Terrain Identification Methods for Planetary Exploration Rovers

by

Christopher Allen Brooks

B.S., Engineering and Applied Science
California Institute of Technology, 2000

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

September 2004

Signature of Author .....
Department of Mechanical Engineering
August 6, 2004

Certified by .....
Steven Dubowsky
Professor of Mechanical Engineering
Thesis Supervisor

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

Terrain Identification Methods for Planetary Exploration Rovers

by

Christopher Allen Brooks

Submitted to the Department of Mechanical Engineering
on August 6, 2004 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

ABSTRACT

Autonomous mobility in rough terrain is becoming increasingly important for planetary exploration rovers. Increased knowledge of local terrain properties is critical to ensure a rover's safety, especially when driving on slopes or rough surfaces. This thesis presents two methods for using on-board sensors to identify local terrain conditions.

The first method visually measures sinkage of a rover wheel into deformable terrain, based on a single color or grayscale image from a camera with a view of the wheel-terrain interface. Grayscale intensity is computed along the rim of the wheel, and the wheel-terrain interface is identified as the location with maximum change in intensity. The algorithm has been shown experimentally to give accurate results in identifying the terrain characteristics under a wide range of conditions.

The second method classifies terrain based on vibrations induced in the rover structure by rover-terrain interaction during driving. Vibrations are measured using an accelerometer on the rover structure. The method uses a supervised learning approach to train a classifier to recognize terrain based on representative vibration signals during an off-line learning phase. Real-time terrain classification uses linear discriminant analysis in the frequency domain to identify gross terrain classes such as sand, gravel, or clay. The algorithm is experimentally validated on a laboratory testbed and on a rover in outdoor conditions. Results demonstrate the robustness of the algorithm on both systems.

Thesis Supervisor: Steven Dubowsky

Title: Professor of Mechanical Engineering

Acknowledgements

I would like to thank Karl Iagnemma and Professor Steven Dubowsky for their guidance and support for my research and this thesis. Their advice has greatly improved the quality of this work.

I would like to thank Matt Carvey for his work with the initial design and construction of the TORTOISE rover and Shingo Shimoda for his help in the final assembly and experiments with the rover. Without their help I couldn't have gotten the results presented in this work.

I would also like to thank the other members of the Field and Space Robotics Laboratory for making the hours in the lab enjoyable. And, of course, I would like to thank my family and friends for making my time outside the lab something to look forward to.

This work was supported by the Mars Technology Program at the NASA Jet Propulsion Laboratory.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement and Motivation

Future planetary exploration missions will require rovers to act with increasing degrees of autonomy. This is necessary for rovers to make the best use of their limited lifetimes, since communication rates with Earth are low, and communication time lags make remote operation infeasible for complex tasks.

Until recently, planetary rover control operations were largely open-loop. That is, control operators were limited to commands such as "Drive the wheels forward 10 turns," which a rover would execute without feedback as to whether driving the wheels made the rover move forward or just dig deeper into the soil. Images of the surroundings were then sent back to Earth, and the rover waited idly until new instructions were received from ground controllers.

On February 8th, 2004, autonomous operation was enabled aboard the Mars Exploration Rover Spirit, allowing it to autonomously plan a path to a distant target destination (NASA/JPL, February 9, 2004). With this capability, rover "drivers" can tell the rover, "Go to that spot." The rover then analyzes range data from its stereo cameras, identifies possible obstacles, and plans a path to the target that avoids the obstacles. More

importantly, it can stop, look around, and re-plan its path along the way, without needing support from Earth.

This new capability relies on machine vision algorithms to identify geometric obstacles, for example rocks or steep slopes, in order to avoid them. However geometric obstacles are not the only threats to a rover's mobility. The terrain itself may be a hazard, as a rover could become entrenched in loose sandy soil even with no rocks or slopes present. When slopes are present, terrain conditions potentially play an even more important role. Recent experiences of the Mars Exploration Rover mission have illustrated this: significant wheel slip during an Opportunity rover traverse up a slope in February delayed the study of a rocky outcrop for a full day (NASA/JPL, February 6, 2004). Plans to navigate Opportunity into Endurance Crater call for it to keep its wheels on solid rock, rather than the fine drift material, to avoid becoming trapped (NASA/JPL, June 8, 2004). Thus, to ensure rover safety and increase autonomy, there is a need for algorithms for autonomous detection and identification of terrain conditions.

1.2 Purpose of this Thesis

The purpose of this thesis is to present two algorithms to provide planetary rovers with important information about local terrain, specifically the depth of wheel sinkage and the type of terrain the rover is traversing. This information is important to allow rovers to safely and autonomously traverse potentially treacherous terrain.

The first algorithm measures rigid wheel sinkage into deformable terrain using a single image containing a view of the wheel-terrain interface. The algorithm will be presented, followed by experimental results from the Field and Space Robotics

Laboratory (FSRL) Wheel-Terrain Interaction Testbed validating the algorithm's measurement accuracy. Wheel sinkage can be used as a standalone estimator of mobility in rough terrain. It can also be used as an input to estimate soil cohesion and internal friction angle, or terrain traversability (Iagnemma, 2001; Kang, 2003).

The second algorithm identifies the gross terrain class (e.g. "sand," "gravel," "clay") based on vibrations induced by rover-terrain interaction during a traverse. The algorithm will be presented, followed by experimental results from the FSRL Wheel-Terrain Interaction Testbed and the FSRL Technology Testbed Rover. These results will demonstrate the algorithm's capacity to robustly and accurately identify real-world terrain. This vibration-based terrain classification algorithm is intended to be used to allow a rover to modify its behavior based on the type of terrain it is traversing. It can also be used to identify submerged terrain features of scientific interest.

1.3 Background and Related Research

Research on rover-terrain interaction has been ongoing at the Field and Space Robotics Laboratory for several years. This work has utilized Bekker's soil models (Bekker, 1956) to allow for on-line identification of soil cohesion and internal friction angle during a traverse (Iagnemma, 2001). Other work in the laboratory has produced a reduced order model to estimate the traversability of deformable terrain (Kang, 2003). Both of these models rely on knowledge of wheel sinkage. The desire for a practical method for estimating wheel sinkage led to the development of the visual wheel-sinkage measurement algorithm.

Previous research into the characteristics of Mars soil has been accomplished using various methods to estimate wheel sinkage. Soil experiments conducted using Sojourner, the Pathfinder rover, required that the rover dig a trench with one wheel while the others remained stationary (Moore *et al*, 1999). Suspension configuration sensors were used to estimate the depth of the wheel relative to its initial position, but no absolute measure of sinkage relative to the terrain surface was available.

In (Wilcox, 1994), a method was presented to estimate wheel sinkage and slip on-line during a traverse. This approach used configuration sensors as well as a look-ahead range sensor as inputs, and assumed that only the front wheels (of a six-wheeled rover) compacted the soil. Due to this assumption it could not accommodate non-straight driving paths in deformable terrain. In summary, to the author's knowledge there exists no documented method for on-line measurement of wheel sinkage for an arbitrary path.

Much research has been performed in visual classification of terrain, an area related to both of the algorithms presented in this thesis. Fundamental studies of visual texture discrimination are presented in (Castaño, Manduchi, & Fox, 2001), (Espinal, Huntsberger, Jawerth, & Kubota, 1998), (Manduchi, 2000), and (Malik & Perona, 1990). Most of the work has used Gabor filters to assign texture properties to a location, with various models for terrain class texture distributions. A combined color, texture, and range-based terrain classification algorithm is presented in (Bellutta, Manduchi, Matthies, Owens, & Rankin, 2000), and another is presented in (Rasmussen, 2002). This research is focused on higher-speed terrestrial vehicles, however, with the goal of keeping vehicles on a semi-structured road surface. An overview of vision for robot navigation may be found in (DeSouza & Kak, 2002).

The development of vibration-based terrain classification was motivated by the desire for an algorithm that could quickly identify a gross terrain class during a traverse. It was hypothesized that an algorithm relying on the vibration induced in the rover structure would be able to sense changes in terrain below a thin surface layer, for example the thin dust coating found on Mars.

Research related to vibration recognition is limited in the mobile robotics field. One robot with limited terrain identification capabilities is described in (Voyles, Larson, Yesin, & Nelson, 2001). This is a two-legged robot which moves in a swimming motion, and the signals used to identify the terrain are visual servoing errors. The presence of a distinct gait led to the use of a Hidden Markov Model, which is inappropriate for wheeled rovers.

Researchers at Carnegie Mellon have presented work on vibration-based classification in (Wu, Siegel, & Khosla, 1999). Their work focused on identifying passenger vehicles using a stationary microphone beside the road. The approach they used to analyze the vibration signature is similar to the terrain classification algorithm presented in Chapter 3.

Recently, independent research has been done at Florida State to classify terrain based on the vertical acceleration data from an IMU on a mobile robot (Sadhukhan & Moore, 2003; Sadhukhan, 2004). Their work is focused on classifying terrain for a high-speed autonomous vehicle with pneumatic tires, for use in battlefield conditions. Their vehicle speed is a factor of 10 higher than what might be expected for a planetary rover (80 cm/s rather than 5-10 cm/s). Since vibration signal amplitudes for a given terrain increase as the square of the vehicle speed, their work is based on classifying vibrations

orders of magnitude larger than can be expected on a planetary rover. Frequency-domain signal analysis is used in their work, similar to the approach presented in Chapter 3. Two different algorithms are presented to perform the classification, one based on a Probabilistic Neural Network (PNN) (Specht, 1988), and another based on a neural network trained using error back-propagation (Rumelhart, Hinton, & Williams, 1986). Their work concludes that the PNN method is superior. A comparison of their PNN method to the linear discriminant analysis approach proposed in this thesis is presented in Section 3.5.

Significant research in vibration-based classification has also been done for speech recognition. Work in this field, for example (Lu, Jiang, & Zhang, 2001) and that described in (Gerhard, 2000), is based on studying the time variation of features such as the number of zero-crossings and the signal energy. Such emphasis on the time variation is not possible for most signals other than speech.

1.4 Outline of this Thesis

This thesis has four chapters, plus appendices. This chapter, the introduction, summarizes the motivation and purpose of the research and provides information on research related to the work.

Chapter 2 describes an algorithm for visually measuring wheel sinkage into deformable terrain. It explains the details of the algorithm and presents experimental results to illustrate its effectiveness.

Chapter 3 describes a vibration-based terrain classification algorithm. It explains the details of the classification algorithm and presents experimental results showing the

effectiveness of the algorithm on two test platforms. It also compares this algorithm to one developed independently by researchers at Florida State University.

Chapter 4 is the conclusion. It describes the contributions of this work and presents potential avenues for continued research.

Background material, equipment descriptions, and detailed experimental results are contained in the appendices. Appendix A describes the matrix formulation used in the visual wheel sinkage measurement algorithm. It also describes the pinhole camera model and how the parameters in the matrix-based model can be related to the CAHV model (Yakimovsky & Cunningham, 1978).

Appendix B provides a foundation for understanding principal component analysis, which is used in the vibration-based terrain classification algorithm. It also presents a brief explanation of singular value decomposition as a useful method for analyzing data.

Appendix C provides detailed information about the FSRL Wheel-Terrain Interaction Testbed. This platform was used for validating both the visual wheel sinkage measurement algorithm and the vibration-based terrain classification algorithm.

Appendix D describes the FSRL Technology Testbed Rover. This rover was used for validating the vibration-based terrain classification algorithm.

Appendix E presents visual sinkage measurement results for a wide variety of data sets.

Chapter 2

Visual Measurement of Wheel Sinkage

2.1 Introduction

The first sensory method described in this thesis is visual measurement of mobile robot wheel sinkage. This is intended to measure the depth a rigid wheel has sunk into deformable terrain, based on an image containing a view of the wheel. This method can also detect the case of a rigid wheel supported by rigid terrain or a rigid object such as a rock.

Wheel sinkage is valuable information describing a robot's wheel-terrain interaction state. For example, a rover traversing loose sand might experience substantial wheel sinkage, leading to poor mobility due to increased motion resistance (see Figure 2.1). Conversely, a rover traversing firm clay might experience little wheel sinkage and maintain high mobility. Previous research has shown that wheel sinkage is a key variable in predicting terrain traversability (Iagnemma, Kang, Brooks, & Dubowsky, 2003; Kang, 2003). With knowledge of wheel sinkage, a mobile robot could modulate its wheel torque to improve traction or revise its motion plan to avoid potentially hazardous terrain.

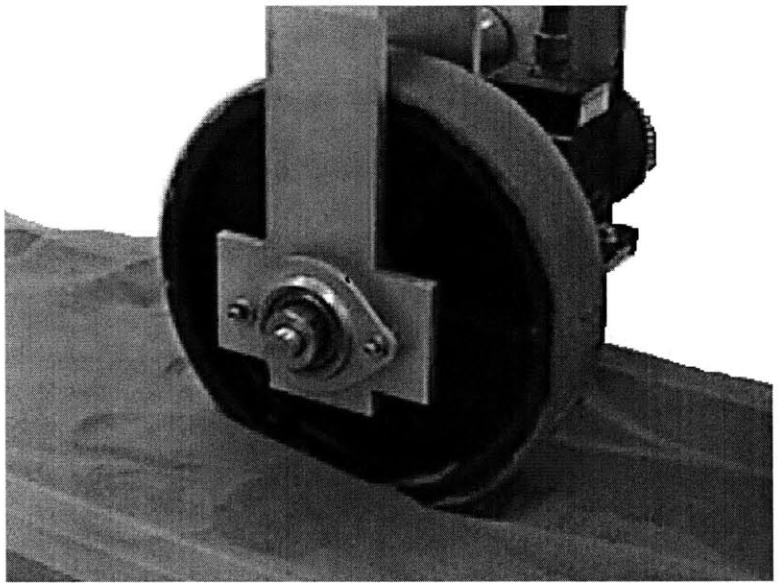


Figure 2.1. Example of rigid wheel sinkage in dry sand

Wheel sinkage is important to terrain identification and classification algorithms (Iagnemma *et al*, 2003; Iagnemma, Shibly, & Dubowsky, 2002). These algorithms are particularly useful in scientific studies of soil properties during planetary exploration missions (Volpe, 2003).

The algorithm presented autonomously measures the wheel sinkage from a single color or grayscale image containing a view of the wheel. This approach is intended for use with underbelly-mounted cameras, such as those currently used for hazard detection aboard the two MER rovers, Spirit and Opportunity. A sample image from an underbelly hazard detection camera on Spirit is shown in Figure 2.2. Such an image would be an appropriate input for this algorithm.

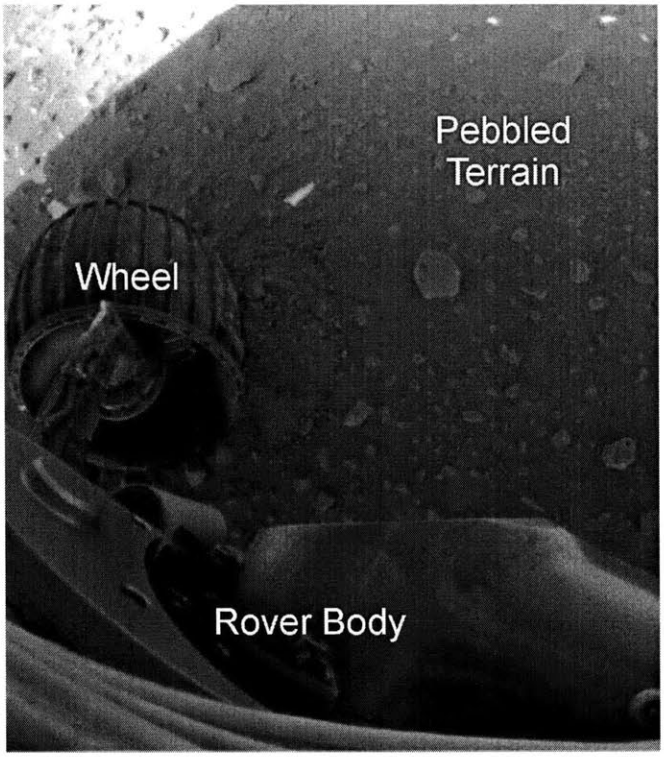


Figure 2.2. Sample MER Image (NASA/JPL, 2004)

The algorithm presented here is intended for vehicles with rigid wheels, such as those found on recent generations of Mars rovers. It can also be applied to pneumatic tires, if the tire inflation pressure is high compared to the terrain stiffness (Bekker, 1956).

Section 2.2 presents an overview of the algorithm, followed by a detailed description in Section 2.3. Experimental results are presented in Section 2.4, showing the performance of the algorithm using images captured using the FSRL Wheel-Terrain Interaction Testbed. These tests show that the algorithm is accurate and robust to variation in terrain and lighting conditions. Several potential modifications to the algorithm to improve performance are presented in Sections 2.4.3 and 2.4.5.

2.2 Algorithm Overview

The goal of the algorithm is to measure wheel sinkage in deformable terrain by analyzing a single color or grayscale image containing the wheel-terrain interface. It is assumed that a camera is mounted on the rover body, with a field of view containing the wheel-terrain interface. Sinkage is defined as a pair of angles from the vertical ($\mathbf{v}_{\mathbf{down}}$) termed the left and right terrain interface angles, θ_L and θ_R (shown in Figure 2.3). This represents a general description of wheel sinkage in uneven terrain. To determine these angles, only an annular region along the wheel rim (between r_{rim} and r_{wheel}) on the lower half-wheel needs to be examined. This reduces computational requirements by eliminating much of the scene.

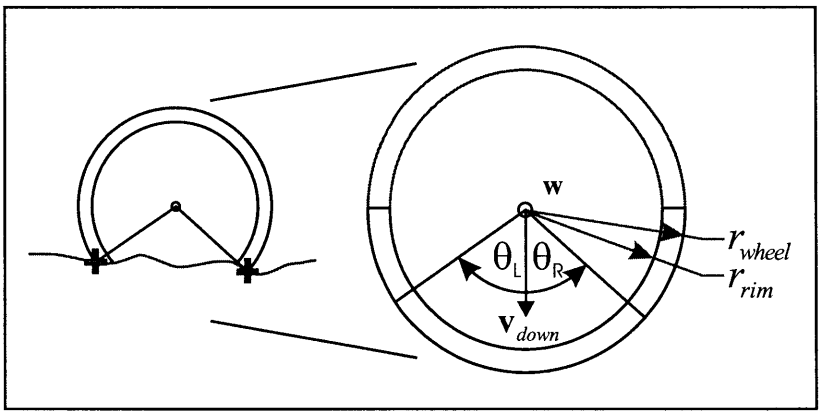


Figure 2.3. Rigid wheel sinking into deformable terrain with left (θ_L) and right (θ_R) terrain interface angles shown

It is assumed that the location of the wheel center, $\mathbf{w}$, relative to the camera is known. This is a reasonable assumption since many mobile robots have rigid suspensions. Robots with articulated suspensions (such as the MSL) are generally instrumented with suspension configuration sensors. Note that visual methods for

identifying the wheel center location could be implemented, however this would increase computational complexity. The proposed approach can be applied to steerable wheels if the steering angle is known.

It is also assumed that the wheel rim is visually distinguishable from the surrounding terrain. This is usually true for rigid, metallic wheels or dark pneumatic tires in natural terrains. Visual contrast can be enhanced by coloring the wheel rim a non soil-like color such as blue. This pixel-level difference in appearance eliminates the need for computationally-intensive texture analysis or stereo-based correlation. The algorithm instead relies on a relatively simple analysis of color or grayscale intensity along the wheel rim.

The algorithm consists of the following three steps: 1) wheel rim identification, 2) pixel intensity computation, and 3) terrain interface identification. The following section describes these steps.

2.3 Sinkage Measurement Algorithm

The algorithm employs a pinhole camera model, with the following coordinate frames (see Figure 2.4):

- **wheel frame:** a non-rotating frame fixed at the wheel hub, with the x-y plane in the plane of the wheel and the z-axis parallel to the wheel axle and pointing towards the camera
- **camera frame:** a frame with its origin at the camera's focus point (the "pinhole" in the camera model) and with its x and y axes aligned to the image axes

- **translated wheel (TW) frame:** a frame with its origin coincident with the origin of the camera frame and with its axes aligned with those of the wheel frame

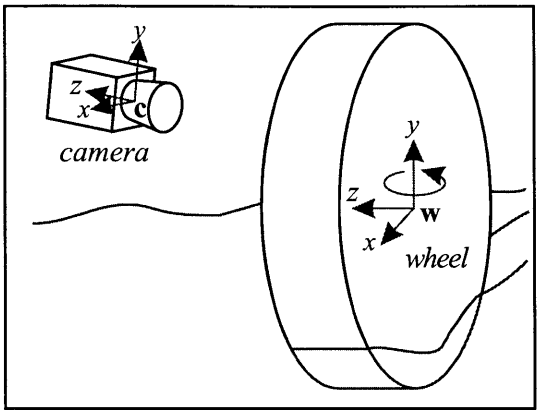


Figure 2.4. Illustration of camera and wheel frames

Matrices $\mathbf{T}_{\text{wheel}}^{\text{TW}}$, $\mathbf{T}_{\text{camera}}^{\text{TW}}$, and $\mathbf{T}_{\text{camera}}^{\text{wheel}}$ are defined as 4×4 transformation matrices that relate a position vector in one frame to a position vector in another. Here, a position vector is represented as a 4×1 column vector. For example, an arbitrary point p is represented by the position vector $\mathbf{p}_{\text{wheel}} = [p_{x,wheel} \quad p_{y,wheel} \quad p_{z,wheel} \quad 1]^{\text{T}}$ in the wheel frame. The same point is represented in the camera frame by $\mathbf{p}_{\text{camera}}$. The transformation matrix $\mathbf{T}_{\text{camera}}^{\text{wheel}}$ relates the two as

$$\mathbf{p}_{\text{camera}} = \mathbf{T}_{\text{camera}}^{\text{wheel}} \mathbf{p}_{\text{wheel}} . \qquad (2.1)$$

See Appendix A for a detailed explanation of the matrix notation and projection using a pinhole camera model.

2.3.1 Wheel Rim Identification and Classification

The first step in the algorithm is to identify all points of interest in the image. Points of interest are defined as points in the plane of the wheel rim which lie in lower half of the annular region between the inner wheel rim diameter, r_{rim}, and the outer wheel rim diameter, r_{wheel} (see Figure 2.3). For rimless wheels or tires, r_{wheel} corresponds to the outer tire diameter, and r_{rim} is chosen to be slightly less than r_{wheel}.

These circles in the plane of the wheel rim can be projected through the camera's focus point as cones. The 4×4 matrices $\mathbf{W}_{TW}$ (coinciding with the outer wheel rim) and $\mathbf{R}_{TW}$ (coinciding with the inner wheel rim) define these cones in the translated wheel frame. (See Appendix A for details on how these matrices are computed.) Using this notation $\mathbf{p}_{TW}{}^{T}\mathbf{W}_{TW}\mathbf{p}_{TW} < 0$ for any point p which will appear within the outer wheel rim in the image. Note that conversion of the cones to the camera frame (and analogously to the wheel frame) is accomplished as:

$$\mathbf{W}_{camera} = \left(\mathbf{T}_{TW}^{camera}\right)^{T}\mathbf{W}_{TW}\mathbf{T}_{TW}^{camera}. \qquad (2.2)$$

Points of interest in the annular area are divided into two regions, corresponding to the left and right halves of the wheel (see Figure 2.5). This is done because terrain entry generally occurs in one half of the wheel, and terrain exit occurs in the other. Thus the algorithm will search for one terrain interface in each region. Left and right annular regions are determined with respect to the vector $\mathbf{v}_{down}$. The vector $\mathbf{v}_{down}$ is a unit vector perpendicular to the pitch angle of the vehicle body (i.e. on flat terrain, $\mathbf{v}_{down}$ is parallel to the gravity vector).

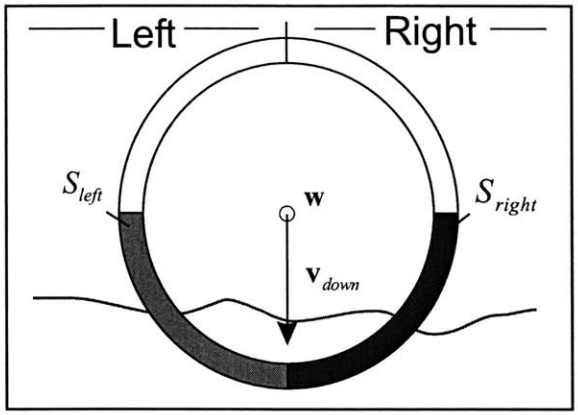


Figure 2.5. Annulus sections and $\mathbf{v}_{down}$

Each pixel in the image can be assigned as a member of one of three sets: 1) points belonging to the lower left quadrant of the wheel annulus, S_{left} ; 2) points belonging to the lower right quadrant of the wheel annulus, S_{right} ; and 3) points that are not in the region of interest on the wheel rim. Using $\mathbf{T}_{\mathbf{camera}}^{\mathbf{pixel}}$ defined in Appendix A, a pixel with image coordinates (x_{pixel}, y_{pixel}) is mapped to a point p on the camera back plane as:

$$\mathbf{p}_{\mathbf{camera}} = \mathbf{T}_{\mathbf{camera}}^{\mathbf{pixel}} \begin{bmatrix} x_{pixel} \\ y_{pixel} \\ 1 \end{bmatrix}. \qquad (2.3)$$

This point p on the camera back plane lies inside the annular region if it satisfies the following inequality:

$$\mathbf{p}_{\mathbf{camera}}{}^{\mathrm{T}}\mathbf{W}_{\mathbf{camera}}\mathbf{p}_{\mathbf{camera}} < 0 < \mathbf{p}_{\mathbf{camera}}{}^{\mathrm{T}}\mathbf{R}_{\mathbf{camera}}\mathbf{p}_{\mathbf{camera}}. \qquad (2.4)$$

If this inequality is satisfied, the point's location on the left or right side can be found by first identifying its corresponding point on the wheel rim, q. This is done by

projecting the point on the back plane through the focus point onto the plane of the wheel rim, using the translated wheel reference frame:

$$\mathbf{q}_{\text{wheel}} = \mathbf{T}_{\text{wheel}}^{\text{TW}} \, \text{proj}_{rim}\left(\mathbf{T}_{\text{TW}}^{\text{camera}} \mathbf{p}_{\text{camera}}\right), \tag{2.5}$$

using proj(·) as defined in Appendix A.

The point lies in the right half of the annulus if $(\mathbf{v}_{\text{down}} \times \mathbf{q}_{\text{wheel}}) > 0$, or in matrix form:

$$\left[\begin{array}{c:c:cc} & & 0 & 0 \\ \mathbf{v}_{\text{down}} & \mathbf{q}_{\text{wheel}} & 0 & 0 \\ & & 1 & 0 \\ & & 0 & 1 \end{array}\right] > 0. \tag{2.6}$$

If $(\mathbf{v}_{\text{down}} \times \mathbf{q}_{\text{wheel}}) \leq 0$, the point is in the left half.

2.3.2 Pixel Intensity Computation

The average grayscale intensity is computed for every row of pixels in S_{left} and S_{right} (see Figure 2.6). A row is a set of pixels aligned perpendicular to $\mathbf{v}_{\text{down}}$. We denote n rows as subsets $r_{left,k} \subset S_{left}$ and $r_{right,k} \subset S_{right}$, where $k \in \{1,...,n\}$. Note that n is a function of the spatial resolution Δy and the wheel diameter. A pixel has membership in row k if the following equation is satisfied:

$$\left\lfloor \frac{\mathbf{v}_{\text{down}}{}^{\text{T}} \mathbf{q}_{\text{wheel}}}{\Delta y} \right\rfloor = k, \tag{2.7}$$

where Δy is an adjustable parameter corresponding to the smallest change in sinkage the algorithm can detect. Note that decreasing Δy below the imaged pixel resolution of the rim will not increase measurement accuracy.

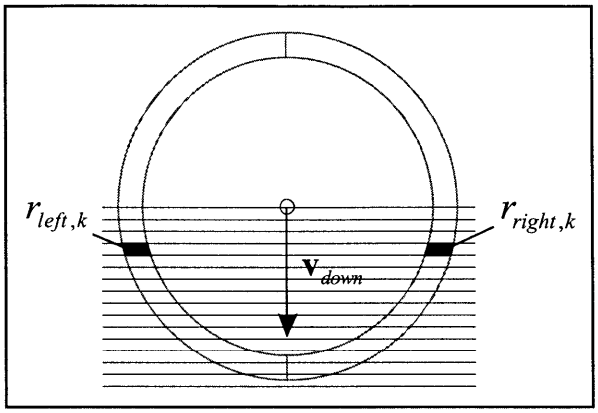


Figure 2.6. Assignment of pixels to rows $r_{left,k}$ and $r_{right,k}$

For each row the summed intensity SI is computed as the sum of each individual pixel's grayscale intensity I :

$$SI_{left,k} = \sum_{\mathbf{p} \in r_{left,k}} I(\mathbf{p}). \qquad (2.8)$$

$$SI_{right,k} = \sum_{\mathbf{p} \in r_{right,k}} I(\mathbf{p}) \qquad (2.9)$$

Two $n \times 1$ arrays of summed row intensities are thus formed.

2.3.3 Terrain Interface Identification

A one-dimensional spatial filter is employed to smooth the intensity arrays and reduce the effects of noise. Here the summed row intensities are weighted by the number of pixels in a row ($c_{left,k}$ and $c_{right,k}$), to minimize the influence of noise in low pixel-count rows.

A Gaussian filter with variance $m/2$ is approximated by a binomial distribution w:

$$w_{l,m} = \frac{(2m)!}{2^{2m}(m+l)!(m-l)!} \tag{2.10}$$

where $l \in \{-m,...,m\}$.

This filter is applied to the summed pixel intensities to produce a pair of filtered intensity arrays FI_{right} and FI_{left}:

$$FI_{left,k} = \frac{\sum_{l=-m}^{m} w_{l,m} SI_{left,k+l}}{\sum_{l=-m}^{m} w_{l,m} c_{left,k+l}}. \tag{2.11}$$

$$FI_{right,k} = \frac{\sum_{l=-m}^{m} w_{l,m} SI_{right,k+l}}{\sum_{l=-m}^{m} w_{l,m} c_{right,k+l}} \tag{2.12}$$

A representative plot of filtered intensity vs. angular position can be seen in Figure 2.7. In this example a dark wheel is partially submerged in light terrain. The terrain interface location is computed as the point of maximum change in intensity between rows. This exploits the fact that the wheel rim is a different intensity from the terrain.

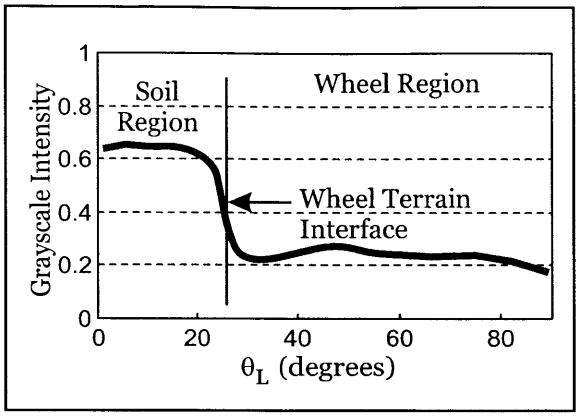


Figure 2.7. Sample plot of average pixel intensity vs. angular position

The row index with the maximum change in intensity is simply:

$$K_{left} = \arg\max_{k}\left(FI_{left,k} - FI_{left,k-1}\right). \quad (2.13)$$

$$K_{right} = \arg\max_{k}\left(FI_{right,k} - FI_{right,k-1}\right) \quad (2.14)$$

The interface angles θ_L^* and θ_R^* are then calculated from K_{left} and K_{right} as follows:

$$\theta_L^* = \cos^{-1}\left(\left(K_{left} + 0.5\right)\frac{\Delta y}{r_{wheel}}\right). \quad (2.15)$$

$$\theta_R^* = \cos^{-1}\left(\left(K_{right} + 0.5\right)\frac{\Delta y}{r_{wheel}}\right) \quad (2.16)$$

2.3.4 Detection of Rigid Terrain

The algorithm assumes the presence of a unique maximum change in intensity along the wheel rim. In practice a unique maximum can nearly always be found at the wheel-terrain interface. However, errors can occur when the wheel contacts a rigid patch of terrain or rigid objects such as rocks. In those situations it is possible that none of the rim is

occluded by terrain (see Figure 2.8). Sensor noise and lighting effects will then lead to false maxima and thus erroneous sinkage values.

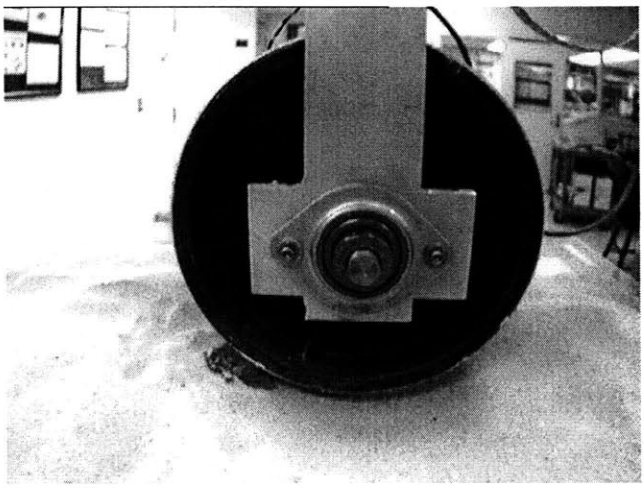


Figure 2.8. Wheel rigidly supported by rock

This problem is addressed by appending a small set of pixels to S_{left} and S_{right}, forming rows with index $n+1$. These pixels are taken from an image region below the center of the wheel rim, and are expected to be representative of the local terrain. If the wheel is resting on a rigid terrain surface, the maximum change in intensity will occur between rows n and $n+1$. This corresponds to a sinkage angle of zero. If the maximum change in intensity occurs in the wheel rim region, the algorithm will operate normally and return the appropriate sinkage angle.

2.4 Experimental Results

2.4.1 FSRL Wheel-Terrain Interaction Testbed

Experiments to validate the algorithm have been performed on the FSRL Wheel-Terrain Interaction Testbed shown in Figure 2.9, and described in detail in Appendix C. The testbed consists of a driven wheel mounted on an undriven vertical axis. Horizontal

movement of the wheel is controlled, and the vertical load on the wheel can be changed. A camera is mounted to the testbed so that it translates horizontally with the wheel, but not vertically. This configuration emulates the motion of a camera attached to the body of a rover, where a wheel may move within the field of view, but never leaves the frame. The vertical position of the wheel relative to the camera is sensed with a potentiometer. Feedback from this sensor is used to determine the transformations between the reference frames, just as suspension configuration sensors would aboard a rover.

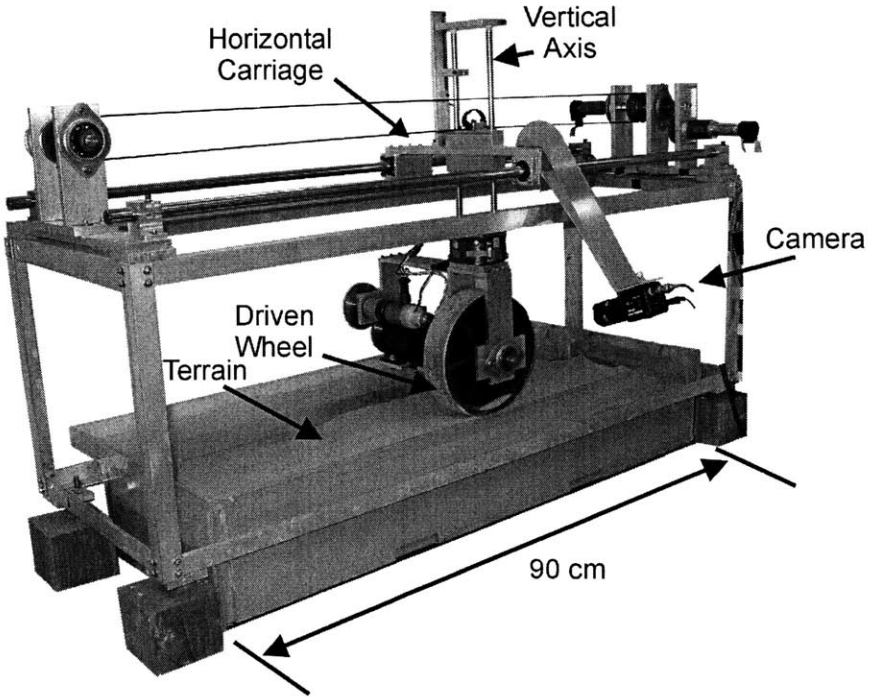


Figure 2.9. FSRL Wheel-Terrain Interaction Testbed with black wheel

Images from the testbed camera were collected under six different terrain and lighting conditions to test the algorithm. Wheel slip and terrain unevenness conditions were varied among these data sets, as were the color of the terrain and the presence or absence of rocks. Lighting was varied from uniform, diffuse illumination to a point

source which cast sharp shadows. Nineteen images were collected at two second intervals for each set of conditions.

Figure 2.10 shows a representative image from each set of conditions. Image set 1 shows a wheel moving through flat bentonite clay under uniform lighting with a high slip ratio. The bentonite clay is dry and granular with a light tan color. The high slip ratio causes the wheel to dig itself into the clay, yielding a wide range of sinkages.

Set 2 shows a wheel moving with high slip ratio through flat JSC Mars-1 soil simulant under uniform lighting (Allen *et al*, 1998). JSC Mars-1 soil simulant is a brown mixture of weathered volcanic ash particles developed to simulate the color and consistency of Martian soil. The dark color tests the algorithm's effectiveness in situations with low contrast between the wheel and terrain.

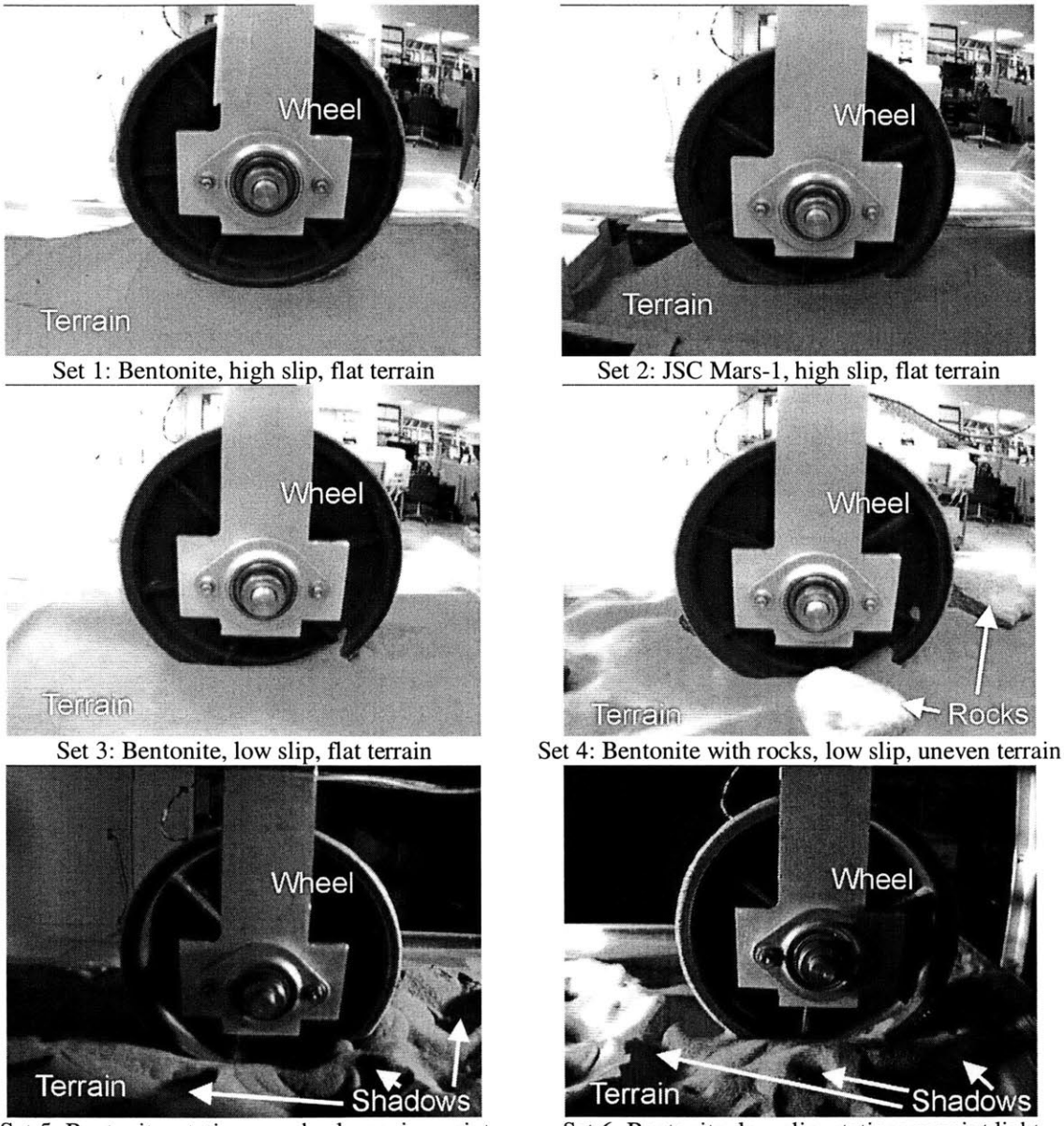
Set 3 shows a wheel moving through flat bentonite clay under uniform lighting with low slip and nearly constant sinkage. This simulates the conditions a rover might experience when driving over homogeneous terrain.

Set 4 shows a wheel moving through uneven bentonite clay under uniform lighting in the presence of rocks. Rocks occlude the wheel-terrain interface or appear as additional potential interfaces. They may also support the wheel rigidly and thus cause conditions with zero sinkage.

Set 5 shows a stationary, sunken wheel in uneven bentonite clay illuminated by a moving point source. The moving light source simulates the effect of a rover moving with respect to the sun or vice versa. For the first image in this set, the light source is far to the right of the wheel. The light source is gradually moved from the right side to a point above and behind the camera. It is then gradually moved down and to the left of the

wheel, so that in the last image the light source is just above the surface of the terrain, far to the left of the wheel.

Set 6 shows a wheel moving through uneven bentonite clay illuminated by a stationary point source casting sharp shadows. This simulates the most difficult conditions for the algorithm, which would occur when the sun is low in a clear sky.



Set 1: Bentonite, high slip, flat terrain

Set 2: JSC Mars-1, high slip, flat terrain

Set 3: Bentonite, low slip, flat terrain

Set 4: Bentonite with rocks, low slip, uneven terrain

Set 5: Bentonite, stationary wheel, moving point source

Set 6: Bentonite, low slip, stationary point light source

Figure 2.10. Sample wheel sinkage measurement images from image sets 1-6

2.4.2 Passive Lighting Results

Representative results are shown in Figure 2.11 and Figure 2.12. (Full results are included in Appendix E.) Figure 2.11 shows the actual and visually-measured sinkage as a percentage of the wheel radius for image set 1. The x axis is the index of the image being analyzed, corresponding to one of the 19 images collected in each image set. The y axis shows the sinkage as a percentage of the wheel radius. The left and right plots show the sinkage for the left and right sides of the wheel, respectively, for the same images. It may be observed that the visually-measured sinkage matches the actual sinkage very accurately.

Figure 2.12 shows similar results for image set 5. Since the wheel is stationary, constant sinkage should result even though the image changes due to the moving light source. The visually-measured sinkage is close to the actual sinkage for most of the images. Sources of error are discussed later in this section.

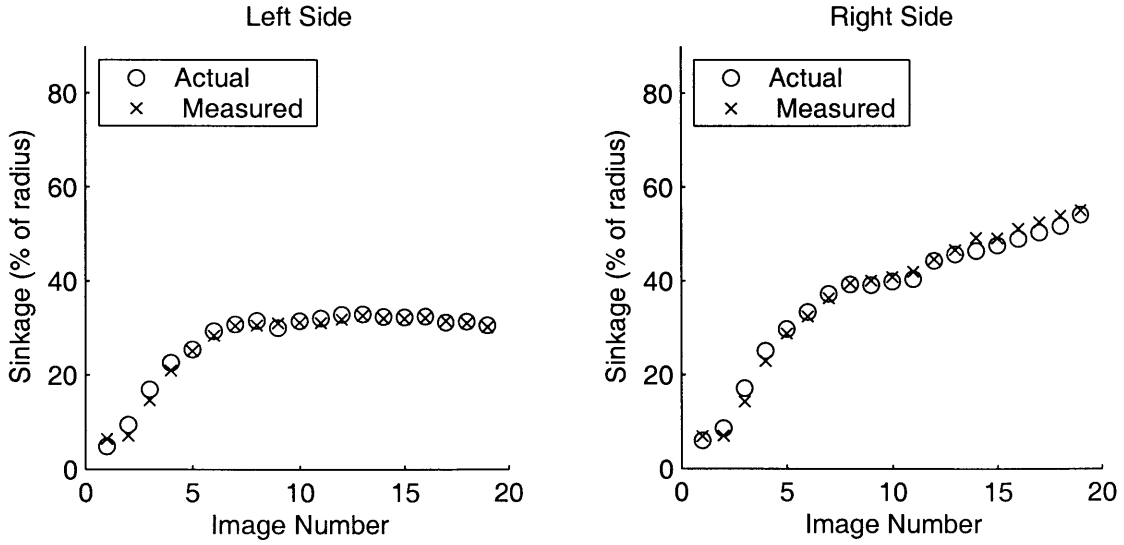


Figure 2.11. Actual and visually-measured sinkages for image set 1 (Bentonite, high-slip, flat terrain, fully-lit)

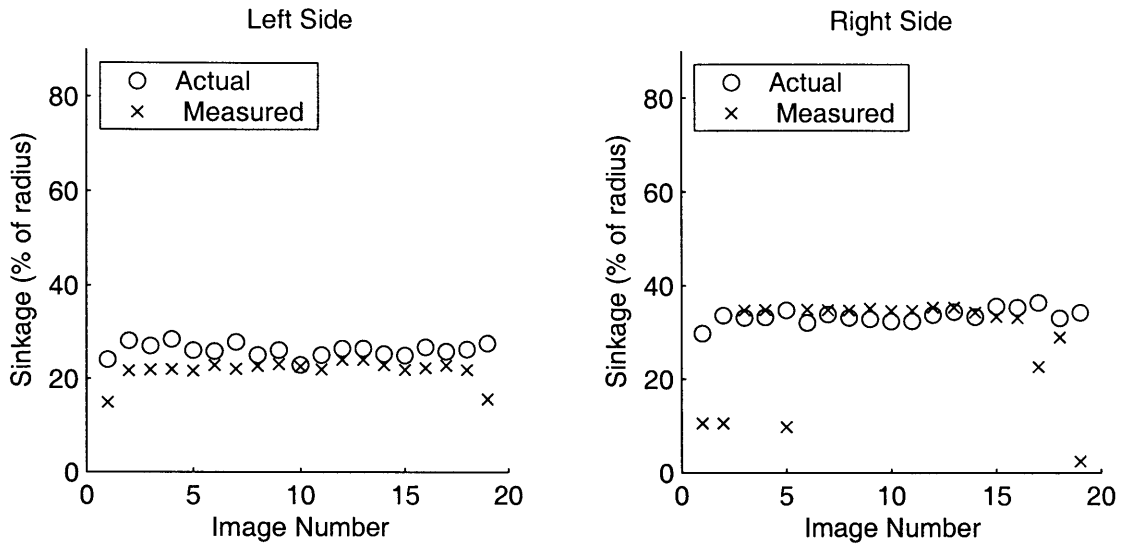


Figure 2.12. Actual and visually-measured sinkages for image set 5 (Bentonite, stationary wheel, moving light source)

Table 2.1 summarizes the results of sinkage angle measurement for all six image sets. Error is computed as the difference between the visually-measured sinkage and the actual sinkage, as a percentage of the wheel radius, for the left and right terrain interface angles.

Image Set	Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
1	1.08	1.61
2	2.40	2.46
3	2.33	2.48
4	5.21	2.06
5	5.10	12.10
6	8.85	14.01

Table 2.1. Visual wheel sinkage measurement results

The algorithm detected wheel sinkage under a wide range of conditions with good accuracy. Errors in set 4 were caused by rocks occluding the wheel-terrain interface. While these small errors could be mitigated by a texture- or geometry-based rock

detection algorithm, adding such an algorithm would increase the computational requirements.

A more significant error source was uneven lighting. Sets 5 and 6 show substantially higher RMS error than sets 1-4. Reflections off the wheel rim occasionally caused misidentifications of the wheel-terrain interface. Other problems, such as those observed in Figure 2.12, were the result of shadows falling on uneven terrain itself. However, these errors tended to appear as easily-identifiable outliers (i.e. the errors appeared as large anomalous changes in the visually-measured angle) that could be mitigated by intelligent filtering.

2.4.3 Active Lighting Results

As described above, the algorithm performs poorly in situations with uneven lighting, which might occur when the sun is low in the sky, casting sharp shadows on the wheel. A method for addressing errors caused by uneven lighting is to employ active lighting. In this approach, a light source aboard the rover is used to illuminate the wheel.

Figure 2.13 shows a sample image from a series in which a strobe was used to illuminate the wheel-terrain interface (set 7). Here the wheel was driving through topsoil, and the wheel rim was colored yellow to provide contrast. Representative results are plotted in Figure 2.14. In shadowy conditions similar to sets 5 and 6, where the RMS errors ranged from 5% to 14% of the wheel radius, tests using a strobe to illuminate the wheel resulted in RMS errors less than 2% of the wheel radius.

The tradeoff between the additional hardware requirements and the improved measurement accuracy must be considered when choosing whether to implement this algorithm with active lighting on a rover.

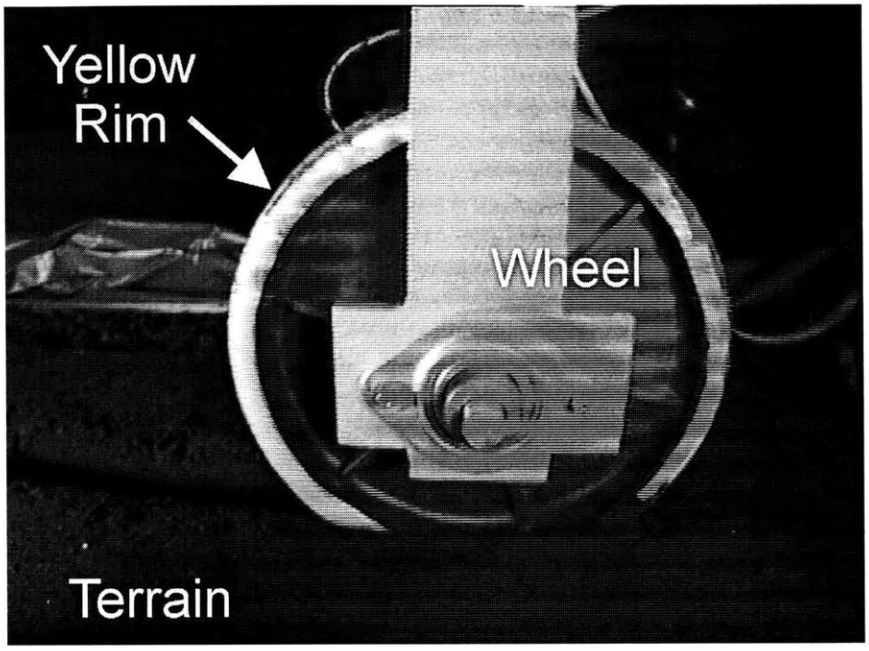


Figure 2.13. Sample image from set 7 (active lighting)

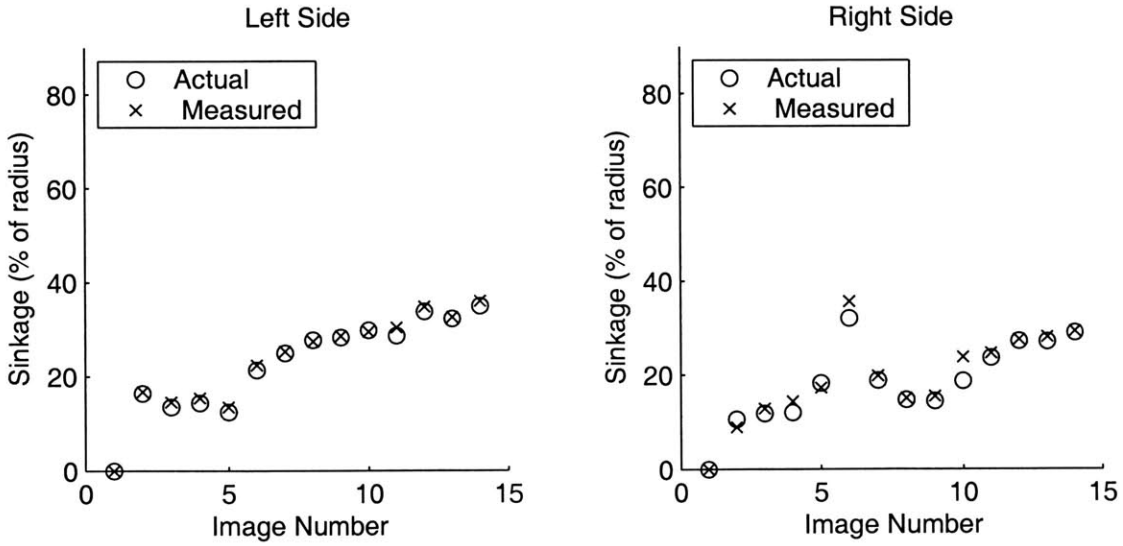


Figure 2.14. Actual and visually-measured sinkages for image set 7 (active lighting)

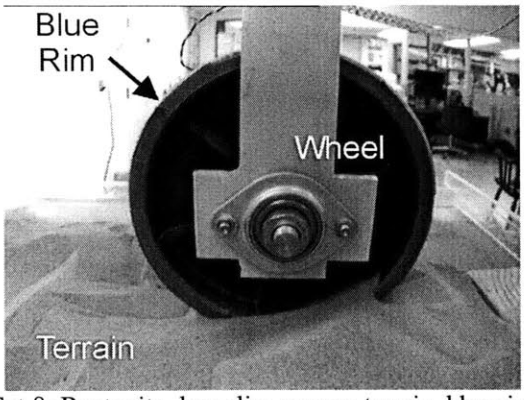
2.4.4 Computational Requirements

Computational requirements for this algorithm are minimal. A Matlab version of the algorithm processed images at 3 Hz on a Pentium III 933MHz PC. An optimized compiled implementation would be expected to run significantly faster. For image sets and settings discussed here, approximately 90,000 floating point operations were required per image. A standalone executable version of the code required 80 KB of memory for the program, with an additional 60 KB of memory for execution. These low computational requirements suggest that the algorithm is suitable for on-board implementation on a planetary rover with limited computing power.

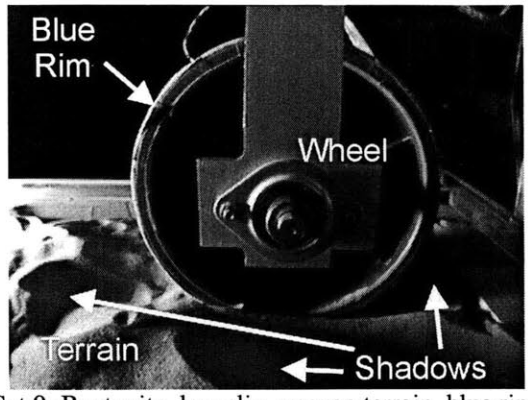
2.4.5 Color Images

Visual sinkage measurement using data from a color camera was also explored experimentally. The wheel rim was colored blue, to aid differentiation between wheel and terrain, and three sets of 19 images were collected. These image sets are shown in Figure 2.15. Set 8 shows a blue-rimmed wheel rolling through uneven bentonite with low slip. This should be compared with the non-color image set 3. Set 9 shows a blue-rimmed wheel moving with low slip through uneven terrain with a stationary point light source. This is most similar to the non-color image set 6. Set 10 shows a blue-rimmed wheel moving with low slip through uneven bentonite with rocks. Results from this image set should be compared with non-color results from image set 4.

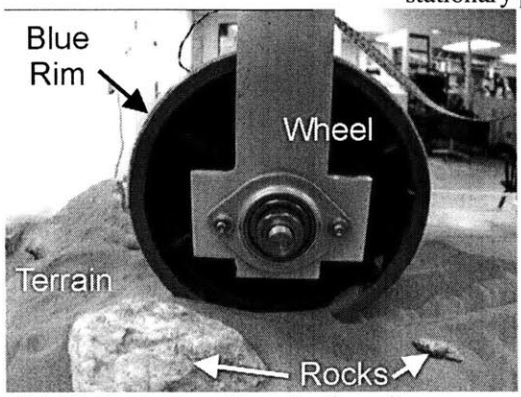
Techniques for exploiting color data included examining various distance metrics in RGB space from the known wheel color to the terrain color. Results from one of the color-based techniques are shown in Table 2.2.



Set 8: Bentonite, low slip, uneven terrain, blue rim



Set 9: Bentonite, low slip, uneven terrain, blue rim, stationary point light source



Set 10: Bentonite with rocks, low slip, uneven terrain, blue rim

Figure 2.15. Sample wheel sinkage measurement images from image sets 8-10 (blue rim)

Image Set	Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
8	3.30	2.61
9	26.3	29.45
10	3.60	2.51

Table 2.2. Visual wheel sinkage measurement results for color-based algorithm

As demonstrated by the results from image set 8, none of the color-based methods have difficulty identifying the wheel-terrain interface in fully-lit conditions with uniform terrain. Several of them appear to be more robust than the grayscale algorithm in rejecting errors caused by intervening rocks. This is may be seen by comparing the color-based results from image set 10 (3.6% and 2.5% RMS error) to the grayscale-based

results from image set 4 (5.2% and 2.0% RMS error). However the color-based techniques appear equally likely to return inaccurate results under low light conditions with shadows. This is shown by the significantly higher RMS errors for image set 9 (26% and 29%) as compared to those for the grayscale results for image set 6 (8.8% and 14%).

In general there appears to be little advantage to employing color-based visual sinkage measurement over grayscale, when the nearly threefold increase in computation time is considered.

2.5 Summary and Conclusions

A vision-based method for measuring the sinkage of a rover wheel in deformable terrain has been presented. The method detects the location of the wheel-terrain interface by finding the maximum change in intensity along the wheel rim. The method is computationally efficient and uses a single color or grayscale vision sensor, making it potentially suitable for systems with limited computational resources such as planetary rovers. Experimental results have shown the method to be accurate and relatively robust to lighting variations. It has also been shown that active lighting can be implemented to further improve measurement accuracy.

Chapter 3

Vibration-based Terrain Classification

3.1 Introduction

The second sensory method investigated in this thesis is vibration-based terrain classification. This method is intended to identify gross terrain classes for local terrain based on the vibrations induced in the rover structure during a traverse. For example, it might identify the local terrain as being either "sand," "clay," or "gravel."

Vibration-based terrain classification uses an accelerometer attached to the rover structure to sense vibrations caused by wheel-terrain interaction. Vibrations in the rover structure are affected by the physical nature of the terrain. (For example, it is obvious that driving a passenger car over a dirt road sounds different from driving over a loose sand or gravel.) These vibration signals are recorded as the vehicle drives over the terrain, and the algorithm returns the terrain class which induced the vibrations. As the vehicle drives, a one-dimensional map of the terrain class along the driving path can be assembled on-line in real time.

Terrain classification based on vibrations in the rover structure has several advantages over the more traditional vision-based terrain classification addressed in (Bellutta *et al*, 2000) and (Rasmussen, 2002). Its classification accuracy is independent of the lighting conditions, a significant challenge to visual classification approaches. In

addition, vibration-based terrain classification has potential to detect changes in terrain buried beneath a shallow surface cover, for instance, the fine drift material on Mars or sparse vegetation on Earth. This could be advantageous for improving the accuracy of rover wheel-odometry position estimates as well as identifying locations of submerged features for scientific study.

Research into vibration-based terrain classification was recently done at Florida State University for application to high-speed autonomous vehicles on Earth (Sadhukhan & Moore, 2003; Sadhukhan, 2004). Section 3.5 compares the FSU work to the algorithm described below.

3.2 Algorithm Overview

The algorithm presented in this thesis takes a signal recognition approach to classifying the terrain based on vibration signals. This is in contrast to an approach that uses a solid mechanics or finite element model to analytically predict how the rover structure will vibrate in response to interaction with terrain of a given type. The algorithm presented here learns to recognize different terrain types based on example vibration data provided during an *a priori* training phase. During training, the algorithm is provided with vibration data sets labeled by terrain class. Once the algorithm has completed its training, it stores the distilled results of the learning in on-board memory. Then it can quickly classify vibration signals on-line as belonging to one terrain or another that it saw in training. An overview schematic of the algorithm is shown in Figure 3.1.

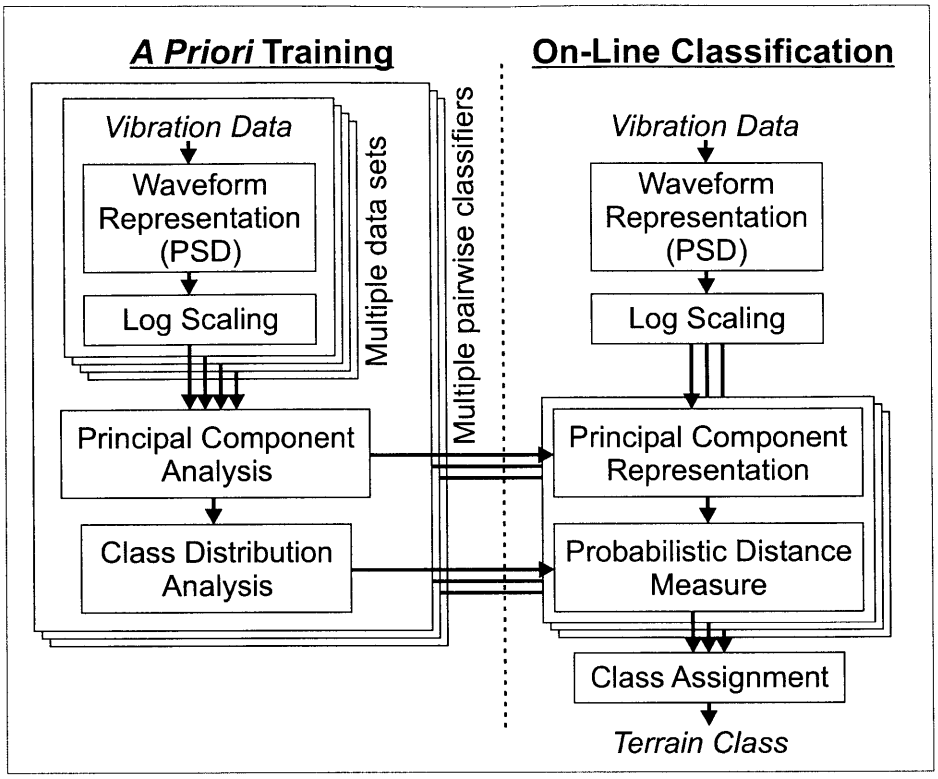


Figure 3.1. Overview flowchart for vibration-based terrain classification algorithm

In this approach, vibration signals are divided into short segments. These are then converted from time-domain voltage signals into power spectral densities. Further analysis is performed in the Fourier domain. Log scaling of the power spectral magnitude is used to reduce the dominating effect of high-magnitude frequency components.

With the signals represented as a time series of Fourier spectra, training is a matter of dividing a high-dimensional space (i.e. the Fourier coefficients) into regions associated with an individual terrain class. To reduce the dimensionality of the comparison, principal component analysis (discussed in Appendix B) is used. Here only the first k components are retained. The value of k is set based on previous experiments with the system. Note that principal components are computed during the training phase. These same principal components are used during the classification phase.

To define class boundaries in this principal component space, linear discriminant analysis (Balakrishnama, 1998) is used as a pairwise classifier. In this approach, sub-classifiers are created to classify terrain as being one of two possible terrains. Separate sub-classifiers are used for each possible pair of terrains. For example, in the case of sand, gravel, and clay, one classifier would distinguish gravel from sand, another would distinguish gravel from clay, and a third would distinguish sand from clay. Linear discriminant analysis considers both the distribution of samples within a single terrain class and the separation between class means to compute an optimal vector along which to compare samples. Classification of a test sample can be done by projecting its principal component representation onto this vector. A number of simple classifiers are available to address the resulting one-dimensional classification problem.

To accommodate classification of more than two terrains, a voting scheme is used. Each pairwise classifier can cast a "vote" for one of the two terrains it distinguishes, or remain "undecided." The winning terrain class is returned.

3.3 Terrain Classification Algorithm

The terrain classification algorithm may be broken into two separate phases, *a priori* training and on-line classification. *A priori* training is computationally intensive and is performed off-line. On-line classification is computationally efficient and is performed during a rover traverse. These phases are described below.

3.3.1 *A Priori* Training

In the *a priori* training phase, the algorithm learns to recognize vibration signatures from various terrain types. These are chosen to correspond to terrains of interest that a robot might encounter during field operations.

The first step in the *a priori* training is to collect vibration data from the terrains to be classified. This data is in the form of a time series of the voltage output of the accelerometer or contact microphone. Data should be collected for the terrain under a range of conditions spanning those for which the classifier is expected to perform (for example, under varying speeds, slip conditions, and loads).

This time series is broken into short segments. The duration of these segments should be scaled to the physical scenario (e.g. wheel diameter, spatial variations). The power spectral density (PSD) of each of these segments is then computed using Welch's method (Welch, 1967), and a log-scaled version of this PSD is stored in a matrix. For example, data for sand would be stored in a matrix $\mathbf{Y}_{sand}$ as:

$$\mathbf{Y}_{sand} = \begin{bmatrix} y_{sand,fmin,t=1} & \cdots & y_{sand,fmin,t=n} \\ \vdots & \ddots & \vdots \\ y_{sand,fmax,t=1} & \cdots & y_{sand,fmax,t=n} \end{bmatrix} \quad (3.1)$$

In this representation, each column contains the log PSD components for a range of frequencies for a given time segment. Each row contains the log PSD components for all time segments corresponding to a given frequency. (Note that this matrix may be visualized as a spectrogram like the one in Figure 3.2, simply by assigning a grayscale intensity proportional to the value of each element. This plot is a convenient way to view the time-varying nature of frequency components in a signal.)

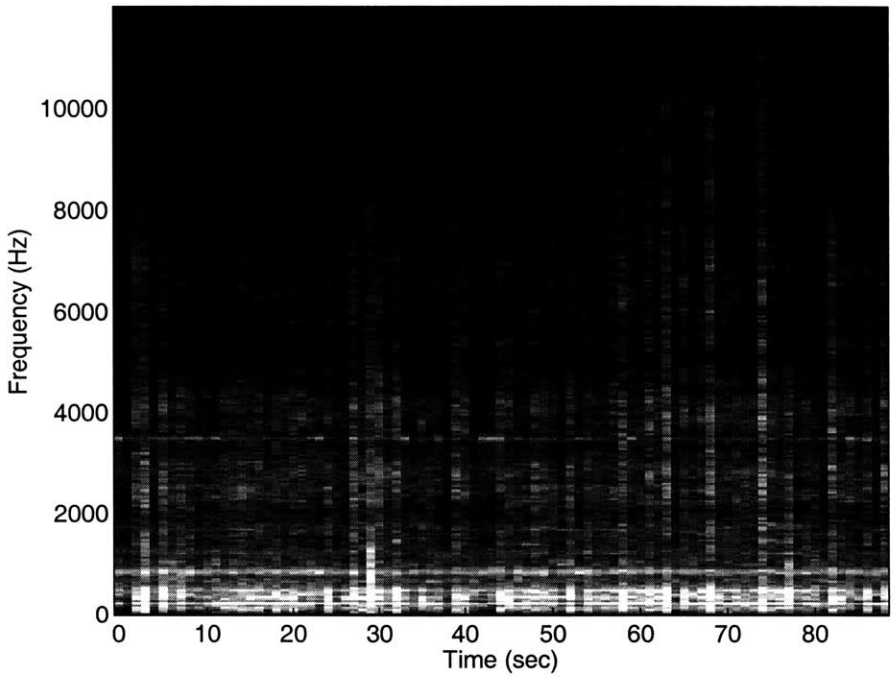


Figure 3.2. Sample spectrogram of terrain vibration data

A separate classifier is used to distinguish between each pair of terrains. For each classifier, the following steps are performed to produce the discrimination vector and terrain class statistics. For illustration, the pairwise classifier presented below is intended to distinguish between gravel and sand.

Two matrices describing the training data in the pairwise classes are combined to form a complete record of the training data: $\mathbf{Y} = [\mathbf{Y}_{sand} \quad \mathbf{Y}_{gravel}]$, $\mathbf{Y} \in \Re^{m \times n}$. The rows of $\mathbf{Y}$ are then mean-adjusted to form the matrix $\hat{\mathbf{Y}}$:

$$\bar{\mathbf{y}} = \begin{bmatrix} \text{mean}(y_{fmin}) \\ \vdots \\ \text{mean}(y_{fmax}) \end{bmatrix} \tag{3.2}$$

$$\hat{\mathbf{Y}} = \mathbf{Y} - \bar{\mathbf{y}}[1 \quad \cdots \quad 1] \tag{3.3}$$

The mean of each row of the matrix $\hat{\mathbf{Y}}$ is equal to zero. This matrix is used to analyze the variation of data signals represented in the entire training data set.

Singular value decomposition is then used to separate $\hat{\mathbf{Y}}$ into three matrices, $\mathbf{U}_a$, $\mathbf{S}_a$, and $\mathbf{V}_a$:

$$\hat{\mathbf{Y}} \xrightarrow{\text{SVD}} (\mathbf{U}_a, \mathbf{S}_a, \mathbf{V}_a^T). \tag{3.4}$$

Here, $\mathbf{U}_a$ is a unitary matrix with the principal components of $\hat{\mathbf{Y}}$ as columns. $\mathbf{S}_a$ is a diagonal matrix of singular values. $\mathbf{V}_a$ is a unitary matrix with the principal components of $\hat{\mathbf{Y}}^T$ as columns. Further details regarding singular value decomposition and principal component analysis can be found in Appendix B.

The matrix $\mathbf{U}_a$ is assumed to be composed of orthogonal signal and noise subspaces. To represent the signal subspace, the first k columns of $\mathbf{U}_a$ (i.e. the first k principal components of $\hat{\mathbf{Y}}$) are used, and are stored in the matrix $\mathbf{U}_{\text{signal}}$. Similarly, the upper-left $k \times k$ block of $\mathbf{S}_a$ represents the singular values associated with the signal space, and will be referred to as $\mathbf{S}_{\text{signal}}$. Using too many principal components here can be detrimental, especially with a limited amount of training data. This would train the algorithm to recognize the noise in the training data to the detriment of its ability to classify new data. In practice we have used $k = 15$, as it appears to give good signal

representation without overfitting. In our experiments, the first 15 principal components accounted for approximately 90% of the variance.

The two matrices $\mathbf{U}_{\mathbf{signal}}$ and $\mathbf{S}_{\mathbf{signal}}$ can be considered to be a map from the full frequency space ($\Re^m$) to the signal space ($\Re^k$). The signal space mappings of the separate data sets $\mathbf{Y}_{sand}$ and $\mathbf{Y}_{gravel}$ are computed as:

$$\mathbf{W}_{sand} = \mathbf{S}_{\mathbf{signal}}{}^{-1}\mathbf{U}_{\mathbf{signal}}{}^{\mathbf{T}}\mathbf{Y}_{sand} \quad (3.5)$$

$$\mathbf{W}_{gravel} = \mathbf{S}_{\mathbf{signal}}{}^{-1}\mathbf{U}_{\mathbf{signal}}{}^{\mathbf{T}}\mathbf{Y}_{gravel}. \quad (3.6)$$

In this representation each column of $\mathbf{W}_{sand}$ and $\mathbf{W}_{gravel}$ corresponds to an individual time segment. Each row of $\mathbf{W}_{sand}$ and $\mathbf{W}_{gravel}$ corresponds to a principal component (i.e. a linear combination of frequency components). Taking each column as a vector to a point in k-dimensional space, $\mathbf{W}_{sand}$ and $\mathbf{W}_{gravel}$ represent point clouds with means $\overline{\mathbf{w}}_{sand}$ and $\overline{\mathbf{w}}_{gravel}$, respectively:

$$\overline{\mathbf{w}}_{sand} = \frac{1}{n}\mathbf{W}_{sand}\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} \quad (3.7)$$

$$\overline{\mathbf{w}}_{gravel} = \frac{1}{n}\mathbf{W}_{gravel}\begin{bmatrix}1\\ \vdots \\ 1\end{bmatrix} \quad (3.8)$$

Figure 3.3 shows the point clouds plotted on the plane spanned by the second and third principal components. (The coefficient of the first principal component is not plotted here because it did not differ significantly between the two data sets.) Here it can be seen that the point clouds from the two data sets lie in separate regions of the space.

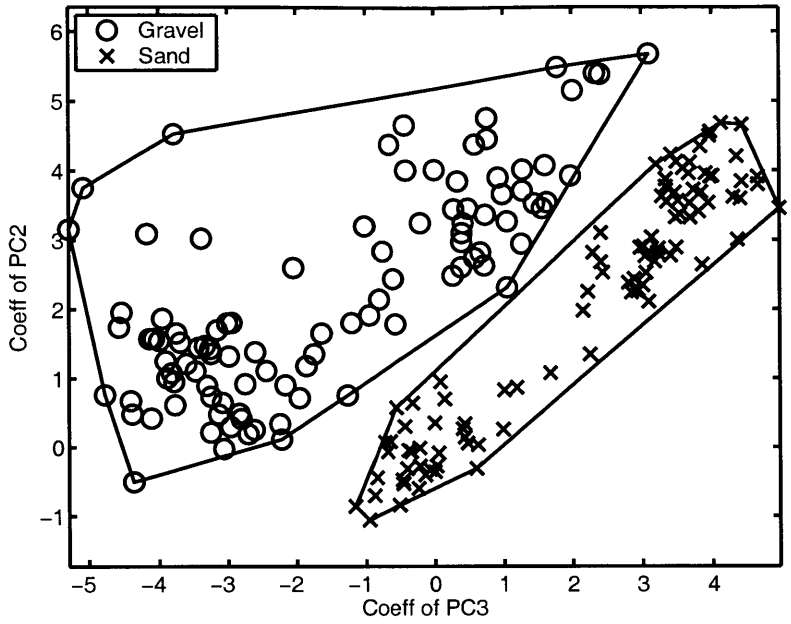


Figure 3.3. Gravel and sand training data plotted on the plane of the second and third principal components

Writing the vectors $\mathbf{W}_{sand}$ and $\mathbf{W}_{gravel}$ in terms of the principal components has reduced the dimensionality of the space for comparing the two terrain types, but still leaves a *k*-dimensional space in which to compare any new vibration signal with the training data. Ideally, we would like to have a single scalar value associated with a signal, which is fully able to capture the difference between the two terrains. Such a scalar value is here referred to as the discrimination metric.

One candidate discrimination metric to classify an arbitrary vector $\mathbf{w}$ (corresponding to a combination of principal components) would be the dot product of $\mathbf{w}$ with the difference between the means of the training data sets, $\overline{\mathbf{w}}_{sand} - \overline{\mathbf{w}}_{gravel}$:

$$d_{proposed}(\mathbf{w}) = (\overline{\mathbf{w}}_{sand} - \overline{\mathbf{w}}_{gravel}) \cdot \mathbf{w} \qquad (3.9)$$

This would reduce the *k*-dimensional classification problem into a 1-dimensional classification problem—classifying the projection of $\mathbf{w}$ onto the line between the means

of the training data sets. Figure 3.4 shows the line between the means and a histogram illustrating the density of points projected on it. The 1-dimensional classification problem can be reduced to dividing the projection of points from one class from the projection of points from the other class.

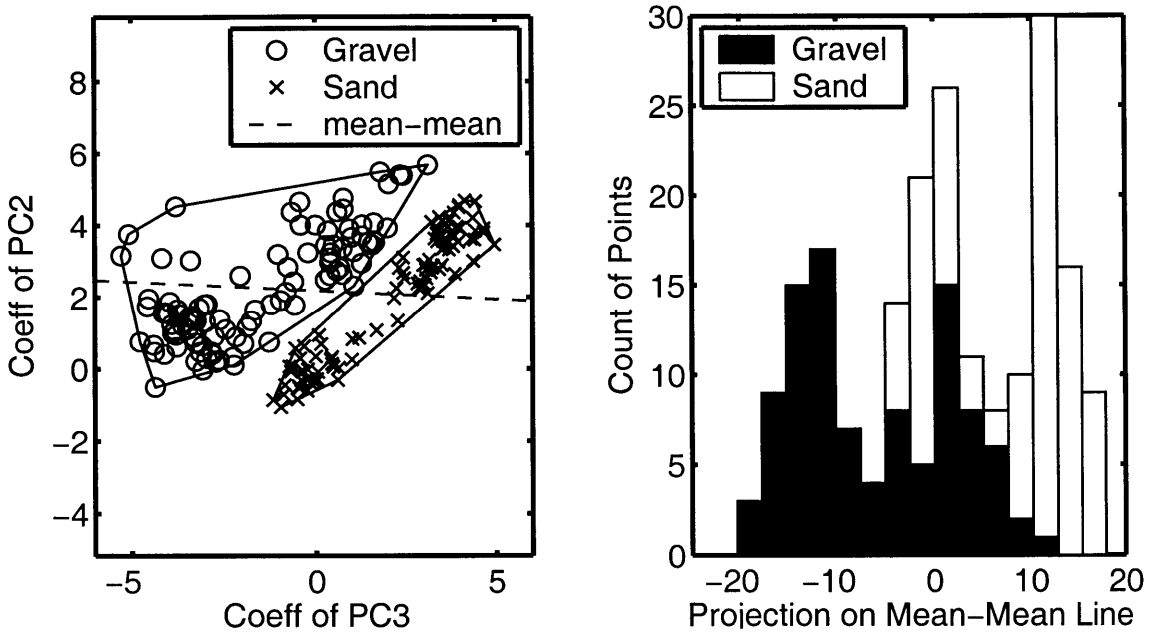


Figure 3.4. Projection of point clouds onto the line between the means

Projecting the points onto the line between the means rarely yields satisfactory results, however, because the distributions of $\mathbf{W}_{sand}$ and $\mathbf{W}_{gravel}$ may have very different scales along different dimensions. This is the case in Figure 3.4. Here there is obviously a line which separates the classes, but it is not perpendicular to the line between the means. The solution is to scale the space so that the class distributions are more uniform across all dimensions, making the situation appear as in Figure 3.5. Here, projecting onto the line between the means successfully discriminates between the two classes.

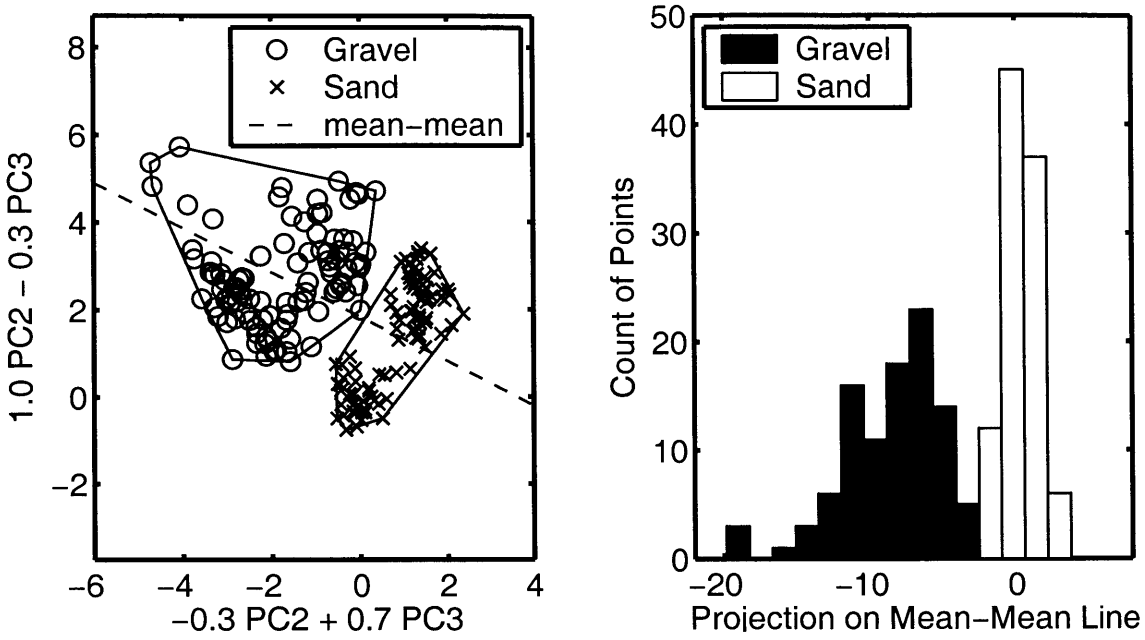


Figure 3.5. Projection onto the line between the means in a scaled space

To determine an appropriate scaling of the *k*-dimensional signal space, it is important to examine the distribution of the points within a terrain class. This is accomplished by performing a second singular value decomposition, this time on a matrix $\hat{\mathbf{W}}$, formed by merging mean-adjusted matrices $\hat{\mathbf{W}}_{sand}$ and $\hat{\mathbf{W}}_{gravel}$:

$$\hat{\mathbf{W}}_{sand} = \mathbf{W}_{sand} - \overline{\mathbf{w}}_{sand}[1 \quad \cdots \quad 1] \tag{3.10}$$

$$\hat{\mathbf{W}}_{gravel} = \mathbf{W}_{gravel} - \overline{\mathbf{w}}_{gravel}[1 \quad \cdots \quad 1] \tag{3.11}$$

$$\hat{\mathbf{W}} = [\hat{\mathbf{W}}_{sand} \quad \hat{\mathbf{W}}_{gravel}] \tag{3.12}$$

$$\hat{\mathbf{W}} \xrightarrow{\text{SVD}} (\mathbf{U}_b, \mathbf{S}_b, \mathbf{V}_b^T) \tag{3.13}$$

Here we focus on $\mathbf{V}_b^T$. Each column of $\mathbf{V}_b^T$ is associated with a particular time segment; each row corresponds to a combination of principal components (i.e. a linear combination of frequency components, still within the signal space). More importantly, due to the

unitary nature of $\mathbf{V}_b^T$, the norm of each row of $\mathbf{V}_b^T$ is equal, meaning that the standard deviations are equal for all of the dimensions. The scaling of the space which produced $\mathbf{V}_b^T$ is therefore appropriate for classifying data. Since $\mathbf{V}_b^T$ can be written as

$$\mathbf{V}_b^T = \mathbf{S}_b^{-1}\mathbf{U}_b^T\mathbf{W} \qquad (3.14)$$

the transformation from the original signal space to this scaled signal space can be written as $\mathbf{S}_b^{-1}\mathbf{U}_b^T$.

The discrimination metric $d(\mathbf{w})$ is thus defined as a dot product in this scaled signal space:

$$d(\mathbf{w}) = (\mathbf{S}_b^{-1}\mathbf{U}_b^T(\overline{\mathbf{w}}_{sand} - \overline{\mathbf{w}}_{gravel})) \cdot (\mathbf{S}_b^{-1}\mathbf{U}_b^T\mathbf{w}) \qquad (3.15)$$

for an arbitrary vector $\mathbf{w}$ in the signal space. Putting the discrimination metric in terms of an arbitrary vector $\mathbf{y}$ in the frequency space (i.e. $\mathbf{y}$ is the log PSD of a data segment):

$$d(\mathbf{y}) = (\mathbf{S}_b^{-1}\mathbf{U}_b^T(\overline{\mathbf{w}}_{sand} - \overline{\mathbf{w}}_{gravel})) \cdot (\mathbf{S}_b^{-1}\mathbf{U}_b^T(\mathbf{S}_{signal}^{-1}\mathbf{U}_{signal}^T\mathbf{y})). \qquad (3.16)$$

Rewriting the dot product as a matrix multiplication yields

$$d(\mathbf{y}) = (\overline{\mathbf{w}}_{sand} - \overline{\mathbf{w}}_{gravel})^T\mathbf{U}_b\mathbf{S}_b^{-1}\mathbf{S}_b^{-1}\mathbf{U}_b^T\mathbf{S}_{signal}^{-1}\mathbf{U}_{signal}^T\mathbf{y} \qquad (3.17)$$

of which all but the last multiplication may be precomputed without knowledge of the vector to be classified. This precomputed row vector is labeled $\mathbf{d}$, and is a compact representation of the *a priori* training data for a pair of training classes:

$$\mathbf{d} = (\overline{\mathbf{w}}_{sand} - \overline{\mathbf{w}}_{gravel})^T\mathbf{U}_b\mathbf{S}_b^{-1}\mathbf{S}_b^{-1}\mathbf{U}_b^T\mathbf{S}_{signal}^{-1}\mathbf{U}_{signal}^T. \qquad (3.18)$$

$\mathbf{d}$ will be referred to as the "discrimination vector." This is the linear combination of the frequencies which best discriminates between the two terrain classes. Writing the discrimination metric in terms of $\mathbf{d}$ gives

$$d(\mathbf{y}) = \mathbf{d}\ \mathbf{y} \qquad (3.19)$$

The last step in the *a priori* analysis is to compute the statistics of the discrimination metrics for the training data. Row vectors of discrimination metrics are computed as:

$$d(\mathbf{Y}_{sand}) = \mathbf{d}\ \mathbf{Y}_{sand} \qquad (3.20)$$

$$d(\mathbf{Y}_{gravel}) = \mathbf{d}\ \mathbf{Y}_{gravel} \qquad (3.21)$$

The means and standard deviations of these metrics are computed as $\bar{d}_{sand}$, σ_{sand}, $\bar{d}_{gravel}$, and σ_{gravel}.

The discrimination vector and the terrain class statistics are stored for use in the on-line classification phase of the algorithm.

3.3.2 On-Line Classification

During a rover traverse, short segments of vibration sensor data are collected, of the same duration as those used in the *a priori* training. For each segment the power spectral density is computed, and the magnitude is log-scaled.

Using this log PSD, each pairwise classifier computes the discrimination metric corresponding to the vibration to be classified. It then computes the Mahalanobis distance (Mahalanobis, 1936) from this test metric to the terrain class means, $md_{sand}(\mathbf{y})$ and $md_{gravel}(\mathbf{y})$:

$$md_{sand}(\mathbf{y}) = \frac{\left|d(\mathbf{y}) - \bar{d}_{sand}\right|}{\sigma_{sand}} \qquad (3.22)$$

$$md_{gravel}(\mathbf{y}) = \frac{\left|d(\mathbf{y}) - \bar{d}_{gravel}\right|}{\sigma_{gravel}}. \qquad (3.23)$$

If the difference between the Mahalanobis distances is less than one (i.e. $\left|md_{sand}(\mathbf{y}) - md_{gravel}(\mathbf{y})\right| < 1$), the pairwise classifier labels the vibration as being "undecided." Otherwise, the pairwise classifier labels the vibration as the terrain with the smaller Mahalanobis distance.

A voting scheme merges the results of the various pairwise classifiers. In this approach, each pairwise classifier may return the label of one of the terrains it distinguishes, or it may return "undecided." If a pairwise classifier returns positive vote for a terrain class, the alternative terrain class gets a negative vote. If the pairwise classifier is undecided, both classes receive an undecided vote.

For a terrain to be positively identified, it must 1) receive more positive votes than any other terrain class, 2) receive only positive and undecided votes, and 3) receive more positive votes than undecided votes. These rules were chosen to provide a conservative estimate that would not become drastically more or less conservative with an increased number of classes. This is based on the belief that returning "unknown" is preferable to returning the wrong terrain class. Figure 3.6 shows an example of the voting algorithm positively identifying gravel. Figure 3.7 shows an example of the voting algorithm unable to positively identify a terrain.

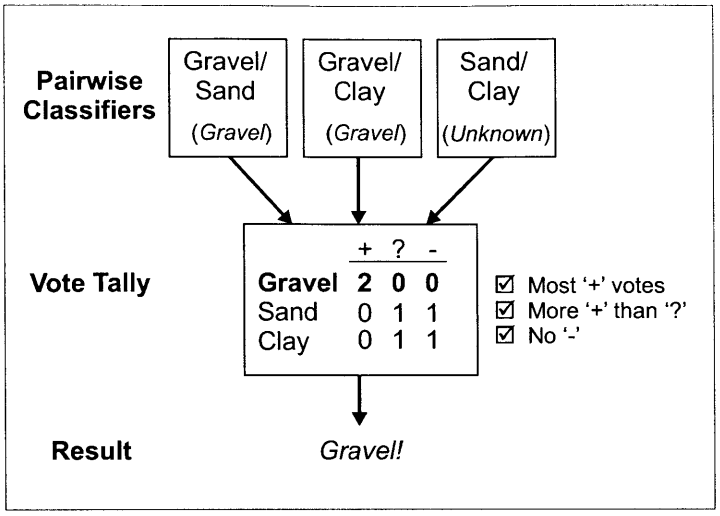


Figure 3.6. Schematic of voting positively identifying gravel

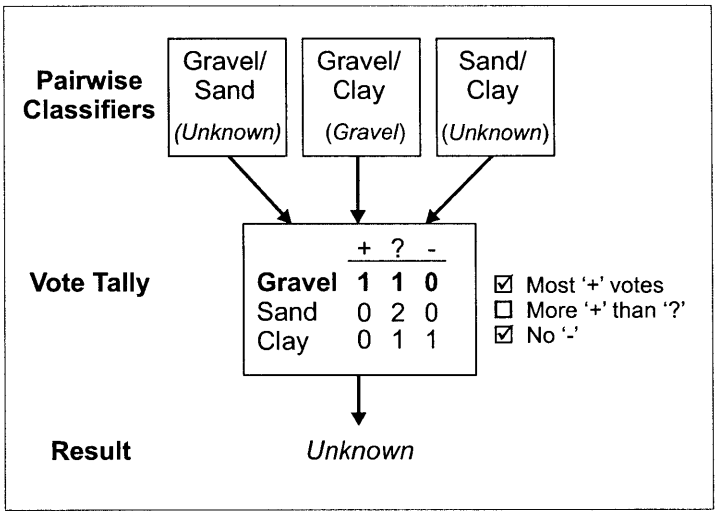


Figure 3.7. Schematic of voting resulting in unknown terrain

3.4 Experimental Results

The algorithm presented above was developed based on data collected on the Field and Space Robotics Laboratory (FSRL) Wheel-Terrain Interaction Testbed (Appendix C). It was later validated using the FSRL Technology Testbed Rover, TORTOISE (Appendix D). Detailed information about the experiments and results are presented in the following sections.

3.4.1 FSRL Wheel-Terrain Interaction Testbed

The FSRL Wheel-Terrain Interaction Testbed, shown in Figure 3.8, consists of a driven wheel mounted on an undriven vertical axis. The wheel-axis assembly is mounted on a driven carriage, so the wheel forward velocity and angular velocity can be controlled independently. These testbed experiments were conducted using a wheel from the FIDO rover supplied by the Jet Propulsion Laboratory (Schenker *et al*, 2001). For these experiments, three terrains were used: landscaping gravel, JSC Mars-1 Soil Simulant (Allen *et al*, 1998), and washed beach sand. Landscaping gravel is a mixture of small rounded pebbles ranging in size from 0.5 cm to 2 cm. JSC Mars-1 Soil Simulant is a glassy volcanic ash, developed by JSC to represent the Martian soil as observed by Viking Lander 1. It contains fine particles as well as solid clumps of particles ranging up to 4 cm. Washed beach sand is a homogeneous fine-grained sand.

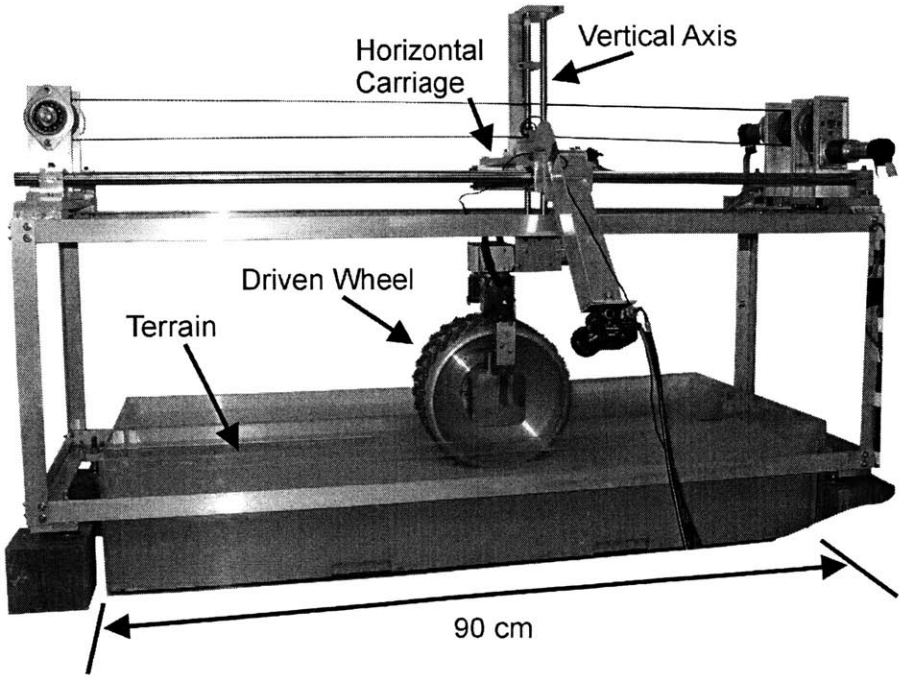


Figure 3.8. FSRL Wheel-Terrain Interaction Testbed with FIDO wheel

In these experiments, wheel forward velocity was set as a constant value for each trial. The speed ranged from 0.5 cm/s to 5 cm/s, values chosen to be similar to planned rover missions. The wheel slip ratio, defined as one minus the ratio of the wheel forward velocity to the wheel angular velocity, was varied from 0 to 0.5. The vertical load on the terrain was varied as well, from 30 N to 50 N (including the weight of the wheel). Data sets were collected in two ways: first, as the wheel traversed a single terrain over the entire length of the testbed, and second, as the wheel traversed one terrain for the first half of the testbed and a different terrain for the second half. Vibration signals were sensed using a contact microphone mounted to the frame of the wheel (see Figure 3.9). These signals were collected using a desktop computer with a sound card. Sixteen-bit samples were collected at a frequency of 44.1 kHz.

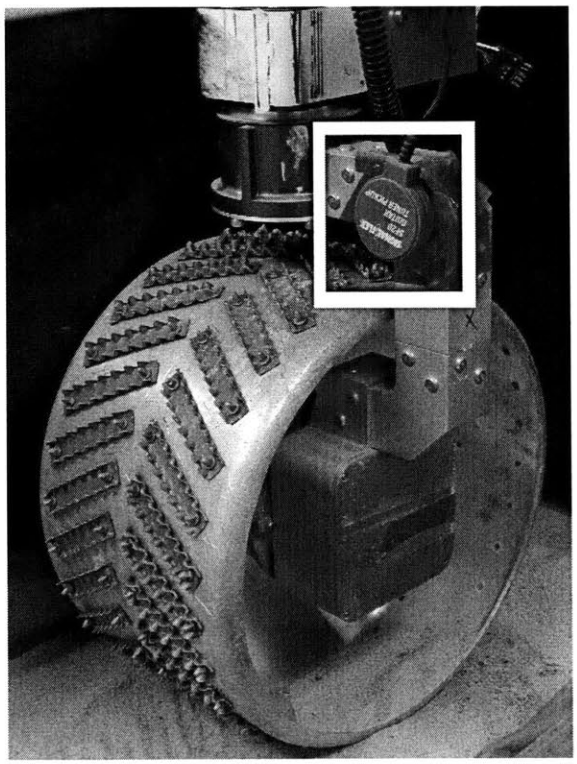


Figure 3.9. Vibration sensor mounted on the FIDO wheel in the FSRL Wheel-Terrain Interaction Testbed

A single data set consisted of vibration data recorded over a full traverse of the wheel across the 90-cm-long testbed, at a specified load, forward velocity, and angular velocity. After each traverse, the wheel was returned to its original position at one end of the testbed, and the terrain was restored to its initial flat and uncompressed state.

Once all data was collected, the algorithm was tuned using the leave-one-out approach (Jaakolla & Haussler, 1999). Tuning consisted of selecting appropriate values for 1) the range and spacing of frequency components for spectral representation, 2) the number of principal components used to represent the signal space, and 3) the discrimination thresholds for the pairwise classifiers. Additionally, linear and square-root scaling of the power spectral density were compared to log scaling. A single combination of tuned parameters will remain constant for all pairwise classifiers.

Once the parameters were tuned, the classification accuracy was estimated. First, the vibration data was randomly divided into training data and test data sets. For each of the three terrains, ten data sets were randomly chosen as test data. This represents approximately 25% of the total data. The remaining data sets were chosen for training.

A three-terrain classifier was trained using the training data sets. Here, a 1-second segment length was used. After the classifier was trained, it was used to classify the test data sets. Classification results are presented in Table 3.1. Values shown are counts of 1-second-long vibration segments. The same results are plotted in Figure 3.10.

		Classification Result				
		Gravel	Mars-1	Sand	Unknown	Total
Actual Terrain	Gravel	302	2	0	8	312
	Mars-1	5	208	3	61	277
	Sand	0	51	139	86	276

Table 3.1. Classification results for FSRL Wheel-Terrain Interaction Testbed vibration data

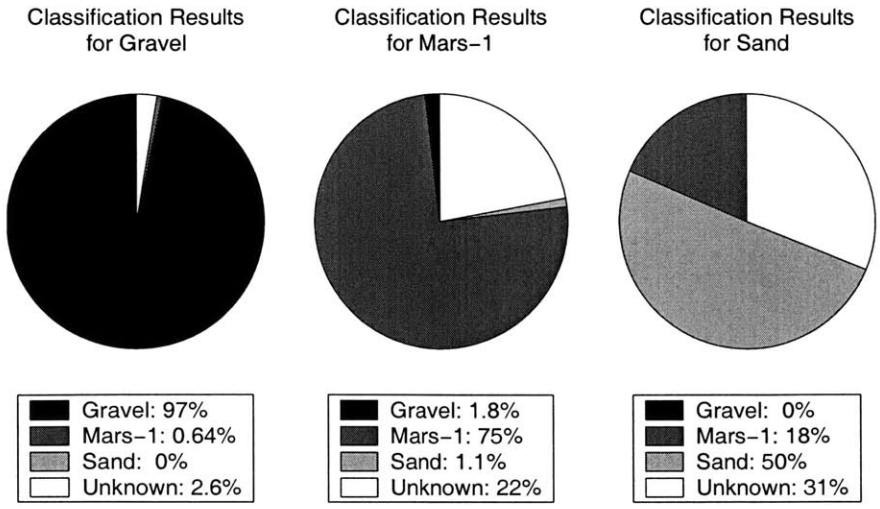


Figure 3.10. Classification results for FSRL Wheel-Terrain Interaction Testbed vibration data

These results show the algorithm's ability to distinguish between multiple terrain types. When attempting to identify gravel-induced vibrations, the algorithm misclassified less than 1% of the test data as Mars-1 or sand. Similarly, when classifying Mars-1 and sand vibration data, less than 1% was misclassified as gravel. This clearly demonstrates the ability of the algorithm to identify terrains which induce obviously distinct vibrations.

The more challenging distinction was between Mars-1 and sand. These two terrains are alike in the fact that they contain small particles which may damp out vibrations in the wheel. Despite this similarity, less than 2% of the Mars-1 vibration data was misidentified as being sand. The difficulty of this distinction reveals itself in the number of sand data sets being misidentified as Mars-1. Nevertheless, these misclassifications comprise less than 20% of the sand vibration data, while most of the data is correctly classified.

Considering the inverse problem—having confidence that the actual terrain matches the classification result—the algorithm performs quite well. Given equal prior likelihoods of the above three terrains, the algorithm is more than 98% confident that

terrain identified as gravel is actually gravel. Similarly, the algorithm is more than 97% confident that terrain identified as sand is truly sand. The confidence for Mars-1 is almost 80%.

It should be noted that these results are based solely on 1-second samples of vibration data, and incorporate no memory of prior classifications. An intelligent algorithm on a rover might incorporate an estimate of the likelihood of a transition from one terrain to another to improve overall classification results. Another way to improve terrain classification accuracy would be to combine the vibration-based classification with visual classification.

3.4.2 FSRL Technology Testbed Rover

In addition to being tuned and tested on the FSRL Wheel-Terrain Interaction Testbed, the algorithm was also verified using data collected on the FSRL Technology Testbed Rover to study the vibration response of a multi-wheeled rover in outdoor terrain. For these experiments, three terrains were used: gravel, concrete, and grassy soil. The gravel used was the same as was used in the FSRL Wheel-Terrain Interaction Testbed data. The concrete was smooth and flat, and could be expected to induce vibration signals similar to solid rock. The grassy soil was a thick cover of grass over packed topsoil. Figure 3.11 shows the rover during a traverse across all three terrains. For scale, the rover is 80 cm long.

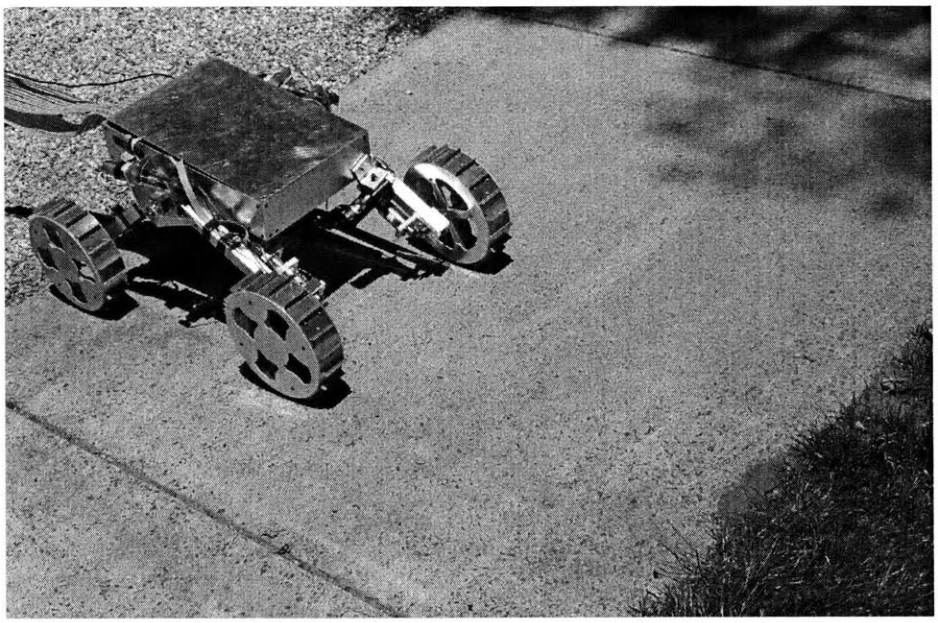


Figure 3.11. FSRL Technology Testbed Rover completing three-terrain traverse

Vibration signals were sensed using a contact microphone mounted to the front right leg of the rover, near the joint with the wheel axle, as seen in Figure 3.12. These signals were collected using a laptop with a sound card. Sixteen-bit samples were collected at a frequency of 44.1 kHz.

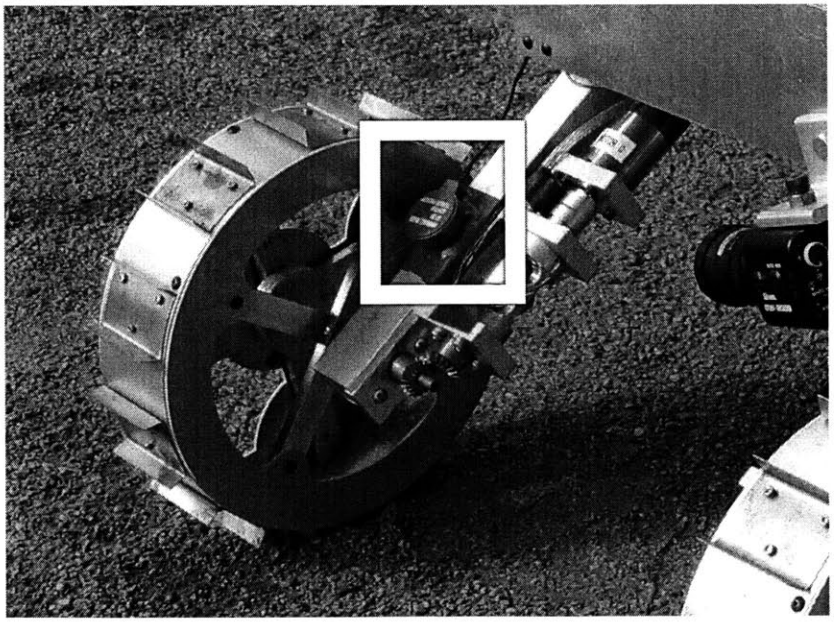


Figure 3.12. Vibration sensor mounted on FSRL Technology Testbed Rover

A single data set consisted of vibration data from a 60-second traverse of the rover across the terrain at a constant velocity. Velocity was varied from 1 cm/sec to 7 cm/sec. Data from both forward and reverse driving directions were collected. Six training data sets were collected from each terrain. Once the training data sets were collected, multi-terrain data sets were collected with the rover driving from gravel to concrete to grassy soil in a single traverse, in forward and reverse. The six training data sets from each terrain were used to train a three-terrain classifier. Due to the increased spacing between the grousers on the rover wheels relative to the wheel on the testbed, the segment length was increased to 3 seconds.

Table 3.2 shows the results for the classification of two test data sets for each terrain. The values are counts of the 3-second-long vibration segments. The same results are plotted in Figure 3.13.

		Classification Result				
		Gravel	Concrete	Grass	Unknown	Total
Actual Terrain	Gravel	37	0	0	2	39
	Concrete	10	27	0	18	55
	Grass	0	0	62	2	64

Table 3.2. Classification results for FSRL Technology Testbed Rover vibration data

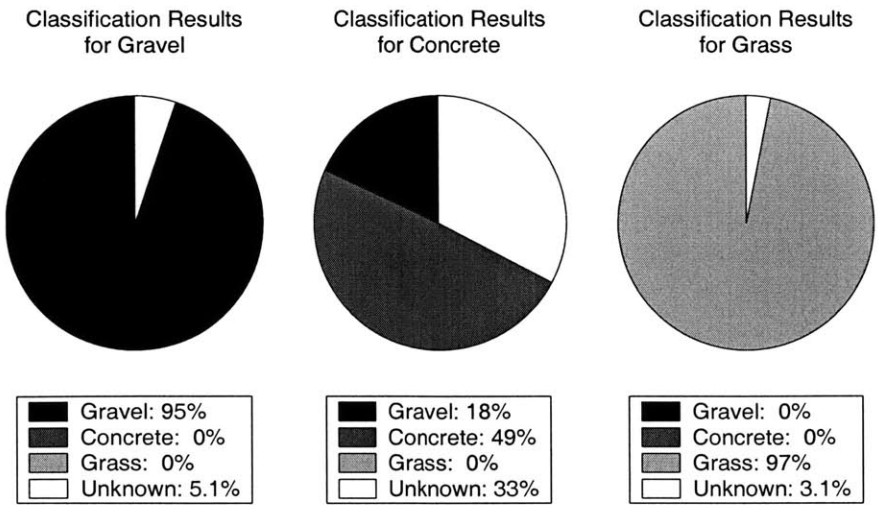


Figure 3.13. Classification results for FSRL Technology Testbed Rover vibration data

These results show the classification accuracy of the algorithm using vibration data collected on the FSRL Technology Testbed Rover. As with the data from the wheel-terrain testbed, the algorithm demonstrates excellent capability in distinguishing terrains which induce qualitatively different vibrations. Here, none of the data collected on grassy soil was misidentified as coming from gravel or concrete, and vice versa.

The most similar terrain types in these data sets were gravel and concrete. Both are hard surfaces unlikely to damp vibrations. It is no surprise that distinguishing between the two terrain types would be challenging. In the results presented above, however, none of the gravel vibration data sets were misclassified as concrete. That a larger number of the concrete data sets were misidentified as gravel—about 18%—illustrates the difficulty.

If the data is viewed as a measure of confidence in the classification result accurately representing the actual terrain, the algorithm continues to perform well. Given equal prior probabilities of each terrain type, there is an 84% confidence level that terrain labeled as gravel is truly gravel. At least in the test data, no classification of vibration data as concrete or grass was wrong, so the estimated confidence level is 100%.

Figure 3.14 and Figure 3.16 show the classification results for distinct single traverses of all three terrain types. These plots may be viewed as strip charts, plotting the result of the terrain classifier against the time the classifier made the identification. In Figure 3.14, it may be clearly observed that the rover is traveling from an area of gravel, to an area of concrete, and then onto an area of grassy soil. This path is illustrated in Figure 3.15. Figure 3.16 shows similar results for the opposite direction. In regions where the classifier returns no terrain class, the rover may not be entirely on one terrain or another. Even when the right front wheel is entirely on one terrain, vibrations from the other wheels may be transmitted through the rover structure and be picked up by the sensor. Thus, regions of ambiguity are to be expected between regions of consistent terrain. These results clearly show the effectiveness of the algorithm in classifying multiple terrains during a single traverse.

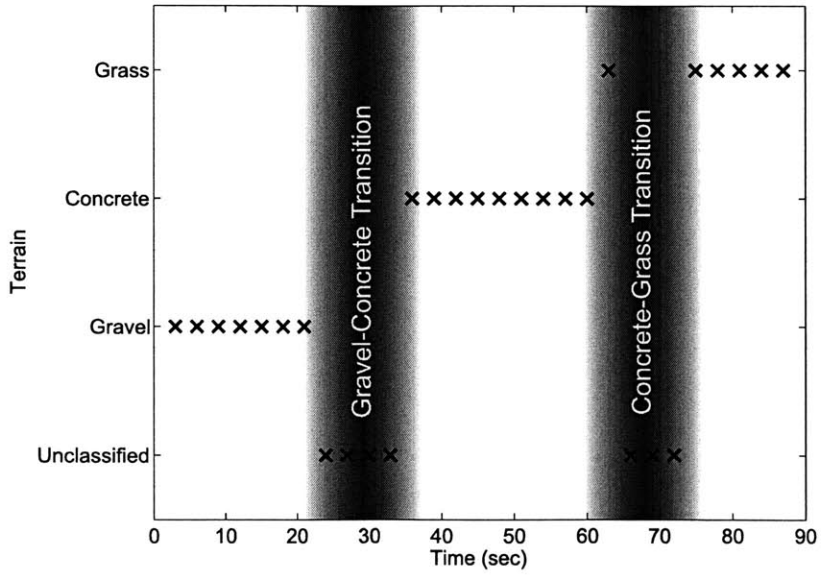


Figure 3.14. Classification results for FSRL Technology Testbed Rover traverse of gravel, concrete, and grassy soil at 4cm/s

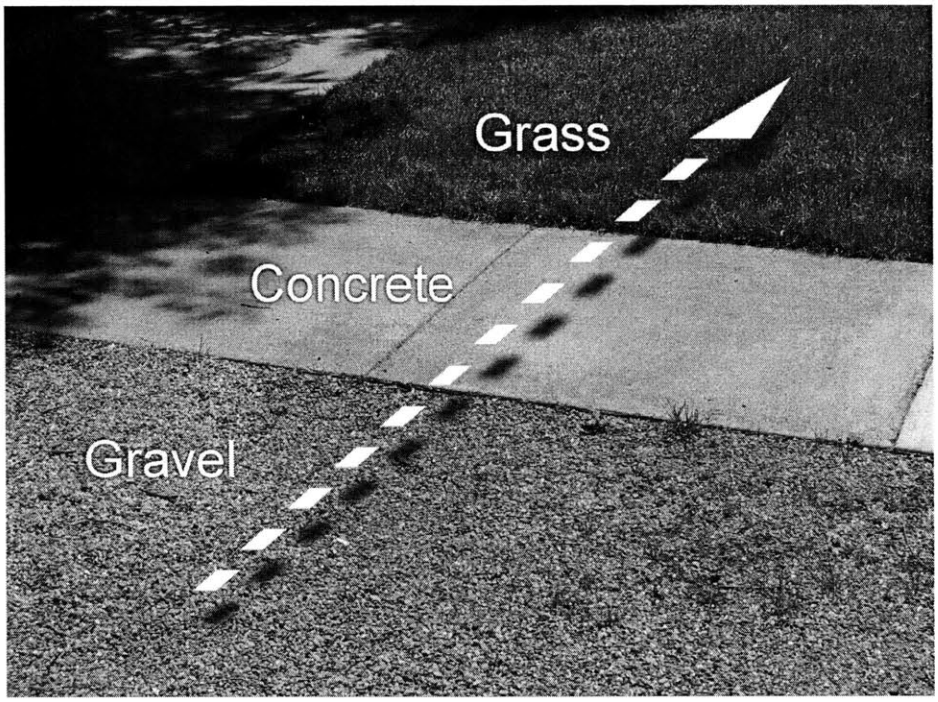


Figure 3.15. Path taken by rover in traverse of gravel, concrete, and grassy soil

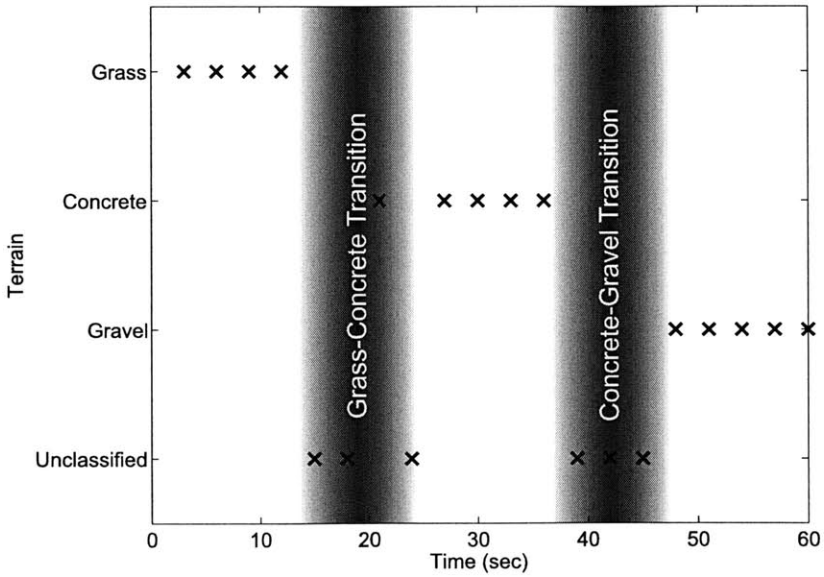


Figure 3.16. Classification results for FSRL Technology Testbed Rover traverse of gravel, concrete, and grassy soil at 6cm/s

3.5 Comparison to FSU Terrain Classification Algorithm

Researchers at Florida State University (FSU) recently presented an algorithm to classify terrain based on vehicle vibrations (Sadhukhan & Moore, 2003; Sadhukhan, 2004). Their work focuses on terrain classification for a high-speed vehicle with pneumatic tires for use on unmanned ground vehicles for the military. They use a frequency-domain signal analysis approach to classify terrain, similar to the algorithm presented in the preceding sections. The two algorithms will be compared below. For convenience, the approach presented as the PNN classifier in (Sadhukhan & Moore, 2003) and (Sadhukhan, 2004) will be referred to as the FSU approach. The approach detailed in Section 3.3 will be referred to as the MIT approach.

A brief summary of the FSU approach is as follows: vibration signals are gathered with a rover driving over various types of terrain, using the vertical acceleration data from an IMU on the rover as the signal. These vibration signals are segmented and then processed with a Fast Fourier Transform (FFT) to get frequency-domain vibration signatures. As with the MIT algorithm, the FSU algorithm compares new vibration signatures to those observed during a training phase. Their algorithm uses a Probabilistic Neural Network (PNN) approach (Specht, 1988) to estimate the posterior probability of a vibration belonging to a class. A Bayesian classifier is used to assign classes based on the prior and posterior probabilities.

Table 3.3 summarizes some of the differences between the FSU and MIT algorithms. Because the FSU algorithm was designed to use the vertical acceleration data from an IMU, the sampling frequency for the vibrations is only 100 Hz, compared to the 44.1 kHz used by the MIT algorithm. This higher sampling rate allows the segment

length of the MIT algorithm to be much shorter, allowing for much quicker classification of the terrain. When the difference in vehicle speeds is taken into account, the MIT algorithm is able to assign terrain classes to terrain patches as short as 0.5 cm, enough to allow wheel-terrain contact length to be the limiting factor. In contrast, the FSU algorithm assigns classes to terrain patches 2 m or longer.

	FSU Algorithm	MIT Algorithm
Sampling Freqency	100 Hz	44.1 kHz
Segment Length	10 sec	1 sec / 3 sec
Vehicle Speed	20-80 cm/sec	0.5-7 cm/sec
Frequency Representation	FFT	Log-scaled Welch PSD
Classification	Probabilistic Neural Network	Linear Discriminant Analysis

Table 3.3. Summary of differences between the Florida State University (FSU) and MIT classification algorithms

Another effect of the difference in sampling rate and vehicle speed is the characteristic length of the highest observable frequency. For the FSU algorithm, undulations in the ground with a period of less than 0.4 cm will be aliased or filtered out. For the MIT algorithm this period is more than 10000 times smaller, enough that the limiting factor will be the sensitivity of the vibration sensor. This difference in scales is appropriate due to the difference in purposes. The MIT algorithm is designed for a planetary rover with rigid wheels and grousers, where high-frequency vibrations are likely to be transmitted from the wheel-terrain interface to the vibration sensor. The FSU algorithm, designed for an unmanned ground vehicle with pneumatic tires, is unlikely to have high frequency vibrations get from the wheel-terrain interface to the IMU. Additionally, an IMU positioned near the center of the vehicle body is unlikely to be able to distinguish between vibration signals coming from each of the four wheels. For this reason, changes in terrain are meaningless on a scale shorter than a single vehicle length.

The difference in vehicle speeds also affects the magnitudes of the vibration signals being sensed. The amplitude of the acceleration increases as the square of the vehicle speed, so the vibration signals classified by the MIT algorithm are expected to be at least an order of magnitude smaller than those classified by the FSU algorithm.

Internally, the algorithms are similar in the fact that they assign terrain classes based on a frequency-domain representation of the signal, but that is the end of the similarity. The FSU algorithm uses a single FFT to represent the signal, and does the classification based only on frequencies in the 10-20 Hz range. The MIT algorithm uses a Welch estimate of the power spectral density, effectively averaging the squared magnitudes from 20 or more FFTs to reduce the effect of sensor noise. Additionally, the vibration signature is the logarithm-scaled PSD, because vibration magnitudes change so drastically in the range from 0 to 12 kHz.

The classification approaches also differ substantially. The FSU algorithm uses a probabilistic neural network to estimate posterior probabilities for a vibration to belong to a class. To accomplish this without spending significant amounts of time training, all of the training data is stored in FFT form. The posterior probability is estimated on-line as the sum of weighted Gaussian distributions centered at each training sample.

Because classification involves comparing each new vibration with every vibration in the training data set, the on-line computational requirements and storage become prohibitive when the training data set becomes large. For this reason, the FSU algorithm was trained using only two vibration segments per terrain per speed, for a total of 8 vibration signatures per terrain or 80 seconds worth of data per terrain. In contrast, the MIT algorithm was trained using significantly more training data: 300 seconds worth

of data per terrain for the FSRL Technology Testbed Rover experiments, or 100 vibration signatures. The MIT algorithm requires a significant amount of time to process the training data. After processing, however, the MIT algorithm only requires space for one vibration signature for each pair of terrains. On-line computation is reduced to performing a dot product with each of these vibration signatures.

The FSU algorithm was tested using a mobile robot on the same scale as the FSRL Technology Testbed Rover, about 90 cm long. The rover was driven over a 3-terrain testbed, with 10-meter-long sections of sand, packed dirt, and gravel. Between twenty and forty 10-second vibration segments were collected for each terrain and speed. These vibrations were used to estimate the classification accuracy of the algorithm.

The classification accuracy of the MIT algorithm compared very favorably with that of the FSU algorithm. The FSU algorithm performed very well at high speeds, as can be seen in Figure 3.17. Here the classification error rate is plotted against the vehicle speed. The performance of the FSU algorithm deteriorated significantly as speeds decreased, going from 5% error at 80 cm/sec to 25% error at 20 cm/sec. In contrast, the MIT algorithm demonstrated an error rate of 7% and 6% on the FSRL Wheel Terrain Testbed and the FSRL Technology Testbed Rover, at speeds of less than 10 cm/sec.

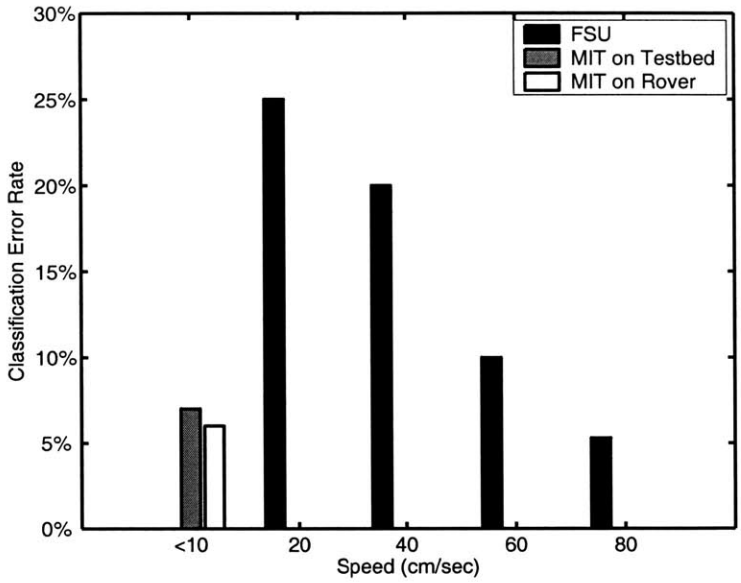


Figure 3.17. Classification error rates for FSU and MIT algorithms

Since the FSU algorithm lacks the capability to return "unknown" in response to vibrations which have similarities to multiple terrain classes, it may not be a fair comparison to exclude "unknown" results from the error rate. Figure 3.18 shows the classification error results with the MIT algorithm forced to return a class for the "unknown" situations. Even with this handicap, the MIT algorithm exceeds the performance the FSU algorithm achieves at 20 cm/sec and 40 cm/sec.

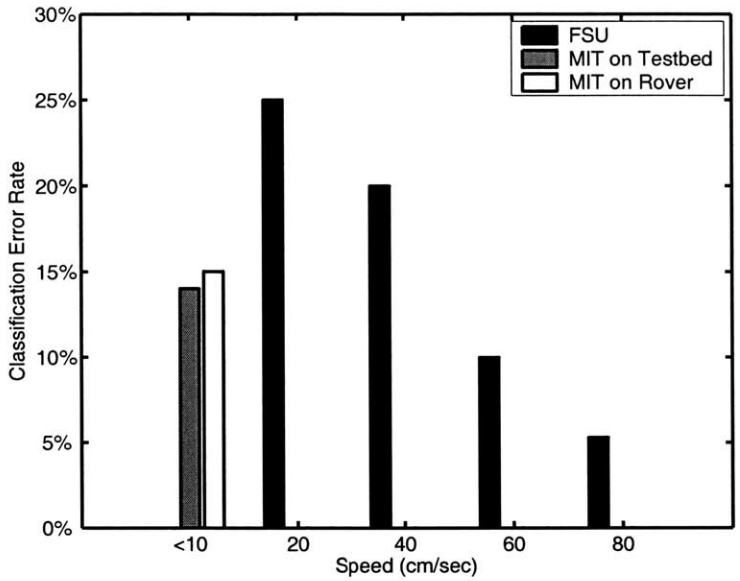


Figure 3.18. Classification error rates for FSU and MIT algorithms, forcing classification as one of known terrain classes

3.6 Summary and Conclusions

An algorithm capable of classifying terrain based on the vibrations in the rover structure induced by wheel-terrain interaction has been presented. This algorithm uses linear discriminant analysis to distinguish between each pair of terrain classes, and uses a voting algorithm to arbitrate between pairwise classifiers. The terrain class is returned if the algorithm can uniquely identify one; otherwise, an "unknown" result is returned.

Experimental results have been presented which show the classification accuracy using vibration data from two different testing setups. Results from the FSRL Wheel-Terrain Interaction Testbed showed the effectiveness of the algorithm in a closely-controlled laboratory environment. Results from the FSRL Technology Testbed Rover demonstrated the algorithm's capabilities in a more natural uncontrolled environment, on a more realistic rover. Classification of three distinct terrain types was demonstrated on each platform. Additionally, results from a multi-terrain data set from the FSRL

Technology Testbed Rover illustrated the use of this algorithm in identifying multiple terrain classes during a rover traverse.

This data clearly shows the potential for vibration-based classification of terrain as an addition to current vision-based terrain classification approaches. Using only a low-cost vibration sensor and a sound card that comes standard on most computers, vibration-based terrain classification presents an inexpensive way to gain information about the local terrain. The presented algorithm is an effective method for extracting terrain class information from the vibration data. It may be used as a component of a meta-classifier, combining data from multiple sensors along with a memory of past classification results, or a similar approach may be used to define terrain class probabilities in a Bayesian classifier. However it is used, vibration-based terrain classification shows great promise in improving rover understanding of local terrain.

Chapter 4

Conclusions and Suggestions for Future Work

4.1 Contributions of this Thesis

This thesis has presented two algorithms designed to analyze the interaction of a planetary rover with its terrain to thereby derive knowledge about the local terrain characteristics.

The first algorithm is an efficient method to measure sinkage of a rigid wheel in deformable terrain, using a single image containing the wheel-terrain interface. This algorithm has the potential to improve rover safety by detecting the possibility of a wheel becoming so submerged that the rover is trapped. Information from this algorithm can also be employed as an input to other algorithms, to assess the soil's traversability or physical properties.

The second algorithm identifies gross terrain classes based on the vibrations induced as the rover traverses the terrain. By providing a rapid classification of the local terrain, this algorithm has the potential to improve rover wheel-odometry position estimates by identifying terrain which is likely to induce high slip. It can also provide information to a path-planning algorithm to improve prediction of how terrain might affect slope traversability. Additionally, the ability of this algorithm to detect subsurface

changes in the terrain may lead to the identification of scientifically interesting sites, which otherwise the rover would miss altogether.

Together, these algorithms work to improve a rover's understanding of the local terrain. The information gained will allow rovers to operate in more treacherous conditions while avoiding threats to their safety and mobility which could jeopardize the success of a mission. The improved confidence in the ability of a rover to maintain safety in challenging conditions could allow mission controllers to send rovers to more scientifically interesting sites such as ridges or ravines. Improved rover positioning accuracy will enable scientists on Earth to collect data in less time, leading to increased data return for the same duration mission.

4.2 Suggestions for Future Work

Future work related to the vibration-based terrain classification algorithm could include studying approaches to modeling terrain class distributions in the frequency domain to further improve classification accuracy. A rigorous Bayesian approach providing class-conditional likelihoods would be very useful in combining vibration data with that of other sensors. For implementation on a rover, an intelligent terrain class estimator should be created which considers not only the most recent vibration data but also previous classification results, basing its current state estimate on the likelihood of a transition from one terrain to another.

Another promising avenue for future work would be to combine vision-based terrain perception algorithms with the methods presented in this thesis. Prediction of terrain characteristics within the field of view of the rover would be based on

characteristics of terrain the rover has driven over in the past. This could produce a hazard map for use in path-planning algorithms. It would also give scientists another way to view the terrain surrounding a rover, allowing them to recognize and study scientifically-relevant sites.

References

Allen, C., Morris, R., Jager, K., Golden, D., Lindstrom, D., Lindstrom, M., & Lockwood, J. (1998). Martian Regolith Simulant JSC Mars-1. *Proceedings of the 29th Lunar and Planetary Science Conference,* Houston, TX, March 16-20, 1998.

Balakrishnama, S., Ganapathiraju, A. (1998). *Linear Discriminant Analysis—A Brief Tutorial.* Retrieved July 8, 2004, from Mississippi State University, Department of Electrical and Computer Engineering Web site: http://www.isip.msstate.edu/publications/reports/isip_internal/1998/linear_discrim_analysis/lda_theory.pdf

Bekker, M. G. (1956). *Theory of Land Locomotion.* Ann Arbor: University of Michigan Press.

Bellutta, P., Manduchi, R., Matthies, L., Owens, K., & Rankin, A. (2000). Terrain Perception for DEMO III. *Proceedings of the IEEE Intelligent Vehicles Symposium*, Dearborn, MI, October 3-5, 2000, 326-332.

Castaño, R., Manduchi, R., & Fox, J. (2001, December). *Classification Experiments on Real-World Texture.* Paper presented at the Workshop on Empirical Evaluation in Computer Vision, Kauai, HI.

DeSouza, G. N., & Kak, A. C. (2002). Vision for Mobile Robot Navigation: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(2),* 237-267.

Espinal, F., Huntsberger, T. L., Jawerth, B., & Kubota, T. (1998). Wavelet-Based Fractal Signature Analysis for Automatic Target Recognition. *Optical Engineering, Special Section on Advances in Pattern Recognition, 37(1),* 166-174.

Forsyth, D. A., & Ponce, J. (2003). *Computer Vision: A Modern Approach.* Upper Saddle River, NJ: Prentice Hall.

Gerhard, D. (2000). Audio Signal Classification: An Overview. In A. Grbavec (Ed.), *Canadian Artificial Intelligence, 45*, Canadian Society for Computational Studies of Intelligence. 4-6.

Golub, G. H., & Van Loan, C. F. (1996). The Singular Value Decomposition and Unitary Matrices. In *Matrix Computations* (3rd ed.) (pp. 70-71 and 73). Baltimore, MD: Johns Hopkins University Press.

Iagnemma, K. D. (2001). *Rough-Terrain Mobile Robot Planning and Control with Application to Planetary Exploration.* Unpublished doctoral thesis, Massachusetts Institute of Technology, Cambridge, MA.

Iagnemma, K., Kang, S., Brooks, C., & Dubowsky, S. (2003). Multi-Sensor Terrain Estimation for Planetary Rovers. *Proceedings of the Seventh International Symposium on Artificial Intelligence, Robotics, and Automation in Space, i-SAIRAS,* Nara, Japan, May 2003.

Iagnemma, K., Shibly, H., & Dubowsky, S. (2002). On-Line Terrain Parameter Estimation for Planetary Rovers. *Proceedings of the 2002 IEEE International Conference on Robotics & Automation*, Washington, DC, May 2002. 3142-3147.

Jaakolla, T., & Haussler, D. (1999). Probabilistic kernel regression models. In D. Heckerman & J. Whittaker (Eds.), *Proceedings of the Seventh International*

Workshop on Artificial Intelligence and Statistics, Fort Lauderdale, FL, January 3-6, 1999. San Francisco: Morgan Kaufmann Publishers.

Jolliffe, I. T. (1986). *Principal component analysis.* New York: Springer-Verlag.

Kang, S. (2003). *Terrain parameter estimation and traversability assessment for mobile robots.* Unpublished master's thesis, Massachusetts Institute of Technology, Cambridge, MA.

Lu, L., Jiang, H., & Zhang, H.-J. (2001). A Robust Audio Classification and Segmentation Method. *Proceedings of the 9th ACM International Conference on Multimedia*, Ottawa, Canada, September 30-October 5, 2001. 203-211.

Mahalanobis, P. C. (1936). On the Generalized Distance in Statistics. *Proceedings of the National Institute of Sciences of India, 2(1),* 49-55.

Maimone, M. (2002). Cahvor Camera Model Information. Retrieved July 8, 2004 from the World Wide Web: http://telerobotics.jpl.nasa.gov/people/mwm/cahvor.html

Malik, J., & Perona, P. (1990). Preattentive texture discrimination with early vision mechanisms. *Journal of the Optical Society of America. A, 7(5),* 923-932.

Manduchi, R. (2000). Mixture Models and the Segmentation of Multimodal Textures. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hilton Head Island, SC, June 13-15, 2000. 98-104.

Moore, H. J., Bickler, D. B., Crisp, J. A., Eisen, H. J., Gensler, J. A., Haldemann, A. F. C., Matijevic, J. R., Reid, L. K., & Pavlics, F. (1999). Soil-like deposits observed by Sojourner, the Pathfinder rover. *Journal of Geophysical Research, 104(E4),* 8729-8746.

Murray, R., Li, Z., & Sastry, S. (1994). *A Mathematical Introduction to Robotic Manipulation.* Boca Raton: CRC Press.

NASA Jet Propulsion Laboratory. (2004). Left Front Hazard Camera Non-linearized Sub-frame EDR acquired on Sol 135 of Spirit's mission to Gusev Crater at approximately 14:54:25 Mars local solar time. Retrieved June 28, 2004 from the Mars Exploration Rovers Web site: http://marsrovers.jpl.nasa.gov/gallery/all/2/f/135/2F138358521ESF5414R2502L0M1.JPG

NASA Jet Propulsion Laboratory. (2004, February 6). Healthy Spirit Cleans a Mars Rock; Opportunity Rolls. *NASA News Release 2004-053*. Retrieved July 6, 2004, from the World Wide Web: http://marsrovers.jpl.nasa.gov/newsroom/pressreleases/20040206a.html

NASA Jet Propulsion Laboratory. (2004, February 9). Mars Rover Pictures Raise 'Blueberry Muffin' Questions. *NASA News Release 2004-054*. Retrieved July 6, 2004, from the World Wide Web: http://marsrovers.jpl.nasa.gov/newsroom/pressreleases/20040209a.html

NASA Jet Propulsion Laboratory. (2004, June 8). Mars Rovers Continue Unique Exploration of Mars. *NASA News Release 2004-144*. Retrieved July 6, 2004, from the World Wide Web: http://marsrovers.jpl.nasa.gov/newsroom/pressreleases/20040608a.html

Rasmussen, C. (2002). Combining Laser Range, Color, and Texture Cues for Autonomous Road Following. *Proceedings of the 2002 IEEE Conference on Robotics & Automation*, Washington, DC, May 11-15, 2002. 4320-4325.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323,* 533-536.

Sadhukhan, D. (2004). *Autonomous Ground Vehicle Terrain Classification Using Internal Sensors.* Unpublished master's thesis, Florida State University, Tallahassee. Retrieved July 8, 2004, from the World Wide Web: http://etd.lib.fsu.edu/theses/available/etd-04092004-171647/

Sadhukhan, D. & Moore, C. (2003). Online Terrain Estimation Using Internal Sensors. *Proceedings of Florida Conference on Recent Advances in Robotics (FCRAR),* Boca Raton, FL, May 2003.

Schenker, P.S. et al. (2001). FIDO: a Field Integrated Design & Operations Rover for Mars Surface Exploration. *Proceedings of the Sixth International Symposium on Artificial Intelligence, Robotics, and Automation in Space,* Montreal, Canada, June 2001. Retrieved July 8, 2004, from the World Wide Web: http://citeseer.ist.psu.edu/schenker01fido.html

Specht, D. (1988). Probabilistic Neural Networks for Classification, Mapping, or Associative Memory. *IEEE International Conference on Neural Networks, 1988,* San Diego, CA, July 24-28, 1998. 1:525-532.

Volpe, R. (2003). Rover Functional Autonomy Development for the Mars Mobile Science Laboratory. *Proceedings of the 2003 IEEE Aerospace Conference,* Big Sky, MT, March 8-15, 2003. 2:643-652.

Voyles, R. M., Larson, A. C., Yesin, K. B., & Nelson, B. (2001). Using Orthogonal Visual Servoing Errors for Classifying Terrain. *Proceedings of the 2001*

IEEE/RSJ International Conference on Intelligent Robots and Systems, Maui, HI, October 29-November 3, 2001. 772-777.

Welch, P. D. (1967). The Use of Fast Fourier Transform for the Estimation of Power Spectra: A Method Based on Time Averaging Over Short, Modified Periodograms. *IEEE Transactions on Audio and Electroacoustics, AU-15,* 70-73.

Wilcox, B. H. (1994). Non-Geometric Hazard Detection for a Mars Microrover. *Proceedings of the AIAA Conference on Intelligent Robotics in Field, Factory, Service, and Space, 2.* Houston, TX, March 21-24, 1994. 675-684.

Wu, H., Siegel, M., & Khosla, P. (1999). Vehicle Sound Signature Recognition by Frequency Vector Principal Component Analysis. *IEEE Transactions on Instrumentation and Measurement, 48(5),* 1005-1009.

Yakimovsky, Y., & Cunningham, R. (1978). A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras. *Computer Graphics and Image Processing 7,* 195-210.

Appendix A

Matrix-Based Optics Using Pinhole Camera Model

The pinhole camera model is an idealized camera model in which all rays of light are assumed to travel in straight lines and pass through the focus point on their way from an object to the sensor on the camera (Forsyth & Ponce, 2003). Figure A.1 shows an illustration of such a system.

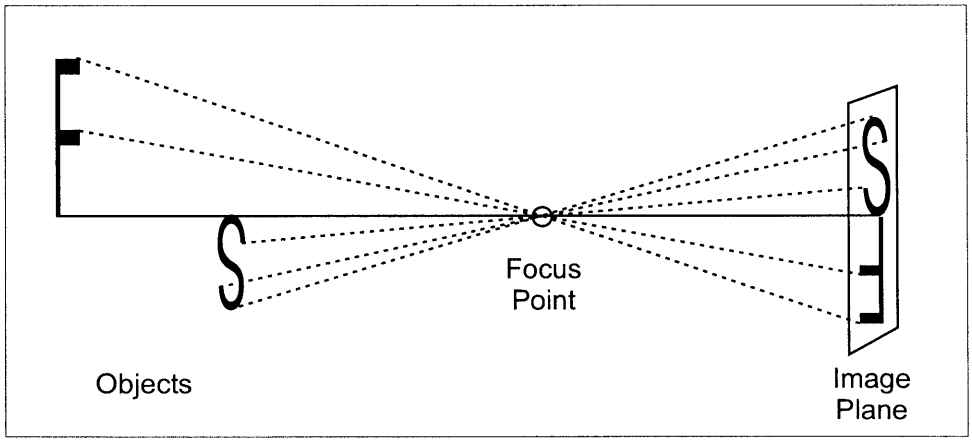


Figure A.1. Projection of objects onto an image plane with a pinhole camera model

The pinhole camera model is one of the simplest models of a camera. It has the advantage of having a small number of parameters necessary to fully define the model, while still modeling the effect of perspective.

Matrix Notation

The matrix-based notation in this thesis is taken from the homogeneous representation of points, vectors, and transformations in (Murray, Li, & Sastry, 1994). In this notation, a fourth element is added to each point and vector, so that any affine transformation can be represented by a single matrix multiplication.

Points

A point p located at (p_x, p_y, p_z) is represented as

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}. \qquad (A.1)$$

Thus, every point has a fourth element equal to 1.

Vectors

Vectors, the difference between two points, have a fourth element equal to 0. Thus, a vector v, normally written as (v_x, v_y, v_z), is represented as

$$\mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}. \qquad (A.2)$$

Transformations

An affine transformation using this notation is a 4×4 matrix, with the last row equal to $[0 \quad 0 \quad 0 \quad 1]$. For instance, translating a point p d_x units along the x axis, d_y units

along the y axis, and d_z units along the z axis would be accomplished by pre-multiplying $\mathbf{p}$ by this matrix:

$$\begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (A.3)$$

A rotation by 45° around the y axis would be accomplished by pre-multiplying $\mathbf{p}$ by this matrix:

$$\begin{bmatrix} \cos(45°) & 0 & \sin(45°) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(45°) & 0 & \cos(45°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (A.4)$$

This notation makes it very easy to convert representations of points from one reference frame to another. Consider two reference frames: "camera" and "wheel." The origin of the "wheel" reference frame can be written as a point in the camera frame: $\mathbf{p}_{\text{camera}}$. (Here the subscript "camera" means that the point is represented in terms of the camera frame.) Similarly, the unit vectors along the x, y, and z axes of the wheel frame can be written as vectors in the camera frame: $\hat{\mathbf{x}}_{\text{camera}}$, $\hat{\mathbf{y}}_{\text{camera}}$, and $\hat{\mathbf{z}}_{\text{camera}}$. The transformation from the wheel frame to the camera frame can be written as

$$\mathbf{T}_{\text{camera}}^{\text{wheel}} = \begin{bmatrix} \hat{\mathbf{x}}_{\text{camera}} & \hat{\mathbf{y}}_{\text{camera}} & \hat{\mathbf{z}}_{\text{camera}} & \mathbf{p}_{\text{camera}} \end{bmatrix}. \qquad (A.5)$$

Using this transformation, a point q, represented in the wheel reference frame by $\mathbf{q}_{\text{wheel}}$, can be written in terms of the camera reference frame as

$$\mathbf{q}_{\text{camera}} = \mathbf{T}_{\text{camera}}^{\text{wheel}} \mathbf{q}_{\text{wheel}} . \qquad (A.6)$$

Projections of Points

In the pinhole camera model, each ray of light passes from an object through the focus point and onto the plane of the sensor, typically a CCD. This is not an affine transformation, so it cannot be accomplished through a simple matrix multiplication. However affine transformations can be used to write a point in terms of a reference frame in which calculating the projection is easy.

If we define the camera frame such that its origin is at the focus point, and its z-axis is perpendicular to the plane of the sensor, F, as shown in Figure A.2, the projection of any point becomes the simple matter of scaling it by its z-component. In other words, for

$$\mathbf{p}_{\text{camera}} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \qquad (A.7)$$

the projection of $\mathbf{p}_{\text{camera}}$ onto the plane F as $\mathbf{q}_{\text{camera}}$ is given by

$$\mathbf{q}_{\text{camera}} = \text{proj}_F(\mathbf{p}_{\text{camera}}) = \begin{bmatrix} \left(\frac{f_z}{p_z}\right) p_x \\ \left(\frac{f_z}{p_z}\right) p_y \\ f_z \\ 1 \end{bmatrix}, \qquad (A.8)$$

where f_z is the z value defining the location of F.

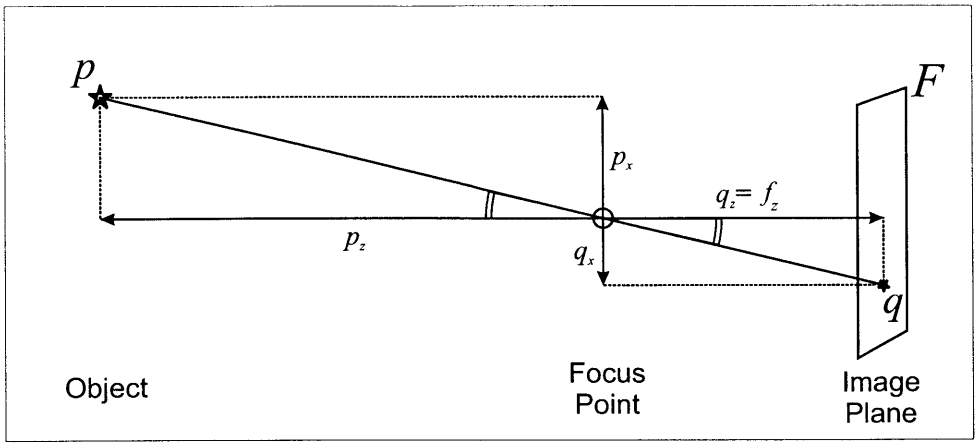


Figure A.2. Projection of a point onto the image plane, with coordinates

Projection of points is also useful for the inverse problem of identifying which point in space corresponds to a given pixel, if it is known which plane in space that point lies on. The procedure for this projection is the same; a frame is defined with its origin as the focus point and its z axis perpendicular to the plane. After a pixel has been written in terms of that reference frame, the projection is done as in Equation (A.8), using the z-coordinate of the plane as f_z.

Projections of Circles

In order to determine whether the projection of a point lies within a circle in space (for instance, to find out whether a pixel corresponds to a point within the rim of a wheel), it is advantageous to define a circle in terms of its projection through a pinhole onto a plane. A straightforward method to accomplish this is to define a cone with its vertex at the focus point, and a circular base coincident with the desired circle, as illustrated in Figure A.3.

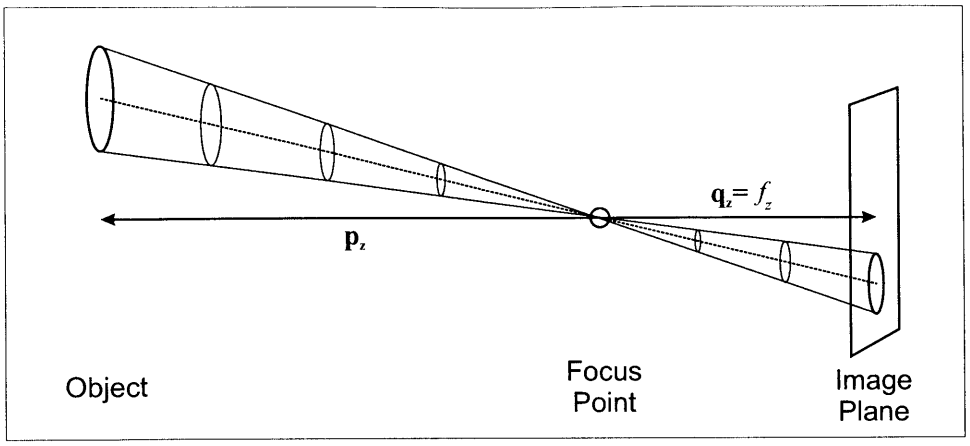


Figure A.3. Projection of a circle using a cone with vertex at the focus point

To review some basic equations, a circle is defined in two dimensions as the set of all points (x, y) such that $x^2 + y^2 - r^2 = 0$. In matrix notation, this becomes

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -r^2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0. \qquad (A.9)$$

Adding a third dimension gives the equation of a right cone, with its vertex at (0,0,0) and radius r at $z = 1$:

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -r^2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0. \qquad (A.10)$$

Thus, a 4×4 symmetric matrix $\mathbf{C}$ can be said to define a cone, such that $\mathbf{p}^T \mathbf{C}\, \mathbf{p} = 0$ for any point p on the cone. Scaling r and switching reference frames yields a general definition for a right cone, however this still requires the focus point (i.e. the vertex of the cone) to lie along the axis of the circle.

To extend this definition to encompass a general cone, it is necessary to allow for a shear transformation. Consider a 4×4 matrix $\mathbf{S}$ defined as

$$\mathbf{S}=\begin{bmatrix}1 & 0 & c_x & 0\\ 0 & 1 & c_y & 0\\ 0 & 0 & c_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}. \qquad (A.11)$$

This transformation scales and shears the z axis such that (0,0,1) becomes (c_x, c_y, c_z), while keeping the origin the same and leaving the x and y axes unchanged. The inverse of $\mathbf{S}$ does the reverse.

Combining the above, consider a reference frame with origin at the camera's focus point and its z axis normal to the plane of the circle. Call this frame "TW." If a point $\mathbf{p}_{TW}$ defined in this frame lies on the cone $\mathbf{C}$ sheared by $\mathbf{S}$, the following equation must be satisfied:

$$\left(\mathbf{S}^{-1}\mathbf{p}_{TW}\right)^{T}\mathbf{C}\left(\mathbf{S}^{-1}\mathbf{p}_{TW}\right)=0. \qquad (A.12)$$

Alternately, a matrix $\mathbf{C}_{TW}$ can be defined in the TW frame,

$$\mathbf{C}_{TW}=\left(\mathbf{S}^{-1}\right)^{T}\mathbf{C}\,\mathbf{S}^{-1}, \qquad (A.13)$$

such that the defining equation is

$$\mathbf{p}_{TW}{}^{T}\mathbf{C}_{TW}\mathbf{p}_{TW}=0. \qquad (A.14)$$

This is the equation defining a cone with a vertex at (0,0,0), and a circular cross-section or radius r in the x-y plane centered around the point (c_x, c_y, c_z). This equation can be used to identify whether any point in the TW frame lies on, within, or outside the cone.

Scaling to Pixel Coordinates

The above projections relate the position in space of an object being viewed with the position in space of the sensor element recording the image. Since CCDs are regularly spaced arrays of pixels, a matrix multiplication can be used to relate discrete pixel (x, y) coordinates to spatial coordinates (p_x, p_y, p_z). This is most easily done in the camera frame:

$$\mathbf{p}_{\text{camera}} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} scaleX & 0 & f_x \\ 0 & scaleY & f_y \\ 0 & 0 & f_z \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \qquad (A.15)$$

Here, the camera x and y axes are assumed to be aligned with the horizontal and vertical axes, respectively, of the sensor array. In the transformation matrix in Equation (A.15), (f_x, f_y, f_z) is the position of the pixel (0,0) in the camera reference frame. The magnitude of *scaleX* is the distance between the centers of adjacent pixels in the horizontal direction; the magnitude of *scaleY* is the distance between the centers of adjacent pixels in the vertical direction. Due to the mirroring effect of projection through a point, *scaleX* is typically negative. The image is mirrored across the horizontal axis as well, however convention calls for the pixel y value to increase from top to bottom, so *scaleY* is typically positive. This 4×3 transformation matrix will be referred to as $\mathbf{T}_{\text{camera}}^{\text{pixel}}$.

Conversely, a point on the plane of the sensor array (i.e. $p_z = f_z$) can be related to its (x, y) pixel coordinates with the following transform:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{scaleX} & 0 & 0 & -\frac{f_x}{scaleX} \\ 0 & \frac{1}{scaleY} & 0 & -\frac{f_y}{scaleY} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}. \qquad (A.16)$$

This 3×4 transformation matrix will be referred to as $\mathbf{T}_{\text{pixel}}^{\text{camera}}$.

Combining the above, the pixel coordinates (x, y) of a point q (represented by $\mathbf{q}_{\text{wheel}}$ in the wheel frame) are given as:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{T}_{\text{pixel}}^{\text{camera}} \, \text{proj}_F \left(\mathbf{T}_{\text{camera}}^{\text{wheel}} \mathbf{q}_{\text{wheel}} \right). \qquad (A.17)$$

The defining equation of the ellipse formed by projecting a circle through a point onto the sensor plane is:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \left(\mathbf{T}_{\text{camera}}^{\text{pixel}} \right)^{\text{T}} \left(\mathbf{T}_{\text{TW}}^{\text{camera}} \right)^{\text{T}} \mathbf{C}_{\text{TW}} \mathbf{T}_{\text{TW}}^{\text{camera}} \mathbf{T}_{\text{camera}}^{\text{pixel}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0, \qquad (A.18)$$

assuming a transformation matrix $\mathbf{T}_{\text{TW}}^{\text{camera}}$ that converts a point in the camera reference frame to a point in the TW frame.

Relationship Between CAHV Parameters and Matrix Parameters

Another common pinhole-based model is referred to as the CAHV model (Yakimovsky & Cunningham, 1978), named for the four vectors parameterizing the model. The CAHV model uses 3×1 vector notation, preferring to use dot products rather than matrix multiplication. The four vectors are defined as follows:

- $\mathbf{C}$ is the position of the camera's focus point

- $\mathbf{A}$ is a unit vector in the direction normal to the sensor plane
- $\mathbf{H}$ is the sum of a vector in the direction of the sensor $+x$ direction and a vector in the same direction as $\mathbf{A}$
- $\mathbf{V}$ is the sum of a vector in the direction of the sensor $+y$ direction and a vector in the same direction as $\mathbf{A}$

In this model, the pixel (x, y) coordinates of a point $\mathbf{P}$ in space can be found using the following equations:

$$x = \frac{(\mathbf{P}-\mathbf{C})\cdot\mathbf{H}}{(\mathbf{P}-\mathbf{C})\cdot\mathbf{A}} \qquad (A.19)$$

$$y = \frac{(\mathbf{P}-\mathbf{C})\cdot\mathbf{V}}{(\mathbf{P}-\mathbf{C})\cdot\mathbf{A}}. \qquad (A.20)$$

Further information about the CAHV model, and its nonlinear extensions, CAHVOR and CAHVORE, can be found in (Maimone, 2002). Note that these nonlinear extensions to the CAHV model are used aboard JPL's planetary rovers.

Because the CAHV model uses the same geometric foundation of projection through a point onto a plane as the matrix-based model presented above, the CAHV parameters can be directly related to the parameters in the matrix-based model. Specifically, since the reference frame for the CAHV model may not be identical any of to those defined for the matrix-based model, it is useful to be able to convert between the coordinate systems.

The first step in relating the two systems is to define conversions of positions of points in space from the CAHV reference frame into the camera reference frame. (Note that the camera reference frame is defined above as having its origin at the camera focus

point and axes aligned with the sensor axes.) For this conversion, we first write basis vectors of the camera frame in terms of the CAHV reference frame. These vectors are $\hat{\mathbf{x}}_{\mathrm{CAHV}}^{\mathrm{camera}}$, $\hat{\mathbf{y}}_{\mathrm{CAHV}}^{\mathrm{camera}}$, and $\hat{\mathbf{z}}_{\mathrm{CAHV}}^{\mathrm{camera}}$, with the superscript indicating that they form the basis of the camera frame, and the subscript indicating that they are written in terms of the CAHV frame. By definition, $\hat{\mathbf{z}}_{\mathrm{CAHV}}^{\mathrm{camera}} = -\mathbf{A}$, as the camera axis is the negative z axis. To find $\hat{\mathbf{x}}_{\mathrm{CAHV}}^{\mathrm{camera}}$ and $\hat{\mathbf{y}}_{\mathrm{CAHV}}^{\mathrm{camera}}$, the component parallel to $\hat{\mathbf{z}}_{\mathrm{CAHV}}^{\mathrm{camera}}$ is removed, and the resulting vectors are normalized:

$$\hat{\mathbf{x}}_{\mathrm{CAHV}}^{\mathrm{camera}} = \frac{\mathbf{H} - (\mathbf{H}\cdot\mathbf{A})\mathbf{A}}{\left|\mathbf{H} - (\mathbf{H}\cdot\mathbf{A})\mathbf{A}\right|} \qquad (A.21)$$

$$\hat{\mathbf{y}}_{\mathrm{CAHV}}^{\mathrm{camera}} = -\frac{\mathbf{V} - (\mathbf{V}\cdot\mathbf{A})\mathbf{A}}{\left|\mathbf{V} - (\mathbf{V}\cdot\mathbf{A})\mathbf{A}\right|}. \qquad (A.22)$$

Using these values, a transformation matrix from the camera frame into the CAHV frame may be written as:

$$\mathbf{T}_{\mathrm{CAHV}}^{\mathrm{camera}} = \begin{bmatrix} \hat{\mathbf{x}}_{\mathrm{CAHV}}^{\mathrm{camera}} & \hat{\mathbf{y}}_{\mathrm{CAHV}}^{\mathrm{camera}} & \hat{\mathbf{z}}_{\mathrm{CAHV}}^{\mathrm{camera}} & \mathbf{C} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (A.23)$$

In this form a point in the camera frame (represented by the 4×1 vector $\mathbf{p}_{\mathrm{camera}}$) may be written as a 4×1 vector in the CAHV reference frame $\mathbf{p}_{\mathrm{CAHV}}$:

$$\mathbf{p}_{\mathrm{CAHV}} = \mathbf{T}_{\mathrm{CAHV}}^{\mathrm{camera}}\mathbf{p}_{\mathrm{camera}}. \qquad (A.24)$$

Projection into pixel coordinates may be done using Equations (A.19) and (A.20). Note that conversion of a point in the CAHV reference frame to the camera reference frame may be done using $\mathbf{T}_{\mathrm{camera}}^{\mathrm{CAHV}} = (\mathbf{T}_{\mathrm{CAHV}}^{\mathrm{camera}})^{-1}$.

The reverse problem, identifying the point in space where the sensor pixel is located, is not fully defined in the CAHV model. Therefore, an arbitrary value is chosen for f_z, the distance from the camera focus point to the sensor plane. Without going into the derivation, the transformation may be written thus:

$$\mathbf{p}_{\mathbf{CAHV}} = \left(\begin{bmatrix} f_z \begin{bmatrix} \mathbf{H}^{\mathbf{T}} \\ \mathbf{V}^{\mathbf{T}} \\ \mathbf{A}^{\mathbf{T}} \end{bmatrix}^{-1} \\ \hline 0 \quad 0 \quad 1 \end{bmatrix} + \left[\begin{array}{cc:c} 0 & 0 & \\ 0 & 0 & \mathbf{C} \\ 0 & 0 & \\ \hdashline 0 & 0 & 0 \end{array} \right] \right) \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \qquad (A.25)$$

The above presentation details a method which uses the CAHV reference frame as the frame in which projections are performed. An alternative method is to use the CAHV model for calibration, and then to extract the camera and lens parameters for use with the matrix-based model. These parameters, used in Equations (A.15) and (A.16) may be found as follows:

$$scaleX = \frac{f_z}{\mathbf{H} \cdot \hat{\mathbf{x}}_{\mathbf{CAHV}}^{\mathbf{camera}}} \qquad (A.26)$$

$$scaleY = \frac{f_z}{\mathbf{V} \cdot \hat{\mathbf{y}}_{\mathbf{CAHV}}^{\mathbf{camera}}} \qquad (A.27)$$

$$f_x = -scaleX\ (\mathbf{H} \cdot \mathbf{A}) \qquad (A.28)$$

$$f_y = -scaleY\ (\mathbf{V} \cdot \mathbf{A}). \qquad (A.29)$$

$\mathbf{T}_{\mathbf{pixel}}^{\mathbf{camera}}$ and $\mathbf{T}_{\mathbf{camera}}^{\mathbf{pixel}}$ may therefore be written without regard for the CAHV frame. By using $\mathbf{T}_{\mathbf{camera}}^{\mathbf{CAHV}}$, any point in the CAHV reference frame may be written in terms of the camera frame. Thus, if it is desired, all operations may be done without using the CAHV frame.

Appendix B

Principal Component Analysis and Singular Value Decomposition

Principal component analysis is a method for reducing the dimensionality of high-dimensional data (Jolliffe, 1986). Singular value decomposition is a mathematical tool for studying the dimensionality of a matrix, and it is used as part of a principal component analysis (Golub & Van Loan, 1996). This appendix will describe the concept of principal component analysis, give the basics of how it is implemented, and then give some mathematical details about singular value decomposition.

Principal Component Analysis

Principal component analysis is a method for analyzing high-dimensional data by reducing its dimensionality. Since in general there is no way to reduce the number of dimensions describing a set of data without losing information, the goal of principal component analysis is to reduce the number of dimensions while maintaining as much of the original information as possible. Here an example is presented that illustrates this fundamental use of principal component analysis.

Suppose an experiment returns two values, x and y, for each trial. A sample plot of the x and y values for 400 trials is shown in Figure B.1.

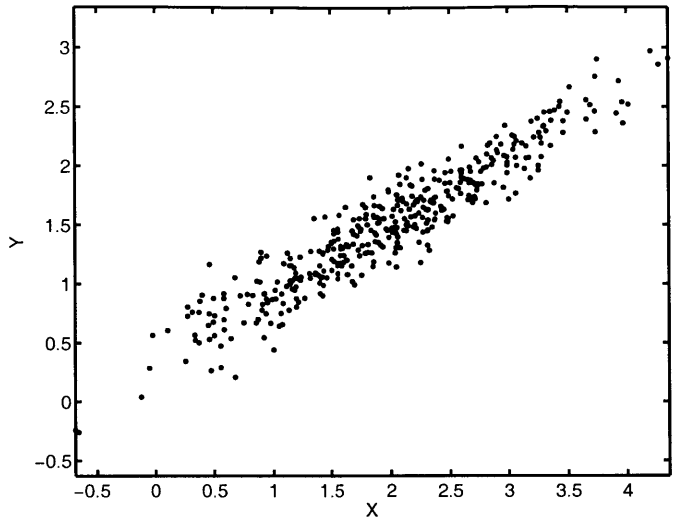


Figure B.1. Point cloud of (x, y) values

Now suppose that only one value can be stored for each of the trials. The dimensionality of the data must be reduced from two to one, so instead of being represented by a point anywhere on a plane, each trial must be represented by a point on a line. Choosing to store only the x value would mean denying that any useful information is stored in y, and vice versa. The obvious choice is to store a combination of x and y, equivalent to projecting the points on the line shown in Figure B.2.

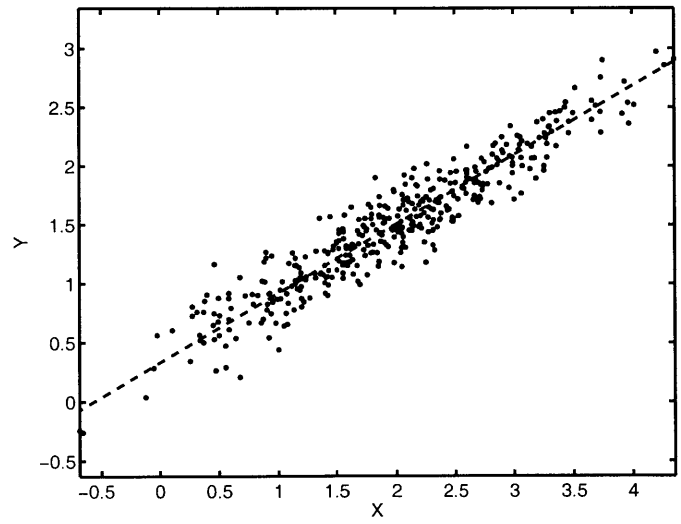


Figure B.2. Point cloud of (x, y) data with first principal component

Thus, knowing the slope and y-offset of the line and the single value for each trial, the x and y values for each of the trials can be estimated as in Figure B.3. This is the projection onto a line that loses the least information. The direction of the line (i.e. the vector with which the dot product is taken in the projection) is the first principal component of the data. In this case, the principal component is (0.86, 0.50), or 30° above the horizontal.

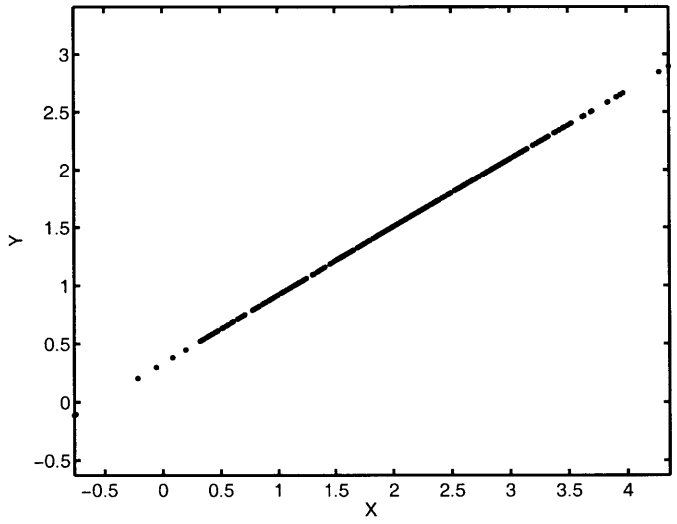


Figure B.3. Point cloud of (x, y) data projected onto the first principal component

There are a number of reasons why a person might want to reduce a large number of dimensions to just one. If it were known *a priori* that the system had only one internal property that was changing between trials, then the true values of x and y might lie on a line as in Figure B.3, with some measurement noise causing the data to look like Figure B.1. In this case, projecting the data onto the line might be done to reduce the noise.

Another use of principal component analysis doesn't become apparent until three or more values are recorded for each trial: the ability to visualize data. Suppose there is an experiment which returns three values for each trial, x, y, and z, and that there are three

different settings for the experiment: A, B, and C. The simulated results must be shown in three plots, one for each pair of axes: *x*-*y*, *x*-*z*, and *y*-*z* (Figure B.4).

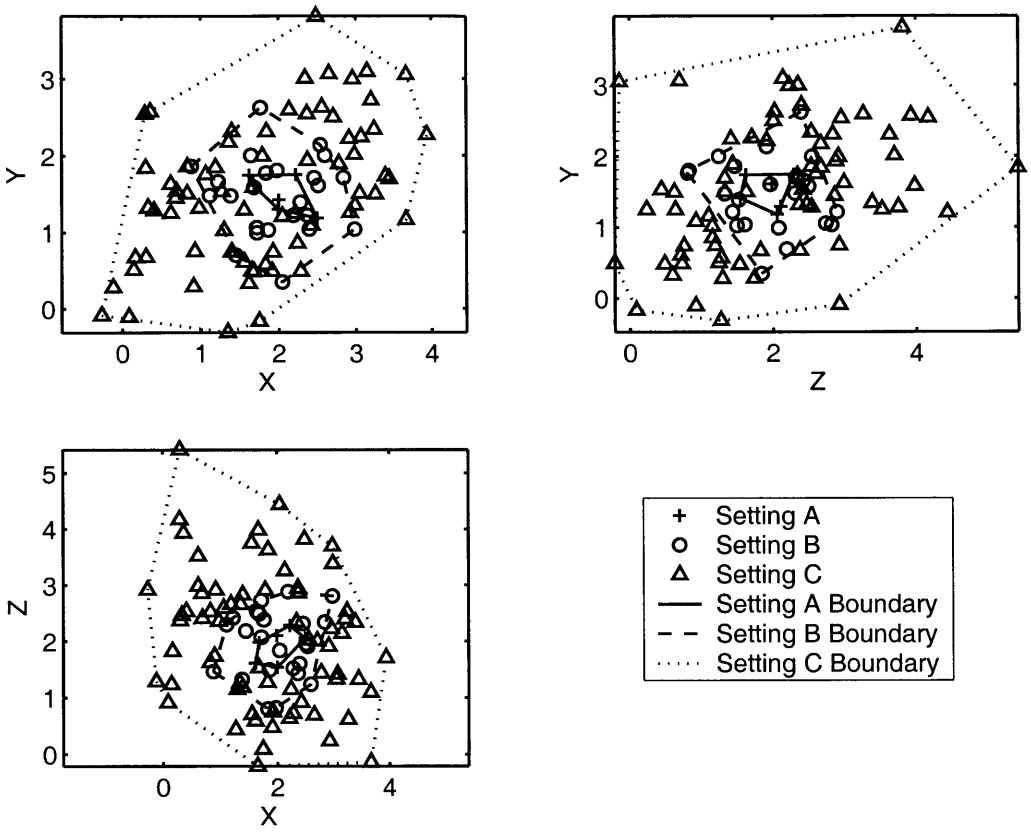


Figure B.4. Point cloud of grouped (*x*, *y*, *z*) data

In this case, the first two principal components may be used to describe the data. Thus, the data is projected from a space onto a plane. This plane can be viewed in terms of the original axes with little noticeable change (Figure B.5), or from a point normal to the plane, using the principal components as the new axes (Figure B.6). It is apparent that viewing the data in terms of the principal components can be very useful in providing insight into trends within the data.

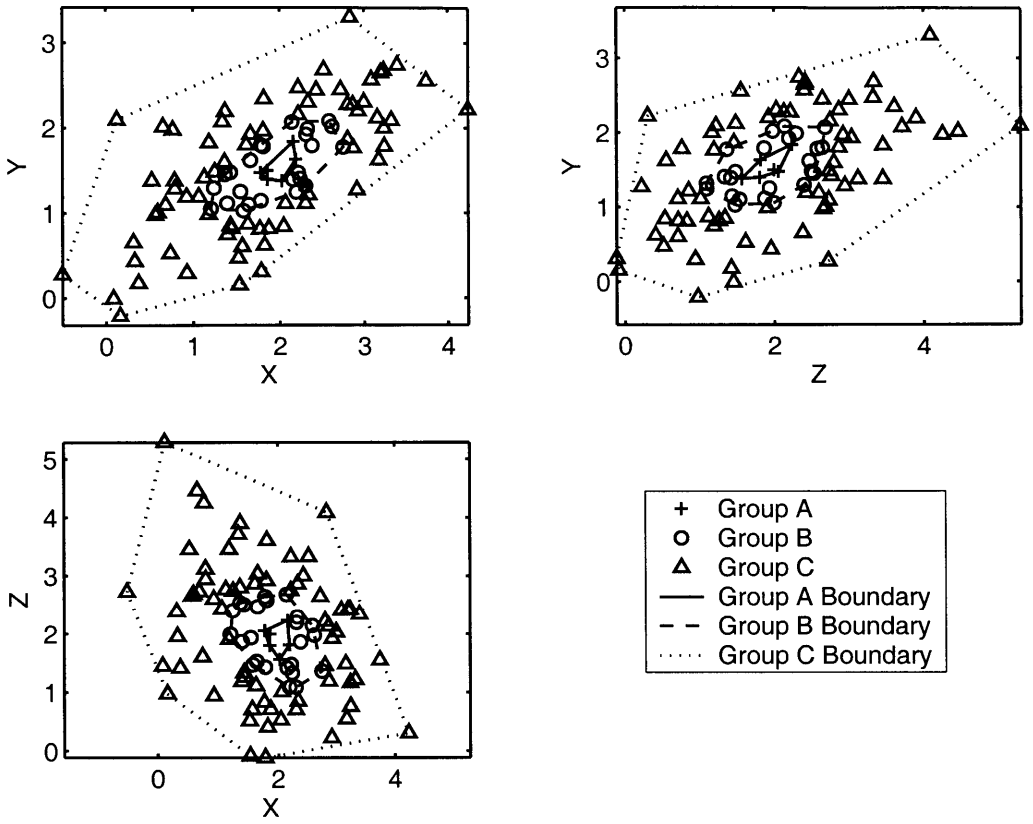


Figure B.5. Point cloud of (x, y, z) data projected onto the plane spanned by the first two principal components

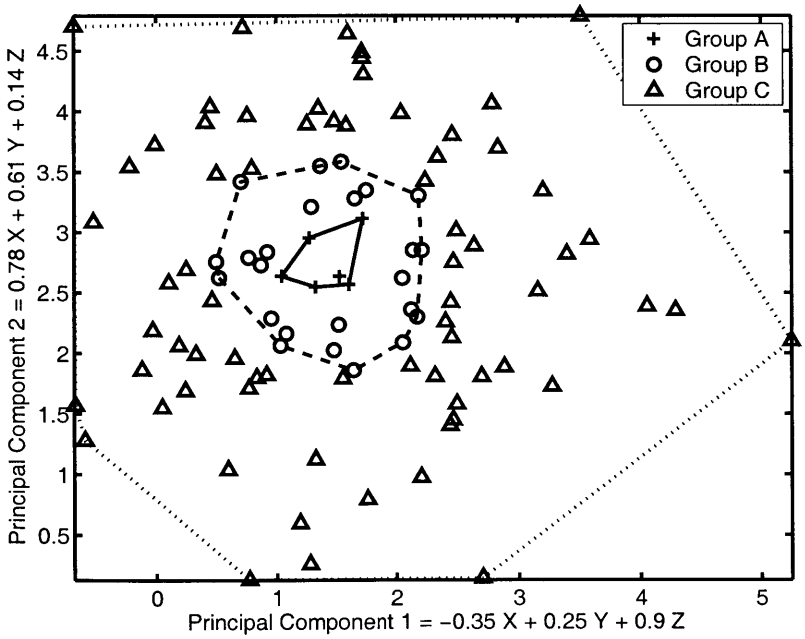


Figure B.6. Point cloud of (x, y, z) data plotted as coefficients of the first two principal components

As the number of dimensions increases still further, it becomes impossible to even attempt to present the initial data as a scatter plot. As an example, consider a system in which a single trial consists of repeatedly sampling a value 100 times. Figure B.7 overlays 16 trials, plotted as time histories. Each point in time can be thought of as a dimension. Already it is difficult to understand what is changing between data sets.

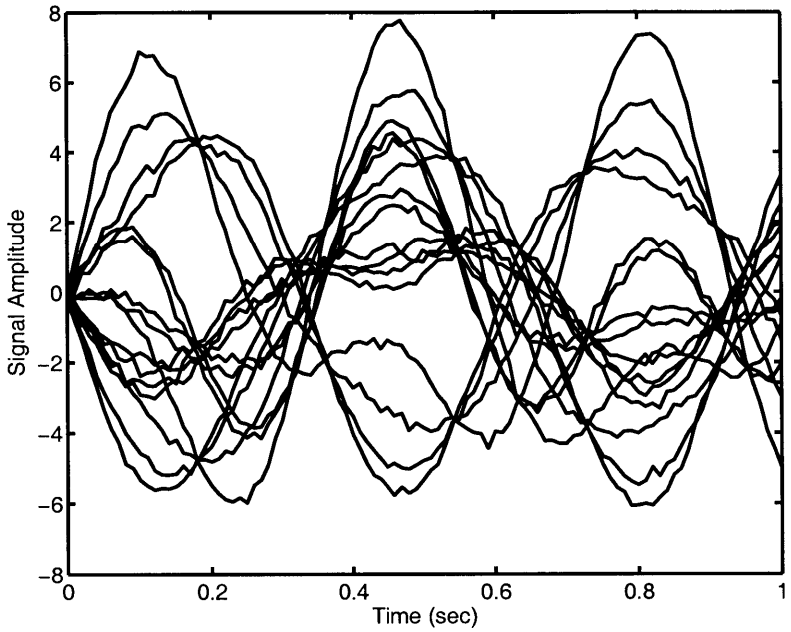


Figure B.7. Time histories of signals from 16 trials

However, plotting coefficients of the first two principal components of the data against each other shows that it is anything but random (Figure B.8). Instead, it is clear that two internal variables (one for each of the first two principal components) take on four values each.

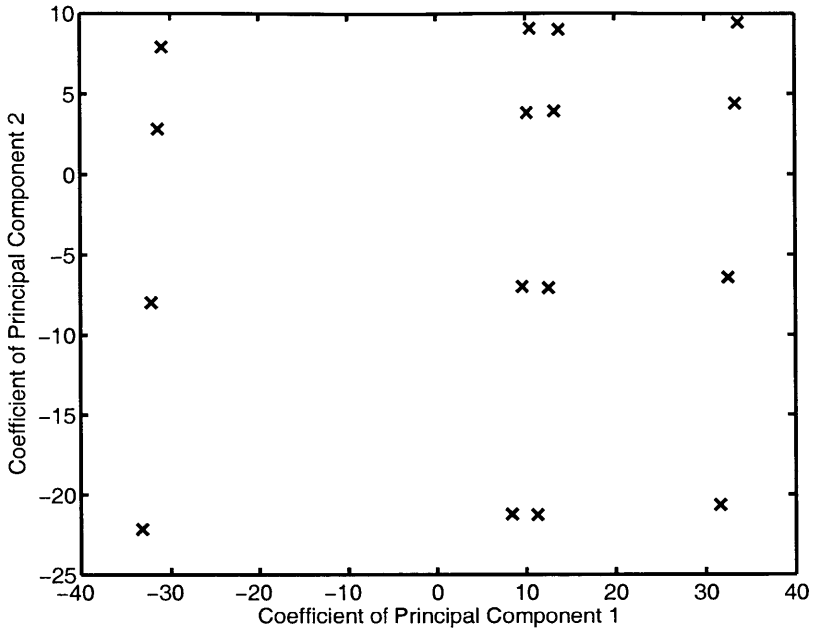


Figure B.8. Signals from 16 trials plotted as coefficients of the first and second principal components

For these higher-dimensional data sets, the principal component vectors themselves may also provide information about the system. Figure B.9 plots the first two principal components of this data set. It may be observed that the data in Figure B.7 can be written as linear combinations of two sinusoids, the principal components, with noise superimposed.

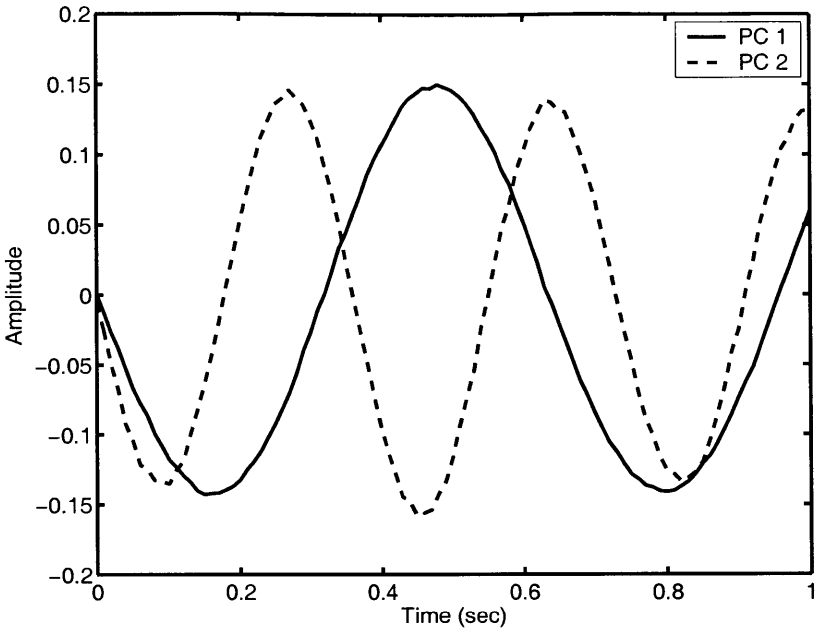


Figure B.9. First two principal components plotted as time histories

With these examples, it may be seen that principal component analysis has several benefits. As seen in the first three figures, it can be used for reducing the dimensionality of a data set, either for noise reduction or to summarize the data in terms of fewer independent variables. It can be used as a visualization tool, as seen in the second three figures, to choose an appropriate plane on which to view data. Or, as in the last three figures, it can be used as an analysis tool, to gain insight into the variables underlying a system and to see how they affect measured signals.

Implementing Principal Component Analysis

The utility of principal component analysis has been demonstrated, however the question remains open as to what these principal components are in a mathematical sense and how they can be implemented. This section will address those issues.

For principal components to be able to summarize the data while retaining as much of the original information as possible, they must be combinations of values which

tend to vary together in the original data set. The traditional method for observing these tendencies in data is the covariance matrix.

The covariance matrix is a generalization of the variance to accommodate vectors of data. For a data set $\mathbf{X}$ of n trials, each resulting in m values, the individual trials may be written as

$$\mathbf{x}_j = \begin{bmatrix} x_{1,j} \\ \vdots \\ x_{m,j} \end{bmatrix}, j = 1,...,n. \qquad (B.1)$$

In this form, the mean $\mu_{\mathbf{x}}$ can be computed:

$$\mu_{\mathbf{x}} = E[\mathbf{x}] = \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j. \qquad (B.2)$$

Here, the original data set can be stored as $\mathbf{X}$, and a zero-mean version may be stored as $\hat{\mathbf{X}}$:

$$\mathbf{X} = [\mathbf{x}_1 \quad \cdots \quad \mathbf{x}_n] \qquad (B.3)$$

$$\hat{\mathbf{X}} = [(\mathbf{x}_1 - \mu_{\mathbf{x}}) \quad \cdots \quad (\mathbf{x}_n - \mu_{\mathbf{x}})]. \qquad (B.4)$$

Using these, the covariance matrix can be written as

$$Cov(\mathbf{X}) = E[(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{x} - \mu_{\mathbf{x}})^{\mathrm{T}}] = \frac{1}{n}\sum_{j=1}^{n}(\mathbf{x}_j - \mu_{\mathbf{x}})(\mathbf{x}_j - \mu_{\mathbf{x}})^{\mathrm{T}} = \frac{1}{n}\hat{\mathbf{X}}\,\hat{\mathbf{X}}^{\mathrm{T}}. \qquad (B.5)$$

Thus, the covariance matrix $Cov(\mathbf{X})$ is a $m \times m$ symmetric matrix indicating how likely each of the m values is to vary with each of the others.

The principal components of $\mathbf{X}$ are the eigenvectors of $Cov(\mathbf{X})$, the directions along which the values vary together. They are sorted in order of decreasing eigenvalue, such that the first principal component will be the eigenvector of $Cov(\mathbf{X})$ with the largest

eigenvalue, thus accounting for the largest fraction of the variation of $\mathbf{X}$. It is convention that each principal component is scaled to have unit magnitude.

These principal components may be written in matrix form as the $m \times m$ matrix $\mathbf{U}$:

$$\mathbf{U} = [PC_1 \quad \cdots \quad PC_m]. \tag{B.6}$$

This matrix is unitary ($\mathbf{U}^T$ is the inverse of $\mathbf{U}$), so the principal component representation of the data set $\mathbf{X}$ can be found as $\mathbf{U}^T\mathbf{X}$.

Using Singular Value Decomposition

In the above presentation, the covariance matrix of $\mathbf{X}$ is computed and its eigenvectors are found as the principal components of the data set. In practice, this is a computationally intensive operation. Singular value decomposition is a method for finding the principal components without explicitly computing the covariance matrix. The computational method resulting in the singular value decomposition will not be addressed here. Rather, the practical details of using singular value decomposition will be presented.

Singular value decomposition is a method for dividing an $m \times n$ matrix $\mathbf{A}$ into component matrices $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$, such that

$$\mathbf{A} = \mathbf{U}\ \mathbf{\Sigma}\ \mathbf{V}^T. \tag{B.7}$$

Here, $\mathbf{U}$ is a $m \times m$ unitary matrix, $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix whose elements decrease from upper-left to lower-right, and $\mathbf{V}$ is a $n \times n$ unitary matrix.

It can be easily shown that the columns of $\mathbf{U}$ are the eigenvectors of $\mathbf{A}\ \mathbf{A}^T$, with the diagonal elements of $\mathbf{\Sigma}\ \mathbf{\Sigma}^T$ as the corresponding eigenvalues. Similarly, the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{A}^T\mathbf{A}$, with the diagonal elements of $\mathbf{\Sigma}^T\mathbf{\Sigma}$ as the corresponding eigenvalues. Thus, the principal component matrix $\mathbf{U}$ of a data set $\mathbf{X}$ can be found by the singular value decomposition of the zero-mean data set $\hat{\mathbf{X}}$.

Additionally, the matrices $\mathbf{\Sigma}$ and $\mathbf{V}$ from the singular value decomposition of $\hat{\mathbf{X}}$ can be useful in principal component analysis. Their product, $\mathbf{\Sigma}\ \mathbf{V}^T$, is a $m \times n$ matrix giving the principal component coefficients for the zero-mean data set. Each column corresponds to one of the n trials. The rows correspond to the principal components. So the first element of the ith column will be the coefficient of the first principal component for $(\mathbf{x}_i - \mu_\mathbf{x})$.

Finally, the matrix $\mathbf{V}$ may be useful on its own. The first m rows of $\mathbf{V}^T$ form a representation of the data set $\hat{\mathbf{X}}$ in terms of scaled versions of the principal components, $\mathbf{U}\ \mathbf{\Sigma}$. The variance of each row of $\mathbf{V}^T$ is equal, so this may be an appropriate way to compare distributions which vary much more in one direction than in another.

Appendix C

FSRL Wheel-Terrain Interaction Testbed

The FSRL Wheel-Terrain Interaction Testbed (Figure C.1) is one of the experimental platforms available in the Field and Space Robotics Laboratory. It was designed for the purpose of studying the behavior of a rigid wheel driving in deformable terrain, and has been used in the past to characterize terrain properties based on the wheel-terrain interaction (Kang, 2003).

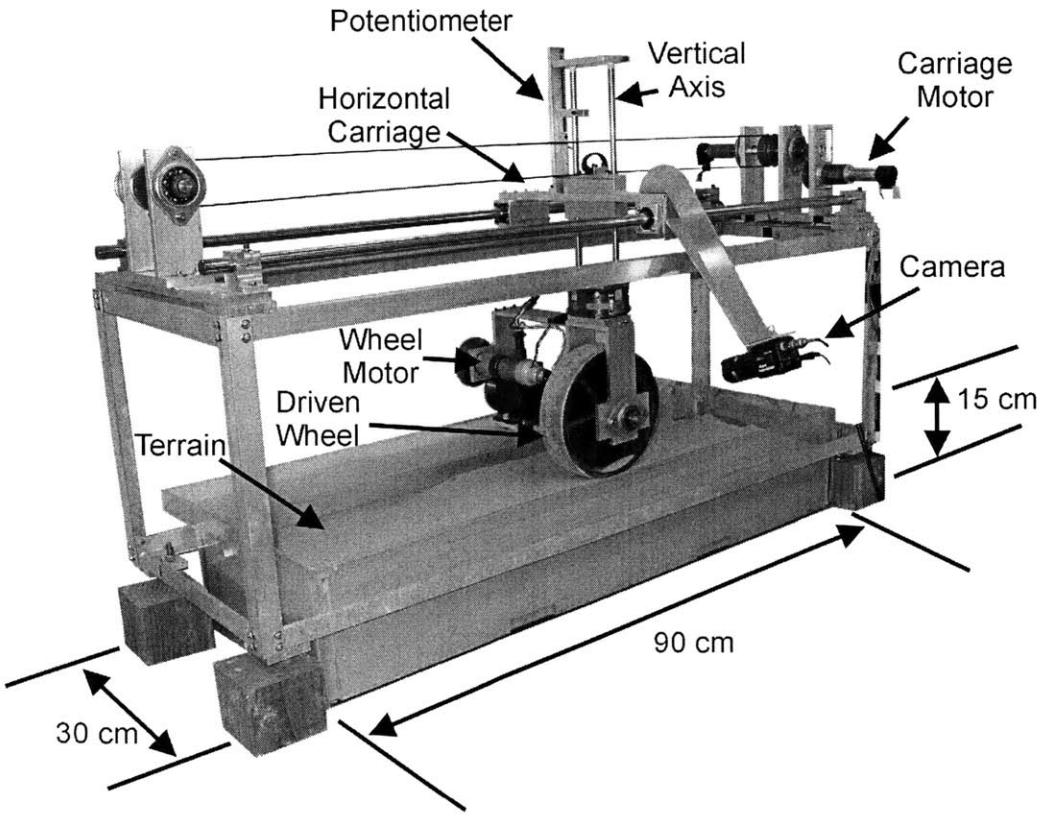


Figure C.1. FSRL Wheel-Terrain Interaction Testbed, with dimensions

The testbed consists of a driven wheel mounted on an undriven vertical axis. The wheel-axis assembly is mounted on a frame such that the wheel's forward velocity may be controlled independent of its angular velocity. The testbed can be fitted with a number of different wheel assemblies, containing both a motor and a wheel. Two in particular were used in the experiments described in this thesis: a black plastic wheel assembly, and a wheel assembly from the JPL FIDO rover (Schenker *et al*, 2001).

The black plastic wheel assembly, shown in Figure C.2 consists of a wheel, a motor, and a torque sensor. The wheel is 4.8 cm wide and 20 cm in diameter. Sand is bonded to the outside of the wheel to improve traction. The motor applying the torque to the wheel is a 14.5-watt DC brush-type motor. It is mounted with a 246:1 transmission, and has a tachometer to measure angular velocity. The maximum linear velocity of the outside of the wheel is 15 cm/sec. Between the motor and the wheel, there is a rotating torque sensor with working range of 7 N-m. A six-axis force/torque sensor is mounted between this wheel assembly and the frame, such that all of the forces and torques applied by the wheel on the terrain may be sensed. Table C.1 provides detailed information about this wheel assembly including model numbers.

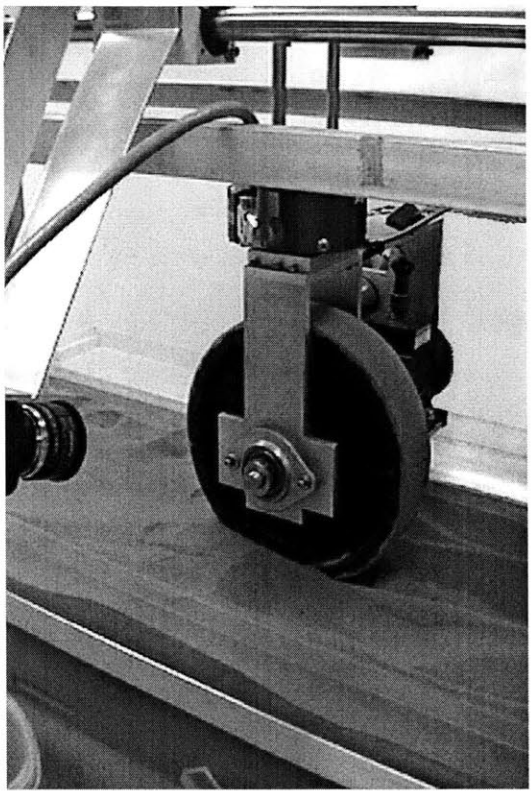


Figure C.2. Black wheel on FSRL Wheel-Terrain Interaction Testbed

Black Plastic Wheel Assembly		
Diameter	20 cm	
Width	4.8 cm	
Motor	14.5-watt DC brush-type	Faulhaber 3557K006C
Transmission	246:1	Faulhaber 38/2
Sensors	Tachometer	
	Torque Sensor	Cooper LXT 962
	Force-Torque Sensor	JR3 UFS-3515A100

Table C.1. Specifications for black plastic wheel assembly

The wheel assembly from the JPL FIDO rover, shown in Figure C.3, is on loan from the Jet Propulsion Laboratory. It consists of an aluminum wheel 20.3 cm in diameter and 12.7 cm in width, with serrated grousers protruding 0.47 cm from the wheel surface. This wheel is driven by a small Maxon DC motor, with a 1621:1 transmission ratio. Its position is measured with a 16-count-per-turn encoder. The maximum linear

velocity of the outside of the FIDO wheel is 5 cm/sec. The steering angle of the wheel is also controllable, but this was not used in the experiments. Table C.2 summarizes this information.

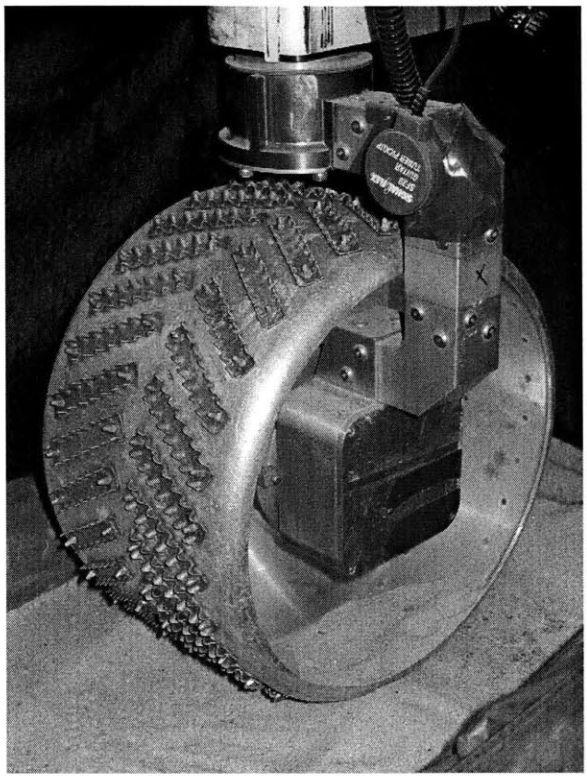


Figure C.3. FIDO Wheel on FSRL Wheel-Terrain Interaction Testbed

FIDO Wheel Assembly		
Diameter	20.3 cm	
Width	12.7 cm	
Motor	DC motor	Maxon
Transmission	1621:1	Maxon
Sensor	Encoder	16 count/turn

Table C.2. Specifications for FIDO wheel assembly

These wheel assemblies are mounted to the carriage, which allows them to move freely in the vertical direction while constraining their forward and lateral movement. The vertical position of the wheel is measured using a linear potentiometer mounted on the carriage. The horizontal position of the carriage is controlled with an 8.5-watt brush-

type DC motor, with an overall 1936:5 transmission ratio mounted to a 5 cm-diameter pulley. The pulley position is sensed with a 2048-PPR encoder. The maximum carriage forward velocity is 5 cm/sec. This information is summarized in Table C.3.

Carriage		
Motor	8.5-watt DC brush-type	Escap 23DT
Transmission	1936:5 overall	
Pulley diameter	5 cm	
Sensor	Encoder	2048 PPR

Table C.3. FSRL Wheel-Terrain Interaction Testbed carriage drive specifications

The testbed also supports a camera mounted to the carriage to capture images of the wheel as it drives. The camera is a color CCD camera with a varifocal 3.5mm-8.0mm lens. It is mounted such that it moves horizontally with the wheel, but not vertically. This positioning allows the wheel to move within the field of view, but not so far as to leave the field of view. Camera and lens model numbers are presented in Table C.4.

Sensors		
Camera	1/3" Color CCD	Genwac GW-202B
Lens	Varifocal 3.5mm-8.0mm	Edmund Optics NT55-255

Table C.4. FSRL Wheel-Terrain Interaction Testbed camera specifications

All signals from the testbed are sent to an AMD K6-2 500MHz desktop computer running Windows 2000. Video signals are sent to a frame grabber. All other signals are sent to an 8-axis 12-bit I/O board. I/O board model numbers are presented in Table C.5. Figure C.4 shows a schematic of the communications between the testbed and the computer. The data sampling and control software was written specifically for this testbed.

I/O Boards	
Analog/Digital I/O	ServoToGo STGII-8
Frame Grabber	Data Translation DT3120

Table C.5. FSRL Wheel-Terrain Interaction Testbed I/O boards

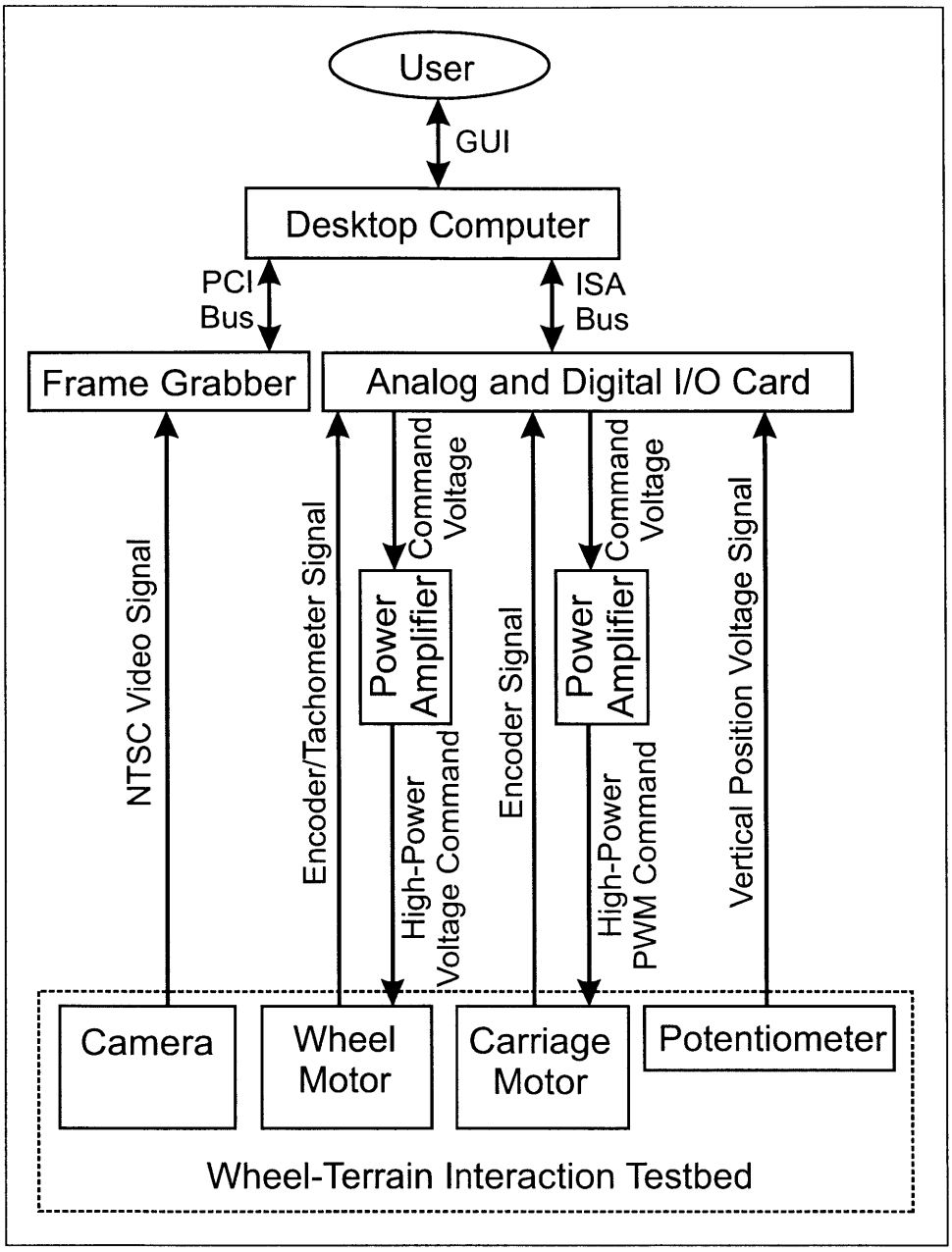


Figure C.4. Communications schematic for FSRL Wheel-Terrain Interaction Testbed

Control signals from the computer to the testbed are output as voltage signals using the same 8-axis I/O board. These voltage signals go to a custom power amplifier card. The power amplifier card sends a high-power voltage signal (15 V, 3A max) to the wheel motor, and a high-power PWM signal (25V, 3A max) to the carriage motor.

The wheel rests in a bin of terrain 90 cm long by 30 cm wide by 15 cm deep. A large number of terrains have been used, including washed beach sand, dry bentonite, JSC Mars-1 soil simulant, gravel, moist clay, and topsoil.

Appendix D

FSRL Technology Testbed Rover

The FSRL Technology Testbed Rover (a.k.a. TORTOISE, for all-Terrain Outdoor Rover Testbed fOr Integrated Sensing Experiments, see Figure D.1) is one of the test platforms available in the Field and Space Robotics Laboratory. It was specifically designed to study terrain interaction and sensing issues affecting planetary rovers.

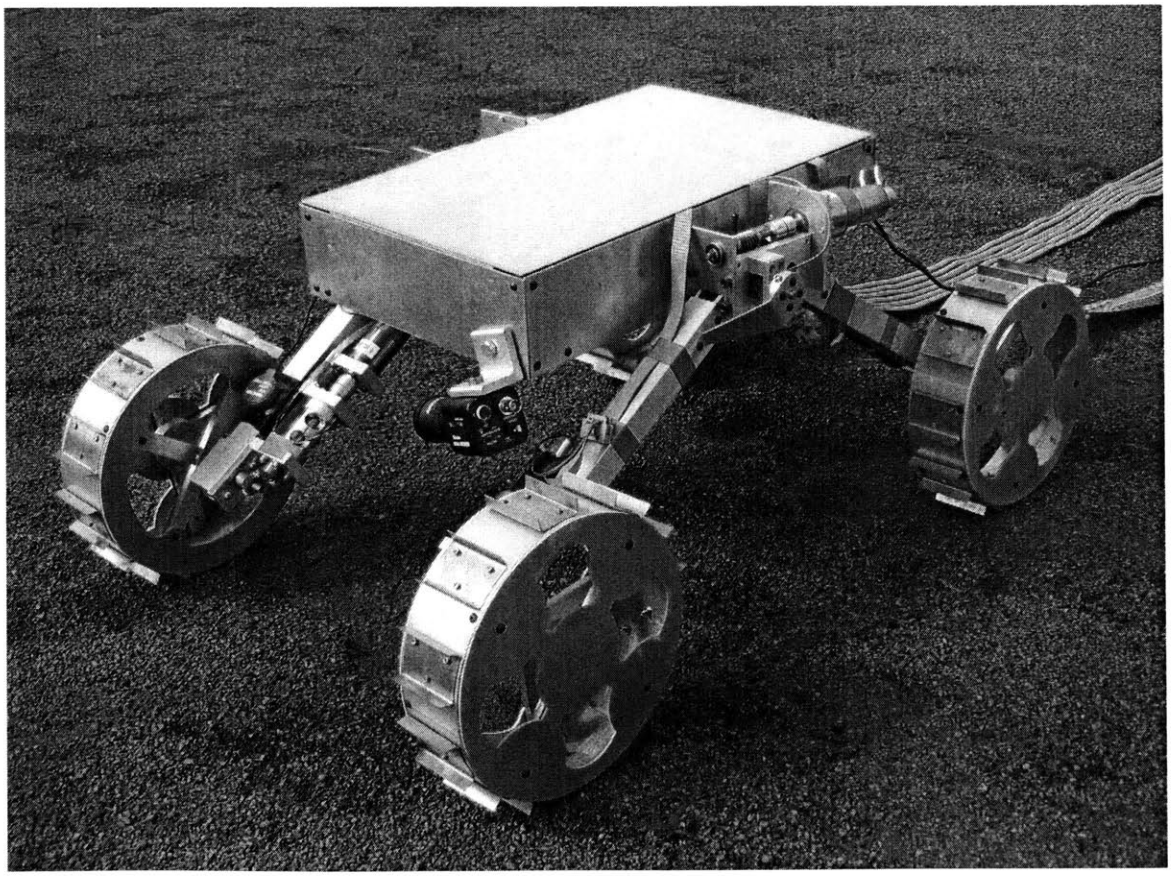


Figure D.1. FSRL Technology Testbed Rover

TORTOISE is a four-wheeled rover with an actively reconfigurable suspension. It is 80 cm long in its longest configuration, is 50 cm wide at its widest point, and has a mass of 12.2 kg. The angles between the two legs on each side can be controlled independently. The two shoulders are connected to the main body with a differential so that one can rotate while the other remains stationary. The main body maintains an angle midway between the two sides. For example, if the right pair of legs rotates 20° with respect to the left pair, the main body will rotate 10°. The direction of the rover is controlled using skid steering. Rover dimensions are summarized in Table D.1.

Dimensions	
Length	80 cm max
Width	50 cm
Mass	12.2 kg
Wheel Diameter	20 cm
Wheel Width	5.1 cm
Grouser Length	1.0 cm

Table D.1. FSRL Technology Testbed Rover dimensions

The four wheels are made of rigid aluminum tubing. Each wheel is 20 cm in diameter and 5.1 cm wide, with 20 stainless steel grousers extending 1.0 cm from the surface of the wheel. The wheels are powered by 12-watt DC brush-type motors with 246:1 planetary gearboxes. The shoulder joints are powered by 10.5-watt DC brush-type motors, with 134:1 planetary gearboxes and a 20:1 worm/worm-gear pair. The motion of all six motors is sensed using magnetic encoders. Motor and transmission details are presented in Table D.2.

Motors and Transmissions		
Wheel Motor	12-watt DC brush-type	Faulhaber 2342S012CR
Wheel Transmission	246:1	Faulhaber 30/1 246:1
Shoulder Motor	10.5-watt DC brush-type	Faulhaber 2842S012C
Shoulder Transmission	268:1 overall	Faulhaber 38/1 134:1 20:1 worm/worm gear

Table D.2. FSRL Technology Testbed Rover motors and transmissions

The front right wheel of the rover (on the left in Figure D.1) is equipped with several sensors to study the terrain it is traversing. A 5.6 N-m torque sensor measures the torque the motor is applying to the wheel (see Figure D.2). A contact microphone is mounted to the leg of the rover near the front right wheel (see Figure D.3) for vibration sensing. Additionally, a color CCD camera with a 3.5mm-8.0mm varifocal lens is mounted to the rover body where it can maintain a view of the inside of the front right wheel.

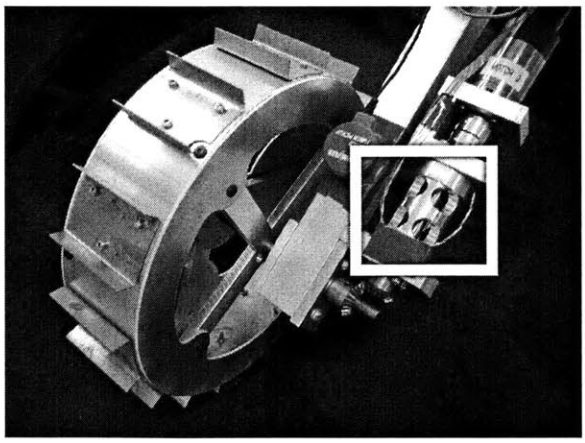


Figure D.2. Torque sensor mounted on TORTOISE

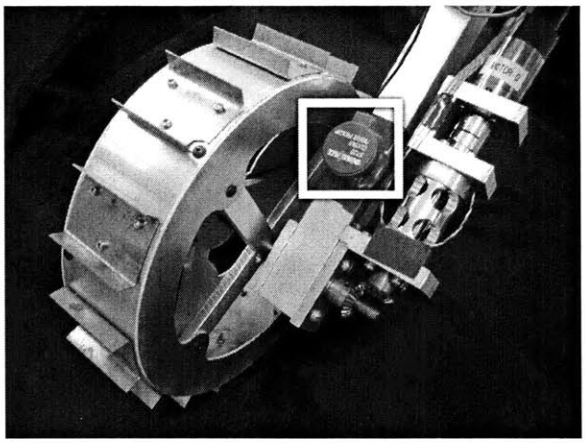


Figure D.3. Vibration sensor mounted on TORTOISE

The rover is also outfitted with sensors to fully estimate its configuration. A two-axis tilt sensor measures the pitch and roll of the rover body. The angles of the two shoulder joints are measured with potentiometers, as is the angle between the right shoulder and the body. Model numbers for all sensors are shown in Table D.3.

Sensors		
Motor Rotation	Magnetic encoders	Faulhaber HEM2342S16
Torque	5.6 N-m Torque sensor	Futek T5160
Vibration	Contact microphone	Signal Flex SF-20
Vision	1/3" CCD camera	Genwac GW-202B
	3.5mm-8.0mm Lens	Edmund Optics NT55-255
Configuration	2-axis Tilt sensor	Crossbow CXTA02
	Potentiometers	Vishay/Spectrol 65700103

Table D.3. FSRL Technology Testbed Rover sensors

All control and data sampling is done off-board. Motor power is sent to the rover via a tether and sensory signals are returned the same way. Motor control is done using an off-board PC104 computer. Sampling of wheel torque and rover configuration is done by the same PC104 system. Image capture from the CCD camera and vibration signal recording is done off-board on a tethered laptop computer. See Figure D.4 for a schematic of the rover communications.

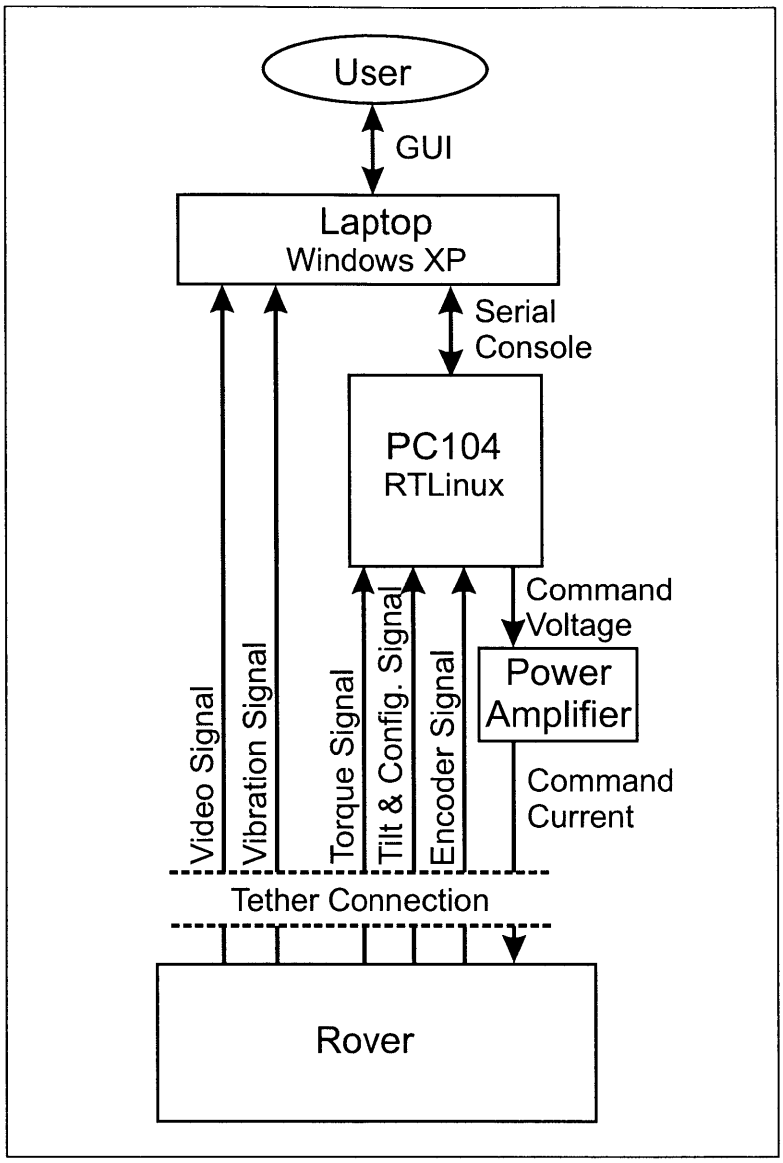


Figure D.4. Rover communications schematic

The PC104 system is a Diamond Systems Prometheus LC, a ZFx86 processor equivalent to a 486-100MHz. Analog signals, including signals from the tilt sensor, the torque sensor, and the potentiometers, are sensed using an analog input card. Encoder signals are received by a quadrature decoder card. Control signals for the motors are sent as voltage outputs from an analog output card. These voltage signals are then translated

into current signals by a custom power amplifier board, based on the National Semiconductor LMD18245 full-bridge motor driver. The current signals are sent to the motors via the tether.

The PC104 system is running Linux with an RTLinux microkernel, for real-time control and data sampling. The control and sampling software was developed specifically for this rover. User interaction with the PC104 system is done via a serial console connection to the laptop computer.

The laptop computer, a PC running Windows XP, interacts with the PC104 system, the CCD camera, and the vibration sensor. It connects to the PC104 system using a null modem serial cable. It connects to the CCD camera with a USB video capture box. The connection to the vibration sensor is via a standard audio cable which plugs into the laptop's microphone port. Model information for the input/output boards is shown in Table D.4.

I/O Boards	
PC104 Analog Input, Digital I/O	Diamond MM-AT
PC104 Quadrature Decoder	Microcomputer Systems MSI-P400
PC104 Analog Output	Ruby MM-4XT
USB Video Capture	ProVideo PV321CE

Table D.4. FSRL Technology Testbed Rover I/O boards

The entire system is run using battery power, so it can be taken to remote locations where electrical outlets are unavailable. Conveniently accessible locations include terrains such as gravel, concrete, grass, sand, and a mixture of sand, silt and clay.

Appendix E

Visual Wheel Sinkage Measurement Plots

Image Set 1:

Bentonite, high slip, flat terrain

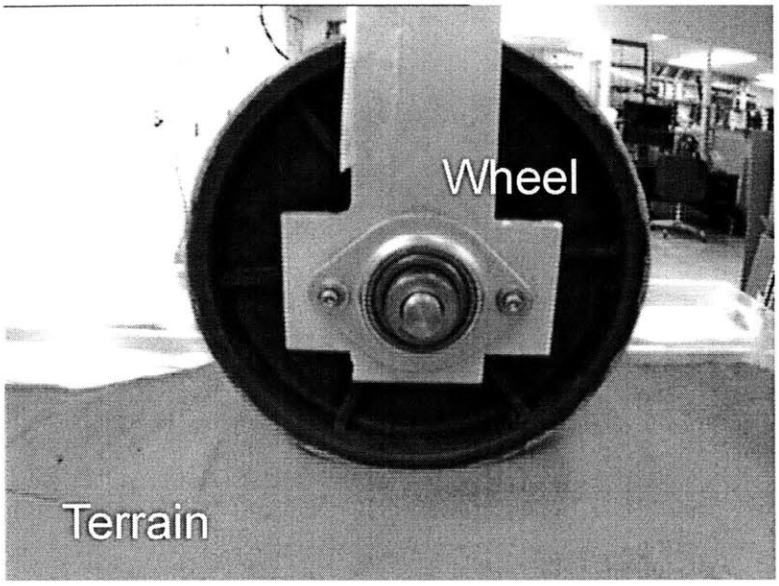


Figure E.1. Representative image from image set 1

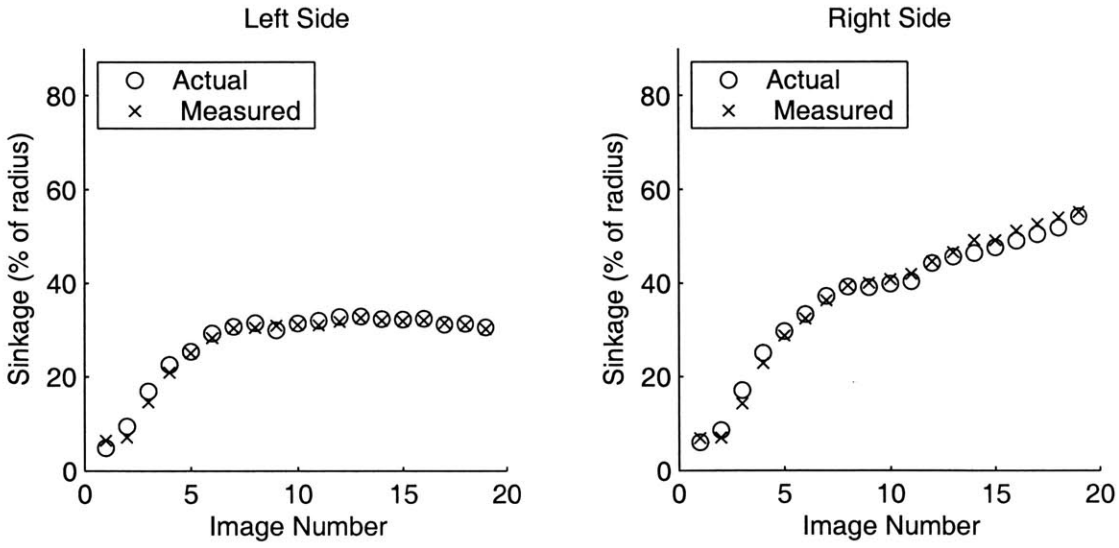


Figure E.2. Actual and visually-measured wheel sinkage, image set 1

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
1.08	1.61

Table E.1. Visual sinkage measurement RMS error, image set 1

Image Set 2:

JSC Mars-1, high slip, flat terrain

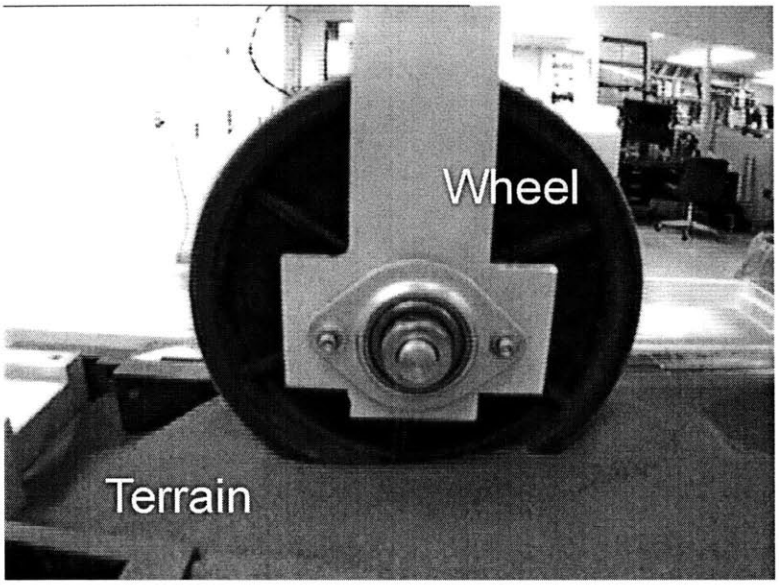


Figure E.3. Representative image from image set 2

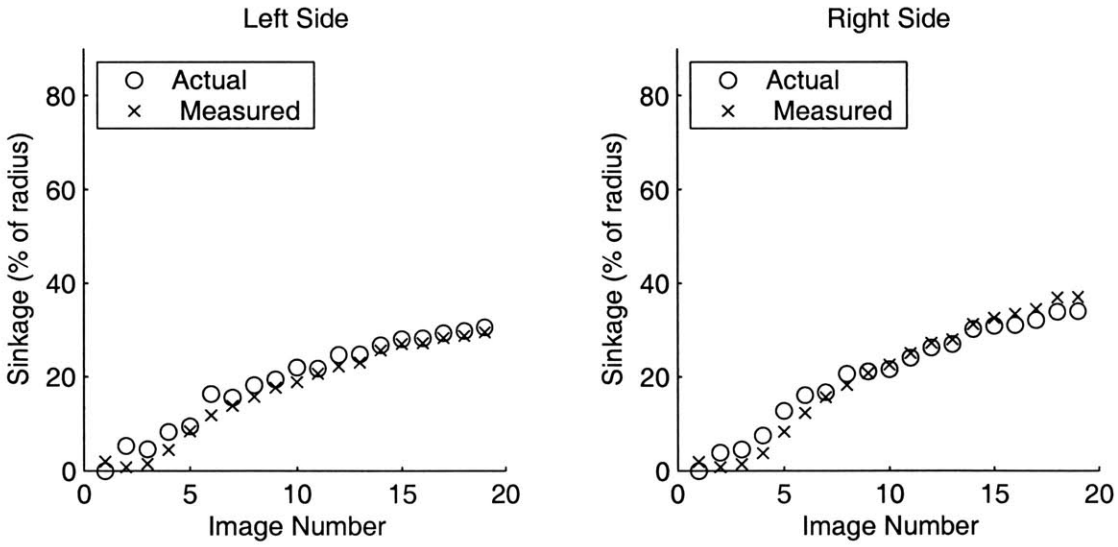


Figure E.4. Actual and visually-measured wheel sinkage, image set 2

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
2.40	2.46

Table E.2. Visual sinkage measurement RMS error, image set 2

Image Set 3:

Bentonite, low slip, flat terrain

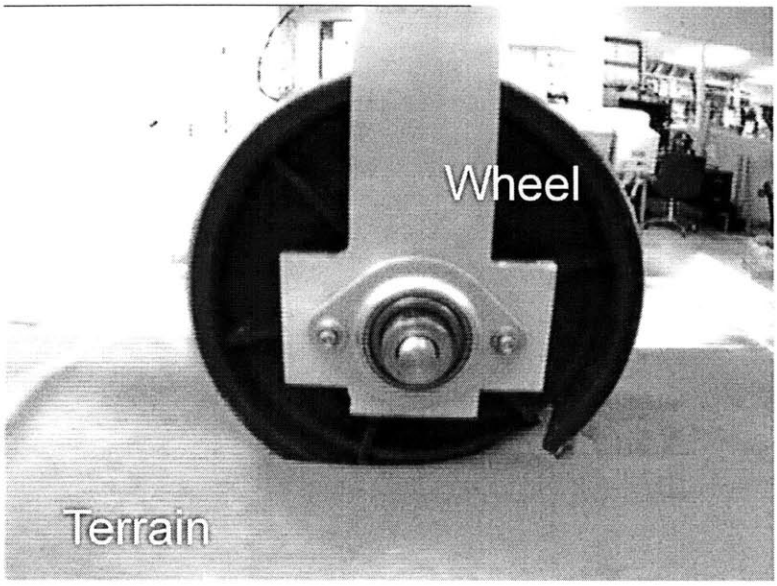


Figure E.5. Representative image from image set 3

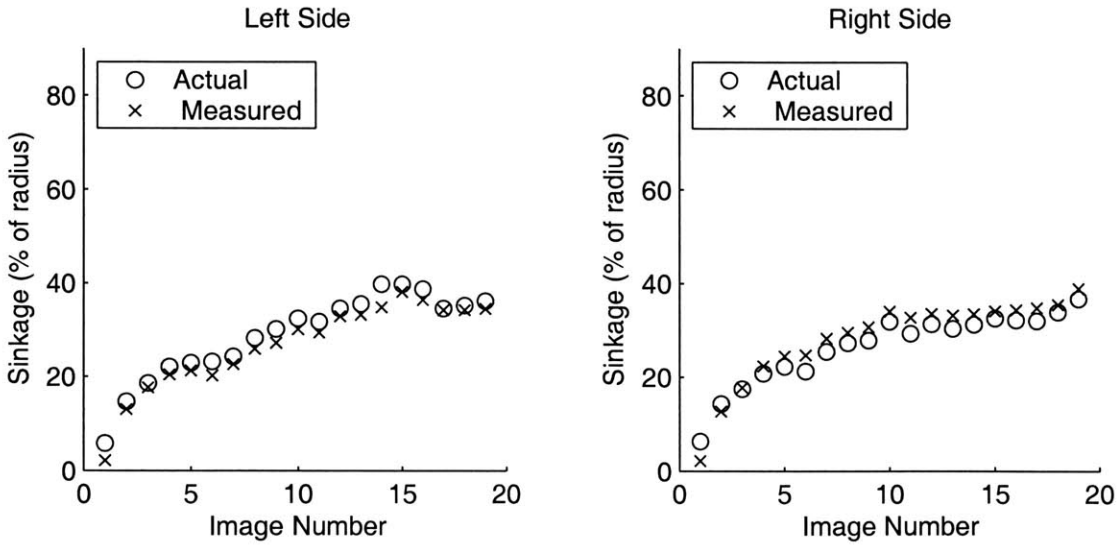


Figure E.6. Actual and visually-measured wheel sinkage, image set 3

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
2.33	2.48

Table E.3. Visual sinkage measurement RMS error, image set 3

Image Set 4:

Bentonite with rocks, low slip, uneven terrain

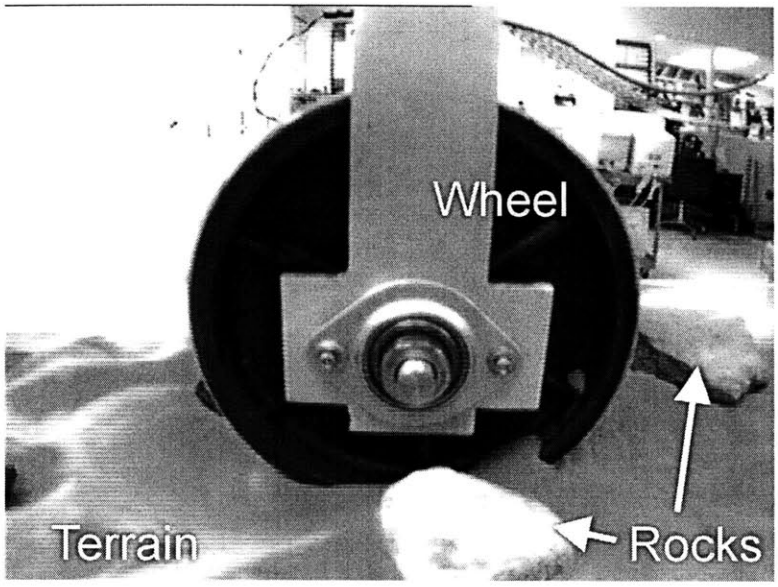


Figure E.7. Representative image from image set 4

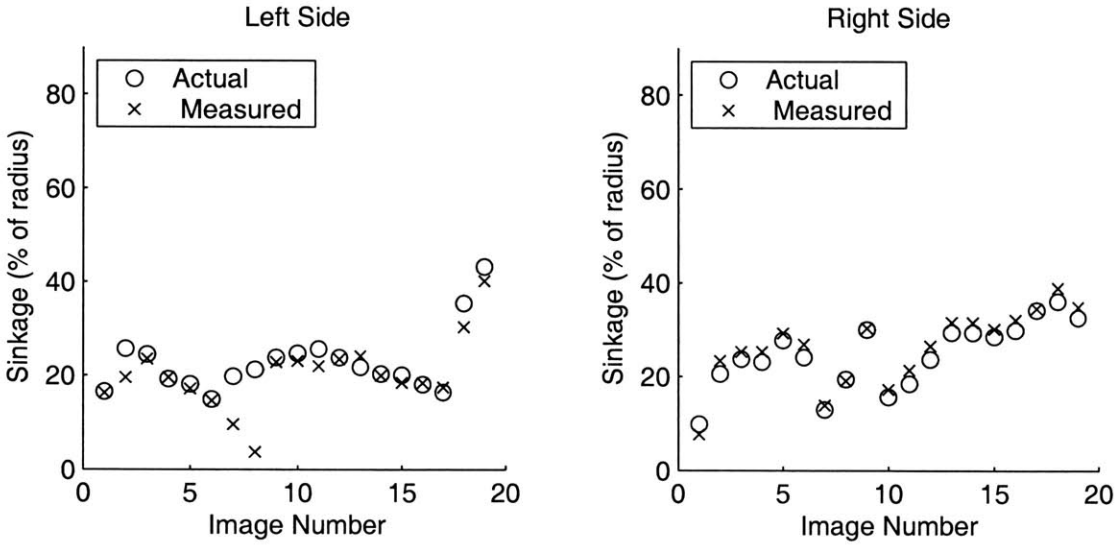


Figure E.8. Actual and visually-measured wheel sinkage, image set 4

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
5.21	2.06

Table E.4. Visual sinkage measurement RMS error, image set 4

Image Set 5:

Bentonite, stationary wheel, uneven terrain, moving point light source

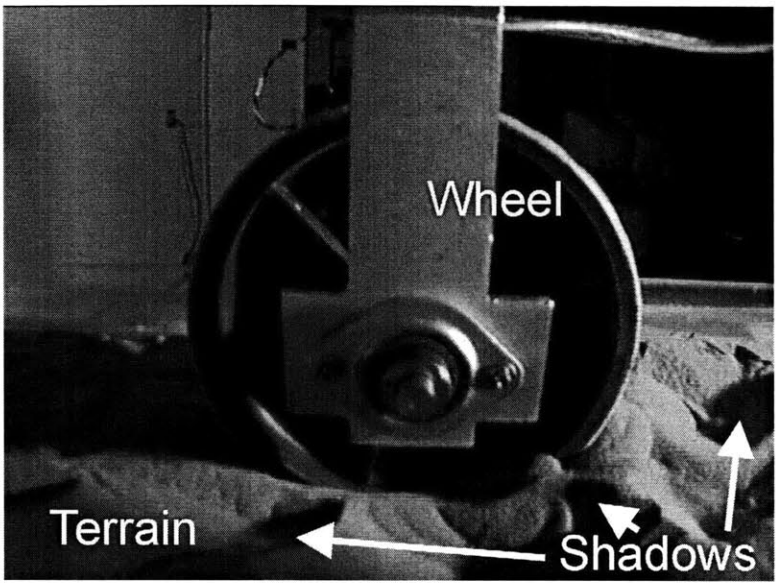


Figure E.9. Representative image from image set 5

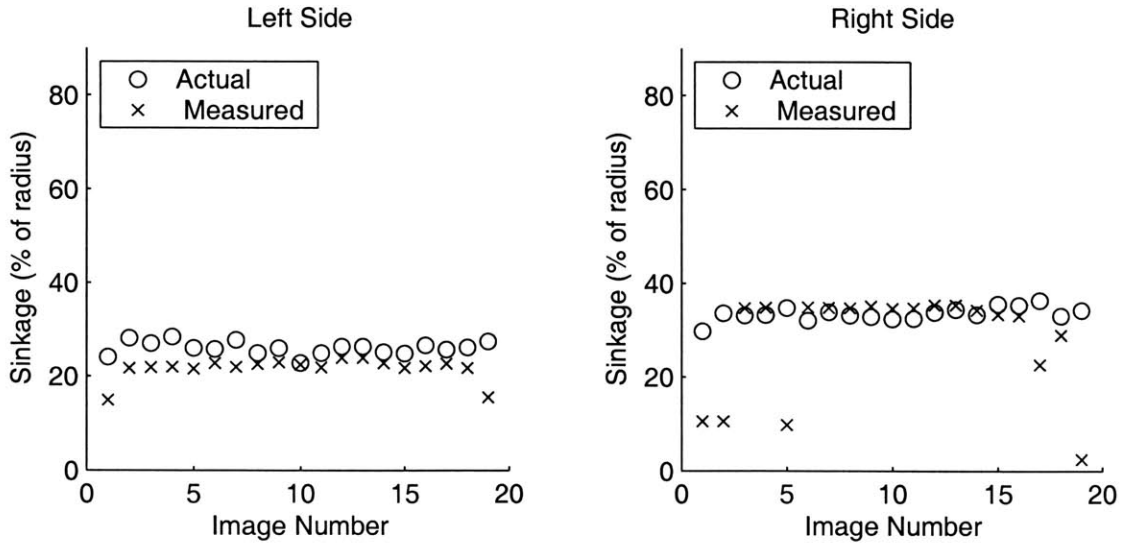


Figure E.10. Actual and visually-measured wheel sinkage, image set 5

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
5.10	12.10

Table E.5. Visual sinkage measurement RMS error, image set 5

Image Set 6:

Bentonite, low slip, uneven terrain, stationary point light source

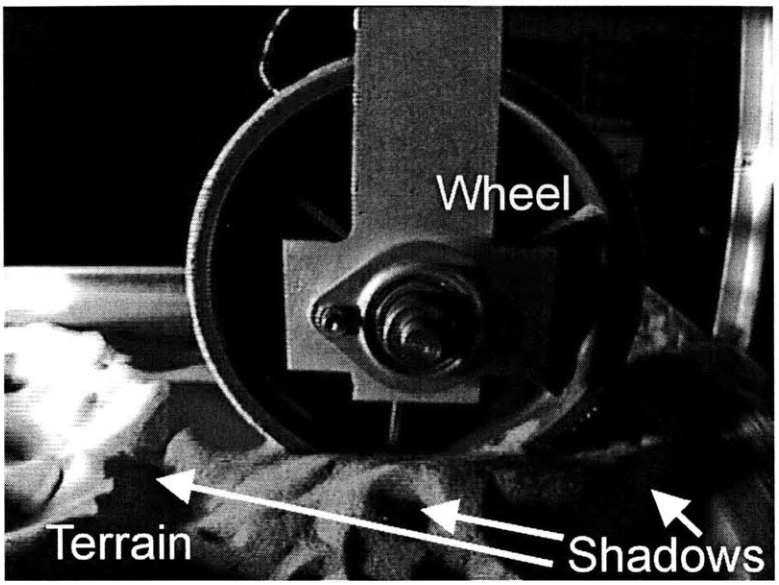


Figure E.11. Representative image from image set 6

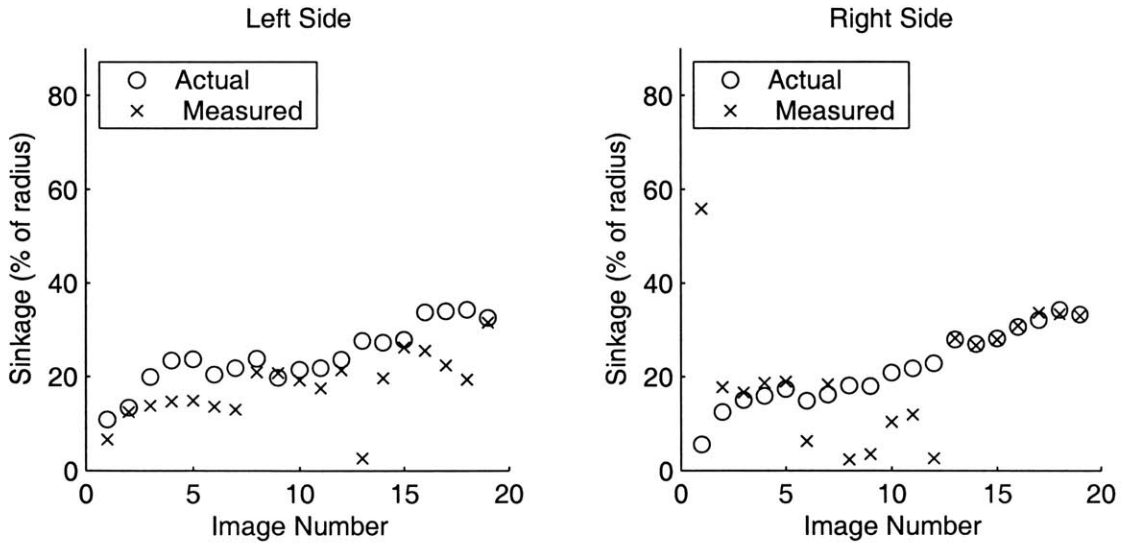


Figure E.12. Actual and visually-measured wheel sinkage, image set 6

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
8.85	14.01

Table E.6. Visual sinkage measurement RMS error, image set 6

Image Set 7:

Topsoil, low slip, flat terrain, active lighting

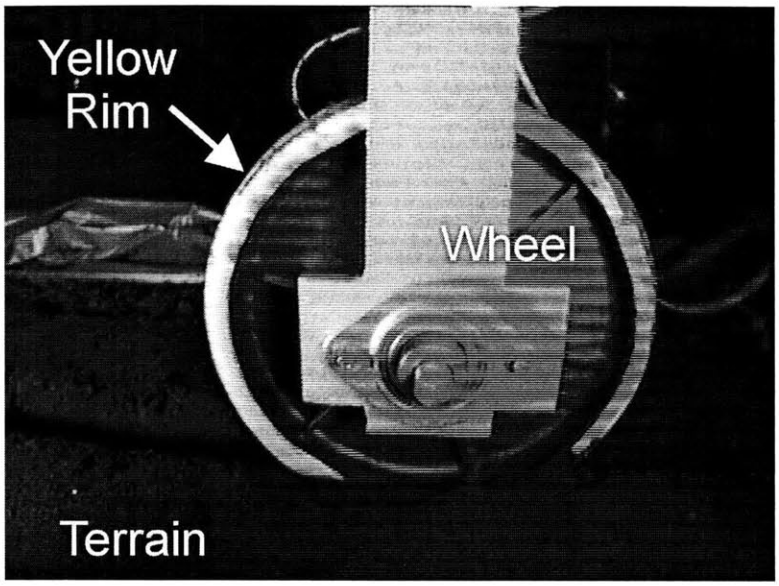


Figure E.13. Representative image from image set 7

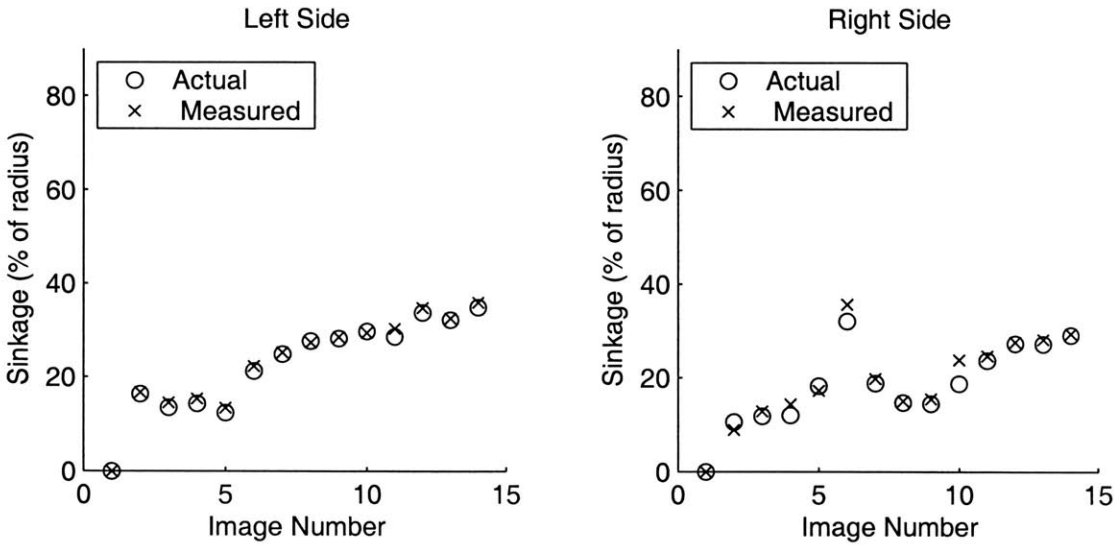


Figure E.14. Actual and visually-measured wheel sinkage, image set 7

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
0.86	1.96

Table E.7. Visual sinkage measurement RMS error, image set 7

Image Set 8:

Color-based algorithm, bentonite, low slip, uneven terrain, blue rim

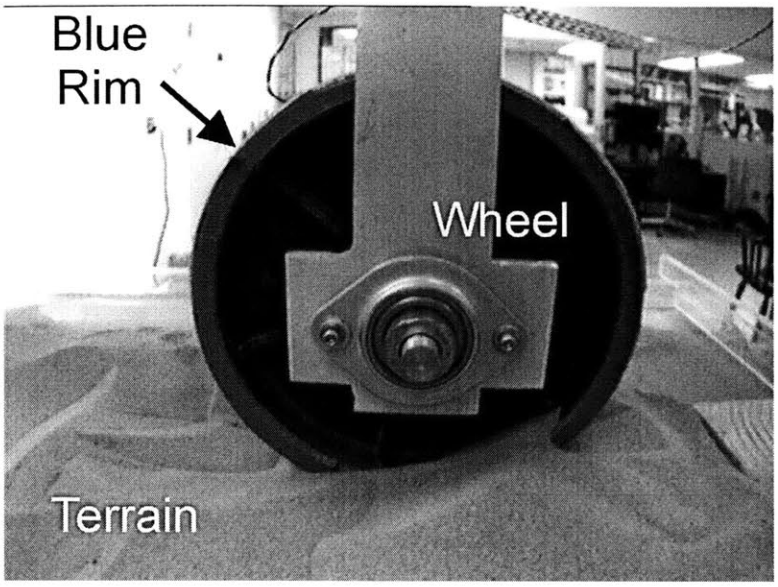


Figure E.15. Representative image from image set 8

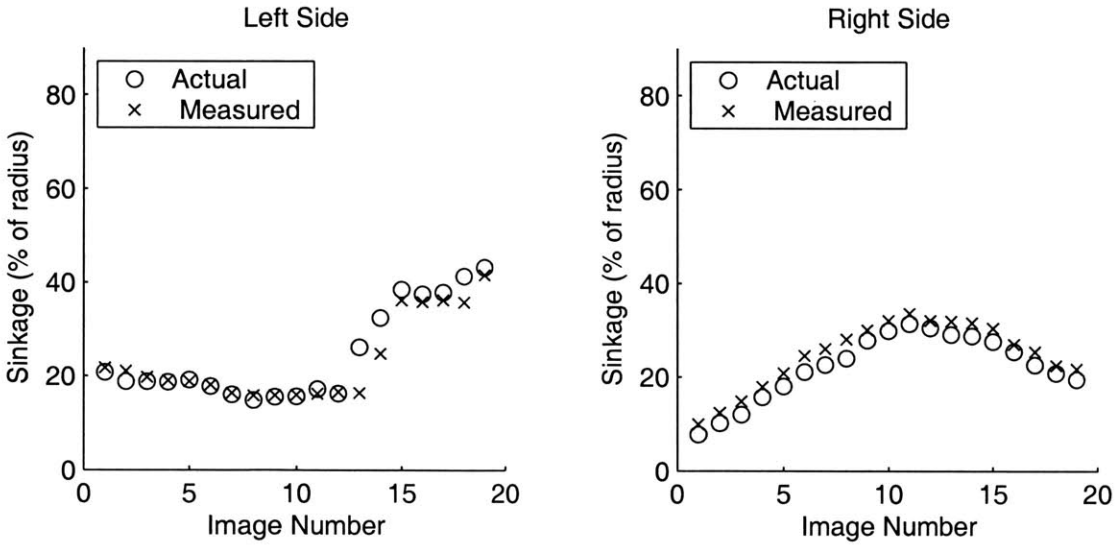


Figure E.16. Actual and visually-measured wheel sinkage, image set 8

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
3.30	2.61

Table E.8. Visual sinkage measurement RMS error, image set 8

Image Set 9:

Color-based algorithm, bentonite, low slip, uneven terrain, blue rim, stationary point light source

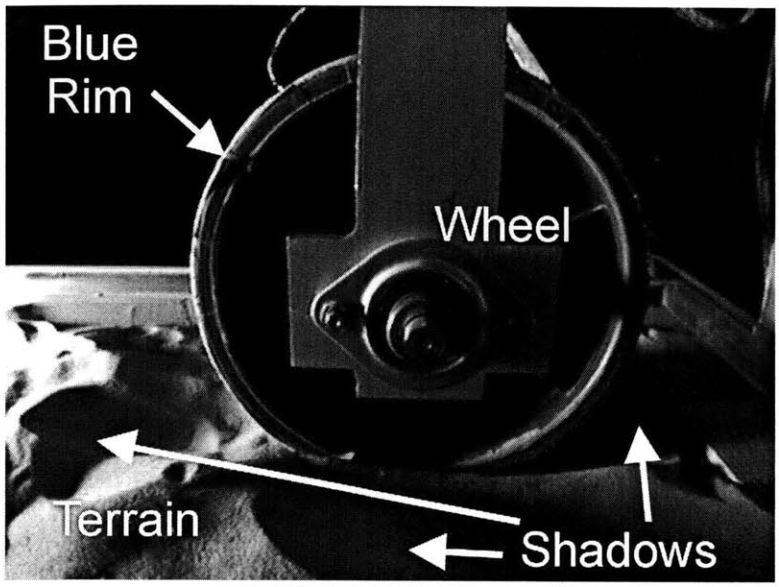


Figure E.17. Representative image from image set 9

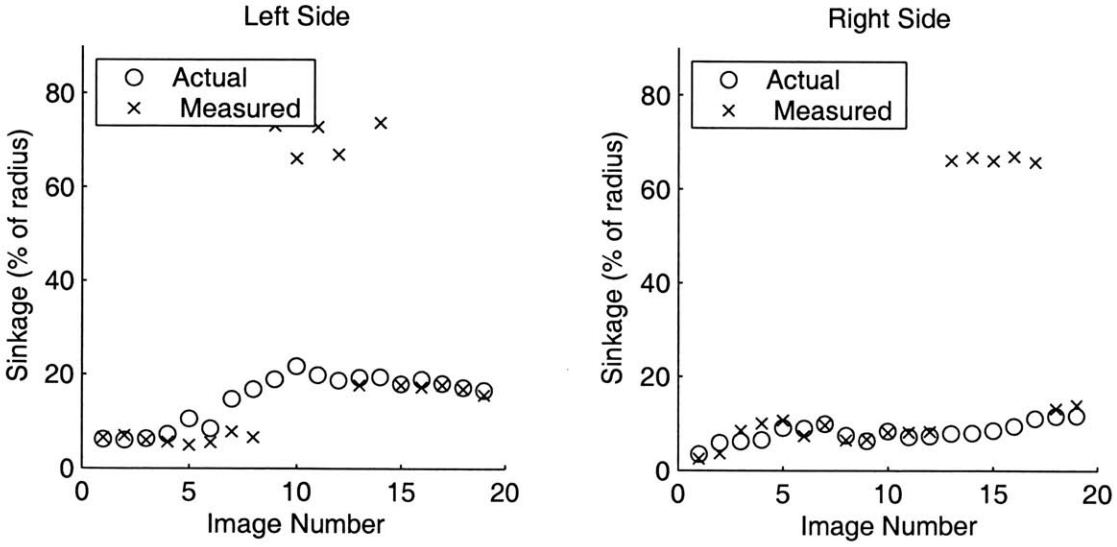


Figure E.18. Actual and visually-measured wheel sinkage, image set 9

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
26.3	29.45

Table E.9. Visual sinkage measurement RMS error, image set 9

Image Set 10:

Color-based algorithm, bentonite with rocks, low slip, uneven terrain, blue rim

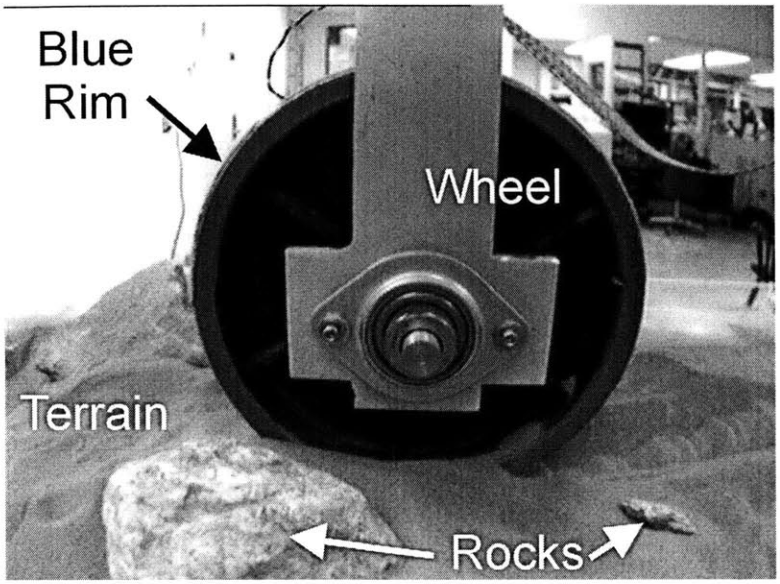


Figure E.19. Representative image from image set 10

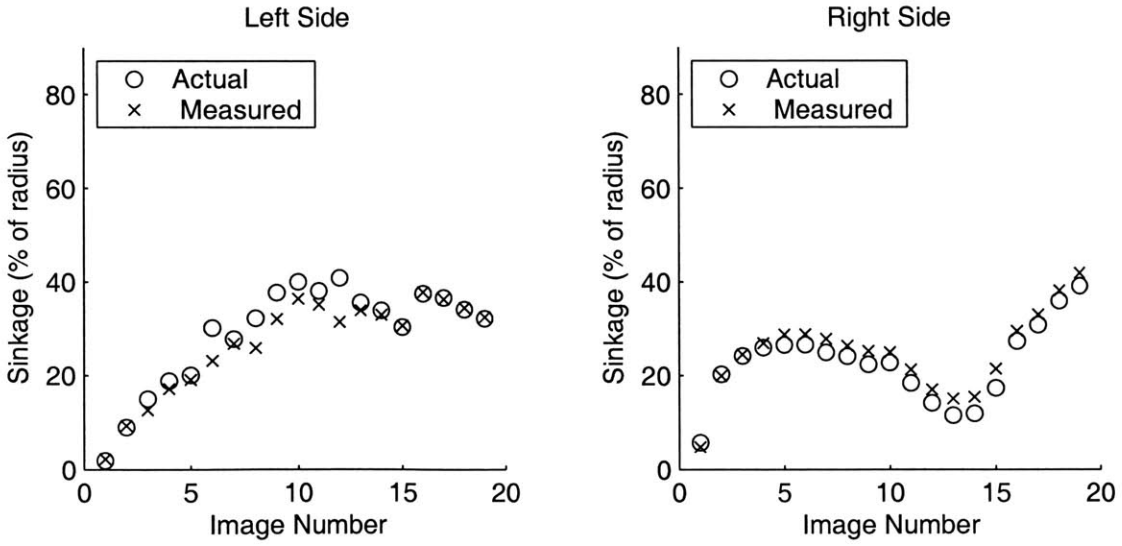


Figure E.20. Actual and visually-measured wheel sinkage, image set 10

Left Side Angle RMS Error (%)	Right Side Angle RMS Error (%)
3.60	2.51

Table E.10. Visual sinkage measurement RMS error, image set 10