HUMAN-IN-THE-LOOP NEURAL NETWORK CONTROL OF A PLANETARY ROVER ON HARSH TERRAIN

A Thesis
Presented to
The Academic Faculty

by

Mathew Joseph Livianu

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical and Computer Engineering in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2008

HUMAN-IN-THE-LOOP NEURAL NETWORK CONTROL OF A PLANETARY ROVER ON HARSH TERRAIN

Approved by:

Professor Ayanna MacCalla Howard, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Yorai Wardi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: August 2008

To my friends and family, who supported me through this process

ACKNOWLEDGEMENTS

I want to first thank my advisor, Dr. Ayanna Howard, for her insight and guidance and for providing me the opportunity to work in the HumAnS lab, all of which made this work a reality. I would also like to share my appreciation for my reading committee, Dr. Patricio Vela and Dr. Yorai Wardi, for taking the time to review my work and for their infinite patience and dedication as teachers.

The slip characterization research described here was carried out at the Jet Propulsion Laboratory, California Institute of Technology with funding from the Human Automation Systems (HumAnS) Lab, Georgia Institute of Technology. Special thanks to Daniel Helmick for the opportunity to spend a summer learning from his experience, sharing his ideas, and for being a patient advisor.

And a special thank you to my family, for supporting me emotionally and financially throughout the process. And lastly, to my friends, whose laughter, kindness, and support made a world of difference.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Wheel slip is a common problem in planetary rover exploration tasks. During the current Mars Exploration Rover (MER) mission, the Spirit rover almost became trapped on a dune because of wheel slip. As rover missions on harsh terrains expand in scope, mission success will depend not only on rover safety, but also alacrity in task completion. Speed combined with exploration of varied and difficult terrains, the risk of slip increases dramatically. We first characterize slip performance of a rover on harsh terrains by implementing a novel High Fidelity Traversability Analysis (HFTA) algorithm in order to provide slip prediction and detection capabilities to a planetary rover. The algorithm, utilizing path and energy cost functions in conjunction with simulated navigation, allows a rover to select the best path through any given terrain by predicting high slip paths. Integrated software allows the rover to then accurately follow a designated path while compensating for slippage, and reach intended goals independent of the terrain over which it is traversing. The algorithm was verified using ROAMS, a high fidelity simulation package, at 3.5x real time speed.

We propose an adaptive path following algorithm as well as a human-trained neural network to traverse multiple harsh terrains using slip as an advantage. On a near-real-time system, and at rover speeds 15 times the current average speed of the Mars Exploration

Rovers, we show that the adaptive algorithm traverses paths in less time than a standard path follower. We also train a standard back-propagation neural network, using human and path following data from a near-real-time system. The neural network demonstrates it ability to traverse new paths on multiple terrains and utilize slip to minimize time and path error.

CHAPTER 1

Introduction

Autonomous navigation on harsh terrains has been a major topic of interest in planetary exploration for many years. Since the first rover on the moon driven by astronauts, to the first Mars rover, Sojourner, to the semi-autonomous Mars Exploration Rovers (MER), which began traversing Mars rocky terrain in early 2004, we have been working to traverse more and more challenging terrains for the purpose of science and exploration. For robotic Mars missions, teleoperation of rovers is subjected to an average delay of almost 10 minutes as the signals travel [1] from Earth to Mars. For moon missions, teleoperation of rovers is subjected to a 0.5 second delay [2]. Unless the astronauts are physically present at the exploration site and trained in rover control, direct control is infeasible. Engineers must rely on rover autonomy for basic navigation tasks.

The difficulty with autonomous exploration of the unknown is that rovers encounter harsh conditions and terrains through which they must survive in order to complete mission objectives. Harsh terrains are characterized by variations in soil characteristics, obstacles, weather conditions, inclines and declines, and sharp level changes. For wheeled rovers, obstacles, such as plants, rocks, and boulders pose a danger, as the rovers are not designed

[1] The distance between Earth and Mars varies from 54.5 million to 401.3 million kilometers. That corresponds to a one-way communication delay of 3 to 22 minutes

[2] The average distance between the Earth and its Moon is 384,403 kilometers, corresponding to a one-way communication delay of 1.24 seconds

to climb over the obstacles and must detect and avoid them. Weather conditions, such as wind, sand storms, rain, and electrical and magnetic storms pose a danger to rover safety and ability to successfully complete missions. On Mars, for example, wind and sand storms have caused communication losses and power problems for the MER rovers. In addition, rovers have trouble with steep inclines and declines because they consume more instantaneous energy as well as overall power. However, for planetary rovers, one of the biggest dangers in terrain navigation is varying soil characteristics, which translates into wheel slip and sinkage.

Slip is a measure of the lack of progress of a wheeled ground robot while driving. It is dependent on the highly non-linear interaction between the wheels of the rover and the soil of the terrain. Wheel-terrain characteristics can be broken down into three main properties of soil: density, friction, and cohesion. Density is a measure of a material's weight relative to its bulk. Friction refers to the ability of two materials to resist shifting position relative to one another. Cohesion refers to the ability of particles of soil to 'stick' together, or resist separation. An interaction in which there is a low coefficient of friction, such as on damp clay, may result in the wheels shifting the soil without moving the rover. If the cohesion of a terrain is low, such as in gravel, the rover may be able to traverse without sinking (high friction and medium density), but its wheels may gain no traction and any vibrations can cause movements in the terrain. Researchers in the field of terramechanics have studied the mechanics of wheels slipping on loose soil. They have modeled the interactions and stress distributions beneath wheels [1]. Terramechanics researchers have also incorporated rover maneuvers into the wheel and vehicle model in order to better understand steering mechanics [2][3].

Many rover controllers have relied on rovers navigating on simple terrains and moving slowly on more complex terrains. One might avoid the problem of slip by limiting rover exploration to safe terrains and moving slowly. However, with rover missions increasingly

devoted to finding signs of life, it is widely accepted that many of the scientifically interesting sites are located in rough terrains with significant chance of slip. Visiting these sites, the rovers inherently must travel over rough terrains such as dry river channels, putative shorelines, and gullies emanating from canyon walls [4]. Rover slippage has been recognized to be a significant limiting factor for the MER rovers while driving on steep slopes [5], [6].

The current navigation system running on the Mars Exploration Rovers (MER) [7], [8], does not incorporate a notion of slip when evaluating or traversing paths. The MER rovers are commanded in semi-autonomous operations with humans monitoring at each step. Mobility Engineers, Rover Planners, and the rovers form a closed-loop human-robot control system. [9]. The rovers themselves traverse terrain, following commands from the rover controllers on Earth. Therefore, the task of avoiding slippery terrains currently falls upon the driver who selects the path for the rover to navigate. Often, images returned from the rovers provide good insight into whether or not a specific terrain is slippery and can be avoided by the operator. However, patches of terrain with similar texture and geometry may lead to drastically different wheel traction and sinkage. As a consequence sometimes a reasonable looking path is selected which leads the rover into a slippery terrain. During the current Mars Exploration Rover (MER) mission, Spirit lost traction in the sand at "Larry's Lookout" when it tried to ascend the 16° slope, causing a 100% slip situation (the wheels spun but no forward progress was achieved). However, on another patch of terrain of 19° only meters away, Spirit only encountered 20% slip. [9] As mission safety is the highest priority, rovers must either avoid dangerous slip conditions, or learn to handle slip in a more beneficial manner.

While rovers are novices at traversing harsh terrains, humans are no strangers to the dangers of traversing unknown terrains. In daily travels, we encounter ice, mud, rock, dirt, sand, crusty desert soil, and soft soil, just to name a few. Humans do not think of friction, density, and cohesion, but classify these terrains subjectively as slippery, squishy, sticky,

hard, loose, soft, crusty, and so on and adjust step size, walking speed, and general methods of traversal based on experiences and learned reactions. When driving on these terrains, either from inside a vehicle, or teleoperating, humans automatically adjust the vehicle's trajectory and speed to account for slip. Since human drivers are generally proficient at handling slip in real-time systems, why not transfer the knowledge of the human to a rover?

The transfer of human-driver knowledge to a rover accomplished in this work using artificial neural networks. An artificial neural network is a paradigm modeled after the way biological systems, such as the brain, process information. The neural network is composed of interconnected processing elements, called neurons, which are trained to react to certain types of inputs and provide output. Often used in classification, pattern recognition, or decision-making, neural networks are extremely adaptive and if trained correctly, considered 'experts' in the category of information they process. Mathematically, a neuron is a mathematical function, such as a sinc function, with an input and output. Given a set of data to model (training), the weights interconnecting the neurons are adjusted. Training is complete when the network, the mathematical function described by the weighted summation of neurons, produces outputs that are correct to within a given margin of error for a given validation set. The trained network is able to produce output for new data it has not seen. For example, if trained to identify circles, squares, and triangles by the angles and positions between sets of pixels, the network should be able to distinguish any circle, triangle, or square with similar accuracy.

The remainder of this paper is organized as follows: Chapter 2 discusses previous published work in the field of navigation and slip detection/avoidance on harsh terrains and human-in-the-loop controllers. Chapter 3 details the simulation work done to characterize slip on harsh terrains as well as the implementation and results of the High Fidelity Traversability Algorithm (HFTA). Chapter 4 discusses adaptive and neural network control methodologies for slip compensation and path traversal on harsh terrains. Chapter 5

discusses the hardware implementation in the Mars Yard, the near-real-time system experimental setup. Chapter 6 discusses the path traversal results from experiments with human teleoperation, adaptive carrot-following, and neural network controllers. Chapter 7 presents conclusions and discusses future work.

CHAPTER 2

Background

Due to the lack of complete information about terrain dynamics and slip characteristics, the need arises for robots to autonomously handle unforeseen slip situations. Current methods include slip-prediction classifiers, and human-in-the-loop control, but few adaptive learning solutions. While neural networks have been utilized in robotic movement, they have not yet been applied to slip control.

It is widely accepted that autonomous slip detection and compensation is a necessary but difficult part of rover navigation. Terrain deemed too hazardous to cross may be traversable in an area with different soil composition. But since soil composition can change with each patch of terrain, slip threatens planetary rover safety at every step. Second only to tip-over hazards, slip poses a great danger to mission success. [5] [6] Most intelligent rover autonomy software does not consider slip a part of terrain traversability. Since it is a highly non-linear interaction, it involves complex modeling. [10] [11] Furthermore, traditional modeling software utilizing only geometrical information cannot detect slip, which is considered a non-geometric obstacle [6].

The first step in measuring slip is to detect it while it is occurring. In simulation, wheel-soil contact models are used to determine slip characteristics of rovers traversing harsh terrains. The Rover Analysis, Modeling, and Simulation (ROAMS) tool models the terrain mechanics as well as the interactions between the wheels and the soil at each time step,

allowing planners to realistically determine slip performance of rovers in various conditions. [12] [13] [14]. The work we present in chapter 3 builds on previous slip modeling efforts, using ROAMS to simulate rover navigation on a harsh terrain and characterizing slip performance and energy consumption using the High Fidelity Traversability Algorithm (HFTA).

While simulation is a useful tool for slip characterization, rover position is not as easily accessible in real-life terrain traversals. Wheel odometry provides the number of wheel rotations as the rover traverses a terrain, but it cannot give an accurate reading of how far the robot has actually moved. Therefore, one method of detection is to detect terrain rover movement using stereo vision cameras mounted on top of the rovers. [15] [16]. A comparison of wheel odometry and visual characteristics of the terrain is then used to generate a slip vector. [17] This technique is called visual odometry and is utilized in current Mars missions. Using the computed slip trajectories, the carrot-following method can be used as a slip compensation controller to modify the rover trajectory. [18] [19]

However, detecting slip while it is occurring is not always sufficient, as in the case when Spirit became stuck on the slope. It is stipulated that using slip prediction in addition to an obstacle detection mechanism is important to navigation because an area of large slip is a non-geometric type of obstacle and cannot be detected with GESTALT, the autonomous navigation algorithm developed and used at NASA [7]. All planetary rover systems involve humans-in-the-loop providing guidance to the rovers. The MER mission employs a semi-autonomous structure in which human operators help select paths that avoid slip by utilizing imagery returned from the rovers. However, there is considerable benefit to developing the capability of detecting slip automatically when computing traversability. A slip prediction algorithm looks ahead at the terrain seen through the stereo cameras on the rover and selects the path to the goal point which presents the least amount of probable slip. Probable slip is based upon the physical and geometrical properties of the terrain [1], which can be extracted from imagery. Researchers have successfully extracted the

geometry (range data) and appearance (images) of terrains [20] [21] from stereo imagery. Combining the extracted information with a learning algorithm yields a terrain classifier. With the classifier integrated into the planning stage, the soil characteristics are determined in real-time and passed onto a slip prediction algorithm.

Slip prediction algorithms help to identify areas of potential slip for the rover, but they do not satisfy the need for real-time slip compensation. Learning systems are useful in this regime as they can utilize the expertise of a human operator in order to model non-linear slip interactions. While limited research has been performed in intelligent rover navigation, much work has been done incorporating teaching-by-showing methods.

Human-in-the-loop systems are utilized in many complex task efforts, often using teleoperation to capture data. In [22], Hidden Markov models were used to segment force data from a teleoperated gripper used for peg-in-a-hole assembly operations. In another teach-by-showing effort, video cameras were used to monitor a human teacher performing a task and recognize object relations and transitions between those relations. The data from the video segmentation was used to reproduce the actions shown by the teacher. [23]. Researchers in [24] classified the human teachers' movements into assembly states, each of which was derived from a geometric CAD model of the parts being manipulated. Human teleoperation data was converted into reusable sensor-driven programs to enable users to program a robot by teaching-by-showing.

In addition, neural networks have been used to control non-linear systems such as rigid-link robots. The DCAL-NN controller utilizes a neural network with on-line learning to teach execute position tracking on a rigid-link robot when manipulator dynamics are unknown. Using a Lyopunov approach, the researchers show that joint position error is globally asymptotically stable and NN weights remain bounded. [28] In addition, MAO [25] is a hybrid human-in-the-loop controlled 5-DOF arm which employs neural networks to generate and manage forces to help the patient guide the arm along a smooth trajectory in force or position control.

While neural networks have been successfully employed in control of rigid link robots, there is very little use of neural networks to aid in slip-controlled navigation on harsh terrains. In this work, we utilize a neural network in a novel way to learn the driving style of a human rover-driver on harsh terrains. The neural network controller learns from a human-driver, who uses slip as an advantage in order to decrease path traversal time and maintain minimal path error.

CHAPTER 3

Characterizing Slip

Prior to implementing a hardware-based real-time system, a simulation study is undertaken to characterize the slip performance of a path follower on harsh terrains and to show the advantages of slip-prediction. We propose a High Fidelity Traversability algorithm, which when integrated with a slip-compensating path follower, provides path cost and energy information that helps determine paths of lowest slip.

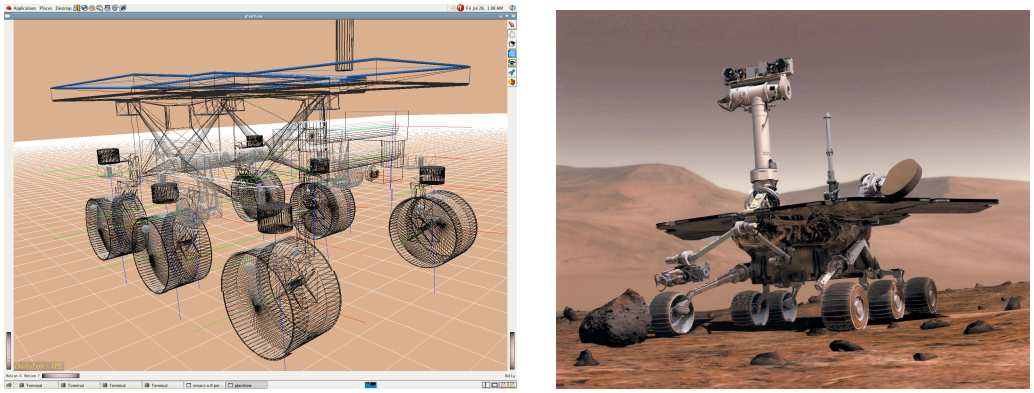
3.1 Methodology

The approach to this task begins with the development and implementation of the High Fidelity Traversability Analysis (HFTA) algorithm. The Rover Analysis, Modeling, and Simulation (ROAMS) tool, developed at NASA's Jet Propulsion Laboratory is utilized for modeling of terrains as well as wheel-soil contact models. Testing of the HFTA algorithm is first validated using ROAMS auto-generated terrains, such as hills and slopes, and then data is taken from traversals on stereo-generated terrains from images taken in the JPL Mars Yard West. Lastly, we perform a simulation feasibility study of ROAMS in order to characterize its effectiveness as a real-time slip predictor.

3.1.1 ROAMS

In order to test the HFTA algorithm in simulation, the ROAMS tool is employed. ROAMS is a physics based simulation tool for the analysis, design, development, test and operation of rovers for planetary surface exploration missions. The tool provides a modular rover simulation framework to facilitate its use by planetary exploration missions for system engineering studies, technology development, and mission operation teams. [26] [27]

3.1.1.1 Terrain

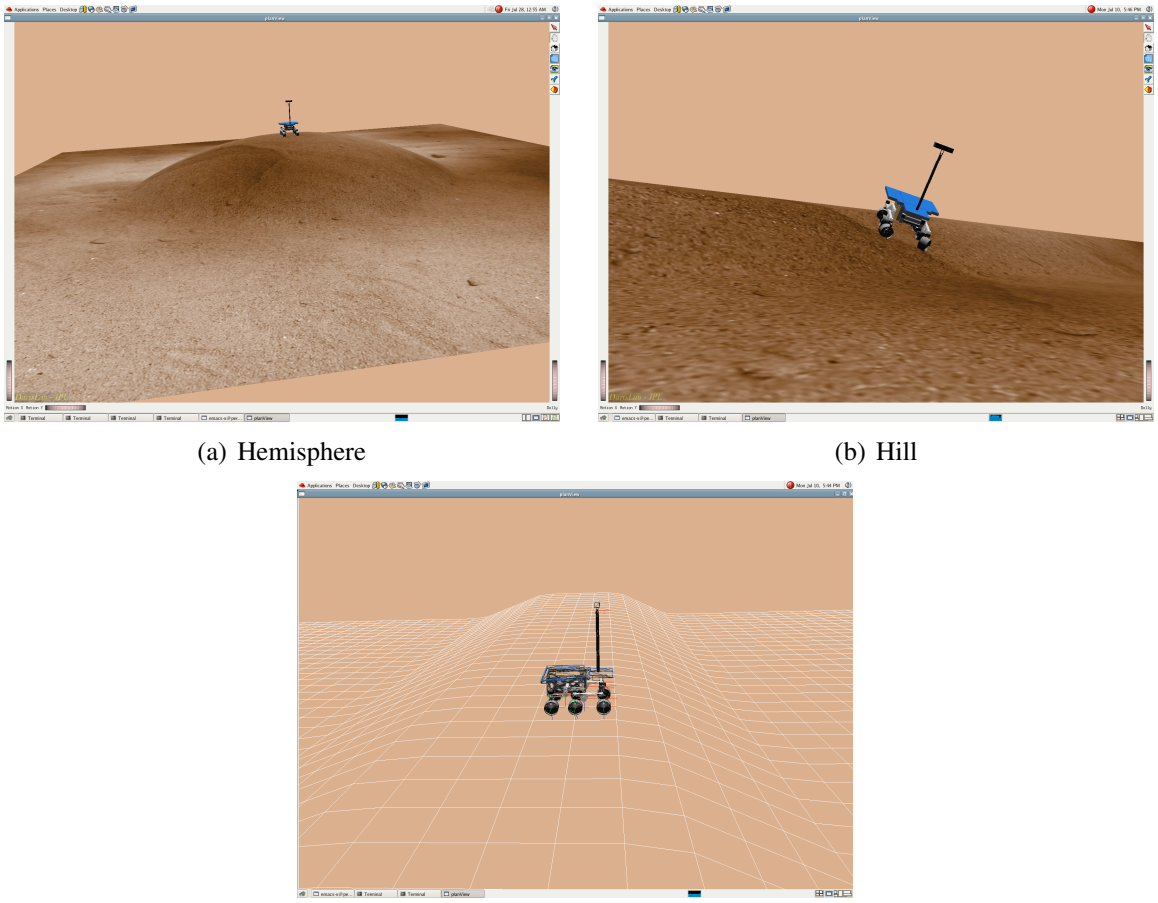


(a) ROAMS ROCKY8 (b) MER

Figure 1: The ROCKY8 rover.

ROAMS models rover-terrain dynamics accurately by defining the terrain as a mesh grid with independent soil mechanical properties for each cell in the grid. These soil mechanical properties: friction, density, and cohesion, describe the interaction between the rover's wheels and the terrain. Using the three parameters, terrain can be defined as slippery sand, hard rock, ice, or any other terrain type. ROAMS utilizes complex wheel-terrain modeling to enable realistic slipping and rolling behaviors as well as transitions between these behaviors [12][14].

We utilize various auto-generated terrains and image-generated terrains in our simulations, as shown in Figure 2. These terrains have constant soil mechanical properties



(a) Hemisphere (b) Hill

(c) Bump

Figure 2: Images from ROAMS showing the Rocky8 rover three auto-generated terrains and one stereo-generated terrain

throughout (ie. Mars sand) and are useful in testing the validity of the algorithms in canonical situations. For more advanced testing, a stereo-image generated terrain of the large Mars Yard West at the Jet Propulsion Lab in Pasadena, CA is loaded into ROAMS. The stereo-generated terrain is built using 3D image data for altitude and a texture map for color.

3.1.1.2 Rover

For simulated traversal in ROAMS, a JPL test rover called ROCKY8 is dynamically simulated. ROCKY8 is similar in design to MER, utilizing the six-wheeled rocker-bogie design. All of the wheels are independently actuated and steerable. Figure 1 shows the ROCKY8

rover as well as the rover Opportunity for comparison. Only the six wheels of the ROCKY8 are simulated dynamically in high detail in order to facilitate accurate rover-terrain dynamics. ROAMS provides dynamic wheel data such as angular velocity and torque as the simulated rover traverses the terrain.

3.1.2 The Adaptive Navigation System

Figure 3 shows the entire rover navigation system, from the image acquisition stage, to the path generation, slip prediction (HFTA), and navigation. A brief discussion will introduce the steps leading to the HFTA algorithm, which is the primary focus of this section.

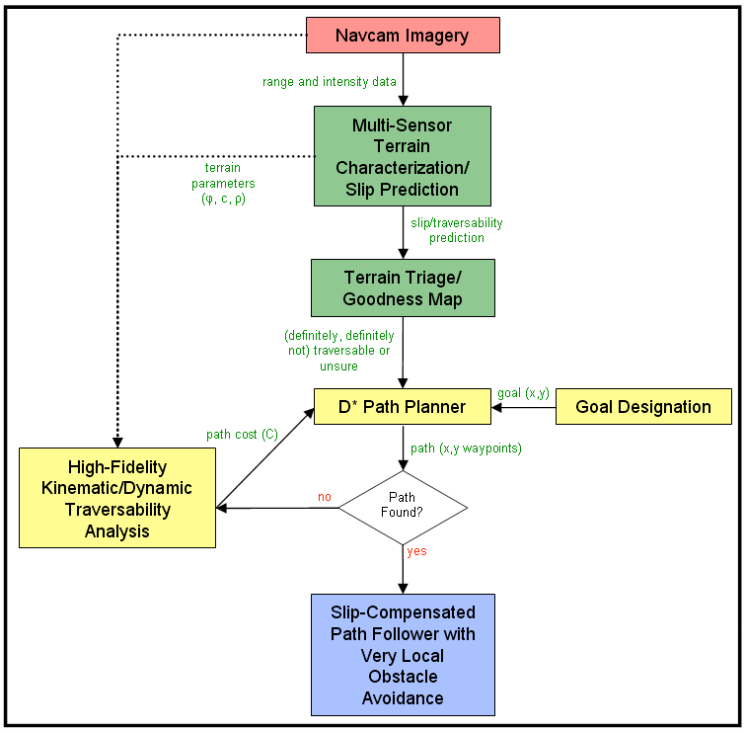


Figure 3: The Adaptive Navigation System from image acquisition to navigation.

3.1.2.1 Image Acquisition

The system begins by acquiring two images simultaneously using the two stereo cameras aboard the rover. Stereo-processing is then done to overlay the two images and create a 3 dimensional image.

3.1.2.2 Map Generation

In order to generate a map of the terrain from the stereo image data, a point cloud is first extracted from the stereo image. The point cloud is a list of the 3D Cartesian position of each pixel in the image. The point cloud is fed into a algorithm which calculates a goodness value for each pixel in the map. A goodness value takes into account each pixel's neighbors in order to ascertain the shape of the terrain at the current point. Goodness values, which range from 0 to 1, determine whether or not a region of terrain is traversable or too dangerous. Next, the goodness map is thresholded into a minimum of three regions: not-traversable, unknown, and definitely-traversable, which roughly correspond to the lower 20%, middle 60%, and upper 20% of the goodness values. The resulting values are used to construct a triage map. Figure 4 shows a triage map of the JPL Mars Yard West terrain.

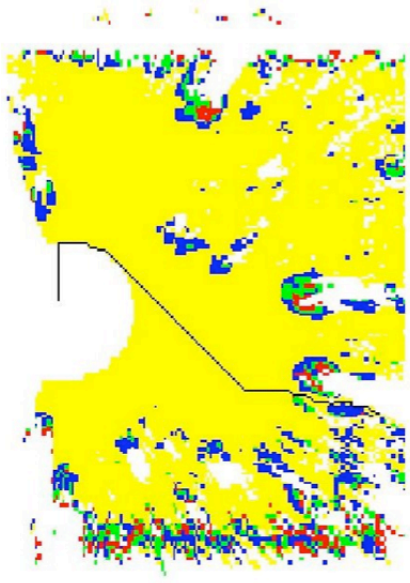


Figure 4: Triage Map created from the JPL Mars Yard West 3D image data. The D* path is overlayed on the map. The main region (through which the D* path travels) is classified as traversable and all other smaller regions are either of unknown traversability or not traversable (edges)

3.1.2.3 Path Generation and Following

Using the triage and goodness maps, a D* path is generated through the terrain from the start position to the goal position. The path is returned as a list of waypoints and from a static analysis standpoint, is ideal. Figure 4 shows a D* path overlayed on the triage map generated from 3D imagery taken in the JPL Mars Yard West.

In order to follow the path, the carrot-following algorithm is used. It draws a circle of a fixed radius around the rover's current position and computes the path error angle as the angle between the rover's current heading and the vector to the intersection point of the circle and the path. [18] This algorithm works well for rover path following at low speeds.

3.2 The HFTA Algorithm

The D* path generated from the triage and goodness maps is statically ideal, but does not account for slip, which is a dynamic phenomenon. For this reason, we implement the HFTA algorithm in order to gather the necessary information needed to compare similar paths and select the path of lowest-probability slip.

The HFTA algorithm takes, as input, a set of waypoints which describe a path through terrain with uncertain traversability characteristics. It outputs a set of waypoints which describe a path with the least probability of slip. Internally, the algorithm uses the ROAMS testbed to monitor rover parameters such as torque and angular velocity during simulation. It then computes path energy and path error for a simulated rover traveling on Mars-like terrain. During simulation, a path following algorithm utilizes a slip compensator to keep the rover as close to the prescribed path as possible.

The HFTA algorithm employs two evaluations, which it integrates over the length of the path: (1) path cost and (2) path energy. Combining the two metrics using a weighted average, the lowest overall cost corresponds to the path with the least amount of predicted slip.

3.2.1 Path cost

While a slip compensator is included with the path follower, it is not always able to remain on the given path. The path error incurred corresponds to driving errors as well as slip for which the system is unable to compensate and is therefore included in the overall cost. Path cost is computed using equation (2) below.

The point-to-line measurement searches along path segments closest to the most recently achieved waypoint.

$$\delta_t = |\vec{v_{w_1,r_t}}| * sin(cos^{-1}(\frac{\vec{v_{w_1,r_t}} \cdot \vec{v_{w_1,w_2}}}{|\vec{v_{w_1,r_t}}||\vec{v_{w_1,w_2}}|})) \quad (1)$$

$$\delta = \int_0^t |\delta_t| dt \quad (2)$$

Where w_1 and w_2 are waypoints, r_t is the rover location at time t, and $\vec{v_{a,b}}$ are vectors from point a to b. The total error δ is the integral of the errors over the path.

3.2.2 Energy cost

Since higher slip paths and terrains tend to require more energy from the system in order for the path follower to remain on course, the HFTA algorithm evaluates total energy consumption using equation 3 below.

$$\sum_{i=0}^{6}(\int_0^t (\omega_i * \tau_i) dt) \quad (3)$$

where ω_i is the angular velocity and τ_i is the torque of the i^{th} wheel. The integral computes the power consumed by each wheel and the sum of the power of all of the wheels provides the total energy cost of the rover.

The algorithm computes path energy and path error for a simulated ROCKY8 rover navigating a path described by a given set of waypoints. The HFTA algorithm also generates perturbations of the given path in order to find the path to the goal with the lowest slip probability.

3.3 Simulation and Testing

We have seen that a terrain goodness map can be thresholded to produce a triage map with three regions: traversable, non-traversable, and unknown. The HFTA algorithm is used to better characterize any section of a D* path whose traversability is uncertain.

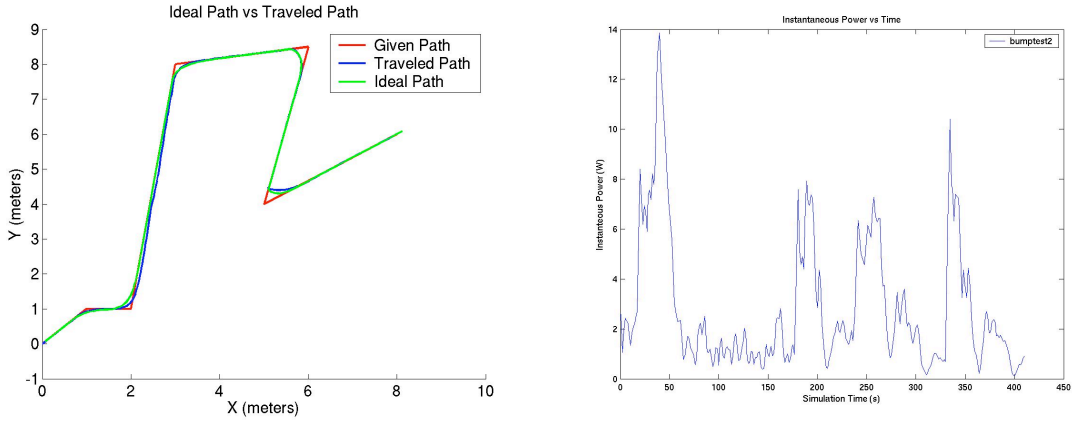
In order to test the HFTA Algorithm, various simulations are run. Keeping all other variables constant, the incline and roll angles are varied in simulation on a sloped, or hill, terrain. Maximum angles measure 20 degrees, as any greater angles cause complete loss of traction and 100% slip situations. The slope tests validate the assumption that the energy and path costs increase with added slip and path deviation. The ROCKY8 is also simulated on a bump terrain with 4 random turns. These turns represent objects the rover has to visit or obstacles it avoids. The energy plots from the bump run provide insight into the validity of the energy cost function. Lastly, the rover is simulated traversing a hemisphere along four different paths: climbing to the peak and down, straight along the curve, driving around a less steep section of the hemisphere, and completely avoiding the hemisphere. The goal is to determine whether or not the algorithm can predicts a safer course of action (ie. less slip) than a human rover driver might otherwise take simply looking at the terrain from imagery.

In addition to the rover simulations, timing tests are performed on the ROAMS software in order to determine its effectiveness as a slip-predictor in real-time. Our intention is to place the simulator and HFTA algorithm on-board a rover and use it to simulate multiple paths in real-time before autonomously deciding the safest path to take. Due to the time cost associated with simulation, the software needs to run faster than real-time in order to be an effective alternative to human drivers.

3.4 Results

In the sloped terrain tests, results indicate high resolution in predicting terrain difficulty, with a range of 1 Joule to 250 Joules per meter for the ROCKY8 rover traversing flat or

inclined/angled terrain.



(a) Ideal and Traveled Paths

(b) Instantaneous Power Plot

Figure 5: (a) The D* path (given path) waypoints are interpolated (ideal path) to smooth the path and the rover follows the path in ROAMS (traveled path). Note that the traveled path deviates slightly from the ideal path because of slip (b) Note that the instantaneous power plot associated with the path traveled on the left exhibits spikes during each of the turns corresponding to rover accelerations.

In the bump test, energy plots showed some anomalies but generally behave as expected, with increased energy during acceleration and deceleration and in high torque situations. Figure 5 illustrates a path followed in the ROAMS simulator and its associated instantaneous power plot. Note that the power spikes when the rover turns, which is characteristic of acceleration. However, due to the design of the path follower, the rover experienced slightly more acceleration and deceleration than necessary to execute each turn. These sudden changes in speed translate into instantaneous energy spikes in the computation of path cost.

In the hemisphere test, the algorithm showed some promising results. Table 1 shows results for the simulation depicted in Figure 6. Results indicate that ascending and descending a hemispherical hill consumes more power than simply traversing flat terrain, but that both paths incur similar path error. In addition, the most direct path has both a high energy cost and a high path cost, which makes it an unlikely candidate.

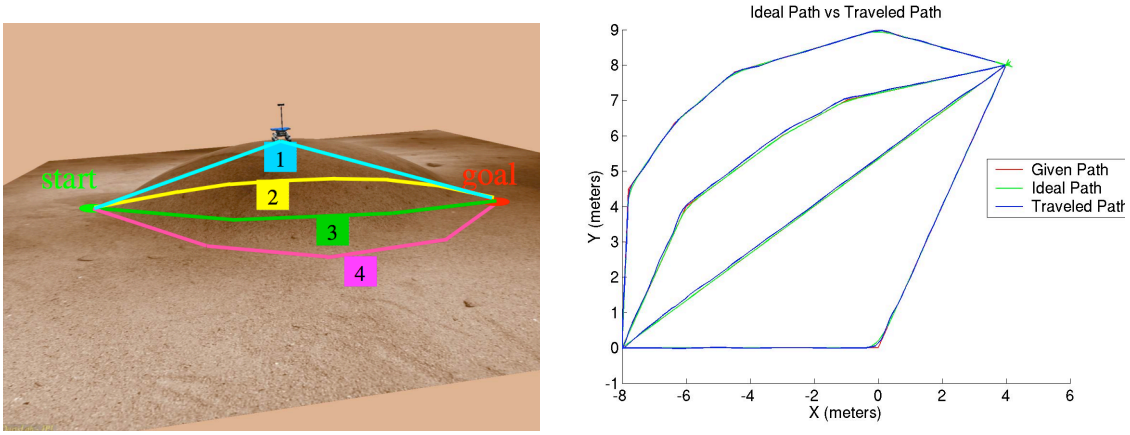


Figure 6: Four paths around a hemispherical bump (darker shading indicates higher altitude). Path 1 climbs to the peak (2 meters) perpendicular to the slope and descends similarly. Path 4 avoids the bump entirely. Paths 2 and 3 traverse the bump with varying degrees of slope.

While this test does not represent all cases of potential hazard to a rover, it does demonstrate that the algorithm identifies the path which entirely avoids the hill as the lowest cost path, even though it is longer. The straight path might have been the obvious path to select from the perspective of a human driver, but the path and energy costs are much higher.

The paths in the hemisphere test were designed to illustrate the operation of the algorithm and were not generated by D*. Paths 1 and 4 show the extreme cases of climbing straight up and down the hill and navigating around the hill respectively.

Table 1: Path and Energy cost for HFTA simulation shown in Figure 6

Path	Energy (J)	Path Error (m-s)
1	5101.46	1.91
2	3324.51	12.32
3	1587.44	12.79
4	938.52	2.21

3.4.1 Real-time slip prediction

While timing is currently of secondary importance compared to safety in a planetary rover mission, it is important to execute movements in a timely fashion so as not to impede the

intended research goals. Future missions will also require higher rover speeds and lower execution times. As such, the ROAMS simulations on board the rover need to be executed faster than real-time for the system to be an effective alternative to human driver control. We performed a test to ascertain the simulation speed of ROAMS.

ROAMS is commanded through a message-passing system. The simulation is advanced in software by transmitting a step-simulation message. There are two clocks, a real-time clock and a simulated clock. For each simulation-step, ROAMS is told to advance the simulation clock by a specified number of seconds. For example, with a simulation step rate of 40Hz and step size of 0.1 seconds, 40 times per second (real time), the simulation advances by 0.1 seconds (simulated time). In other words, for every second of real time in the system, the simulation progresses by 4 seconds, achieving a 4x real-time rate.

In simulation, many messages commanding rover velocity and position and retrieving rover data are sent every second and ROAMS is limited by the rate at which it can process messages. A benchmark test was conducted to find the fastest simulation rate possible. Table 2 summarizes the results. The best rate achieved for the ROCKY8 rover traversing various sloped terrains was 3.5x real time.

Table 2: ROAMS real-time benchmarking data

Type	Slope (deg)		Energy (J)	System Time	ACE Time	Real-Freq
	X	Y		(s)	(s)	(Hz)
UP	0	-20	139.547	30.6	106.9	3.49x
UP	0	-10	61.783	26.2	92.5	3.53x
EVEN	0	0	1.324	24.9	87.9	3.53x
DOWN	0	10	54.237	21.4	84.7	3.95x
DOWN	0	20	99.403	19.9	77.9	3.91x
ANGLE	5	0	5.595	25.5	88.5	3.47x
ANGLE	10	0	9.769	25.3	88.6	3.50x
ANGLE	15	0	12.433	25.4	89.8	3.54x
ANGLE	20	0	17.572	25.9	90.2	3.49x
UP/ANGLE	20	-10	85.149	26.6	100.4	3.78x
UP/ANGLE	20	-20	353.041	54.5	192.4	3.53x

Since time for planning between movements should take seconds and not tens of minutes, ROAMS processing speed would have to be approximately 50 to 100 times faster in order to be an effective real-time slip-prediction tool.

Due to the current infeasibility of ROAMS as a real-time path analyzer, full integration of the slip predictor, path planner, and traversability analyzer is left as future work. However, utilizing the simulation results from the HFTA algorithm as a starting point, we propose alternative control methods for slip compensation on harsh terrains.

CHAPTER 4

Control for Slip Compensation

Thus far, we have discussed simulation-based slip-prediction methods for rovers. As the slip simulations show, using the HFTA algorithm to analyze path error and energy cost can help characterize slip performance and select a low-energy path. In addition, the carrot-following algorithm is effective at following a path on harsh terrains so long as slip is not excessive and the speed of the rover is low.

In current missions, rover safety is a primary concern when guiding rovers in dangerous environments. In path traversal on harsh terrains where obstacles, cliffs, and other hazards exist, this correlates closely to keeping as close to the planned path as possible.

However, in future exploration missions, time will become more of a critical factor. In order to minimize mission time, rovers will travel at higher speeds and on multiple terrains in a single mission. These mission changes inevitably lead to a greater probability of wheel slip while traversing these harsh terrains. In order to minimize mission time while also closely following the path, the need arises to either further compensate for slip or utilize it in the favor of the rover in order to maintain path. In the simulation study from chapter 3, safety was maintained by minimizing path error and slip at low speeds. However, at higher speeds, new methods of path traversal are crucial to success.

Using slip to the advantage of the rover enables the traversal of harsh terrains while minimizing time and maximizing safety. We first determine the parameters necessary to

characterize slip in a real-time system. In order to better utilize slip in traversing harsh terrains, we present the adaptive carrot-following algorithm. We also teach a rover the knowledge of a human-expert driver driving over multiple harsh terrains with varying slip characteristics with the use of a neural network.

4.1 Slip and path error detection

Effectively utilizing slip in rover navigation on harsh terrains requires detection of slip and path traversal performance. During navigation, a path error magnitude ($|\vec{\delta_t}|$), path error direction ($\vec{delta_t}$), slip magnitude ($|\sigma_t|$), and slip direction ($\vec{\sigma_t}$) are computed at set intervals. For path error, see equation 2.

In order to compute the slip vector of the rover, we compare the rover's movement in the world frame to the rover's assumed position using on-board dead-reckoning. Dead reckoning on the rover is computed utilizing left and right wheel rotations as well as the wheel radii. World frame movement is computed as the vector from the previous known position to the current position. For each slip computation (equation 4), the vectors of travel have the same starting location, which is assumed to be the origin.

$$\vec{\sigma_t} = \vec{p_t} - \vec{p_r} \tag{4}$$

where $\vec{p_r}$ and $\vec{p_t}$ are the rover-estimated and world-frame vectors of travel respectively.

4.2 Adaptive Carrot-Following algorithm

The carrot-following algorithm, introduced in chapter 3, finds the first forward-looking intersection between a circle of fixed radius centered on the rover's world position and the path. Since the path is in discretized into waypoints, we first find the intersection of the line intersecting the two waypoints and the circle and then determine if that point is in the segment between the waypoints.

$$p_{intersect} = \begin{bmatrix} \frac{-mb+\sqrt{r^2(m^2+1)-b^2}}{m^2+1} \\ mx+b \end{bmatrix} \tag{5}$$

where m is the slope of the line segment between two consecutive waypoints, b is the segment's intersection with the y-axis in the terrain, and r is the radius of the circle about the rover's position.

The difference in orientation and position between the current position of the rover, r_t and the intersection of the circle with the path are the error angle ϕ and error magnitude ϵ.

The adaptive version of the algorithm changes the radius of the circle based on speed of the rover.

$$r_t = \alpha * |\vec{r_t}| \tag{6}$$

where $\vec{r_t}$ is the velocity vector of the rover at time t and α is a scaling factor.

In the case of a driver or controller desiring to slip around a corner rather than slow on approach, a greater average rover speed is needed in the turn. Utilizing adaptive carrot-following, as the rover approaches a turn at high speed, the rover will enter the turn sooner. Since the rover is moving faster, the controller expects the rover to lose traction with the surface, and 'drift' around the curve. Properly controlled, the rover maintains speed around a turn while keeping on the path.

The tuning of the adaptive carrot-following algorithm is highly dependent on terrain type. In addition, the rover incurs instability when it loses traction with the terrain. However, if tuned correctly, the algorithm allows for a low-error path traversal while minimizing traversal time. See the results in section 6 for performance of this algorithm.

4.3 Capturing Human Knowledge

Humans are experienced at using slip as an advantage when navigating a path. For example, drift racers utilize controlled side slip around turns in order to maintain high velocities and

stay on the road. However, with planetary rovers, the human is often not sitting in the rover itself, but teleoperating it either from close by, or from a distance out of sight. If a human can learn to teleoperate a rover over harsh terrains, utilizing slip to minimize both mission time and maximize mission safety, we propose to transfer the human's rover controlling ability and slip adaptation to the rover.

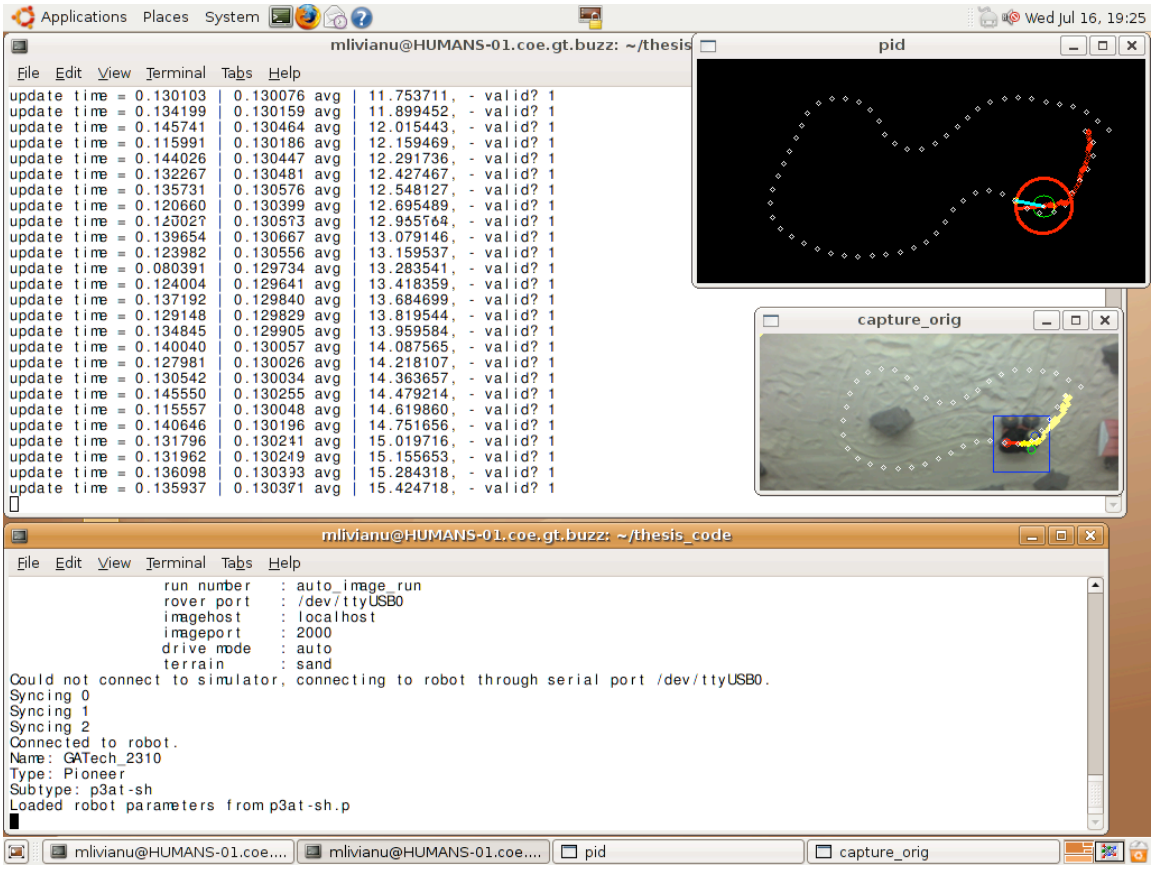


Figure 7: Screenshot of the user interface for the near-real-time system. Shows the total run time and image update rate (upper left), the world-frame image with the path and rover position (upper right), the overhead image with the rover (center right), and the connection information (bottom).

In order to capture human driving methods, we build a near-real-time system in which a human controls a rover remotely and receives path and slip error data visually. The system records the slip and path error parameters as well as the human responses to the data. In order to evaluate the effectiveness of the system, we characterize driver learning rates and driving performance. Results are presented in chapter 5.2.3.1.

4.4 Slip Control Neural Network

Alone, the adaptive carrot-following algorithm is not sufficient for guiding path traversal with slip on multiple harsh terrains because of the need for tuning. Using the human-driver data obtained from the near-real-time system, we employ a neural network in a novel way to learn slip characteristics and driving methodology on harsh terrains.

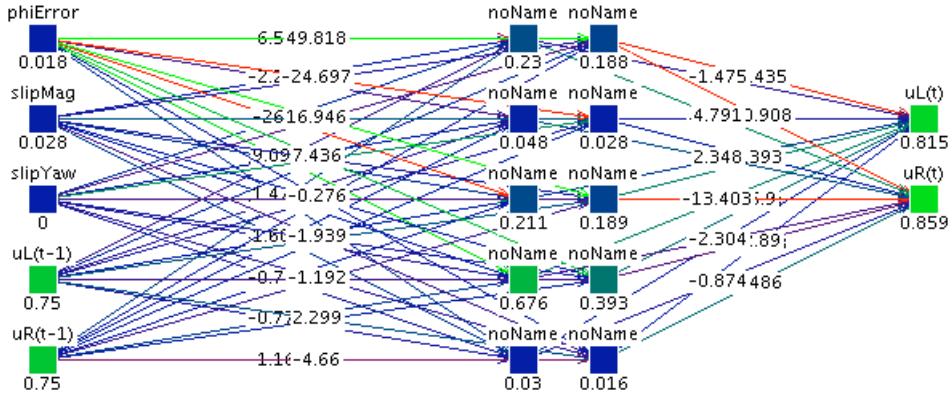


Figure 8: Back-propagation Artificial Neural network with 5 inputs, 1 hidden layer with 10 nodes, and 2 outputs

The image above shows the neural network used to control the rover. The network has five inputs: path error direction, $\vec{\delta_t}$ (labeled 'phiError'), slip magnitude, $|\sigma_t|$ (labeled 'slipMag'), slip direction, $\vec{\sigma_t}$ (labeled 'slipYaw'), current left wheel velocity (labeled: 'uL(t-1)'), and current right wheel velocity (labeled: 'uR(t-1)'). The network outputs two values: left wheel velocity (labeled: 'uL(t)') and right wheel velocity (labeled: 'uR(t)').

In order to train the neural network, we gather real data by teleoperating the rover on various terrains on varying paths. We also drive the rover using the adaptive-carrot following algorithm and the original carrot following algorithm for comparison. Training is achieved using the standard back-propagation method with values of 0.2 for ν and 0.1 for d_{max}

By using slip information in the computation of velocity, the neural network captures the driving methods of the human driver. If the training set had no slip information, the controller would be blind to the rover slipping on the terrain. With the inclusion of slip,

the terrain interactions with the rover as well as the human driving method are taken into account when traversing the harsh terrain.

CHAPTER 5

Implementation of a Near-Real-Time System

5.1 The Mars Yard Environment

A small-scale Mars Yard, located in the basement of the Human Automation Systems Lab, provides an excellent rover testing environment. approximately 20 feet by 10 feet in size and elevated 1 foot, the yard is large enough for the Pioneer rover to navigate between obstacles. Simulated terrains include beach sand, which fills the yard, wooden boards, which are placed on top of the sand to create a more dense surface, and sand-covered wooden boards, which simulate a dense, but slippery surface. In addition, large plastic boulders strategically placed throughout the yard provide obstacles for the rover. Together, the sand, the boards, and the sand-covered boards along with boulders provide three different rough terrain types for navigation testing.

The Pioneer 3-AT rover is used for all navigational testing discussed in this work. It was

Figure 9: The Pioneer 3-AT, by ActiveMedia Robotics

selected for its size, torque capabilities, and its skid-steered control, which guarantees slip in turns. The rugged P3-AT 50cm x 49cm x 26cm aluminum body with 21.5cm diameter drive wheels loves to run outdoors. The four motors use 38.3:1 gear ratios and contain 100-tick encoders. This skid-steer platform is holonomic and can rotate in place moving both wheels, or it can move wheels on one side only to form a circle of 40cm radius. P3-AT can climb a 45% grade and sills of 9cm. The P3-AT has a maximum velocity of 0.7m/s and a maximum rotational velocity of 180 degrees per second. Control is achieved through software from the vendor called ARCOS. It provides C++ functions for sending and receiving data from the Pioneer 3AT via serial communication.

In order to achieve location and pose estimation via overhead camera, three large paper circles of known colors are placed on top of the rover in an isosceles triangle. The method for localizing is discussed in section 5.2.2.

Figure 10: The 1.3 Megapixel Logitech Quickcam Fusion

In order to localize and obtain a pose estimate for the rover in the Mars Yard, a 1.3 Megapixel Logitech Quickcam Fusion web cam is placed 20 feet above the yard looking down. The camera was selected because of its wide-angle view, simple installation, compatibility with Linux, and high resolution capability. The camera sends images to a desktop computer via a long cable and a usb connection.

A 2.4 GHz dual-core Dell MX460 with 2GB RAM and an 80GB hard drive running Ubuntu Linux 6.10 is used for all vision, neural network, and controls processing. The Pioneer 3AT on-board computer receives command serially and performed low level motor control.

A joystick allows the user to command rover wheel velocities. User teleoperation is discussed further in section 5.2.3.1.

5.2 Near-Real-Time System

The scope of this work does not require the use of visual odometry for localization during navigation. The near-real-time system developed for the experiments described in this work is utilized to localize the rover and compute slip and path error.

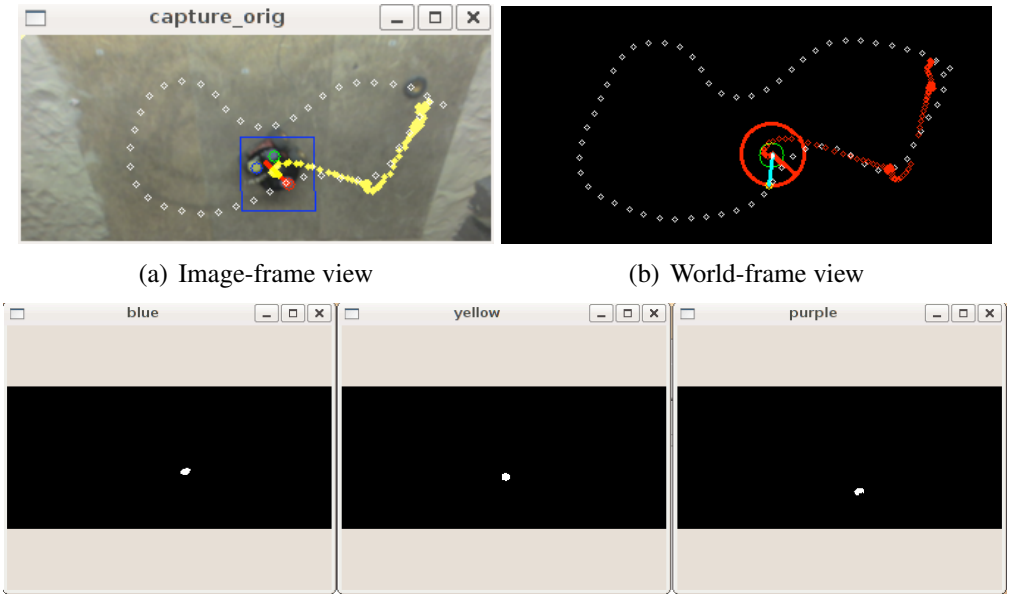
Obstacles are placed strategically throughout the Mars Yard such that the rover can navigate in between them to traverse the terrain. The rover is placed on one end of the Mars Yard with three differently colored dots in a triangle on top. A path is generated using sparse user-selected waypoints. The overhead camera takes images at 10Hz and sends them via USB to the command computer, which processes them to find the position of the three colored dots on the rover, and extract the orientation of the rover. The position and orientation of the rover are overlayed on a live image displayed to the user. The path error vector is computed using the carrot-following algorithm and optionally displayed to the user along with the generated path, which is overlayed on the image or displayed in its own image. In addition, all previous positions are also optionally displayed on the map to indicate to the user where the rover has been.

The system updates rover position and orientation at approximately 7-8Hz. In closed-loop control, the system computes wheel velocity commands at 7-8Hz, delayed in real-time by 130ms. The velocity commands sent to the rover are read in an independent loop that operates at 10Hz, adding up to 100ms of delay. Thus, from image capture to robot response, there delay ranges from 130ms to 230ms. In human-teleoperated mode, the user sees images that are 130ms delayed in time, but further delay is dependent upon reaction time of the individual user. System loop timing is asynchronous because of wide variation in image capture and processing delays, but the slip computation accounts for timing of each cycle.

5.2.1 Path Generation

Prior to starting the image server, the user specifies a list of waypoints in the Mars Yard to create a virtual path for the rover to follow. When the image server is instantiated, the system searches for the robot. Once it obtains an initial position and orientation, the path generator prepends it to the user-selected waypoints. The path generator then follows the path using the carrot-following algorithm from one waypoint to the next, adding intermediate points, until it reaches the last waypoint. By utilizing a high velocity and small time step, the user can interpolate points closer together, maintaining sharper corners. Conversely, utilizing slower velocities and larger time steps, the user can spread the points out further, smoothing corners. The smoother the path, the easier it is for the robot to follow with greater accuracy at higher speeds. The sharper the edges, the more carefully the rover has to maneuver to maintain minimal error.

5.2.2 Vision processing



(a) Image-frame view

(b) World-frame view

(c) Segmented colors

Figure 11: The color locators in the image-view (a) are found through segmentation of blue, yellow, and purple (c). The positions of these locators are utilized to compute the rover position in the image-frame. That position is transformed to locate the rover in the world-frame coordinates (b)

The camera sends 320x240 images at 8Hz to the Dell desktop computer via a long cable and a usb connection. Higher resolutions of 640x480 are also used, but at the cost of higher latency. Packet loss and corruption interrupts the signal occasionally, contributing to localization errors. This is further discussed in section 5.3).

The vision server processes each image three times, looking for each of the three colored dots on the rover. Once it finds the center of each dot, it uses those values to compute the center and orientation of the triangle. Localization errors in color detection of each dot contribute to errors in the detection of position and orientation of the robot. For more details on vision processing algorithms used in this work, see Appendix A.

5.2.3 Controlling the Pioneer Rover

As the rover traverses the path, either via human or neural network control, a path error magnitude(δ_t), slip magnitude($|\sigma_t|$), and slip direction($\vec{\sigma_\phi}$) are computed looking back n dt (we call this the 'slip-dt'), where n is dependent on cycle time. These variables are discussed in more detail in section 4. Because the computation of slip and path error is complicated by image localization errors, by ensuring sufficient backwards information, we ensure that the slip magnitude lies outside the localization margin of error (see section 5.3).

Using the slip direction and magnitude as well as the path error, the user via joystick or a closed-loop controller, can update the rover left and right wheel velocities.

5.2.3.1 Control via Human Teleoperation

In human-mode or training-mode, the user's objective is to teleoperate the robot, guiding it along a pre-defined path in the Mars Yard. The user is is shown two images: 1. An overhead image of the terrain and path, and 2. a processed world-frame image of the path over a black background. Both images show the current position and orientation of the rover. Image 1 shows the post-processed locations of the three colored dots on the rover as well as a box centered on the rover to help the user quickly identify its position. Image 2

shows the path in world-frame coordinates. It also displays a small red circle representing the rover position and a larger red circle representing the carrot-following radius. As the user commands the rover, the image shows the active slip direction and magnitude. The slip is reset every slip-dt (approximately 0.5 seconds), so that the vector remains relevant to the user as they use the joystick to direct motion. Image 2 also leaves a small red dot every slip-dt at each position the rover has visited to show the user their progress in following the path. See figure 7 for a the teleoperators view of the system.

The user controls the rover wheel velocities via a joystick. Depressing the 'fire' button with the index finger, the user enables the joystick. Moving the joystick left and right command the rover to turn the wheels in opposite directions, which turns the rover in place. Moving the joystick forward and backwards commands the rover to turn the wheels with equal positive and negative speeds respectively. Moving the joystick left or right as well as forwards and backwards moves the rover along a curved path by commanding different left and right wheel speeds.

Due to the delays in the system and the ARIA software on the rover, human motions of the joystick do not affect changes in the wheel velocities immediately. In order to effectively drive the rover, the human must adapt to these delays. Human teleoperation results are discussed in section 6.3.

5.2.3.2 Control via human-trained neural network

In intelligent path-following mode, slip magnitude, direction, path error, and current left and right wheel velocities are computed at each slip-dt. Computation of these values as well as a neural network structure is discussed in section 4. The neural network, trained with data from human-teleoperation runs, receives these values as input and outputs left and right wheel velocities in order to keep the rover on the path.

The Java Neural Network Simulator (JavaNNS) is utilized for neural network implementation and training. JavaNNS is a simulator for neural networks developed at the

Wilhelm-Schickard-Institute for Computer Science (WSI) in Tbingen, Germany. It is based on the Stuttgart Neural Network Simulator (SNNS) 4.2 kernel. It is selected because of its ability to generate C functions for any trained network as well as its feature-rich interface.

5.3 Challenges

Several factors contribute to localization uncertainty in the near-real-time system. Due to the time delay between image acquisition at the camera and processing at the image server, users must adjust their driving methods to compensate. In addition, the delay affects the ability of the controllers to react to disturbances quickly. However, this controller delay is similar to that which might be experienced by a rover on Mars using visual odometry to determine position and orientation.

Uncertainty in localization is also introduced because of occasional image corruption from undetectable UDP packet loss during transfer. Color segmentation variations also contribute to image-frame position errors, reducing the accuracy of rover localization. In addition, translating the position from the image frame to the world frame contributes to the localization error due to the slight non-linearity of image coordinates caused by the trapezoidal transform and the slight fish-eye distortion.

Overall, the system performs well enough to localize the rover at 8Hz with an accuracy of about two centimeters. The system compensates by accumulating slip and path error data aver several time steps in order to obtain values well outside the margin of error.

CHAPTER 6

Testing and Results

6.1 Experiment Overview

In order to show the effectiveness of the adaptive carrot following algorithm and demonstrate that a human driver can teach a neural network to traverse multiple harsh terrains, we run several experiments using our near-real-time system.

To begin experimentation, we designate three paths in the Mars Yard. These paths are generated by selecting points of interest and then using the path generator to interpolate waypoints in between, thus creating a curved path with maneuverable transitions.

We first teleoperate the rover, training five drivers and categorizing their performance in order to improve the control and user interface. We then control the rover along paths in the Mars Yard using the carrot following algorithm and the adaptive carrot following algorithm and compare their performance. Next, a single driver trains on the system in order to develop the ability to teleoperate the rover to slip around turns and minimize both time and path error. The human driver guides the rover over several terrains and paths to gather slip-control data for the neural network. Lastly, we train several neural networks using the human driver data as well as the adaptive-carrot following data and compare their performance

We use a wood and a sand terrain for maximum disparity and train on three separate paths: a circle path, a path that weaves in a clockwise direction, and a path that weaves in a

counter-clockwise direction. Each path begins and ends in the same area such that the path made a complete circle. The rover does not always start at the exactly the same place each run, but final metric data is normalized for path distance.

6.2 Experiment Metrics

The goal of the human drivers, the carrot-following algorithms, and the trained neural networks is to control the rover in such a way as to maximize two primary metrics in following paths in the Mars Yard: (1) maximize average speed (v_{avg}) and (2) minimize average path error (δ_{avg}).

Average speed, v_{avg}, is computed by dividing the total path length by the total time. Total time, τ, is measured from the moment the user starts commanding rover motion (t_0), to the moment the rover reaches the final waypoint in the path (t_f). Path length is the sum of the distance between interpolated waypoints, as shown in equation 7 below.

$$v_{avg} = \frac{\sum_{i=0}^{n} |w_{i+1} - w_i|}{\tau} \tag{7}$$

where w_i is the i^{th} waypoint of n total waypoints.

Average path error, δ_{avg} is computed by dividing the total path error, δ by the run time, τ. The total path error, δ (equation 2), is measured at set time steps along the path and is computed as the magnitude of the shortest distance between the rover's position and the path.

Another metric of note is average slip (σ_{avg}), which inversely correlates to average speed, but also indicates the quality of the connection between the rover wheels and the terrain. In addition, total slip (σ) relates to total energy usage, as discussed in chapter 3. The less slip incurred over the entire path, the more energy efficient the path traversal.

6.3 Human-Rover Teleoperation Results

6.3.1 Human Driver Training

Each rover driver is instructed not to watch the rover, but to utilize the world-frame image on the teleoperation screen, which shows the path, the rover position and orientation, path error, and slip. Drivers are also instructed to first learn to follow a path, and then to try to minimize traversal time.

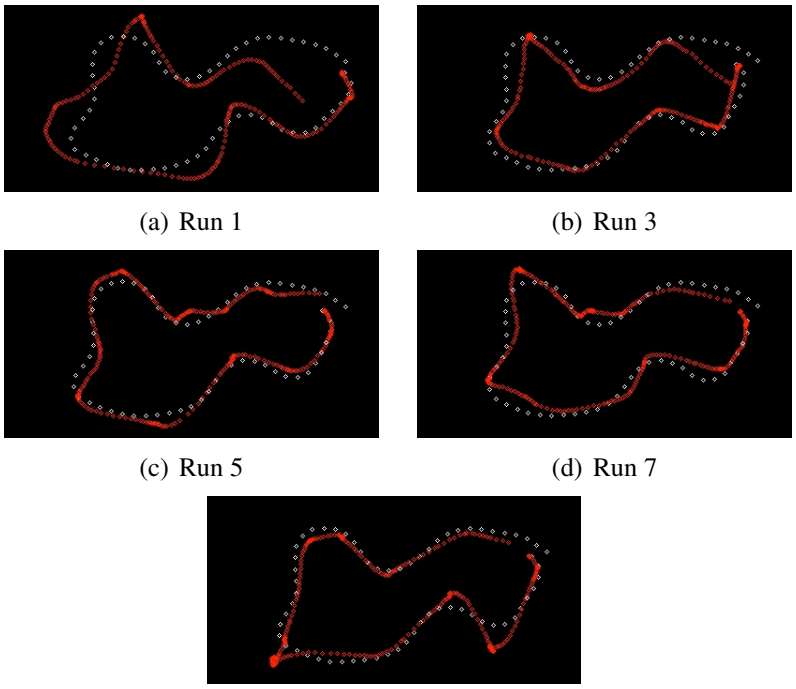
Training multiple drivers reveal several important experiment considerations. The delays associated with teleoperation require the human to only to wait for commands to affect the rover, but also to pre-emptively act in order to minimize path error. The adaptive-carrot following algorithm is a direct result of this realization. In addition, most drivers have trouble learning to slip effectively. Drivers manage to learn basic rover control and improve traversal time over several consecutive runs. However, none of the drivers manage to effectively use lateral slip when navigating turns during the training period. This leads to the conclusion that an expert driver is necessary in order to effectively use slip in traversing harsh terrains.

Figure 12 below shows the progression of one driver learning to teleoperate the rover over several runs.

6.3.2 Expert Driver Results

The expert driver more effectively employ lateral slip to minimize traversal time and path error on the wood terrain than on the sand terrain.

Recall that the goal of the experiment is to maximize the effectiveness of slip in traversing a path in minimum time and minimum path deviation; This involves utilizing wheel slip in conjunction with lateral motion in order to reduce time maneuvering at higher speeds. We still strive to minimize stationary or near-stationary slipping. Thus, on terrains with characteristics similar to sand, we expect much more stationary slipping than on a terrain with properties like wood.



(a) Run 1 (b) Run 3

(c) Run 5 (d) Run 7

(e) Run 9

Figure 12: Series of path following images showing progressive improvement for a new driver

Table 3: Results from teleoperated traversal with expert-driver

Terrain	Path	Avg Speed (cm/s)	Avg Path Error (cm)	Avg Slip (cm)	Total Path Error (cm-s)	Total Slip (cm-s)
Wood	Circle	37.44	6.03	3.04	110	54.2
Wood	Curvy CW	25.77	8.82	2.64	263.82	79.35
Sand	Curvy CW	17.69	4.46	4.37	199.38	193.81

Table 6.3.2 shows the results of the expert driver traversing the sand and wood terrains. While the path error is greater on the wood terrain, the average slip is twice as large on the sand terrain. In addition, since the average speed is lower on the sand terrain, the benchmark energy usage, shown by slip, is more than twice as high as on the wood terrain when traversing the same path. As the results indicate, the wood terrain allows the expert driver to more effectively minimize time and path error than the sand terrain given the rover's skid-steering configuration. In addition, the rover's handling ability on wood allows for more lateral slip than on sand, which greatly contributes to the expert driver's success

on wood.

The expert driver data is utilized to train the neural network and is discussed further in section 6.5

6.4 Adaptive Carrot Controller Results

The new adaptive carrot-following algorithm shows improved performance over the regular carrot-following algorithm on both terrains in a comparison of average speed, average path error, and average slip.

Table 4: Teleoperated vs Carrot-following path traversal on sand and wood

Control	Terrain	Path	Avg Speed (cm/s)	Avg Path Error (cm)	Avg Slip (cm)
Teleoperated	sand	Curvy CW	17.69	4.46	4.37
Carrot-Following	sand	Curvy CW	12.55	2.57	2.17
Adaptive Carrot	sand	Curvy CW	13.29	1.85	2.17
Carrot-Following	sand	Curvy CCW	15.15	8.06	3.28
Adaptive Carrot	sand	Curvy CCW	19.86	4.64	2.55
Teleoperated	Wood	Curvy CW	25.77	8.82	2.64
Carrot-following	Wood	Curvy CW	18.51	3.86	2.56
Adaptive Carrot	Wood	Curvy CW	19.97	3.14	2.49
Carrot-following	Wood	Curvy CCW	19.01	8.8	4.05
Adaptive Carrot	Wood	Curvy CCW	24.48	3.22	2.77

Table 6.4 shows the averaged results from data collected during experiments with the carrot-following and adaptive carrot-following algorithms. Each experiment is run at least 10 times and then averaged. The maximum average speed is approximately 0.25 meters per second. Speeds on wood are generally higher because of the increased traction between the rover and the terrain, reducing stationary slipping.

Table 6.4 shows the percentage improvements of the adaptive carrot-following algorithm over the regular version. On sand, the adaptive algorithm improves by 6% to 31% in average speed while lowering average path error by 28% to 42% and average slip by 0% to 22%. On wood, the adaptive algorithm increases by 8% to 29% in average speed while decreasing path error by 19% to 63% and slip by 3% to 32%.

Table 5: Adaptive Carrot-Following vs Carrot-Following performance on sand and wood

Terrain	Path	Avg Speed	Avg Path Error	Avg Slip
sand	Curvy CW	+6%	-28%	0%
sand	Curvy CCW	+31%	-42%	-22%
wood	Curvy CW	+8%	-19%	-3%
wood	Curvy CCW	+29%	-63%	-32%

The disparity in improvement between the CW and CCW paths arises from the fact that the adaptive carrot following algorithm excels in high speed situations, where a turn is better anticipated and overshoot is reduced. The clockwise path had more gradual turns while the counter-clockwise path contained more straight sections in which the rover could pick up speed.

It is also important to note that the regular carrot-following algorithm on wood jeopardizes the safety of the rover because of its tendency to overshoot turns at high speeds.

6.5 NN-based Slip Controller Results

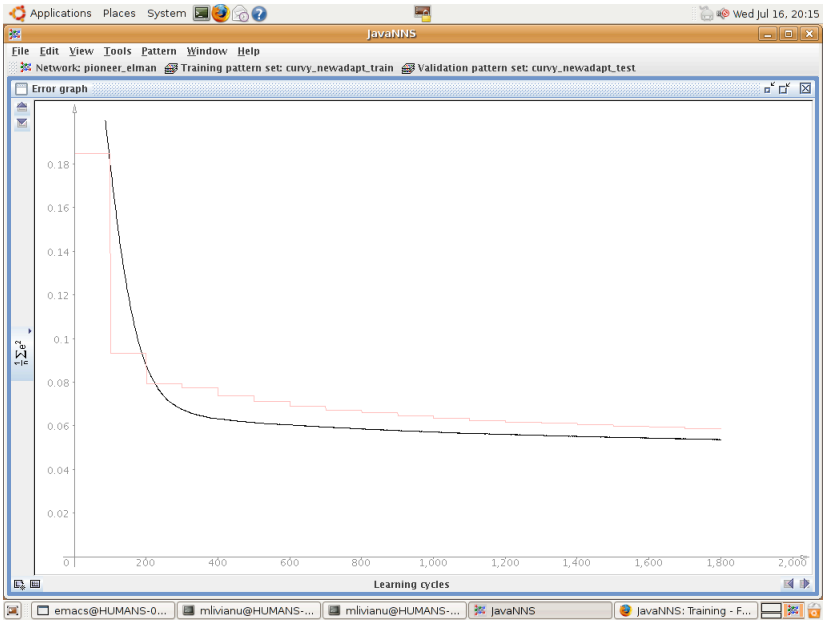
As table 6.4 shows, the adaptive carrot-following algorithm is able to better minimize path error and slip than the expert driver. However, the expert driver sacrifices some path error in order to drive at higher speeds and utilize lateral slip to minimize total time. In order to validate the neural network performance utilizing slip, we train using both adaptive carrot-following data and expert driver teleoperation data.

6.5.1 Training the Neural Network

In this stage of the experiment, we train the neural network utilizing data from expert driver teleoperated runs as well as the adaptive carrot-following runs.

Training is accomplished by splitting the recorded data evenly into a training set and a test set. Neural networks require a great deal of deal of data for training and validation. Even with a human expert driver, tens of runs are required to obtain enough data to bring the network to a low enough error for success. Data collection of this magnitude would

make neural network use infeasible for planetary exploration. Thus, a novel monte carlo approach is used to augment the data set. For each data point, four more data points are created by varying the individual neural network inputs by +/-4%. In this way, we generate enough data points to train the each neural network while only increasing our error margin by 4%, which is within the acceptable range for the experiments.



(a) Adaptive-carrot on wood terrain

Figure 13: Error plots for trained networks. Lower error represents the training set and higher error represents the test set

Figure 13 shows the error curve example for training the neural network using the adaptive carrot-following data from a wood terrain traversal. The data is augmented by a factor of 4 using our monte carlo approach. The error asymptotically approaches approximately 0.07. The two outputs of the neural network range from -1 to 1, which corresponds to the maximum rover speed in the negative and positive directions respectively. Thus, an error of 0.07 corresponds to a variation of 7% in speed, or approximately 7cm/s, which is acceptable.

6.5.2 Running paths

With the neural networks trained, we run experiments on both sand and wood terrains with both the adaptive carrot-following and the teleoperation networks. In order to validate each neural network, we traverse paths it had not seen in training data. For example, the human expert driver data comes from multiple traversals of the curvy CW path, and while the trained neural network is tested on the curvy CW path, it is also tested on the previously unseen curvy CCW and circle paths.

Table 6: Teleoperated vs Carrot-following path traversal on sand

	Control	Terrain	Path	Avg Speed (cm/s)	Avg Path Error (cm)	Avg Slip (cm)
TRAINING	Adaptive Carrot	sand	Curvy CW	13.29	1.85	2.17
RESULTS	Neural	sand	Curvy CW	3.58	8.21	4.99
		sand	Curvy CCW	8.6	6.75	3.64
COMPARE	Adaptive Carrot	sand	Curvy CCW	19.86	4.64	2.55
TRAINING	Tele-operation	sand	Curvy CW	17.69	4.46	4.37
RESULTS	Neural	sand	Curvy CW	10.2	22.51	3.78
		sand	Circle	21.88	14.58	3.23
		sand	Curvy CCW	17.56	25.27	4.79

As shown in Table 6.5.2, adaptive carrot neural networks achieve with lower average path error than the expert driver neural networks on the sand terrain. However, the expert-driver neural networks completes paths in a fraction of the time compared to the adaptive-carrot networks. Slip performance is very similar for all networks, however, average slip decreases for the expert-driver network and increases for the adaptive-carrot network.

As shown in Table 6.5.2, carrot-following neural network achieved lower average path error than the expert driver neural network on the wood terrain. However, the expert-driver neural networks complete paths at a higher speed compared to the carrot network. The carrot-following network performs twice as well in regard to average slip compared to the

Table 7: Teleoperated vs Carrot-following path traversal on wood

	Control	Terrain	Path	Avg Speed (cm/s)	Avg Path Error (cm)	Avg Slip (cm)
TRAINING	Teleoperation	Wood	Curvy CW	25.77	8.82	2.64
RESULTS	Neural	Wood	Curvy CW	19.41	30.86	3.66
TRAINING	Carrot-following	Wood	Curvy CW	18.51	3.86	2.56
RESULTS	Neural	Wood	Curvy CW	16.33	5.88	1.8
TRAINING	Adaptive-carrot	sand	Curvy CW	13.29	1.85	2.17
RESULTS		wood	Curvy CW	17.5	3.04	2.11

teleoperation network, however, average slip decreases for the expert-driver network and increases for the adaptive-carrot network.

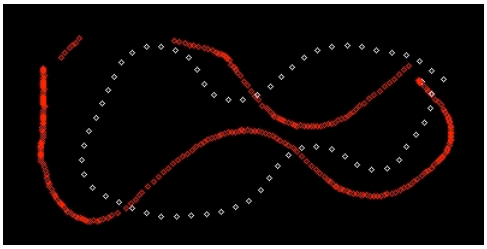


Figure 14: The neural network, trained with adaptive carrot-following traversals on wood, performs well in average speed, but effectively low-pass-filters the curves in the path as it traverses at higher speeds.

The teleoperation neural network is successful at traversing the path on which it is trained as well as a simple circular path. It has greatly increased path error compared to its training set. The network effectively low-pass-filters the curvature of the path, turning the rover less than required in each curve given the higher velocity of the rover. Image 14 shows the neural network performance on wood.

The carrot-following network maintains similar path performance results compared to its training set. Though speeds are higher in the teleoperation network, the carrot-following network also learns to drive fast along straightaways when error and slip are small in magnitude. It improves upon the teleoperation network by turning quickly when the error increases, thus maintaining smaller path error.

6.5.3 Control Comparisons

Num	Data Set	Neural	Avg Speed (cm/s)	Avg Path Error (cm)	Avg Slip (cm)
1	A,S,C1	N(1),S,C1	-73%	345%	130%
1	A,S,C1	N(1),S,C2	-35%	265%	68%
2	A,S,C2	N(1),S,C2	-57%	45%	43%
3	T,S,C1	N(3),S,C1	-42.34%	404.81%	-13.53%
3	T,S,C1	N(3),S,C2	-0.74%	466.69%	9.61%
3	T,S,C1	N(3),S,CIR	23.71%	226.92%	-26.10%
4	CF,W,C1	N(4),W,C1	-12%	52%	-30%
5	A,W,C1	N(4),W,C1	-18%	87%	-28%
5	A,W,C1	N(1),W,C1	-12%	-3%	-15%
6	T,W,C1	N(6),W,C1	-25%	250%	39%

Table 8: Neural network performance on both sand and wood. The 'Num' column assigns a training set number to each entry in 'Data Set'. 'Data Set' and 'Neural' are specified as 'Controller,Terrain,Path'. Controller is either (A)daptive-carrot, (C)arrot-(F)ollowing, or (N)eural-trained-on-data-set(Num). Terrain is either (S)and, or (W)ood, and the path is either (C)urvy(1), (C)urvy(2), or (CIR)cle, where 1 and 2 are CW and CCW respectively.

Table 6.5.3 shows comparisons of performance using teleoperation and carrot-following control, versus neural network control. It identifies each neural network by the data set which was used to train it. For the purpose of analysis, we perform various data comparisons: 1) A comparison of a data set and a neural network trained on that same data set 2) A comparison of a data set and a neural network trained on a different path.

In training ability, the adaptive-carrot network shows a 73% reduction, 345% increase, and 130% increase in average speed, average path error, and average slip respectively when compared to its training data on the same path, curvy CW. On a new path, curvy CCW, the network exhibits almost twice as good performance as on the training path, with a 35% reduction, 265% increase, and 68% increase in average speed, path error, and slip respectively in comparison to the training set. However, when compared to baseline adaptive-carrot runs on curvy CCW, the network runs 57% slower, with 45% more average path error, and 43% more average slip. These results indicate that the network is able to mimic a path follower, learning to drive on not only the path on which it was trained, but also on a new path it has

not seen. It also indicates that the network does not perform as well as the controller that trained it, which is expected.

In comparison to the adaptive-carrot networks, the expert-driver teleoperated networks show a 42% reduction in average speed, a 405% increase in average path error, and a 14% reduction in average slip on the same path when compared to the performance of the original data. When run on a new path, curvy CCW, the network runs at the same speed as the original training, and with less slip, but with much higher path error.

From performance on sand terrain, we can safely conclude that the neural network has learned to traverse a terrain it had not previously seen in training. Taking into account that the curvy CW path is more difficult than the curvy CCW path, the neural network does not manage to minimize path error or traversal time as well as the controllers which trained them. Performance on wood is much more promising, with the networks achieving higher speeds, lower path error, and lower slip (in observation, more lateral slip, and less stationary slip). The most interesting data point is when the neural network trained from adaptive carrot-following on sand, are used to traverse a path on wood. The performance of the neural network is better than the performance of the adaptive carrot-follower on the same path and terrain. From observations, the neural network is able to better utilize lateral slip in negotiating turns at high velocities than it's non-neural counterpart on the same path.

CHAPTER 7

Conclusions and Future Work

We characterize slip in a simulation environment using the High Fidelity Traversability Algorithm (HFTA), showing that the algorithm is able to, in conjunction with the ROAMS environment, compute path and energy costs for a simulated rover following a path over terrain. The HFTA algorithm also produces a lower cost for a path over which a rover incurs less slippage of the wheels and is thus safer to traverse. While ROAMS, at 3.5x real-time, is not yet fast enough to handle real-time traversability assessment, the robustness of the simulator lends itself to more precise results. In its current form, the HFTA algorithm can only analyze one path at a time. Future work includes completing the HFTA algorithm by adding a path perturbing algorithm. In addition further work remains to integrate the HFTA algorithm with the stereo processing algorithms, the path planner, and the path follower as well as perform the simulations aboard the ROCKY8 rover in the JPL Mars Yard.

Using what we learn from the HFTA results, we propose a novel human driver-trained neural network solution to slip-controlled navigation on harsh terrains. We also present an effective new adaptive carrot-following algorithm which anticipates sharp curves and maintains minimum path error and slip more effectively than the original carrot-following algorithm from which it is derived.

Our near-real-time system allows for controlled navigation experiments with human-teleoperation, the new adaptive carrot-following algorithm, and neural networks trained on

real-time data. Through these rover navigation experiments, we show that a neural network trained by a human-expert driver can successfully navigate new paths in real-time.

In training a neural network with human-driver data, we find that the network learns to drive similarly to the human: fast along straightaways when error and slip small in magnitude and turn quickly when error increases. The network trained with adapt carrot driving data performs better than the network trained with human-driver data due to the extra precision provided by a computer controlled driver. We also utilize a monte carlo approach with a +/-4% error to artificially increase the number of data points obtained from real-time experiments. The new data points provide added data to train the neural network, though it does not lead to a significant decrease in overall training error. In future experiments of this kind, much more data is needed to ensure improved network controller learning. In addition, the limitations of pioneer as a skid-steered platform add a great deal of stationary slip in path traversal on low-traction terrains such as sand. Further work on a faster rover that is not skid steered may yield positive results in increasing lateral slip. In addition, characterizing stationary slip vs slip in motion may improve neural network results.

The long delays in the near-real-time system compare to delays incurred by a teleoperated rover on Earth's moon. However, in neural network training, the teleoperation data delays contribute to slower controller reaction times in path traversal. These delays act as an effective low-pass-filter on rover path curve tracking, causing the higher average path error seen in experiments. Future work should include a higher frame-rate camera and lower slip-dt for improved results in neural network performance.

Neural network approaches to slip-controlled navigation are not yet mainstream. However, with the human-rover combined missions around the corner, the desire for higher rover speeds, and the need to traverse more and more hazardous terrains, intelligent autonomous navigation solutions will be utilized more and more. Combined with high performance visual odometry techniques, effective path planning, and current path following

techniques, we believe teach-by-showing methods are effective and will be advantageous in future rover exploration missions.

APPENDIX A

Vision Processing Algorithm

The following pseudo-code describes the process used in image segmentation and rover localization.

```
while (running)
{
  Capture an image frame
  Re-size to 320x240
  Flip the image to correct coordinates
  Crop image to desired area of Mars Yard
  Convert image from RGB to YCrCb color space
  For each of {yellow, blue, purple}
     Separate out red (Cr) and blue (Cb{ frames
     Window the image around the rover if a location is known
     Threshold using static values
     Dilate the image to fill in gaps in detected pixels
     Erode the image
     Median filter to remove noise
     Get the centroid of the detected pixels
     Return the centroid
  Check for unreliable data using minimum and maximum pixel counts
  Use a projective transform to convert the position of each detected
    dot to the world-frame
  Compute the center of the triangle formed by the three localized dots
  Compute the heading of the triangle
  Update the pose of the rover
}
```

REFERENCES

[1] Bekker, M. *Introduction to Terrain-vehicle Systems* Univ. of Michigan Press.

[2] G. Ishigami andK. Yoshida, *Steering Characteristics of an Exploration Rover on Loose Soil Based on All-Wheel Dynamics Model,* Proceedures of the 2005 IEEE Intl. Conference on Intelligent Robots and Systems, pp. 2041-2046, 2005.

[3] G. Ishigami, A. Miwa, and K. Yoshida, *Steering Trajectory Analysis of Planetary Exploration Rovers Based on All-Wheel Dynamics Model,* Proc. of the 8th Intl. Symposium on Artificial Intelligence, Robotics and Automation in Space, 2005.

[4] A. Diaz, C. Elachi, et al., *A Roadmap for the Robotic and Human Exploration of Mars*, May, 2005, http://science.hq.nasa.gov/strategy/road5farch.html.

[5] Biesiadecki, J., et al. *Mars Exploration Rover surface operations: Driving Opportunity at Meridiani planum.* IEEE Conference on Systems, Man, and Cybernetics. 2005.

[6] Leger, C. et al. *Mars Exploration Rover surface operations: Driving Spirit at Gusev Crater.* IEEE Conference on Systems, Man and Cybernetics. 2005

[7] S. Goldberg, M. Maimone, and L.Matthies, *Stereo Vision and Rover Navigation Software for Planetary Exploration*, IEEE Aerospace Conference, Volume 5, Big Sky, Montana, March 2002.

[8] M. W. Maimone and J. J. Biesiadecki, *The Mars Exploration Rover Surface Mobility Flight Software: Driving Ambition,* IEEE Aerospace Conference, Big Sky, Mon- tana, March 2006.

[9] Howard, A. and E. Tunstel. *Intelligence for Space Robotics*. San Antonio, TX: TSI Press, 2006

[10] Andrade, C. et al. *Modeling robot-soil interaction for planetary rover motion control.* International Conference on Intelligent Robots and Systems. 1998.

[11] Iagnemma, K., Shibly, H.A., and Durbowsky, S. *On-line terrain parameter estimation for planetary rovers.* International Conference on Robotics and Automation, and Automation in Space. 2002.

[12] Sohl, Garett and Abhinandan, Jain, *Wheel-Terrain Contact Modeling in the Roams Planetary Rover Simulation,* Proceedings of IDETC - ASME International Design Engineering Technical Conferences, Long Beach, CA, September 24-28, 2005.

[13] G. Sohl and A Jain,= *Wheel-Terrain Contact Modeling in the Roams Planetary Rover Simulation*, Fifth ASME International Conference on Multibody Systems, Nonlinear Dynamics and Control, Long Beach, CA, September 2005.

[14] Jain, et al. *Recent Developments in the ROAMS Planetary Rover Simulation Environment* Proceedings of Aerospace Conference, IEEE Volume 2, March 6-13, 2004 pgs: 861 - 876

[15] Clark F. Olson, Habib Abi-Rached, Ming Ye, Jonathan P. Hendrich, *Wide-Baseline Stereo Vision for Mars Rovers,* In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1302-1307, 2003.

[16] Clark F. Olson and Habib Abi-Rached, *Wide-Baseline Stereo Experiments in Natural Terrain*, In Proceed- ings of the 12th International Conference on Advanced Robotics, pages 376-383, July 2005.

[17] Helmick, et al, *Slip Compensation for a Mars Rover,* IEEE/RSJ International Conference on Intelligent Robots and Systems, 2005: 1419-1426.

[18] Helmick, D., Cheng, Y., Roumeliotis, S., Clouse, D., and Matthies, L. *Path following using visual odometry for a mars rover in high-slip environments.* IEEE Aerospace Conference. 2004. Big Sky, Montana.

[19] D. M. Helmick, S. I. Roumeliotis, Y. Cheng, D. Clouse, M. Bajracharya, L. Matthies, *Slip-Compensated Path Following for Planetary Exploration Rovers,* Journal of Advanced Robotics, October, 2006.

[20] Angelova,A.,Matthies,L., Helmick, D., and Perona, P. *Slip prediction using visual information*. Robotics: Science and Systems Conference. 2006

[21] Angelova,A.,Matthies,L., Helmick, D., Sibley, G., and Perona, P. *Learning to predict slip for ground robots*. International Conference on Robotics and Automation. 2006.

[22] Hannaford, B. and P. Lee, *Hidden Markov Models Analysis of Force/Torque Information in Telemanipulation*, International Journal of Robotics Research. pages 528-538, Oct 1991.

[23] T. Takahashi, H. Ogata, and S-Y Muto. *A Method for Analyzing Human Assembly Operations for Use in Automatically Generating Robot Commands*, IEEE Intl. Conference on Robotics and Automation, Atlanta, GA. vol. 2, pages 695-700. 1993

[24] D. Mypers, M. Pritchard, and M. Brown *Automated Programming of an Industrial Robot through Teach-By Showing*, IEEE International Conference on Robotics and Automation, Seoul, Korea. May 2001

[25] Zhang, T. *Neural Network-Based Hybrid Human-in-the-loop Control for Meal Assistance Orthosis*, IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol. 14, No. 1, March 2006.

[26] Jain, et al. *http://dartslab.jpl.nasa.gov* NASA Jet Propulsion Laboratory

[27] Biesiedecki, J., Henriquez, D. and Jain, A. *A Reusable, real-time spacecraft dynamics simulator.* 6th Digital Avionics System Conference, 2004.

[28] C. Kwan, D.M. Dawson, and F.L. Lewis. *Robust Adaptive Control of Robots Using Neural Network: Global Stability*. Asian Journal of Control, Vol. 3, No. 2, pages 111-121, June 2001.