41

# Increased Autonomy for Planetary Rover Operations

by

Brian D. Hoffman

S.B. Mechanical Engineering
Massachusetts Institute of Technology, 1997

Submitted to the Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 1998

Signature of Author ..............................................................
Department of Mechanical Engineering
May 8, 1998

Certified by ..
Jean-Jacques E. Slotine
Professor of Mechanical Engineering and Information Science
Professor of Brain and Cognitive Science
Thesis Supervisor

Accepted by ..............................................................
Ain A. Sonin
Chairman, Department Committee on Graduate Students

# Increased Autonomy for Planetary Rover Operations

by

Brian D. Hoffman

Submitted to the Department of Mechanical Engineering
on May 8, 1998 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

## ABSTRACT

We have implemented a suite of algorithms which enable increased autonomy in planetary rovers. Specifically, we have implemented an iterative, semi-sparse range map generation algorithm which is adaptive in both texture and distance, allowing the computationally efficient generation of range maps. Using these range maps we have implemented a simple obstacle avoidance methodology. To these two algorithms we add an image segmentation algorithm based on correlating texture-based image segmentation with edge detection to achieve approximately 94% correct segmentation in pseudo-natural environments. Combining the above algorithms, we demonstrate roaming obstacle avoidance with autonomous sample acquisition.

Implementing a range map registration algorithm provides us with one measure of the rover's inter-frame state change. By fusing this measurement of the rover's change in state with the commanded motion via Kalman filter, we are able to achieve a position estimate error for the rover in soft soil simulant and rocky terrain of approximately 0.4 m after a 6 m traverse. This higher-quality state estimate relative to dead reckoning alone allows us to perform navigation to user designated goals using a real time potential flow based path planning methodology. We present several experimental trials in challenging terrain where the rover is successful in navigating to relatively directly accessible and very obstructed user designated goals.

Thesis Supervisor: Jean-Jacques E. Slotine
Title: Professor of Mechanical Engineering

## Acknowledgements

I would like to acknowledge my parents William J. Hoffman, Jr. and Deborah D. Hoffman, for their loving support, comments, insight, and financial backing.

I would also like to acknowledge my wonderful friend Samantha F. Edgington for all her support - mental, emotional, and otherwise.

To Professor Jean-Jacques E. Slotine, I give thanks for capturing and fostering my initial interest in controls through 2.14, 2.152, and 2.160, and for serving as my advisor in UROP, Bachelor's Thesis, and and Master's Thesis work.

The entire JPL Telerobotics research team under Dr. Paul Schenker deserves much thanks for their support of this work, especially Dr. Eric Baumgartner, Hrand Aghazarian, Mike Garrett, Lee Sword, Anthony Ganino, Randy Lindemann, and David Zhu.

Additionally, I would like to acknowledge Hans Jacob Feder, MIT, for the original version of the potential flow code, Todd Litwin, JPL, for the original version of the stereo triangulation and camera calibration code, and Professor Terry Huntsberger, University of South Carolina, for the original version of the wavelet transform and texture-signature computation code.

This work has been supported by the NASA Telerobotics Technology Program directed by Dave Lavery, NASA Headquarters, and Charles Weisbin, JPL Technology and Applications Program (TAP) Directorate.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

NASA's Pathfinder mission to Mars, carrying the Sojourner rover, was a phenomenal success. Both the rover and lander outperformed their expected nominal missions, sending back stunning pictures of the Martian surface, and performing a variety of science functions, including temperature, atmospheric, and soil characteristic measurements. However, the Sojourner rover was very limited in many respects. It required frequent commands input from the ground team on earth, as its autonomous capabilities were very limited. Additionally, the rover was limited to explorations within the visual range of the lander cameras (approximately 10 meters), due to quickly growing errors in the rover's dead-reckoned-only position state estimate, given the difficult terrain. Staying within the range of the lander cameras allowed the ground team to update the rover's state estimate manually. Given the success of the Pathfinder mission, NASA/JPL is moving forward with additional mars rover missions, in 2001, 2003, and 2005. [37]

The current planned objectives for NASA's 2001 and 2003 rover missions to Mars include traverses totaling tens to hundreds of kilometers, over time frames spanning several months to a year or more. During these long distance traverses, various science functions are to be performed, including sample acquisition and cache, spectroscopic imaging, micro-camera imaging, and sample abrasion. Sample acquisition and cache activities have the highest priority, in preparation for the planned 2005 Mars sample return mission. The 2005 Mars sample return mission, in contrast to the 2001/2003 missions, is envisioned to last only a matter of days, as a rover sprints out from its descent vehicle to acquire and retrieve a sample cache from one of the "dead" sample collection rovers, quickly traversing back to its ascent vehicle for return to earth. [13], [27].

In order to adequately achieve these ambitious desired mission objectives within the specified time frames, increased autonomy must be introduced and incorporated into NASA's 2001, 2003, and 2005 rover platforms. Teleoperation is infeasible as signal delay times are on the order of 7 to 10 minutes. Additionally, command uplink / data downlink sessions are able to occur only approximately twice per earth day. Thus, increased autonomy is necessary, as high average ground speeds are desired to maximize the amount of terrain area covered per unit time.

Of paramount importance are the robustness and computational efficiency of these autonomous behaviors. Not only must the algorithms function with repeatedly high performance in varying terrains and lighting conditions, but also work well given the computational limitations imposed by radiation hardened processor requirements and power constraints. Additionally, these behaviors must take into account, and be capable of dealing with, the possibilities of sensor and actuator degradation over time, both during transit, landing, and surface traverse.

To accomplish the stated mission objectives within the specified time frames, we desire that the rover be capable of robustly and autonomously completing the following tasks:

- terrain map generation
- obstacle avoidance
- sample acquisition, and cache
- sample inspection using onboard science instruments
- soil trenching
- roaming behavior
- state estimation
- real time path planning
- navigation to a user designated goal

For the task of obstacle avoidance, we desire that the rover be capable of distinguishing hazardous terrain features with a minimum of false positives or undetected hazards, with the former requiring the rover to perform excessive, and time consuming, avoidance maneuvers, and the latter being directly hazardous to the health and well being of the rover.

Our implementation of vision-based autonomous obstacle avoidance, using a texture-based interest operator [5], coupled with stereo correlation and triangulation to generate 3D terrain maps, is adaptive to varying surface and lighting conditions, as well as being computationally efficient.

To successfully accomplish the task of sample acquisition and cache, we have implemented an image segmentation algorithm, also using the same texture-based interest operator mentioned previously. The output of this texture-based interest operator, correlated with edge detection, reliably segments out rock samples from background soil in camera images. Once segmented, the 3D locations of any samples may be computed using stereo correlation and triangulation. Using the inverse kinematics of the rover mounted manipulator, we may then acquire and cache the located samples.

Coupling simple obstacle avoidance behaviors with our segmentation algorithm, we then demonstrate the implementation of roaming behaviors, where the rover may wander the surface, looking for 'interesting' rock samples. In this case, a sample is acquired if it is deemed "interesting" based on the output of the texture-based interest operator, the known texture signatures of the previously collected samples, and the approximate empty volume remaining in the sample cache container.

Similarly, rather than being acquired and cached, an interesting sample may be inspected by deploying onboard science instruments which operate on the surface of the sample. These autonomous activities are typically initiated for selected samples by remote scientists. To address the need to perform sample inspection, we have implemented a suite of autonomous routines, which may be initiated through a network based remote graphical user interface.

In many cases, planetary geologists may wish to examine unexposed soil. Thus, the rover must be capable of autonomous force-adaptive trenching, such that it varies its digging patterns in differing soil conditions to maximize both soil volume excavated per unit time, as well as the safety of the rover mounted manipulator. We show one possible implementation of this functionality.

In soft soil, wheel slip is significant, resulting in degradation of the dead reckoned vehicle state estimate. Likewise, in terrain with dense rock fields, obstacle climbing and numerous rotations in obstacle avoidance maneuvers significantly degrade the dead reckoned vehicle estimate.

Thus, for the task of state estimation, we implement an extended Kalman filter [1] to fuse together two estimates of the rover's change in state. The first estimate is visually based, using the registration of successive terrain maps generated at each frame during the rover's traverse to derive the rover's between-frame rotation and translation [32]. The second estimate is the dead reckoned estimate, using the rover's wheel odometry. The combination of the two measurements gives us a much more accurate state estimate than relying on either measurement alone. Once a state estimate is available, it becomes possible to place successive terrain maps in a common reference frame, incrementally generating a world terrain map.

In addition to simple roaming behavior, planetary geologists may wish to have the rover traverse to specific landmarks or large rocks visible from a mast generated panorama, or from the navigation cameras. Thus, to address the problems of navigation and path planning to a user designated goal, we must implement not only autonomous path planning, but also path planning which is able to continuously replan based on our current state estimate and our position error estimate during our traverse to the designated goal.

Using our current Kalman filtered state estimate, we continuously replan our desired path to the user designated goal. We demonstrate the successful application of a real-time 2D potential-flow based path planning methodology

[7], capable of dealing with dynamic environments, a needed feature for dealing with uncertainty in the rover's state estimate.

Finally, we wish to emphasize that all approaches described have been experimentally verified using NASA/Jet Propulsion Laboratories' Lightweight Survivable Rover (LSR), pictured in Figure 2.1. These algorithms, although they perform well as individual units, also work together in support of one another, as an integrated system, to successfully demonstrate a suite of robust and adaptive autonomous behaviors.

## 1.2 Previous Work

Much work has been done in range map generation, some using "texture" features such as corners or other interest point identification methods [2] [20]. Some generation algorithms use parallel approaches [8]. Given the computational limitations imposed by power and radiation hardened processor requirements, we have developed a computationally efficient semi-sparse range map generation algorithm which is adaptive to both distance and texture.

Likewise, much work has been done in the area of mobile vehicle state estimation, often using vision [1] [34] [20] [12] [22] [10] [17], and often using statistical estimation techniques to fuse together measurements of the rover state from multiple sensors [1] [20] [25]. However, experimental validation of statistical state estimation approaches such as Kalman filtering for mobile robotics generally only been performed in laboratory settings, benign terrain, [1] [20], [17], [3] or only in simulation [25]. Some navigation performance evaluation has been pursued [19], however state estimation during these trials relied on dead reckoning only. We add experimental validation in very difficult terrain, as might be experienced by a rover performing difficult traverses on the surface of Mars, and show the feasibility of such a sensor fusion approach to state estimation.

Similarly, potential flow based path planning has been demonstrated extensively in simulation, [7], [15], [4], [23], [16], (see [7] for more references). Here, we demonstrate experimental validation of real time path planning using harmonic potentials in dynamic environments. Additionally, the environment consists of soft soil, and relatively dense rock fields, validating the potential flow based path planning approach in complex and challenging environments.

### 1.2.1 Summary

Finally, we show the unification of these many various techniques in system level trials. This includes adaptive range map generation, successive range map registration, Kalman filtering of vision and dead reckoning for improved state estimation, and real time path planning using potential flow-based methodologies. Path planning relies on the state estimate, which relies on Kalman filtering of the registered range maps, which relies on range map registration, all within an integrated system. However, this also validates each of the system components, which could of course be used separately in another system.

Additionally, we demonstrate successful roaming obstacle avoidance behavior, autonomous sample acquisition, and soil trenching.

Section 2 provides an overview of the experimental hardware we use to validate our algorithms, broken down into mechanical hardware, sensor suite, computing environment, and system interconnections. Section 3 provides a more detailed discussion of the algorithms we have implemented for the autonomous routines discussed above. In section 4 we detail the results of testing our autonomous algorithm suite in difficult terrains. We discuss our results in section 5, and provide concluding remarks and suggestions for further work in section 6.

# Chapter 2

# Experimental Hardware

## 2.1 Mechanical Hardware

Our mechanical hardware, or rover platform, consists of two main sub-sections, these being the rover body itself, or Lightweight Survivable Rover (LSR), and the rover mounted manipulator arm, MicroArm-I.

The Lightweight Survivable Rover was developed in an attempt to satisfy many of the challenging requirements for a successful Mars rover design. These include a thermally stable warm-electronics box, small stowed volume, large deployed volume and high ground clearance, minimal mass with high structural integrity, and science-instrument deployment capability.

MicroArm-I was designed with similar intent, to have high structural integrity while minimizing mass and maximizing task- flexibility as a robotic manipulator.

In satisfying these design requirements, a variety of novel materials and structures were developed and employed, as discussed in the following sections.

The LSR Rover and MicroArm-I manipulator were developed in NASA / JPL's Science and Technology Development Section (354), in a task lead by Dr. Paul Schenker/JPL, and supported by Dr. Eric Baumgartner/JPL, System's Engineer.

### 2.1.1 LSR Rover

The LSR rover, pictured in figure 2.1, is a 6-wheeled, skid-steered, rocker-bogie design, designed by a mechanical engineering team lead by Lee Sword/JPL/IS-Robotics. To satisfy the requirement for a thermally stable warm electronics box, or WEB, designed by Gregory Hickey/JPL, which is used to protect the delicate computing hardware from external Mars temperature swings of 100 degrees Celsius or more, a lightweight composite honeycomb design was employed, filled with an ultra-lightweight aerogel material for insulation. This WEB also serves as the main rover body/chassis, to which the rocker-bogie running gear are mounted.

2D composite tubes with aluminum end-fittings were used in the construction of the running gear, which pivots in-plane about its attachment on the side of the chassis. As LSR is a skid-steered vehicle, it sees significant bending moments about these pivots, due to the length of the running gear. It was thus retrofitted, by Anthony Ganino/JPL, with aluminum reinforcements mounted through the wall of the WEB, directly to the more rigid honeycomb WEB-base for increased structural integrity.

The 2D composite running-gear also featured a novel wheel design, again developed by Lee Sword, allowing the wheel to be stowed in a collapsed state, then deployed into a stable, rigid configuration for surface traverse.

The rover drive motors are located in the center of each wheel. The three motors on each side of the rover are currently ganged together electrically, allowing the rover forward, reverse, and rotational motion, but disallowing traction improving techniques derived from independent wheel control.

A 2D composite crossbar connects the two sets of running gear, pivoting about a point on the top front of the chassis, which stabilizes the WEB and disallows free rotation about the two side running gear pivots. This connecting crossbar also saw increased loading during experimental testing, as was hence retrofitted with an aluminum coupling for increased structural integrity.

A science instrument tray is located beneath the rover. Normally stowed, it may be deployed down and forwards from the rover using a 4-bar linkage, to bring a statically mounted science instrument into contact with a sample of interest.

The front of the rover, to which MicroArm-I is mounted, was retrofitted with the equivalent of an optics bench, using a very rigid, yet lightweight, honeycomb composite material, into which regularly spaced attachment points were installed.

This plate provides not only a rigid attachment base for MicroArm-I, but also an effective mounting surface for the rover's stereo camera pairs and our sample cache container. The cache container may be seen in figure 2.1, as a white cylinder with black cross markings, used in targeting and cache acquisition by a Sample Retrieval Rover (SRR).

### 2.1.2 MicroArm-I

MicroArm-1, mounted to the front of LSR, is a 4 degree of freedom robotic manipulator arm, with an additional degree of freedom in its end effector. MicroArm-I, designed by Randall A. Lindemann/JPL, is constructed nearly in its entirety of 3D and 2D carbon fiber composite material. The two links of the arm are 2D composite tubes, with the joint housings and end effector being made from the newer 3D composite material. The 3D composite material has the advantage of being machinable into complex geometries, and also being much lighter than aluminum. However, we chose to retrofit the arm with aluminum joint housings for increased structural integrity, given the large amount of experimental testing to be performed.

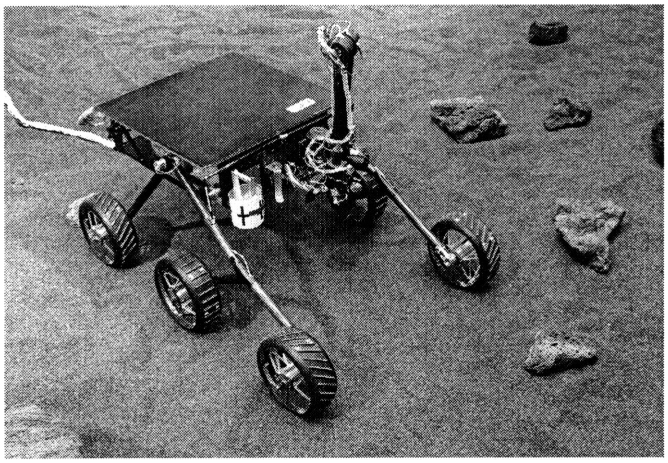


Figure 2.1: Lightweight Survivable Rover (LSR)

MicroArm-I, 0.7 m at full extent, is driven by 1-inch-pound torque capability piezoelectric ultrasonic motor actuators. These motors, developed by the Shinsei Corporation, rely on the frictional interaction between two piezoelectric discs, one located in the motor rotor and the other located in the motor stator. A standing wave is set up in the discs, with the phase difference between the waves in the two discs resulting in the motion of the rotor, and hence the motor shaft. These motors have the advantages of high torque-to-weight ratios, and the ability to operate in very low temperature and near vacuum environments, making them an excellent choice for space applications. Some investigation may be performed in the future with regard to innovative control methodologies to compensate for the low-speed stall of the motors. Additional investigation is being performed by a team lead by Yosi Bar-Cohen/JPL to develop higher-performance ultrasonic motors for space applications.

The revolute joints of MicroArm-I are driven by the aforementioned ultrasonic motors through 200:1 harmonic gear reductions, chosen for their large gear reduction and low volume.

The powered multifunction end effector designed for MicroArm-I has available a close distance color imaging camera, a rotary abrading tool, and a clam shell scoop/gripper.

MicroArm-1 is pictured in figure 2.2. Further discussion of the LSR / MicroArm-I hardware may be found in [26].

The forward and inverse kinematics for MicroArm-I are provided in appendix A for completeness.

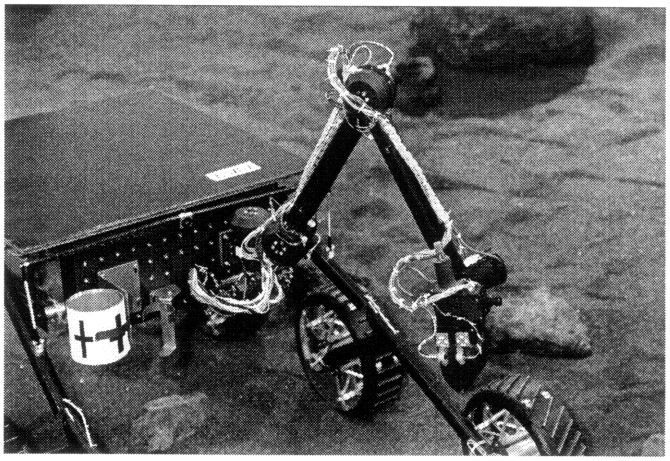


Figure 2.2: MicroArm-I

## 2.2 Sensor Suite

For a Mars rover to perform effectively, it must be able to sense or measure some aspects of its internal state, as well as measure parts of its environment. Thus, we have outfitted LSR with a variety of sensors. However, the sensor suite would be benefited by many additions, which we will discuss in section 6.2.

### 2.2.1 Stereo Camera Pair

LSR is outfitted with a pair of SuperCircuits CCD cameras capable of producing grayscale images. The cameras are mounted in a plane to a 3D composite camera fixture as a binocular pair, with a 10 cm baseline, mounted directly beneath MicroArm-I, at an angle with the horizontal of approximately $-20°$.

Each CCD camera produces an image with a resolution of 512 columns by 486 rows, which we generally subsample to 256x243. Each camera is equipped with a 120° field-of-view lens, automatic gain control (AGC), runs on 12 volts DC, and draws approximately 100 mA of current.

The stereo camera pair as mounted on LSR may be seen in figure 2.3.

### 2.2.2 Strain Gauges

As MicroArm-I is actuated by non-backdrivable piezoelectric / ultrasonic motors, through 200:1 gear reductions, using motor current measurements to determine the forces experienced by the end effector would be extremely difficult.

Thus, we have instrumented each of the arm links with extremely sensitive strain gauges, able to measure the deflection of the links in response to externally applied loads. Using the Jacobian, we may then obtain some rough estimate

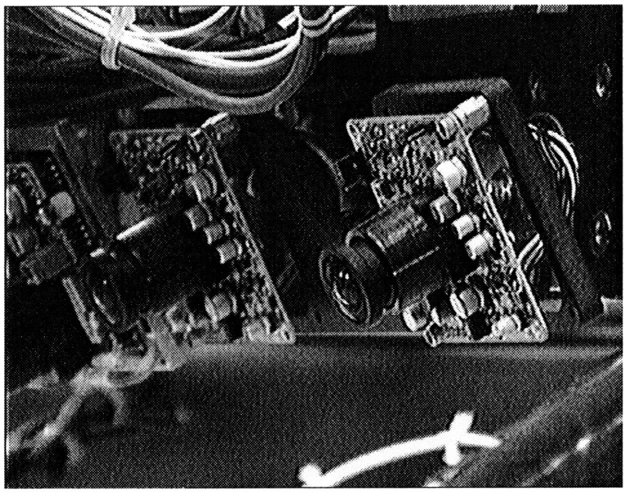


Figure 2.3: Stereo CCD Cameras

as to the applied cartesian tip force (although not technically the cartesian tip force as we do not have a square Jacobian, having only two links instrumented).

Each strain gauge passes through a signal amplifier and filter, before being sampled.

### 2.2.3 IR Home Position Sensors

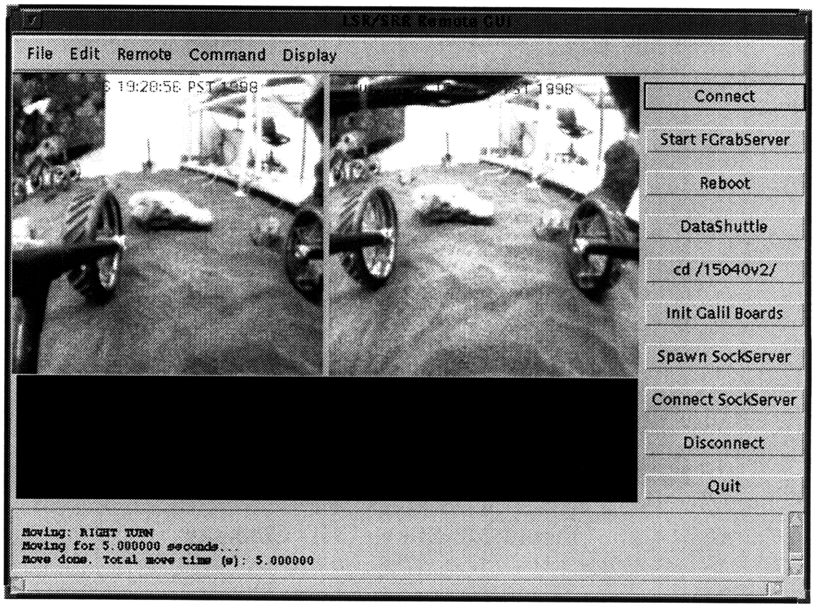
To accurately calibrate the relationship between the cartesian workspace of MicroArm-I and the workspace of the stereo cameras, we require that the manipulator be capable of locating itself absolutely in cartesian space. Thus, we have equipped the manipulator with IR emitter/detector-pair home position sensors, which allow it to be repeatedly homed to within approximately 1 degree at each joint. Each IR sensor signal is passed through a simple inverting buffer to preserve its integrity, as our A/D converter has relatively low impedance.

## 2.3 Computing Environment

The computing environment for the LSR/MicroArm-I platform is currently a distributed/hybrid (offboard) SUN Sparc-20 / Silicon Graphics Indigo-2 / VME-bus / Sun Sparc-SLC architecture. This computing environment is currently in the process of being integrated into an on-board 133 MHz 586/PC-104 architecture.

### 2.3.1 User Interface

A multi-threaded Java-based graphical user interface (GUI) is provided to initiate autonomous routines on the remote LSR rover, as well as display images



```
Moving: RIGHT TURN
Moving for 5.000000 seconds...
Move done. Total move time (s): 5.000000
```

Figure 2.4: LSR Rover Graphical User Interface

and data as collected by the rover to the remote scientist/operator. This GUI runs on the Sun Sparc 20.

When the GUI is executed, the rover controller, through the GUI, starts a frame-grab-request server on the SGI Indigo-2, via a Telnet connection over a pre-existing TCP/IP network. The remote rover controller, in a similar fashion, also connects to the VMEbus, through a Sun Sparc SLC, and initializes the motion controllers for the rover, then starts a command parsing process, discussed further in section 2.3.3.

A typical screen capture of the User Interface during operation of the rover may be seen in figure 2.4. Discussion of a similar Java-based graphical user interface for telerobotic control may be found in [14].

### 2.3.2 Image Acquisition

As the LSR rover does not have frame grabbing hardware on-board, we make use of the frame grabbing hardware provided by the Silicon Graphics Indigo-2 workstation. Unfortunately, the SGI workstation is only able to capture frames from one video signal at a time, and thus we must multiplex the video signal provided to the SGI, if we wish to be able to capture stereo image pairs, for example.

Thus, we pass the video signals through a computer controlled video-mux, and switch the signals by writing commands over a serial connection between the VMEbus and the video-mux.

### 2.3.3 Rover Command and Control

The majority of the computation for the rover's autonomous routines is performed on the Sparc 20, by the GUI / Application, either in Java or by calling native code compiled from C.

Low level control of the rover is performed by two VMEbus-based Galil motion controller boards. Also in the VMEbus is a single 68040 processor board, running the VxWorks 5.2 real-time operating system. The command parsing process, originally designed by Hrand Aghazarian/JPL for the control of a lander mounted manipulator, MarsArm-II, is started by the remote rover controller through the Java-based GUI. This parsing process receives commands from the GUI via TCP/IP socket connections, interprets them, and sends the appropriate commands to the Galil motion control boards.

The functionality of the Galil motion control boards is not limited solely to motor control, but also includes on-board A/D and digital output. The A/D converters are used to sample the strain gauges mounted on MicroArm-I, as well as the output from the IR emitter-detector pair home-position sensors. The digital output lines are used to control the LSR left and right drive motor gangs. The output signals from the Galil boards to the LSR rover, and the sensor input to the A/D converters and SGI frame grabbing hardware all passes through a 50 foot tether attached to the back of LSR.

Of course, the Galil motion control boards were originally designed to control analog motors, not ultrasonic piezoelectric motors. Thus, the analog MicroArm-I motor control signals from the Galil boards pass through conditioning circuitry designed by Mike Garrett/JPL before being presented to dedicated control circuitry provided by Shinsei for the motors. Similarly, the digital output lines used to control the LSR drive motors in a bang-bang fashion are passed through motor drive circuitry, also conceived by Mike Garrett.

### 2.3.4 System Interconnections

Although obviously this system was constructed using the available hardware, connected in a complicated fashion to achieve the desired functionality, the system interconnections are more clearly visible when depicted in figure 2.5.

It is a testament to the algorithms's robustness and flexibility that they are able to perform as well as they do given the system configuration.

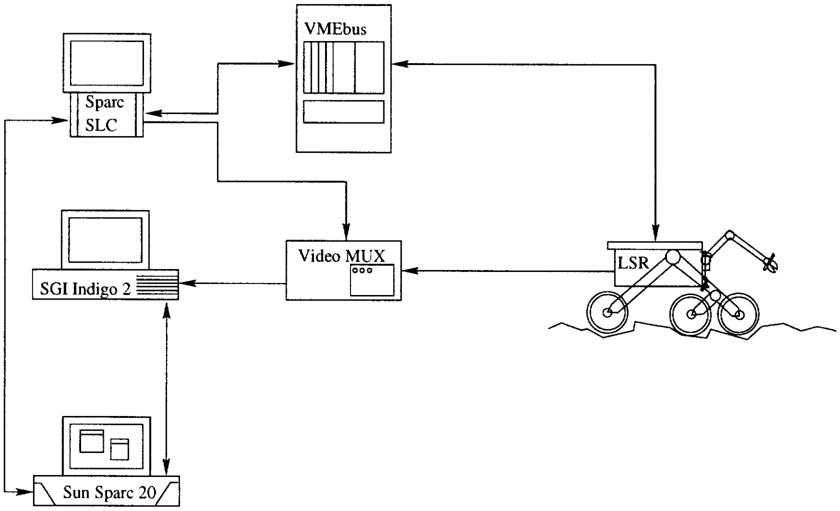


Figure 2.5: System Setup

# Chapter 3

# Algorithms

In this chapter, we present a suite of algorithms to enable a 2001/2003 Sample Acquisition rover to autonomously perform obstacle avoidance, sample segmentation, acquisition, and cache, roaming behavior, state estimation, real-time path planning, and navigation to a user designated goal. The algorithms are designed with memory and computational limitations in mind, given the radiation hardened processor requirements of the Mars rovers.

This algorithm suite for terrain map generation, obstacle avoidance, state estimation, path planning and navigation, as described below, borrows heavily from the description presented in [13]. The reader familiar with that presentation may proceed directly to section 3.5, containing additional algorithm developments not presented in that initial paper.

## 3.1 Terrain Map Generation

If the rover is to effectively navigate and avoid potentially hazardous obstacles, it must be able to generate some internal representation of the world around it, e.g. a terrain map. Although such representations do not have to be euclidean in nature [24], [18], a 3D reconstruction of the world simplifies further algorithm development in some respects. We choose to use passive stereo vision rather than active measurement methods such as sonar, or laser scanning, because such camera technology has been space proven, typically has low power requirements, and stereo algorithms can be used to generate range maps quickly.

While moving through its initially unknown environment, the rover will construct maps of the terrain immediately in view, using its wide field of view navigation cameras for input. A sample stereo image pair taken from the rover navigation cameras may be seen in figure 3.1.

To address the problem of robust and computationally efficient terrain map generation, beginning with a stereo image pair, we present the novel application of a texture-based interest operator in conjunction with stereo correlation and triangulation, such that our algorithm is adaptive in refinement to both texture

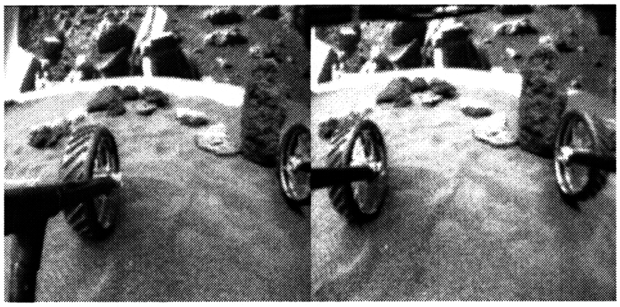


Figure 3.1: Stereo Pair

and distance.

### 3.1.1 Wavelet Transform

After acquiring a stereo image pair (e.g. figure 3.1), we compute the wavelet decomposition of each image using the standard two dimensional Harr wavelet basis. The algorithm for this decomposition, which effectively decomposes the image into horizontal, diagonal, and vertical channels at multiple resolution levels is given below for completeness, from [29].

```
func Harr2D(I: array $[1...2^j, 1...2^k]$ of reals)
    loop: $row = 1...2^j$
        DecompRC($\mathbf{c}[row, 1...2^k]$)
    end loop
    loop: $col = 1...2^k$
        DecompRC($\mathbf{c}[1...2^j, col]$)
    end loop
end function
```

```
func DecompRC(c: array $[1...2^j]$ of reals)
    $\mathbf{c} = \mathbf{c}/\sqrt{2^j}$
    $g = 2^j$
    while: $(g >= 2)$
        DecompSub($\mathbf{c}[1..g]$)
        $g = g/2$
    end while
end func
```

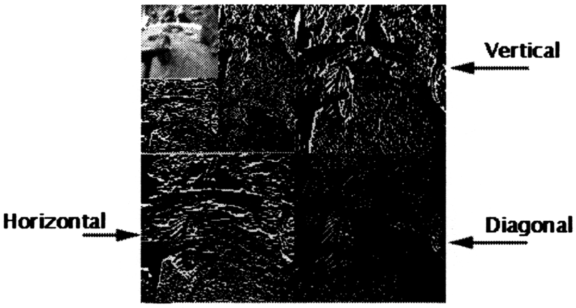


Figure 3.2: Wavelet Transform, Showing Horizontal, Vertical and Diagonal Channels, for a $l = 2$ Decomposition

```
func DecompSub(c: array [1...2^j] of reals)
    loop: $i = 1...2^j/2$
        $\mathbf{c}'[i] = (\mathbf{c}[2i-1] + \mathbf{c}[2i])/\sqrt{2}$
        $\mathbf{c}'[2^j/2 + i] = (c[2i-1] - c[2i])/\sqrt{2}$
    end loop
    $\mathbf{c} = \mathbf{c}'$
end function
```

The output of the wavelet decomposition is thresholded, based on pixel value, to 0 or 255 (i.e. 0/1). In our implementation, we use a threshold value of 4, and a decomposition level, $l$, of 2, which gives us adequate results. A representative wavelet decomposition from one of the images in a stereo image pair, showing the horizontal, vertical, and diagonal channels, in 2 levels, may be seen in figure 3.2.

### 3.1.2 Texture Signature Computation

The "ground" texture is then computed in several known locations at the bottom of each image, between the rover wheels, as shown in figure 3.3 using a texture operator on the wavelet decompositions. The output of this operation is a texture signature, $TS$, in the space defined by the horizontal (H), vertical (V), and diagonal (D) channels of the wavelet decomposition [5] and, for each channel, is defined as

$$TS = \sum_{i=0}^{l} \frac{\frac{\sum_{j=-n/2}^{n/2} \sum_{k=-m/2}^{m/2} |W(a+j,b+k)|}{(nm)}}{l} \tag{3.1}$$

where the parameter $l$ refers to the level to which the wavelet decomposition, $W$, has been computed. Thus, for each channel (H,V,D) in the wavelet decomposition, we sum the contributions to the texture signature from each level of the decomposition, by summing the absolute value of the thresholded wavelet decomposition over a window of size $n \times m$ centered on $(a, b)$ as indicated by

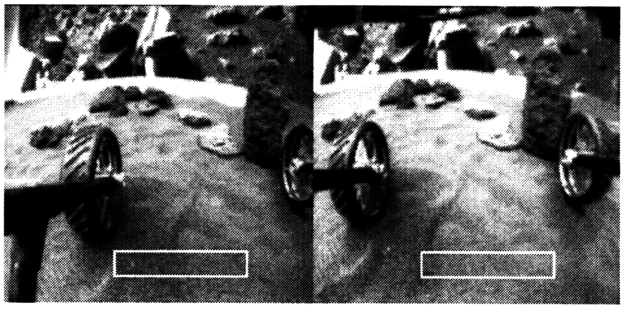


Figure 3.3: Ground Texture Computation

equation 3.1. The values of $n$ and $m$ are dependant upon the current level, and the coordinates $(a, b)$ are the image coordinates $(x, y)$ in the wavelet transform at level $l$ and channel $(H, V, D)$. In our implementation, $n = m = 8$ at $l = 1$ and $n = m = 4$ at $l = 2$.

### 3.1.3 Adaptive Texture Based Interest Point Selection

Once we have an average H,D,V texture signature which we assume corresponds to the ground, we search the left image with a stepsize of 16 pixels, in an area where we expect to find obstacles (between the rover wheels), looking for points which do not match the average ground texture signature which has been previously computed. These "interesting" points are marked, as shown in figure 3.4.

A range map computed using a stepsize of 16 will be referred to as a "Level 16" range map. Similarly, a range map computed with a stepsize of 8 will be referred to as a "Level 8" range map, and so on.

### 3.1.4 Correlation

#### Epipolar Geometry

In performing correlation between stereo images, we will make use of a simple geometric property, known as the epipolar constraint, to reduce our search for a matching point from an area in the image (2D) to a line in the image (1D). A simple depiction of the epipolar geometry arrangement may be seen in figure 3.5 [33].

For a point in the left image, $m_1$, its corresponding point in the right image $\mathbf{m_2}$ must lie along the line $\mathbf{Fm_1}$ in the right image, where $\mathbf{F}$ is the $3 \times 3$ fundamental matrix. We will avoid further digression on the subject of the epipolar constraint and refer the interested reader to [33] or [6].

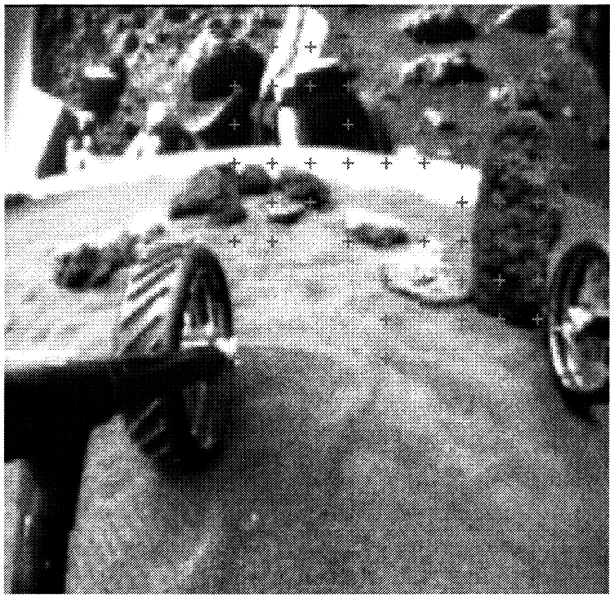


Figure 3.4: Points of Interest, Level 16

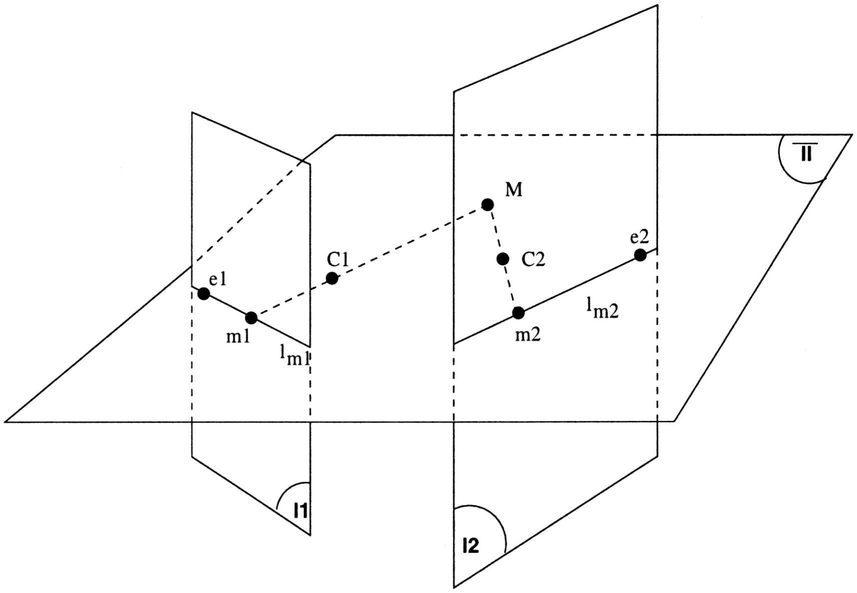


Figure 3.5: Epipolar Geometry

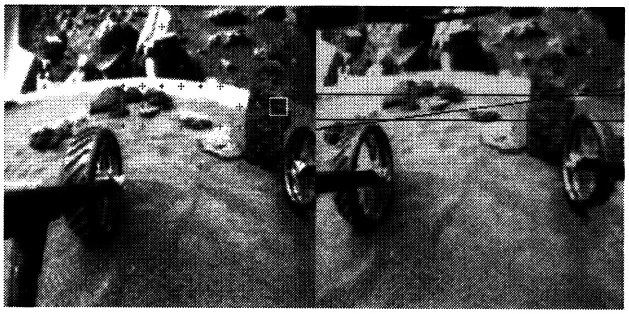


Figure 3.6: Correlation Using Fundamental Matrix

### The Epipolar Geometry and Warped Images

The epipolar geometry is technically valid only for cameras which can be reasonably modeled using pinhole camera models. A camera with a 120° field of view lens can not normally be reasonably modeled as a pinhole camera.

However, although we are using 120° field of view lenses on the stereo camera pair, we have found that the error in using the fundamental matrix to compute a line of correspondence (rather than computing a curve of correspondence as would be expected for such highly distorted images) is on the order of approximately one pixel [36]. This experimental result is corroborated by the investigation performed in [35].

### Dewarping and Rectification?

Normally, it is considered computationally efficient to dewarp and rectify the images in a stereo image pair, such that the subsequent correlation may be performed by searching along scanlines (rows) in the image. However, in the case where we are only searching for a small number of points, to generate a relatively sparse range map, it may prove more computationally efficient to forego this dewarping and rectification process. Additionally, if the camera models used in the dewarping process are not very accurate, the resulting dewarped and rectified images may be offset from one another quite a bit, in which case finding correct matches by searching along scanlines (±N scanlines) may prove difficult.

### Pixel Correlation

Using the epipolar geometry described above, we attempt to correlate the interesting points found in the left stereo image with their corresponding points in the right stereo image, using the fundamental matrix [33] to reduce this search from a 2D problem to a 1D problem.

In correlating, we search along the epipolar line of left image point $\mathbf{m_l}$ given by $\mathbf{F}\mathbf{m_l}$, where $\mathbf{F}$ is the $3 \times 3$ fundamental matrix, and $\mathbf{m_l}$ is a vector in the form $[\ x\ y\ 1.0\ ]^T$, where $x$ and $y$ are the column and row pixel locations, respectively. We assume that the cameras are reasonably aligned to the same horizontal axis, and thus limit our search vertically to ±10 rows. Figure 3.6 details a left image point of interest and its corresponding epipolar line and search band. To account for possible errors in the epipolar geometry, we also search left and right at each point on the epipolar line by ±8 columns. However, if we use cameras which can reasonably be modeled as orthographic, then this error is greatly reduced and, correspondingly, the number of columns to the right or left of the epipolar line we bother searching is also reduced. The method by which the fundamental matrix is computed is described further in section 4.1. For our stereo cameras, the **F** matrix is

$$\begin{bmatrix} -8.907e-06 & 1.159e-03 & -7.025e-02 \\ -1.155e-03 & 2.457e-04 & 1.966e-01 \\ 6.521e-02 & -2.082e-01 & 9.532e-01 \end{bmatrix}$$

where this particular **F** matrix was derived from point matches in the environment in a stereo image pair taken with the LSR cameras, using a freely available software package [36].

A valid match between a left image point and its corresponding point in the right image is defined by the greatest correlation score found during the search along the right epipolar line, larger than some threshold. To compute the correlation scores between points in the left and right image, we employ the standard sum minus average divided by standard deviation over an $n \times m$ box in each image.

In this normalized definition of correlation, the maximum correlation value is +1 (identical) and the minimum value is −1 (totally uncorrelated). In our implementation the correlation threshold is chosen to be 0.80, following [33]. Likewise, $n$ and $m$ are chosen to be 7, which has become the de-facto standard correlation window size [8]. The correlation score between a left image point $\mathbf{m_1}$ and a right image point $\mathbf{m_2}$ is defined as

$$Score(\mathbf{m_1}, \mathbf{m_2}) = \frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} \begin{smallmatrix} [I_1(u_1+i, v_1+j) - \overline{I_1(u1,v1)}] \times \\ [I_2(u_2+i, v_2+j) - \overline{I_2(u2,v2)}] \end{smallmatrix}}{(2n+1)(2m+1)\sqrt{\sigma^2(I_1) \times \sigma^2(I_2)}} \tag{3.2}$$

where $\overline{I_k(u,v)}$ is defined by equation 3.3 and is the average image intensity over a window of size $(2n+1) \times (2m+1)$ centered on $(u,v)$ in image $I_k$, where $k = 1, 2$. The average image intensity is defined as

$$\overline{I_k(u,v)} = \sum_{i=-n}^{n} \sum_{j=-m}^{m} \frac{I_k(u+i, v+j)}{[(2n+1)(2m+1)]} \tag{3.3}$$

The variable $\sigma(I_k)$, in equation 3.2, is defined by

$$\sigma(I_k) = \sqrt{\frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} I_k^2(u,v) - \overline{I_k(u,v)}}{(2n+1)(2m+1)}} \tag{3.4}$$

and is the standard deviation in the image intensity over a window of size $(2n+1) \times (2m+1)$ centered on $(u, v)$ in image $I_k$, where $k = 1, 2$.

### Sub-Pixel Correlation

Once we have an initial point match between the left and right images, we compute the subpixel location of the matching point in the right image, which is obtained by fitting a parabola to the pixel-level correlation scores in both the row and column directions, as given by [31]

$$\begin{aligned} \Delta col &= \frac{S_{col-} - S_{col+}}{2(S_{col-} + S_{col+} - 2S_{col})} \\ \Delta row &= \frac{S_{row-} - S_{row+}}{2(S_{row-} + S_{row+} - 2S_{row})} \end{aligned} \tag{3.5}$$

where $S_{row}$ and $S_{col}$ refer to the correlation score between a point in the left image $\mathbf{m_1}$ and a point in the right image $\mathbf{m_2}$ at position $(col, row)$, where the maximum pixel level correlation score has been found. $S_{col-}$ is the score between $\mathbf{m_1}$ and the point in the right image one column to the left of $\mathbf{m_2}$. $S_{row-}$, $S_{col+}$ and $S_{row+}$ are computed similarly.

The right image row and column are then set equal to their integer value found during the pixel level correlation, plus the delta row and column found during the subpixel correlation. This procedure greatly improves the smoothness of the derived range map, with little extra computation.

## 3.1.5 Computation of 3D Location

Using the set of matching image points computed to subpixel precision, we then triangulate to obtain 3D position information, using metrically calibrated camera models that take into account radial lens distortion [11]. The calibration procedure will be described further in section 4.1.

## 3.1.6 Adaptive Mesh/Terrain Map Refinement

Using the computed 3D position information, we can further iteratively improve the density of our range map by returning to those interesting points computed in the previous iteration that are found to be within a threshold "danger distance" from the rover.

Searching the image over an area of $(Stepsize_{i-1})^2$ centered on each interesting point using a stepsize of $(Stepsize_{i-1})/2$, we find a new set of "interesting points" based on texture and compute their 3D position information, as can be seen in figure 3.7.

In this fashion we perform what we have dubbed "adaptive mesh refinement", being somewhat akin to the process of adaptive mesh refinement in finite element analysis, in that we only bother computing the range map in areas where we care

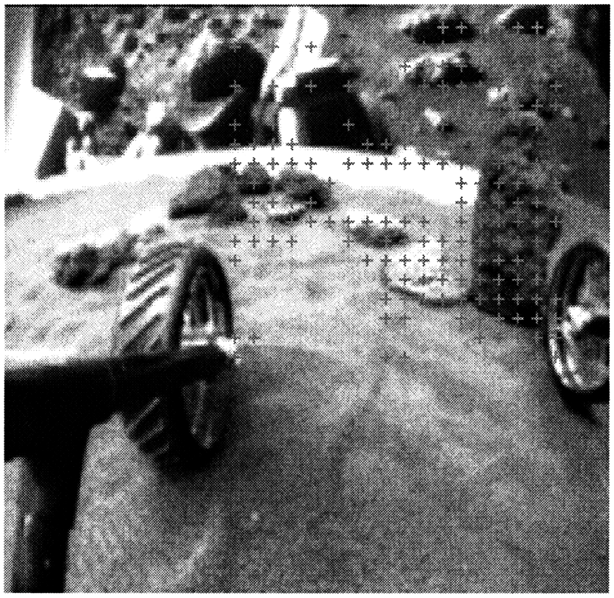


Figure 3.7: Points of Interest, Level 8

about what we are going to find - i.e. areas that are both texturally interesting and "dangerous", or close to the rover.

The density of the final range map is user specified. The initial Level 16 range map is computed with a pixel stepsize of 16, which at 2 meters from the rover corresponds to approximately 26 cm between adjacent range map points for our 256 x 243 stereo image pairs. Level 8 is the next step up in density, with Levels 4, 2, and finally, 1, becoming increasingly more dense.

Figure 3.8 shows how the range map has been computed adaptively such that the areas corresponding to the "ground" in texture are not computed. Likewise, it can be seen in the upper left corner that initially interesting points texturally are not further refined, as they are too far away to be either accurate in distance or immediately dangerous to the rover.

Figures 3.9 and 3.10 offer some comparison between the density of the final range maps generated, between Level 4 and Level 1.

### Adaptive Range Map Generation Summary

We can thus summarize the procedure of adaptively generating the range map based on texture and distance "danger" as follows:

1. Obtain stereo image pair (256x243, 8 bit grayscale)

2. Compute wavelet transform (2D Harr Basis) using the procedure given by [29]

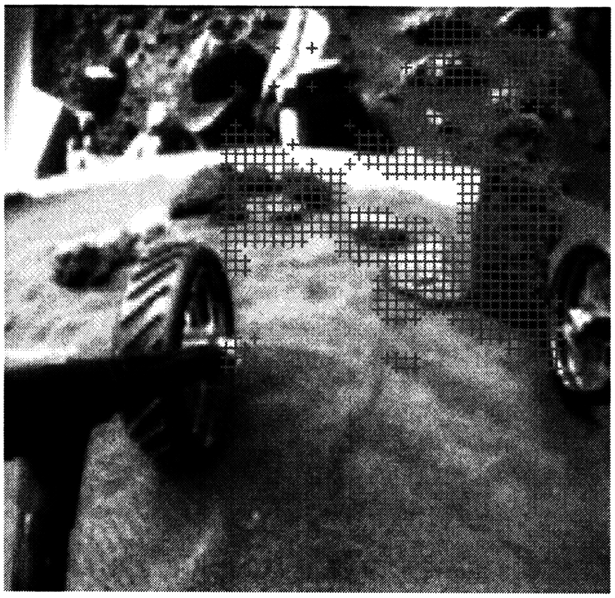


Figure 3.8: Points of Interest, Level 4

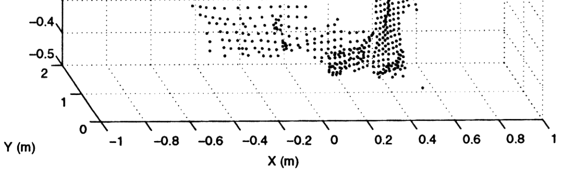


Figure 3.9: Generated Terrain Map (Level 4)

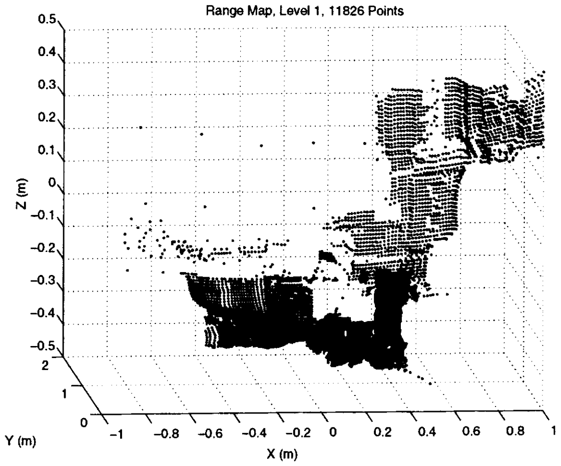


Figure 3.10: Generated Terrain Map (Level 1)

3. Set initial stepsize ($Stepsize_i = 16$) for iteration $i = 0$
4. Search for points of interest using equation 3.1 between some max and min row and column locations known to probably contain points of interest

LOOP ($i < MIN_I$):

5. Correlate points Using equation 3.2 and the fundamental matrix
6. Compute subpixel matches (optional, recommended)
7. Compute 3D locations using triangulation
8. Perform **Adaptive Mesh Refinement (see below) on "dangerous" points**
9. $i = i/2$

END LOOP

**Adaptive Mesh Refinement:**

**For each point (x,y) preserved as interesting from iteration i:**

1. Search for points of interest using equation 3.1 between $x \pm Stepsize_{i-1}, y \pm Stepsize_{i-1}$ using $(Stepsize_{i-1})/2$
2. Add any points found to running list of new interesting points
3. $Stepsize_i = (Stepsize_{i-1})/2$

### Timing and Performance

We have time tested the adaptive range map generation algorithm using the 172 image pairs gathered during our first autonomous navigation exercise.

At Level 4, the maximum time required to complete range map generation (AND path selection, as described in section 3.2) was 17 seconds (wall clock time) on a Sparc20, running as a user process, with an average time of approximately 9 seconds. At Level 8, the maximum time was approximately 5 seconds (wall clock time) on the same system, with an average time of approximately 2.5 seconds.

The same code timed on a Pentium 166 Mhz processor took 7.5 seconds at Level 4, and 1.9 seconds at Level 8, as measured by the profiler under Microsoft Visual C++.

These times reflect no optimization - we are presently transferring the math to binary point integer math, and expect speed increases on the order of approximately 2x, based on initial tests. Computation time is linear in the number of points computed.

The strength of the algorithm lies in the following features:

- Range map computed selectively, only for texturally interesting points, which automatically eliminates those areas where no correlation or an incorrect correlation would be found
- Range map computed adaptively, only for points which are close enough to be accurately located and of possible near-term danger to the rover
- De-warping and registering the image pairs is not required, as the epipolar geometry is used explicitly to find matching points.
- Easily implementable
- Low memory requirements

## 3.2 Obstacle Avoidance

Once we have adaptively computed the range map, we make use of the terrain information to perform obstacle avoidance. To do so, we sum the number of terrain points encountered when sweeping a box having an outline of the frontal area of the rover through 1.5 meters along the Y axis (directly ahead of the rover). We choose 1.5 meters as our lookahead distance because of the accuracy limitations imposed by low resolution images taken through 120° field of view lenses. We repeat this sum at several different rotation angles, examining the possible direction the rover might travel. The terrain point integration volume is depicted in figure 3.11.

The path having both the fewest number of encountered points and also the least amount of required rotation is chosen. If the absolute value of the rotation requested is less than 5 degrees, then the rover proceeds forward for 5 seconds

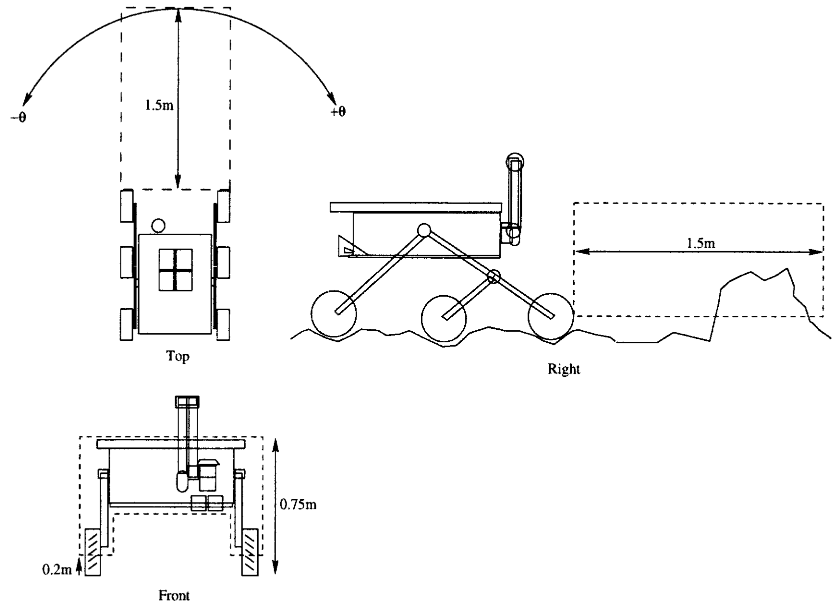


Figure 3.11: Reactive Obstacle Avoidance: Terrain Point Integration Volume

(approximately 20 cm). If the rotation is greater than +5 degrees, the rover rotates clockwise for 5 seconds, otherwise the rover rotates counterclockwise for the same period of time, which corresponds to a rotation of approximately 10 degrees. We find this to give us adequate obstacle avoidance performance.

Using only current information and the above strategy may lead to oscillatory behavior during obstacle avoidance maneuvers. The rover thus maintains a history of its past executed motions and initiates an oscillation-breaking maneuver (20 degree rotation) if it detects oscillatory behavior.

## 3.3 State Estimation

### 3.3.1 Vision Based State Estimation

At each iteration through our navigation algorithm, we compute a range map, which can be used to identify obstacles in the field of view of the rover. We can register these successive range maps such that they are all in a single frame of reference, effectively computing a global world map of the environment which the rover is traversing through. The process of registering these maps also has the advantage of providing an estimate of the rover's current state (position and orientation relative to it's initial frame of reference).

### Extraction of Inter-Frame Rotation and Translation

The algorithm we have implemented for the registration of two range maps is described in [32] and [34]. Here, a brief overview of the algorithm is presented. The interested reader is referred to [32] and [34] for additional details.

This iterative registration process minimizes a performance criteria, $\mathcal{F}$, given by

$$\mathcal{F}(\mathbf{R}, \mathbf{T}) = \frac{1}{\sum_{i=1}^{m} p_i} \sum_{i=1}^{m} p_i d^2(\mathbf{R}\mathbf{x_i} + \mathbf{T}, \mathbf{S}') \tag{3.6}$$

which is related to the difference in distance between the two clouds of 3D range points. In equation 3.6, $\mathbf{R}$ represents the rotation between the two range maps and $\mathbf{T}$ is the translation between the two range maps. Also in equation 3.6, the variable $p_i$ is 1 if the point $\mathbf{x_i}$ is matched to a point on the surface $\mathbf{S}'$ defined by the second set of 3D range points, and 0 otherwise, relating to points which are visible only from one frame or the other. The variable $m$ refers to the number of points in the first terrain map, and the function $d(\mathbf{x}, \mathbf{S}')$ is defined by

$$d(\mathbf{x}, \mathbf{S}') = \min_{j \in \{1, \ldots, n\}} d(\mathbf{x}, \mathbf{x}'_\mathbf{j}) \tag{3.7}$$

where $d(\mathbf{x}, \mathbf{x}')$ is the euclidean distance between two points.

We are then able to find a set of closest point matches between the two frames where the distances between the two points in a matched pair is less than an adaptively computed maximum distance tolerance $D^I_{max}$. The search for closest points is accomplished through the use of a *k-dimensional binary search tree* [32] for speed.

In each iteration we adapt the maximum distance tolerance based on the mean $\mu$ and standard deviation $\sigma$ of the distance between the points in each set of matched pairs which satisfied the maximum distance tolerance $D^{I-1}_{max}$ from the previous iteration. The mean and standard deviation are given by

$$\mu = \frac{1}{N} \sum_{i=1}^{N} d_i \tag{3.8}$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (d_i - \mu)^2} \tag{3.9}$$

where $N$ is the number of matched pairs and $d_i$ is the distance between the points in matched pair $i$.

We then adaptively specify a new $D^I_{max}$ based on the following criterion:

```
if $\mu < \mathcal{D}$
    $D^I_{max} = \mu + 3\sigma$
else if $\mu < 3\mathcal{D}$
    $D^I_{max} = \mu + 2\sigma$
else if $\mu < 6\mathcal{D}$
    $D^I_{max} = \mu + \sigma$
else
    $D^I_{max} = \xi$
```

We reject any points which do not satisfy the newly computed $D^I_{max}$ distance tolerance, and use the remaining matched pairs to compute the motion between the two frames. To compute the motion between the two frames, a dual quaternion methodology [30] [32] is used to solve optimally for the rotation and translation between the two frames simultaneously.

Once the motion between the two frames has been determined, the computed motion is applied to the first frame. We then iterate, finding the next set of closest point matches, updating the point matches using the statistically computed distance tolerance $D^I_{max}$, computing the motion between the two frames, and applying that motion to the first set of points, until the change in the computed motion reaches a lower threshold, or the algorithm reaches the maximum allowed number of iterations.

One variable referenced above is $\mathcal{D}$, which is computed to be the average distance between points and their closest neighbors in a terrain map, and is related is effectively the resolution of the terrain map data. The second variable, $\xi$, is computed from the point match set distance histogram. The reader is referred to [32] for a more thorough explanation of $\mathcal{D}$ and $\xi$, and their implications to the convergence of the algorithm.

### Experimental Validation

We now present experimental validation of our implementation of the algorithm described above, using terrain maps generated by the adaptive range map generation algorithm presented in Section 3.1. Figure 3.12 shows a terrain map generated in one position by the rover. Figure 3.13 shows a terrain map generated by the rover after a nominal 10° counterclockwise rotation.

Applying the range map registration algorithm to the above two range maps, with the following initial rotation matrix **R** and translation vector **T**:

$$\mathbf{R}_{\mathbf{initial}} = \begin{bmatrix} 0.984807 & -0.173648 & 0.000000 \\ 0.173648 & 0.984807 & 0.000000 \\ 0.000000 & 0.000000 & 1.000000 \end{bmatrix}$$

$$\mathbf{T}_{\mathbf{initial}} = \begin{bmatrix} 0.000000 & 0.000000 & 0.000000 \end{bmatrix}$$

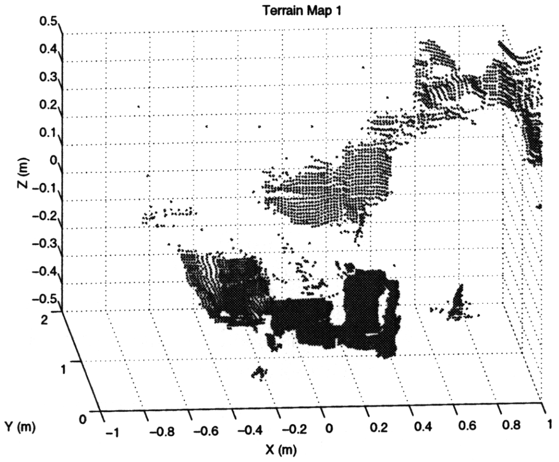


Figure 3.12: Range Map Position 1, 12,634 Points

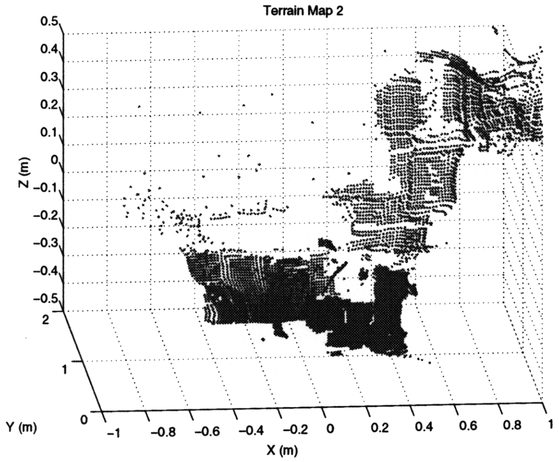


Figure 3.13: Range Map Position 2, 12,464 Points

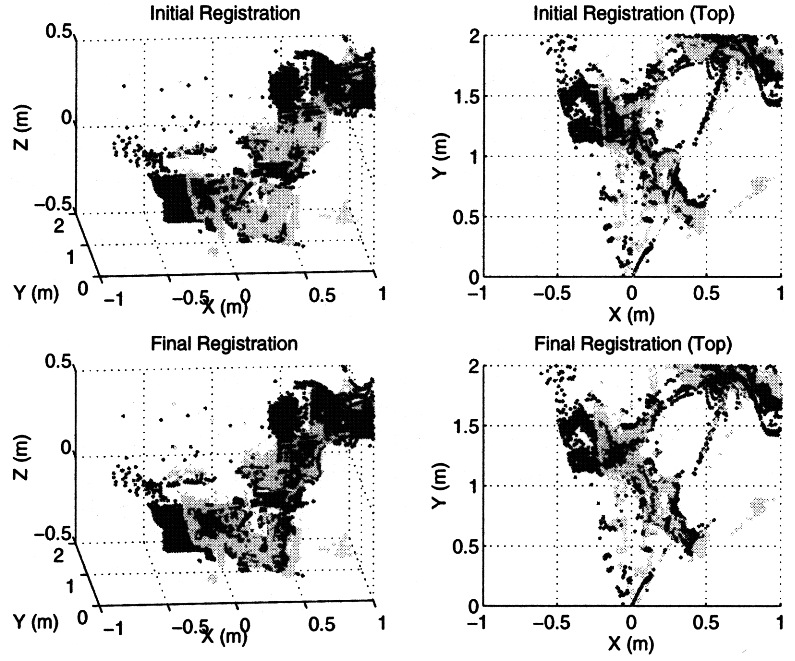


Figure 3.14: Initial Registration vs. Final Registration; Light Points: Terrain Map 1, Dark Points: Terrain Map 2; Computed Rotation: 8.26 degrees counterclockwise

The final computed rotation matrix **R** and translation vector **T** is computed to be:

$$\mathbf{R_{final}} = \begin{bmatrix} 0.989621 & -0.135732 & -0.047190 \\ 0.136498 & 0.990550 & 0.013388 \\ 0.044927 & -0.019690 & 0.998796 \end{bmatrix}$$

$$\mathbf{T_{final}} = \begin{bmatrix} -0.044082 & -0.026828 & 0.013626 \end{bmatrix}$$

The vehicle rotation computed from $\mathbf{R_{final}}$ results in an 8.26° counterclockwise rotation which corresponds well with the observed motion of the rover, and with the commanded rotation of 10.0° counterclockwise. Figure 3.14 shows the initial registration between the two terrain maps, after the application of $\mathbf{R_{initial}}$ and $\mathbf{T_{initial}}$ to terrain map 1, and the final registration between the two terrain maps, after the application of $\mathbf{R_{final}}$ and $\mathbf{T_{final}}$ to terrain map 1.

### 3.3.2 Formulation of the Extended Kalman Filter (EKF)

One method for obtaining the position and orientation of the rover relative to its starting position is to use the successive **R** and **T** estimates provided by the range map registration algorithm described in the previous section. Another method for obtaining the position state of the rover is to use dead reckoning (i.e. assuming that the rover achieved the 20cm forward motion or 10° nominal rotation).

However, both estimates of the rover's position and orientation will inevitably contain errors. The vision estimate may be degraded by poor metric camera calibration, as it takes the range maps which have been generated using that camera calibration as input. Similarly, the performance criterion in 3.6 will generally not converge to identically zero, and thus we stop the iterative algorithm when its rate of change becomes lower than a certain threshold. Likewise, the dead reckoned estimate will be seriously affected by the quality of terrain over which the rover is traversing. If it attempts to cross soft ground and is forced to climb over small rocks, its dead reckoned estimate will be generally very poor.

To improve the rover's final state estimate, we have implemented an extended Kalman filter [1] to fuse together the two state estimates to obtain a more optimal estimate. We describe the formulation of the Kalman filter in this section.

### 3.3.3 Dead-Reckoned Estimates

The state equations describing the rover's motion based on its (dead-reckoned) wheel odometry are given by equation 3.10. For simplicity, we will consider only in-plane translations and rotations.

$$\begin{aligned} \frac{d\mathbf{x}(\alpha)}{d\alpha} &= \begin{bmatrix} \frac{dX(\alpha)}{d\alpha} \\ \frac{dY(\alpha)}{d\alpha} \\ \frac{d\phi(\alpha)}{d\alpha} \end{bmatrix} = \begin{bmatrix} R_w \cos\phi(\alpha) \\ R_w \sin\phi(\alpha) \\ \frac{R_w}{B} u \end{bmatrix} \\ &= \mathbf{f}(\mathbf{x}(\alpha), u) \end{aligned} \tag{3.10}$$

In equation 3.10, $\mathbf{x} = [X \; Y \; \phi]^{\mathrm{T}}$ represents the position and orientation of the rover relative to a global reference frame, generally taken to be the rover's starting position in the environment. The rover's nominal wheel radius is defined to be $R_w$, with $B$ defined as half the distance between the rover's wheel base. The variable $\alpha$ is defined by equation 3.11.

$$\alpha = \frac{\theta_r + \theta_l}{2} \tag{3.11}$$

In equation 3.11, $\theta_r$ and $\theta_l$ are the absolute rotations of the right and left wheels, respectively. The variable $u$ is defined by equation 3.12.

$$u = \frac{d\theta_r - d\theta_l}{d\theta_r + d\theta_l} \tag{3.12}$$

Similarly, in equation 3.12 $d\theta_r$ and $d\theta_l$ are the differential left and right wheel rotations, respectively.

For LSR, which is a skid-steered rover, the state equations given in equation 3.10 are valid all maneuvers except for turning-in-place, which is exactly the method by which we perform rotation of the rover. During a turn-in-place maneuver, the variable $\alpha$ becomes zero (as $\theta_l = -\theta_r$), and the state equations become singular, (as $d\alpha = 0$). Thus, to handle turn in place maneuvers, we redefine $\alpha$ as

$$\alpha = \frac{\theta_r - \theta_l}{2} \tag{3.13}$$

which results in a new set of state equations used during the turn in place maneuver, as may be seen in equation 3.14.

$$\frac{d\mathbf{x}(\alpha)}{d\alpha} = \begin{bmatrix} \frac{dX(\alpha)}{d\alpha} \\ \frac{dY(\alpha)}{d\alpha} \\ \frac{d\phi(\alpha)}{d\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{R_w}{B} \end{bmatrix} = \mathbf{f}(\mathbf{x}(\alpha)) \tag{3.14}$$

In a rover system with wheel encoders, the rover's wheel rotations could be measured over time and used in the numerical integration of the above state equations to produce the dead-reckoned estimates of the rover motion. However, as LSR is not equipped with wheel encoders, and moves in discrete increments, we will not perform numerical integration of the state equations to obtain the rover's dead reckoned estimate, but rather use the commanded motion.

Similarly, the estimation error covariance matrix, $\mathbf{P}(\alpha)$, as part of the Kalman filter is propagated after each motion by the rover using equation 3.15.

$$\frac{d\mathbf{P}(\alpha)}{d\alpha} = \mathbf{F}(\alpha)\mathbf{P}(\alpha) + \mathbf{P}(\alpha)\mathbf{F}(\alpha)^{\mathrm{T}} + \mathbf{Q} \tag{3.15}$$

In equation 3.15, $\mathbf{F}(\alpha)$ is the Jacobian of the state equations and $\mathbf{Q}$ is the covariance matrix associated with the process noise. The process noise covariance matrix is assumed to be diagonal, i.e. $\mathbf{Q} = \mathrm{diag}[Q_{XX}, Q_{YY}, Q_{ZZ}]$, and represents the confidence placed in the dead-reckoned estimate of the state, in a qualitative sense.

### 3.3.4 Kalman Filter Update via Vision

Using the terrain map registration technique described in section 3.3.1, inter-frame rotation and translation of the rover may be extracted from two successive vision-based range maps, given some initial knowledge of the commanded motion.

In the framework of the Kalman filter, the $\mathbf{R}$ and $\mathbf{T}$ generated by the range map registration technique will be considered a measurement of the rover's state.

We will then fuse this measurement with the dead-reckoned state estimate, obtaining a more accurate estimate of the rover's state. The measurement of the rover state is given by equation 3.16.

$$\mathbf{z}(\alpha_i) = \begin{bmatrix} X(\alpha_{i-1}) + \Delta X(\alpha_i) \\ Y(\alpha_{i-1}) + \Delta Y(\alpha_i) \\ \phi(\alpha_{i-1}) + \Delta\phi(\alpha_i) \end{bmatrix} = \mathbf{h}(\mathbf{x}(\alpha_i)) \tag{3.16}$$

In equation 3.16, $\Delta X$, $\Delta Y$, and $\Delta\phi$ are computed directly from the $\mathbf{R}$ and $\mathbf{T}$ which are computed as detailed in section 3.3.1. Additionally, $\alpha_i$ represents the value of $\alpha$ at the end of the stereo pair acquisition → terrain map generation → range map registration cycle (prior to making the next move) and $\alpha_{i-1}$ is the value of $\alpha$ at which the previous vision-based update to the Kalman filter was computed.

The update to the rover state is thus given by

$$\begin{aligned} \hat{\mathbf{x}}(\alpha_i) &= \hat{\mathbf{x}}(\alpha_i|\alpha_{i-1}) + \\ &\quad \mathbf{K}(\alpha_i)[\mathbf{z}(\alpha_i) - h(\hat{\mathbf{x}}(\alpha_i|\alpha_{i-1})] \end{aligned} \tag{3.17}$$

with the Kalman gain, $\mathbf{K}(\alpha_i)$, being given by equation 3.18.

$$\begin{aligned} \mathbf{K}(\alpha_i) &= \mathbf{P}(\alpha_i|\alpha_i - 1)\mathbf{H}^{\mathrm{T}} * \\ &\quad [\mathbf{H}\mathbf{P}(\alpha_i|\alpha_i - 1)\mathbf{H}^{\mathrm{T}} + \mathbf{R_{mnc}}]^{-1} \end{aligned} \tag{3.18}$$

In equation 3.18, $\hat{\mathbf{x}}$ refers to the Kalman filtered estimate of the rover state. $\mathbf{H}$ is the Jacobian of the measurement equation, 3.16, which is the $3 \times 3$ identity matrix in this case. The measurement noise covariance matrix, $\mathbf{R_{mnc}}$, is assumed to be a diagonal matrix, i.e. $\mathbf{R_{mnc}} = \mathrm{diag}[R_{mnc_{XX}}, R_{mnc_{YY}}, R_{mnc_{ZZ}}]$, and represents the confidence placed in the vision-based estimates of the state, in a qualitative sense.

The estimation error covariance matrix is updated via

$$\mathbf{P}(\alpha_i) = (\mathbf{I} - \mathbf{K}(\alpha_i)\mathbf{H})\mathbf{P}(\alpha_i|\alpha_{i-1}). \tag{3.19}$$

In equation 3.19, and in equation 3.18, $\mathbf{P}(\alpha_i|\alpha_{i-1})$ represents the propagated estimation error covariance matrix resulting from the integration of equation 3.15. Similarly, in equation 3.17, $\hat{\mathbf{x}}(\alpha_i|\alpha_{i-1})$ represents the propagated state estimates resulting from the integration of the rover state equations given either by equation 3.10 or by equation 3.14.

The experimental performance of the EKF-derived state estimator described in this section is presented in section 4.

The performance of the filter is greatly affected by the assumed process noise covariance matrix, $\mathbf{Q}$, and the measurement noise covariance matrix, $\mathbf{R_{mnc}}$, as it is in any Kalman filtered application.

This is especially true for the case of a planetary rover where terrain style (e.g. soft soil, hard-packed ground, etc.) can change the quality of the dead reckoned state estimates through wheel slip and other factors. Similarly, the number of features in the landscape will affect the quality of the vision based state estimate through the ruggedness, and hence quality of registration, of the terrain maps.

Therefore, for the results presented in this paper, the state-estimate gains are defined using equation 3.3.4.

$$G_X = \sqrt{\frac{R_{XX}}{Q_{XX}}},\ G_Y = \sqrt{\frac{R_{YY}}{Q_{YY}}},\ G_\phi = \sqrt{\frac{R_{\phi\phi}}{Q_{\phi\phi}}}$$

This allows the relative weight given to the dead-reckoned state estimate with respect to the vision-based state estimate to be varied more easily, by reducing the number of tunable parameters.

## 3.4 Navigation

Once the rover's state estimate is available, navigation to a user designated goal becomes possible. This section describes the methods by which we perform real-time path planning based on a potential-flow based path planning methodology capable of dealing with dynamic environments [7], using the EKF-based state estimate.

### 3.4.1 Potential Flow Based Navigation and Obstacle Avoidance

As an alternative to simply performing reactive obstacle avoidance, as described in section 3.2, we can also navigate to a user designated goal and avoid (either autonomously located or user designated) obstacles along the way. To do this, we have implemented a real-time path planner based on potential flow methodologies [7]. During each iteration of the navigation algorithm, this path-planner computes the local flow velocity (i.e. the streamline direction at the rover's location in the environment), to which the rover aligns itself or travels along to reach the desired goal. Currently, we limit path planning to 2D and obstacles to be limited to cylinders or ellipses at some specified angle, although arbitrarily shaped objects may be modeled using panel-approximations to the object's boundary.

#### Obstacle Modeling

Here we present a brief treatment of the potential flow based path planning methodology described more fully in [7]. The interested reader is reader is referred there for more details.

If we consider the rover workspace as the complex plane, the potential for a plane flow at an angle $\alpha$ with the real axis is given by

$$w = -Uze^{-i\alpha}\ . \tag{3.20}$$

The potential for a circular cylinder centered at $z = X_0 + iY_0$ with radius $r$ in a plane flow with velocity $U$ at an angle $\alpha$ with the real axis is given by

$$w = -U(ze^{i\alpha} + \frac{r^2 e^{i\alpha}}{z - z_0}) \tag{3.21}$$

Finally, the flow around an ellipse with major axis $a$ and minor axis $b$ may be obtained using a conformal mapping as given by

$$z = \xi + \frac{r^2}{4\xi} \tag{3.22}$$

Applying the conformal mapping given by equation 3.22 to equation 3.21, we obtain the equation for a cylinder in the $\xi$ plane, as may be seen in equation 3.23.

$$w = -U(\xi e^{-i\alpha} + \frac{(a+b)^2 e^{i\alpha}}{4\xi}) \ . \tag{3.23}$$

If we then solve for $\xi$, we obtain the following

$$\xi = \frac{1}{2}(z \pm \sqrt{z^2 - r^2}), r^2 = a^2 - b^2 \tag{3.24}$$

whose solution describes the velocity field around the ellipse.

### Potential Construction

We define the external plane flow using where $X_f$ and $Y_f$ are the $(X, Y)$ goal position, and $X$ and $Y$ are the rover's current position.

$$w = -Uze^{-i\alpha}, \alpha = \tan^{-1}(\frac{Y_f - Y}{X_f - X}) \tag{3.25}$$

An approximate harmonic potential solution to the continuous potential flow is achieved by the superposition of the closed form potentials for the plane flow and all of the individual obstacles added to the flow. More details on the construction of this harmonic potential are provided in [7].

### Rover Motion

Once the harmonic potential is constructed, the $u$ and $v$ components of the flow can be found using equation 3.26.

$$\begin{array}{rcrcr} u = & \frac{\partial \phi}{\partial X} = & \frac{\partial \psi}{\partial Y} = & -\Re(\frac{dw}{dz}), \\ v = & \frac{\partial \phi}{\partial Y} = & -\frac{\partial \psi}{\partial X} = & \Im(\frac{dw}{dz}). \end{array} \tag{3.26}$$

The $u$ and $v$ components of the flow define a vector in the rover workspace. The potential flow based path planning methodology was originally developed for an omni-rover. However, our rover may only move in the direction which it is facing. Thus, we must align the pointing angle of the rover with the current

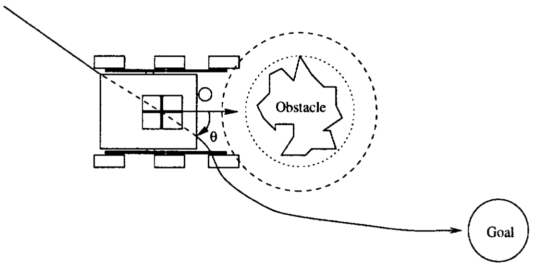


Figure 3.15: Traveling Along A Streamline

streamline vector. Once the rover has rotated to be nominally aligned with the streamline by making 10° nominal rotations, (in our implementation, the pointing error must be less than 5°), it may proceed forward, in steps of 20 cm nominal. This process is depicted in figure 3.15.

Currently obstacle locations are defined *a priori* by the user in the global reference frame of a ceiling mounted camera with 120 degree field of view, which we will refer to as "the sky camera." This camera and its calibration are described more fully in section 4.3 However, given a manipulator-mounted or mast-mounted stereo pair on a rover platform, all obstacles could be located either by the user, or located automatically by processing the terrain map generated by this stereo pair, all in the reference frame of the rover. We plan to pursue this avenue in the future.

### State Estimate Error and Expanding Obstacles (Roving in Dynamic Environments)

It is explicitly the ability of the real-time path planner to deal with dynamic environments which allows its application to the context of rover navigation. If all obstacle locations are not known *a priori*, then incremental discovery of obstacles and their addition to the world potential flow requires a path planner capable of dealing with dynamic environments. In other words, the rover must be capable of replanning its path to the goal in real time in response to newly discovered obstacles which invalidate it's previously planned path.

We can further exploit the dynamic features of the potential flow based path planning algorithm by making use of the rate of change of the estimation error covariance matrix. If the rover is bounded by an estimation error ellipse which is changing size at some rate $\delta r$ then we may define the obstacles in the environment to also be expanding at the same rate $\delta r$, such that the path taken by the rover is guaranteed never to intersect any of the known obstacles, regardless of the error in the rover's state estimate.

Both incremental introduction of obstacles into the environment and the use of the estimation error covariance matrix to expand obstacle boundaries to maintain rover safty are topics which we plan to pursue in the future.

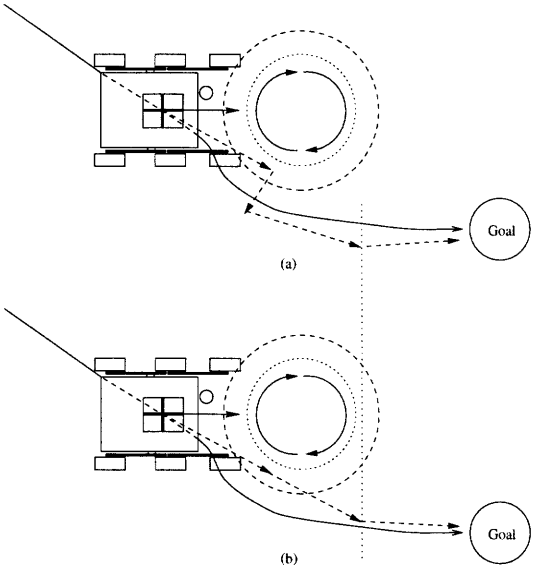


Figure 3.16: Coping With Finite Stepsizes

### Dealing With Finite Stepsizes and NonHolonomic Properties

As our rover is commanded to move in discrete increments, there exists the possibility that it may jump "inside" the sphere of influence of an object. The stream line along which the rover is commanded to travel then rotates by some large value, so move the rover quickly outside of the object. However, the rover is a nonholonomic vehicle, not an omni-rover. Therefore, the rover must rotate through a very large angle to move along the streamline and out of the obstacle, before again rotating through a similar large angle to continue on toward the goal. We may possibly deal with this problem in the manner depicted in figure 3.16, which we plan to explore in the future.

In figure 3.16 (a), we see that as the rover approaches the obstacle on the right side, if it is not precisely aligned with the streamline (as in the case where it is able to make discrete rotations), it will move inside the obstacle boundary on the next movement. The streamline it is asked to align itself to is then nearly orthogonal to its present direction of motion, requiring a large rotation before moving outside the obstacle, and then another large rotation to realign itself with the streamline.

Rather than using this method to move to the goal, we could look effectively one time step ahead to see where the final goal of the motion is, as shown in figure 3.16 (b), and take an intercept angle from the present position, rather than simply following the streamline.

Of course, the problem is also solved if we move continuously, rather than in discrete jumps, and were able to align ourselves to the streamline more precisely with finer rotational control.

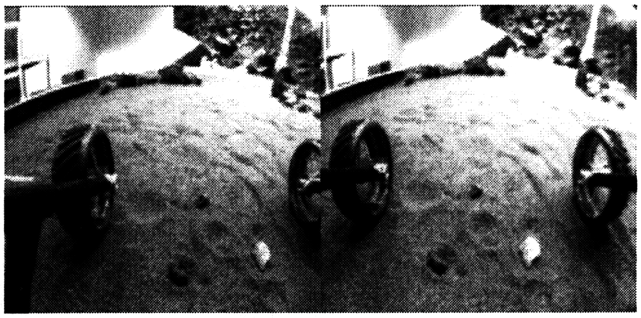


Figure 3.17: Autonomous Sample Acquisition: Starting Image Pair

## 3.5 Sample Segmentation and Acquisition

While autonomously navigating from point to point and avoiding obstacles in the environment, the rover will invariably come across small rock or soil samples which are sufficiently "interesting" enough to warrant collection.

An example stereo pair containing possibly interesting samples may be seen in figure 3.17

As these small samples may not initially be visible given images available to the rover command crew, the rover should, in essence, be "on the lookout" for these "interesting samples", and collect them if they are considered "interesting enough."

This level of autonomy introduces two difficulties. First, how does the rover effectively segment collectable samples from the background environment? and second, how does the rover determine if the samples are "interesting"?

We address the first question in sections 3.5.1 through 3.5.4, and the second question in section 3.5.5.

### 3.5.1 Texture Based Segmentation

Using the texture signature described in section 3.1.2, we can step over the image with a stepsize of two pixels in the area between the wheels of the rover, corresponding to the "maximum manipulability" region for the rover mounted manipulator, detailed in figure 3.18. As before, we have precomputed the ground texture such that this selection of interesting points is adaptive to the current terrain.

The results of this interest point selection, or initial segmentation of possible samples from the background, may be seen in figure 3.19.

As may be seen in figure 3.19, the texture segmentation alone is sensitive to the lighting and variations in the terrain, such that it tends to be overly cautious and select an abundant number of interesting points. However, it gives a useful first pass on which we may refine our segmentation.

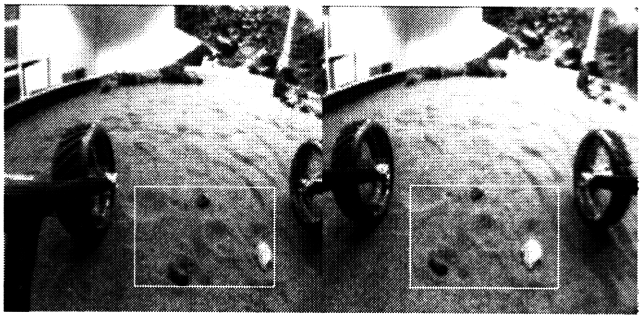


Figure 3.18: Sample Search Region

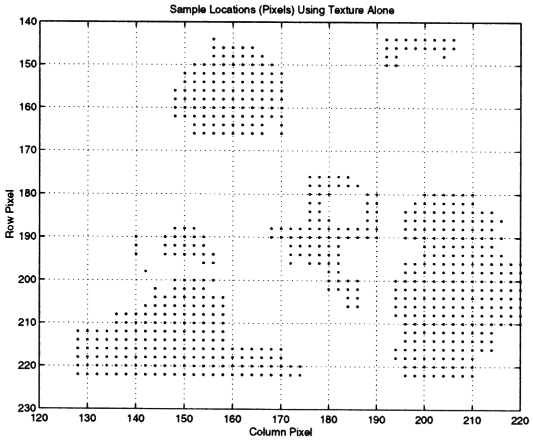


Figure 3.19: Texture Only Sample Segmentation: Interest Point Locations (Pixels)

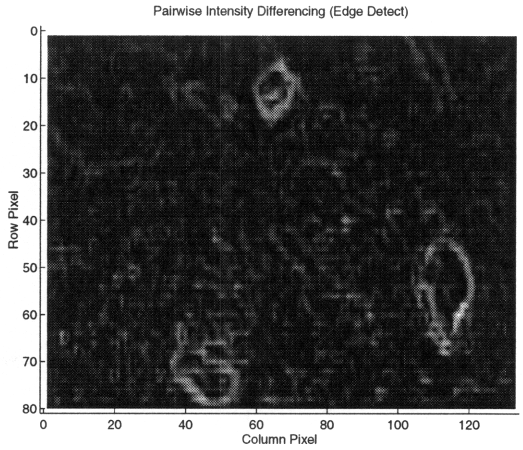


Figure 3.20: Raw Edge Detection

### 3.5.2 Edge Detection

Computing the horizontal and vertical intensity difference between adjacent pixels in the original (non-wavelet-transform) left image gives a rough edge detection, as may be seen in figure 3.20.

To make this algorithm adaptive to the current terrain, we pass a 20x20 window over the image, in the same adaptation area used by the texture segmentation algorithm for computation of the ground texture. If we compute the average intensity of the edge detected image over that 20x20 window, we are able to obtain some idea of the "edginess" (or average intensity gradient) of the terrain we are currently occupying.

Using this average intensity value, we can then threshold the raw edge detection values to obtain a binary representation, as shown in figure 3.21.

A window of size 20x20 pixels was chosen after determining that a sample small enough to be acquired by the rover generally fit into an area of size 20x20 pixels, when inside workspace of MicroArm-1 and viewed from the rover mounted navigation cameras.

### 3.5.3 Correlation of Texture Segmentation and Edge Detection

For each point marked as interesting during the texture-segmentation phase, we then sum the intensity of the edge detected image over a 20x20 window centered on that point. If the average edge detected intensity over that window is greater than some constant multiplied by the ground edginess, then the point is preserved as an interesting point. Otherwise, it is eliminated from consideration.

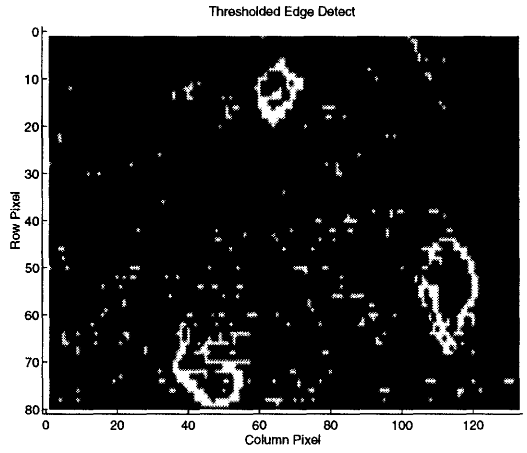


Figure 3.21: Thresholded Edge Detection

| Sample | X (m) | Y (m) | Z (m) |
|---|---|---|---|
| Top | 0.0539 | 0.3958 | -0.3836 |
| Right | 0.1659 | 0.2325 | -0.3942 |
| Left | -0.0179 | 0.2135 | -0.3961 |

Table 3.1: Segmented Samples, 3D Locations

### 3.5.4 Blobification and Sample Acquisition

The final retained, texture/edge detect-correlated points, which were originally selected using a stepsize of two pixels are then interpolated. By interpolated, we refer to the process by which if two or more retained points are separated by only one nonvalid pixel, then that pixel is marked as a valid point also. This in effect also tends to smooth the resultant correlated points. The results of this correlation may be seen in figure 3.22.

Once interpolated, the points are then subjected to "blobification", or the process of individual blobs of points being numbered for retrieval, using a recursive blob coloring algorithm, which also calculates the centroid of each blob [9]. The pixel at this location is correlated with the right image to obtain the corresponding right image point. The point pair is then used to triangulate the 3D position of the sample.

The 3D sample locations computed for the samples in figure 3.17 are listed in table 3.1.

The 3D position of the sample can then be used, in conjunction with the inverse kinematics for the manipulator arm, which may be found in appendix A to acquire each numbered sample and store it in the cache, as may be seen in figure 3.23.

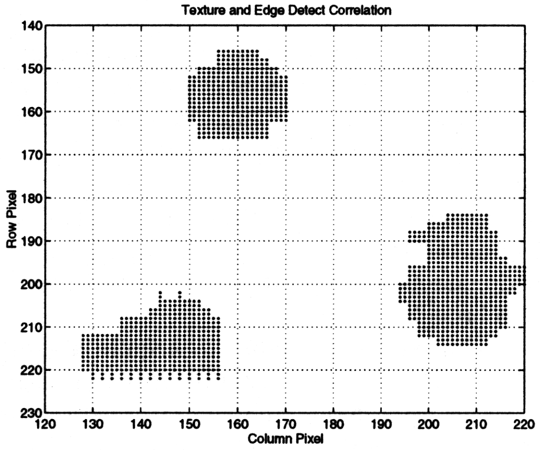


Figure 3.22: Texture and Edge Detect Correlation

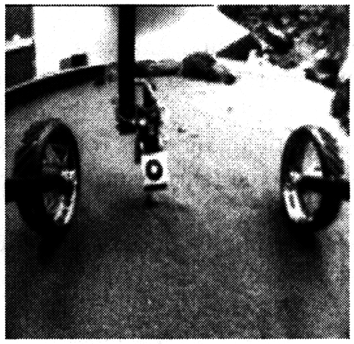


Figure 3.23: LSR Acquiring an Autonomously Segmented and Selected Sample

### 3.5.5 What Constitutes Interesting?

Until this point we have concentrated on the problem of how to segment a sample from the background and localize it autonomously for acquisition. The second, perhaps more important question is: when is a sample interesting enough to acquire?

In the context of a year long mission over tens to hundreds of kilometers, a rover may come across many samples which could be acquired. However, the space available in its cache is limited. Thus, there is an obvious tradeoff between picking up something now and waiting until later in hope that something more interesting might be found.

A very detailed simulation study of this problem has been done [28]. However, independent of the knowledge of this study, we have implemented a simpler, salient functional relationship, which takes into account many of the salient points which were brought to light.

Effectively, we can model the desire to acquire a sample as a functional relationship between the euclidean distance in texture signature between the current sample we are considering and the recorded texture signatures of all other samples we have already acquired, weighted by the estimated space left available. Thus, it is easy to pick up samples early in the mission, when the cache is relatively empty, but as time progresses, the samples must be increasingly interesting to warrant being collected. This relationship is presented in equation 3.27, where $\beta$ is a tunable parameter setting the falloff rate, and $v$ is the estimated percentage of volume used in the cache can.

$$AcquireInterest = (min_i\sqrt{(H-H_i)^2+(D-D_i)^2+(V-V_i)^2})e^{(-\beta v)} \quad (3.27)$$

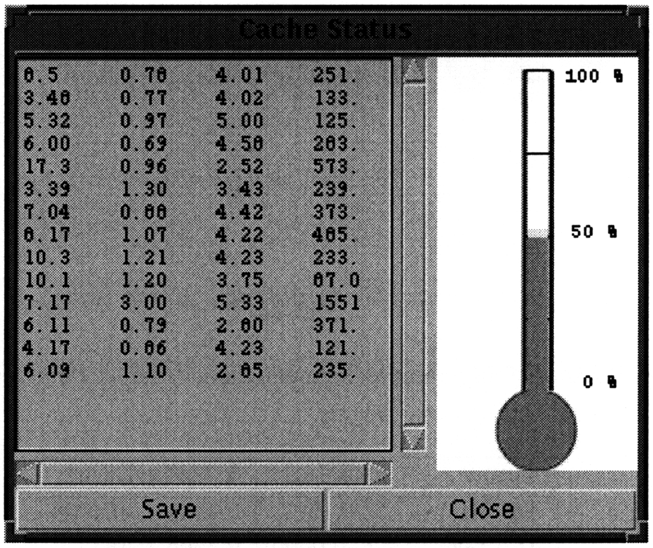
Thus, for a sample to be acquired, the value of AcquireInterest must simply pass a user specified threshold. This threshold is generally tuned based on the relative abundance of samples in the environment, and the quality to which samples are homogeneous in texture. Likewise we can normalize the euclidean distance by some expected average texture difference between samples, such that the value provided by equation 3.27 is nominally approximately 1.0 for an empty cache, set 1.0 to be the threshold, and thus rocks must be more interesting than the norm in order to be collected (this results in more selective behavior by the rover).

An example texture signature list and cache can representation may be seen in figure 3.24.

### 3.5.6 Denser Sample Distributions

The small samples seen in figure 3.17 are fairly well defined against the background and fairly widely spaced. Thus, we might ask what happens if we place the samples more closely together, as shown in figure 3.25.

Computing the texture-only segmentation of the image results in the blobs shown in figure 3.26.



```
8.5     0.78    4.01    251.
3.48    0.77    4.02    133.
5.32    0.97    5.00    125.
6.00    0.69    4.58    283.
17.3    0.96    2.52    573.
3.39    1.30    3.43    239.
7.04    0.88    4.42    373.
8.17    1.07    4.22    485.
10.3    1.21    4.23    233.
10.1    1.20    3.75    87.0
7.17    3.00    5.33    1551
6.11    0.79    2.80    371.
4.17    0.86    4.23    121.
6.09    1.10    2.85    235.
```

Figure 3.24: Cache Status, Showing H,D,V Texture Signature, Pixel Area, and Approximate Cache Volume Filled

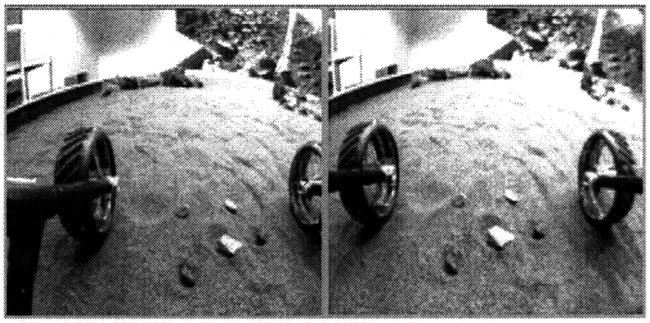


Figure 3.25: Autonomous Sample Acquisition: Starting Image Pair

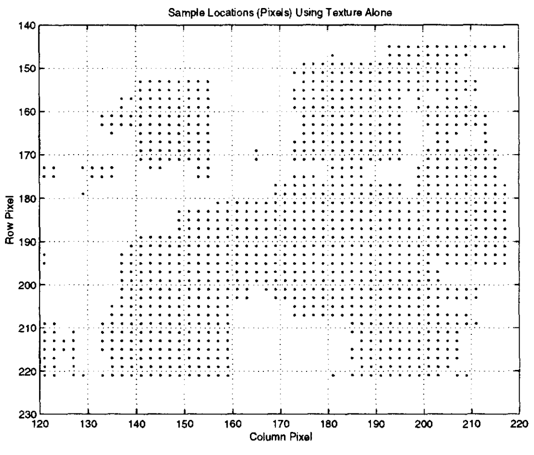


Figure 3.26: Texture Only Sample Segmentation: Interest Point Locations (Pixels)

| Sample | X (m) | Y (m) | Z (m) |
|---|---|---|---|
| Top Right | 0.1517 | 0.4245 | -0.4063 |
| Top Left | -0.0090 | 0.3765 | -0.3823 |
| Middle Right | 0.1152 | 0.2752 | -0.3906 |
| Bottom Left | -0.0144 | 0.2226 | -0.3995 |

Table 3.2: Segmented Samples, 3D Locations

Computing the raw edge detect on the image results in figure 3.27.

Figure 3.28 details the result of the thresholding operation on the edge detection.

Finally, correlating the thresholded edge detection with the texture-only segmentation results in the sample segmentation shown in figure 3.29.

The 3D sample locations computed for the samples in figure 3.25 are listed in table 3.2.

As we can see, the segmentation algorithm also performs well on more closely spaced samples, although it does consider the most closely placed sample pair to be a single sample.

## 3.6 Other Algorithms

In addition to those algorithms which we have mentioned previously which apply to terrain map generation, state estimation, navigation, and sample acquisition, we have implemented a number of user initiated autonomous algorithms. These algorithms, of course, could also be integrated with our algorithm for sample segmentation, such that they are run rather than the sample acquisition and cache routine.

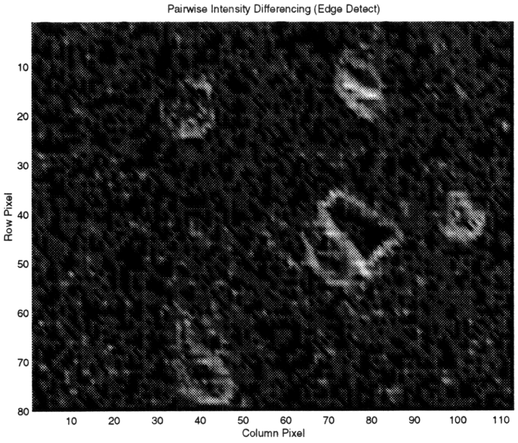


Figure 3.27: Raw Edge Detection

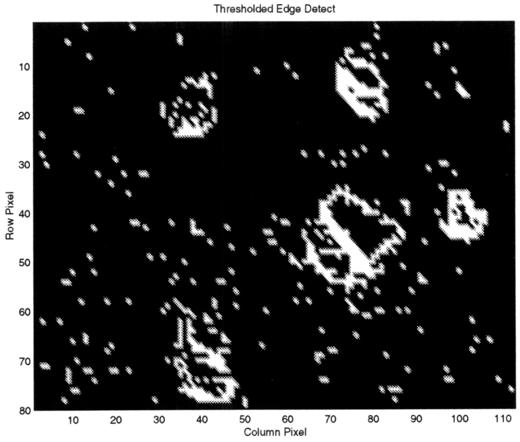


Figure 3.28: Thresholded Edge Detection

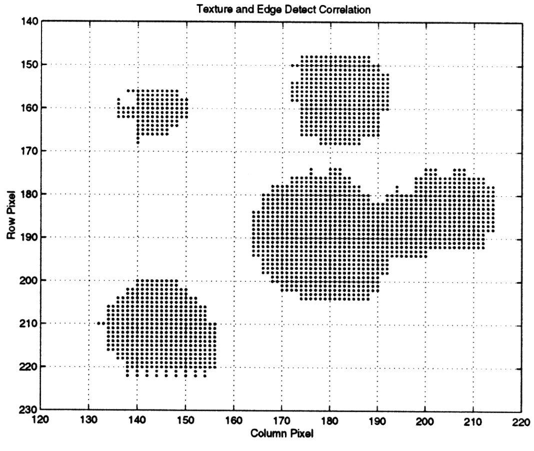
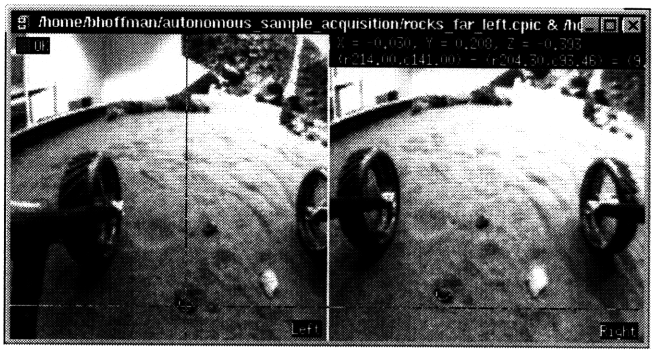


Figure 3.29: Texture and Edge Detect Correlation


/home/bhoffman/autonomous_sample_acquisition/rocks_far_left.cpic & /h

X = -0.030, Y = 0.208, Z = -0.393

(r214.00,c141.00) - (r204.30,c93.46) = (9

Figure 3.30: User Interface for Sample Manipulation

To initiate any of the following algorithms, the remote user first obtains a stereo image pair from the rover's navigation cameras. Using this image pair, the remote user then designates a sample, as shown in figure 3.30, which the rover mounted manipulator (MicroArm-1) will interact with.

Some possible interactions which have been implemented include close range sample imaging, sample abrasion using the end-effector mounted abrading tool, science instrument deployment and placement against the sample, soil trenching, and of course, user initiated sample acquisition and cache.

Results for the implementations of these activities are presented in section 4.7

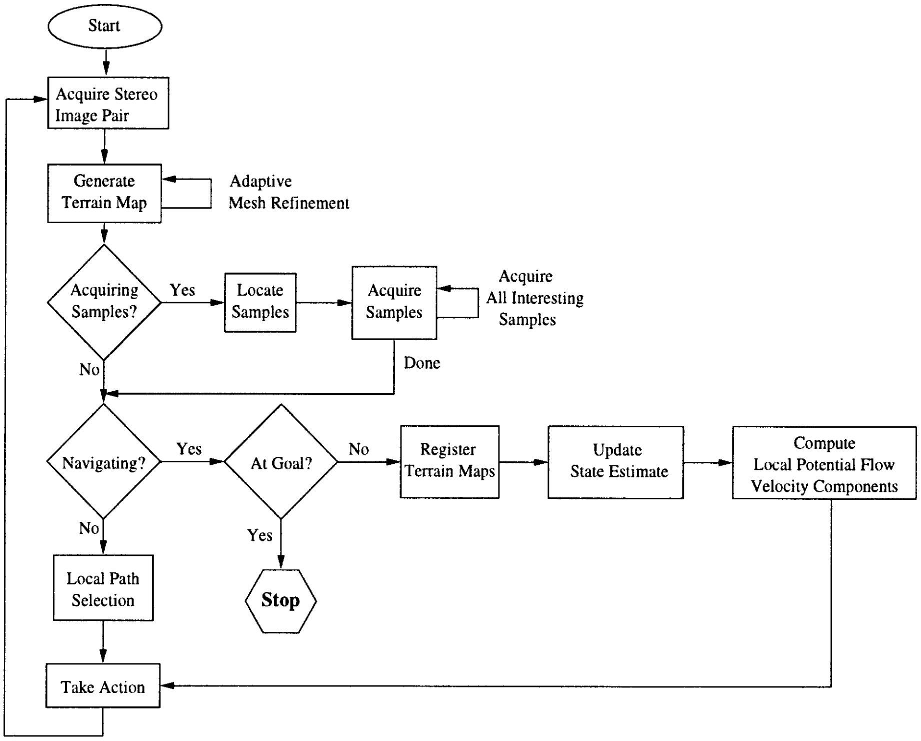


Figure 3.31: Command Cycle Showing Algorithm Interconnections

## 3.7 Algorithm Suite Interconnections and Summary

In this chapter we have presented a variety of algorithms, which we have implemented and tested experimentally, of use in increasing autonomy in planetary rover operations. We now step back and view the interconnections between the individual algorithms and are better able to gain a sense of perspective as to how they all fit together.

### 3.7.1 Interconnections

The interconnections between the various algorithms described in this chapter can best be seen in terms of a flow chart through a navigation time-step, as depicted in figure 3.31.

Several switches in the algorithm flow can be seen in this diagram. For example, the user may switch between reactive obstacle avoidance for roaming behavior and navigation to a user designated goal. Likewise, sample acquisition may be turned on or off.

Similarly, some parts of the flow may be replaced with other modules, e.g.

the module where we acquire samples may be replaced with viewing samples or abrading samples, or any of the other sample manipulation algorithms described in section 3.6, which all (except soil trenching) may act on the output of the sample segmentation and location procedure.

We view the depicted algorithm structure as a reasonable framework in which the 2001/2003 Mars rover missions could perform their mission successfully.

### 3.7.2 Summary

Overall, we have presented a suite of algorithms which will enable a planetary rover, such as those which will be used in NASA's planned 2001/2003 Mars rover missions, to function in a highly autonomous fashion. This will enable, to a large part, the rover to accomplish desired mission objectives such as sample acquisition and cache, and exploration of large terrain areas. The ground is then able to more effectively intervene, on a more infrequent basis, to specify desired navigation goals, or desired samples to interact with.

We now present the experimental results for the algorithm suite which we have successfully implemented.

# Chapter 4

# Experimental Results

The algorithms described in Chapter 3 have all been implemented and tested on the LSR / MicroArm-1 platform. Here, we present our experimental results.

First, we present the methods used to calibrate the LSR navigation cameras and state estimate ground truthing cameras in sections 4.1 through 4.3. We then present our results for autonomous roaming obstacle avoidance behavior in section 4.4. The results from potential flow based path planning and navigation to both unobstructed and unobstructed goals are presented in section 4.5. Finally, we present the results for our autonomous sample segmentation and acquisition algorithms in section 4.6. The results for all other sample manipulation routines are presented in section 4.7.

## 4.1 Camera Calibration

In order to perform operations which make use of a mapping between a pair of camera image points and a 3D point, or a single camera image point and a 2D point, we must first discover this mapping (generally nonlinear) through the process of camera calibration.

Once we have this mapping, we can perform metric reconstructions (of the environment, for example), using the image data provided by the cameras. This allows us to generate 3D terrain maps, as described in section 3.1, perform sample manipulation, as described in section 3.6, and also to provide ground truthing for the rover's state estimate in the environment.

We describe the method used to calibrate the LSR stereo navigation camera pair in section 4.2, and our method of calibrating the camera used in ground truthing the rover's state estimate in section 4.3.

## 4.2 Navigation Camera Calibration

To effectively calibrate the LSR stereo navigation pair, the stereo cameras were shown several black calibration fixtures having white dots of varying diameters

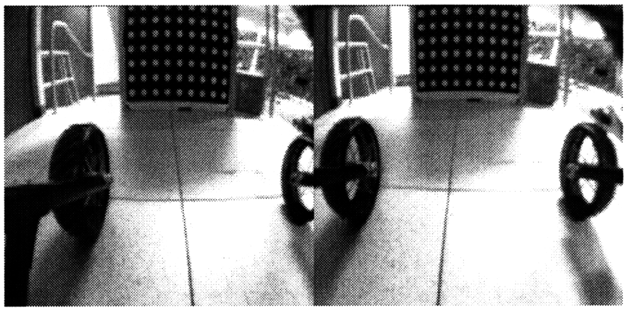


Figure 4.1: Sample stereo calibration pair.

in 34 different positions to provide input data for a least-squares calibration technique which attempts to generate a camera model compensating for radial lens distortion [11]. Given the large field-of-view of our cameras, radial lens distortion was quite significant. An example pair of calibration images may be seen in figure 4.1.

Given images with a resolution of 256 columns by 243 rows, we found that the camera calibration was accurate to approximately 2 cm at a distance of approximately 0.3 m from the navigation cameras, and approximately 10 cm at a distance of approximately 1.3 meters from the navigation cameras. Generally, the accuracy of the camera calibration improves with proximity to the cameras, as expected.

## 4.3 Ground Truth (Sky Camera) Calibration

To measure the performance of our state estimation and navigation routines effectively, we require a method by which we may obtain the ground truth position measurement for the rover.

To address this problem we mounted a single SuperCircuits CCD camera to the ceiling of the laboratory, pointing directly downward and centered on the rover workspace. Using this camera, fitted with a 120° field of view lens, we are able to capture nearly the entirety of the rover workspace.

We then placed a calibration fixture on the floor of the laboratory, having a $9 \times 9$ grid of white, 2 cm diameter calibration points on a black background, with point spacing of 10 cm. The location of each calibration point in 2D laboratory space, and its corresponding pixel location are extracted using the software developed for 3D camera calibration by [11] from a series of images where the calibration fixture is moved to cover the entire surface of the laboratory floor. A sample calibration image may be seen in figure 4.2. The extracted 2D spatial locations of the calibration points may be seen in figure 4.3. The corresponding pixel locations for these points may be seen in 4.4, where the degree of radial lens distortion present when using 120° field of view lenses is readily apparent.

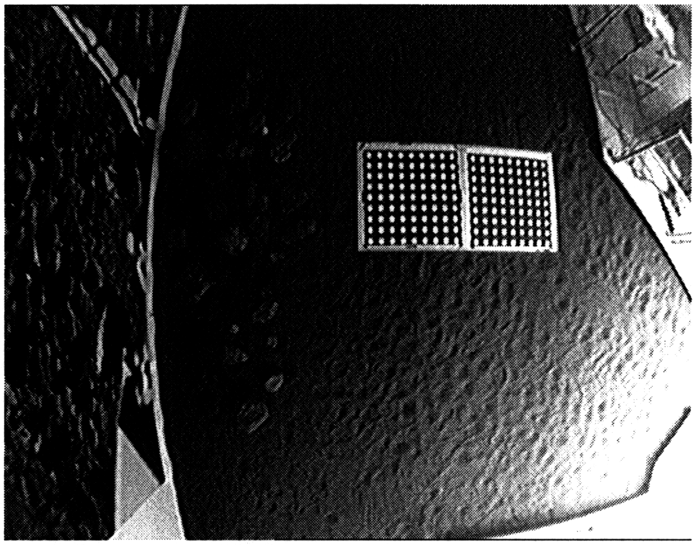


Figure 4.2: Sample Image from Ground Truth Camera Calibration

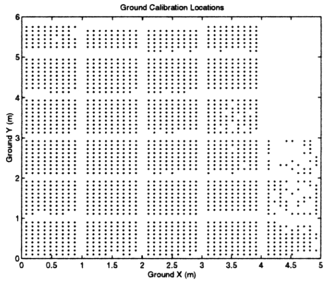


Figure 4.3: Ground Truth Points

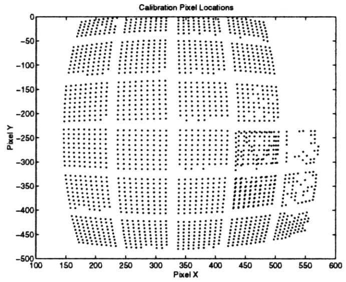


Figure 4.4: Corresponding Pixel Points

We then fit third degree polynomials to the data to determine the functional mapping between pixel column number and the X position in the laboratory coordinate frame in meters. Similarly, we find the mapping between pixel row and laboratory frame Y position. Of course, we also compute the inverse functional mapping, so that we may overlay a graphical representation of the rover on the ground truth images, based on the rover's state estimate. These functions may be seen in figures 4.5, 4.6, 4.7, and 4.8.

As we have a relatively accurate fit to the calibration data, as may be seen from figures 4.5 through 4.8, the accuracy of the ground truth position estimate should be on the order of the resolution of the image, or approximately 5 cm. The ground truth data for each of the rover traverses was extracted by hand by selecting the center of a white cross pattern on the top of the rover. However, in the future a template matching or correlation based approach could be used

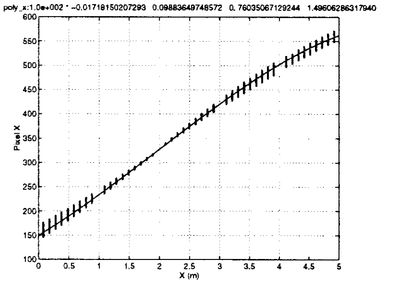


Figure 4.5: Pixel Column to Ground Truth X (m) Mapping

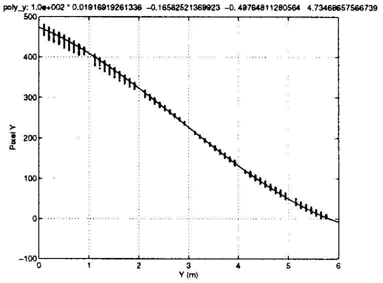


Figure 4.6: Pixel-Row to Ground Truth Y (m) Mapping

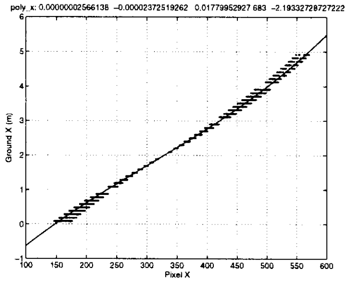


Figure 4.7: Ground Truth X (m) to Pixel Column Mapping

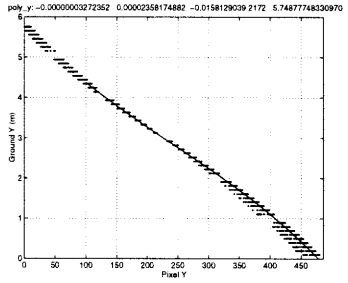


Figure 4.8: Ground Truth Y (m) to Pixel-Row Mapping

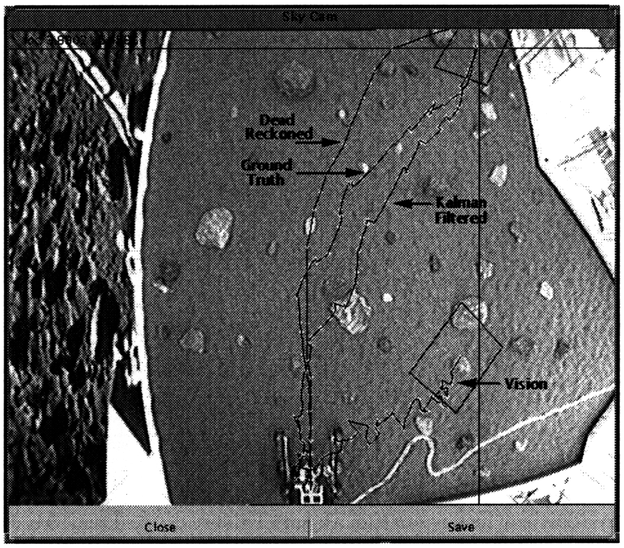


Figure 4.9: Kalman Filtered State Estimate Using Reactive Obstacle Avoidance

effectively, given that the rover's internal state estimate may be used to reduce the search area in the image significantly, ultimately saving a great deal of time in the extraction of the ground truth measurements over the manual method.

## 4.4 Obstacle Avoidance Results

To experimentally test the rover's wandering behavior, i.e. reactive obstacle avoidance, we released the LSR rover into its environment, pictured in figure 4.9. It successfully avoided obstacles as it wandered from its starting position to the upper right of the rover workspace. While wandering, the rover maintained an estimate of its position and orientation relative to its initial position. It was not instructed to acquire samples during this run.

As can be seen in Figure 4.9, the Kalman filtered vehicle state estimate is still quite good after completing a traverse of approximately 6 meters. The final error between the filtered state and the ground truth is approximately 0.22 meters, with a maximum absolute error over the entire traverse of 0.64 meters, and a mean absolute error of 0.38 meters. The vehicle made 79 iterations through the reactive obstacle avoidance algorithm, making 20 right turns (10° nominal), 11 left turns (10° nominal), and 48 forward motions (20 cm nominal). Figure 4.10 details the position error of the Kalman filtered estimate with respect to the ground truth. The gains used in the Kalman filter were $G_X$: 1.9, $G_Y$: 1.9, $G_\phi$:1.2.

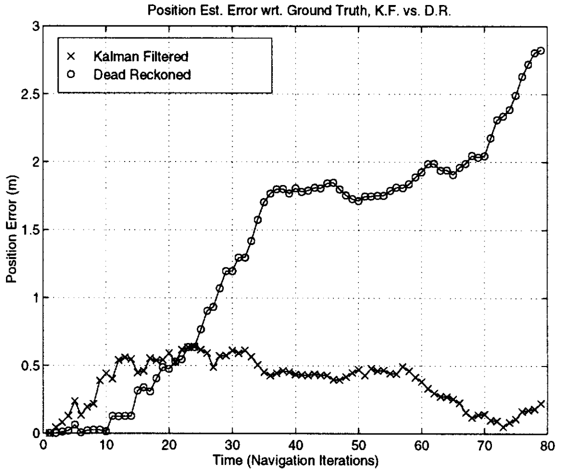


Figure 4.10: Position Error (m); Kalman Filtered vs. Dead Reckoned; Reactive Run

## 4.5 Navigation Results

In all experimental trials of the potential flow based path planning and navigation methodology, the obstacles and goal were specified *a-priori*, using the overhead ground truthing camera, as described in section 3.4.1. However, as previously mentioned, these obstacles could also be specified using a manipulator mounted stereo pair or extracted from an incrementally generated world terrain map.

Figure 4.11 details the path taken by the LSR rover in its navigation through soft soil-simulant and over medium sized rocks to a nearly directly accessible goal. The final error between the filtered state and the ground truth was 0.63 meters, with a maximum absolute error over the whole traverse of 0.76 meters, and a mean of 0.40 meters. The vehicle made 125 iterations through the potential flow based navigation algorithm, making 41 right turns, 44 left turns, and 40 forward motions. The gains used in the Kalman filter for this first run were $G_X$: 1.9, $G_Y$: 1.9, $G_\phi$: 1.2.

Figure 4.12 details the LSR rover's traverse around an obstruction to a goal. In the context of the sample return mission, the same software, implemented on the Sample Return Rover (SRR) testbed, will be used to navigate around obstructions to its desired goal, the "dead" LSR rover. The final error between the filtered state and the ground truth was 0.67 meters, with a maximum absolute error over the whole run of 1.09 meters and a mean of 0.59 meters. The vehicle made 69 motions, with 8 right turns, 22 left turns, and 39 forward motions. The gains used in the Kalman filter for the second run were $G_X$: 5.0, $G_Y$: 5.0, $G_\phi$: 2.5.

Table 4.1 summarizes our results for all of the experimental trials presented in this section.

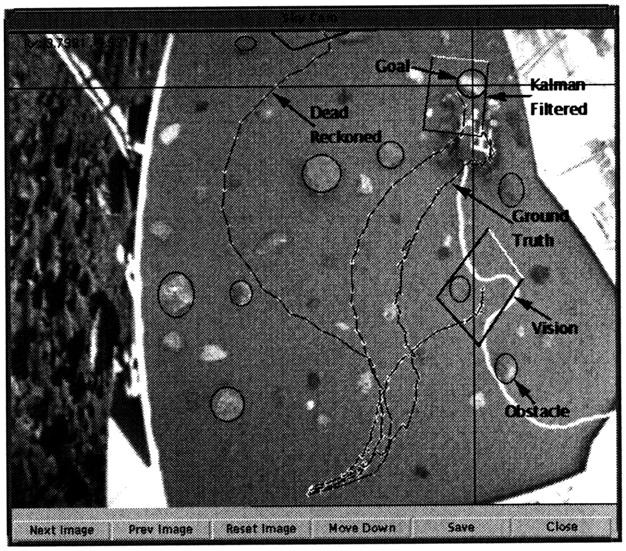


Figure 4.11: Kalman Filtered State Estimate Using Potential Flow-Based Path Planning: Unobstructed Goal

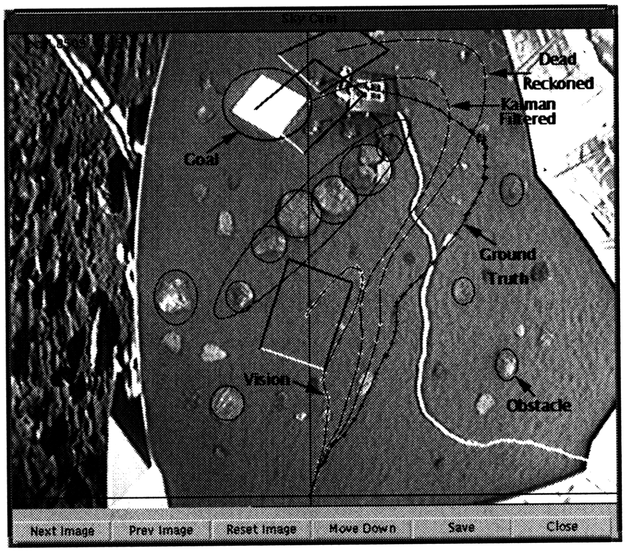


Figure 4.12: Kalman Filtered State Estimate Using Potential Flow-Based Path Planning: Obstructed Goal

| Trial | Kalman Filtered<br>Max/Mean | Dead Reckoned<br>Max/Mean |
|---|---|---|
| Reactive | 0.68/0.38 m | 2.83/1.34 m |
| Unobstructed | 0.76/0.40 m | 2.99/1.38 m |
| Obstructed | 1.09/0.59 m | 1.34/0.70 m |

Table 4.1: Estimation results for all experimental trials.

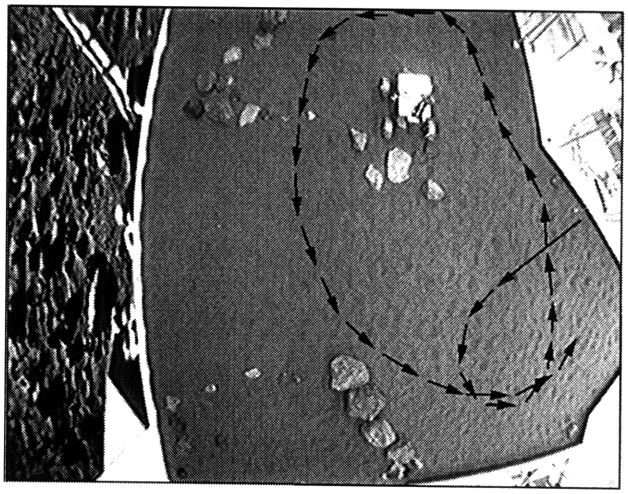


Figure 4.13: Approximate Path Taken During Sample Acquisition Trial

## 4.6 Sample Acquisition Results

To experimentally test the effectiveness of the image segmentation algorithm for sample acquisition, as described in section 03.5, we released the rover into its environment, configured as shown in figure 4.13.

The approximate path followed by the rover is overlayed on figure 4.13. The absolute position via. the ground truth camera is not available, as this camera was not installed at the time this trial was conducted.

While traversing the environment and performing reactive obstacle avoidance, the rover encountered a variety of small rocks which were previously randomly distributed. The rover was instructed to be looking for samples and to attempt to cache any of these which it found to be interesting.

The rover made 172 iterations through its terrain map generation → sample segmentation and acquisition → obstacle avoidance loop, acquiring 172 stereo image pairs which it attempted to segment, looking for samples.

The results of its attempts at identifying samples in the workspace using the sample segmentation algorithm described in section 3.5 are presented in table 4.2.

From table 4.2, we can see that the segmentation algorithm correctly classi-

| | Correct<br>Sample | Correct<br>No Sample | Q | False<br>Positive | False<br>Negative |
|---|---|---|---|---|---|
| Segmentation<br>Performance | 13<br>7.56% | 148<br>86.0% | 4<br>2.33% | 3<br>1.74% | 4<br>2.33% |

Table 4.2: Sample Segmentation Results

fies 93.6% of the images. Moreover, the most detrimental result, a false positive, from the standpoint of wasted time and energy, accounts for only 1.74% of the results.

## 4.6.1 Selected Images

Additionally, we show here the correctly identified samples, the false positives (sample found where there is no sample), and the false negatives (sample exists but not found). We will omit the correctly classified true negative (no sample exists and none was found), as these are the majority of the 172 image pairs and are generally less interesting.

### Correct Sample

The images where our segmentation algorithm correctly segmented out the target sample from the soil background are shown in figures 4.14 and 4.15.

### False Positive

The images where a sample was falsely detected (false positive) are shown in figure 4.16, accounting for 1.74% of the results.

### False Negative

The images where a sample went un-noticed are shown in figure 4.17, accounting for 2.33% of the results.

### Questionable

Finally, we include a questionable section, where the rover showed what might be deemed questionable judgment in its classification. The images which resulted in a questionable segmentation are shown in figure 4.18, accounting for 2.33% of the results.

# 4.7 Other Sample Manipulation Results

Figures 4.19 through 4.23 show the LSR/MicroArm-1 platform successfully engaging in a variety of user- initiated autonomous activities.

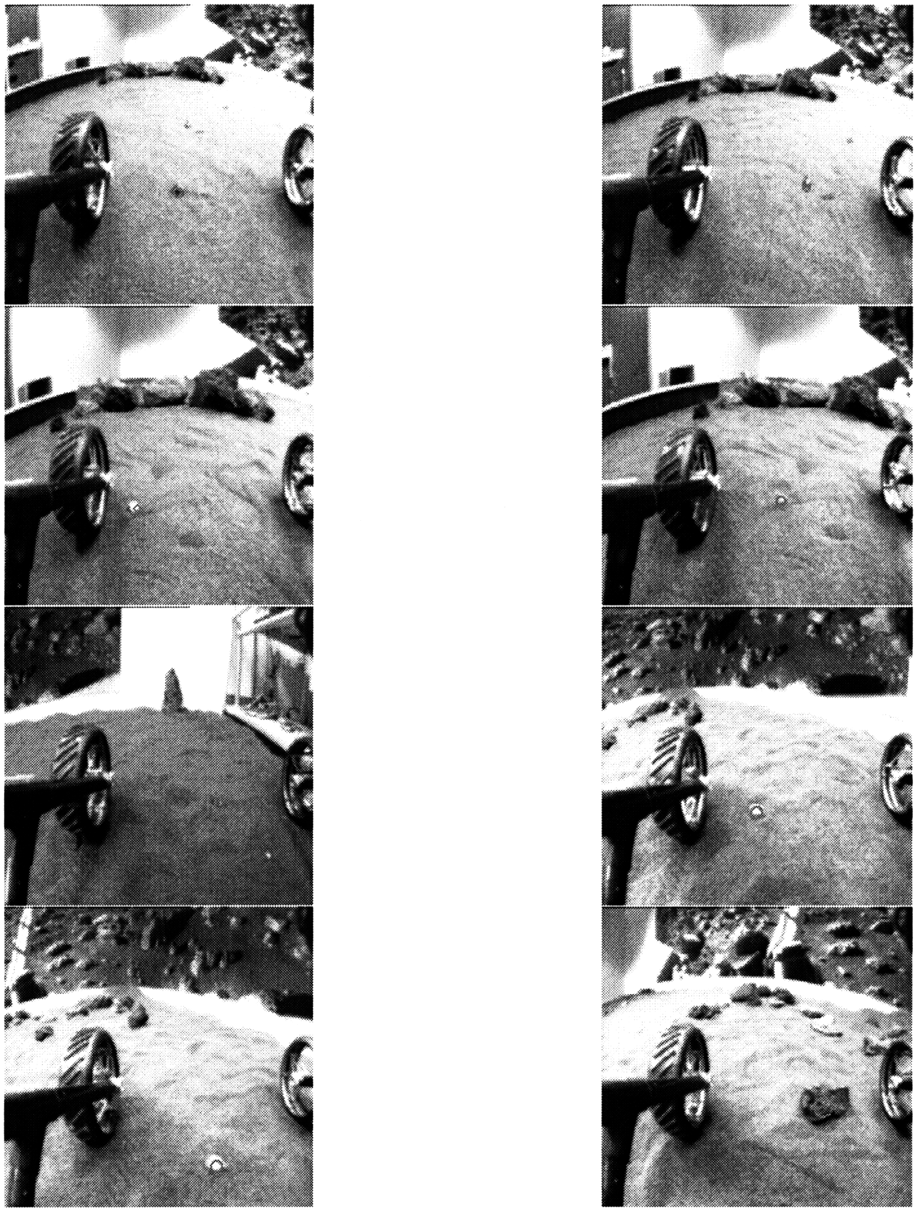


Figure 4.14: Correct Segmentation (1-8)

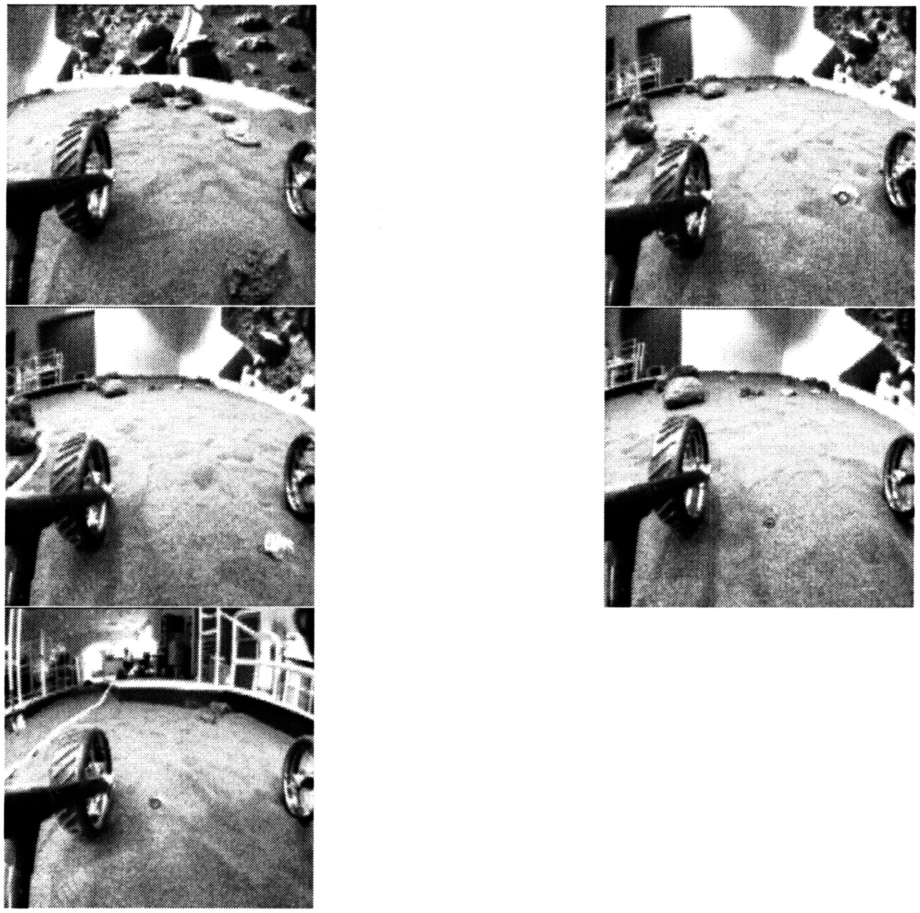


Figure 4.15: Correct Segmentation (9-13)

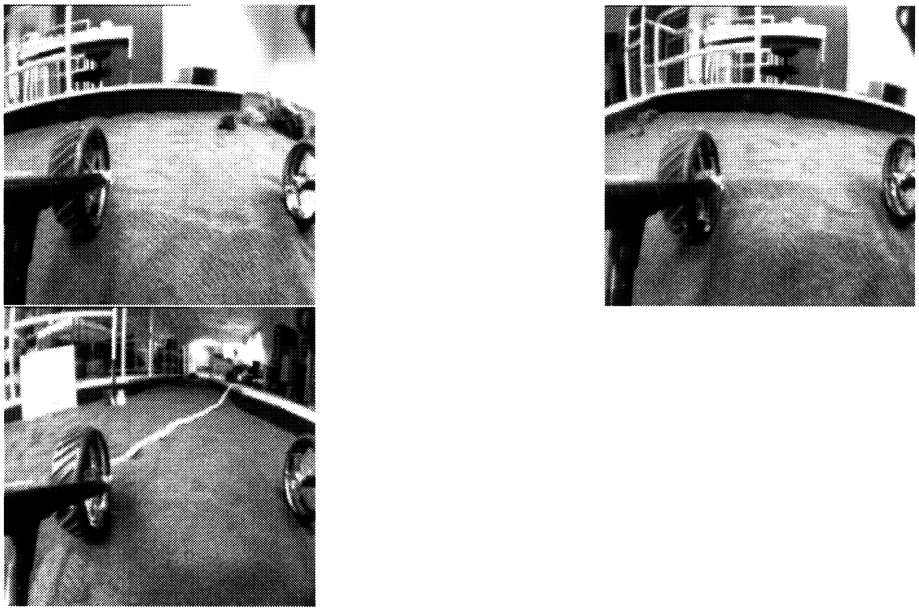


Figure 4.16: False Positive Segmentation

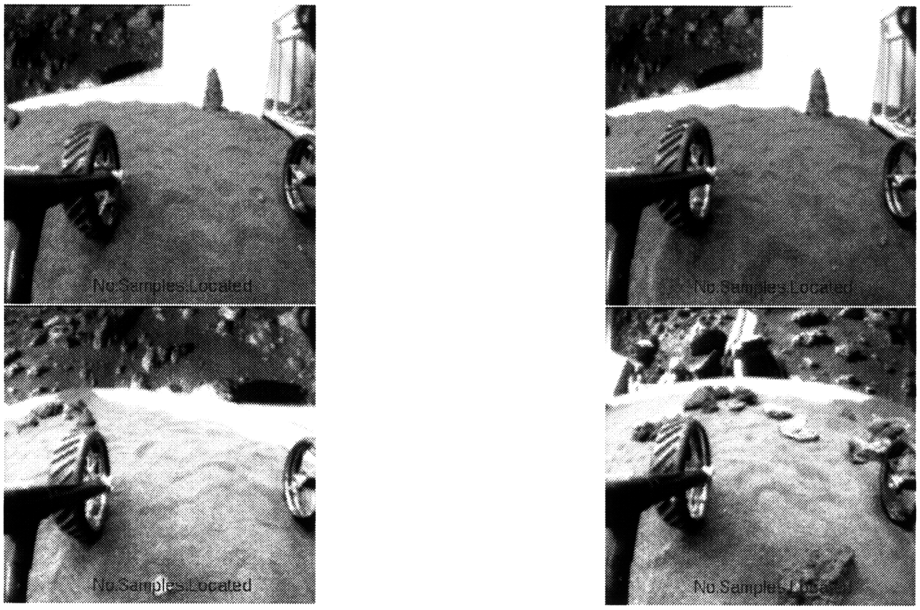


Figure 4.17: False Negative Segmentation

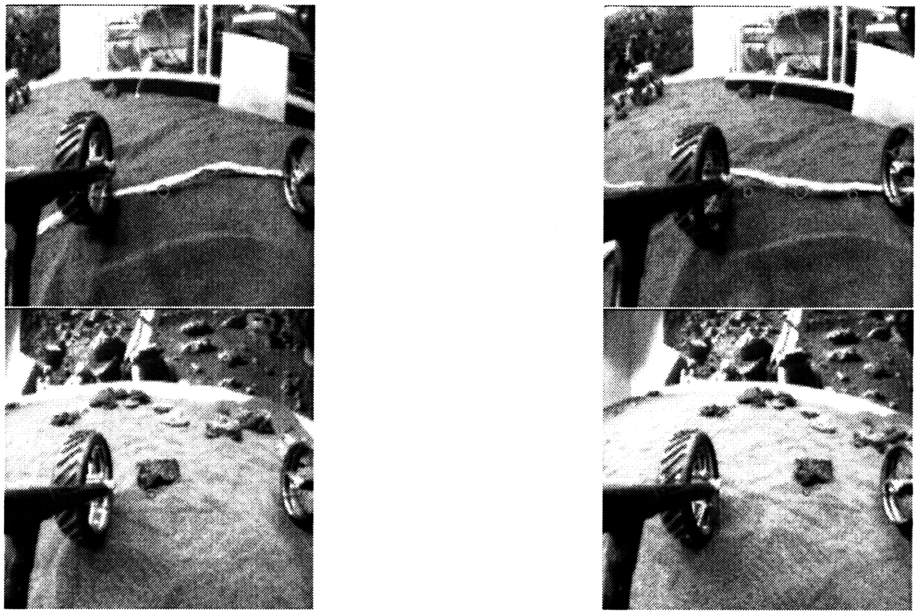


Figure 4.18: Questionable Segmentation

These autonomous manipulation routines all rely on simple inverse-kinematics. As such, they generally performed with reasonable success given the limitations of the hardware.

The configuration of the MicroArm-I end-effector, being similar to a clamshell in design, often results in "pressing" behaviors on the surface of the sand, rather than "cutting" or "scooping" behaviors, which would have been more effective in acquiring small rock samples. As such, sample soil-interaction routines (sample acquisition, soil trenching) were generally successful approximately 75samples with respect to the soil surface.

MicroCamera Imaging and Science Instrument Deployment were successful approximately 95positioning errors being related to errors in camera calibration.

Sample abrasion succeeded effectively approximately 60time, due to the precision positioning required to place the end effector rotary tool repeatedly against the rock sample to be abraded.

Visual verification of autonomous activity results may be introduced in the future.

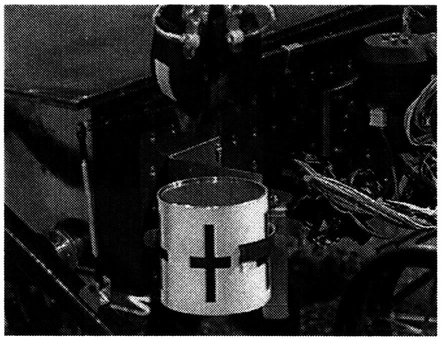


Figure 4.19: Sample Acquisition

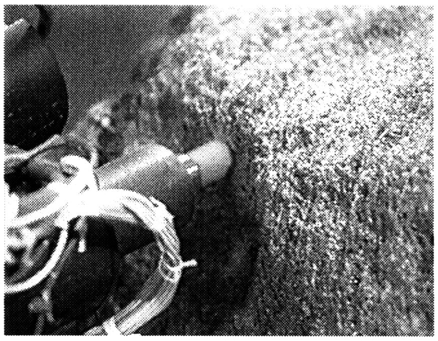


Figure 4.20: Sample Abrasion

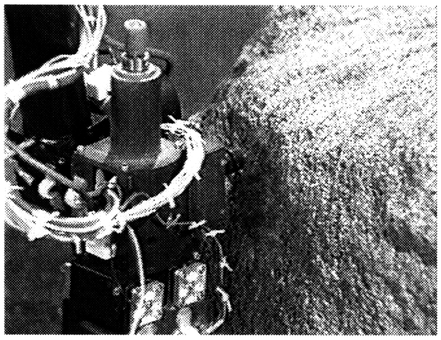


Figure 4.21: MicroCamera Imaging

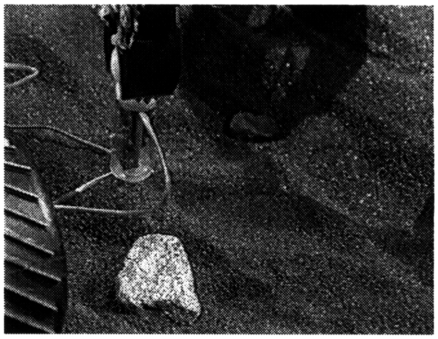


Figure 4.22: Instrument Deployment

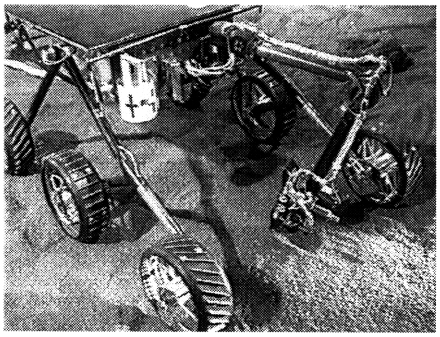


Figure 4.23: Soil Trenching

# Chapter 5

# Discussion

In this chapter we discuss the performance of our algorithms, including both their strengths and weaknesses, and any implementation or testing caveats. Additionally, we highlight what additions might increase the performance of the described algorithms. Comparisons with other algorithms, or with algorithms from which these algorithms were derived are also made.

## 5.1 Terrain Map Generation

### 5.1.1 Algorithm Performance

We have tested our semi-sparse texture-distance-adaptive range map generation algorithm on several hundred image pairs as our rover has navigated through our laboratory environment, with very good results.

At Level 8, a terrain map generally contains approximately 100 points. At Level 4, a terrain map generally contains approximately 400 to 700 points. Level 2 and Level 1 have correspondingly larger numbers of points. Of course, the actual number of points selected by the algorithm is dependent on both the texture quality of the ground and the number of texturally interesting features in the environment which are close to the rover. However, the user does have control over the approximate number of points generated by selecting the maximum iteration depth (Level number) for the algorithm.

### 5.1.2 Implementation Caveats

In our current implementation of the adaptive terrain map generation algorithm, the map generation occurs with reasonable rapidity and computational efficiency, requiring approximately 2 seconds at Level 8, and 5 seconds at Level 4, on an unloaded Sparc 20. Given that the implementation is currently not optimized and uses floating point arithmetic, we expect to see large improvements in speed when all arithmetic operations are converted to integer.

### 5.1.3 Testing Caveats

Unfortunately, due to the nature of the experimental hardware (i.e. the tether between the LSR rover and the system VMEbus), on which the terrain map generation system was implemented, we were unable to test the algorithm extensively on totally natural imagery. However, we have no reason to expect that it should have reduced performance when operating on natural imagery.

### 5.1.4 Comparison With Other Methods

In the context of planetary rover operations, the passive stereo approach to the problem of terrain map reconstruction is clearly the most desirable, given power and mass budgets which limit the usefulness of laser range finding, and uncertain / minimal atmospheric constituents, which limit the usefulness of acoustic / sonar based approaches. Other electromagnetic based approaches (e.g. radar) generally also have relatively high power requirements with respect to passive CCD camera systems. Passive CCD camera systems also have the added advantage of being an already space proven technology.

In terms of comparisons with other possible reconstruction methodologies, we find that our methodology strikes a good compromise between purely feature based (non iterative refinement) approaches, which tend to produce comparatively sparse depth maps (50 to 100 points in a 256x243 image), and area based (attempt to correlate all points) approaches, which produce comparatively dense depth maps (several thousands of points or more in a 256x243 image).

Depending on the iteration level selected for our algorithm, it may become more efficient to dewarp and rectify the images (at Levels 2 and 1, for example, where greater than 1000 to 1500 points may be computed). For Level 4, it is questionable whether dewarping and rectification buys anything in terms of speed, and for Level 8, the additional overhead makes it less efficient.

### 5.1.5 Algorithm Strengths

Again, the strength of this particular algorithm lies in its iterative terrain map refinement approach, where correlation and triangulation are performed for points which are both texturally interesting (also helping to avoid ambiguous matches during correlation) and distance-relevant from a danger perspective.

In the development of this algorithm, we make two implicit assumptions. The first is that the texture of the ground is recognizably different from the texture of the objects in the environment.

The second implicit assumption made concerns to the structure of the environment: we have tacitly assumed that the environment is divided into areas which we consider interesting (and worth computing distance values for), and areas which are not interesting. If the environment turns out to be generally interesting everywhere, then we do not gain anything in attempting to find interesting points, and really this just introduces additional overhead into the algorithm. No range information will be returned in the case of a flat plain, but this is actually a feature. It is questionable whether a sloping sand dune, which might be too difficult for the rover to climb, would be texturally interesting, being generally of the same texture as the ground. However, irrespective of whether the dune is texturally interesting, a standard correlation window, used to find matching points between images (in this algorithm and others) would most likely have a very difficult time finding correct correspondences on a sand dune-like structure.

Regarding the first implicit assumption, in general, we find that even if the ground is relatively texturally interesting, its signature is different enough from the objects in the environment that they are picked up as interesting.

Regarding the second implicit assumption, in general, we find that for the type of terrain which will be traversed by a planetary rover, the images will consist of rock fields, in which some rocks may rise up in the visual field, and others lie close to the ground. Thus, the environment appears to be filled in a semi-sparse manner, making the second implicit assumption a valid one.

### 5.1.6 Algorithm Improvements

Improvements to the terrain map generation algorithm could be realized with smaller field of view cameras, having less radial lens distortion, which would improve the efficiency of the correlation search, by reducing the error in using the fundamental matrix (epipolar line) approximation to the epipolar curves.

Additional improvements could be realized by performing the wavelet transform of the stereo images in real time using a DSP chip or other dedicated hardware. Similar improvements in speed could be realized by using a DSP chip or dedicated hardware for faster correlation. Of course, on a system with less restrictive hardware constraints, frame grabbing hardware capable of high speed memory to memory transfer would reduce the bottleneck on our system in the image acquisition phase.

## 5.2 Obstacle Avoidance

### 5.2.1 Implementation Caveats

Given that our obstacle avoidance methodology is quite simple, we will only discuss implementation caveats. As it is currently implemented, our obstacle avoidance integration volume does not consider points below half a rover wheel height to be threatening to the rover. As such, it can not deal effectively with pits, trenches, or cliffs. This functionality could be easily implemented, but was not relevant to our laboratory testing environment.

## 5.3 Terrain Map Registration

### 5.3.1 Algorithm Performance

Our implementation of Z. Zhang's terrain map registration algorithm was also a success, and helps to further validate the approach taken. We have tested this algorithm on a significantly larger dataset (again, on the order of several hundred stereo image pairs) than that presented in the papers [32], [34], etc., with good results.

Over all terrain maps generated, we found that the time to register the maps was on the order of only a few seconds.

Additionally, we have validated the use of this algorithm for data sets which contain significantly fewer 3D points (on the order of 400) with larger amounts of noise both in the initial motion estimate between the viewing frames, and in the estimates of the point positions themselves, compared with the rock scenes in [32] and [34].

Finally, we have shown that this vision-based motion estimate may be effectively used to generate or, in our case, assist in the generation of a rover state, which is alluded to in [34] but without the presentation of experimental results. We show positive results in challenging (rocky, with soft soil) terrain.

### 5.3.2 Performance Caveats

On our particular system, we found that the motion estimate was more reliable in rotation than in translation, due to the characteristics of our generated terrain maps. In general, points farther from the rover are less accurate in position due to the resolution of the CCD cameras, as well as the quality of their calibration.

### 5.3.3 Implementation Caveats

The larger amounts of noise in our system stems from our lack of high quality between-frame motion estimates, as the LSR rover is not equipped with wheel encoders, gyro/accelerometers, or any other means by which to gauge position and orientation changes (excepting straight time-based dead-reckoning).

Additional error is introduced by our wide field of view (120°) navigation cameras, which are generally difficult to calibrate to a high degree of accuracy.

### 5.3.4 Algorithm Strengths

The strengths of the this approach include the ability to deal with features which are present in one view and not in another (appearance and disappearance), be they from rotation in and out of the field of view, or from occlusion by another object. The iterative and somewhat statistical nature of the algorithm makes it robust to outliers and noise in the 3D dataset. [32]

### 5.3.5 Algorithm Weaknesses

Caveats to the use of this algorithm include the requirement that there be some features to match in the environment. Obviously the algorithm will fail on a featureless plane. At this time, additional methods of motion estimation might be explored, including ridge-line/ horizon tracking, if there are mountains in the distance.

### 5.3.6 Algorithm Improvements

Currently, the points in the 3D terrain maps are not given a quality estimate. Such an estimate could be incorporated into the terrain map registration algorithm, such that points with higher quality estimates are weighted more heavily in the registration process. The introduction of a Kalman filter into the registration process for just this purpose is actually suggested in [32].

Higher quality estimates of inter-frame motion would of course also improve the accuracy of the range-map registration algorithm, which relies on an initial motion estimate, such that after application of the initial motion estimate, inter-frame motion may be assumed to be small. Such estimates might be generated or measured using gyro/accelerometers, wheel encoders, and absolute heading sensors such as sun sensors. Additionally, matching range maps generated from a mast or arm-mounted stereo pair, rather than from a lower-mounted navigation camera pair, would also generate higher quality motion estimates, we suspect. Such a higher vantage point should allow a larger number of widely spaced points to be correlated between frames.

## 5.4 State Estimation

We were pleasantly surprised by the quality of the state estimate produced after Kalman filtering the dead-reckoned inter-frame motion estimate with the vision-based motion estimate.

Clearly, the use of Kalman filtering techniques can greatly improve the overall state estimate of a mobile vehicle such as a planetary rover (as we have demonstrated here, with 0.38 m vs. 1.34 m, 0.40 m vs. 1.38 m, and 0.59 m vs. 0.70 m mean absolute error in three different trials), especially if many measurement sources are available (e.g. gyro/accelerometer, wheel encoders, sun sensors, etc.).

### 5.4.1 Implementation Improvements

Currently, state estimation (via Kalman filtering) is only performed for in-plane rotations and translations. In difficult terrains, the assumption of planar motion is not necessarily correct as the rover moves over small obstacles in its efforts to achieve its goals or avoid larger obstacles. The extension of the Kalman filtering approach to out-of-plane translations and rotations could be used to improve the initial estimate used to start the terrain map registration algorithm, which would improve its subsequent output.

Additionally, currently we only use our dead reckoned guess and the output from the range map registration algorithm for the inter-frame rotation and translation of the rover. The addition of other sensor inputs (e.g. gyro/accelerometer, sun heading sensor, wheel encoders, etc.) into the Kalman filter framework would also serve to (perhaps dramatically) improve our estimate of the rover's state.

## 5.5 Path Planning and Navigation

### 5.5.1 Algorithm Performance

We have performed approximately 10 to 15 experimental trials of the path planning and navigation system, with various rock configurations in our laboratory. In general (approximately 80time), the rover was capable of achieving the desired goal position. In those cases where the rover was unable to achieve the goal, this was due to large errors introduced in the state estimate via interactions between the rover wheels and various normally benign rocks (given the open configuration of the rover wheels, it was possible for the edges of various rocks to actually extend inside the wheel, preventing the desired motion of the rover from being completed. This is in contrast to normal interactions with relatively benign rocks which did not drastically affect the Kalman filtered estimate of the rover's state.

### 5.5.2 Algorithm Strengths

Potential flow based path planning has been studied extensively in computer simulation [7]. To this we add the validation of it's use in real-time path planning for planetary rovers in challenging terrain.

As an approach, the strengths of the potential flow based path planning methodology lie in that it is capable of computing the desired motion of the rover in real time. This is in contrast to other approaches which rely on searches or other linear programming techniques which attempt to minimize the expected cost of the traverse to the desired goal. In using the potential flow based approach, a near optimal (in a distance sense) path is found between the rover and the goal.

### 5.5.3 Algorithm Weaknesses

Unfortunately, the path planning methodology used is currently only two-dimensional in nature. This precludes the use of the ground clearance of the rover to traverse obstacles which are higher than half a wheel height by straddling them and passing over them.

### 5.5.4 Algorithm Improvements

More precise control of the rover's position and orientation will prevent a number of the artifacts introduced by using finite stepsize motions (ie. jumping inside obstacle boundaries, or not staying exactly on a streamline)

Additionally, it should be noted that the real-time potential flow based path planning methodology employed is capable of dealing with dynamic environments, such that as the error in the state estimate of the rover grows over time, the obstacles in the environment of the rover may also grow with time, which guarantees that the rover will never intersect an object on it's path to the goal. We may explore this functionality in the future.

## 5.6 Sample Acquisition and Cache

### 5.6.1 Algorithm Performance

The correlated edge detection / texture segmentation-based sample segmentation algorithm we have developed worked very well in our lab environment (93.6% correct segmentation, over 171 stereo image pairs).

### 5.6.2 Testing Caveats

Again, as our rover was tethered to the VMEbus in our laboratory, we were unable to test the implemented algorithms on naturally occurring terrain. However, given the high quality of performance seen in the laboratory setting, we believe that the segmentation routine would perform similarly in a natural environment.

### 5.6.3 Algorithm Improvements

The addition of color CCD cameras into our suite of available sensors would possibly aid in the efficient segmentation of samples from the background environment, particularly in cases where the environment might be of similar textural quality to the samples of interest. The introduction of color would also add another identifying signature against which our decision to acquire or not acquire a sample given limited time and cache resources could be made.

## 5.7 Sample Manipulation and Soil Trenching

The sample manipulation algorithms all perform as expected, given the known limitations imposed by the nature of the hardware and the accuracy of the stereo camera imaging system, as detailed in section 5.7.

# Chapter 6

# Conclusions

In this chapter, we present our conclusions regarding our implementation of a suite of enabling algorithms for autonomous activity in planetary rover operations. Additionally, we present a look at where future work based on these results might proceed.

## 6.1 Summary

In summary, we have implemented a suite of enabling algorithms for increased autonomy in planetary rovers, and a modular framework which allows each of the component algorithms to be used independently. Although each algorithm may be used independently, the interaction between the various algorithms allows the implementation of fairly complex high level autonomous behaviors, including roaming obstacle avoidance, sample acquisition, and navigation to user designated goals.

Specifically, we have demonstrated a new method by which semi- sparse range maps may be generated in a computationally efficient fashion based on an iterative algorithm which is adaptive to both texture and distance. Depending on the number of 3D points desired by the user, the algorithm timing ranges from sub-second to multi-second. We then used this terrain map generation algorithm, together with our simple obstacle avoidance methodology to enable roaming obstacle avoidance.

Having demonstrated roaming obstacle avoidance, we then presented a sample segmentation algorithm, capable of approximately 94% correct segmentation in a semi-natural laboratory setting. This algorithm, coupled with roaming obstacle avoidance, gave rise to our first desired behavior, namely autonomous roaming obstacle avoidance and sample acquisition.

The implementation of an range map registration algorithm [32] allowed one measurement of the rover's state change during wandering or navigation to be performed. Combining this measurement with the commanded motion (dead-reckoned estimate) via Kalman filtering [1] enabled the LSR rover to maintain a state estimate accurate to approximately 0.4 m over a 6 m traverse in very soft soil and in rocky terrain. Additionally, our implementation of the range map registration algorithm [32] adds additional experimental performance validation over multiple navigation runs and validation of the use of the algorithm in conjunction with Kalman filtering as suggested in [32].

With this state estimate available, potential flow based path planning and navigation to user designated goals became possible. Here, we have presented navigation through an obstacle field to both a relatively unobstructed goal and a more obstructed goal using this real-time path planning approach [7]. Thus, in addition to the experimental validation of the algorithm, we have presented a demonstration of its feasibility in difficult terrain as might be encountered by a planetary rover.

## 6.2 Future Work

Currently, the algorithm suite has been implemented and runs on the LSR rover platform. As many of the algorithms (terrain map generation, obstacle avoidance, range map registration and Kalman filtering, real time potential flow based path planning and navigation to user designated goals) are applicable to the problems faced by the Sample Return Rover, the described algorithms are currently being implemented on the SRR platform, pictured in figure 6.1.

Specifically, the Sample Return Rover will have to generate terrain maps and avoid obstacles in its attempt to quickly rendezvous with one of the "dead" science rovers, such that it may retrieve the sample cache held by the science rover and return to its ascent vehicle for return to earth. In addition to avoiding obstacles, the SRR rover will have to maintain a state estimate to effectively navigate to the science rover goal, which may possibly be quite obstructed, requiring significant path-replanning, as demonstrated in figure 4.12.

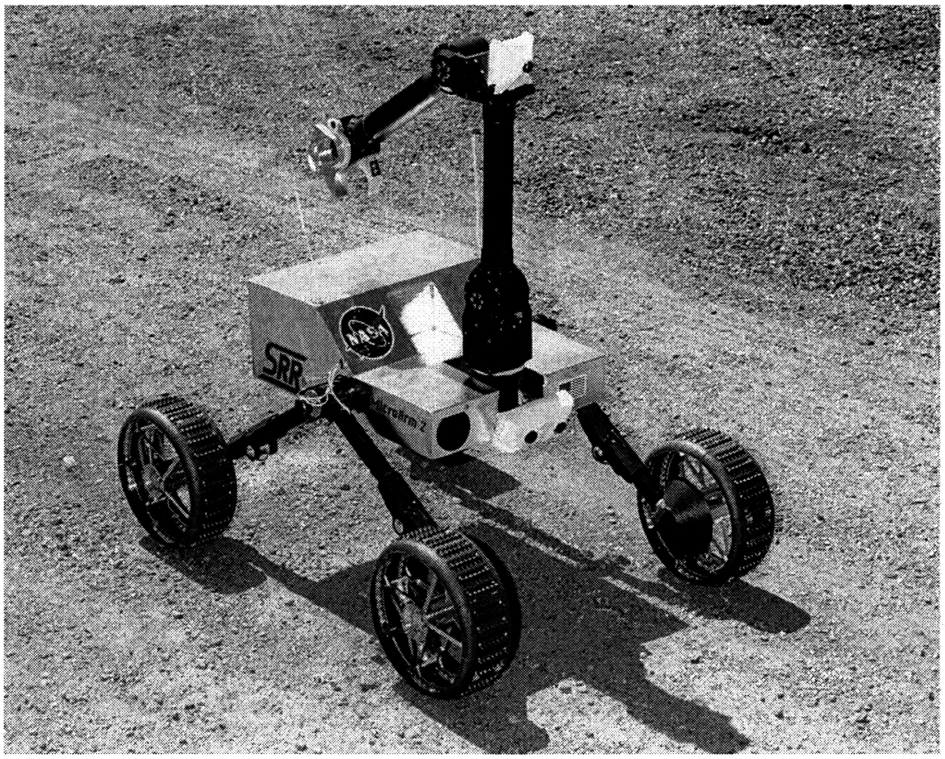


Figure 6.1: Sample Return Rover

# Appendix A

# MicroArm-1 Kinematics

## A.1 Forward Kinematics

The forward kinematics for MicroArm-1 are defined by the following transformation matrices between each of the joint coordinate frames.

In the following, for MicroArm-1, the values of the constants A1, A2, A3, A4, D2, D3, D4, and L5 are listed in table A.1.

| Variable | A1 | A2 | A3 | A4 |
|---|---|---|---|---|
| Value (m) | 0.0752 | 0.2847 | 0.2779 | 0.0363 |

| Variable | D2 | D3 | D4 | L5 |
|---|---|---|---|---|
| Value (m) | 0.0508 | 0.0508 | 0.0462 | 0.0762 |

Table A.1: Kinematic Constants

$\mathbf{T_{01}}$ :

$$\begin{bmatrix} cos(\theta_1) & -sin(\theta_1) & 0.0 & 0.0 \\ sin(\theta_1) & cos(\theta_1) & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$\mathbf{T_{12}}$ :

$$\begin{bmatrix} cos(\theta_2) & -sin(\theta_2) & 0.0 & A1 \\ 0.0 & 0.0 & 1.0 & D2 \\ -sin(\theta_2) & -cos(\theta_2) & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$\mathbf{T_{23}}$ :

$$\begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 0.0 & A2 \\ sin(\theta_3) & cos(\theta_3) & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & D3 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$\mathbf{T_{34}}$ :

$$\begin{bmatrix} cos(\theta_4) & -sin(\theta_4) & 0.0 & A3 \\ sin(\theta_4) & cos(\theta_4) & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & D4 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

$\mathbf{T_{45}}$ :

$$\begin{bmatrix} 1.0 & 0.0 & 0.0 & A4 + L5cos(\theta_5) \\ 0.0 & 1.0 & 0.0 & -L5sin(\theta_5) \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

Given the joint angles $(\theta_1, \theta_2, \theta_3, \theta_4, \theta 5)$, the end effector position is then given by:

$$X = \mathbf{T05}[0][3] = (\mathbf{T01} \times \mathbf{T12} \times \mathbf{T23} \times \mathbf{T34} \times \mathbf{T45})[0][3] \tag{A.1}$$

$$Y = \mathbf{T05}[1][3] = (\mathbf{T01} \times \mathbf{T12} \times \mathbf{T23} \times \mathbf{T34} \times \mathbf{T45})[1][3] \tag{A.2}$$

$$Z = \mathbf{T05}[2][3] = (\mathbf{T01} \times \mathbf{T12} \times \mathbf{T23} \times \mathbf{T34} \times \mathbf{T45})[2][3] \tag{A.3}$$

## A.2 Inverse Kinematics

The inverse kinematics for MicroArm-I are computed as follows (as derived by Dr. Eric Baumgartner, JPL).

### A.2.1 Theta 1

If we let $P_x$, $P_y$, and $P_z$ correspond to the position of the end effector, the desired end effector pointing vector be given by $\mathbf{n} = [nx, ny, nz]$, and the opening angle of the end effector ($\theta_5$) be given, then the two possible solutions for $theta_1$ are given by equation set A.4, where atan2 is the arctangent of $P_y/P_x$ in the range $-\pi \longrightarrow \pi$.

$$\begin{align*}
ex_1 &= \sqrt{P_x^2 + P_y^2 - (D2 + D3 + D4)^2} \\
\theta_{1_{r1}} &= atan2(P_y, P_x) - atan2((D2 + D3 + D4), ex_1) \tag{A.4} \\
\theta_{1_{r2}} &= atan2(P_y, P_x) - atan2((D2 + D3 + D4), -ex_1) \tag{A.5}
\end{align*}$$

The choice of $\theta_1$ from $\theta_{1_{r1}}$ and $\theta_{2_{r2}}$ is made by constraining the result to be in the range from $0 \longrightarrow \pi$.

### A.2.2 Theta 2

Once we have the base joint angle, $\theta_1$, the manipulator may be thought of as planar. We can then compute the end effector position in the plane of the manipulator arm using equation set A.6.

$$\begin{align*}
p_{tp_x} &= P_x cos(\theta_1) + P_y sin(\theta_1) - A1 \tag{A.6} \\
p_{tp_y} &= -P_x sin(\theta_1) + P_y cos(\theta_1) - (D2 + D3 + D4) \tag{A.7} \\
p_{tp_z} &= Pz \tag{A.8}
\end{align*}$$

Once we have the position of the end effector in the trenching plane, we can compute the position of the MicroArm-I wrist (we will assume that the end effector scoop is pointing down and introduce the additional complexity of multiple orientations for the end effector below). The wrist position of the manipulator arm is given by equation set A.9.

$$\begin{align*}
P_{wrist_x} &= P_x - (A4 + L5)\mathbf{n}_x \tag{A.9} \\
P_{wrist_y} &= P_y \tag{A.10} \\
P_{wrist_z} &= P_z - (A4 + L5)\mathbf{n}_z \tag{A.11}
\end{align*}$$

The solutions for theta 2 can then be computed using equation set A.12

$$\begin{align*}
cospsi &= \frac{P_{wrist_x}^2 + P_{wrist_z}^2 + A2^2 + A3^2}{(2)(A2)\sqrt{P_{wrist_x}^2 + P_{wrist_z}^2}} \\
sinpsi &= \sqrt{1 - cospsi^2} \\
psi_1 &= atan2(sinpsi, cospsi)
\end{align*}$$

$$\begin{aligned} psi_2 &= atan2(-sinpsi, cospsi) \\ \theta_{2_{r1}} &= -(atan2(P_{wrist_z}, P_{wrist_x}) + psi_1) \quad \text{(A.12)} \\ \theta_{2_{r2}} &= -(atan2(P_{wrist_z}, P_{wrist_x}) - psi_1) \quad \text{(A.13)} \\ \theta_{2_{r3}} &= -(atan2(P_{wrist_z}, P_{wrist_x}) + psi_2) \quad \text{(A.14)} \\ \theta_{2_{r4}} &= -(atan2(P_{wrist_z}, P_{wrist_x}) - psi_2) \quad \text{(A.15)} \end{aligned}$$

The solution for the value of $\theta_2$ is chosen to be that solution which lies between $-\pi/2$ and $\pi/2$

### A.2.3 Theta 3

The solution for $\theta_3$ is found in a similar fashion, using equation set A.16.

$$\begin{aligned} costh3 &= \frac{P^2_{wrist_x} + P^2_{wrist_z} - (A2^2 + A3^2)}{(2)(A2)(A3)} \\ sinth3 &= \sqrt{1 - costh3^2} \\ \theta_{3_{r1}} &= atan2(sinth3, costh3) \quad \text{(A.16)} \\ \theta_{3_{r2}} &= atan2(-sinth3, costh3) \quad \text{(A.17)} \end{aligned}$$

The solution for $\theta_3$ is selected as the solution which falls between 0 and $\pi$.

### A.2.4 Theta 4

The solution for $theta_4$ is then constrained by the solutions for $theta_1$, $theta_2$, $theta_3$, and the given $theta_5$, to be given by equation A.18.

$$\theta_4 = atan2(-\mathbf{n}_z, \mathbf{n}_x) - \theta_2 - \theta_3 \quad \text{(A.18)}$$

And thus we have solved for the joint angles of the manipulator given it's x,y,z position, it's end effector opening angle, and it's end effector pointing vector.

### A.2.5 Multifunction End Effector

Given that MicroArm-I has a multifunction end effector, with a micro-imaging camera, and abrading device, in addition to it's gripper- scoop, the end effector orientation is not necessarily always tip-down. (Generally, if we are performing soil or sample manipulation actions, the end effector pointing vector is specified as $[0, 0, -1]$, and the end effector is assumed to have the scoop pointing downward). However, if we wish to use the camera or abrading tool, we must modify the kinematics slightly, by using equation set A.19 for the wrist position.

$$\begin{aligned}
&\textbf{Abrader}: && \\
\phi &= 0.0 && \text{(A.19)}\\
\gamma &= \pi && \text{(A.20)}\\
wristlength &= A4 + LDR && \text{(A.21)}\\
&\textbf{Camera}: && \\
\phi &= atan(YCAM/(A4 + XCAM)) && \text{(A.22)}\\
\gamma &= (\pi/2) - phi && \text{(A.23)}\\
wristlength &= \sqrt{(YCAM^2)} && \text{(A.24)}\\
&\textbf{Gripper}: && \\
\phi &= 0.0 && \text{(A.25)}\\
\gamma &= 0.0 && \text{(A.26)}\\
wristlength &= A4 + L5 && \text{(A.27)}\\
&\textbf{PointingVector}: && \\
\mathbf{n}_{new_x} &= cos(\gamma)\mathbf{n}_x + sin(\gamma)\mathbf{n}_z && \text{(A.28)}\\
\mathbf{n}_{new_y} &= 0 && \text{(A.29)}\\
\mathbf{n}_{new_z} &= -sin(\gamma)\mathbf{n}_x + cos(\gamma)\mathbf{n}_z && \text{(A.30)}\\
&\textbf{WristPosition}: && \\
\mathbf{P_{wrist}} &= \mathbf{P_{ee}} - \mathbf{n_{new}}(wristlength) && \text{(A.31)}
\end{aligned}$$

Similarly, we then use equation A.32 for $\theta_4$.

$$\theta_4 = atan2(-\mathbf{n}_z, \mathbf{n}_x) - \theta_2 - \theta_3 + phi \qquad \text{(A.32)}$$

# Bibliography

[1] Baumgartner, E. T., and Skarr, S. B., "An Autonomous Vision-Based Mobile Robot," *IEEE Transactions on Automatic Control*, Vol. 39, No. 3, March 1994.

[2] Bernard, S. T., Fischler, M. A., "Computational Stereo," Comput. Surv. 14(4):553-572, 1982.

[3] Betge'-Brezetz, S., Chatila, R., Devy, M., "Object-based Modelling and Localization in Natural Environments," *IEEE International Conference on Robotics and Automation*, pp. 2920-2927, 1995.

[4] Connolly, C. I., Burns, J. B., Weiss, R. "Path Planning Using Laplace's Equation," *Proc. IEEE International Conference on Robotics and Automation*, pp. 2101-2106, 1990.

[5] Espinal, F., Huntsberger, T., Jawerth, B., Kubota, T., "Wavelet-Based Fractal Signature Analysis for Automatic Target Recognition," *Optical Engineering*, Vol. 37, No. 1, pp. 166-174, January 1998.

[6] Faugeras, Oliver, "Three-Dimensional Computer Vision - A Geometric Viewpoint", The MIT Press, Cambridge, MA, 1993.

[7] Feder, H. J. S., and Slotine, J-J. E., "Real-Time Path Planning Using Harmonic Potentials in Dynamic Environments," *IEEE International Conference on Robotics and Automation (ICRA)*, April 1997.

[8] Fua, P., "A parallel stereo algorithm that produces dense depth maps and preserves image features," *Machine Vision and Applications*, 6(1), Winter 1993.

[9] Gauch, John, blob_color.c, via. Internet, University of Kansas, 1994-1997.

[10] Gennery, D. B., "Visual terrain matching for a Mars rover.", *Proc. International Conference on Computer Vision and Pattern Recognition*, pp. 483-491, San Diego, CA, June 1989.

[11] Gennery, D. B., "Least-Squares Camera Calibration Including Lens Distortion and Automatic Editing of Calibration Points," *Calibration and Orientation of Cameras in Computer Vision*, A. Grün and T. Huang, editors, Springer-Verlag, 1993.

[12] Hebert, M., Caillas, C., Krotkov, E., Kweon, I.S., Kanade, T. "Terrain Mapping for a Roving Planetary Explorer", *Proc. IEEE International Conference on Robotics and Automation*, pp. 997-1002, 1989.

[13] Hoffman, B., Baumgartner, E., Huntsberger, T., Schenker, P., "Improved Rover State Estimation in Challenging Terrain," submitted to Autonomous Robots, Special Issue.

[14] Hoffman, B., "Scene Analysis Using Active Vision", S.B. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, 1997.

[15] Kim, J-O., Khosla, P., "Real-Time Obstacle Avoidance Using Harmonic Potential Functions," *Proc. IEEE International Conference on Robotics and Automation*, 1991.

[16] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robotics," *The International Journal of Robotics Research*, **5**, No. 1, pp. 90-98, 1986.

[17] Lacroix, S., Fillatreau, P., Nashashibi, F., "Perception for Autonomous Navigation in a Natural Environment," *Workshop on Computer Vision for Space Applications*, Antibes, France, Sept. 22-24, 1993.

[18] Lorigo, Liana M., "Visually-Guided Obstacle Avoidance in Unstructured Environments," M.S. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.

[19] Matthies, L., Gat, E., Harrison, R., Wilcox, B., Volpe, R., Litwin, T., "Mars Microrover Navigation: Performance Evaluation and Enhancement," *Autonomous Robots Journal*, special issue on Autonomous Vehicles for Planetary Exploration, Volute 2(4), pp. 291-312, 1995.

[20] Matthies, L., "Dynamic Stereo Vision," Ph.D. Thesis, Computer Science Department, Carnegie Mellon, 1987.

[21] Matthies, L., Olson, C., Tharp, G., Laubach, S., "Visual Localization Methods for Mars Rovers Using Lander, Rover, and Descent Imageery," *International Symposium on Artificial Intelligence, Robotics, and Automation in Space* (i-SAIRAS), Tokyo, Japan, July 1997.

[22] Olson, C., "Mobile Robot Self-Localization by Iconic Matching of Range Maps," *Proc. of the 8th International Conference on Advanced Robotics*, pp. 447-452, 1997.

[23] Quinlan, S., Khatib, O., "Elastic Bands: Connecting Path Planning and Control," *Proc. IEEE International Conference on Robotics and Automation*, **2**, pp. 802-808, 1993.

[24] Robert, Luc, Zeller, Cyril, Faugeras, Oliver, Hebert, Martial, "Applications of non-metric vision to some visually guided robotics tasks," INRIA, Research Report Number 2584, June 1995

[25] Roumeliotis, S., Bekey, G., "An Extended Kalman Filter for frequent local and infrequent global sensor data fusion," *Proc. SPIE Vol. 3209*, 1997.

[26] Schenker, P.S., Baumgartner, E.T., Lee, S., Aghazarian, H., Garrett, M.S., Lindemann, R.A., Brown, D.K., Bar-Cohen, Y., Lih, S., Joffe, B., Kim, S.S., Hoffman, B.D., Huntsberger, T., "Dexterous robotic sampling for Mars *in-situ* science", *Intelligent Robotics and Computer Vision XVI*, Proc. SPIE 3208, Pittsburgh, PA, Oct 14-17, 1997.

[27] Schenker, P.S., Sword, L.F., Ganino, A.J., Bickler, D.B., Hickey, G.S., Brown, K, Baumgartner, E.T., Matthies, L.H., Wilcox, B.H., Balch, T., Aghazarian, H., and Garrett, M.S., "Lightweight Rovers for Mars Science Exploration and Sample Return", Intelligent Robotics and Computer Vision XVI, SPIE Volume 3208:24-35, Pittsburgh, Pennsylvania, October 1997.

[28] Stoica, Adrian, et al., "Fuzzy-Logic-Based Decision Making for Mars Sample Selection", JPL Survey and Example Application of Advanced Technology to Autonomy, JPL D-13900, Volume 2, Final Report, 14 October 1996.

[29] Stollnitz, E. J., DeRose, T. D., Salesin, D. H., 'Wavelets for Computer Graphics, Theory and Applications," Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1996.

[30] Walker, M. W., Shao, L. and Volz, R. A., "Estimating 3-D location parameters using dual number quaternions," CVGIP: *Image Understanding* 54(3), 358-367.

[31] Xiong, Y., Matthies, L., "Error Analysis of a Real-Time Stereo System," 1997.

[32] Zhang, Z., "Iterative Point Matching for Registration of Free-Form Curves and Surfaces," *International Journal of Computer Vision*, 13:2, 119-152, 1994.

[33] Zhang, Z., Deriche, R., Faugeras, O., Luong, Q-T., "A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry," *Institute National De Recherche En Informatique et en Automatique (INRIA)*, Rapport de recherche No. 2273, May 1994.

[34] Zhang, Z., "A Stereovision System for a Planetary Rover: Calibration, Correlation, Registration, and Fusion," *Proc. IEEE Workshop on Planetary Rover Technology and Systems*, Minneapolis, Minnesota, April 23, 1996.

[35] Zhang, Z., "On the Epipolar Geometry Between Two Images With Lens Distortion," *Proc. Int'l Conf. Pattern Recognition (ICPR)*, Vol. I, pp. 407-411, Vienna, Aug. 1996.

[36] Zhang, Z., Image-Matching software.
URL:*http://www.inria.fr/robotvis/personnel/ zzhang/softwares.html*

[37] http://www.jpl.nasa.gov