha = 15^\circ$ (the maximum angle), 24 camera locations are required (please note that the terms "camera locations" and "cameras" can be used interchangeably here, since having 24 separate cameras or 1 camera moved 24 times are equivalent to the reconstruction algorithms).

Next, a setup was created on the laboratory floor to provide a way to perform repeatable manual tests with the camera. A circle with the 120 cm diameter was drawn and 24 equally spaced locations for camera placement were marked with tape around it. Major lines for quadrants and quadrant bisections were also taped out through the circle. A picture of the test setup is shown below.

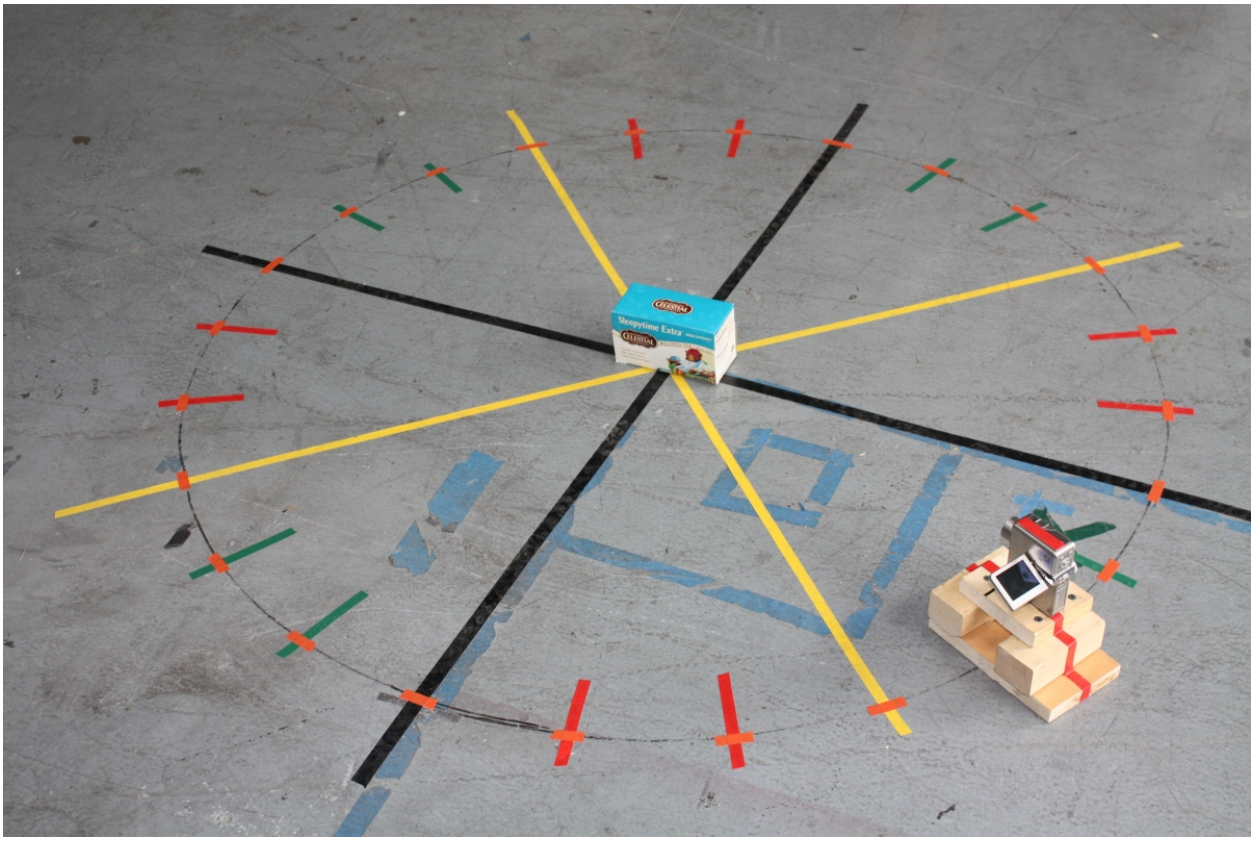


Figure 4.2: Camera setup for 3D reconstruction tests with 24 taped positions.

The camera used for all experiments, as well as on the robot, was a Canon PowerShot TX1. It has a 7.1 megapixel 1/2.5" CCD sensor with 10x optical zoom. Its dimensions are 6 cm x 3 cm x 9 cm and it weighs approx. 240g (with battery). Notice that the camera has been placed atop a movable set of small wooden blocks. These blocks lifted the camera to the height at which it would be mounted on the USL ground robot (16.5 cm above ground) so it could see over the robot treads. The angle of the camera could also be adjusted by inserting a small wooden wedge between it and the movable blocks.

The initial tests performed with only the minimum 24 images (camera positions 15 cm apart) produced inconsistent reconstruction results. In some cases the SfM software would work perfectly and provide an accurate reconstruction; in others, cameras would be poorly positioned by the bundle adjustment, and some cameras were not reconstructed at all.

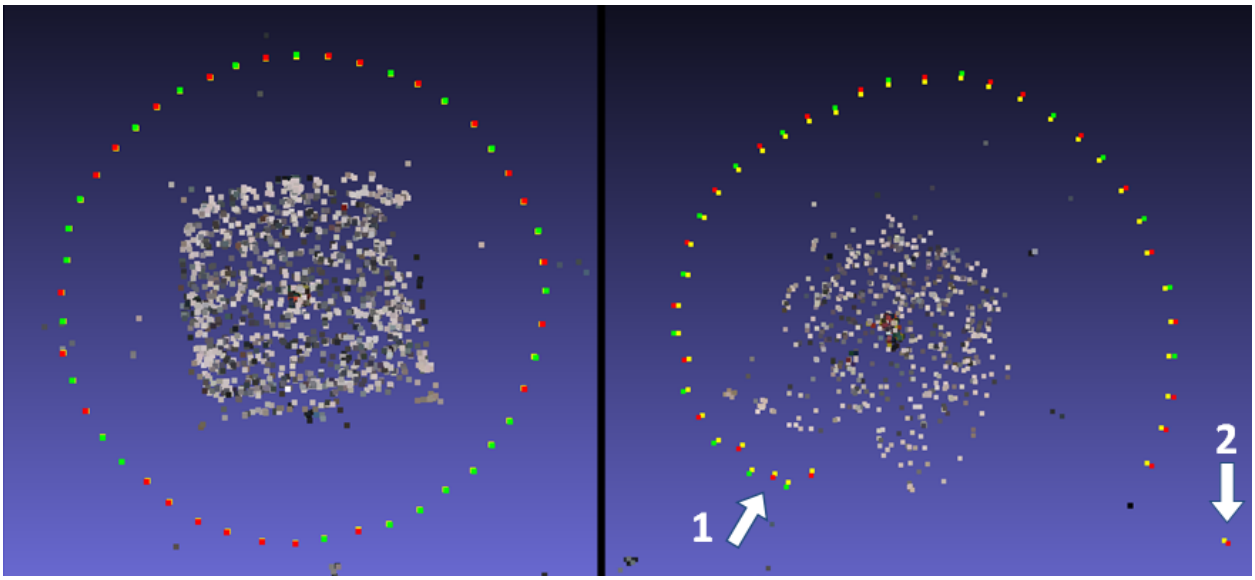


Figure 4.3: Comparison of good (left) and bad (right) structure-from-motion outputs.

In Figure 4.3, the camera positions (red and green dots) on the good reconstruction are equally spaced around the circle (as they were located in reality). In the bad reconstruction, cameras are not correctly placed: they are all spiraling outward, too bunched together (1), or completely out of line (2).

Due to this unpredictable performance more tests were run with 48 camera positions (i.e. cameras placed on and halfway between the marks). Results with 48 cameras were far better, usually creating an accurate and complete reconstruction. From the equation above, this yields an angle between cameras of 7.5°, well within the range of SIFT. Thus, moving forward with the examination of other reconstruction parameters, sets of 48 images were always used.

Image properties

These sets of tests involved varying the properties of the 48 images input into the Bundler/ PMVS pipeline. Two factors were considered: the resolution of the images, and the angle

of the camera. Three different resolutions were used: 640 x 480 (0.3 MP), 1600 x 1200 (2 MP), and 2048 x 1536 (3 MP). Two different angles were used: 0° (parallel to the ground) and -12° (pointed downwards at the center of the circle). With these 3 x 2 combinations, a total of 6 image datasets were acquired and processed. For examples of the input images and output reconstructions, please see the following section. The results from the datasets are summarized in Table 4.1.

Table 4.1: Summary of image properties test results.

Image Dataset		Results				Time
Resolution	Angle	Bundler Points	PMVS Points	Target Quality	Scene Quality	Total Time
640 x 480	-12°	3425	28498	7	9	8:23
640 x 480*	0°	3874	11411	1*	4*	10:17
1600 x 1200	-12°	8316	173466	7	8	29:58
1600 x 1200	0°	8300	98713	7	5	30:34
2048 x 1536	-12°	9969	269940	8	7	45:51
2048 x 1536	0°	9085	152063	8	5	45:06

* some cameras not reconstructed properly in this run

Each dataset was analyzed based upon the number of points from the Bundler and PMVS reconstructions, target quality and scene quality, and total time. Target and scene quality were subjective scores from 1 (worst) to 10 (best) assigned to the point clouds based upon how accurately they resembled the real-world scene. The results were not unexpected. The number of points reconstructed by both Bundler and PMVS rose with the increasing resolution of the images. Processing time also increased substantially for larger images, as anticipated. However, some very interesting trends were also uncovered. More points were reconstructed by PMVS in the downward-facing images, perhaps because their view of the nearby ground area contained a lot of close-up detail. This would be an advantage to using downward-facing images, since more 3D points would be located in the target sampling area.

More importantly, the qualitative scores did not vary too much between image sets, except for the 2^{nd} set where some cameras were not properly reconstructed. The "Target Quality"

score was always between 7 and 8, indicating that the object to be collected was reconstructed well. "Scene Quality" was more varied, however. The average scene quality score for the downward-facing image sets was 8, compared with only 4.67 for the horizontally-directed ones (or 5 if the bad set is excluded). Therefore, it appeared that the downward-facing camera position was superior to the horizontal one. Among those downward-facing sets, run times became a consideration. A faster run time, if not associated with a significant degradation in quality, would be better than a slower one, since the robot (and operator!) would spend less time sitting around waiting for the collection command. Looking to scores again, the highest combined qualitative score was 16, for the 640 x 480 downward-facing image set. Since it also had the fastest run time of all of the datasets, and proved to be consistently well reconstructed with other images, it was chosen as the best configuration for the next round of testing.

PMVS settings

The Patch Multi-View Stereo (PMVS) reconstruction algorithm allows users to modify several of the reconstruction variables via the pmvs_options.txt text file. Modification of three settings was examined: minImageNum, sequence, and level, and compared to the default case.

Table 4.2: Settings modified in PMVS tests

Setting	Case 1	Case 2	Case 3	Default
minImageNum	2	3	3	3
sequence	-1	3	-1	-1
level	1	1	0	1

Each of these settings changes one aspect of the reconstruction. "MinImageNum" specifies how many images a 3D point must be visible in to be reconstructed; the default is 3. "Sequence" determines whether or not to force PMVS to use the images in the order they are provided (the value of 3 only allows the software to use images $\pm$ 3 positions of the current one). Finally, "level" tells the software whether or not to scale down the images when it performs the reconstruction. Its default value is 1, or a 50% reduction in the width and

height of an image (a 75% reduction in area). Larger images (level 0, no size reduction) will obviously increase processing time because more pixels are being examined.

Again, several factors were considered to assess the quality of the PMVS outputs. The number of points, target quality, scene quality, and total time were all recorded after each run with the different settings. The same 48 images and SfM (Bundler) output were provided as PMVS inputs in each case. The results are presented below.

Table 4.3: Summary of PMVS settings modification results.

	Results			Time
Case	# of Points	Target Quality	Scene Quality	Total Time
1	43232	8	7	2:56
2	35354	8	8	2:42
3	122900	8	7	9:50
Default	35528	7	8	3:44

As before the results correspond to what can be anticipated by changing each setting. Interestingly, all cases produced 3D point clouds of essentially equal quality, but not of equal quantity. Either reducing the number of images required to see a point (Case 1) or increasing the size of the input images (Case 3) resulted in a reconstruction with more points. Forcing an order on the software (Case 2) decreased processing time significantly as well. Like the previous image properties tests, it made the most sense to choose the option with the best quality and lowest processing time. Therefore Case 2, the sequence constraint applied, gave the best PMVS settings choices.

4.2.3 Typical 3D Point Cloud Results

So much time is being spent discussing 3D reconstructions and their point clouds that it implores the author to devote a small section to explaining what the input images and output colored 3D point clouds typically look like. First, here is another view of the test setup with a box designed to replicate an outdoor sandy/rocky environment (it was used for

the tests described in the three previous sections).

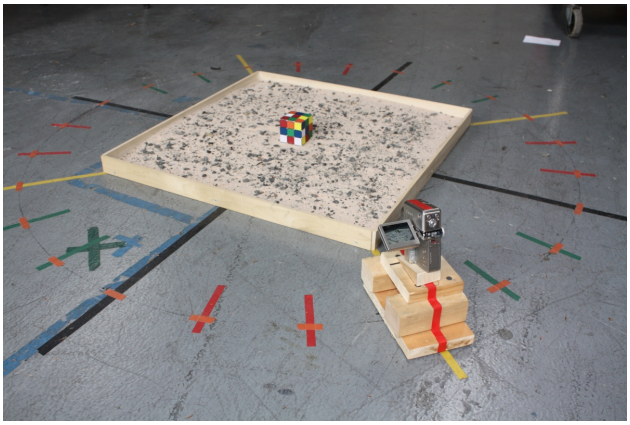


Figure 4.4: Outdoor environment replica test setup with cube. The target object to collect would be the colored cube in the center of the box.

The first step in the reconstruction was to feed a set of images into the Structure-from-Motion software (Bundler). The image sets typically appeared as shown in Figure 4.5. Notice the shift in camera angle between images.

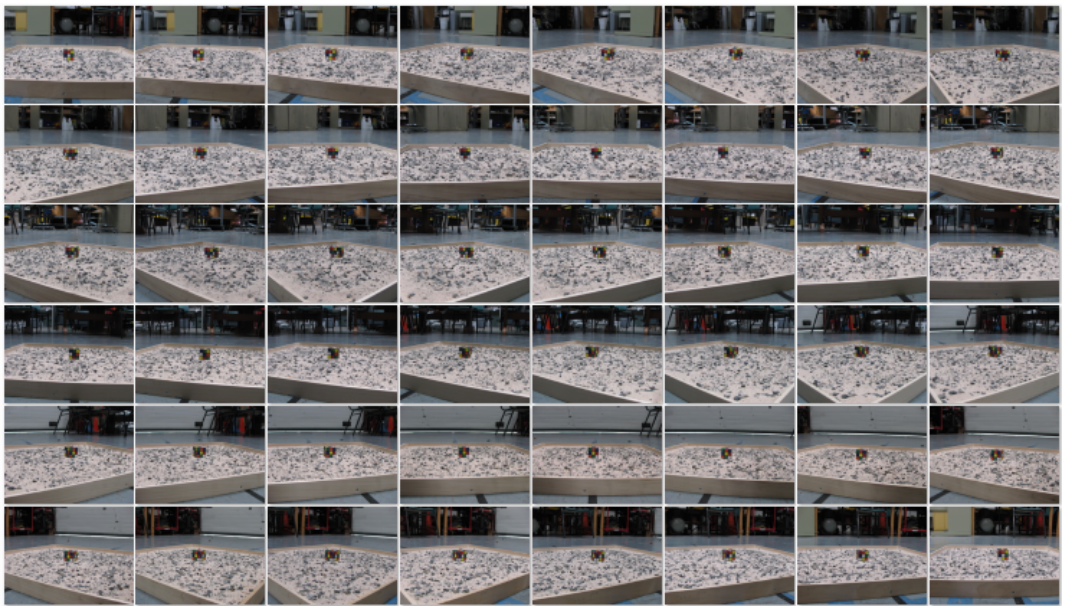


Figure 4.5: Sample image dataset (48 640 x 480 pictures).

Bundler processed these images and outputted a sparse point cloud that included reconstructed estimates of the camera positions (red and green dots) and view directions (yellow).

The number of points in a reconstruction varied with the dimensions of the input images; this sparse reconstruction from 48 640 x 480 pictures contains 3425 points.

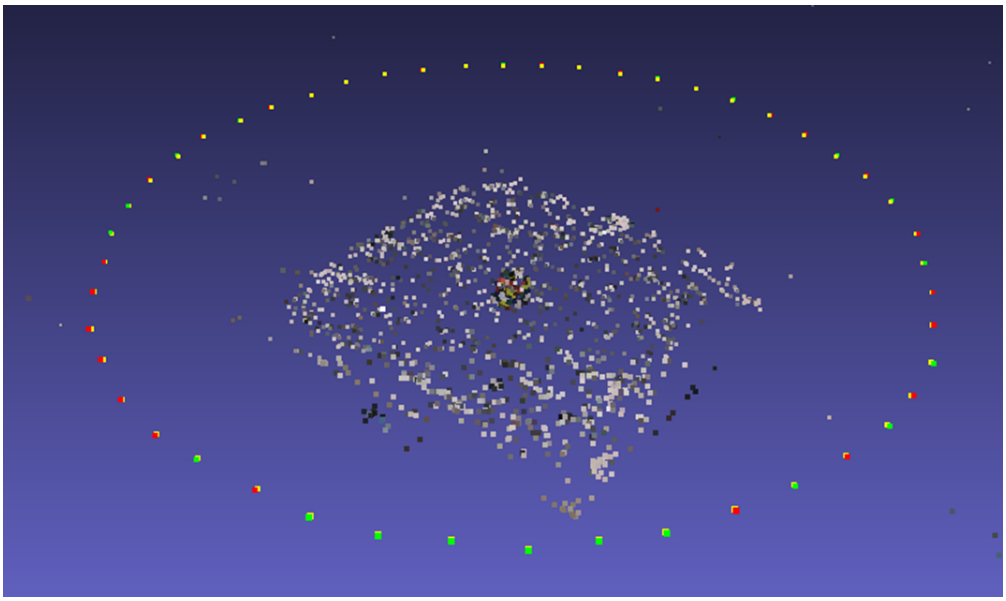


Figure 4.6: Bundler SfM output.

The images, along with the Bundler output, went as inputs into PMVS. PMVS created a dense point cloud such as the following (with 28,498 points):

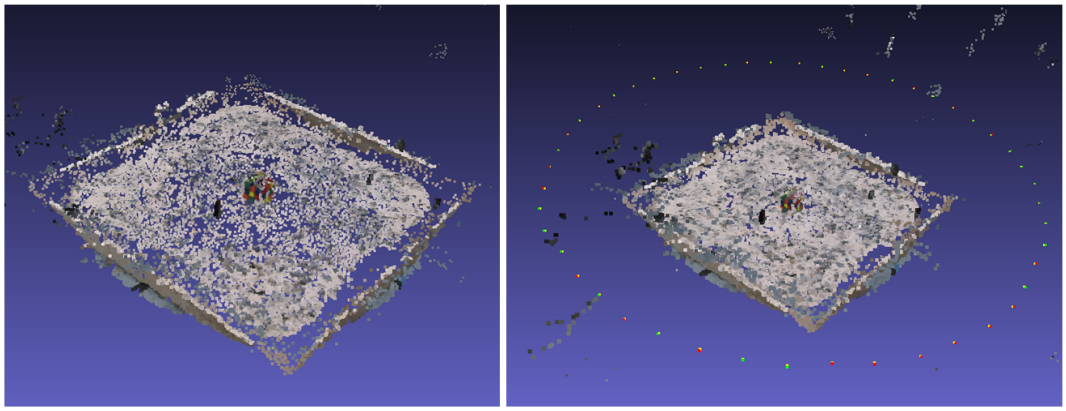


Figure 4.7: PMVS output alone (left), and combined with Bundler output (right).

Both the Bundler and PMVS reconstructions were used with custom software written to process the point clouds and provide the robot with movement commands.

4.3 Software and Point Cloud Processing

4.3.1 Overview of Software Architecture

Interpreting and processing the 3D point clouds output by the reconstruction pipeline required developing a suite of custom software programs. An additional emphasis was placed on making the user interaction as simple and straightforward as possible, since the underlying code and operations were fairly complex. Ultimately, the software could operate in two modes: one for semi-autonomous sample collection (some user interaction needed), and one for fully autonomous sample collection (user merely launches the program, then leaves for a cup of coffee). Both modes implement the same software base, but the user interaction portion is not present in the autonomous mode. These differences will be explained where applicable; otherwise all details of operation apply to both cases.

The software architecture was created as a series of Matlab functions and programs which manipulate the data and then pass modified 3D points or variables along to the next function in the sequence. Figure 4.8 shows the overall operation of the programs in semi-autonomous mode. Here RobotInterface (see Figure 4.9) is the main GUI window that is presented to the user. The user can select to run any of the right hand side programs in Figure 4.8. There is an overall order in which the programs are run (from top to bottom), but the user can click a button and rerun any of the individual modules as necessary (for example, to see the preview animation again).

For convenience, the UI is divided into labeled frames from Imaging (Step 1) to Scaling to Selection to Collection (Step 4). Normally the user just selects all of these steps in direct sequence. The UI also reports relevant data for assessing whether or not the sample can be collected using the scoop. If any of the dimensions exceed their scoopability limits, the green text reading “Object is scoopable” changes to a red error message “WARNING: Too big to scoop”. The sample’s material composition is not known, but its volume is, so masses for common material types are reported as well. For reference, the scoop’s mass design limit

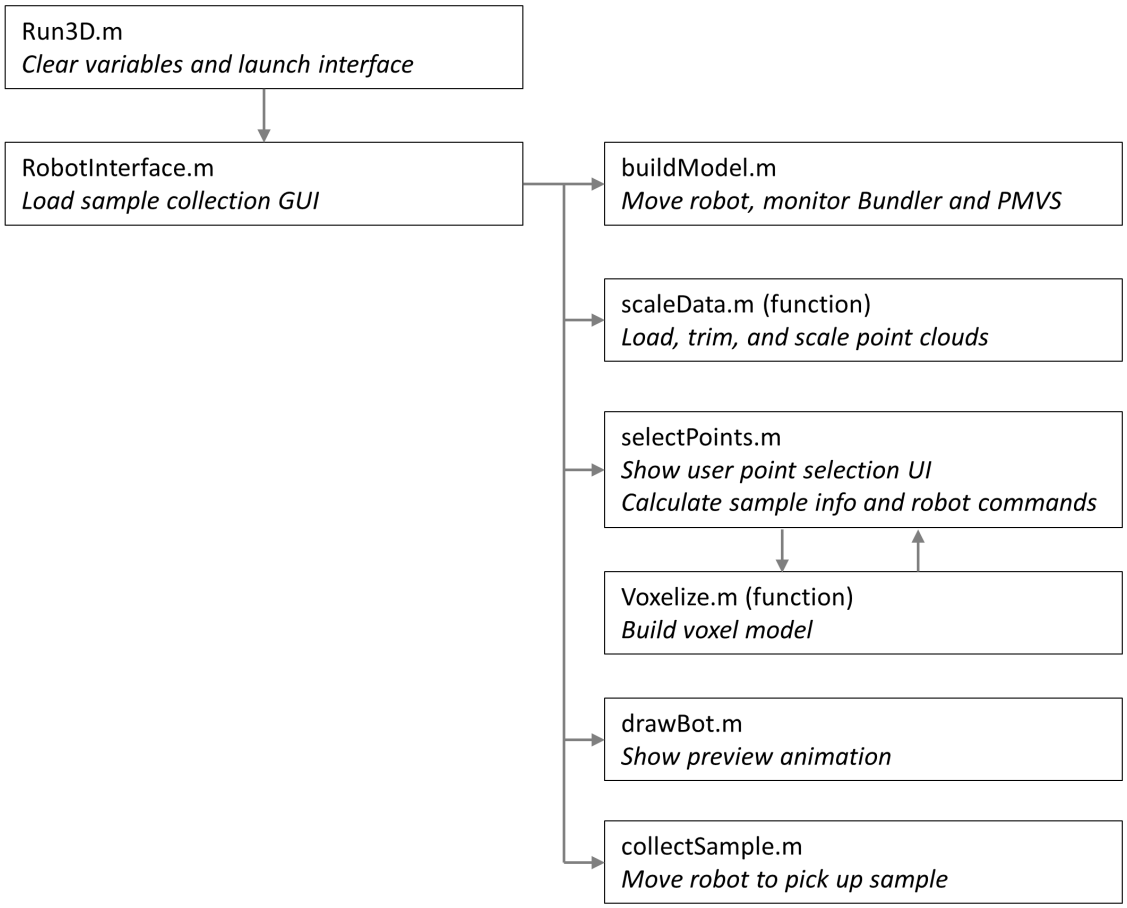


Figure 4.8: Software code organization for semi-autonomous sampling.

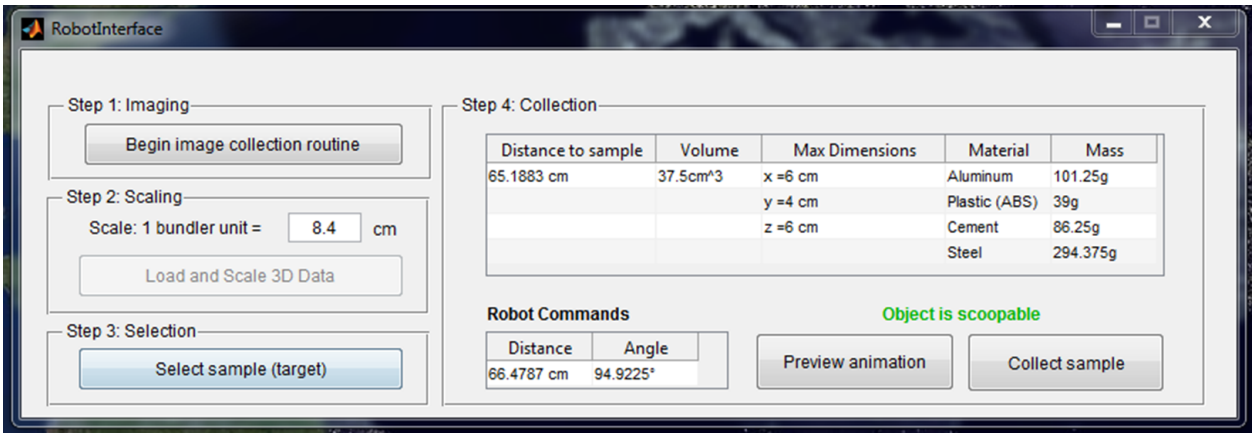


Figure 4.9: Main program GUI for semi-autonomous sampling.

was a 250g sample.

The programs also exchange, import, and export data (see Figure 4.10). Blue arrows represent external data inputs, red arrows indicate data calculated and transferred internally, purple arrows show data traded at the function level, and the green arrow shows where data is output to the ground robot.

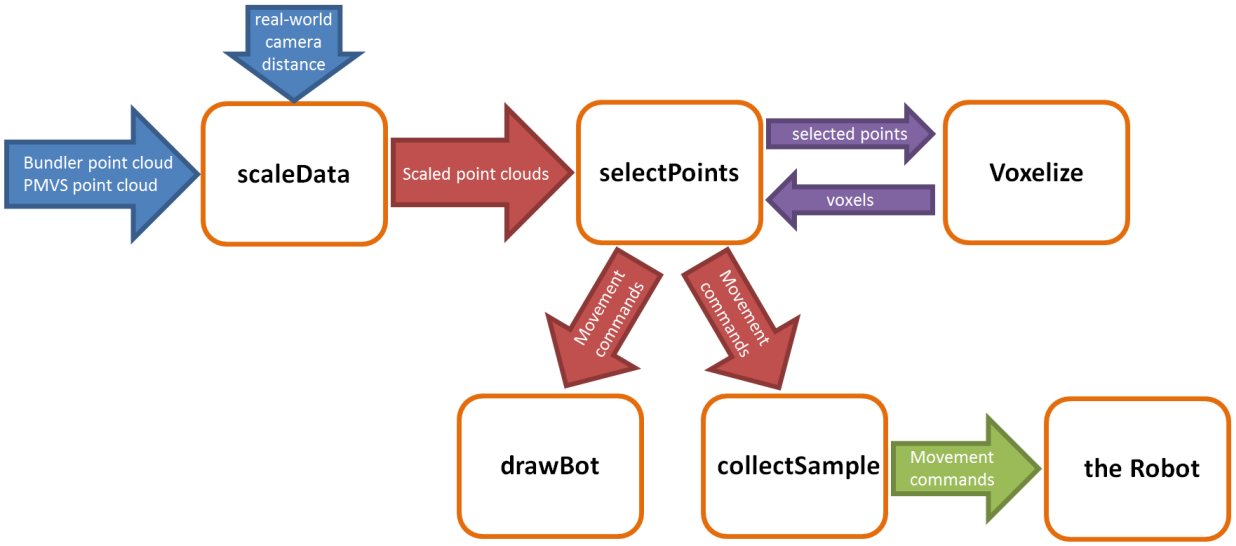


Figure 4.10: Program data flow.

The data flow structure is the same for both collection cases (autonomous and semi-autonomous).

The overall operation of the fully autonomous code is more streamlined since no user inputs are required, as seen in Figure 4.11.

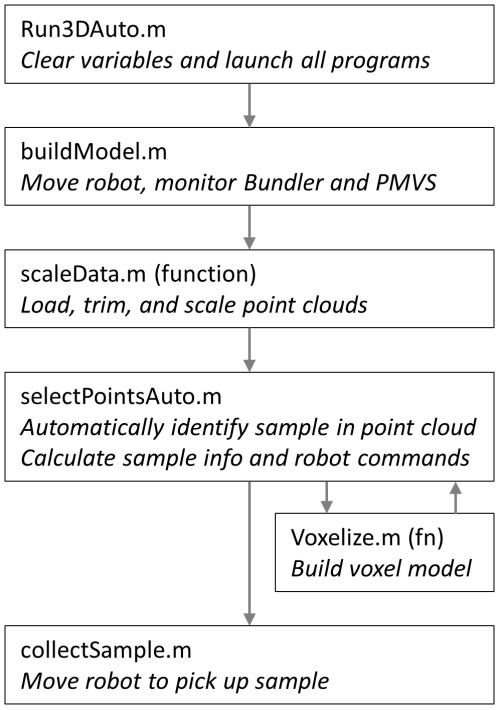


Figure 4.11: Software code organization for autonomous sampling.

Because this system is fully autonomous, the only user interaction required (assuming the robot is at the area of interest) is running the "Run3DAuto" program, which starts and waits for all subsequent ones shown on the diagram. The voxelize function only communicates with selectPoints, so it is not directly attached to the program flow. Next, the algorithms governing each component/program of the process will be discussed in greater detail.

4.3.2 Trimming and Scaling

The 3D point clouds produced directly from the structure-from-motion (Bundler) and multi-view stereo (PMVS) software suffer from a limitation inherent in image-based reconstructions, and images themselves. Considering a normal picture, it is easy to see that there is no inherent scale associated with it. For example, is a picture of a house a small model or

one lot of a suburban neighborhood? A human can easily examine context clues present in the image (people's heights, everyday objects, etc.) and infer an overall scale to the scene. Computers, however, cannot do this except under very specific circumstances. Therefore, being unable to determine the scale of a scene presents a major obstacle to being able to command a robot to drive a specified distance and pick up an object.

Fortunately, the camera positions and view directions are reconstructed by Bundler along with the 3D locations of feature points. If the real-world locations of the camera were known when it gathered the images, it should be possible to correlate the size of the reconstructed scene with its actual dimensions. This is indeed the case. The ground robot is equipped with motors and encoders (or alternatively stepper motors), so the distance it travels between pictures is known. Finding two adjacent cameras in the reconstruction, the overall scene can be multiplied by a scaling factor to make the scene distance equal to a real-world distance, i.e. 1 unit in the scene = 1 cm in reality. This scaling and other manipulation is handled by the function scaleData.m; its overall algorithm is listed as Algorithm 4.1 and described in the following paragraph.

Starting from the top of the algorithm, the first task is to call the scaling function with the distance value measured/set by the robot between camera locations (*dist*). Then the two point clouds (BPC and PPC) are loaded into memory. Next an optional rotation step can be performed to align the point cloud ground plane with the x-z axes. This is useful for when the operator manually selects the bounding box around the sample, because it helps ensure that their view of the sample won't be obstructed by the points forming the ground around it. It's also useful in aligning the point clouds for later voxelization. Then the camera positions are extracted by checking the colors of the points at the bottom of the BPC file (cameras area always either pure red or green, and viewing directions are always pure yellow and paired with a camera). However, cameras are usually not added to the Bundler scene in the order in which the pictures were taken. Therefore, another file (BundleFastOutput.txt) is used to provide the correct camera order [27]. After reordering the cameras, the distance between each camera pair is calculated using the three-dimensional distance formula (Equation 4.2).

Algorithm 4.1 Scaling and trimming 3D point clouds.

```
Distance passed to function when called (dist)
Load PMVS point cloud (PPC)
if rotate points enabled then
    Perform 2-axis rotation to align ground plane with x-z axes
end if
Extract camera and view positions from BPC (check point color)
Import camera order list (BundleFastOutput.txt)
Rearrange cameras into sequential order
while i < # of cameras - 1 do
    delta(i) = computed distance between camera pair
end while
delta_cam = median(delta)
scalefactor = dist/delta_cam
Multiply point clouds (x,y,z) by scalefactor
if distance check enabled then
    while i ≤# of PMVS points do
        if point (x,y,z) > camera positions + margin then
            set point (x,y,z) = null
        end if
    end while
end if
Save scaled BPC
Save scaled PPC
Save scaled final camera position and view direction
```

$$delta(i) = \sqrt{(x_{cam(i)} - x_{cam(i+1)})^2 + (y_{cam(i)} - y_{cam(i+1)})^2 + (z_{cam(i)} - z_{cam(i+1)})^2} \tag{4.2}$$

Next an average distance value *delta_cam* is found by taking the median of all camera deltas. The median is used instead of mean because it is more robust to outliers, so if one camera were reconstructed incorrectly it would have only a minor influence on the reported average value. Dividing the input distance by the average *delta_cam* produces a scaling factor:

$$scalefactor = dist/delta\_cam \tag{4.3}$$

which will be used to convert up from scene scale into real-world scale. The x, y, and z coordinates of the BPC and PPC are multiplied by this value and later saved into two scaled point cloud files. For the current setup the real-world value of *dist* is 8.4 cm between cameras.

Another optional (but frequently very useful) manipulation is trimming of the PPC which is shown to the user. To perform this check, points are compared to a bounding box established by the maximum and minimum coordinates of the camera positions in the x, y, and z directions. Any 3D point lying outside of the box by more than a predefined margin (currently set at 0 cm for x-z plane, and 30 cm for y height) is discarded. PMVS usually is able to reconstruct some surfaces far outside of the sampling area, so eliminating those speeds later computations without sacrificing useful data.

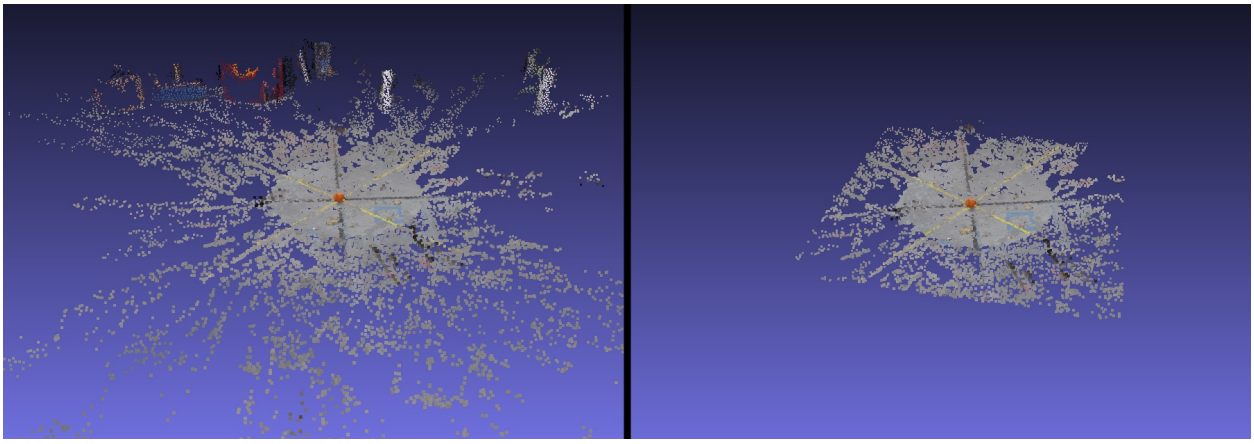


Figure 4.12: PMVS point cloud before (left) and after (right) automatic trimming.

The scaleData.m function also saves the final (last robot position) camera's scaled location and view direction to a separate text file, since these parameters will be required for other calculations, and only the PPC will be needed by the other software programs.

4.3.3 Selection, Voxelization and Analysis

There is still a lot of processing to be done once a scaled and trimmed point cloud is created. The sample needs to be selected, its location determined, and its mass and dimensions

reported to ensure that collection is possible. These operations are performed by selectPoints or selectPointsAuto and its companion function voxelize. If the user manually identifies the target, selectPoints is run, otherwise selectPointsAuto will perform the same selection process automatically. In both cases voxelize creates a voxel (volume element, a.k.a. 3D pixel) representation of the selected points for analysis. The algorithms for these processes are listed below and subsequently explained in greater detail.

Algorithm 4.2 selectPoints - manual user selection of sample

```
1: Load scaled, trimmed PPC
2: begin user sample selection
3:       Display 3D plot of point cloud to user in x-y plane
4:       User selects upper left-hand and lower right-hand corners of bounding box (2D)
5:       Rotate plot 90 degrees
6:       Display 3D plot of point cloud to user in x-z plane
7:       User selects upper left-hand and lower right-hand corners of bounding box (2D)
8: end user sample selection
9: Display 3D bounding box around sample to user
10: for i = 1 to numpts do
11:    if x,y,z(i) inside bounding box  then
12:       Add point to box_pts
13:    end if
14: end for
15: Plot points found inside of bounding box (box_pts)
16: — call voxelization —
```

After the trimmed PMVS point cloud (PPC) is loaded a 3D plot is shown to the user. This plot's initial orientation is a view of the x-y plane (i.e. a side view, since x-z has been previously defined as the ground plane). Then the user selects upper left-hand and lower right-hand points to define the perimeter of the bounding rectangle in the x-y plane. Next, the view rotates through 90° into the x-z plane (a top-down view) and the user picks another pair of bounding points.

Notice that Figure 4.13 shows the selection of the upper left-hand corner of the bounding rectangle around an orange rock which appears in the center of the figure. After picking the second point in this view, four points total in perpendicular planes would be defined (2 in

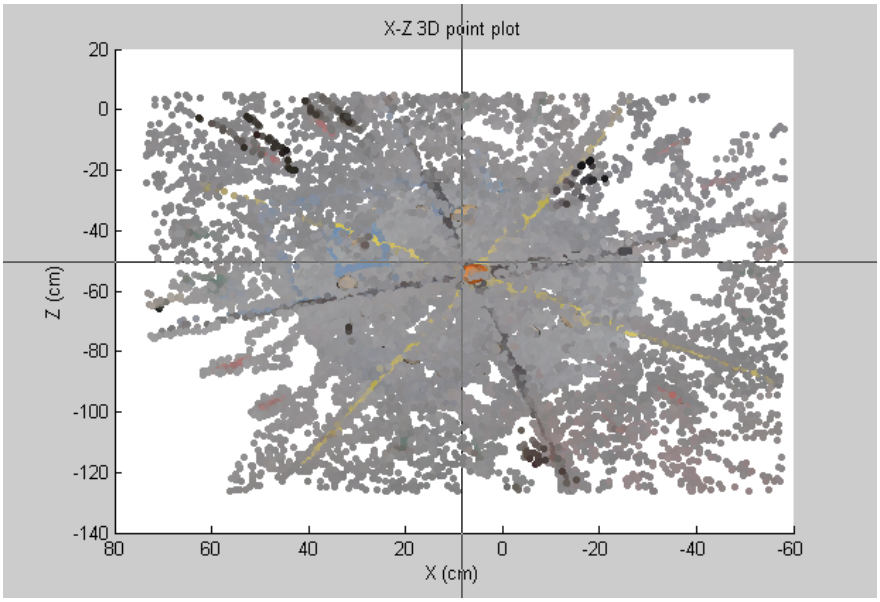


Figure 4.13: Selection of first corner of sample bounding rectangle in x-z plane.

x-y and 2 in x-z), establishing a perimeter for the bounding box volume. The algorithm goes through all of the points in the PPC (since they are not indexed spatially) and selects the ones which fall inside the bounding box. These points are assumed to belong to the sample to collect and sent to the voxelization routine.

Alternatively, selectPointsAuto (Algorithm 4.3) can be used to automatically locate the sample if fully autonomous collection is being employed. It operates identically to selectPoints (Algorithm 4.2) except that the user selection process has been replaced with an automatic one. It is extremely important to note that while a color-based heuristic is currently used to find the orange rock in the point cloud, any heuristic based on any distinctive sample feature (size, color, etc.) could be easily substituted into the code.

Algorithm 4.3 selectPointsAuto - automatic identification and selection of sample

```
Load scaled, trimmed PPC
Set color deviation threshold
for i = 1 to numpts do
    if color <r,g,b>(i) within threshold of sample color  then
        Add point to sample_pts
    end if
end for
Find maximum and minimum of sample_pts in x,y,z directions
Assign max/min x,y,z to dimensions of bounding box
for i = 1 to numpts do
    if x,y,z(i) inside bounding box  then
        Add point to box_pts
    end if
end for
Plot points found inside of bounding box (box_pts)
— call voxelization —
```

Figure 4.14: Automatic bounding box around sample in 3D point cloud.

The algorithm loops through all 3D points, picking out those which appear orange to within a given threshold value (the sample rock was the only solid orange object in the scene). Orange is defined as a color whose RGB triplet has the relation (red, green = red*2, blue = red*4). After experimentation, a threshold of 25/255 (in Matlab R, G, and B range from 0-1) was used because it consistently identified the majority of points associated with the sample without returning false matches elsewhere. A bounding box is then created based

upon the maximum and minimum orange point locations in the x, y, and z directions. After running either selectPoints or selectPointsAuto, the user sees a bound box drawn on the 3D point cloud (Figure 4.14).

The next step in the process is the voxelization routine in Algorithm 4.4.

Algorithm 4.4 voxelize - build voxel representation of sample

```
Inputs: Receive list of box_pts
Find bounding box dimensions (min/max x,y,z of box_pts)
Set voxel grid resolution (xinc, yinc, zinc)
- Build voxel grid -
for x = 1 to xmax − xmin step xinc do      {These are referred to as the XYZ loops}
   for y = 1 to ymax − ymin step yinc do
      for z = 1 to zmax − zmin step zinc do
         Initialize voxel at x,y,z
         Set points contained = 0
         Set occupancy = 0
      end for
   end for
end for
- Find point voxel locations -
for k = 1 to # of box_pts do
   flag = 0 (break when correct voxel is found for point(k))
   for XYZ loops do
      if point(k) x,y,z coordinates within voxel x,y,z then
         Increment voxel's point count
         flag = 1 (break)
      end if
   end for
end for
Set occupancy threshold (number of points per voxel)
- Check voxel occupancy -
for XYZ loops do
   if voxel point count ≥ threshold  then
      Mark as occupied
   end if
end for
```

```
- Check and fill interior voxels based upon external occupancy -
(i.e. go from shell to solid object)
x-z plane is assumed to be horizontal, y is vertical direction
for y = 1 to ymax − ymin step yinc do
   for x = 1 to xmax − xmin step xinc do
      Check +z direction
      for z = 1 to zmax − zmin step zinc do
         if voxel is unoccupied then
            Mark as examined
         else if voxel is occupied or already examined then
            Exit this for loop
         end if
      end for
      Check -z direction
      for z = zmax − zmin TO 1 step zinc do
         if voxel is unoccupied then
            Mark as examined
         else if voxel is occupied or already examined then
            Exit this for loop
         end if
      end for
   end for
end for
- Draw and display voxel grid -
for All voxels do
   Color orange for occupied
   Color grey for internal
   Leave empty voxels blank
end for
- Find center of mass -
for XYZ loops do
   if voxel is occupied then
      Add contribution to x,y,z center of mass
   end if
end for
Report total number of occupied voxels, center of mass, etc.
```

The algorithm makes the voxelization routine appear more circuitous than it is in reality, since most of the procedures have a number of loops which run though the x, y, and z directions in sequence. The only input to the voxelization function is the set of points which fell inside of the (automatically or manually) drawn bounding box.

The first step in voxelization is to build the blank voxel grid. This is done by finding the maximum and minimum values in the x, y, and z directions from the input points. The number of voxels created will be:

$$n_{voxels} = (\Delta x * 2) * (\Delta y * 2) * (\Delta z * 2) \tag{4.4}$$

Where the Δ is the difference between the minimum and maximum point locations in that direction. Remember that the point cloud is scaled up to centimeters (the above Δ's are in cm), so with this setup each voxel will be created with dimensions of 0.5 cm x 0.5 cm x 0.5 cm. These dimensions offer a good trade-off between speed and accuracy. A blank voxel has its number of points, color, and occupancy all set to zero.

The second step is to loop through all of the sample points and place them into voxels. This is done by checking each point's 3D location versus each voxel's and stopping once the point's location is falls inside of the voxel being examined by the current loop iterations. The following, and closely related step, is to run through all of the voxels and count how many points each one contains. If this value is greater than or equal to a threshold (set to 1 point for the 640 x 480 image sets) then the voxel is marked as being occupied.

The third step of the voxelization process is perhaps the most complex. It involves establishing which voxels lie on the inside (should be filled) versus the outside (should be empty) of the sample. Neither of these sets of voxels will contain 3D points, so their state must be inferred from their position. When the voxels were initialized they were set by default to what would eventually be interpreted as a filled state. At the end of the process any voxels not marked as empty would be considered filled and part of the sample. Hence, it

is mandatory to check for empty voxels. The 3D reconstruction can only rebuild what it sees, and that is the exterior surface of the object. Therefore, a method was developed to mark voxels as either exterior or interior given the fact that only the ones associated with points forming the "shell" of the object would be correctly marked as filled by the second thresholding step.

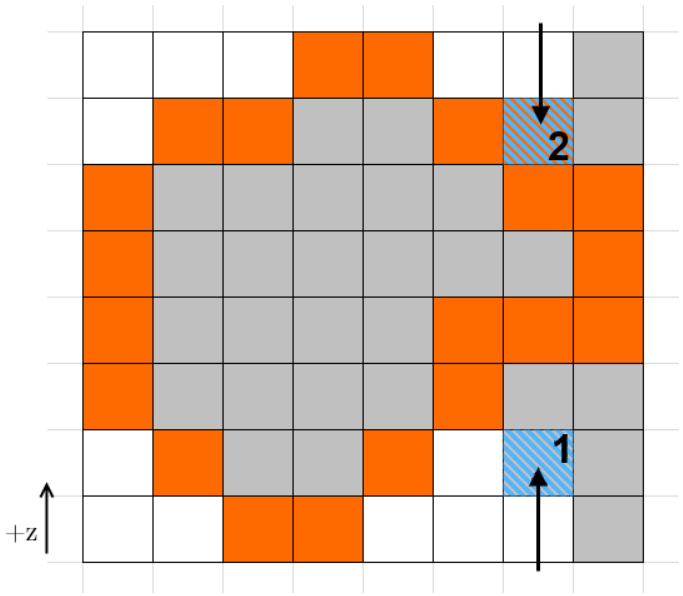


Figure 4.15: Horizontal slice of voxel grid interior/exterior marking scheme.

Here orientation is of significance: since the ground plane is the x-z plane, cutting the object through parallel planes at various heights y provides slices of the object. Each slice of voxels is examined along columns from the $-$z toward the +z direction, and then from the +z towards the $-$z direction. The algorithm marks voxels as exterior (empty) until it reaches one filled during the second step of the process or intersects with a voxel set empty from the other direction. This is illustrated in Figure 4.15, where the arrows demonstrate the direction of the checking along columns as the overall examination proceeds from left to right horizontally. Orange voxels have been marked as occupied for containing 3D points, grey voxels are unchecked (assumed filled), and white voxels have been marked empty. Voxels currently being examined have a diagonal hashing. Voxel 1 is being checked and will be found to be empty since an orange or another white voxel has not yet been encountered traveling

from the bottom up. Voxel 2 will be found to be filled since it is orange, and checking in the downward direction will cease. Ultimately, this procedure will leave all voxels inside the object unchecked and filled but set all those outside to be definitively empty.

For the user's benefit (and to help ensure that voxels are being correctly identified), now the points in the sample point cloud and the voxels are drawn onscreen. As Figure 4.16 demonstrates, the point cloud is shown in its true colors, and voxels are displayed using the orange/grey/white-empty designations described above. At this point all figure dimensions are real-world measurements in centimeters.

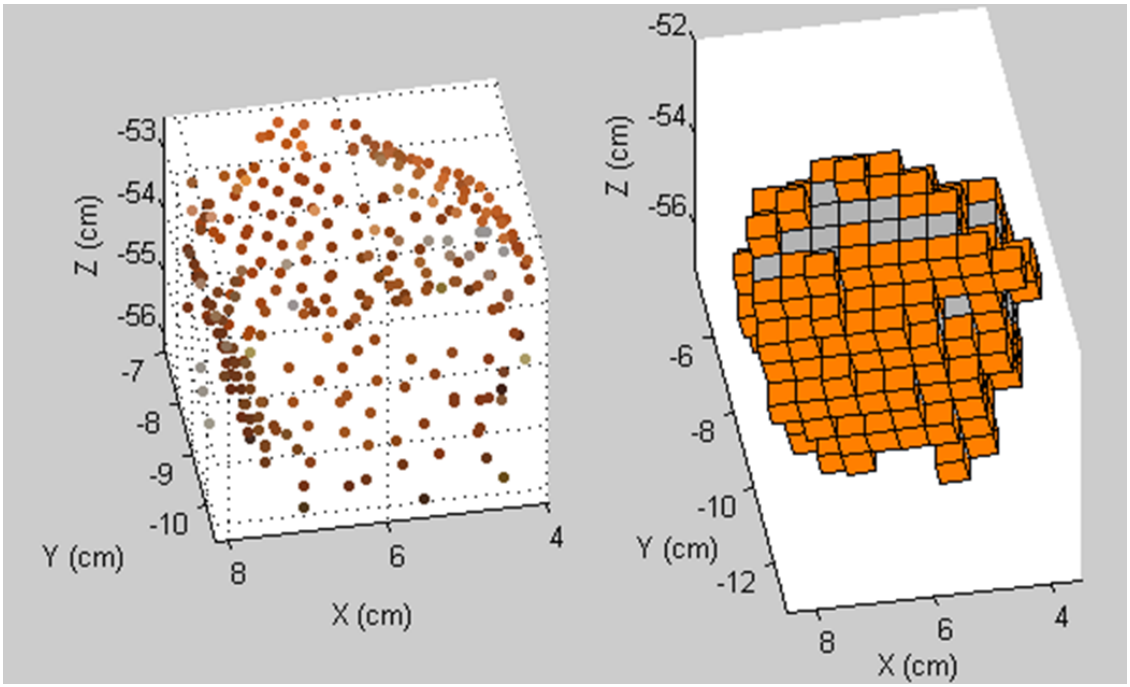


Figure 4.16: 3D view of sample point cloud and its corresponding voxel representation. This view is fully interactive and can be rotated to any orientation desired by the user.

The fourth and final step of the voxelization routine is to calculate the spatial center of the sample (later used for collection computations and also equal to the center of mass if the sample has a uniform composition). A series of loops perform this operation by "weighing" each *occupied voxel* into the overall value, detailed in Equation 4.5.

$$\sum_{x=1}^{\Delta x}\sum_{y=1}^{\Delta y}\sum_{z=1}^{\Delta z}\binom{ctr_x=ctr_x+voxel\_ctr_x}{ctr_y=ctr_y+voxel\_ctr_y}{ctr_z=ctr_z+voxel\_ctr_z} \tag{4.5}$$

Here x,y,z represent a position in the voxel grid and the Δ's are as defined for Equation 4.4. The number of filled voxels is also counted, and the aggregate sample x,y,z center is calculated by dividing the numerical sum of centers (ctr_x, ctr_y, ctr_z from above) by it. Finally, after all of these steps are performed, control is passed back to the selectPoints program along with centers, dimensions, and the other quantities calculated in the voxelization function.

4.3.4 Calculating Robot Commands

After all of the point cloud processing has been completed, a few calculations are performed to figure out the distance and angle which the robot must travel to reach the sample. This procedure was developed using the Scoopbot described in Section 4.5 but its considerations apply to the USL ground robot as well.

Remember from Section 4.3.2 that the camera positions have been reconstructed by the SfM algorithm and are considered known locations (even though they might contain some minor error since they are determined via optimization). Also recall that each camera position has an associated view angle which can be calculated from the reconstruction. For the purposes of sample collection, the last camera position and its view angle are the most important, since they indicate the robot's location at the conclusion of the image-gathering routine (any reference to "camera" or "view" in this section refers to that position). The robot's final location following image-gathering is its starting location for traveling to collect the sample since it does not move while processing is performed on the base station computer.

The camera is rigidly mounted to the robot frame, so its view angle should define the robot's orientation as well. With the current setup, the camera is perpendicular to the direction of travel, as shown in Figure 4.17.

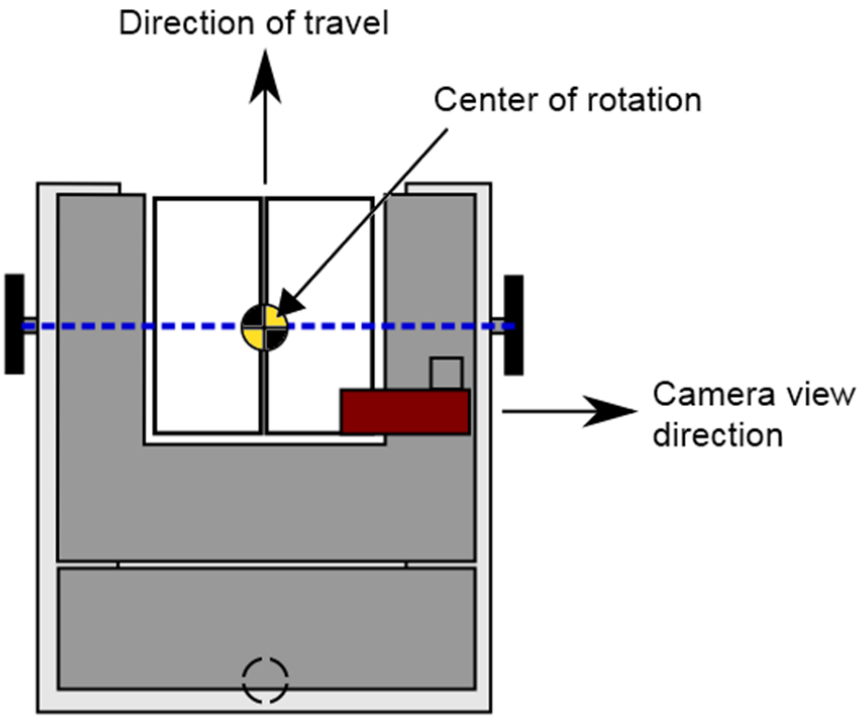


Figure 4.17: Schematic of camera view and position on Scoopbot robot.

The exact reasons for placing the camera perpendicular are explained later, but for the time being the more important consideration is that the robot's center of rotation is not co-located with the camera position. The robot will pivot around a point halfway between its two drive wheels since the robot's third contact point with the ground is an unactuated omnidirectional caster. It is also extremely important to note that in this case the center of the scoop (sample collection device) is coincident with the center of rotation of the robot so no additional calculations need to be performed for it. Still, planar offsets in two directions must be accounted for when calculating the movements to collect the sample. These offsets are shown in Figure 4.18.

Here we see that the camera's view direction establishes the direction of *offsetX* and that *offsetY* is perpendicular to it. Thus, rotation offsets for any ground robot could be defined by changing the values of *offsetX* and *offsetY*. For the Scoopbot they carry values of 7.5 cm and 2.5 cm respectively. The view angle in the world coordinate system (x-z in Figure 4.18)

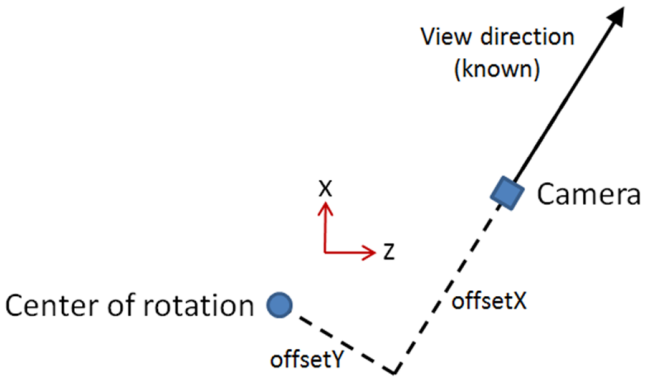


Figure 4.18: Planar offsets for collection calculations.

is calculated by taking the inverse tangent of the x and z components of the view direction vector (Equation 4.6).

$$\theta = tan^{-1}\left(\frac{v_z}{v_x}\right) \tag{4.6}$$

Where:

θ = view angle

v_z = z-component of view direction vector

v_x = x-component of view direction vector

Once this value has been calculated the center of rotation is determined using the following relations:

$$p_x = c_x - offsetX * cos(\theta) - offsetY * sin(\theta) \tag{4.7}$$

$$p_z = c_z - offsetX * sin(\theta) + offsetY * cos(\theta) \tag{4.8}$$

Where:

p_x = center of rotation in the x direction

p_z = center of rotation in the z direction

c_x = camera center in the x direction

c_z = camera center in the z direction

Finally, the position direction vector is found by subtracting the new position (p_x and p_z) from the target sample position produced from the voxelization routine. This vector will indicate the angle from the robot's center of rotation to the sample, as shown in Equation 4.9:

$$\beta = tan^{-1}\left(\frac{r_z}{r_x}\right) \tag{4.9}$$

Where:

β = angle of vector in x-z plane from robot's center of rotation to sample

r_x = x-component of robot-to-sample vector

r_z = z-component of robot-to-sample vector

Now both the current view angle θ (which also defines the robot's orientation) and current robot-to-sample angle β are known. Therefore, the angle Δ through which the robot must turn about its center of rotation to face the sample is the difference between the two:

$$\Delta = \theta - \beta \tag{4.10}$$

This angle will have a negative value if the sample is a counter-clockwise rotation from the current view direction and a positive value if the sample is a clockwise rotation from the current view direction. The angle Δ is added to the fixed 90° offset between the camera and robot directions to produce the total angle of rotation:

$$robot\_angle = \Delta + 90^\circ \tag{4.11}$$

Finally, the very last step in the robot movement calculations is to find the distance between the robot's center of rotation and the sample. This is done using the 2D version of the distance formula from Equation 4.2. Figure 4.19 shows a graphical representation of the angles and distances involved.

$$robot\_dist = \sqrt{(p_x - s_x)^2 + (p_z - s_z)^2} \tag{4.12}$$

Where:

s_x = sample center in the x direction

s_z = sample center in the z direction

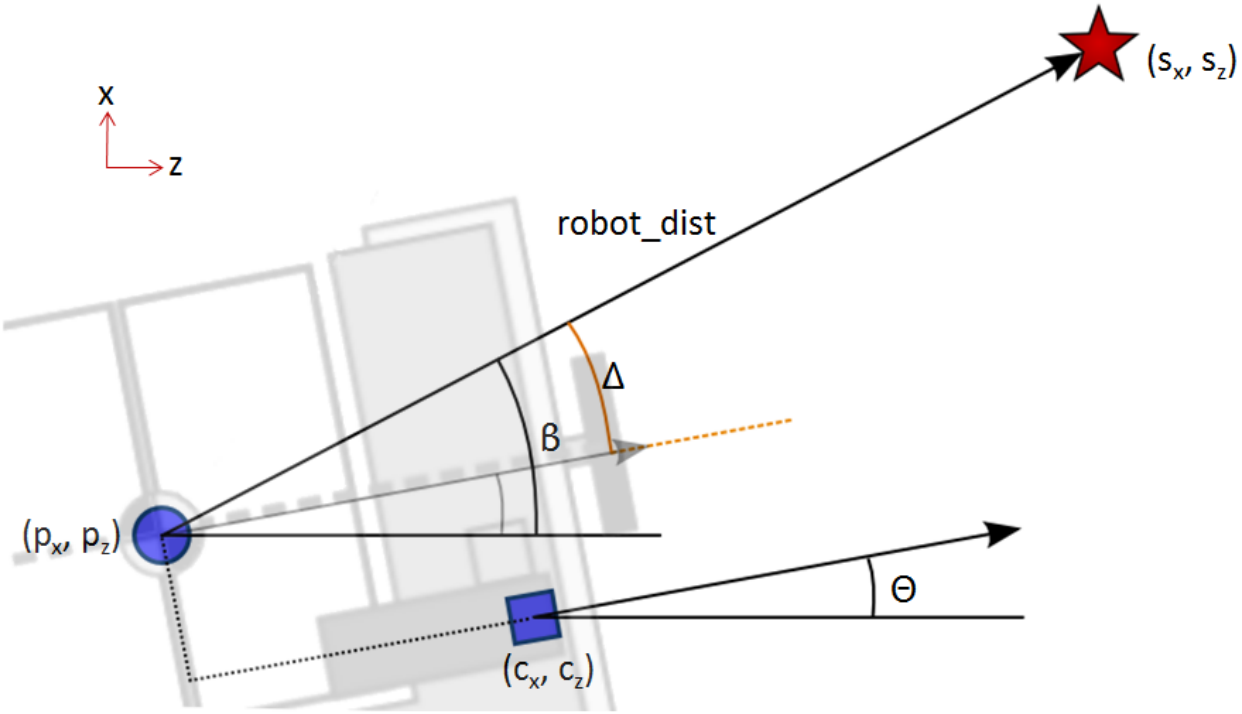


Figure 4.19: Angles for robot movement calculations.

At this point the two commands *robot_angle* and *robot_dist* needed to move to robot to collect the sample have been calculated and are ready to transmit. Now one additional check is

performed to determine if the object is "scoopable". Sample size limits are defined by the dimensions of the scoop as described in Table 4.4. Note that in the program the x and z (planar) limits are interchangeable, i.e. the object must be smaller than both limits but in either order.

Table 4.4: Scoopability limits for sample collection.

Direction	Maximum Dimension
x	8 or 10 cm
y	6 cm
z	10 or 8 cm

If the object to collect is reported to violate any of the maximum dimensions a warning message is displayed to the user. Mass is also calculated based upon the density of common materials and number of occupied voxels (Equation 4.13) and displayed in a table. For example tables please see the Results section.

$$m = \rho * V \tag{4.13}$$

Where:

m = Sample mass

ρ = density of a material

V = volume of filled voxels (each voxel = 0.125 cm^3)

All of the preceding calculations and checks are summarized in Algorithm 4.5 below.

Algorithm 4.5 Calculate sample properties and robot movement commands.

```
Load final camera position and view direction
Calculate location of center of robot rotation relative to final camera position (pos)
dist = distance from pos to sample object center of mass
angle = angle between pos and sample
robot_dist = distance(pos to object)
robot_angle = 90 + angle
xdim = xmax − xmin                    {Find sample dimensions from its 3D points}
ydim = ymax − ymin
zdim = zmax − zmin
if xdim, ydim, zdim < scoop dimension then
    Report object is scoopable
else
    Report object is too large to scoop
end if
Calculate volume from number of occupied voxels
Report mass=density*volume for common materials
```

4.4 Simulation of Sample Collection

Prior to implementing the autonomous sample collection routine on robotic hardware it was necessary to verify the operation of its software components. The image and point cloud processing software could be run independently using manually collected photo sets, but the robot commands could not be fully tested without creating a virtual model. Accordingly, a software simulation of sample collection was developed in Matlab. This simulation would also provide the user with an animation to check that the angle and distance the robot would travel to collect the sample were correct. Essentially it models the USL ground robot (including the offset center of rotation), final camera position and view direction, and sample location. The user can view and recall this animation from the main program GUI at any time after completing Steps 1-3 as shown in Figure 4.9. Once loaded, an animation window appears as in Figure 4.20.

Notice the robot icon, perimeter circle (dashed black line), final camera position (small blue circle), sample to collect (red X) and robot path (dashed blue line). If the robot commands

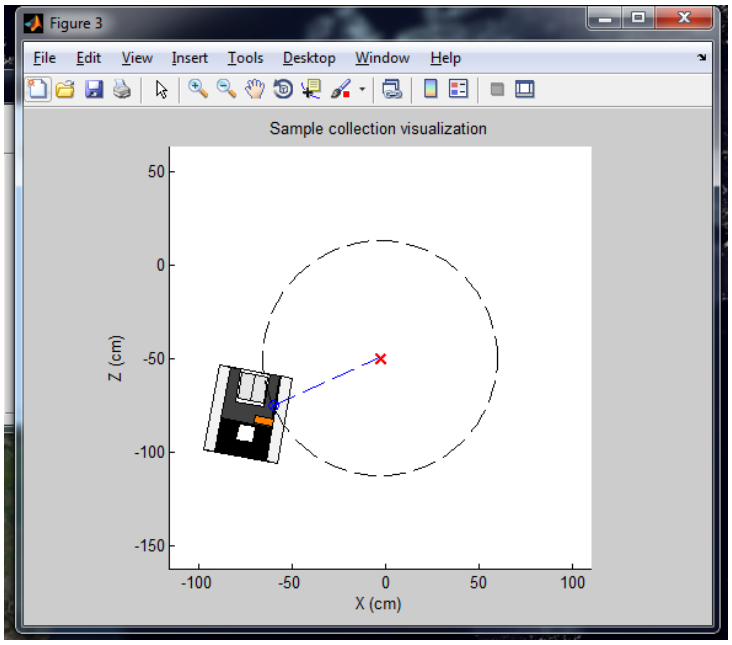


Figure 4.20: Sample collection preview animation window.

are computed incorrectly the blue line will not point directly toward the red X, and the robot path calculations and sample selection routines may need to be re-run. In the animation the robot will turn from its final position, proceed along the path, and stop when the sample is at the center of the scoop. Even the scooping motion is animated, right down to the disappearance of the red X upon collection!

The second simulation which was created is designed to determine what movement accuracy would be required of the robot platform for it to consistently perform sample collection (based upon the assumption that software side, i.e. 3D reconstruction, was accurate). Two factors were taken into account: a) the robot's ability to turn through a specific angle (yaw) and b) its ability to drive straight for a given distance (linear forward motion). These factors would be a function of the sensors and actuators used, so it was essential to select hardware which could perform adequately and to know what specifications had to be met to ensure this would be the case.

Each step of the sample collection process was modeled. First, the robot turns 90° from facing towards the area to reconstruct to facing tangentially to it. Next, the robot travels

a distance equal to the circumference of the reconstruction area divided by the number of photos, and then turns through an angle (7.5° for 48 photos) to take a picture. This process is repeated for each photo until a complete circle has been made, for a total of 47 driven photo segments plus the starting photo. The last step is to travel the calculated distance and angle to collect the sample (approximately 90° if the sample is near the center of the area).

Pseudorandom numbers with a normal distribution were generated by Matlab and this distribution was scaled to have a standard deviation equal to the angular or linear accuracy of the hardware. This would represent, for example, a GPS position that can only be known with a certain level of accuracy. It is also assumed that this would be a worst-case (least accurate) scenario for wheel encoders or other hardware. At each step this potential error was calculated and used to modify the distance and/or angle the robot traveled. Thus, as the simulated robot traveled around the circle, it could precess or otherwise wander from its ideal location. One run of the simulation with a distance accuracy of 1 cm and an angular accuracy of 1° can be seen in Figure 4.21.

The black circle shows the ideal path of the robot's center around the perimeter of the area to reconstruct, and the red lines linking the blue circles (locations where images were collected) show its actual path. In this particular run the robot begins by tracking the circle well, wanders off to the right side of it, and finishes close to it again. Sample collection was successful since the black scoop bounding box fully encloses the small circle at the center of the figure (representing a sample with diam. = 5 cm). All units are in centimeters. The offset between the center of the scoop and the center of the sample is also reported; if this value is greater than 2 cm then the scoop cannot fully enclose the sample and that run is considered a failure. Running the simulation 10 times in a row produces a plot such as Figure 4.22, with a median offset of 1.45 cm, 8 successes, and 2 failures (80% success rate).

The USL ground robot could be equipped with quadrature encoders (128 counts per revolution) along with a high-end IMU with ±0.5° orientation accuracy. The encoder accuracy is

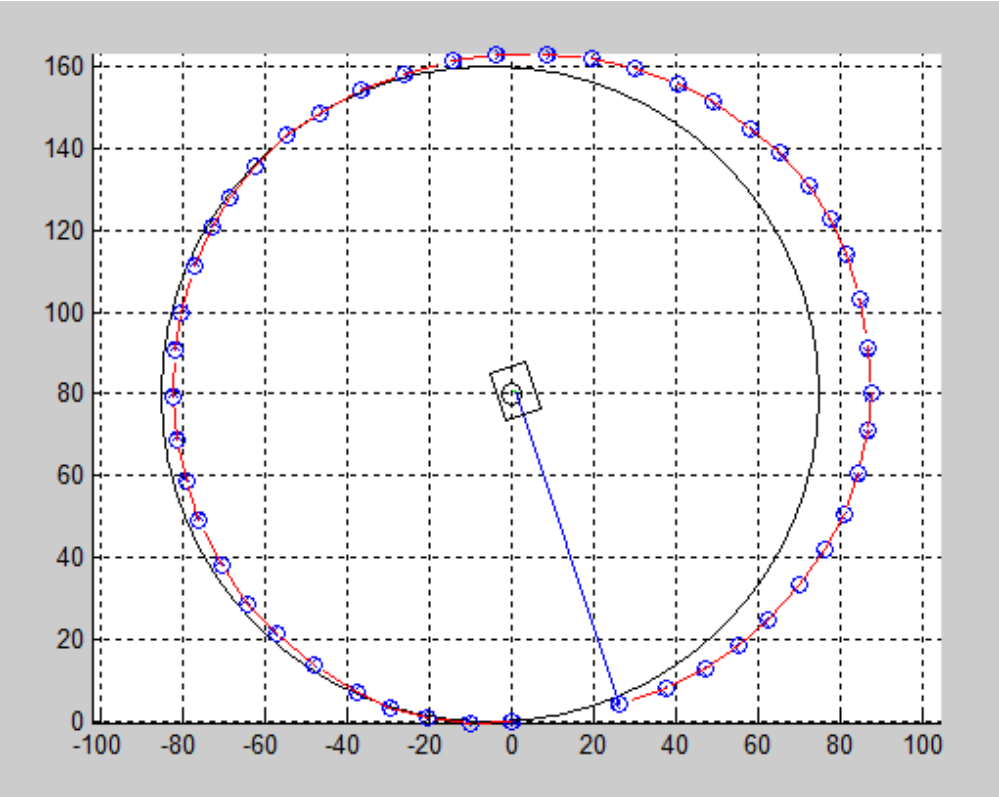


Figure 4.21: One run of simulation of accuracy of reference robot movement.

translated into a distance accuracy (in cm) using Equation 4.14.

$$\sigma = \frac{\Phi_w * \pi}{n} \tag{4.14}$$

Where:

σ = distance accuracy in centimeters

Φ_w = wheel diameter in centimeters

n = number of encoder counts/steps per revolution

For the USL ground robot $\Phi_w = 13$ cm and $n = 128$, so $\sigma = 0.32$ cm. The motors which were available for the Scoopbot that was used to test autonomous sample collection were stepper motors with 5.1 cm wheels and 1600 counts per revolution (using 1/8 microstepping), so σ

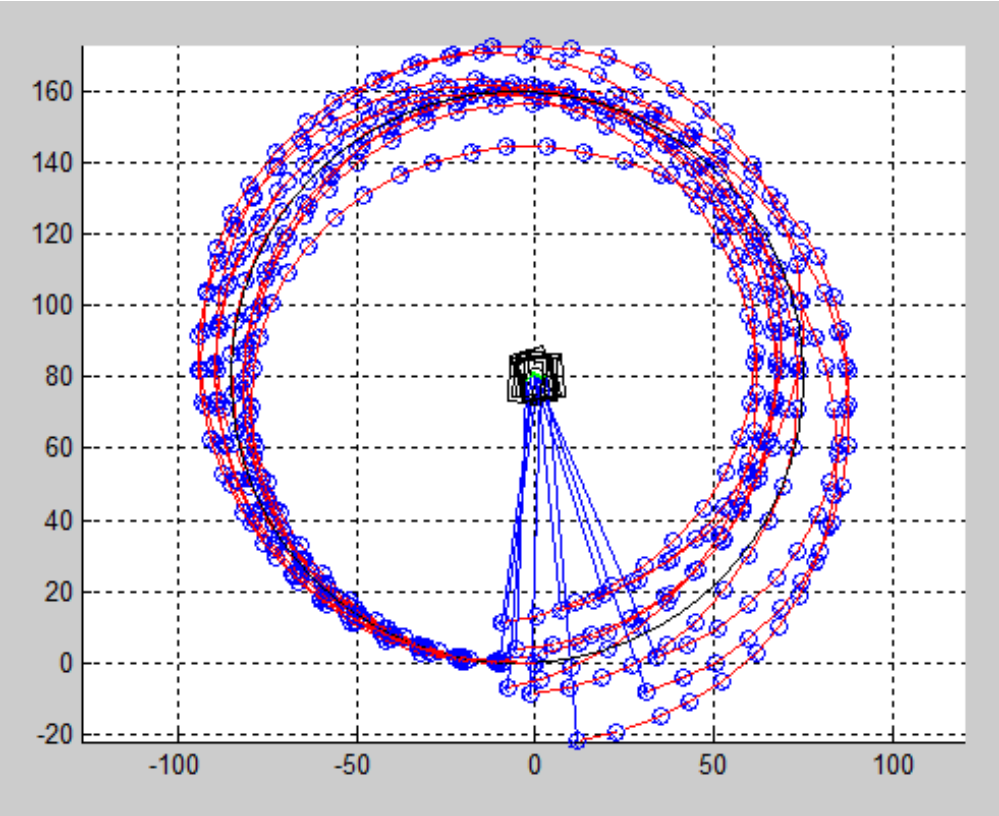


Figure 4.22: Ten runs of simulation of accuracy of reference robot movement.

= 0.01 cm. Angular accuracy for the Scoopbot was determined using σ and trigonometric relationships and found to be equal to 0.04°. Bear in mind that these are theoretical best performance values achievable only with ideal motors and under perfect no slip conditions. Simulations of 10,000 trials for the ground robot and Scoopbot are compared to a reference case (1 cm, 1°) in Table 4.5.

Table 4.5: Simulation of sample collection success using different robotic platforms.

Platform	**(cm)**	**angle (deg)**	**Successes**	**Failures**	**% success**	**# of trials**
USL robot	0.32	0.5	9950	50	99.5	10000
Scoopbot	0.01	0.04	10000	0	100	10000
Reference	1	1	7492	2508	74.9	10000

As the table shows, both the USL robot and the Scoopbot perform with better than 99% success rates. This indicates that using either robotic platform to test autonomous sample

collection will not produce any hardware constraints on the system's overall accuracy. As would be expected, a plot of 10 simulation runs with the Scoopbot parameters (Figure 4.23) looks much better than the previous one shown in Figure 4.22, as all runs trace out the perimeter circle almost exactly.

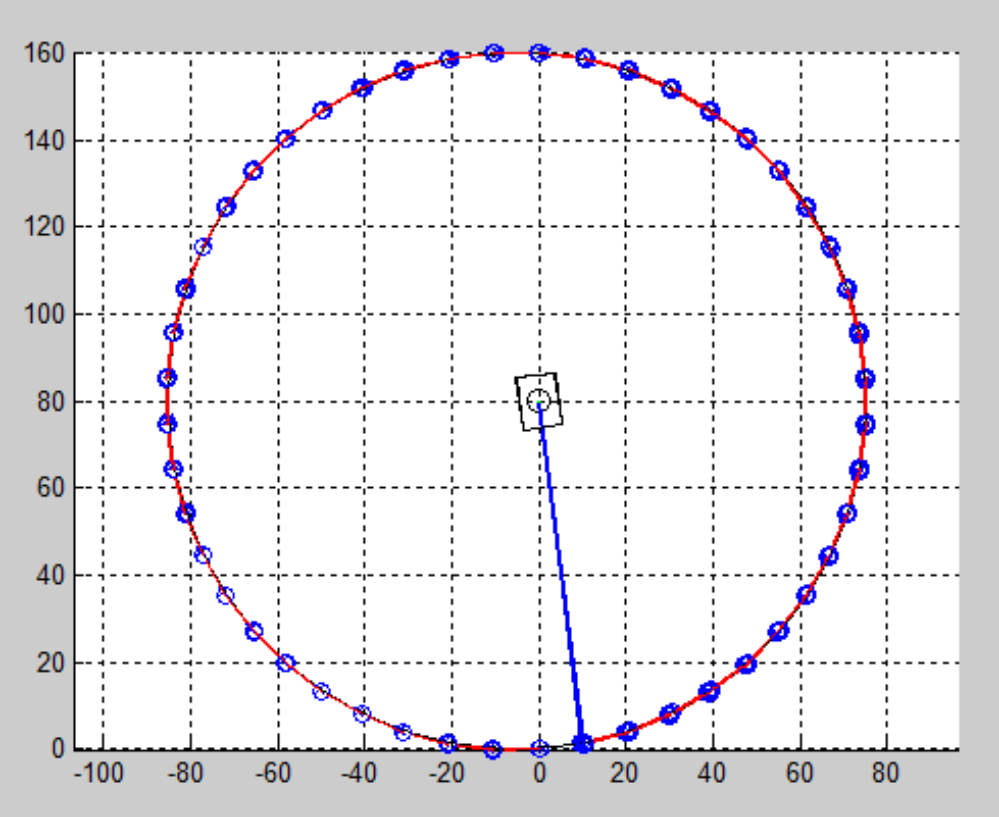


Figure 4.23: Ten runs of simulation of accuracy of Scoopbot movement.

Since both simulations indicated that autonomous sample collection using image-based 3D reconstructions should be achievable, the Scoopbot platform was constructed to prove the concept in real-world testing.

4.5 Implementation using Scooping Robot

Originally the autonomous sample collection technique, like the scoop itself, was intended to be deployed on the USL ground robot. Prior to full implementation, however, flaws developed in the ground robot's electronics which remained unrepaired and prevented deploying the system on that platform. This led to the creation of a custom "scooping robot" or "Scoopbot" mobility platform designed specifically to accommodate the scoop sample collection system and camera required to perform 3D reconstruction.

4.5.1 Scoopbot Design

The Scoopbot provides most of the same functionality as the USL ground robot but some kinematic simplifications were made to streamline its control. The ground robot had two treads for enhanced mobility, but this also meant that it was in a skid-steer configuration i.e., sensing feedback for turning would be required. The Scoopbot relies instead on two forward wheels for turning and propulsion and a rear omnidirectional ball caster for support. This means that moving either wheel will, in theory, move the entire robot (with no slip). Thus, any command given to the wheels can be considered to move the whole platform even if no feedback is provided. The propulsion system and other principal Scoopbot components are illustrated in Figure 4.24.

The Scoopbot platform is approx. 36 cm (l) x 28 cm (w) x 4.5 cm (h), making it somewhat smaller than the USL ground robot. The wheelbase is a 30 cm span between the two 5.1 cm diameter drive wheels. Each drive wheel is directly attached to the axle of a NEMA 14 size stepper motor with 200 full steps per revolution (1.8° step angle) and 10 N-cm of torque. The entire Scoopbot, including camera and batteries, has a mass of 2.15 kg.

All onboard processing and control is handled by an Atmel ATMega 328 microprocessor with an Arduino bootloader. Commands are transmitted wirelessly to the robot over a pair of 2.4 GHz Xbee radios (one on the robot and one attached to the computer base station). The

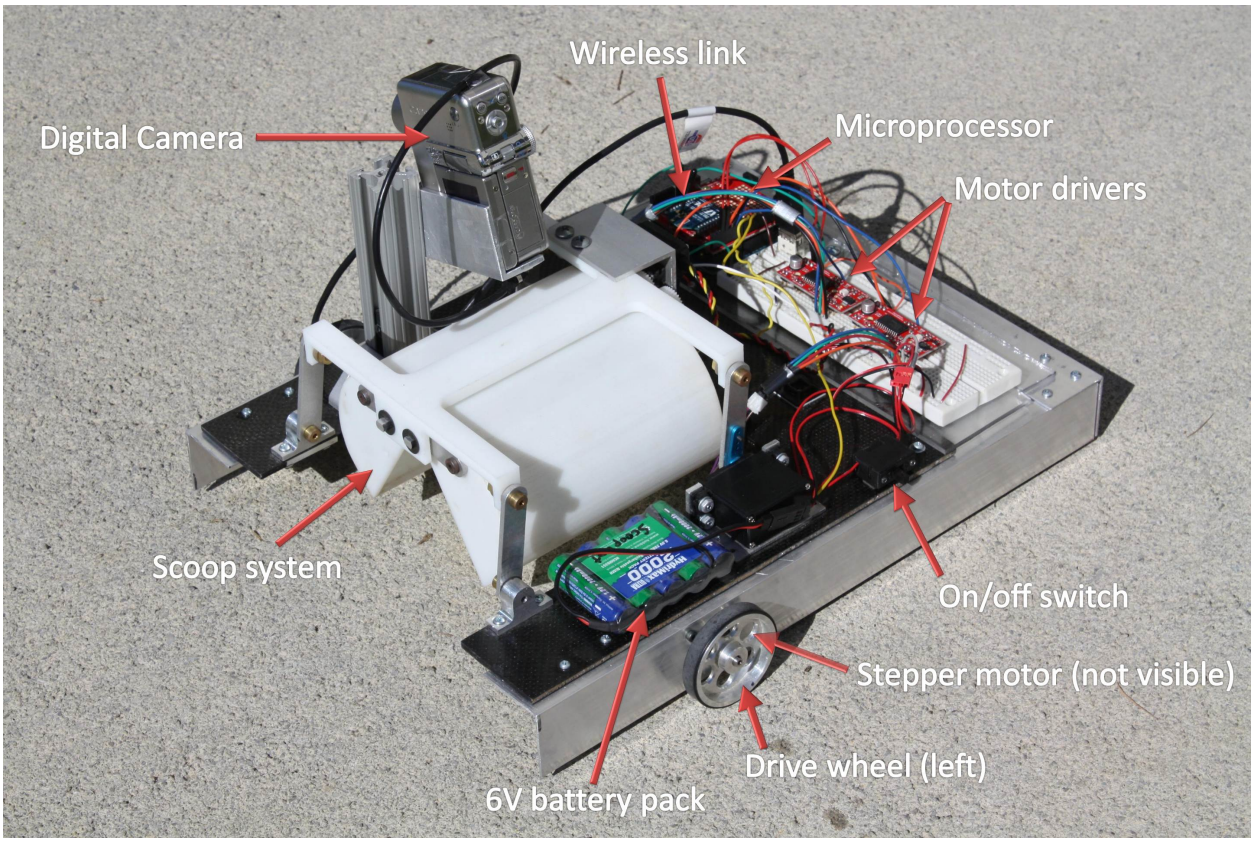


Figure 4.24: Diagram of principal Scoopbot components.

digital camera's (Canon Powershot TX1) shutter is remotely triggered by the Arduino over a USB connection, and images are transmitted by a special SD flash memory card back to the base station over a WiFi network. As the pictures are taken and transmitted the base station automatically saves them into the image source directory for 3D reconstruction.

Since the Scoopbot incorporates a variety of electrical components and moves independently an onboard power source was required to run it. This system power is provided by a 6.0V 2000 milliamp-hour (mAh) nickel-metal-hydride (NiMH) battery. It provides power to all equipment with the exception of the digital camera which contains its own separate battery. All components run at the native battery voltage excluding the USB trigger line to the camera which uses the Arduino's regulated 5V. Power to the various Scoopbot components is distributed as shown in Figure 4.25.

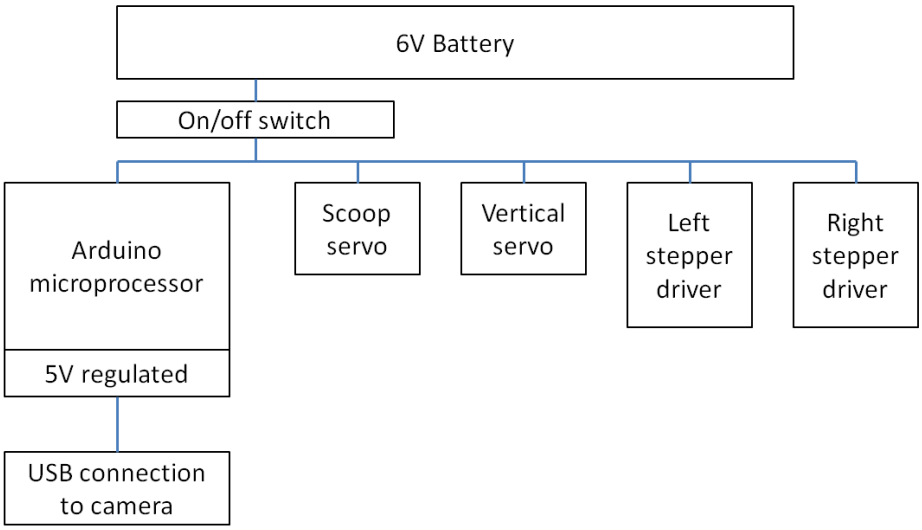


Figure 4.25: Power distribution to Scoopbot component hardware.

Recall from Figures 4.17 and 4.24 that the digital (still) camera is mounted perpendicular to the robot's direction of travel. There is a very good reason for this configuration! If the camera were forward-facing it would be fine for collecting images while initially approaching the sample collection area, but it would be very difficult to reorient it to capture images around the perimeter. A forward-facing camera would require the robot to take a picture, rotate 90°, drive along the circle, and rotate 90° the opposite direction just to get the camera facing inward again. Instead, the perpendicular configuration only requires one 90° turn initially to rotate the camera towards the center of the area. Then, as the robot drives along the perimeter, it only needs to turn through α degrees (see Equation 4.1) at each interval to keep the camera pointing at the center. Only one final larger turn (e.g. 80-100°) is necessary following image acquisition when the sample is actually collected. Mounting the camera perpendicular and the resulting shorter drives means that less error will be introduced from any wheel slippage or surface irregularities that impede the robot's motion. Overall, this camera configuration proved to be very effective and would be suggested for the USL ground robot as well.

4.5.2 Scoopbot performance testing

Due to the use of open-loop control to drive the Scoopbot it was necessary to determine whether the calculated accuracies would translate directly into real-world distances and angles as predicted by using Equation 4.14. Performance testing beforehand would ensure that when the robot was given a command (assuming no slip at the wheel-ground interface) it executed its maneuvers with sufficient precision to perform an accurate 3D reconstruction or collect the sample.

The performance testing setup was as follows: two 60 cm x 60 cm large sheets of paper were printed with lines, one for distance testing and one for angular (turning) testing. The distance sheet had major gridlines every 1 cm and minor gridlines every 1 mm. Similarly, the angle sheet had major gridlines every 10° and minor gridlines every 1° (polar grid configuration). Both sheets of paper were taped down to a smooth, flat surface and checked for wrinkles which might impede motion. The test setup is shown in Figure 4.26.

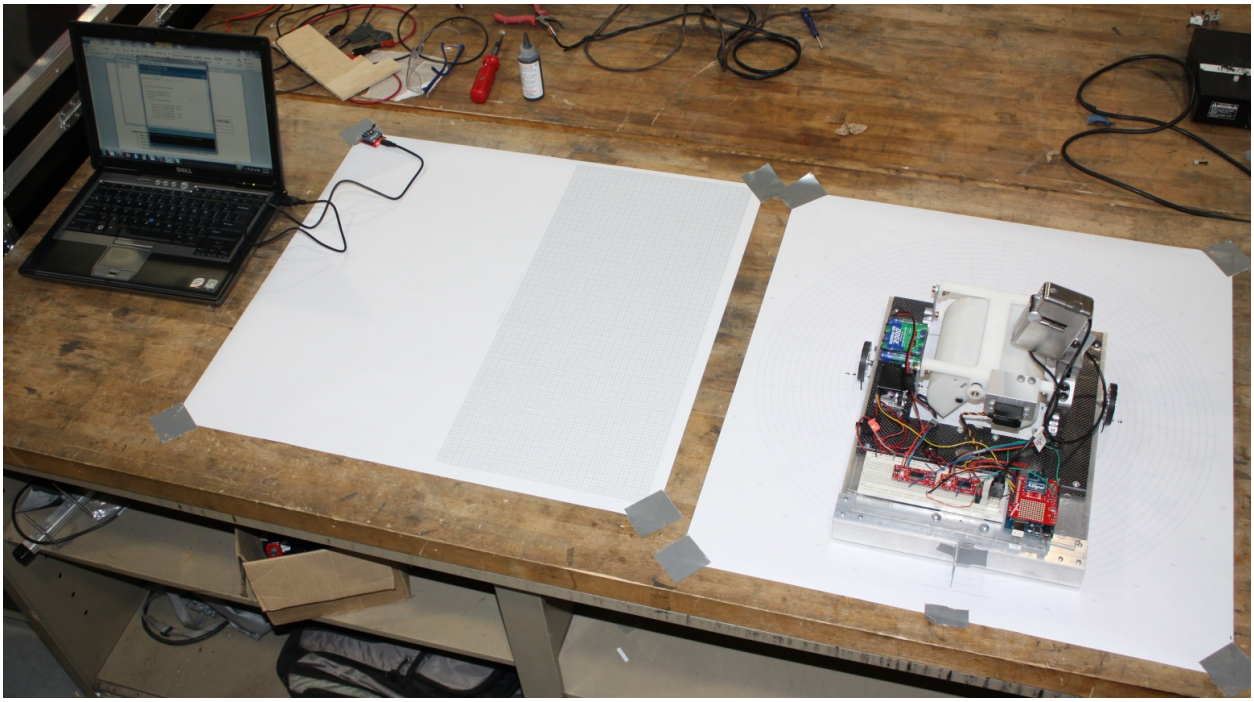


Figure 4.26: Setup for Scoopbot performance testing. Distance grid is on left and angle grid on right.

To run a test the robot was placed at a known location on one of the sheets and a distance or angle command was transmitted wirelessly to it. After the robot had completed its movement the resulting distance/angle was recorded. Several runs of different distances and angles were completed to provide sufficient data for analysis.

The initial calculations for distance driving showed that 1 step on the motors ($1/1600^{th}$ of a revolution) should cover 0.01 cm. Therefore, commands were sent to travel distances of 10 cm and 20 cm, or 1000 and 2000 steps (10 cm is the approximate distance between subsequent robot positions around the perimeter circle with 48 images). The average distances traveled were 10.16 cm (for 100 steps, 5 trials) and 20.3 cm (for 200 steps, 2 trials). After analyzing these results it appeared that the calculated number of centimeters per step was a slight overestimate, so the next tests were performed with 1 step = 0.00985 cm. Distances of 10 and 50 cm (985 and 4,925 steps) were driven in these trials. The results were the robot traveled an average of exactly 10.0 cm (2 trials) and 49.77 cm (3 trials). Since this performance was very close to ideal no further refinements were made.

Angular performance testing was done in a similar fashion. Calculations showed that there should be approx. 26 steps for every degree turned. Angles of 90°, 360°, 7.5°, and 10° were used (7.5° = α for 48 images). The average angle turned and number of steps for each value are shown in Table 4.6.

Table 4.6: Results of angle performance testing for Scoopbot.

Goal (angle)	**Goal (steps)**	**Actual (avg. angle)**
90°	2344	90.1°
360°	9376	360.8°
7.5°	195	7.44°
10°	260	10.0°

The angle performance testing produced accurate and consistent results so no modifications were made to any parameters for Scoopbot turning (note that a 29.85 cm wheelbase instead of 30 cm was used for the tests, however, since it was a more accurate measurement of the wheel-to-wheel distance). For a more complete description of the testing procedures and

data the reader is referred to the Appendix.

4.6 Results

Once all of the performance testing had been completed automated 3D reconstruction and sample collection were ready to be undertaken. The goal was to validate use of the entire system in the cases of no (fully autonomous) and selection (semi-autonomous) user input. Ultimately two modes of picture-taking operation were developed, a "full mode" in which images are acquired completely surrounding an area, and a "fast mode" in which only a few images are collected, producing a partial object model/reconstruction but greatly speeding the entire process. In both cases, tests began with the robot approx. 60 cm from the target object and facing towards the area to reconstruct. All software was run on the base station computer with a six core 3 GHz processor and 8 GB of RAM.

4.6.1 Full Mode Reconstruction Tests

The first set of reconstruction tests employed the comprehensive system concept as explained in previous sections. First, a command was transmitted wirelessly from the base station to the robot to begin the picture-taking process. Images were uploaded to the computer as they were taken, and once all 48 (forming a complete circle around the area to reconstruct) were received the 3D reconstruction pipeline and processing began. Then either the user selected the sample or it was identified automatically and a command was sent to the robot to collect it. This process with 48 images completely surrounding the object is termed "full mode".

Full mode tests were performed using the circle taped out on the concrete floor of the Unmanned Systems Laboratory as the approximate area to reconstruct. An orange rock (see Figure 4.27) was placed near the center of that area to be the target object to collect

Figure 4.27: Orange rock used as sample for collection tests.

(parameters in the autonomous collection algorithm were also set up to look for it). In these tests the robot traveled around the 1.2m diameter circle and took 48 pictures which showed a complete 360° view of the scene. This test setup is illustrated in Figure 4.28.

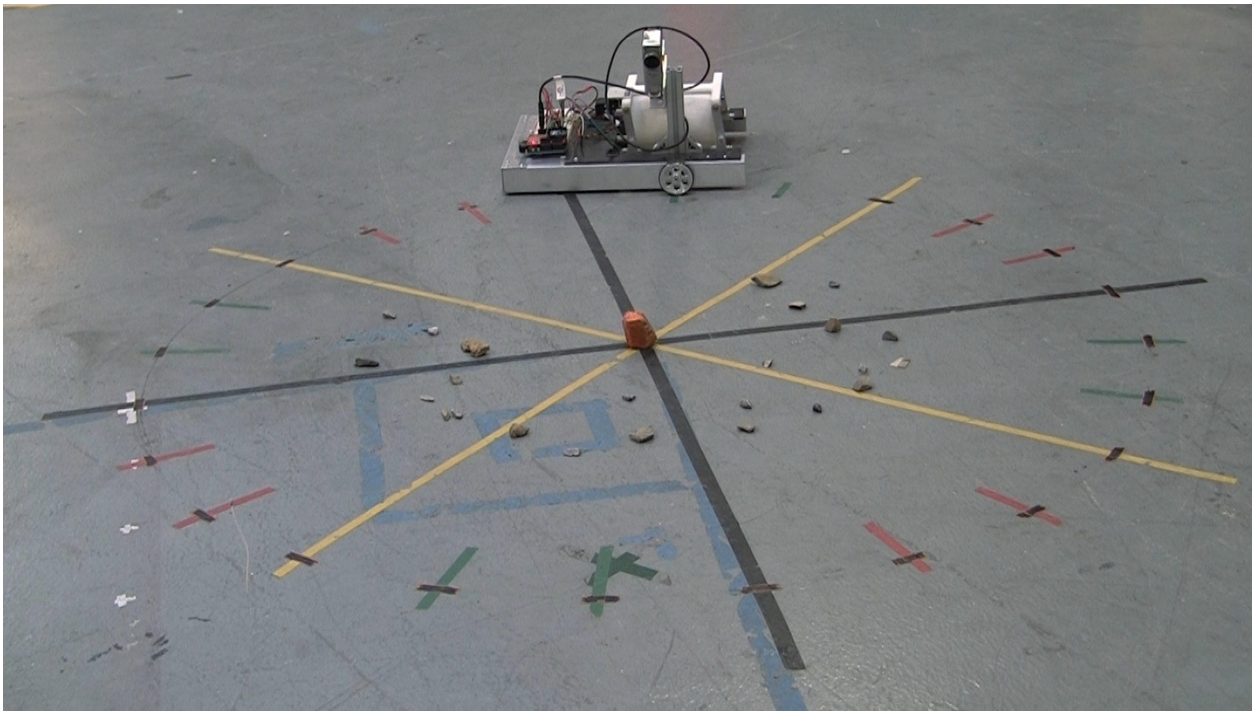


Figure 4.28: Setup for full mode sample collection tests.

Notice that some additional rocks have been scattered around the test area. These rocks were used to create a more realistic test environment and also to provide additional feature points for the 3D reconstruction.

Initially two full mode trial runs were completed but the robot was unable to successfully travel to the sample because the angle computed for the final turn was not quite correct, being off by approx. 3.5° each time (possibly due to the SfM optimization routine). After incorporating this fixed offset into the final angle calculations for full mode two successful runs of the system were completed. In the first successful trial run the orange rock was manually identified by the operator from the 3D plots, and in the second test the autonomous algorithm was used. Table 4.7 shows the timing breakdown of the fully autonomous sample collection trial, and Table 4.8 summarizes the results of the run. Results from the manual selection trial were also comparable.

Table 4.7: Timing results from full mode, fully autonomous collection trial run (48 images).

Operation	**Image Acquisition (robot)**	**SfM**	**PMVS**	**Matlab Processing**	**Collection (robot)**
Time Required (m:ss)	5:30	4:16	2:33	1:38	0:35
Percent of total	38%	29%	18%	11%	4%

Table 4.8: Summary data for full mode trial run.

Total time	**SfM Points**	**PMVS Points**	**Trimmed Points**	**Collection?**
14:32	3054	34904	26525	Yes

As previously stated, sample size and mass properties were also automatically calculated from the scaled 3D reconstruction model and displayed to the user. These two tables are shown here: Table 4.9 for dimensions and Table 4.10 for masses.

Due to variations in composition no mass estimate is provided for stone/rock, but the actual mass of the orange rock was measured to be 103 g, which is reasonably close to the mass value for cement. Furthermore, the sample volume (from filled voxels) was reported as 39.875 cm^3. This is 99.7% of its measured value of 40 cm^3, an incredibly accurate volume estimate!

Several things are apparent when reviewing the results of full mode sample collection. First, the overall procedure was successful - the robot could autonomously gather all of the required

Table 4.9: Reported versus actual sample dimensions.

Dimension	Reported	Actual*	Scale
x	4.2 cm	4.4 cm	95%
y	3.8 cm	4.0 cm	
z	3.8 cm	4.0 cm	

*actual [approximate] values were determined by estimating the orange rock's orientation relative to XYZ axes when measuring with a caliper

Table 4.10: Reported sample masses for common materials.

Material	Mass
Aluminum	107.7g
Plastic (ABS)	41.5g
Portland Cement	91.7g
Steel	313.0g

images and pick up the sample by itself. Second, the volume and dimensions reported for the sample were quite accurate. Third, however, is the realization that the process required a fair amount of time to complete. A major factor in this was the slow speed of the robot, but the processing of the large number of images required substantial time as well (nearly 50% of the total time). These two time-consuming steps inspired the creation of the "fast mode" collection process described in the following section.

4.6.2 Fast Mode Reconstruction Tests

The "fast mode" reconstructions were thusly named because they were created using only a fraction of the number of images of a "full mode" reconstruction. This meant that the robot spent less time gathering the images and that the base station computer spent less time processing them as well, greatly speeding the entire collection process. The software pipeline remained the same as in full mode; however, the reported size and volume of the sample were smaller than their true values. Since the fast mode image set subtended approx. 45-60° of the perimeter circle usually only one side of the sample was seen by the camera.

Consequently, only the parts of the sample visible in the pictures could be reconstructed and modeled in 3D (this did not adversely affect the autonomous collection algorithm though).

Initial fast mode tests were conducted indoors on a smooth tile floor using a set of six images taken consecutively around 1/8th of the circumference of the perimeter circle. Several trial runs of these tests were performed, with an 80% success rate for sample collection (4 out of 5 tries, one reconstruction was poorly scaled). Figure 4.30 shows the test setup and Table 4.11 and Table 4.12 provide additional data describing one of the successful collection runs. The average time from start to finish for these trials was 2 minutes and 45 seconds.

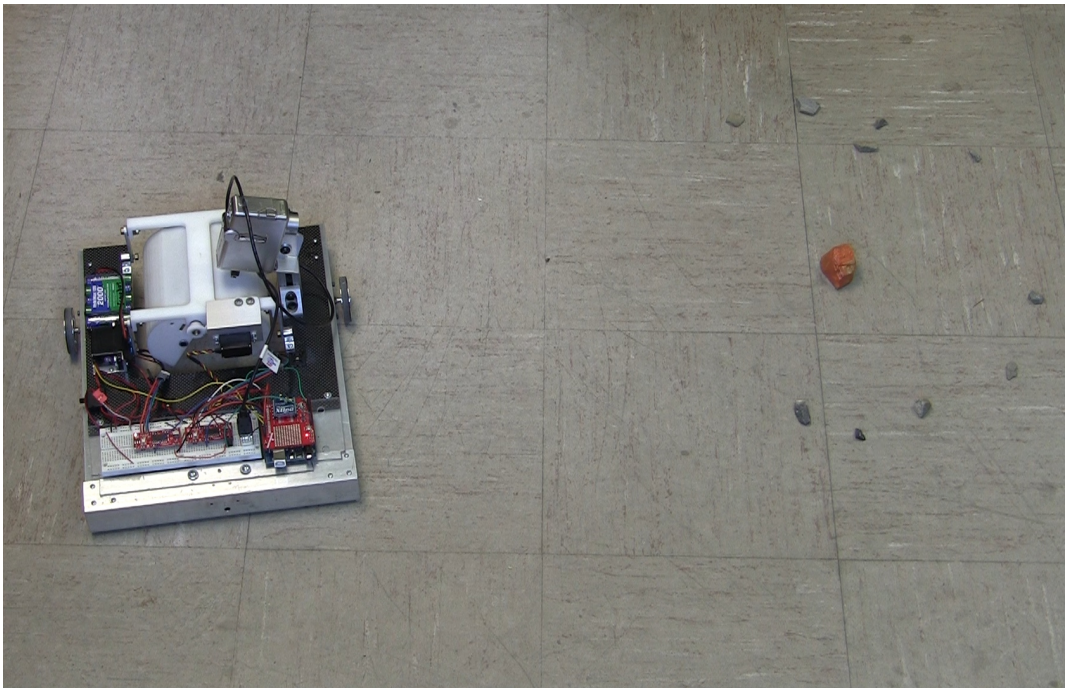


Figure 4.29: Setup for fast mode indoor sample collection tests.

Table 4.11: Timing results from fast mode indoor collection trial run (6 images).

Operation	**Image Acquisition (robot)**	**SfM**	**PMVS**	**Matlab Processing**	**Collection (robot)**
Time Required (m:ss)	1:00	0:23	0:20	0:31	0:36
Percent of total	35%	14%	12%	18%	21%

Table 4.12: Summary data for fast mode indoor trial run.

Total time	SfM Points	PMVS Points	Trimmed Points	Collection?
2:50	154	8137	6907	Yes

Once indoor testing was completed one final set of tests was performed to validate the system in an outdoor environment. These tests also employed fast mode picture taking and processing, but it was found that using only six input images produced poorly scaled reconstructions, so the robot would travel too far past the sample to pick it up. Therefore the number of images was increased to eight (1/6th perimeter circle) for outdoor tests. The outdoor tests were conducted on a worn, angled concrete surface and again the orange rock was used as a sample. An image of the outdoor test environment is visible in Figure 4.30.

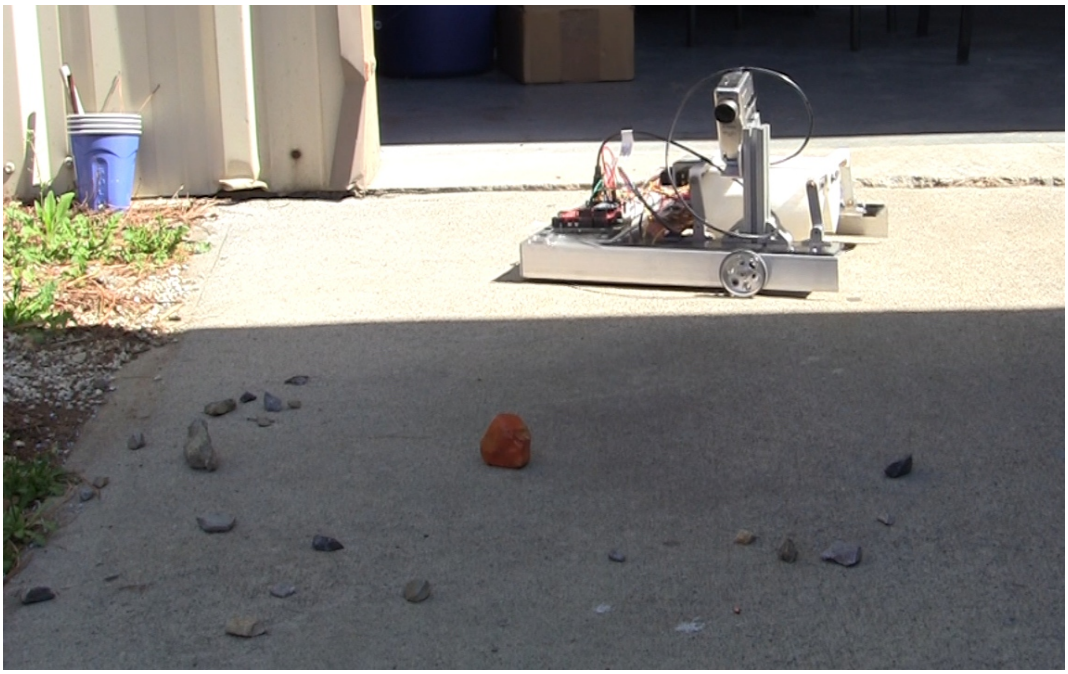


Figure 4.30: Setup for fast mode outdoor sample collection tests.

The Scoopbot robot was able to successfully locate and collect the sample autonomously in this test as well. Detailed data for the outdoor fast mode test is shown in Table 4.13 and Table 4.14. The entire procedure using 8 images required 3 minutes and 27 seconds.

Table 4.13: Timing results from fast mode outdoor trial run (8 images).

Operation	**Image Acquisition (robot)**	**SfM**	**PMVS**	**Matlab Processing**	**Collection (robot)**
Time Required (m:ss)	1:18	0:43	0:20	0:30	0:36
Percent of total	38%	21%	10%	14%	17%

Table 4.14: Summary data for fast mode outdoor trial run.

Total time	**SfM Points**	**PMVS Points**	**Trimmed Points**	**Collection?**
3:27	883	10714	8929	Yes

Overall, (semi-)autonomous sample collection was successfully demonstrated in all modes. Fast mode testing required only 19% and 24% of the time of full mode for indoor and outdoor tests respectively. Additionally, fast mode could be more useful than full mode if the robot were unable to fully traverse the sample collection area (for example, if an obstacle blocked part of its path). Taken together, these tests validate the use of image-based 3D reconstructions for autonomous sample collection using a mobile robot.

Chapter 5

Automatic Building Crack Detection

Other avenues for the use of the 3D point clouds generated by image-based reconstructions were explored in addition to the autonomous sampling applications discussed in Chapter 4. This chapter delves into the creation of an automatic building crack detection algorithm which is derived from the 3D models. Background, preliminary tests, and full-scale testing results with an actual building element are described, in addition to an explanation of the details of the crack detection algorithm (CDA) itself. Furthermore, some thought is given to the potential lifesaving applications of the CDA when used in conjunction with mobile robots following a disaster.

5.1 Concept of Operations

5.1.1 Background and Application of Mobile Robots

Natural and man-made disasters are becoming an increasing concern as the population of urban areas continues to grow. In dense urban environments these disasters can cause extensive structural damage to a large number of homes, offices, and public buildings. Human lives are at great risk in the initial disaster; nonetheless, search and rescue efforts afterwards (and

later, damage assessment and reoccupation) can still be perilous due to uncertain knowledge of building stability. If the information required to assess stability could be gathered and processed remotely, early responders and engineers could more effectively respond, perform detailed assessments, and better plan for rescue and recovery in the affected areas. Here photography represents one quick and effective method of collecting data about a building's condition. Pictures collected in the proper fashion can be used to create a 3D model of a damaged structure. It may be safe for people to walk around and take these images from the exterior of buildings; however, examining interior supporting elements such as columns and beams would require entering into a compromised building and potentially put a responder's life at risk.

To address these challenges, this chapter presents a new automated post-disaster building assessment that could be carried out by a small mobile robot. The operator would remain safely outside the structure while the robot, equipped with a digital camera and wireless controls, enters the building to gather the images needed to reconstruct a 3D model. Once a 3D point cloud model is reconstructed, the output would be assessed using a new algorithm to identify and color-code the locations of cracks. The resulting models enable the geometrical characteristics of the cracks to be quickly and easily measured and ultimately lead to an accurate assessment of all building elements. Details of how this system could be deployed are explained in the motivating scenario that follows.

5.1.2 Motivating Scenario

After a tremor in an earthquake-prone region of the U.S. first responders are dispatched to search for survivors. Once they complete their initial searches in the hours following the quake, surveying and recovery efforts begin, and engineers and building inspectors start to examine the integrity of potentially damaged structures. A building which shows some visible cracking and loose debris on the exterior is declared unsafe to enter for a full assessment. Inspection teams equipped with a small ground robot (with a high resolution camera) arrive

on the scene and set up at a safe distance from the damaged structure. They remotely navigate the robot into the building and locate a vital structural support (e.g., a concrete column on the first floor of an office building). Once the entire column is visible to the robot's camera an autonomous algorithm drives it around the target, collecting a series of 40-50 images as it moves. These images are transmitted back to the base station where a powerful computer begins to construct a detailed 3D model. The robot is then driven to the next structural element, and the process is repeated.

Once the images have been collected and 3D models generated, they are analyzed with a crack detection algorithm such as the one presented herein. The inspectors then examine the color-coded models, assessing the extent of the damage, and determine if the building is safe for reoccupation or a more detailed human investigation.

This robot-assisted process could also help complete the "Rapid Building and Site Condition Assessment" form provided by the National Center for Preservation Technology and Training (NCPTT, form also used by the Federal Emergency Management Agency) [28]. The form lists a variety of basic information regarding the inspection date and time, type of structure, amount of damage, etc. For instance, in the "Evaluation" section, the robot-based reconstruction would be ideal for identifying and quantifying a "Collapsed or off foundation", "Leaning, other structural damage", and "Foundation damage", to name a few areas of use. It could also easily deliver a detailed structural evaluation if it were recommended in the "Further Actions" subsection, and provide an estimate of the overall building damage. This NCPTT form is referenced here simply to illustrate how robotic assessment could be integrated into existing emergency procedures.

5.2 Process Pipeline and Detection Algorithm

5.2.1 3D Point Cloud Creation and Processing

The 3D mesh model used by the crack detection algorithm (CDA) was generated using a semi-automated pipeline (all steps except trimming are automated). Figure 5.1 illustrates the image-based 3D reconstruction pipeline used in this work:

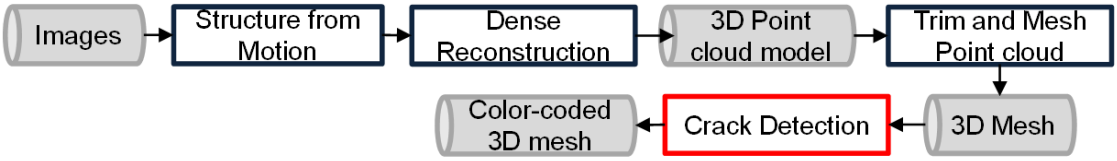


Figure 5.1: Pipeline used to process images.

First, a series of approximately 50 images is collected from locations equally spaced around the target object to reconstruct. These images are fed into the previously mentioned Structure-from-Motion algorithm [16], which was modified to use the SIFTGPU algorithm [29] for rapid feature point detection. The outputs go into a Clustering Multi-View Stereo algorithm (an extension of PMVS) [30] to generate a dense 3D point cloud model. Next, the point cloud is trimmed and using the Poisson surface reconstruction approach [31], and a colored mesh model is created. Once the colored 3D mesh model is created, it is rotated so that the object's sides are aligned with the XYZ axes, and the Y direction is along the long axis of the object. At this point the main new work presented in this chapter (represented in Figure 5.1 by the box labeled "Crack Detection") begins.

5.2.2 Automatic Crack Detection Algorithm

The new crack detection algorithm (CDA) is derived from a very basic principle: if an element of a building (e.g., a column) is undamaged, its surface normals should be perpendicular to the element's axial direction. An undamaged element can have any consistent cross-section

(even circular) and this principal will still hold true. Inversely, if an element is damaged, then some of its surface normals should not be perpendicular to the element's axial direction.

These criteria provide the basis for creating the CDA. The Poisson mesh creates hundreds of thousands of individual triangular mesh elements (depending upon its settings), and the building element's axial direction is aligned with the Y-axis. By examining the surface normal for every mesh element, its orientation relative to the axial direction can be computed. If the normal is within a threshold angle of perpendicular the element is considered undamaged (not cracked), but if it is far from perpendicular the element is considered to be damaged (cracked) and marked as such. Functionally, what this will do is identify all Poisson mesh elements whose surface normals point more up or down than directly outward from the building element. A diagram of the normals comparison is shown in Figure 5.2.

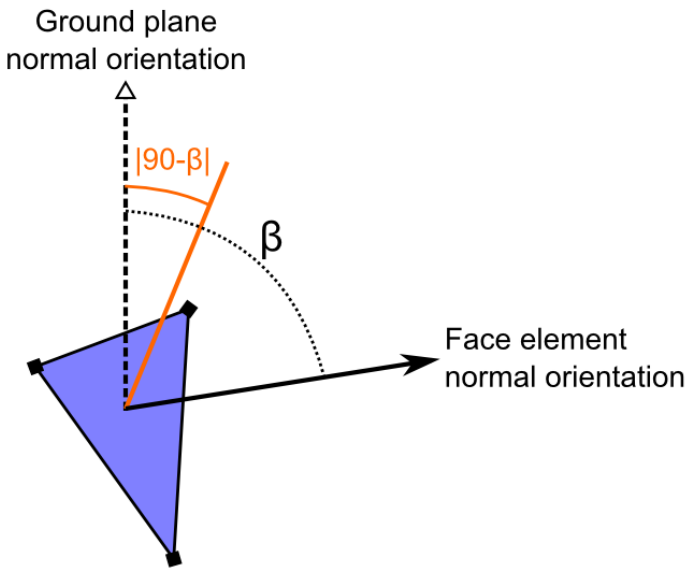


Figure 5.2: Demonstration of the normals angle comparison criteria for crack detection. The building element's axial direction is along ground plane normal.

In the figure the angle β represents the difference between the axial direction and the normal of a given surface mesh element. Taking the absolute value of 90-β creates a tolerance for slight deviation either up or down from perfectly perpendicular. If $|90 - \beta|$ is greater than a

threshold angle (e.g., 15°) then the element does not align well with the rest of the column and is considered part of a crack. The pseudo code for the CDA is presented in Algorithm 5.1 below.

Algorithm 5.1 Pseudocode for automatic crack detection algorithm

```
Load all mesh face elements (3, 3D points per element)
Examine a face element
Compute its surface normal (from 2 vectors connecting the 3 points)
Compare its normal to the ground plane/axial direction (angle )
if |90 − β| > threshold (e.g. 15°) then
    Mark as cracked (tint red)
else
    Mark as uncracked (tint green)
end if
Move to next face element
Save all tinted elements
```

Color-coding enables rapid crack identification when the output model is viewed. In the proposed method, red tinting is used to indicate detected cracks whereas green tinting is used to indicate uncracked areas. In practice this clearly demarcates the cracked regions, as can be seen in Figure 5.3 and the figures in the results section.

Another type of detection that is used but was not listed in the pseudocode is a determination of surface completeness. When a Poisson mesh model is created, all of its points are by default set to pure white. A simple closest point heuristic is used to transfer the colors from the dense point cloud to the mesh model. Any mesh vertices remaining pure white (RGB 255,255,255) after the texture-mapping do not have a corresponding point in the original point cloud. They are artifacts of the Poisson reconstruction. Therefore, the area associated with face elements containing these points can be computed and marked as "incomplete". Comparing this value to the total area provides a percentage of surface completeness. This is an important parameter to report, since if the physical surface is not completely reconstructed, some cracks or structural defects existing in reality may not be automatically detected and identified from the incomplete model.

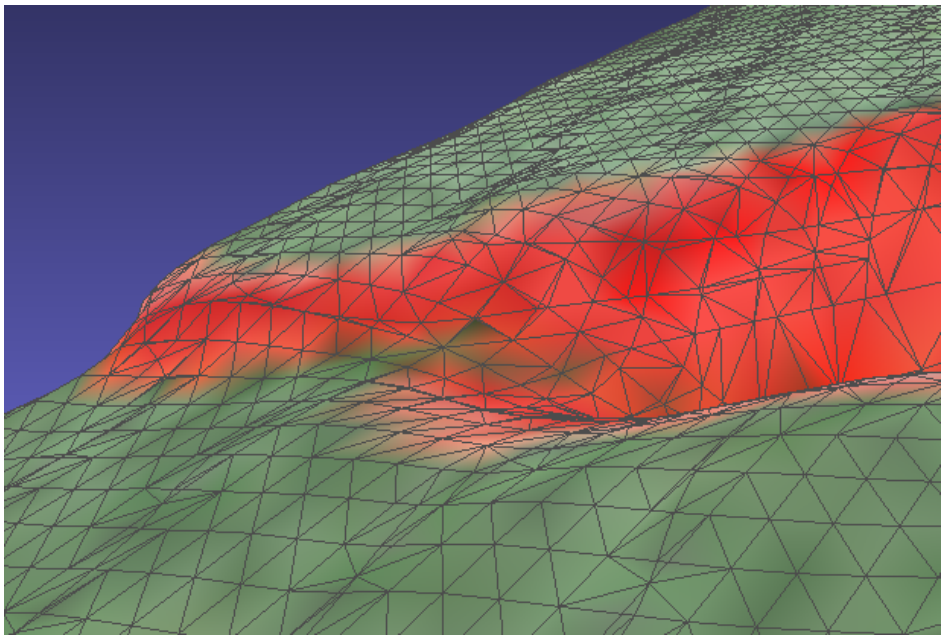


Figure 5.3: Close-up view of the color-coded 3D mesh model, showing individual mesh elements (triangles) and a cracked area clearly indicated with reddish hues.

5.3 Testing

A series of tests were performed to validate the CDA in real-world use. First, small-scale tests were conducted indoors with a set of concrete cinder blocks comprising the object(s) to reconstruct. Later, after the algorithm had proven effective, the experiment was scaled up to a full-size building element in an outdoor environment.

5.3.1 Small-scale Indoor Tests

The preliminary step in validating the CDA was to create a controlled, small-scale test setup with a cracked object to reconstruct. The 120 cm diam. indoor circle laid out for autonomous sampling was selected as the test environment since 3D reconstructions had already been successfully created using it. A cinder block with a sizeable horizontal crack placed in it was the target object because it 1) represented a common building material and 2) contained a textured surface to provide feature points for the reconstructions. The camera was manually

placed at 48 locations around the circle and a 640x480 image was collected each time. A picture of the block and resulting 3D point clouds (sparse and dense together) are shown in Figure 5.4.

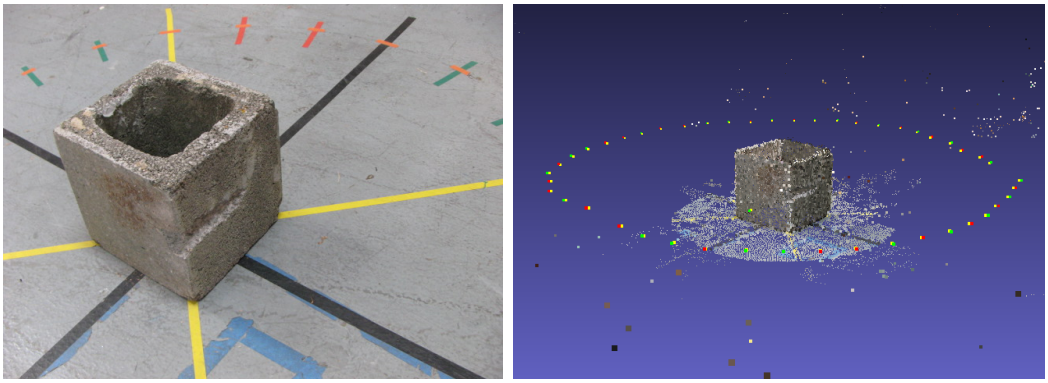


Figure 5.4: Small-scale test block (right) and point clouds (left) for CDA verification.

The dense point cloud was trimmed and meshed with a Poisson mesh once the point clouds had been output from the reconstruction pipeline. This meshed model was fed into the Matlab program, producing the colored-coded model shown below.

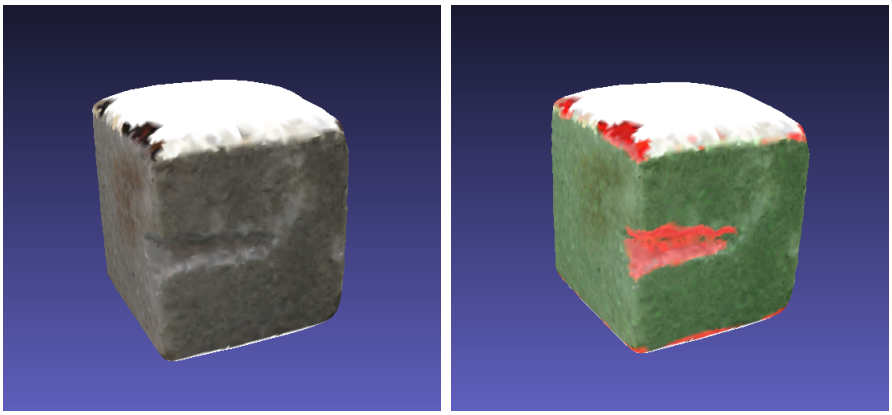


Figure 5.5: Mesh model of cinder block before (left) and after (right) running CDA.

As can be clearly seen in Figure 5.5, the crack detection algorithm successfully identified the horizontal gash in the center of the cinder block and color-coded it with a red tint (some non-crack fringes at the top and bottom of the model were also colored). This result was

very encouraging and full-scale tests were pursued.

5.3.2 Full-scale Outdoor Tests

A full-scale experimental setup was created in order to assess the real-world performance of the crack detection algorithm. The target object to reconstruct was a 140(h) x 53(w) x 23(d) cm section of concrete cracked during a load test. It contained two major cracks on the "front" side, with the upper, larger one having a height of approx. 10 cm and a depth of 5-6 cm, and the lower, smaller one having dimensions of approx. 3 cm(h) x $<$ 2 cm(d). There were several smaller cracks appearing on the column as well (see Figure 5.6).

Figure 5.6: The concrete column used as a test building element in outdoor environment.

This concrete column was placed in the center of a flat, clear area and photographed using a point-and-shoot digital camera attached to the USL ground robot (Figure 5.7).

From previous testing it was determined that 48 images should be sufficient to produce a

detailed reconstruction. Therefore, these images were collected by positioning the robot at 30 cm intervals along a circle with a radius of 2.4 m around the column (this distance was chosen since it allowed the entire column to be visible in every camera frame).

Figure 5.7: Ground robot configured with digital camera for image collection.

Once all 48 of the 7 megapixel images were collected (which took about 5 minutes) they were transferred to a six core, 3 GHz computer for processing. They were run through the image-based 3D reconstruction pipeline described above, and after one hour of computational time, a dense reconstruction of the column and surrounding area was created (the current implementation performs all reconstruction steps sequentially and as a result has a high computational cost). This point cloud was trimmed to contain only the column and meshed. Afterwards, it was rotated and scaled prior to running the crack detection algorithm implemented in Matlab.

5.4 Results

The point cloud model of the column captured a full 360 degrees and contained some 309,000 colored 3D points. The generated Poisson mesh from the dense reconstruction contained 228,000 vertices and 455,000 faces (solver divide = 12, octree depth = 12). Using the completeness criterion discussed above, the meshed model was determined to be 100% complete. A sample image of the column and its various models are shown in Figure 5.8 below.

Figure 5.8: From left to right, a photo of the concrete column, the point cloud model, and the Poisson mesh model.

Following the creation of the 3D mesh model the crack detection algorithm was run to test its performance. The CDA successfully identified a wide range of cracks visible on the column, including both the 10 cm and 3 cm high ones previously mentioned. It was also able to pick up several other smaller cracks and surface defects in the concrete, including a crater-shaped depression appearing on the back of the column (not visible in the images shown here). However, cracks which were perfectly vertical or too small to be reconstructed in 3D (less than 0.5 cm height) could not be automatically identified. Finding perfectly vertical cracks is not possible using the current version of the algorithm, since it only examines normals for deviation from perpendicular, regardless of the direction they point in that plane.

Figure 5.9 presents the color-coded model after analysis with the CDA. Notice that the cracked regions (areas whose mesh element faces are not perpendicular to the column's axis) are clearly identified with a reddish hue. The figure also demonstrates the effect of changing the threshold angle for detection. In the upper left corner, the more conservative angle criterion of 20° is used, which allows for greater deviation before an element is flagged as part of a crack. The larger image uses the standard angle criterion of 15°, and the lower left corner uses 10°. The 10° threshold, as expected, finds the largest number of cracks. In these experiments, the range of angles chosen represents the region in which cracks were successfully identified without returning too many false positives.

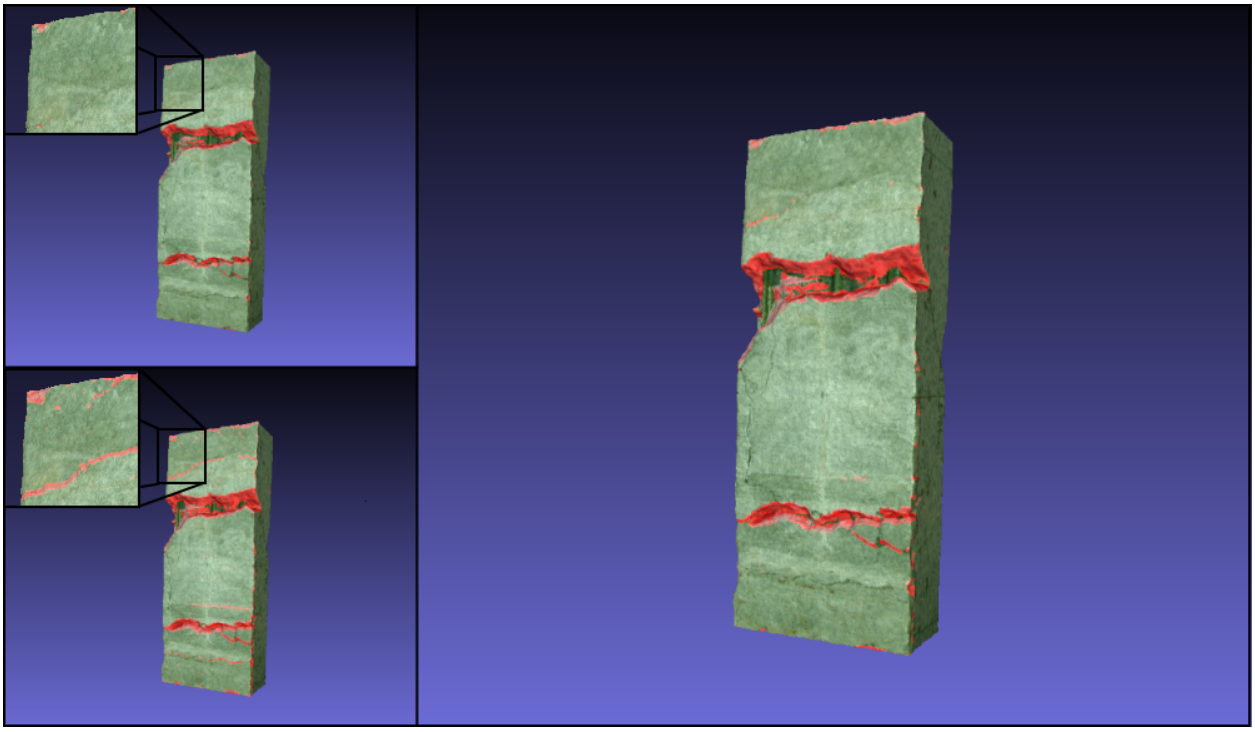


Figure 5.9: Colored mesh model after running crack detection algorithm. The threshold angle is varied between 20° (upper) to 15° (right) to 10° (lower).

Model scaling based upon knowledge of the first two camera positions (approx. 30 cm apart) also worked well. Using only those two points in the reconstruction, the model was automatically scaled to 97% of its true dimensions, and from the model the estimated cracked area was 1322 cm^2. Additionally, the crack detection algorithm was able to locate and mark

some cracks which were difficult to identify in the original images (see Figure 5.10, below).

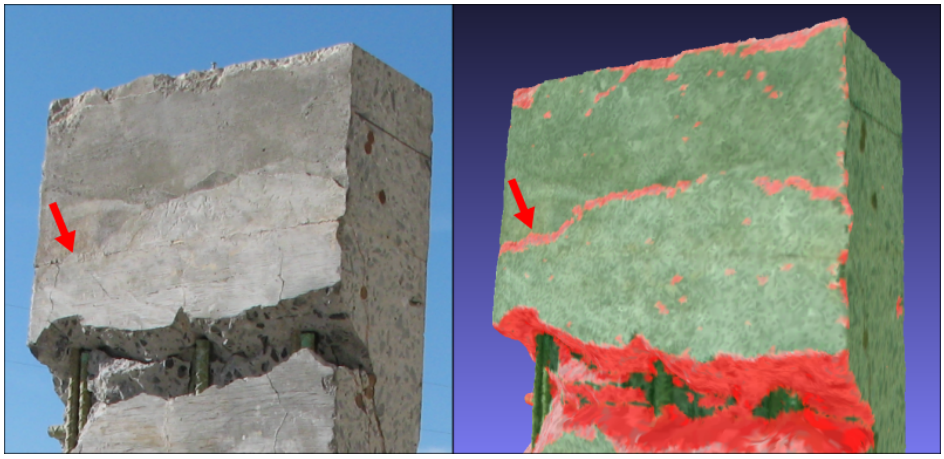


Figure 5.10: An illustration of one of the benefits of the algorithm a crack that is difficult to locate in a 2D image is found and clearly identified in 3D.

In summary, a new automated 3D method for identification of cracks in building elements was successfully demonstrated. Cracks of heights 10 cm and 3 cm were found and color-coded on the 3D model of the 140 cm tall concrete column, and some were able to be identified in 3D that were very difficult to see in 2D images. Finally, cracks smaller than 0.5 cm could not be detected because they were not reconstructed in 3D.

Chapter 6

Conclusions and Recommendations

This thesis presents a novel scoop sample collection system which is incorporated into both semi-autonomous and fully autonomous collection routines and tested on a mobile robot. It also advances a new 3D methodology for using a set of images gathered by a mobile robot, along with 3D reconstructions, to identify cracks in buildings. In this chapter the work completed towards those goals is summarized along with a detailed discussion of improvements and recommendations for future work to further enhance the applicability and robustness of the systems.

6.1 Conclusions

A scoop sample collection system was created to pick up discrete "chunk" samples weighing no more than 250 grams using the USL ground robot. This system was sized to fit into the robot's interchangeable payload tray and also designed to minimize weight since it could be deployed from the RMAX helicopter. Once the scoop concept was verified at a smaller scale the full-scale system was assembled, and following assembly an auto-scooping algorithm was developed to relieve the operator of performing the procedure manually. The scoop system

also incorporated several novel features, including a magnetic attachment system to maintain sample containment inside the scoops even if the robot's power were shut off. Testing was performed in a variety of common outdoor (and man-made) environments, including concrete, asphalt, dirt, gravel, grass, and sand. Scoop performance was generally very good and the sample was successfully contained when driving the robot over rough terrain following collection.

Image-based 3D reconstructions were also used for major components of the work done for this thesis. Existing open source software provided colored 3D point clouds and camera positions from a set of input images. These formed the basic building blocks for performing autonomous and semi-autonomous sample collection and later a crack detection algorithm as well. A major contribution of this work, for both systems implementing the 3D reconstruction techniques, was the ability to successfully employ point clouds generated using uncalibrated cameras for detailed analysis.

Autonomous sample collection concepts combined the scoop system with 3D reconstructions. In order to do this a suite of custom software programs was written to perform additional processing on the 3D point clouds and compute commands for the robot. The user could select an object to pick up in the reconstructed scene or the software could perform this function automatically based upon a set of parameters which defined the sample. Originally, the USL ground robot was intended to be used for autonomous sample collection, but hardware problems with it led to the creation of a custom Scoopbot robotic platform instead. The Scoopbot included two independent drive wheels, a camera, microprocessor, the scoop system, and the necessary hardware to transmit commands and images wirelessly back to the base station computer. It drove along a predefined circumferential path taking pictures of a sample-containing area which was subsequently modeled in 3D. Then a sample could be selected and the commands computed to order the Scoopbot to collect it. Ultimately the system was able to demonstrate fully autonomous sample collection in both "full mode" which created a complete 360° model of the scene, and a less time-consuming "fast mode" where only a limited area was traversed.

Finally, the point clouds generated by the image-based 3D reconstructions were used to create an automatic building crack detection algorithm. It operated on principles similar to "full mode" sample collection, i. e. employing a 360° model of a building element whose images could be gathered by a mobile robot. The algorithm itself examined the surface normals of the triangular elements created by a Poisson reconstruction. If these normals did not point perpendicular to the building element's axial direction within a given threshold they were considered to indicate cracked areas. Software was written to perform this analysis automatically and it was also used to color-code cracked areas in the output file that was displayed to the user. Automatic building crack detection was shown to work at small scale on individual concrete blocks and at full-scale on a column as well.

6.2 Recommendations for Future Work

While the initial work for these systems has been completed, there are many potential improvements which could be made to broaden the spectrum of their applications. This section describes in some detail potential modifications which can be foreseen for the scoop collection system, autonomous sampling, and crack detection.

The scoop collection system's overall performance was quite good; however, performance was slightly impaired by gravel and completely impaired by grass. Adding a row of tines to each side of the scoop would be a simple, straightforward way to facilitate more aggressive sample collection. They could be machined out of aluminum and attached via small bolts or adhesives to the underside of the current scoops and aligned with the plastic beveled edge. The tines should be arranged in an alternating pattern so that they interlock when closed, and using tines would allow the scoop to contain a sample even if wasn't able to close completely around it.

Also, in the current configuration, the torque links attach near the top of the scoops and there is some slack in the system when it closes. Moving the attachment point of the torque

links down nearer the ground edge of the scoop (i.e. changing to longer links) could help eliminate some of the slack by applying force directly where the scoops are closing around a sample. Another possible addition would be an interior liner that formed a complete seal when the scoops shut, if such isolation from the exterior environment proved to be necessary for a specific collection scenario.

Regarding autonomous sample collection, the most significant challenge remains obtaining high quality 3D reconstructions from the open source software. Unfortunately, improving those processes would involve a tremendous amount of work far beyond the scope of this thesis. However, the field of image-based 3D reconstruction is still evolving rapidly and it is likely that performance, speed, and quality will only increase in the future. Along those lines, new SfM techniques have recently been developed which reduce computation times by at least an order of magnitude over the ones implemented herein. That, combined with feature detectors like ASIFT (instead of SIFT), could allow for substantial improvements to the current system, advancing towards real-time processing and analysis.

The next logical step for autonomous sample collection would be implementing the system on the USL ground robot. Analysis showed that with encoders and an accurate IMU/magnetometer there should be no difficulty achieving sufficient movement resolution to perform image gathering and sample collection. Of course, integrating those sensors, testing the system, etc. would be very time consuming, but the USL robot's increased mobility vs. Scoopbot would open a much wider range of environments to testing. On the custom software side several improvements could be made as well, primarily for reporting the sample parameters in fast mode. For the time being only the full mode reconstruction builds a complete model of the sample (and therefore produces accurate dimensions and volume) but techniques could be used to estimate the sample size from what is modeled in fast mode. For instance, fast mode can view approximately half of the target object. The dimensions of the other half could be predicted by mirroring the existing points across their back plane, producing a model of the entire object. Alternatively, some sort of meshing (such as Poisson) could be used to create an enclosed volume from a limited number of 3D points for

the sample. Either of these methods should allow a more accurate volume and size to be reported in fast mode.

Other voxelization approaches using octrees or concentric spherical shells might also provide worthwhile improvements in speed and/or accuracy. Furthermore, right now occlusions or partial visibility of the target sample are not accounted for. Introducing a new voxel coloring scheme (e.g. dark orange = visible in most images, light orange = visible in few images) along with a (potentially probabilistic) means of predicting the size of buried or occluded objects could supply additional valuable information to the user and the autonomous algorithm. It would also be useful to have a means of automatically identifying the sample's material composition prior to collection so a better mass estimate could be provided.

Much more could also be done with the area point cloud models from the 3D reconstructions. Particularly, it would be very nice to determine whether or not the selected sample was accessible from the robot's final position after collecting the images. A surface mesh of the ground plane could be made and its traversability assessed relative to robot parameters (e.g. maximum climbable slope). Then a path planning algorithm could be run to determine how to best reach the sample across the terrain, rather than simply approaching it directly as is now done. Finally, in terms of robot position sensing during navigation, implementing ground-facing optical flow sensors could provide accurate location and angle feedback during operations.

Improvements could be made to the automatic building crack detection algorithm as well. For starters, it would be beneficial to run the algorithm on a set of images from an actual damaged/cracked building. The algorithm itself could also be tuned or trained to automatically identify the proper threshold for crack detection in a given scenario. Furthermore, if building element reconstructions could be correlated with CAD data (such as from a building information model, or BIM), more processing and analysis could be performed. Developing a system to make that correlation could allow entire buildings to be analyzed at once and greatly speed disaster recovery efforts.

Implementation of the future work ideas is strongly encouraged should further development of these projects be pursued. The work presented in this thesis provides a solid theoretical foundation and basic systems to build upon, but their current form leaves lots of room for improvement. Overall, novel approaches to a sampling device, autonomous collection, and automatic building crack detection have been presented, and it is hoped that they will be used to produce better engineering solutions to the complex problems they have begun to address.

Bibliography

[1] Michael Rose. "Design of a Helicopter Deployable Ground Robotic System for Hazardous Environments". MS thesis. Blacksburg, VA: Virginia Polytechnic Institute and State University, 2010.

[2] Michael Couch. "Pneumatic Particulate Collection System for an Unmanned Ground Sampling Robot". MS thesis. Blacksburg, VA: Virginia Polytechnic Institute and State University, 2010.

[3] Shajan Thomas. "Design of a Teleoperated Rock Sampling System". MS thesis. Blacksburg, VA: Virginia Polytechnic Institute and State University, 2011.

[4] C. Fagerer, D. Dickmanns, and E.D. Dickmanns. "Visual grasping with long delay time of a free floating object in orbit". In: *Autonomous Robots* 1.1 (1994), pp. 53–68.

[5] NASA. *Mission Timeline: Surface Operations*. 2012. URL: `http://marsrover.nasa.gov/mission/tl_surface_nav.html`.

[6] Thomas L. Hopkins. "A survey of marine bottom samplers". In: *Progress In Oceanography* 2.0 (1964), pp. 213–256.

[7] iRobot. *iRobot PackBot(TM) Accessories*. 2009. URL: `http://www.irobot.com/filelibrary/pdfs/gi/accessories/irobot_accessories.pdf`.

[8] J. H. Lever et al. "Semi-Autonomous Robotic Sampler for Explosives Residues in Surface Soils." In: (2004), pp. 1–24.

[9] R.E. Arvidson et al. "Results from the Mars Phoenix Lander Robotic Arm experiment". In: *Journal of Geophysical Research - Part E - Planets* 115.E1 (2010), pp. 00–02.

[10] P. Younse et al. "Sample acquisition and caching using detachable scoops for mars sample return". In: *IEEE Aerospace Conference Proceedings*. 2009.

[11] Blake Carpenter Varun Ganapathi Oussama Khatib Andrew Y. Ng Ellen Klingbeil Deepak Drao. "Grasping with Application to an Autonomous Checkout Robot". In: *International Conference on Robotics and Automation (ICRA)*. 2011.

[12] T. Kubota et al. "Robotics and autonomous technology for asteroid sample return mission". In: *12th International Conference on Advanced Robotics*. 2005, pp. 31–38.

[13] D. Gaines D. Thompson R Castao R. C. Anderson C. de Granville M. Burl M. Judd T. Estlin B. Bornstein and S. Chien. "AEGIS Automated Targeting for the MER Opportunity Rover". In: *Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space*. 2010.

[14] Daniel Scharstein and Richard Szeliski. "High-Accuracy Stereo Depth Maps Using Structured Light". In: *CVPR 2003* 1 (2003), pp. 195–202.

[15] K. Kutulakos and S.M. Seitz. "A theory of shape by space carving". In: *International Journal of Computer Vision* 38.3 (2000), pp. 199 –218.

[16] N. Snavely, S.M. Seitz, and R. Szeliski. "Photo tourism: exploring photo collections in 3D". In: *ACM Transactions on Graphics* 25.3 (2006), pp. 835–46.

[17] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *International Joint Conference on Artificial Intelligence* (1981), 674679.

[18] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, 2004.

[19] D.G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[20] Y. Furukawa and J. Ponce. "Accurate, Dense, and Robust Multi-View Stereopsis". In: *CVPR 2007*. 2007, pp. 1–8.

[21] C. Koch and I. Brilakis. "Pothole detection in asphalt pavement images". In: *Journal of Advanced Engineering Informatics* 25.3 (2011), 507515.

[22] T. Saar and O. Talvik. "Automatic asphalt pavement crack detection and classification using neural networks". In: *Proceedings of the Biennial Baltic Electronic*. 2010, pp. 345–348.

[23] Lifang L. Guiying Y. Zhang B. and G Tiantai. "A portable digital detection technique of building surface crack". In: *Proceedings of the SPIE*. 2011, pp. 819110–16.

[24] German S. Zhu Z. and I. Brilakis. "Visual Retrieval of Concrete Crack Properties for Automated Post-earthquake Structural Safety Evaluation". In: *Journal of Automation in Construction* 20.7 (2011), pp. 874–83.

[25] Dietmar Meinel Olaf Paetsch Steffen Prohaska Valentin Zobel Karsten Ehrig Jurgen Goebbels. "Comparison of Crack Detection Methods for Analyzing Damage Processes in Concrete with Computed Tomography". In: *International Symposium on Digital Industrial Radiology and Computed Tomography*. 2011, Poster 2.

[26] Mohammad R. Jahanshahi and Sami F. Masri. "Adaptive vision-based crack detection using 3D scene reconstruction for condition assessment of structures". In: *Automation in Construction* 22 (2012), pp. 567–576.

[27] Peter Fanto. "Modified Bundler.exe". In: *Unpublished* (2011).

[28] M. Streigel. *NCPTTs updated assessment tools aid in disaster response and recovery.* 2011. URL: `http://ncptt.nps.gov/ncptts-updated-assessment-tools-aid-in-disaster-response-and-recovery/`.

[29] C. Wu. *SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)*. 2007. URL: `http://cs.unc.edu/~ccwu/siftgpu`.

[30] Y. Furukawa et al. "Towards Internet-scale multi-view stereo". In: *CVPR 2010*. 2010, pp. 1434–1441.

[31] Bolitho M. Kazhdan M. and H Hoppe. "Poisson surface reconstruction". In: *Proceedings of the Symposium on Geometry Processing*. 2006, pp. 61–70.

Appendix A

StepperBot Performance Testing Data

StepperBot Performance Testing/Characterization

Basic parameters:

100 steps per cm
26.04 steps per deg
11.75" baseline (inside wheel distance)

Papers:

60 x 30 cm gridded area for distance
60 cm diam circle for angle

Test Procedure:

1. Print 2 papers, 1 for distance (1 cm grid, 0.1 cm resolution)
   1 for angle (1 deg resolution)
2. Clear surface, attach paper
3. Drive known distance/angle, measure, record results

Distance			Angle		
Goal (dist)	**Goal (steps)**	**Actual (dist)**	**Goal (angle)**	**Goal (steps)**	**Actual (angle)**
10	1000	10.2,10.1,10.15, 10.15,10.2	90	2344	90, 90.8, 90,89.8, 89.8
20	2000	20.3,20.3	360	9376	361, 360.5
10	985	10.0,10.0	7.5	195	7.3,7.5,7.5,7.5,7.4
50	4925	49.7,49.7, 49.8	10	260	10.0,10.0,10.0

Angle test procedure:

1. Align robot axles with horizontal axis
2. Place wheels equidistant apart on axis (as marked)
3. Read starting angle from perpendicular bracket on back of robot
4. Command rotation
5. Read ending angle from bracket, note any precession of robot center

Appendix B

Calculations for Vertical Servo Motor

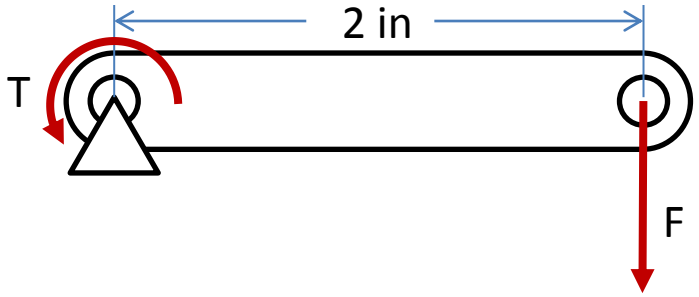
Vertical Scoop Servo Calculations

The vertical scoop servo will be responsible for lifting the scoops once they contain the sample. At the maximum extension of the vertical links the force will be applied at their full 2" length from the servo.

Maximum sample design weight = 250 grams (8.8 oz)

Weight of scoop center assembly = 11.2 oz (est. from CAD)

Force-Moment diagram:



Where:

T = motor torque to lift F at a distance of 2 inches

F = load applied by sample + load applied by scoop center assembly

Equation:

$$T = 2 * F$$

$$T = 2 * (11.2 + 8.8)$$

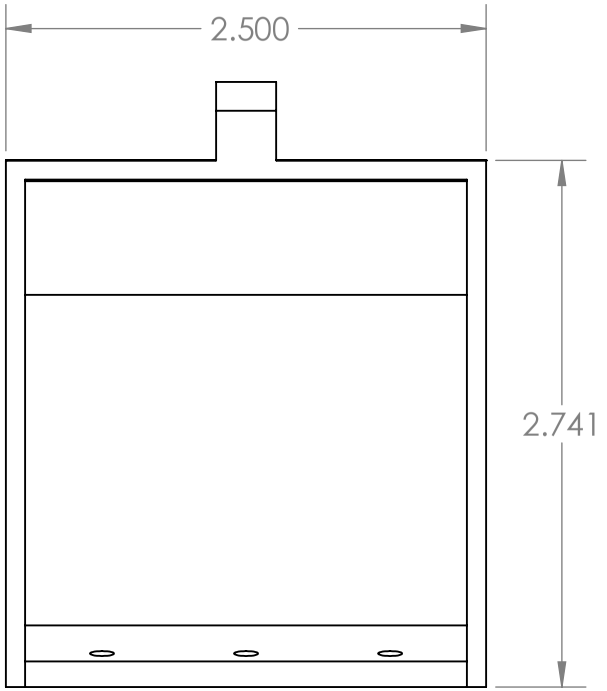
$$T = 40$$

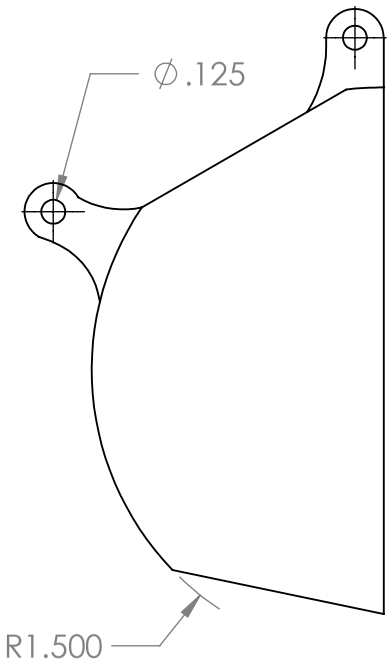
Therefore T must be greater than or equal to 40 oz-in.

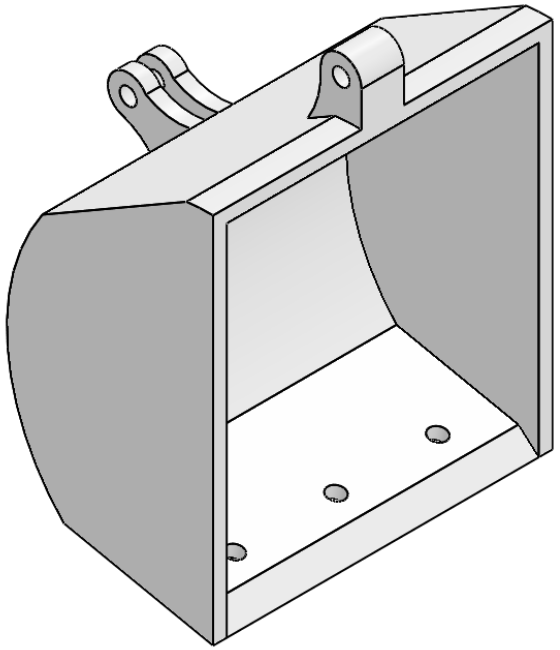
Servo used: Hitec HS-5485HB → 6.0V stall torque = 89 oz-in → safety factor of 2 on torque requirement

Appendix C

CAD Detail Drawing for Mini Scoops







		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	M. Torok	4/12/2012
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL ABS	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

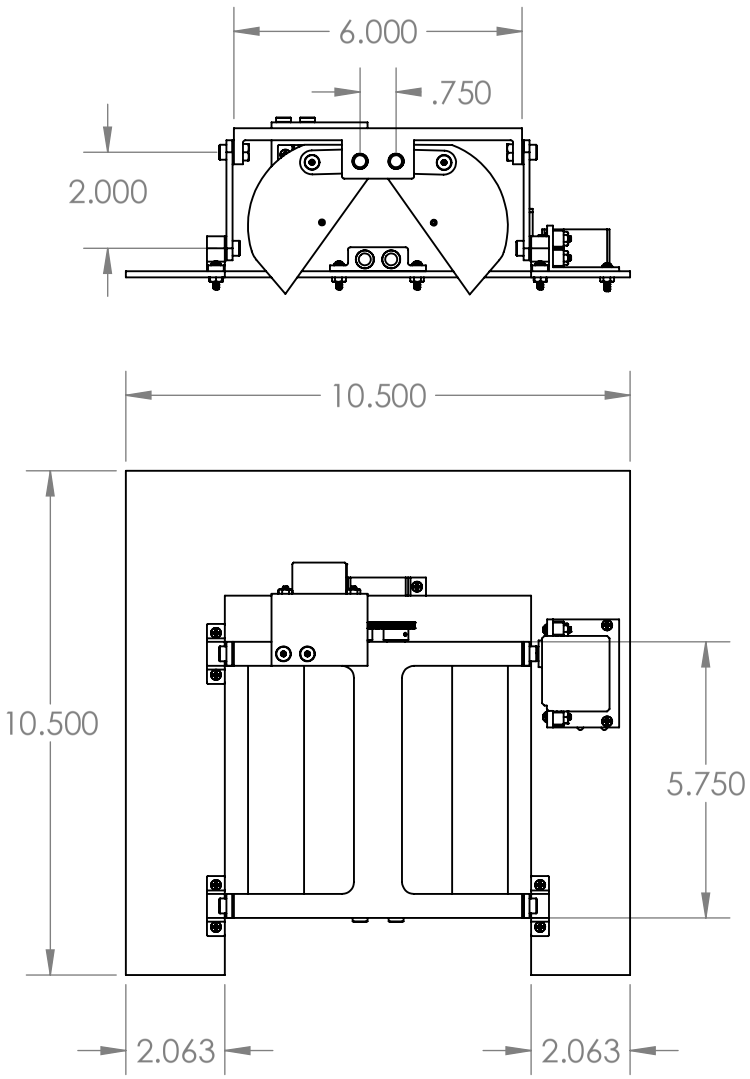
Mini Scoop

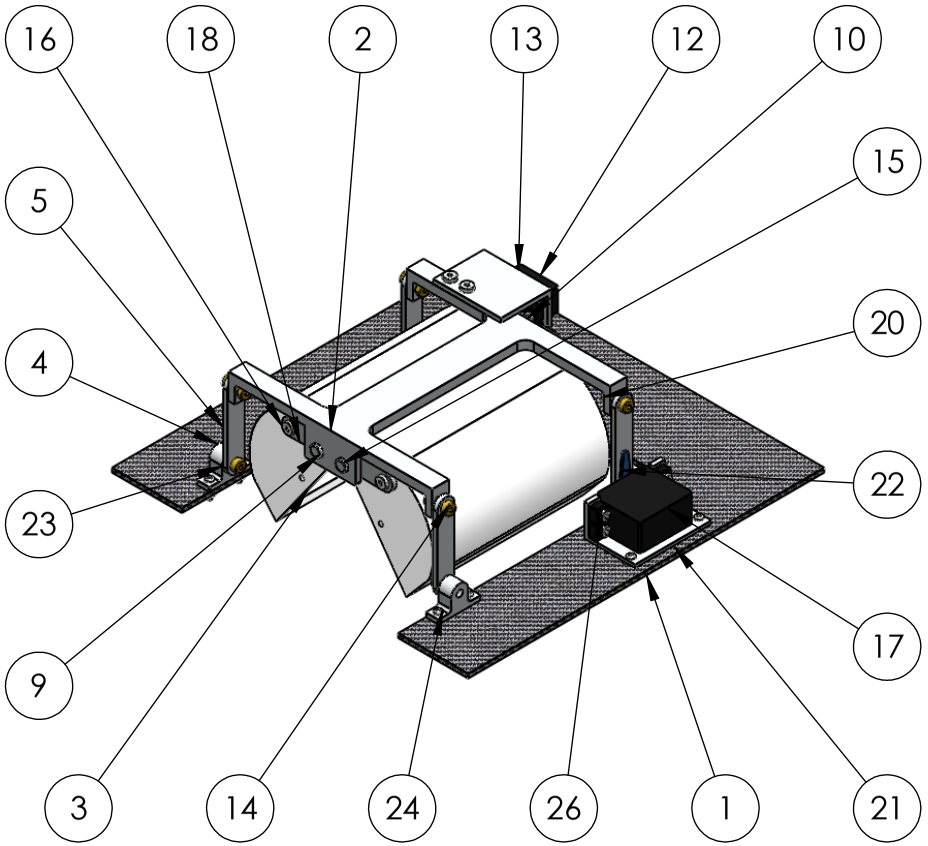
SIZE	DWG. NO.	REV
A	MiniScoop	3
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

Appendix D

CAD Detail Drawings for Scoop Collection System





		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	M. Torok	4/12/2012
			CHECKED		
			ENG APPR.		
			MFG APPR.		
			Q.A.		
		INTERPRET GEOMETRIC TOLERANCING PER:	COMMENTS:		
		MATERIAL Varied			
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

Scoop Assembly

SIZE	DWG. NO.	REV
A	Scoop Assembly v3	3

SCALE: 1:4 | WEIGHT: | SHEET 1 OF 2

5 4 3 2 1

NOTE:
PLEASE SEE DIGITAL CAD FILES
FOR PART DIMENSIONS, ETC.

ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Tray		1
2	Bracket RP		1
3	Scoop2		2
4	Attachment (new)		3
5	Link		4
6	Latch Base		2
7	Latch Holder		1
8	Latch Plate		2
9	Shaft with Flats		2
10	Gear		2
11	ServoGear_Mini		1
12	Servo		1
13	Servo Bracket		1
14	95446A260		7
15	Sleeve Bearing		4
16	92220A162		4
17	Link Servo		1
18	Torque Link		2
19	Placeholder		1
20	92671A009		6
21	Link Servo Bracket		1
22	Servo Horn		1
23	Washer		7
24	90279A110		18
25	91253A077		2
26	90480A005		18

UNLESS OTHERWISE SPECIFIED:

DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±

INTERPRET GEOMETRIC TOLERANCING PER:

MATERIAL

FINISH

DO NOT SCALE DRAWING

	NAME	DATE
DRAWN	M. Torok	4/12/2012
CHECKED		
ENG APPR.		
MFG APPR.		
Q.A.		

COMMENTS:

NEXT ASSY | USED ON

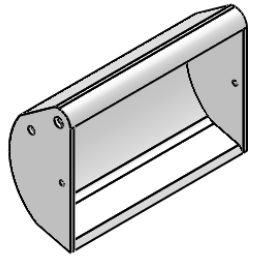
APPLICATION

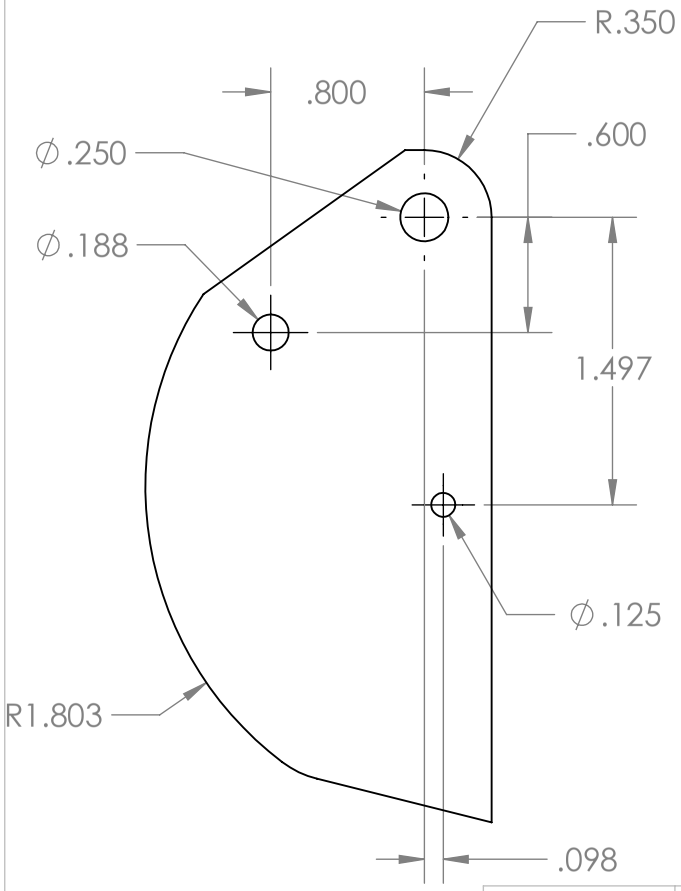
TITLE:

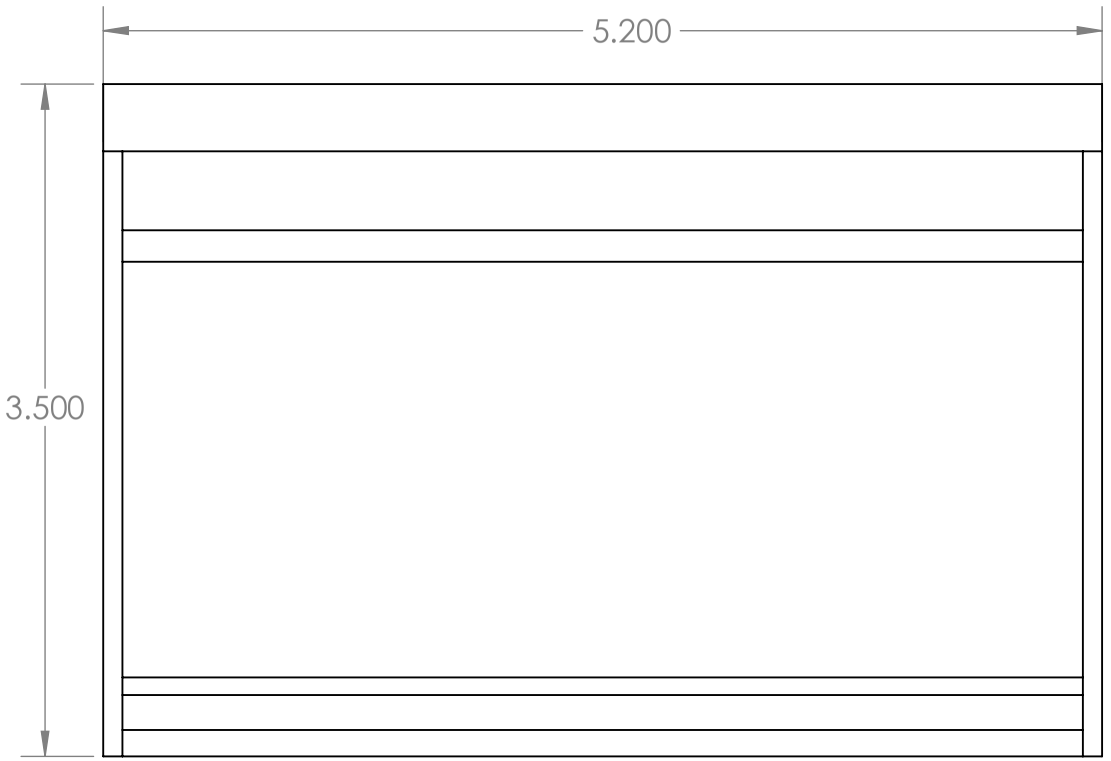
SIZE A | DWG. NO. Scoop Assembly | REV 3

SCALE: 1:4 | WEIGHT: | SHEET 2 OF 2

5 4 3 2 1







		UNLESS OTHERWISE SPECIFIED:		NAME	DATE
		DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±	DRAWN	M. Torok	4/12/2012
			CHECKED		
			ENG APPR.		
			MFG APPR.		
		INTERPRET GEOMETRIC TOLERANCING PER:	Q.A.		
		MATERIAL ABS	COMMENTS:		
NEXT ASSY	USED ON	FINISH			
APPLICATION		DO NOT SCALE DRAWING			

TITLE:

Full-Scale Scoop

SIZE	DWG. NO.	REV
A	Scoop2	2
SCALE: 1:1	WEIGHT:	SHEET 1 OF 1

5 4 3 2 1

The End