Motion Planning for High-Speed and Kinematically Reconfigurable Mobile Robots in Rough Terrain

by

Adam K. Rzepniewski

B.S., Mechanical Engineering
University of Notre Dame, 1999

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

at the

Massachusetts Institute of Technology

September 2001

Signature of Author . . [illegible] . . . .
Department of Mechanical Engineering
August 10, 2001

Certified by . . . . [illegible] . . . .
Steven Dubowsky
Professor of Mechanical Engineering
Thesis Supervisor

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

Motion Planning for High-Speed and Kinematically Reconfigurable Mobile Robots in Rough Terrain

by

Adam K. Rzepniewski

Submitted to the Department of Mechanical Engineering
on August 10, 2001 in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

ABSTRACT

Robotic rovers are important scientific tools used to explore and gather data from other planets. However, current rovers are limited in capability due to their conservative control algorithms and kinematic design. This thesis reviews two approaches to help planetary vehicles take full advantage of their abilities. The first is an algorithm to maximize the stability of kinematically reconfigurable rovers moving in rough terrain. This method has been introduced in Iagnemma *et al* [1]. The author's contributions are aid in theory application and simulation and experimental validation. This method is intended for robots with redundant degrees of freedom, i.e. those with a greater number of actuators than the minimum needed for motion. A stability metric that optimizes the vehicle pose based on stability and ground clearance is proposed. The reconfiguration algorithm is verified both in simulation and through experimental testing. Results from field trials at the Arroyo Seco in Altadena, California are presented.

The second approach explores an algorithm for off-line, planar motion planning of high-speed vehicles operating in rough terrain. A nine degree of freedom planar vehicle model that accounts for body-wheel interaction is developed. Vehicle failure criteria are discussed. A method for generating maximum and minimum velocity trajectories for a vehicle to traverse a given terrain profile is presented. The algorithm shows that aggressive behaviors such as jumping may safely be used to traverse particularity rough terrain. However, the algorithm also proves not to be practically applicable due to high computational requirements and the need for *a priori* knowledge of the full terrain profile.

Thesis Supervisor: Steven Dubowsky
Title: Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

First, I would like to thank my parents, Zdzislaw and Helena Rzepniewski, and my sister, Eva, who have supported me throughout my whole academic career. Thank you for all of your personal, spiritual and, yes, financial help without which none of this would have been possible. Thank you for sharing in all the good, and bad, times along the way.

Thanks to Dr. Steven Dubowsky, director of MIT's Field and Space Robotics Laboratory (FSRL), and the whole FSRL team who have taught me research skills. This is knowledge that I will keep for a lifetime. Special thanks to Karl Iagnemma and Vivek Sujan with whom I have spent countless hours discussing various approaches to robotic vehicle control.

I would also like to thank Dr. Paul Schenker, Dr. Terry Huntsberger, Dr. Paolo Pirjanian, Mike Garrett, and Hrand Aghazarian of JPL's Planetary Robotics Laboratory. Thank you for allowing me to become a part of your research team. Special thanks to Anthony Ganino and Brett Kennedy who have helped me understand high-speed rough terrain motion.

This work is supported by NASA/JPL under contract number 1216342.

Contents

FIGURES

TABLES

CHAPTER 1 INTRODUCTION

1.1 Introduction

This thesis presents the author's contribution to a research program designed to increase the capabilities of planetary explorer vehicles. This program includes investigation of new or improved methods of mobility, manipulation, sensing, and control of planetary rovers. The project is a collaborative effort between MIT's Field and Space Robotics Laboratory and NASA's Jet Propulsion Laboratory (JPL). The program is centered on the development and use of rover mechanics models that are simple enough for on-board implementation, yet capture all relevant aspects of vehicle motion. This thesis reviews a method for kinematic reconfigurability, [1], and shows its application in simulation and through experimental testing on the JPL Sample Return Rover [2]. This thesis also presents the results of a study towards generating a motion planner for high-speed rough terrain traversal. Only planar motion is considered. The resulting algorithm shows that aggressive behaviors such as jumping may safely be used to traverse rough terrain. However, the algorithm also proves not to be practically applicable due to high computational requirements and the need for *a priori* knowledge of the full terrain profile.

1.2 Motivation

On July 4, 1997 the Pathfinder lander and the Mircrorover Flight Experiment (MFEX), Sojourner, landed on Mars. The mission lasted approximately 78 Martian days during which Sojourner traveled approximately 100 m (total integrated distance) and performed numerous scientific experiments [3]. This mission sparked the public's interest and confirmed the viability of using small robotic vehicles for planetary exploration.

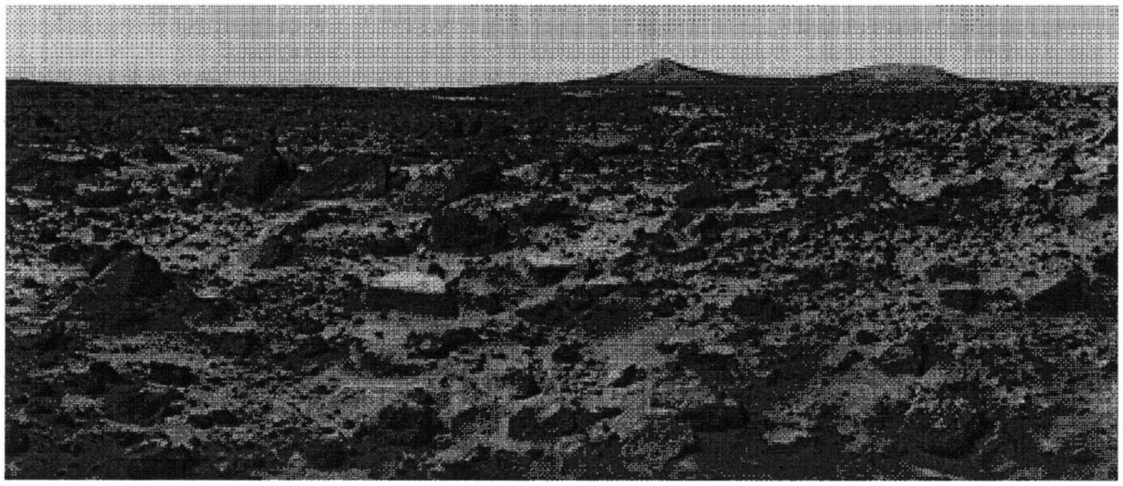

Figure 1: Mars landscape as seen by Pathfinder cameras [3].

NASA/JPL intend to continue to use small robotic rovers for planetary surface exploration [2, 4]. However, future vehicles must have capabilities far beyond those of Sojourner. Future missions include geologic sample acquisition, long-range travel, and mapping [4]. To accomplish these missions rovers will have to travel into potentially dangerous areas such as rock outcroppings or descend down the face of a cliff. Long-range travel and terrain mapping will certainly require travel through regions of challenging traversability.

Figure 1 shows the landscape of Mars as seen by the Pathfinder cameras. Even though this is the spacecraft landing area and should therefore possess benign terrain, it is characterized by a large number of rocks that can threaten rover safety and mission success. Clearly, Mars presents many mobility challenges for small mobile robots.

Current planetary rovers and laboratory prototypes are limited in their mobility capabilities. They are slow moving with conventional, fixed suspension designs. Figure 2 shows the Sojourner rover traveling over terrain in the proximity of the lander. The figure illustrates two near-failure situations and, thus, demonstrates the need for more advanced vehicle designs and motion planning paradigms. Physical limitations coupled with conservative control algorithms often result in rover endangerment or overly cautious vehicle motions that restrict the scientific return of planetary missions.

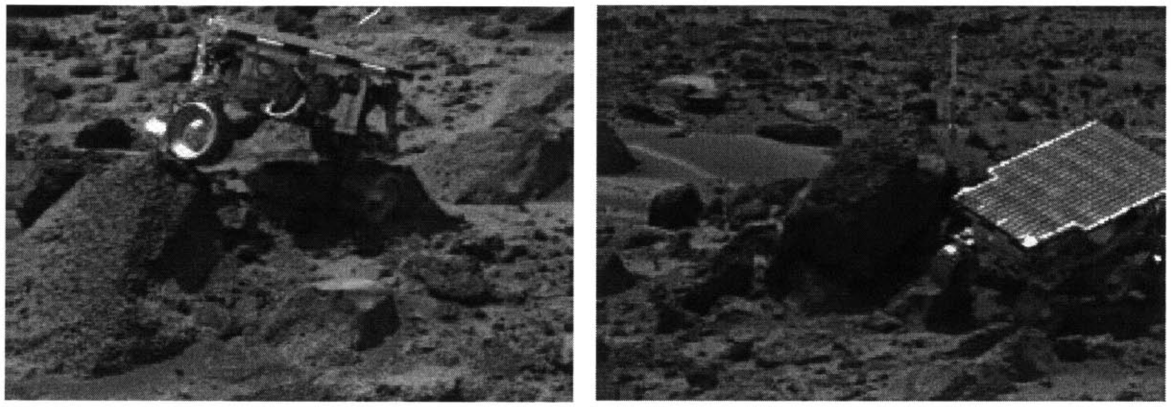

Figure 2: Sojourner rover traveling over Martian terrain [3].

One important characteristic of a planetary rover is its ability to handle rough, unknown terrain. The vehicle must be able to negotiate large rocks, loose soil, and steep slopes. Figure 3 and Figure 4 show advanced rovers that are designed to meet increasing demands on system mobility. Both the Sample Return Rover (SRR) and the nanorover, shown in Figure 3, are designed with reconfigurable shoulder joints that allow each vehicle to change its body roll and height [5, 6]. The SRR is also equipped with a manipulator that can be used to alter the location of the center of mass to prevent tipover. The nanorover, on the other hand, is capable of traveling both right side up and upside down. URBIE, shown in Figure 4, is a tactical mobile robot capable of changing its front flipper angles [7]. This capability, along with a treaded-track design gives the system high mobility. These rover designs require new control algorithms that take advantage of their increased capabilities.

Autonomous vehicle systems must be able to operate at high-speed in rough terrain. Although high-speed off-road motion is common in many racing and military applications, it has rarely been seen in autonomous vehicles. A high-speed vehicle can accomplish more tasks than a slow moving one in equal time. New and aggressive motion planning algorithms that take full advantage of rover capabilities are needed to fulfill the requirements of future planetary missions.

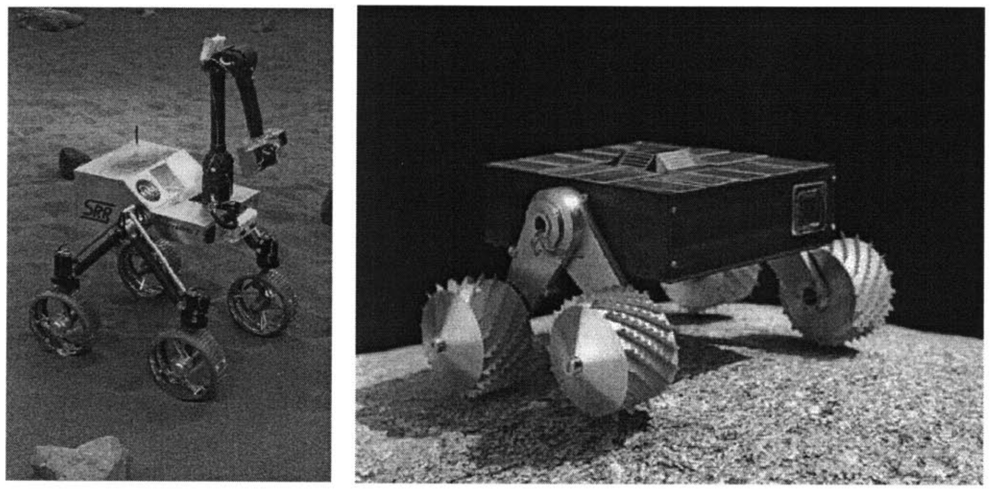

Figure 3: Reconfigurable rovers (a) Sample Return Rover (SRR) [5], (b) Nanorover [6].

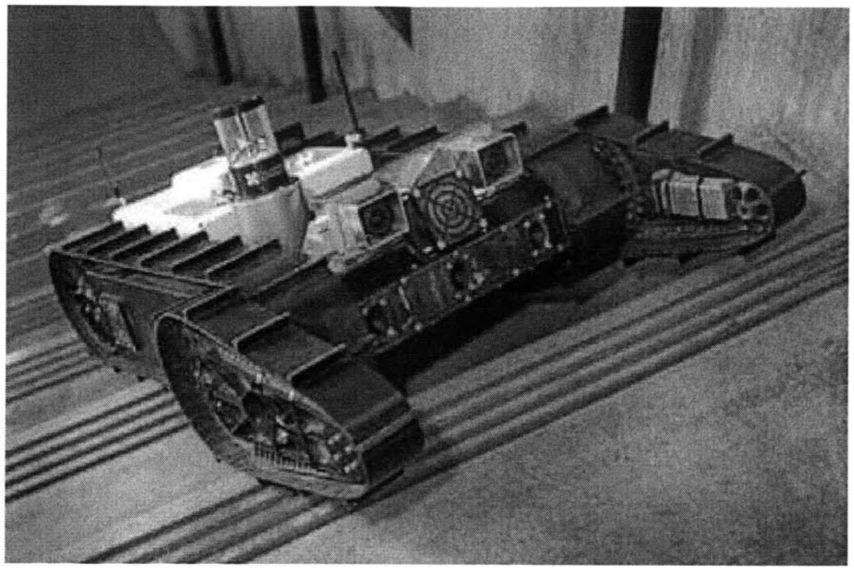

Figure 4: Tactical mobile robot URBIE [7].

1.3 Literature Review

Many researchers have studied rough terrain motion of planetary surface robots. A full body of work extending from motion planners to path planners is available for the interested reader [8-29]. A motion planner is defined here as an algorithm that, given a path, constructs a safe system motion profile subject to vehicle and interaction constraints. A path planner is defined here as an algorithm that is able to construct a feasible path through a given terrain and then use a motion planner to determine a motion profile.

1.3.1 Kinematic Reconfigurability

The use of kinematic reconfigurability to enhance rough terrain performance has been proposed by previous researchers [8, 9, 10, 11, 12]. However, its benefits have not been previously verified experimentally. Kinematic reconfigurability algorithms are designed for robots with redundant actuators and available self motions. Examples of such robots are shown in Figure 3, Figure 4, and Figure 5.

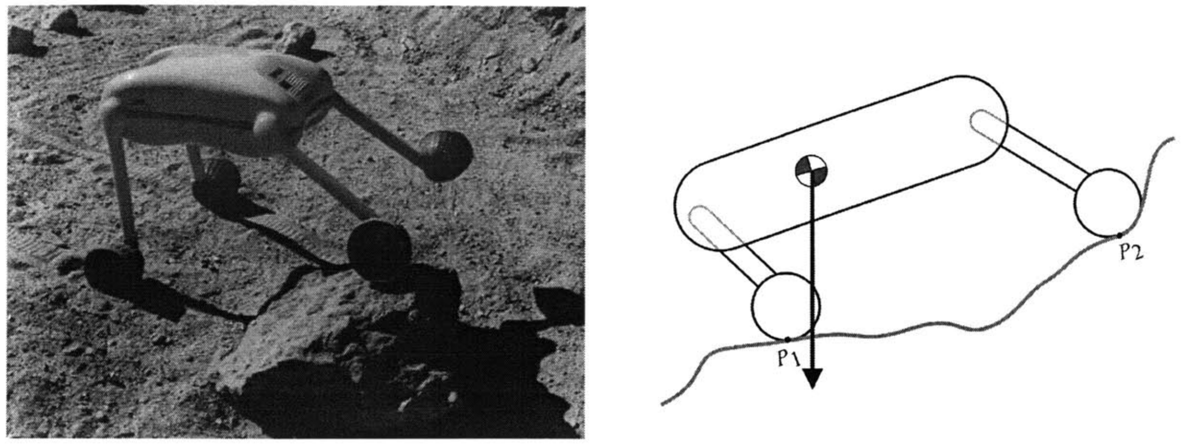

Figure 5: (a) NASA's Gofor rover [8], (b) Near tipover failure.

A method to improve vehicle traction through pose reconfiguration is presented in [8, 10]. The control algorithm is designed for the NASA/ JPL Gofor rover, see Figure 5. Vehicle traction is improved by altering the vehicle pose until system weight is placed over the wheels with the most favorable wheel-ground contact angles. However, the goals of obtaining optimal traction and maintaining vehicle stability are often conflicting. Figure 5 shows a case where traction-based force allocation places the vehicle close to tipover failure. A minimum safety margin is suggested to cope with such instances. However, stability is not explicitly considered in determining the optimal vehicle pose.

Kinematic reconfigurability methods to improve vehicle stability are presented in [11, 12, 13]. Reconfiguration based on maintaining zero roll and pitch of the body is suggested in [11]. This however, does not explicitly account for vehicle stability. In fact, this may lead to instability. Figure 6 shows a case where leveling out the body would cause tipover failure. In [12, 13], a planning algorithm to determine the optimal vehicle pose is presented. The method requires detailed knowledge of the terrain profile. Such information is often not available. Also, due to its complexity, this method is not feasible for on-line implementation. Thus a need exists for an algorithm that can maintain system safety and is simple enough for on-line implementation. Such an algorithm is presented in [1] and reviewed in Chapter 2. In this thesis, this algorithm is verified both in simulation and through experimental testing on the JPL SRR [2].

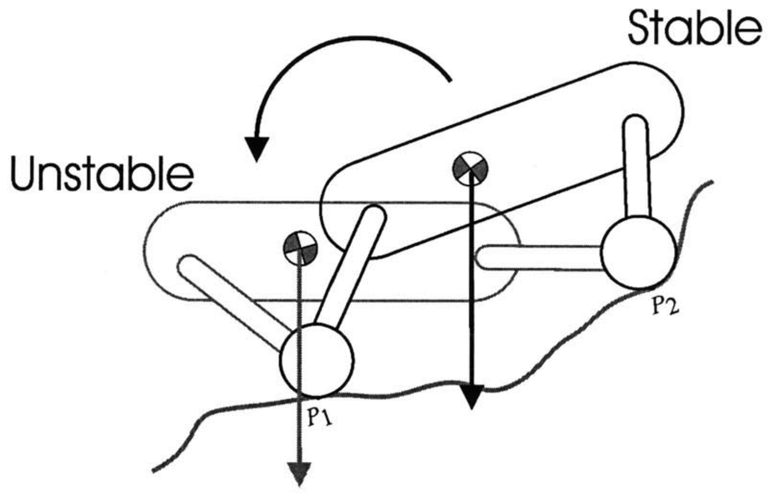

Figure 6: Tipover failure caused by leveling out the vehicle body.

1.3.2 High-Speed Rough Terrain Motion

High-speed rough terrain motion has also been studied in detail [14-33]. A method for real-time motion planning while accounting for body dynamics is presented in [14]. The vehicle plans its motion based on a canonical solution that gives the time-optimal path within the robot's sensing range. Time optimal motion considering vehicle dynamics is also presented in [15]. The authors find the motion solution by dividing the path into translational and rotational (turning) sections. Although both this algorithm and [14] account for body dynamics, they consider only smooth terrain with clearly defined obstacles.

Motion planning with physical interaction constraints, no skidding and constant ground contact, is presented in [16]. The algorithm combines a graph-search of the configuration space, to determine a feasible path, and the use of landmarks, to correct position errors accumulated by relying on odometry and inertial data. This work integrates rough terrain with vehicle dynamics to determine a solution. However, interaction constraints prevent aggressive vehicle behavior.

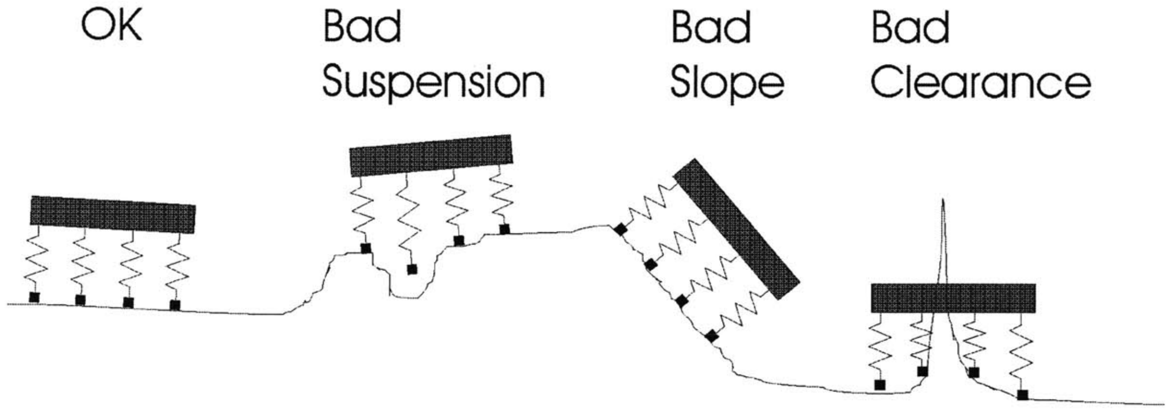

Figure 7: Vehicle model used in Hughes AIC control algorithm [17].

An approach to cross-country navigation is presented in [17]. The control algorithm consists of a map-based planner, which provides a description of the intended vehicle route, and a reflexive planner, which steers the vehicle in response to perceptual data received during the actual traverse. A model-based evaluation of terrain traversability is performed only when requested by the control algorithm. A static, 3D vehicle model was developed for this purpose. Figure 7 shows, in a planar view, the three types of obstacles detected with the vehicle model. Although this control algorithm is made for rough-terrain operation, it only considers conservative traverses that need not account for vehicle dynamics.

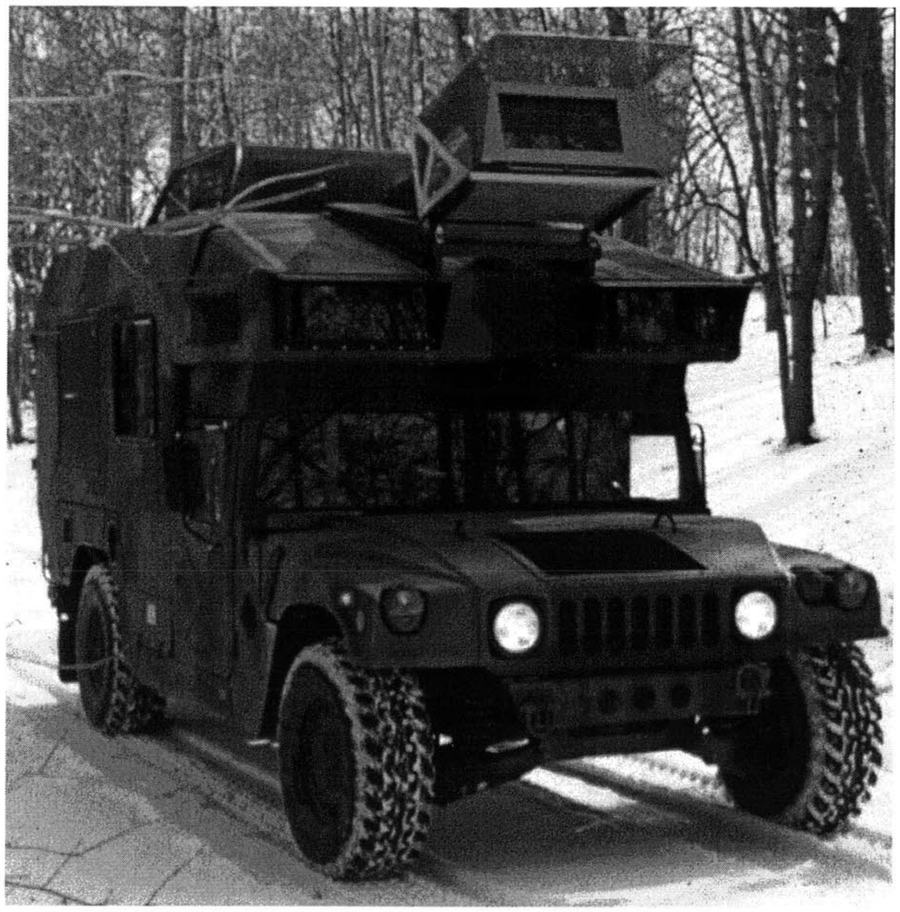

Figure 8: The CMU Unmanned Ground Vehicle Navlab II [20].

A different approach to controlling fast moving vehicles in rough terrain is taken in [18, 19, 20, 21, 22]. Figure 8 shows the Carnegie Mellon University Unmanned Ground Vehicle used for the Navlab and RANGER projects. The control architecture for this vehicle is an arbitration scheme. Desired vehicle action is determined by votes from such behaviors as "avoid obstacles" and "maintain heading." The control algorithm considers only feasible solutions in the vehicle actuation space (steering, throttle, and brake). A simple system model, which considers the vehicle body as the only mass bearing element, is used for this evaluation. Frequent on-line evaluation of the equations of motion with the current system state as the input vector is used to determine vehicle motion solutions. A feasible solution is one that satisfies the differential equations of motion and a ground contact constraint. Once again, the combination of an overly simplified system model and a ground contact constraint prevents consideration of non-benign off-road terrain and truly aggressive vehicle behavior.

High-speed, up to 35 km/h, off-road autonomous driving is discussed in [23, 24]. The key features of the control algorithm are: increasing the clearance between the side of the vehicle and obstacles when traveling at higher speeds and only planning the next 20 m with dynamically feasible trajectories. The first feature means that only binary obstacle classification is allowed; no model-based traversability evaluation is undertaken. For the second, paths extending 20 m from the vehicle consist of clothoidal segments. A clothoid is a curve whose curvature varies linearly with arc-length, i.e., a constant curvature plus a constant rate of change of curvature:

$$\frac{1}{r} = c_o + c_1 s$$

where r is the radius of curvature and s is the distance traveled. A large number of clothoid segments (paths) is computed off-line. Only those segments that meet the vehicle state and are not blocked by obstacles are considered during on-line operation. The vehicle model considers vehicle speed and actuator dynamics (steering) in order to determine feasible paths. Due to this simplification, the experimental system is limited to travel over smooth (rolling) off-road terrain.

Dynamic motion and time-optimal trajectory planning for autonomous vehicles is presented in [25, 26, 27]. This work, borne out of motion planning for robotic manipulators, has been demonstrated in simulation. Knowledge of the full terrain profile from the starting point to the goal is required. The terrain is represented by B spline patches. Path evaluation is then carried out to satisfy engine torque, sliding, contact, and tipover constraints. The evaluation yields a set of maximum velocity curves that are used to generate a time-optimal trajectory in $s-\dot{s}$ space, where s is the distance along the path and $\dot{s}$ is the velocity. A simple vehicle model and rolling constraints allow this manipulator-based kinematic analysis of the system motion space. The vehicle is modeled as a point mass suspended above the terrain. All reaction and friction forces are transferred to the vehicle center of mass. Although this formulation allows a closed form solution to vehicle motion, it does not permit sliding and ballistic motion and thus prevents aggressive travel over natural terrain.

Alternative methods of motion generation are presented in [28, 29]. Both of these papers use fuzzy logic techniques to determine optimal paths. Genetic algorithms are used to optimize inference tables in [28]. These tables are subsequently used by a fuzzy logic planner. The resulting paths are superior to those obtained from random decision tables generated using ad-hoc rules [28]. Fuzzy logic and vehicle behaviors are used in [29] to determine terrain traversability and perform motion planning. The method requires no *a priori* information about the environment and is suitable for real-time implementation. Although these are both innovative solutions to high-speed rough terrain motion planning, they require constant ground contact and thus permit only conservative motion.

Methods for controlling vehicles by modifying the relative movement of the sprung (body) and unsprung (wheels) weight are presented in [30, 31, 32, 33]. Clearly control of this motion is important since it affects vehicle handling and safety. However, overly simplified vehicle models used in the bulk of path planning algorithms often ignore such motion.

The bulk of previous work does not consider and exploit the full motion capabilities of today's high-speed mobile robots. While this problem remains unsolved, Chapter 3 presents the results of a study towards generating such a motion planner, based on a nine degree of freedom planar vehicle model. This planar motion planning algorithm shows that aggressive behaviors such as jumping may be safely used to traverse particularly rough terrain. However, it also proves not to be practically applicable due to high computational requirements and the need for *a priori* knowledge of the full terrain profile.

1.4 Research Overview

Rough terrain motion is a formidable problem that has been studied for many years. Constant advances in materials and computing technology allow vehicles to become more sophisticated and present new challenges to the controls engineer. One major challenge, studied here, is to make rovers more robust, safe, and, at the same time, daring.

Planetary explorer vehicles need to do just that, explore. Often this means they must travel through regions that may threaten their safety. Some of the most interesting scientific samples are thought to be found in rock outcroppings and lie near cliff sides [2]. This research is a small part of a joint MIT-NASA/JPL program to develop new control strategies that increase the mobility capabilities of rovers. Physics-based, mechanical models of robotic systems are used to take advantage of vehicle capabilities. Whenever possible, experimental validation of control algorithms is used as proof of their practical effectiveness.

1.5 Thesis Outline

This thesis is divided into four chapters. Chapter 1 presents the motivation, a review of past work, and a brief research overview. Chapter 2 reviews a method for kinematic reconfigurability used to increase system stability, [1], and presents simulation and experimental results from field trials with the JPL Sample Return Rover (SRR). Chapter 3 presents the results of a study towards developing a motion planner suitable for high-speed off-road motion. Only planar motion is considered. The algorithm uses a nine degree of freedom planar model that accounts for both body and wheel dynamics. The planner is then used to determine maximum and minimum velocity trajectories, defined in Section 3.3.3, between a starting and goal point. The algorithm shows promising results. However, it proves not to be practically applicable due to high computational requirements and the need for *a priori* knowledge of the full terrain profile. Chapter 4 presents conclusions of the research to date and suggestions for future work.

CHAPTER

2 KINEMATIC RECONFIGURABILITY FOR ROUGH TERRAIN TRAVERSAL

This chapter reviews work previously published in Iagnemma *et al.* [1], Iagnemma [34], and Schenker *et al.* [13]. This work is a result of a collaborative effort between the author and K. Iagnemma. The author's contributions to this work are aid in theory application and simulation and experimental validation.

2.1 Motivation

While significant recent progress has been made in the development of planetary explorers many challenges remain, including traversal of rough terrain. This can result in loss of stability, leading to tipover, or loss of wheel traction, leading to vehicle entrapment. Either one can jeopardize rover safety and mission success.

NASA's Jet Propulsion Laboratory has developed the Sample Return Rover (SRR) with the ability to actively modify its kinematic configuration to adapt to terrain and enhance its rough terrain mobility [35]. The SRR is a 7 kg, four-wheeled robot with independent steering and active, articulated shoulder joints. It can also be equipped with a 2.25 kg three degree of freedom manipulator. Reconfigurable robots can adjust their kinematic configuration and reposition their center of mass to improve stability in rough terrain. Figure 9 shows a diagram of how an SRR-like robot can lower one side to adjust to a hillside and improve its stability margin. This is accomplished by using its two active shoulder joints to modify the angles θ_1 and θ_2, see Figure 9 and Figure 10. The SRR can also use its manipulator to reposition its center of mass. Both of these capabilities can be used to improve system stability.

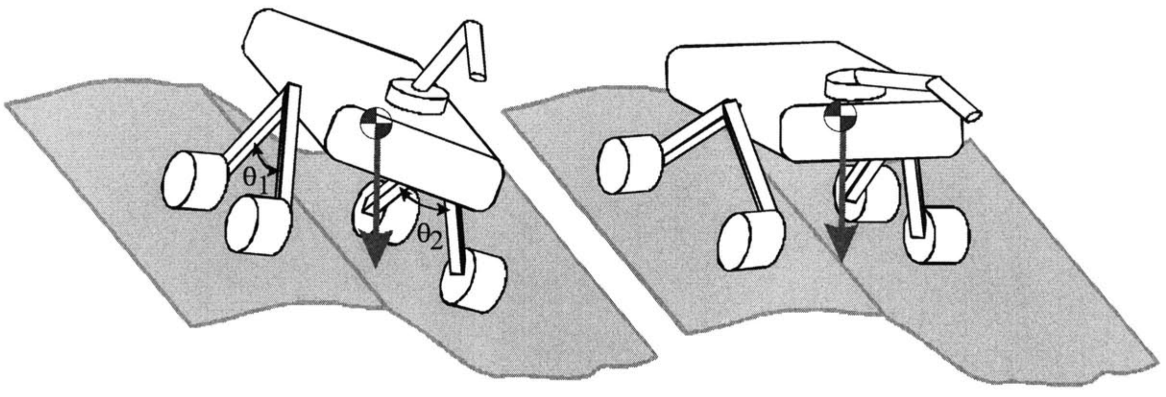

Figure 9: Example of reconfigurable robot improving rough-terrain stability by adjusting shoulder joints [1].

Previous researchers have suggested the use of kinematic pose optimization to enhance rough terrain mobility [8, 36, 37]. However, its practical effectiveness has not yet been demonstrated on a real rover system operating in rough terrain. This chapter reviews a stability-based kinematic reconfigurability method that was applied to the JPL SRR [1] and presents simulation and experimental results. Kinematic equations relating the actively actuated shoulder and manipulator joints to a vehicle stability measure are written in closed form. A performance index is defined based on this stability measure and a function that maintains adequate ground clearance, an important consideration when traveling in rough terrain. The performance index is optimized by a conjugate-gradient method, subject to kinematic constraints. The method does not consider a detailed map of the terrain, preventing global optimization (the reasons for this are explained in Section 2.2). However, local wheel-terrain contact angles are estimated using simple on-board sensors [38], permitting local optimization. The method considers forces acting on the system, including those due to gravity and ones arising from manipulation of the environment.

Computational requirements are light, permitting on-line implementation even with the limited resources of planetary rovers [2]. Simulation and experimental results gathered in an unstructured environment show that kinematic reconfigurability can substantially improve vehicle stability, and consequently the vehicle mobility, in rough terrain.

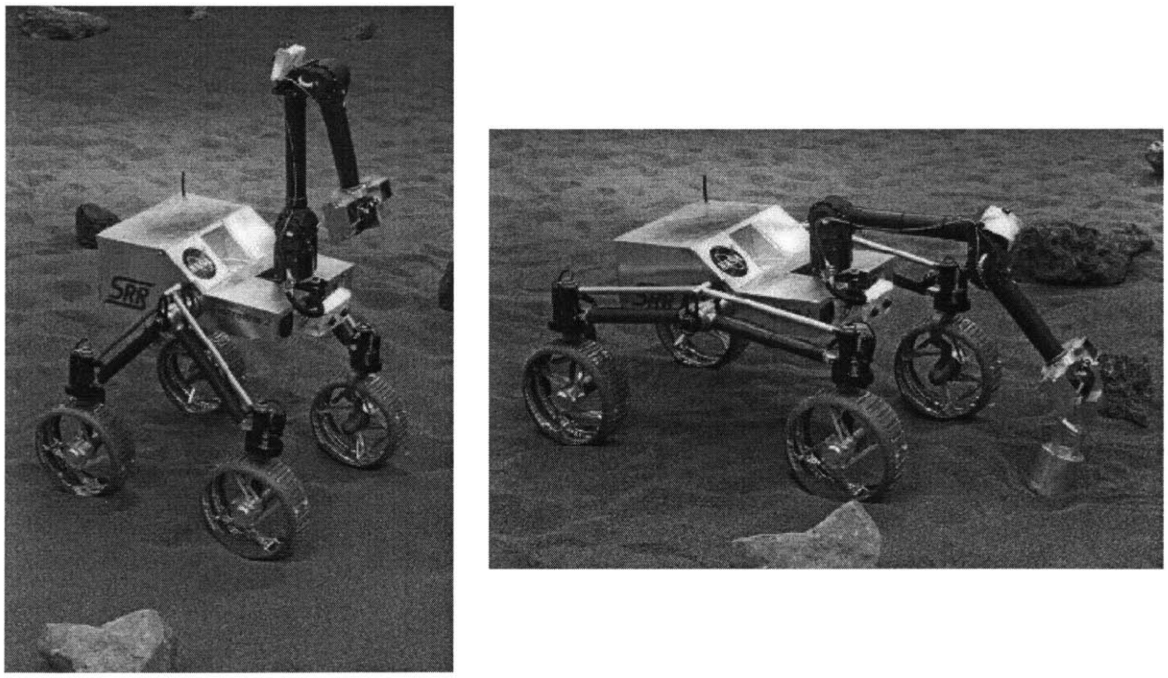

Figure 10: Jet Propulsion Laboratory's Sample Return Rover (SRR) [5].

2.2 Kinematic Reconfigurability

This section reviews two types of kinematically reconfigurable mechanisms described in Iagnemma *et al* [1].

Kinematically reconfigurable mechanisms may be divided into two classes: internally reconfigurable and externally reconfigurable. Internally reconfigurable mechanisms can change the position of their center of mass without changing their link-terrain contact points. Externally reconfigurable mechanisms, on the other hand, need to change their link-terrain contact points in order to change the position of their center of mass. In such mechanisms, link-terrain contact points must be treated as higher-order pairs [39]. The externally reconfigurable SRR might consequently be represented as a crank-slider mechanism.

The goal of kinematic reconfigurability is to optimize a user-defined performance metric by modifying the joint variables θ_i. Examples of performance metrics include static stability, wheel (or foot) traction, and optimal force distribution. A stability-based metric, discussed in Section 2.5, is used in this work.

On-line kinematic reconfigurability requires three steps:

1) Evaluation of the mechanism kinematic configuration based on on-board sensor readings. The configuration is defined as $Q = (\theta_1,...,\theta_m,\alpha_i,...,\alpha_m,\phi,\chi)$ where ϕ and χ denote the pitch and roll, respectively, of the vehicle body. The link-terrain contact angles, α_i, can be estimated by various methods [38, 12].

2) Calculation of the configuration Q^*, which optimizes the performance metric based on the joint angles θ_i.

3) Transition from the current configuration Q to the optimal configuration Q^*.

Detailed knowledge of the terrain is required to obtain a globally optimal solution for externally reconfigurable mechanisms since link-terrain contact points change during reconfiguration. This information is often not available. The computational burden of continually obtaining and processing detailed range maps usually makes the task impractical. However, local link terrain contact angles may be used to obtain a locally optimal vehicle pose without much computational burden. A method for obtaining these angles is described in [38]. The local terrain angles describe the terrain in the immediate vicinity of the contact point. Thus, a local optimization problem may be posed with the additional constraint that only small changes in the location of the contact points are permitted.

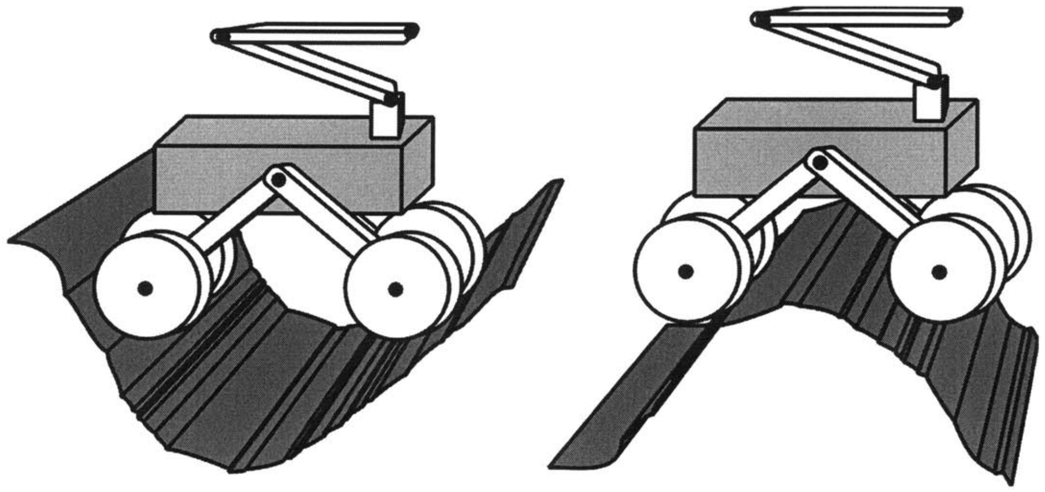

Figure 11: The effect of local terrain profile on an externally reconfigurable vehicle.

The importance of knowing the local terrain profile is demonstrated in Figure 11. Here the terrain requires two drastically different courses of action from the same starting pose; when traveling in a ditch, closing the shoulders will increase stability, whereas the opposite action is required when traveling on a hill.

2.3 Vehicle Kinematics

Knowledge of the system kinematics is necessary to determine the optimal pose. Since forward and inverse kinematics of hybrid serial-parallel chains are, in general, difficult to solve in closed form, numerical optimization techniques are often used [8]. Although numerical solutions are often required to obtain full loop-closure equations for these tree-structured systems, analytical formulation for each independent branch may often be obtained with only moderate effort. Thus, each vector $\bar{p}_j$, describing the location of the link terrain contact point P_j, may be written independently, see Figure 12. Appendix A reviews the development of the SRR kinematics through the use of Denavit-Hartenberg parameters [40].

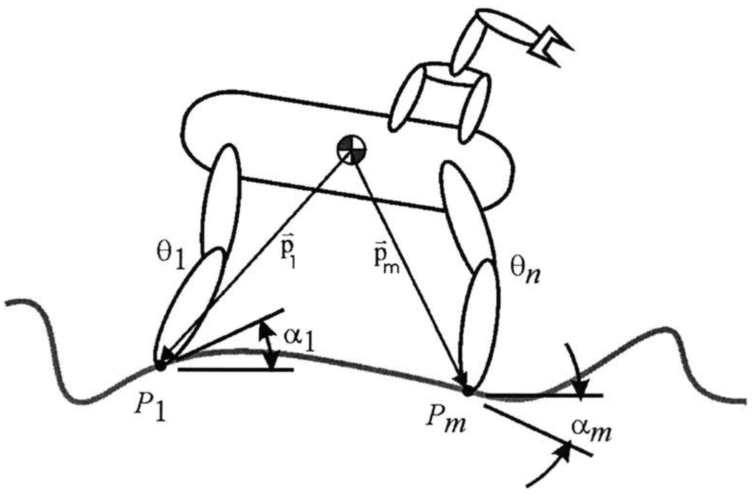

Figure 12: A general *n*-link robotic mechanism [1].

2.4 Force-Angle Stability Measure

This section reviews a stability metric proposed by Papadopoulos and Rey [41].

For the general n-link mechanism shown in Figure 1, m link-terrain contact points $P_j, j = \{1,\dots,m\}$ are numbered in ascending order in a clockwise manner when viewed from above, see Figure 13. The lines that join the contact points are the candidate tipover axes, a_i. The i^{th} tipover axis is given by:

$$\vec{a}_i = \vec{p}_{i+1} - \vec{p}_i \tag{1}$$

$$\vec{a}_m = \vec{p}_1 - \vec{p}_m \tag{2}$$

where $\vec{p}_j$ is defined from the vehicle center of mass to the contact point P_j. Note that a vehicle with m contact points will have m axes about which an untripped tipover may happen. A tripped tipover occurs when one of the ground contact points encounters an obstacle or a sudden change in ground conditions.

Tipover axis normals that intersect the center of mass are given by:

$$\vec{l}_i = (\hat{1} - \hat{a}_i\hat{a}_i^T)(\vec{p}_{i+1} - \vec{p}_c) \tag{3}$$

where $\hat{1}$ is a 3x3 identity matrix, $\hat{a}_i = \vec{a}_i / \|\vec{a}_i\|$, and $\vec{p}_c$ describes the location of the center of mass in the inertial reference frame.

Stability angles may then be computed for each tipover axis as the angle between the general force vector $\vec{f}$ and the axis normal $\vec{l}_i$, and is given by:

$$\gamma_i = \sigma_i \cos^{-1}(\hat{f}_i \cdot \hat{l}_i), \quad i = \{1,\dots,m\} \tag{4}$$

where

$$\sigma_i = \begin{cases} +1, & (\hat{l}_i \times \hat{f}_i) \cdot \hat{a}_i \\ -1, & otherwise \end{cases} \tag{5}$$

and

$$\hat{f}_i = (\hat{1} - \hat{a}_i \hat{a}_i^T)\vec{f} \tag{6}$$

The sign of σ_i is positive if the force vector passes through the interior of the contact polygon. The overall stability angle is defined as the minimum of all i stability angles:

$$\alpha = \min(\gamma_i), \qquad i = \{1,...,m\} \tag{7}$$

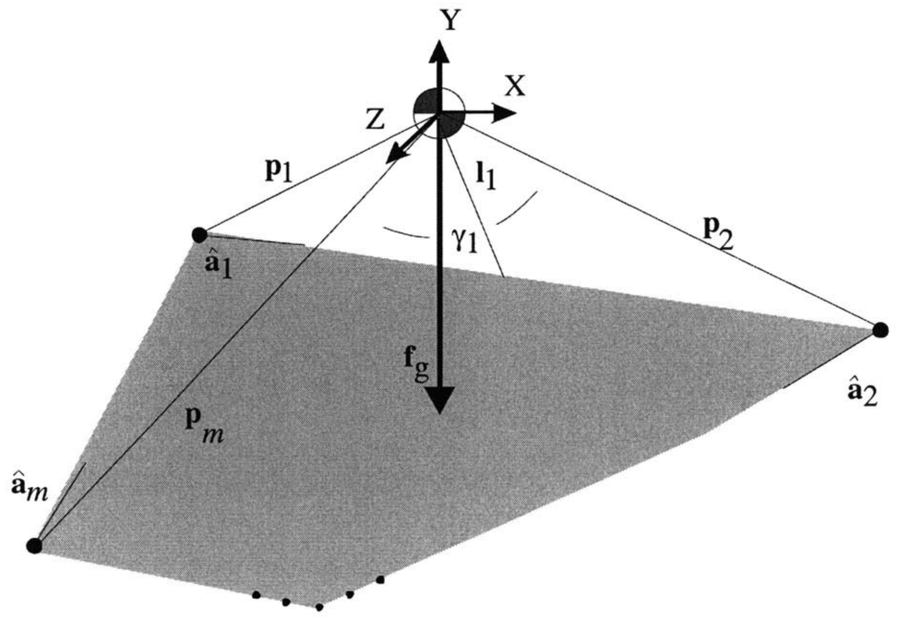

Figure 13: Stability definition diagram [41].

Note that the reconfigurability method presented here only considers gravitational forces. This, however, need not be a constraint. Any additional forces, such as ones arising from manipulation of the environment, can be included:

$$\vec{f} = \sum(\vec{f}_{grav} + \vec{f}_{manip} + ...) \tag{8}$$

Similarly, any moment $\vec{n}$ about a tipover axis $\vec{a}_i$ may be included:

$$\vec{f}^* = \vec{f} + \frac{\hat{l}_i \times \vec{n}_i}{\left\| \hat{l}_i \right\|} \tag{9}$$

with

$$\vec{n}_i = (\hat{a}_i \hat{a}_i^T)\vec{n} \tag{10}$$

The stability angle γ_i is then computed using the net force $\vec{f}^*$.

2.5 Performance Index and Optimization Method

A stability-based performance metric, Φ, is defined based on the force-angle stability measure. The metric takes the following form:

$$\Phi = \sum_{i=1}^{n} \left(\frac{K_i}{\gamma_i} + K_{n+i} (\theta_i - \theta_i')^2 \right) \tag{11}$$

where γ_i is the stability angle as described in Section 2.4, θ_i' is the nominal value of the i^{th} joint (i.e. the value of θ_i when the robot is in a user-specified configuration), and K_i is a constant weighting factor.

The performance metric, Φ, was chosen to have three key properties.

1. The first term in the equation above tends to infinity when any stability angle γ_i tends to zero. This elicits a strong response from the system as it becomes marginally stable.

2. The second term allows the vehicle to maintain ground clearance. This is accomplished by penalizing deviation from the user specified configuration θ_i'.

3. The weighting factors K_i allow a user to specify the relative importance of stability and ground clearance. Stability could be weighted more heavily when traveling in difficult terrain. Minimizing joint movement, and thus power usage, could be weighted more when traveling in benign terrain.

The goal of stability based kinematic reconfigurability is to minimize the value of the performance metric Φ subject to joint limits, interference, and kinematic and loop-closure constraints. When applied to the SRR, Φ possesses a unique minimum within an optimization cycle and a conjugate-gradient method is sufficient for rapid convergence on the solution. Figure 14 shows a typical optimization search space.

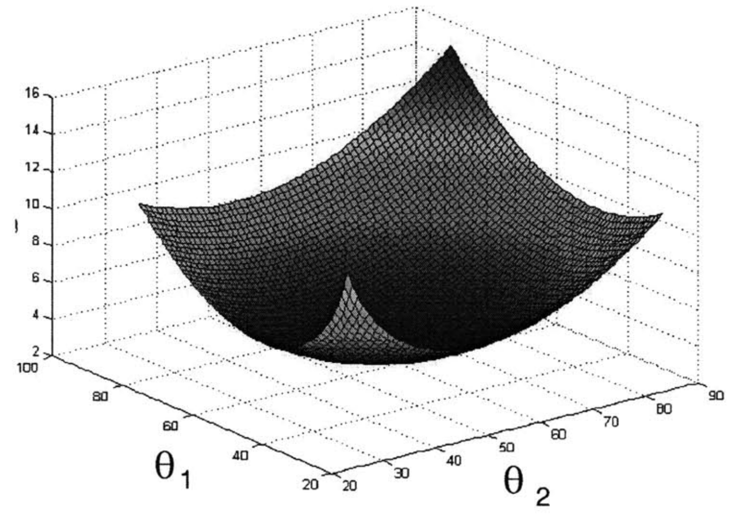

Figure 14: A typical kinematic reconfigurability optimization search space for the SRR.

2.6 Experimental System

Figure 15 shows the Sample Return Rover (SRR), which was used for experimental validation of the kinematic reconfigurability algorithm. The SRR is a 7 kg, 4-wheeled (20 cm diameter) robot that can be equipped with a 2.25 kg three degree of freedom manipulator. The SRR has a passive, instrumented, rocker-type suspension with a spur-gear differential and actively articulated shoulder joints [13]. The parallel linkage suspension, 50 cm total length, permits simultaneous operation of the articulated shoulders, the passive rocker, and the steering actuators. The manipulator and active shoulder joints can be used to reposition the center of mass.

The SRR is equipped with a Pentium 266Mhz/32MB computer along with a boot-able Solid State Disk and a full sensor suite, including an inertial navigation unit that allows the robot to determine the body roll and pitch. The differential rocker angle as well as the shoulder joint angles are determined through potentiometer readings. The SRR has a VxWorks 5.4 real-time operating system and is controlled via a finite state machine architecture [13]. It is also equipped with a 1.5Mb/s Ethernet wireless modem for off-board communication.

Although the SRR has a maximum speed of 21 cm/sec (~0.47 mph), the speed is usually operator limited to 6 cm/sec (~0.13 mph) or less when traveling in rough terrain. Because these are such low speeds, dynamic effects are negligible and static-stability based kinematic reconfiguration is appropriate.

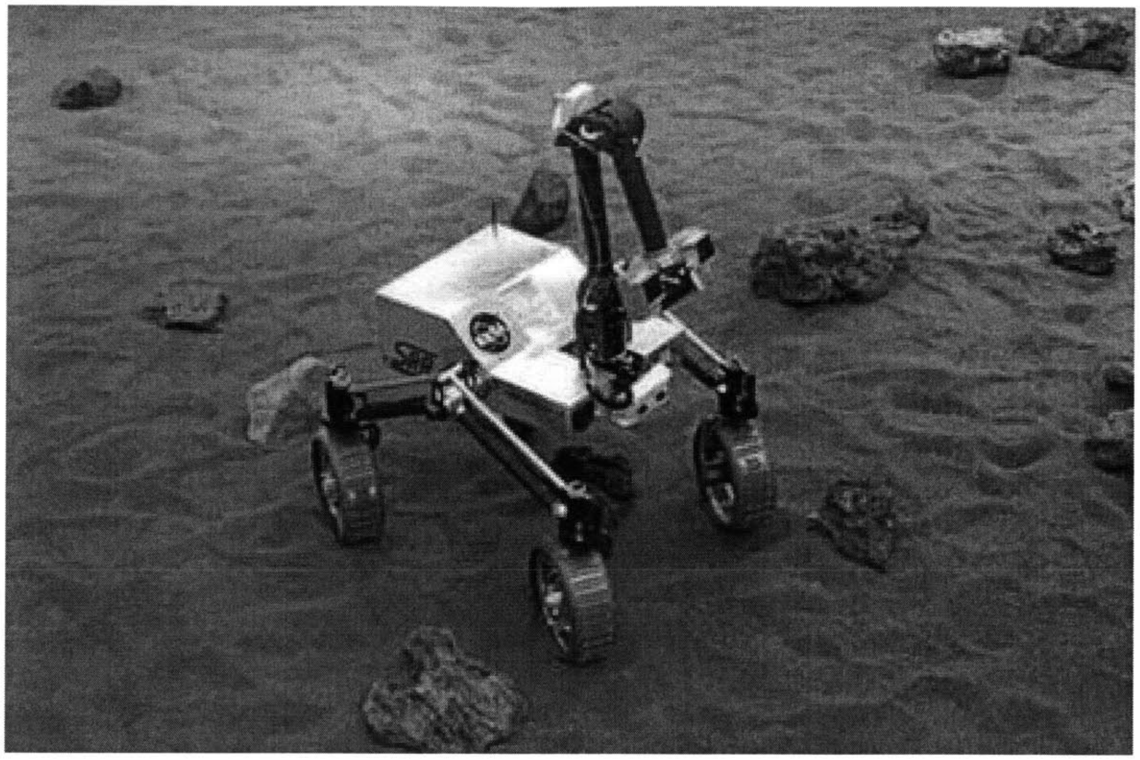

Figure 15: JPL's Sample Return Rover [5].

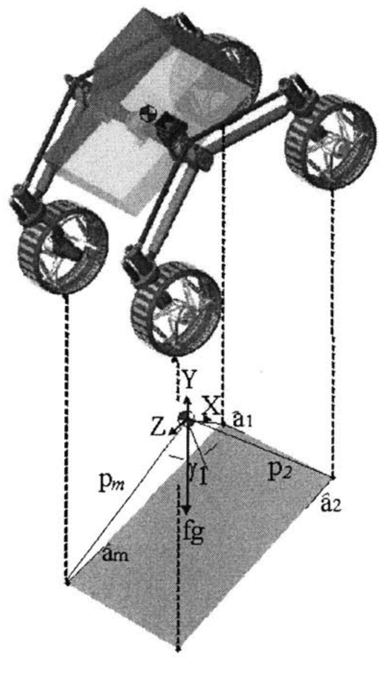

Figure 16: Force-angle stability measure applied to the SRR.

Figure 16 shows how the force-angle stability measure is applied to the SRR. Appendix A reviews the development of the rover kinematics. The optimization performance metric takes the form:

$$\Phi = \sum_{j=1}^{4} \frac{K_j}{\gamma_j} + \sum_{i=1}^{2} K_{i+4} (\theta_i - \theta_i')^2 \tag{12}$$

Note that the stability angles γ_i are a function of the shoulder and manipulator degrees of freedom.

2.7 Simulation Results

A 3D simulation of the SRR operating in rough terrain was written in MATLAB. The rover model was then simulated on representative terrains and allowed to reconfigure. Configuration optimization was performed using a 300 MHz AMD K6 processor. The average processing time for a single constrained optimization computation was 40 μsec. Thus, kinematic reconfiguration is a simple and efficient way of improving system safety in rough terrain.

Typical simulation results are shown in Figure 17. The stability margin, as defined by Equation (12), is plotted for both a reconfigurable system and a conventional, fixed-configuration system. The stability of the reconfigurable system was 37.1% greater than the fixed-configuration system. The fixed-configuration system came within 1.1° of tipover failure. With such a small stability margin, disturbances such as wheel slip could lead to tipover. The reconfigurable system maintained a comfortable $12.5\,^{\circ}$ minimum safety margin.

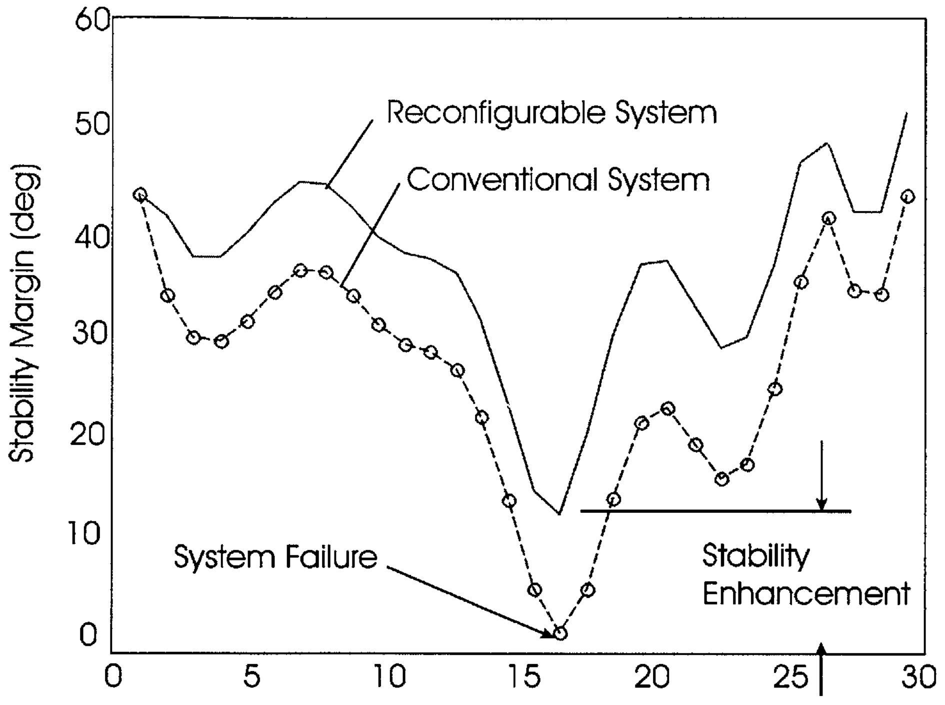

Figure 17: SRR stability margin for a reconfigurable system (solid) and a non-reconfigurable system (dotted) [1].

Figure 18 shows the effects of using the reconfiguration metric to position the manipulator without reconfiguring the suspension. The stability margin is plotted for systems with and without active reconfiguration of the manipulator degrees of freedom. Note that this is a different terrain profile from that used to generate Figure 17. The actively reconfiguring system is 21.1% more stable than the fixed-configuration system. More importantly, the reconfigurable system maintains a minimum 11.5^{o} safety margin, while the conventional system comes within 2.5^{o} of tipover failure.

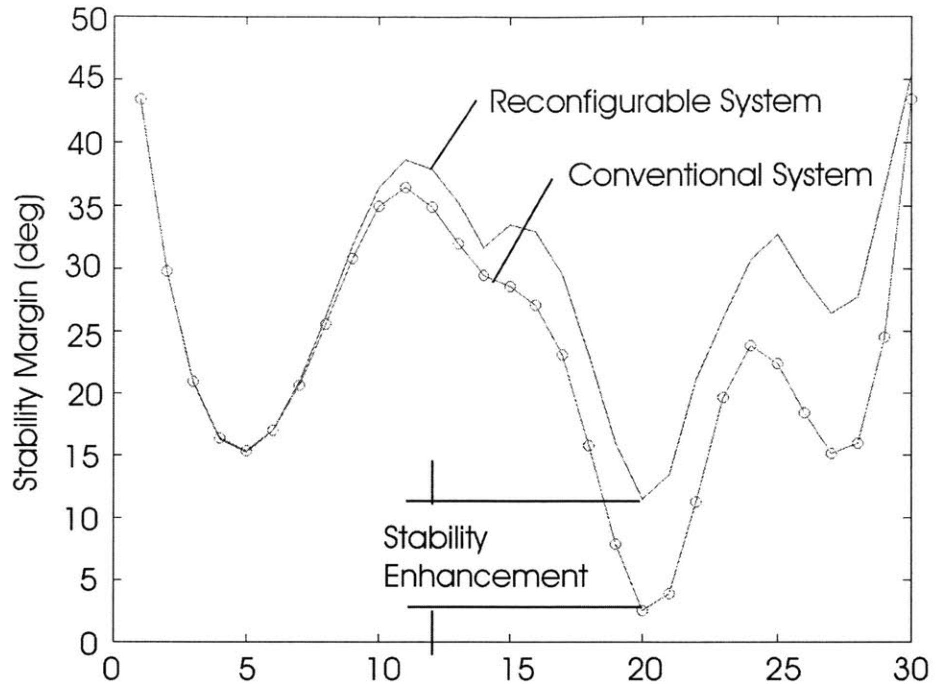

Figure 18: SRR stability margin with only manipulator reconfiguration (solid) and a non-reconfigurable system (dotted).

Figure 19 shows reconfiguration behavior when the SRR is artificially pitched forward. Note that this is a test case simulating descent down a steep slope following manipulator activity at the front of the rover. Also note that a conventional, fixed-configuration rover would undergo tipover failure if passed through the same pitching motion. As can be expected, the reconfigurable rover ends with its shoulders open and its manipulator as far back and as low as joint limits and interference constraints will allow. The transition states are interesting as they show that the rover does not simply move its manipulator arm up and over the body, which would raise the location of the center of mass and potentially cause tipover. Rather it swings the arm *around* and adjusts its shoulder pose during the movement to keep the center of mass on the centerline of the rover body.

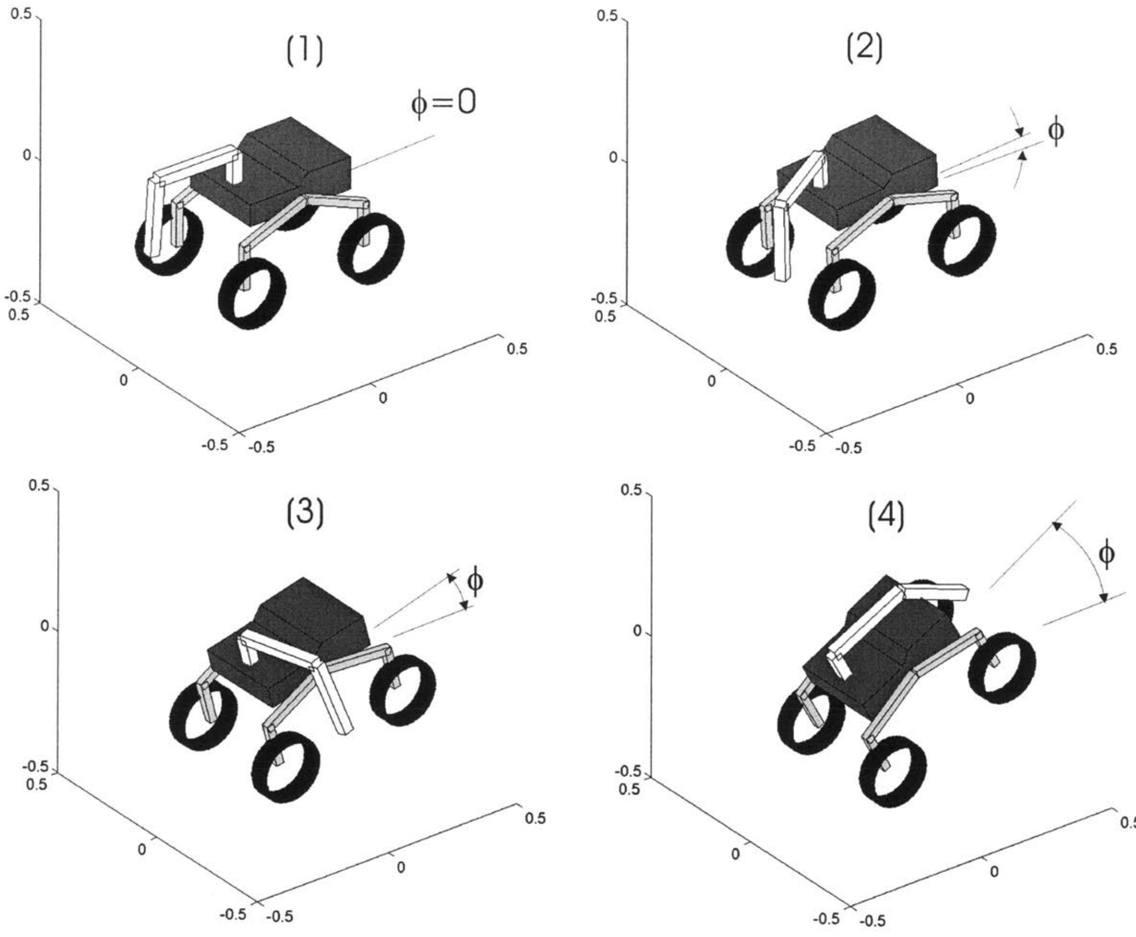

Figure 19: SRR rover pitch simulation with both shoulder and arm reconfiguration.

2.8 Experimental Results

Experimental trials were carried out at the Jet Propulsion Laboratory's Planetary Robotics Laboratory and at the Arroyo Seco in Altadena, California with the assistance of JPL researchers. Kinematic reconfigurability code was written in C to interface with the SRR (described in Section 2.6). This code is presented in Appendix B.

The Planetary Robotics Laboratory is equipped with a large, indoor "sandbox" filled with ruby garnet designed to simulate loose soils. Laboratory trials included traverses over terrain composed of this garnet and small stones (~4-18 inches). They were designed to test the mobility limits of the SRR. The SRR was commanded to travel over difficult terrain that often threatened rover safety. Each trial included a path traverse with a reconfiguring and a non-reconfiguring system. Note that, although the reconfiguration routine is designed for real-time operation permitting simultaneous driving and reconfiguration, at the time of experimental validation, the SRR was equipped with a finite state machine framework that allowed the execution of only one task at a time. Subsequently, the SRR was commanded to travel 10cm between reconfigurations. Representative results from experimental trials are discussed below.

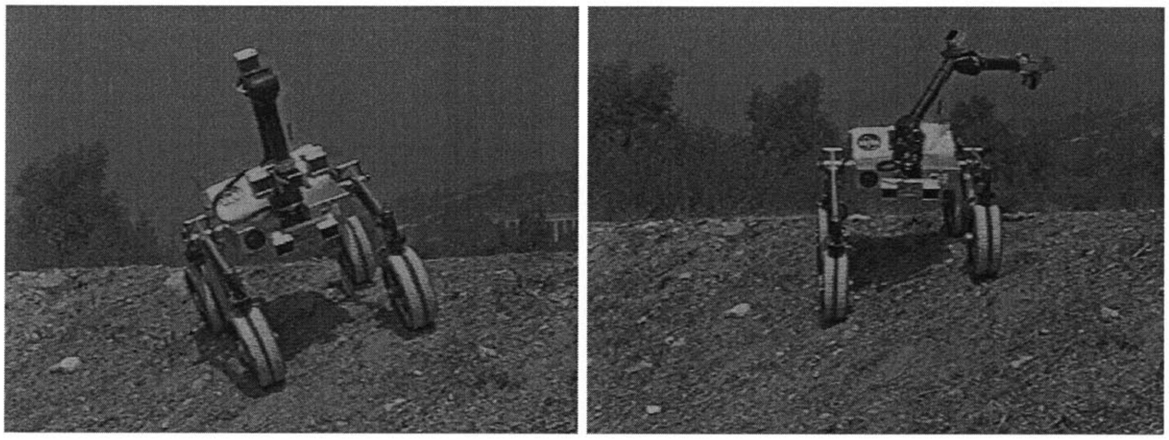

Figure 20: SRR experimental trials in the Arroyo Seco. Non-reconfigurable system (left), reconfigurable system (right).

Figure 20 shows the SRR during an experimental field trial in the Arroyo Seco. The left picture shows a fixed configuration rover. The right picture shows rover behavior with both shoulder and manipulator reconfiguration. The benefits of reconfiguration can be readily seen. Both system safety and path tracking are improved through kinematic reconfigurability. Note that due to the large relative weight of the SRR manipulator, nearly 25% of full rover weight, much less shoulder reconfiguration is required to maintain system stability.

Experimental data from SRR field trials is shown in Figure 21 and Figure 22. Left and right shoulder angle histories are shown in Figure 21. Both reconfigurable and non-reconfigurable system data is shown; small variations in fixed-configuration shoulder angles are due to actuator compliance. Note that shoulder reconfiguration by as much as 30° is required to maintain system stability. Also note that both shoulders remain within the joint limits of $\pm 45^{\circ}$ of their initial values.

Vehicle stability margin for a reconfigurable and a non-reconfigurable system is plotted in Figure 22. Note that the two plots do not match up along the horizontal axis due to different wheel-slip conditions during each traverse. The average stability of the reconfigurable system was 48.1% greater than that of the fixed-configuration system. The non-reconfigurable system came within a precarious 2.1° and 2.5° of tipover during the traverse. The reconfigurable system maintained a minimum 15.0° safety margin. Clearly, kinematic reconfigurability greatly improves vehicle stability in rough terrain.

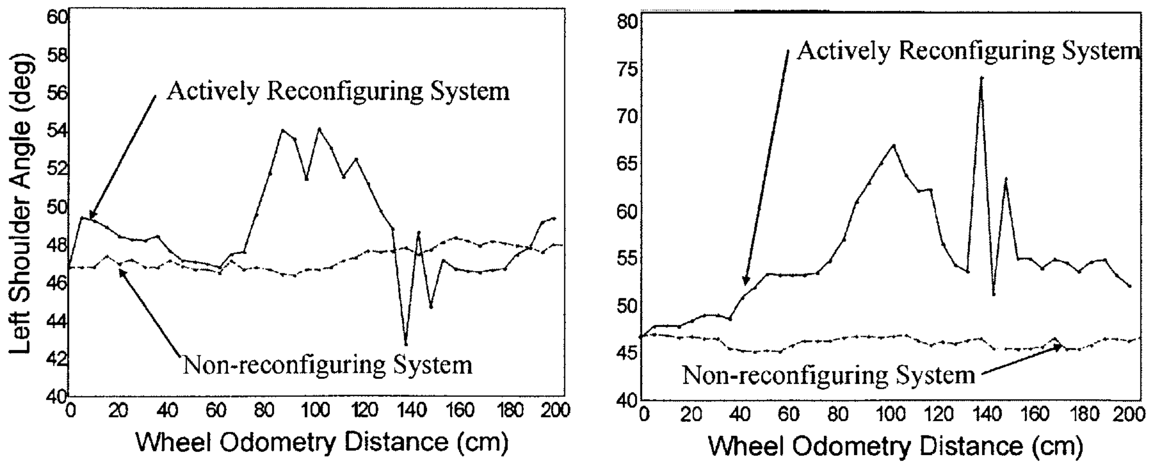

Figure 21: SRR left (a) and right (b) shoulder angles during rough terrain traverse for reconfigurable system (solid) and non-reconfigurable system (dashed) [1].

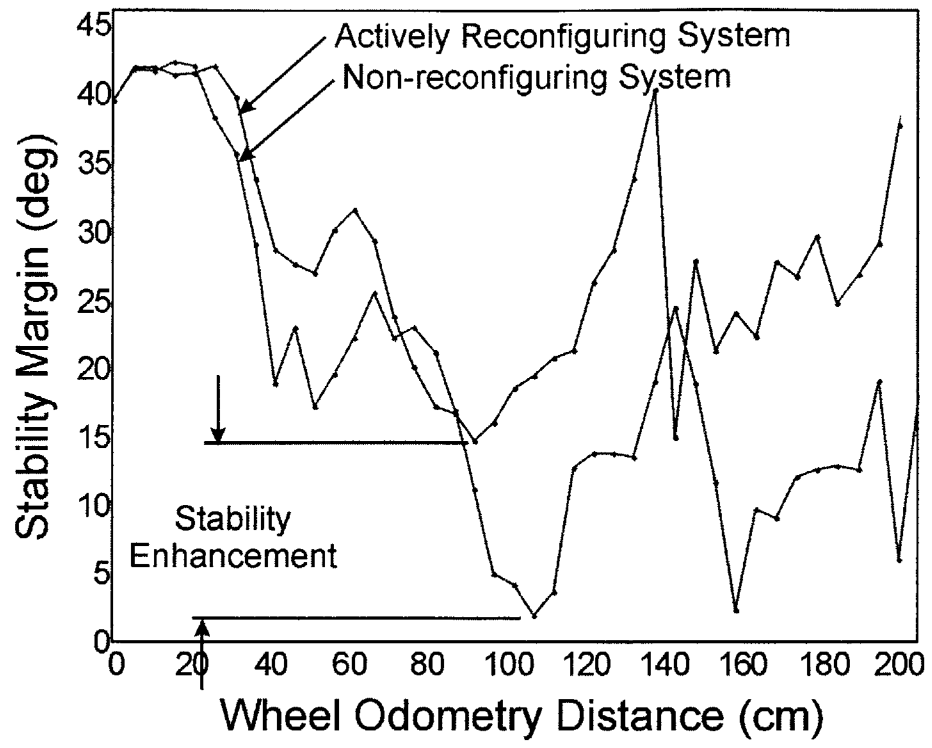

Figure 22: SRR stability margin for reconfigurable and non-reconfigurable system [1].

2.9 Conclusions

This chapter reviewed the use of a kinematic reconfigurability algorithm to increase system stability, and, thus, safety. A method for calculating the optimum vehicle pose was reviewed [1], subject to kinematic constraints and the local terrain profile. The method was then verified in simulation and in field trials with the SRR rover. The average processing time for a single constrained optimization computation was 40 μsec on a 300 MHz AMD K6 processor. Overall, kinematic reconfigurability proved to be an effective way of increasing vehicle safety.

CHAPTER

3 HIGH-SPEED ROUGH TERRAIN TRAVERSAL

3.1 Motivation

In addition to requiring increased rough terrain mobility, future autonomous and semi-autonomous planetary rovers will be required to accomplish much more within their lifetime than today's would permit. They will need to explore areas many times larger than the one explored by Sojourner during the highly successful Mars Pathfinder mission [2]. To accomplish this, rovers will need to operate at high speeds in planetary, rough-terrain environments.

High-speed off-road motion is also important in military applications. Current manned battlefield vehicles, as shown in Figure 23, and laboratory Unmanned Ground Vehicles (UGV's) will have to function faster and more reliably to contend with ever increasing functional demands [42]. Possible tasks include reconnaissance, surveillance, and target acquisition [43]. Each one of these will put the vehicle in jeopardy and might cause mission failure if the system is poorly or conservatively controlled. Aggressive motions such as jumping, see Figure 23, are performed by human military personnel. However, these motions are often not allowed by current automated vehicle control algorithms [20, 23, 27]. A need exists for a behavior planner that allows advanced system motions.

Figure 23: The Chenowth Fast Attack/Light Strike Vehicle used by U.S. SEAL's during Operation Desert Storm [43].

Control of high-speed motion over rough terrain is a difficult problem. A vehicle must perform all perception and path planning activities that slow-moving, quasi-static rovers perform, in addition to maintaining dynamic system safety. An appropriately detailed vehicle model is key to achieving this task. The model is important to controlling speed, handling, and stability of the vehicle. While the research work discussed in Section 1.3.2 offers innovative solutions to this difficult problem, it relies on overly simplified system models and kinematic rolling constraints. The use of an overly simplified system model can lead to incorrect prediction of system performance, which could lead to system failure at high speeds.

Kinematic models have been used in some motion planning algorithms [34, 44, 45]. These algorithms evaluate terrain based on static safety, kinematic, and joint limit analysis, without the need for dynamic analysis. This results in a light computational load. However, these planning algorithms are not applicable to high-speed rough terrain motion, where dynamic effects are important.

Although fully detailed vehicle models exist, they are not used in motion planning algorithms due to their high computational complexity. Motion planning algorithms reduce the system model to a point mass moving along a terrain profile to minimize computational complexity [17, 18, 21, 24, 27]. This, however, ignores any effects that the suspension has on vehicle behavior and could lead to incorrect motion prediction and system failure at high-speeds.

Motion control algorithms recognize the importance of a vehicle's suspension and commonly use three simplified vehicle models to alter vehicle behavior [30, 31, 33]. Figure 24 shows the one degree of freedom (DOF), skyhook, and two DOF models. These are often referred to as quarter-car models. Each of these assumes (1) all mass bearing elements are point masses, (2) the suspensions move only along the vertical axis, and (3) kinematic rolling (constant ground contact and no slip).

Each of these models has unique properties. The one DOF model ignores the unsprung mass of the wheels and considers only road input and body mass motion to determine suspension forces. The skyhook model also ignores any unsprung mass. However it assumes inertial damping. The two DOF model considers both sprung and unsprung mass and uses the relative motions of ground-to-wheel and wheel-to-body to determine the suspension forces.

Although these models are useful for simple motion, they do not capture the salient features of high-speed off-road motion. Both the one DOF and skyhook models ignore the effects of the unsprung mass (wheels). These are important to vehicle handling and stability. The two DOF model, although it considers unsprung mass, still does not account for higher order effects such as rotational inertia and weight shifting due to throttle/brake input, which are important in high-speed dynamics.

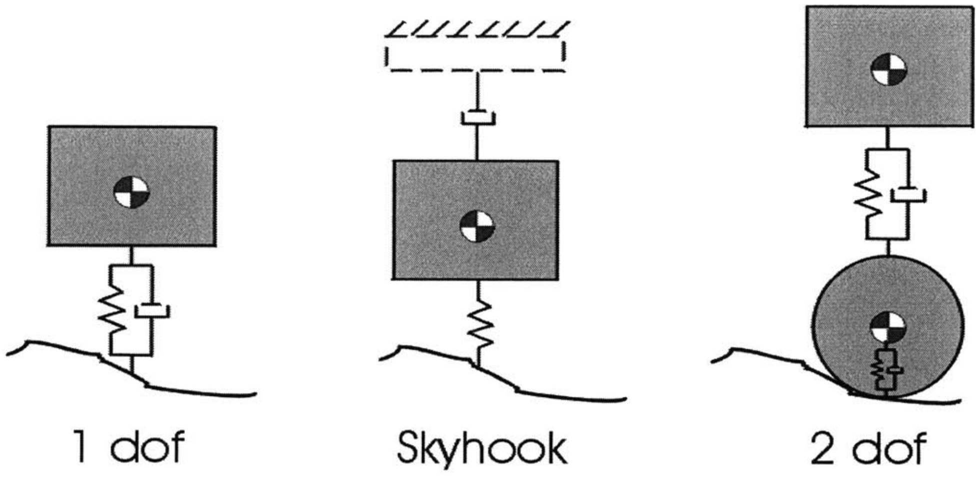

Figure 24: Quarter-car vehicle models.

In the remainder of this chapter, a planar vehicle model is developed which considers suspension characteristics and accounts for rotational inertia and weight shifting due to throttle/brake input. An algorithm is developed which uses this model to determine maximum- and minimum- velocity profiles, defined in Section 3.3, for a given path. The algorithm allows wheel slip and ballistic motion since these effects are often seen in fast off-road vehicles whether they are a mountain bike traveling on singletrack or a racecar in the Paris-Dakar rally which travels over natural, ungroomed terrain [46]. Note that low-speed planetary rovers, which are designed to operate in low gravity environments, can also become airborne.

The algorithm assumes knowledge of the full terrain profile. Due to a high computational load, the algorithm is not suitable for on-line implementation and off-line motion evaluation is required.

3.2 Modeling

A model considering both the mass and inertia of the body and wheels is proposed. This model was chosen because it captures all the effects that the simpler point mass and quarter-car models do, as well as some important higher order effects. Only planar motion is examined in this study. Thus, important 3D effects, such as rollover, are not considered. Figure 25 shows the vehicle model.

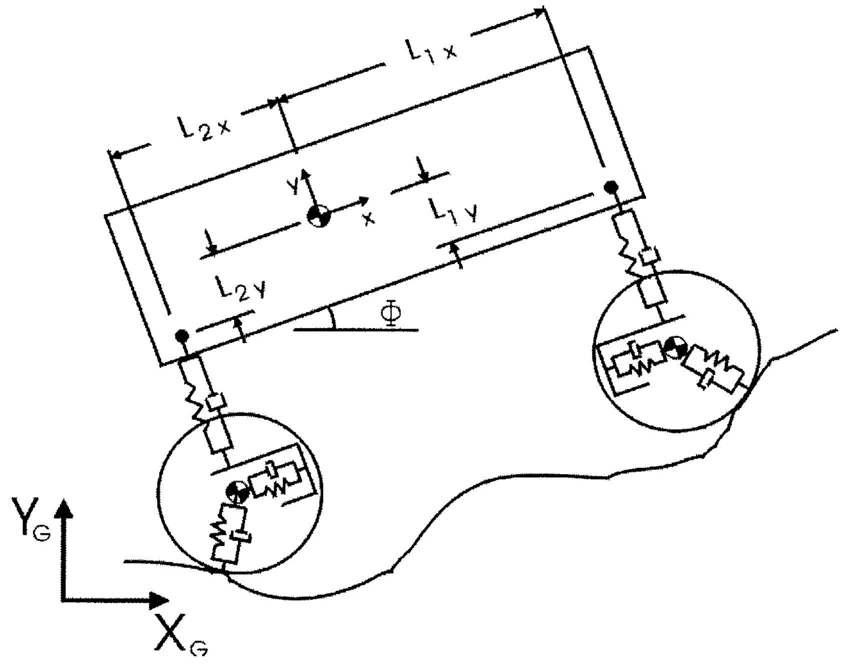

Figure 25: Nine DOF planar vehicle model.

The planar vehicle model is a nine degree of freedom system (two translational and one rotational degree of freedom for the front and rear wheels and the body). Wheel compliance is modeled as a spring-damper element. The body and wheels are coupled by spring-damper elements. Thus, important effects such as rear-end squat and front-end nosedive are captured. These effects are important because they influence traction and ballistic vehicle motion.

The equations of motion for the planar system are derived in Appendix C. A few assumptions are made in their derivation. First, coulomb friction is assumed at the wheel-terrain interface. Second, point contact with the terrain is assumed. This is consistent with the first assumption, since the Coulomb friction law does not consider contact area, just the contact force. Third, although the deflection of each tire is considered in determining wheel-terrain interaction forces, it is not considered for the calculation of wheel inertia, i.e., the wheels are assumed to remain round.

Vehicle parameters used in simulation are listed in Table 1. These parameters were chosen because they are characteristic of a passenger vehicle.

Vehicle Mass	534 kg
Wheel Mass	36.6 kg
Suspension Stiffness (x direction)	1.87×10^6 N/m
Suspension Stiffness (y direction)	1.87×10^4 N/m
Wheel Stiffness	1.84×10^5 N/m
Suspension Damping Coefficient	1398 Ns/m
Wheel Damping Coefficient	1398 Ns/m

Table 1: Vehicle parameters used in simulation [31].

3.3 Maximum and Minimum Velocity Trajectories

Maximum and minimum velocity trajectories are a way of determining safe vehicle motions and evaluating a path's traversability. A maximum velocity trajectory gives an estimate of the maximum velocity, at discrete points along the path, that allows the vehicle to reach the goal point and is consistent with the vehicle state at that point, the dynamic equations of motion, and all traction and torque limits. This is an estimate because it displays only the body velocity and does not explicitly display information related to the suspension state.

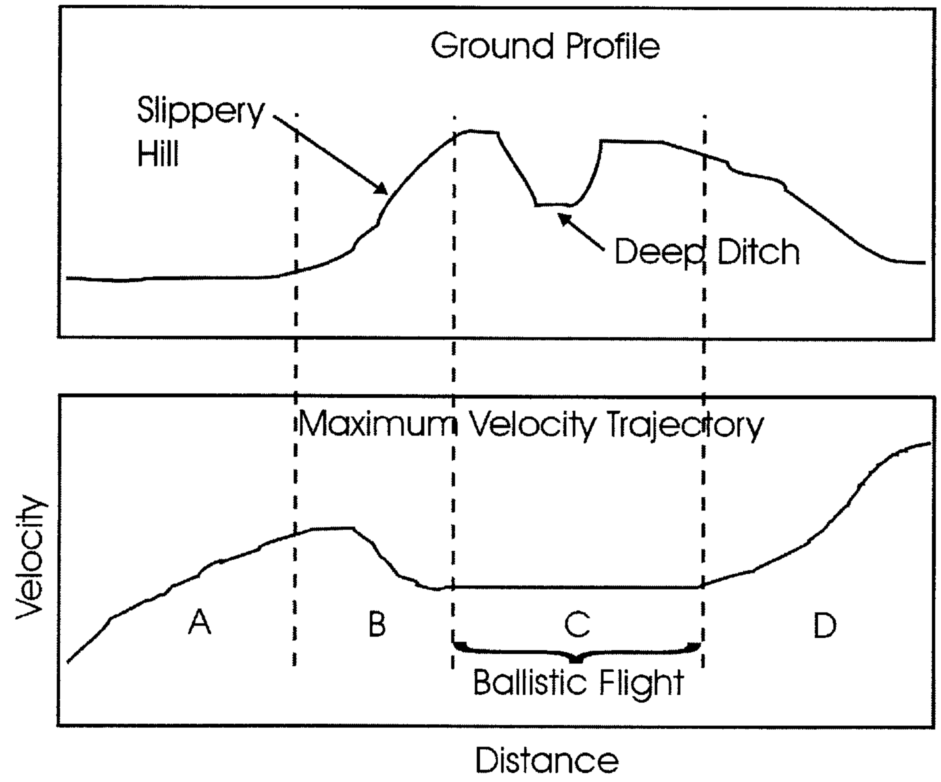

Figure 26: A representation of a maximum velocity trajectory.

An illustrative example of such a trajectory is shown in Figure 26. Note that this is a not an actual data set. The terrain profile includes a steep, slippery hill and a ditch that cannot be traversed and must be jumped. This terrain would be deemed untraversable by most rough terrain motion planning algorithms due to static stability violations. However, if wheel slip and ballistic motion are allowed, the terrain can be crossed.

The observed maximum velocity trajectory includes four key regions. The first region includes an acceleration that is at the traction and torque limits. This action is denoted as A. The next region, B, includes travel over the steep, slippery hill. This region is classified by a velocity drop since the available tractive force is lower than the opposing gravity force. A ballistic flight region, denoted as C, begins once the vehicle has reached the top of the hill. The ability to jump allows the vehicle to clear the statically impassible ditch. Note that the velocity shown on the graph, defined as vehicle velocity along the horizontal axis, remains constant in this region. This is true since the only acceleration (due to gravity), is normal to this horizontal axis. The final section, denoted D, sees a sharp increase in speed as the vehicle travels down hill.

A minimum velocity trajectory gives an estimate of the minimum velocity, at discrete points along the path, that allows a vehicle to reach the goal from that point and is consistent with the vehicle state at that point, the dynamic equations of motion, and all traction and torque limits. This is an estimate because it displays only the body velocity and does not explicitly display information related to the suspension state. This information is trivial in flat terrain where a vehicle can stop anywhere and restart again to reach the goal. However, in rough terrain a vehicle might have to "get a running start" to successfully traverse a hill or jump a ditch.

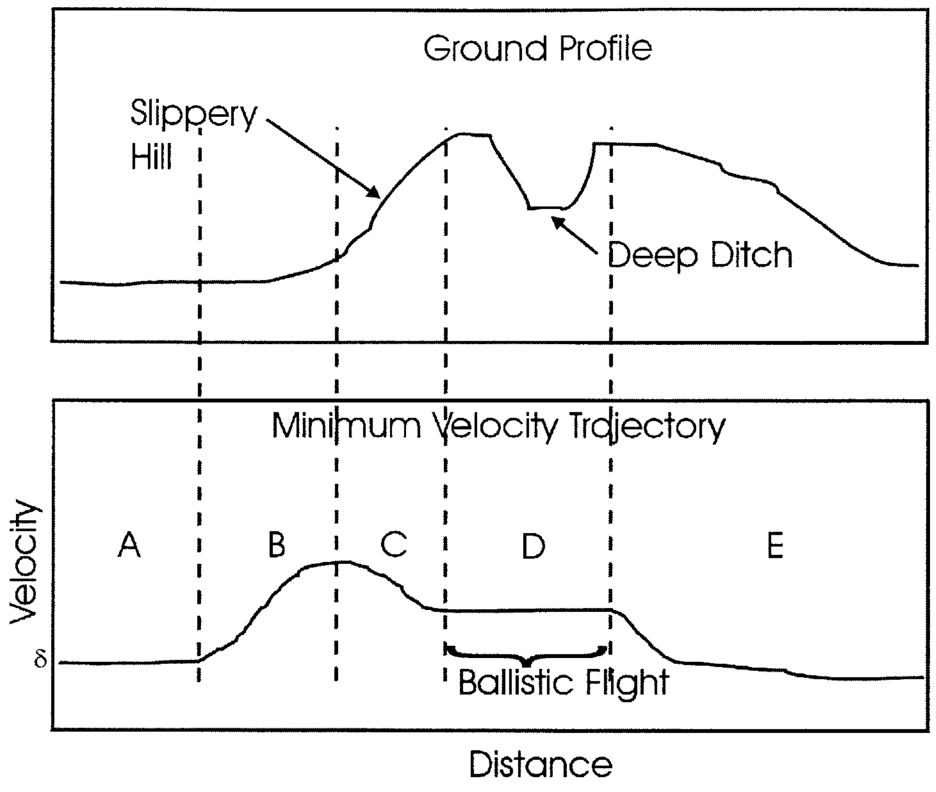

Figure 27: A representation of a minimum velocity trajectory.

Figure 27 shows a ground profile and a minimum velocity trajectory. Again, this is not an actual data set. The trajectory may be divided into five parts. Travel over benign terrain is shown in part A. The velocity is a small δ. Part B, although still benign terrain, shows an increase in speed required to safely traverse the upcoming hill and ditch. Parts C and D show a decrease in speed during uphill travel and constant velocity during ballistic flight, respectively. Note that ballistic flight is required even for this minimum velocity trajectory since the ditch cannot be traversed due to static stability violations. Finally, part E shows a deceleration back down to the velocity δ.

Maximum and minimum velocity trajectories do not explicitly display information related to the vehicle suspension state. In order to allow an intuitive, graphical visualization, the effects of suspension initial conditions on a maximum velocity trajectory are shown in Figure 28. The trajectory generation procedure is discussed in Section 3.3.3. Note that the data sets were created with different initial conditions for the suspension, but identical initial conditions for the vehicle body. These simulations used a vehicle with a three meter wheelbase; other vehicle properties are listed in Table 1. Due to different initial conditions of the suspension states, the maximum velocity trajectories do not converge to the same curve. Note, however, that the profiles are similar in shape. This is expected since the vehicle body dynamics dominate the system response.

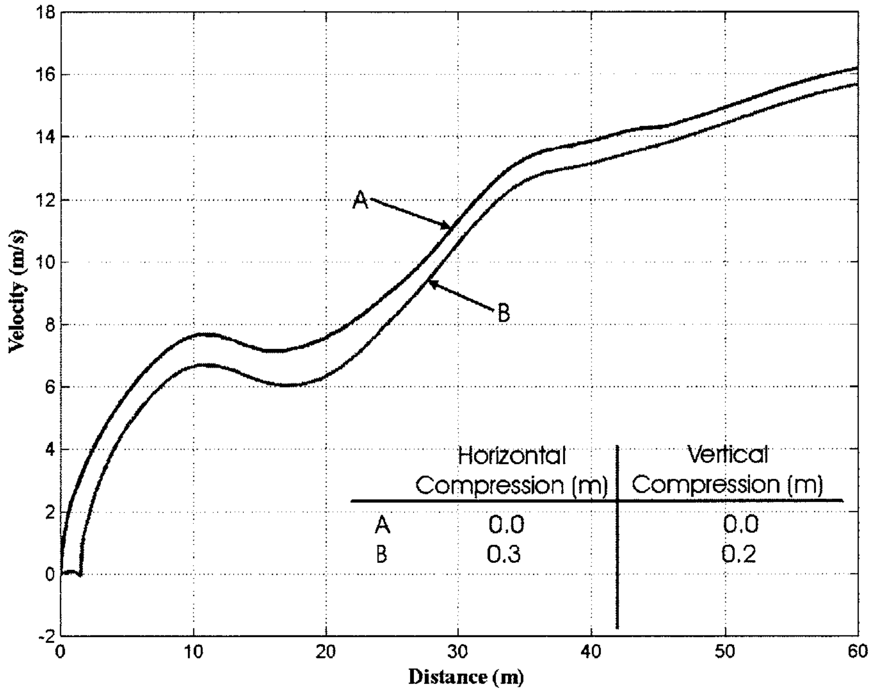

Figure 28: Maximum velocity trajectories with identical initial conditions on the vehicle body but differing suspension initial conditions.

The effect of vehicle body initial conditions on the maximum velocity trajectory is shown in Figure 29. All profiles used the same initial suspension state, but different initial body velocities. The trajectory generation procedure is discussed in Section 3.3.3. The figure shows that trajectory profiles converge to a band similar to that seen in Figure

28. Such convergence may be attributed to differences in suspension state that are experienced during terrain traversal. Due to this lack of convergence, the maximum and minimum trajectories should be used as guidelines and not as true boundaries.

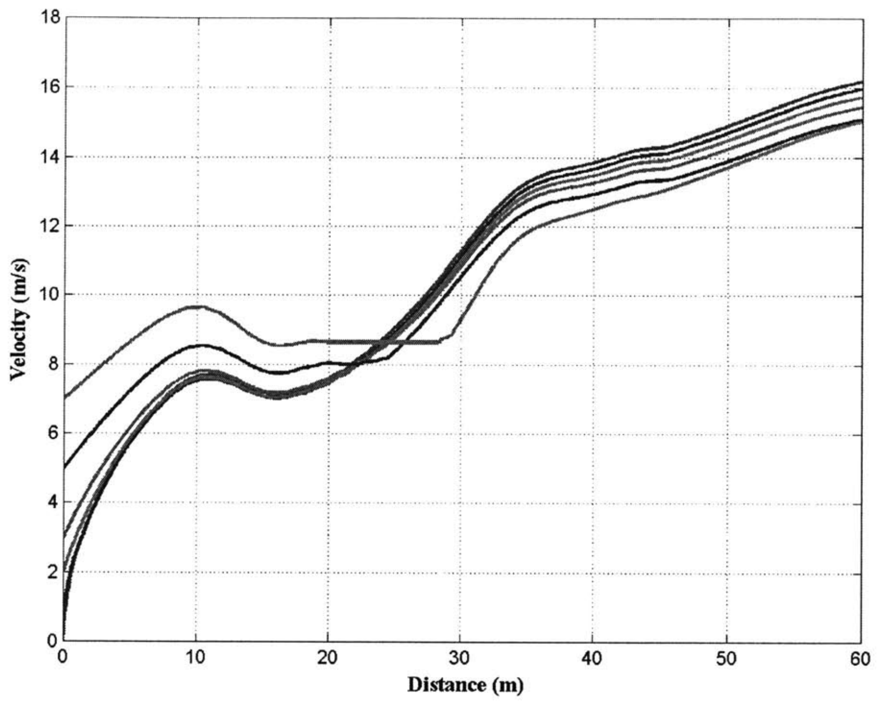

Figure 29: Maximum velocity trajectories constructed with different initial velocities.

The maximum and minimum velocity trajectories may also be used as a measure of path traversability. Benign terrain will have widely separated maximum and minimum velocity trajectories. Rough terrain will have the trajectories close to each other due to traction, torque, and safety constraints. Note that safety between the curves is not guaranteed. Cases of failure that occur between the maximum and minimum velocity trajectories are discussed later in this section.

Figure 30 shows the maximum and minimum velocity trajectories for the ground profile used in Figure 26 and Figure 27. Note that these are not actual data sets. Also note that critical regions, such as the ones before the onset of ballistic motion, often have narrow zones of safe operation. Maximum and minimum velocity trajectories may be constructed with matching initial conditions, as in Figure 30, or with differing initial conditions, as in Figure 31. Differing initial conditions, zero velocity for the minimum velocity trajectory and traction and torque limited velocity for the maximum velocity trajectory, may be used to assess the traversability of a given terrain path.

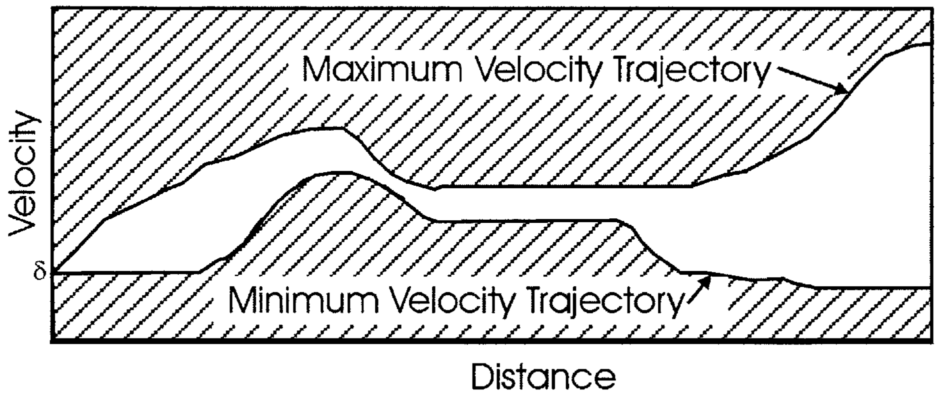

Figure 30: Maximum and minimum velocity trajectories.

An issue not discussed thus far is that of islands of inadmissibility [47]. An island of inadmissibility is a region of unsafe operation that occurs between the maximum and minimum velocity trajectories. An example of such an island appearing in a velocity trajectory plot is shown in Figure 31. Note that this is not an actual data set. Islands of inadmissibility are the result of special combinations of terrain profile and vehicle state. To fully identify the shape of such an island an exhaustive search of all possible motion states would be required.

An example of an island of inadmissibility is shown in Figure 32 and Figure 33. The ground profile, as shown in Figure 32, is a set of two ramps. Figure 32 shows the body center of mass paths for three velocity cases. Snapshots from a dynamic simulation of vehicle behavior for the minimum velocity trajectory and a velocity trajectory passing through the island of inadmissibility are shown in Figure 33. The trajectory generation procedure is described in Section 3.3.3. The minimum velocity trajectory allows the vehicle to jump over the first ramp, land at the foot of the second, and safely jump off of the second ramp. The maximum velocity trajectory allows the vehicle to leap over the second ramp and land safely. A velocity profile passing through the island of inadmissibility has the vehicle partially land on the second ramp (only the rear wheels land), thus causing the car to flip over and fail.

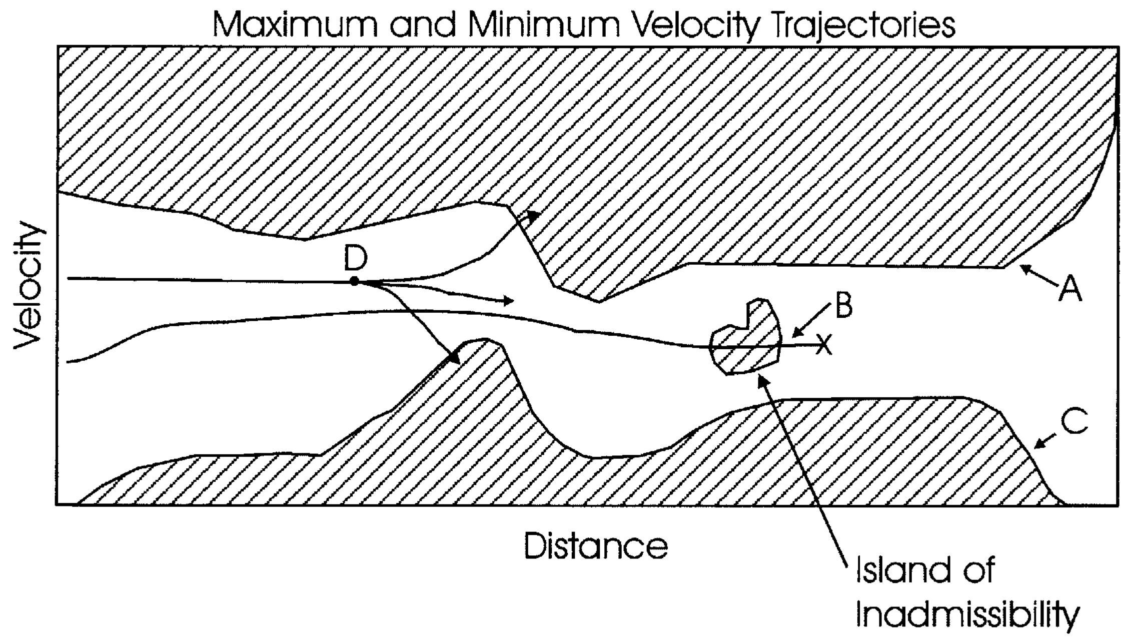
Maximum and Minimum Velocity Trajectories
Velocity
D
B
A
C
Distance
Island of Inadmissibility

Figure 31: An island of inadmissibility in between the maximum and minimum velocity trajectories.

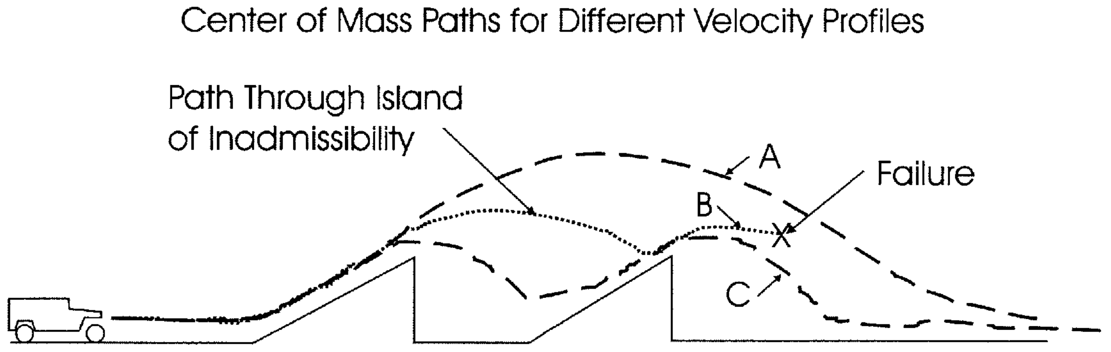
Center of Mass Paths for Different Velocity Profiles
Path Through Island of Inadmissibility
A
Failure
B
C

Figure 32: Center of mass paths for maximum, minimum, and "through island of inadmissibility" velocity trajectories.

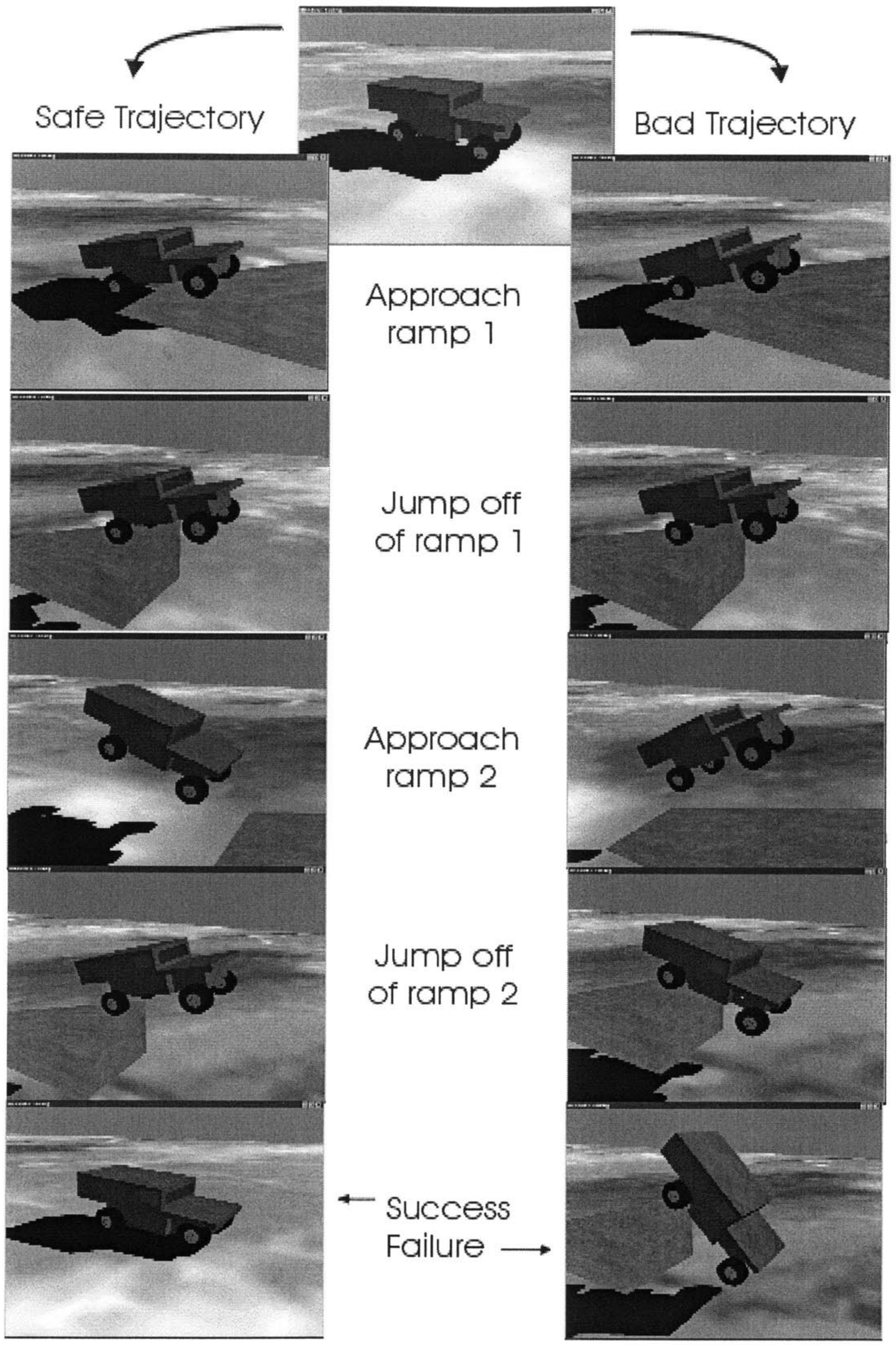

Figure 33: Island of inadmissibility example, two ramp jump. Minimum velocity trajectory, left. Velocity trajectory through island of inadmissibility, right.

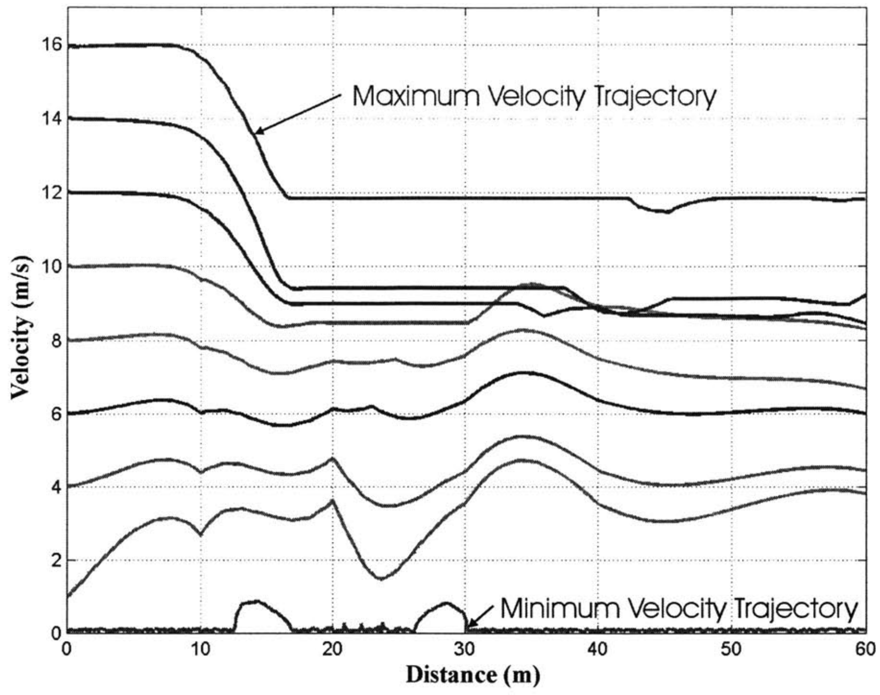

Figure 34: Safe velocity profiles passing between the maximum and minimum velocity trajectories.

Not all velocity profiles between the maximum and minimum trajectories, which avoid an island of inadmissibility, may be proven safe due to the interaction of the body and wheel masses. Thus, an arbitrary velocity profile, denoted D in Figure 31, may penetrate one of the bounding trajectories and result in vehicle failure. Figure 34 shows a sample of safe velocity profiles in support of the author's conjecture that the maximum and minimum velocity trajectories may be used as a measure of path traversability.

3.3.1 Failure Criteria

A clear definition of failure is required to determine the maximum and minimum velocity trajectories. Three failure modes are considered: stability, rearward slip, and stall, see Figure 35.

The stability criterion in 2D is reduced to a limit on the vehicle pitch, ϕ, where failure occurs if

$$\phi > \phi_{\max}$$

where $\phi_{\max}$ is a user defined maximum pitch angle. Note that comprehensive stability criterion, such as the one presented in Section 2.4 may be used.

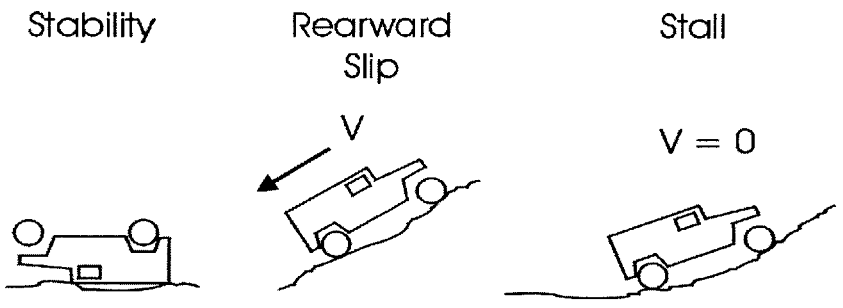

Figure 35: Vehicle Failure Modes.

The rearward slip criterion is defined as vehicle motion away from the goal. Note that slip towards the goal is not a failure. If velocity, V, towards the goal is defined as positive, then failure occurs if:

$$V < 0$$

The final failure mode is vehicle stall. This occurs when the available tractive force is balanced by gravity. This is characterized by a lack of progress along the path, $V = 0$, along with significant wheel slip.

3.3.2 Algorithm Assumptions

A few assumptions are made in generating velocity trajectories. The first is point contact at the wheel-ground interface. This assumption is used in conjunction with a

Coulomb friction model of the terrain. Since contact area is not needed in determining the available tractive force, point contact is a simple and consistent assumption.

Second, no air resistance is considered. Although air resistance plays an important role in extreme or very high-speed motion, it is not significant in the motions considered in this work. However, this effect can be easily incorporated in the general model and algorithm.

Third, direct torque specification instead of a velocity controller is used in generating the velocity profiles. A velocity controller is not used because it would contribute to the dynamics of the system. Figure 36 shows a general control system with a controller transfer function, G_C, and the vehicle (plant) transfer function, G_P. If a controller is used, the open loop transfer function, H, becomes

$$H = G_C G_P$$

with the controller dynamics affecting the output. However, if no controller is used, only the plant transfer function G_P, the one of interest, remains. A side benefit of this approach is the generation of an ideal input trajectory, which an infinitely stiff controller would be able to follow.

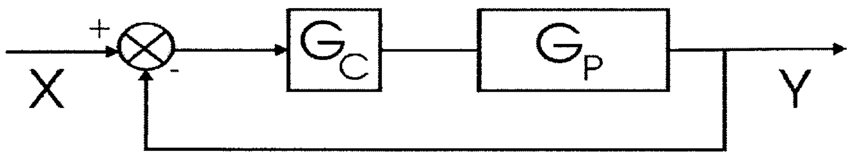

Figure 36: General negative feedback control system.

Two additional assumptions are made regarding wheel torque. First, both wheels are assigned the same input torque value. This is done to reduce the computational complexity of the velocity trajectory generation process. The second is that only enough torque to compensate for wheel bearing friction is assigned during ballistic flight. This is done to prevent artificially pitching the vehicle.

The final assumption is knowledge of the full terrain profile. This may be obtained from satellite imagery that gives sub 1 meter resolution (often ~15 cm) [48].

3.3.3 Velocity Trajectory Generation

The maximum and minimum velocity trajectories are obtained using a depth-first search [49]. The terrain profile, S, is decomposed into n segments such that $S = \{s_1, s_2, \ldots, s_n\}$. Each segment s_i, is bounded by points p_i and p_{i+1}, $i = \{1, \ldots, n\}$. The terrain segments are represented as cubic splines. The actuation space (i.e., torque available for acceleration and braking) is decomposed into m segments such that $Q = \{q_1, q_2, \ldots, q_m\}$, see Figure 37. The division numbers n and m are user specified to suit the detail required.

Knowledge of the initial vehicle state, R_1, is assumed. The vehicle state is defined as:

$$R = [\overline{X}_B, \overline{X}_{w1}, \overline{X}_{w2}]^T$$

where the body state vector is:

$$\overline{X}_B = [x_B, y_B, \phi_B, \dot{x}_B, \dot{y}_B, \dot{\phi}_B]^T$$

and the first and second wheel state vectors are:

$$\overline{X}_{w1} = [x_{w1}, y_{w1}, \theta_{w1}, \dot{x}_{w1}, \dot{y}_{w1}, \dot{\theta}_{w1}]^T$$

$$\overline{X}_{w2} = [x_{w2}, y_{w2}, \theta_{w2}, \dot{x}_{w2}, \dot{y}_{w2}, \dot{\theta}_{w2}]^T$$

where the vectors are the planar position and velocity components of each mass and inertia bearing element.

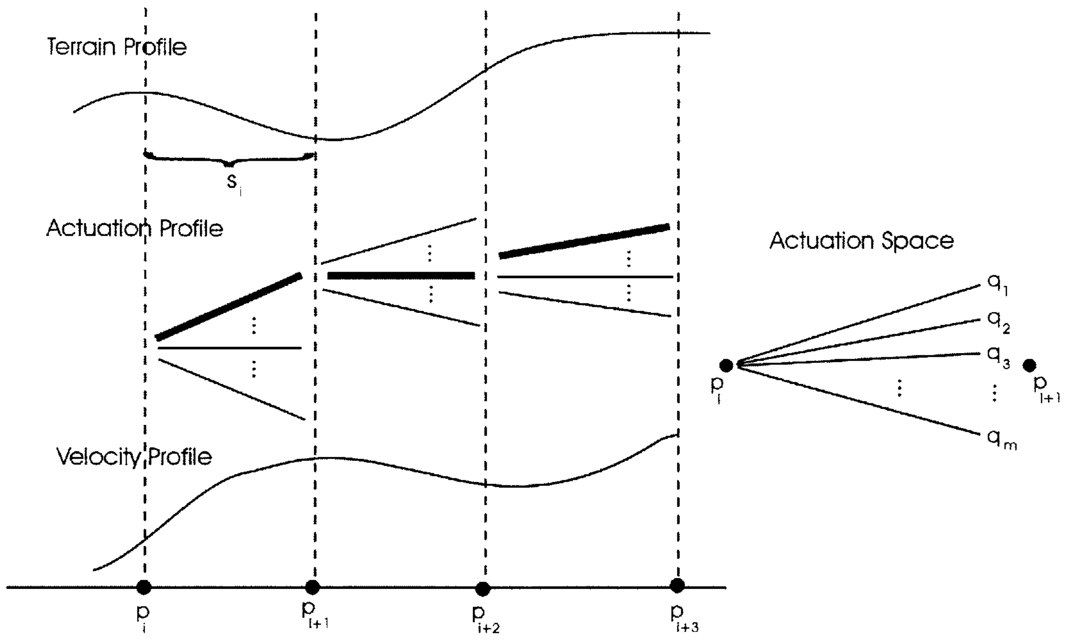

Figure 37: Velocity profile generation schematic.

The vehicle model is then "driven" over the first terrain segment with an assigned torque value q_j, $j = 1 \ldots m$. Note that both wheels see the same q_j. The first q_j that allows the vehicle to successfully travel from point p_1 to p_2 is then stored in memory. Note that the q_j are ordered from lowest to highest for a minimum velocity trajectory and highest to lowest for a maximum velocity trajectory. The vehicle state at the end of the terrain segment is also stored in memory R_2. This state becomes the starting state for the next terrain segment. Once again, the actuation space is searched for a q_j that allows the vehicle to travel from p_2 to p_3 with the starting state vector R_2. This process is repeated for each segment s_i, $i = 1 \ldots n$. A full actuation profile,

$$\tilde{Q} = \{q_a, q_b, \ldots, q_c\},\ a,b,c \in \{1,m\}$$

and a continuous state trajectory,

$$\tilde{R} = \{R_1, R_2, \ldots, R_n\},$$

are thus generated. Note that part of the trajectory is a continuous velocity profile, see Figure 37.

The first successful q_j for the i^{th} terrain segment might not provide appropriate starting conditions for a successful traverse of the *i+1* segment. In such an instance, the i^{th} terrain segment is then re-evaluated with q_{j+1} until a successful traverse of the *i+1* segment or a saturation of the actuators. Sequential back-ups and exploration of the actuation space are performed until the full path is successfully traversed. Note that the appropriate starting state vector is used for each point p_i, $i = 1 \ldots n$.

3.4 Results

Figure 38 shows a single hill terrain profile that was examined for traversability. A vehicle model with 0.5 m radius wheels and a 3.0 m wheelbase was used for evaluation. The terrain was divided into *n*=600 segments and the actuation space was divided into *m*=80 segments. The Coulomb friction coefficient was set to $\mu = 0.7$ and the wheel bearing viscous damping coefficient was set to 5.0 Ns/m. The total time for calculating both velocity profiles was 18 minutes and 10 seconds on a 550 MHz Pentium III computer. If extrapolated into 3D, doubling the number of degrees of freedom and adding an extra control variable (steering), one might expect an order of magnitude increase in the time required to calculate these trajectories.

The planar maximum and minimum velocity trajectories are shown in Figure 39. The actuation profile for the minimum velocity trajectory, $\tilde{Q} = \{{}^1q_a, {}^2q_b, \ldots, {}^nq_c\}$, is shown in Figure 40. Both profiles are created with matching initial conditions. Note that although the terrain appears simple, it is untraversable if slip is not allowed. Two interesting regions appear in the minimum velocity profile. The first, shows that a "running start" is needed to successfully climb the large hill. This is due to a combination of wheel slip and actuator saturation. The second region shows slip that occurs on the far side of the same hill. Note that the vehicle is applying the maximum braking torque in this region to reduce the velocity. The maximum velocity trajectory shows an expected slow-down while the vehicle is going up the hill. This is expected since the vehicle is traveling at maximum effort before it begins to climb the hill. The actuation profile for this trajectory is shown in Figure 41. The actuation profile reveals two regions of ballistic motion. The first region is characterized by two short losses of ground contact due to traversal of the large hill. The second is due to the high-speed traverse of the second hill.

The shapes of the actuation profiles are also interesting. Note the similarity between the minimum velocity actuation profile, shown in Figure 40, and the terrain slope profile, shown in Figure 42. This is expected as the motion algorithm is designed to find the lowest torque required to reach the goal; torque will decrease on the down-slope and increase on the up-slope. The actuation profile for the maximum velocity trajectory is shown in Figure 41. Torque is applied during ballistic motion to overcome wheel bearing resistance which is included in the model.

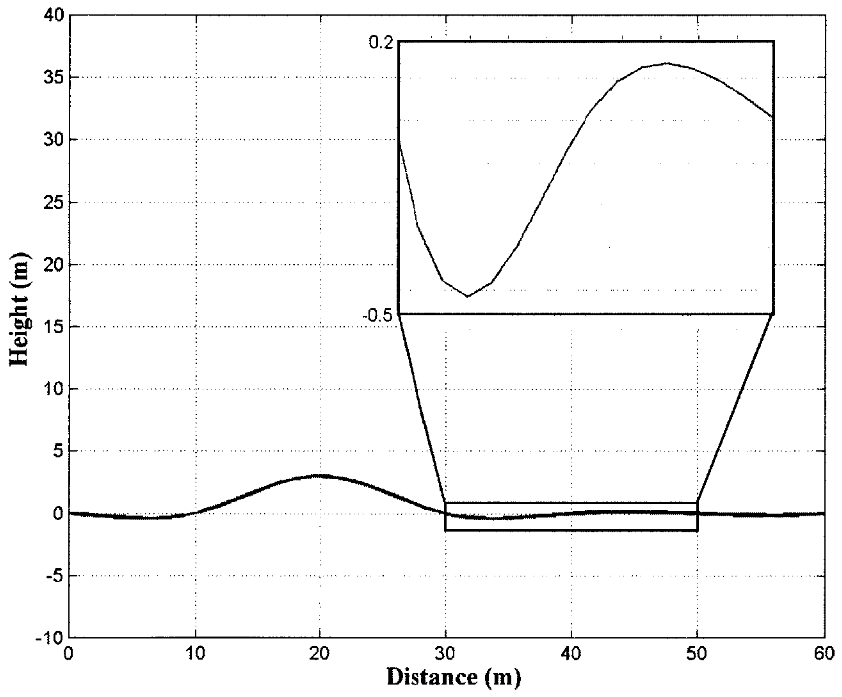

Figure 38: Single hill terrain profile.

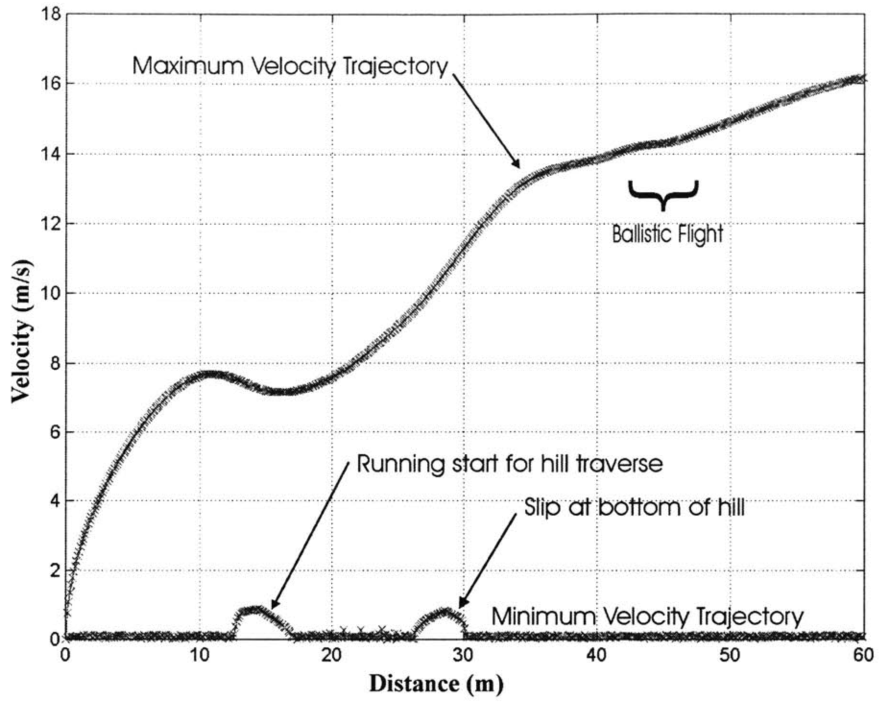

Figure 39: Single hill maximum and minimum velocity trajectories.

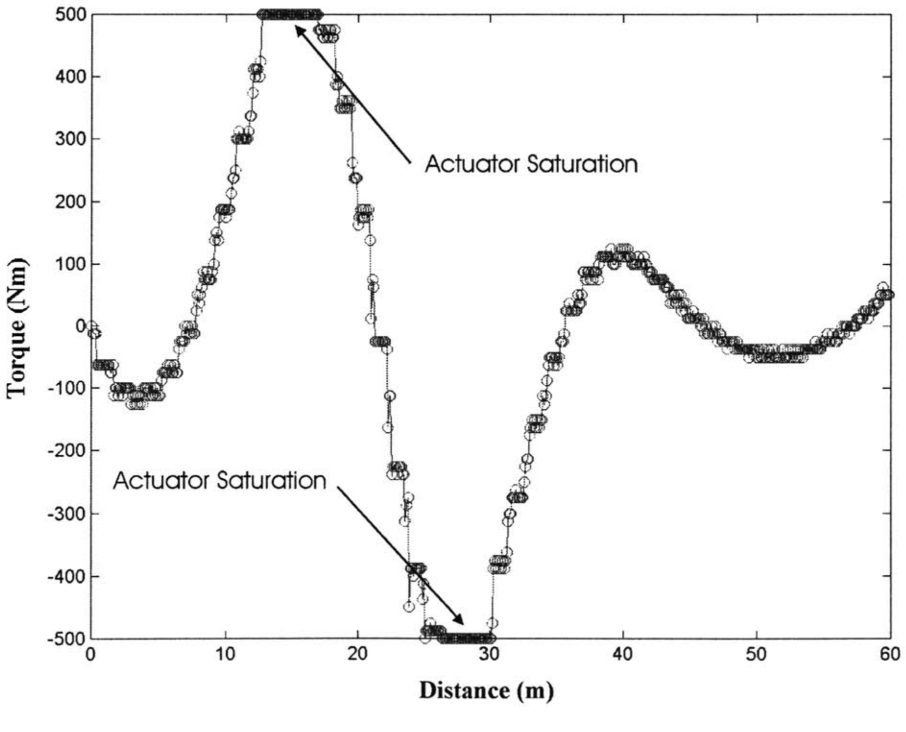

Figure 40: Actuation history for the minimum velocity trajectory.

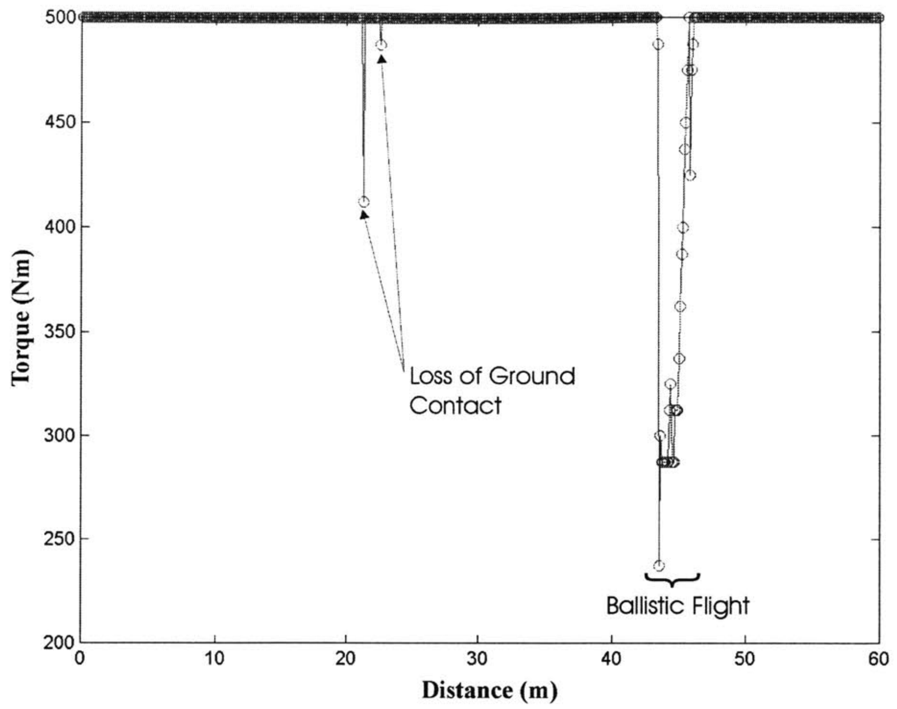

Figure 41: Actuation history for the maximum velocity trajectory.

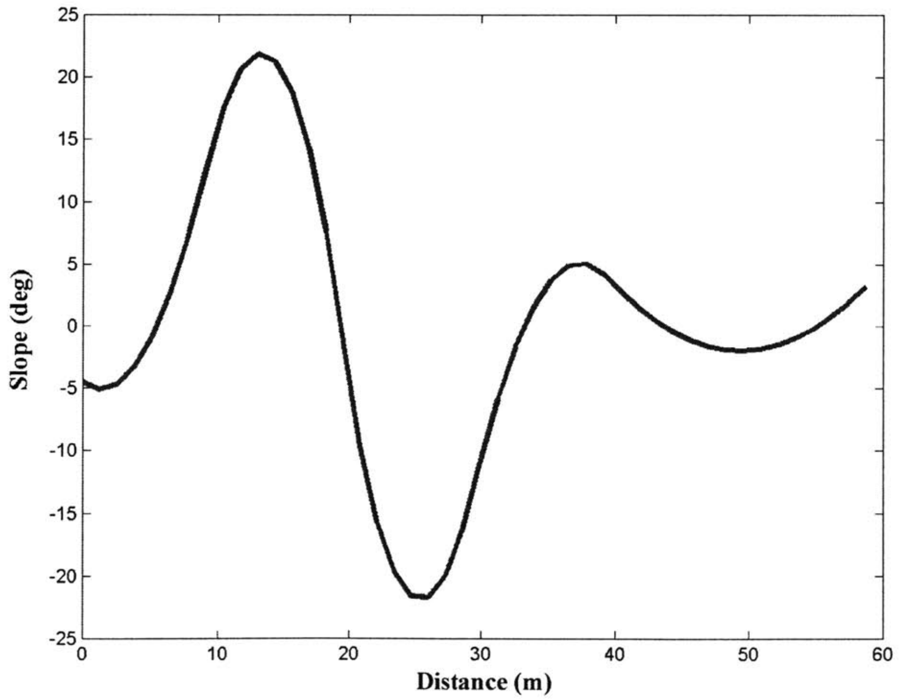

Figure 42: Single hill slope profile.

A second example is provided to illustrate the difference in results obtained by employing two different vehicle models. A comparison is made between work presented in this thesis, which uses a nine DOF vehicle model, and a planar version of the approach presented by Shiller and Gwo [27], which uses a point mass model. Path evaluation is performed using a 3 meter long rover. Desired boundary states are specified for all velocity profiles. The terrain profile, along with the slope history is shown in Figure 43. Note that this is a simple terrain profile that is easily evaluated with most vehicle models. As was presented in Section 3.1, terrain evaluation is carried out to satisfy engine torque, sliding, contact, and pitch-over constraints. The time optimum velocity profile, generated according to [27], along with constraint limit curves is shown in Figure 44. Maximum velocity profiles generated using the more complex model are shown in Figure 45. Note that velocity profiles with and without interaction constraints (slip and pitch-over) are presented.

Several differences may be seen among the three velocity profiles. First, Figure 45 shows that a higher overall velocity and shorter path traversal time is obtained when ballistic motion is permitted. Second, the constrained velocity profile generated using the complex model, Figure 45 (b), is much lower than the one generated using the simpler model, Figure 44. This may be attributed to the appearance of higher order effects captured in the complex model, such as load transfer due to vehicle pitching and the interaction of the three various vehicle masses. In order to keep the no-slip and constant contact constraints, a real vehicle would have to travel much slower than calculated in [27]. Finally, the maximum velocity profile, Figure 45 (a), which appears to be nearly identical to the time optimum profile from [27], Figure 44, contains a brief region of ballistic flight. Ballistic flight, as defined here, is minimally characterized by one wheel leaving the ground. Thus, in order to accurately capture vehicle motion, an appropriately complex vehicle model must be used.

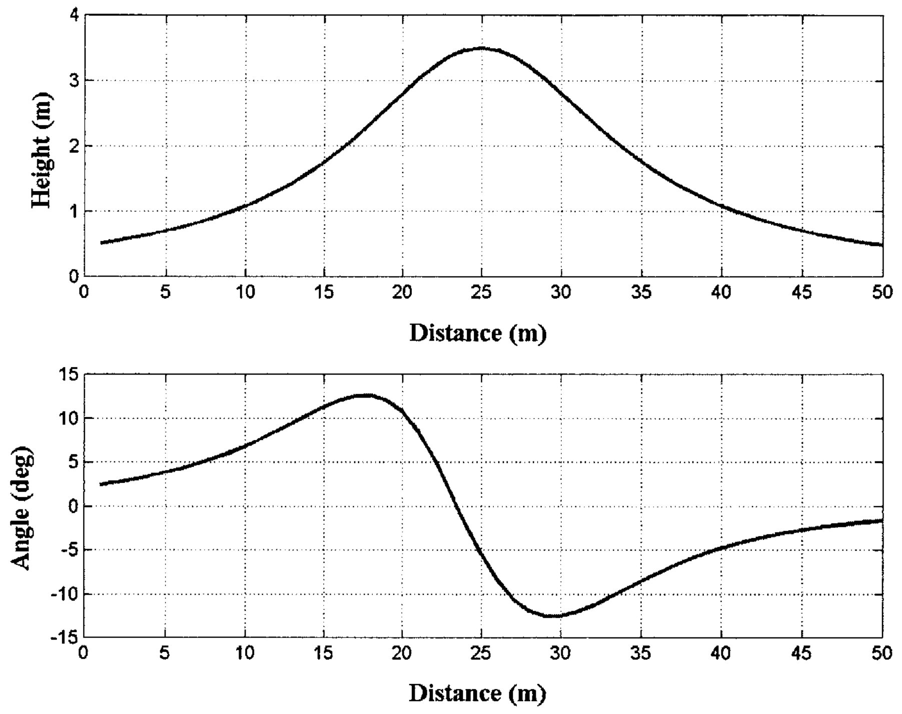

Figure 43: Terrain profile (a) and slope history (b).

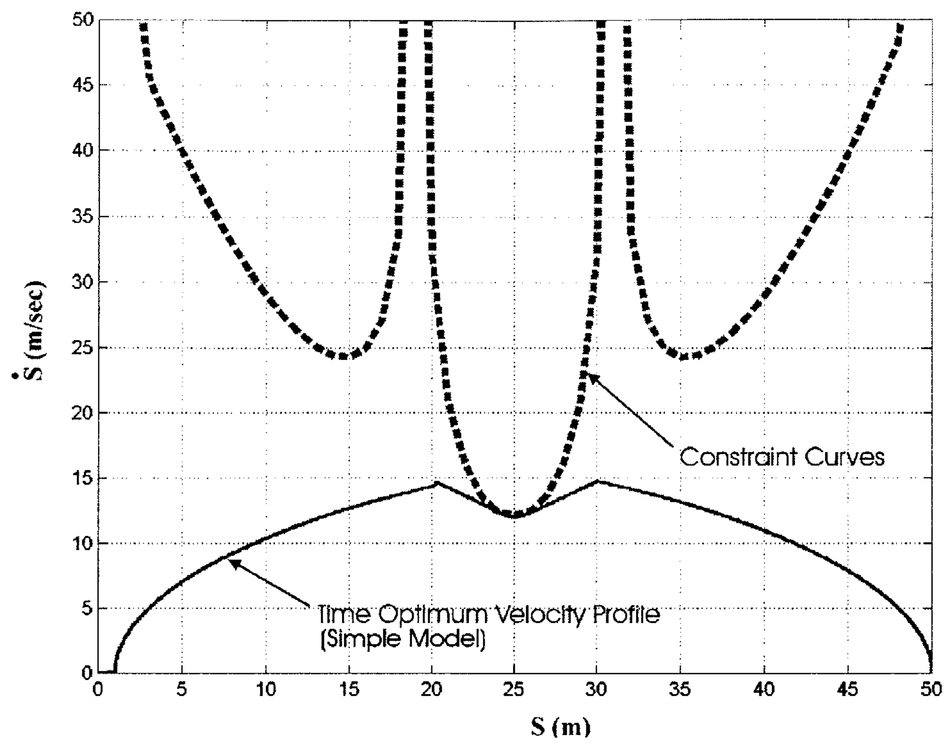

Figure 44: Time optimum velocity profile and physical interaction constraint limits as reproduced from Shiller and Gwo [27]. Courtesy K. Iagnemma.

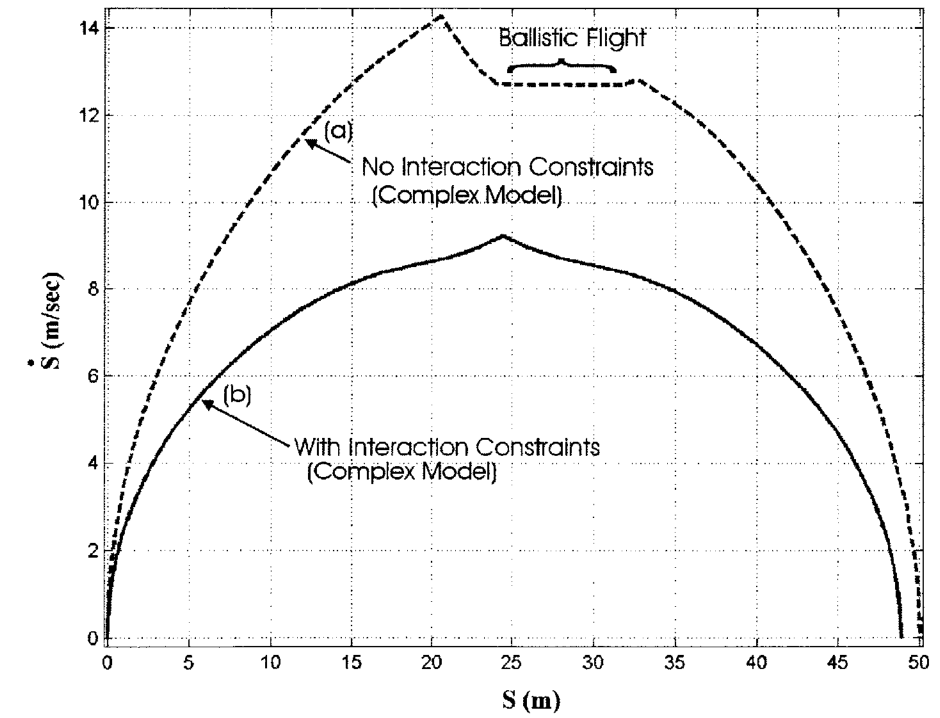

Figure 45: Maximum velocity profiles generated using a multi-DOF vehicle model with and without physical interaction constraints (no slip and ground contact).

CHAPTER

4 CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

4.1 Conclusions

This thesis has reviewed a stability-based kinematic reconfigurability algorithm, previously introduced in [1], and has presented simulation and experimental results from its implementation on the JPL SRR. The results of a study towards generating a method for high-speed rough terrain motion planning free of terrain interaction constraints have also been shown.

Chapter 2 reviewed a method for kinematic reconfigurability [1]. The method is intended for robots with redundant degrees of freedom, i.e. those with a greater number of actuators than the minimum needed for motion and available self motions. The algorithm uses both a previously existing Force-Angle stability measure and a method to estimate wheel-terrain contact angles without using force-torque sensors. This is coupled with a 3D kinematic model to determine the vehicle stability angles. A stability metric that optimizes the vehicle pose based on stability and maintaining ground clearance was proposed. The reconfiguration algorithm was verified both in simulation and through experimental testing. Its effectiveness was also shown for vehicles using just their manipulator to change the location of their center of mass. Results from field trials with the JPL Sample Return Rover at the Arroyo Seco in Altadena, California were presented.

Chapter 3 introduced an off-line planar motion planning method for high-speed vehicles operating in rough terrain. A nine degree of freedom planar vehicle model that accounts for body-wheel interaction was developed. Vehicle failure criteria were discussed. The definition of and a method for generating the maximum and minimum velocity trajectories were also presented. The algorithm shows that aggressive behaviors such as jumping may safely be used to traverse particularly rough terrain. However, the algorithm also proves not to be practically applicable due to high computational requirements and the need for *a priori* knowledge of the full terrain profile.

4.2 Suggestions for Future Work

In order to fully enhance rough terrain mobility, the stability-based kinematic reconfigurability algorithm should be combined with a traction-based algorithm. The two optimization criteria often have conflicting goals. A performance metric that considers both at the same time would allow true vehicle pose optimization. The metric could optimize traction while continuously monitoring vehicle stability angles. Such a method is superior to one that optimizes *either* stability *or* traction with an ad-hoc consideration for the other.

Another proposed research direction is the integration of kinematic configuration planning with high-speed vehicle motion. Such a planner could use knowledge of the terrain profile and a vehicle model to null out unfavorable system rotational rates (pitch or roll). These rates are often the result of asymmetric force application to the vehicle wheels. The development of this planner would allow a vehicle to operate even faster in more varied rough terrain.

High-speed rough terrain motion planning also warrants further research. Thus far, only a planar model of both the vehicle and terrain has been used. To be relevant to experimental system applications, a full 3D model is needed. 3D motion presents many important problems that are not observed in planar motion, such as vehicle pitch and its effect on turning behavior. Analysis of these problems is needed before such an algorithm may be experimentally verified.

A method for integrating limited terrain knowledge should be studied. One of the assumptions of the current algorithm is *a priori* knowledge of the full terrain profile. This is not always feasible since natural environments are often dynamic systems that may not hold the same profile over long periods of time. Depending on system scale, such information may not even be available. Planning methods that include only the terrain profile within sensor range should be explored.

The final, and perhaps most pertinent suggestion, is to determine a way to simplify high-speed motion planning algorithms for on-line implementation. The algorithm presented in this thesis required over 18 minutes to determine the maximum and minimum velocity trajectories for planar motion. One may expect an order of magnitude increase in time required to generate the same trajectories in 3D. Such long calculation times are unacceptable and severely limit this algorithm's practical effectiveness. Clearly, this is an important problem that requires further investigation.

REFERENCES

[1] Iagnemma, K., Rzepniewski, A., Dubowsky, S., Huntsberger, T., Schenker, P., "Mobile Robot Kinematic Reconfigurability for Rough-Terrain," *Proceedings of the SPIE Symposium on Sensor Fusion and Decentralized Control in Robotic Systems III*, Boston, September 2000.

[2] Schenker, P., Sword, L., Ganino, A., Bickler, D., Hickey, G., Brown, D., Baumgartner, E., Matthies, L., Wilcox, B., Balch, T., Aghazarian, H., and Garrett, M., "Lightweight Rovers for Mars Science Exploration and Sample Return," *Proceedings of SPIE XVI Intelligent Robots and Computer Vision Conference*, 1997, 3208, p. 24-36.

[3] Mishkin, A.H., Morrison, J.C., Nguyen, T.T., Stone, H.W., Cooper, B.K., Wilcox, B.H., "Experiences with Operations and Autonomy of the Mars Pathfinder Mission", *IEEE Aerospace Conference*, 1998, v 2, p 337-350.

[4] Hayati, S., Volpe, R., Backes, P., Balaram, J., Welch, W., "Microrover Research for Exploration of Mars," *AIAA Forum on Advanced Developments in Space Robotics*, 1998.

[5] Volpe, R., Baumgartner, E., Schenker, P., Hayati, S., "Technology Development and Testing for Enhanced Mars Rover Sample Return Operations," *IEEE Aerospace Conference Proceedings*, 2000, v 7, p 247-257.

[6] Wilcox, B.H., Jones, R.M., "The MUSES-CN Nanorover Mission and Related Technology," *IEEE Aerospace Conference Proceedings*, 2000, v 7, p 287-295.

[7] Weisbin, C.R., Blitch, J., Lavery, D., Krotkov, E., Shoemaker, C., Matthies, L., Rodriguez, G. "Miniature robots for space and military missions," *IEEE Robotics & Automation Magazine*, September 1999, v 6, issue 3, p 9-18.

[8] Sreenivasan, S., Wilcox, B., "Stability and Traction Control of an Actively Actuated Micro-Rover," *Journal of Robotic Systems*, 1994, v 11, no 6, p 487-502.

[9] Sreenivasan, S.V., Wladron, K.J., Mukherjee, S., "Globally Optimal Force Allocation in Active Mechanisms with Four Frictional Contacts," *Journal of Mechanical Design*, September, 1996, v 118, p 353-359.

[10] Sreenivasan, S.V., Nanua, P., "Force Distribution Characteristics of Actively Reconfigurable wheeled vehicle systems," *The 24th ASME Mechanisms Conference*, Irvine, CA, August 1996.

[11] BenAmar, F., Budanov, V., Bidaud, Ph., Plumet, F., Andrade, G., "A High Mobility Redundantly Actuated Mini-rover for Self Adaptation to Terrain Characteristics," *CLAWAR Proceedings of the 3rd International Conference*, 2000.

[12] Balaram, J., "Kinematic State Estimation for a Mars Rover," Accepted for publication in *Robotica, Special issue on Intelligent Autonomous Vehicles*, 2001.

[13] Schenker, S., Pirjanian, P., Balaram, B., Ali, K., Trebi-Ollennu, A., Huntsberger, T., Aghazarian, H., Kennedy, A., Baumgartner, E., Iagnemma, K., Rzepniewski, A., Dubowsky, S., Leger, P., Apostolopoulous, A., McKee, G., "Reconfigurable Robots for All-Terrain Exploration," *Proceedings of the SPIE Symposium on Sensor Fusion and Decentralized Control in Robotic Systems III*, Boston, September 2000, v 4196.

[14] Shkel, A.M., Lumelsky, V.J., "The Jogger's Problem: Accounting for Body Dynamics in Real-Time Motion Planning," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995, v 2, p 441-447.

[15] Choi, J.S., Kim, K.K., "Near-Time-Optimal Trajectory Planning for Wheeled Mobile Robots with Translational and Rotational Sections," *IEEE Transactions on Robotics and Automation*, 2001, v 17, issue 1, p 85-90.

[16] Hait, A., Simeon, T., Taix, M., "Robust Motion Planning for Rough Terrain Navigation," *International Conference on Intelligent Robots and Systems*, 1999, v 1, p 11-16.

[17] Daily, M., Harris, J., Keirsey, D., Olin, K., Payton, D., Reiser, K., Rosenblatt, J., Tseng, D., Wong, V., "Autonomous Cross-Country Navigation with the ALV," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1988, v 2, p 718-726.

[18] Kelly, A., Stentz, A., "Rough terrain Autonomous Mobility - Part 2: An Active Vision, Predictive Control," *Autonomous Robots*, May 1998, no 5, p 163-198.

[19] Dellaert, F., Pomerleau, D., Thorpe, C., "Model-Based Car Tracking Integrated With a Road-Follower," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1998, v 3, p 1889-1894.

[20] Langer, D., Rosenblatt, J.K., Hebert, M., "A Behavior-Based System for Off-Road Navigation," *IEEE Transactions on Robotics and Automation*, 1994, v 10, p 776-783.

[21] Langer, D., Rosenblatt, J.K., Hebert, M., "An Integrated System for Autonomous Off-Road Navigation," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1994, v 1, p 414-419.

[22] Kelly, A., Stentz, A., "Analysis of requirements for high speed rough terrain autonomous mobility. I. Throughput and response," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1997, v 4, p 3318-3325.

[23] Lacaze, A., Moscovitz, Y., DeClaris, N., Murphy, K., "Path Planning for Autonomous Vehicles Driving Over Rough Terrain," *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1998, p 50-55.

[24] Coombs, D., Murphy, K., Lacaze, A., Legowik, S., "Driving Autonomously Offroad up to 35 km/h," *Proceeding of the Intelligent Vehicles Symposium*, 2000, p 186-191.

[25] Fiorini, P., Shiller, Z., "Motion Planning in Dynamic Environments Using Velocity Obstacles," *International Journal of Robotics Research*, 1998, v 17, no 7, p 760-772.

[26] Fiorini, P., and Shiller, Z., "Time Optimal Trajectory Planning in Dynamic Environments," *Journal of Applied Mathematics and Computer Science, Special Issue on Recent Development in Robotics*, 1997, v 7, no 2, p 101-126.

[27] Shiller, Z., Gwo, Y.-R., "Dynamic Motion Planning of Autonomous Vehicles," *IEEE Transactions on Robotics and Automation*, 1991, v 7, issue 2, p 241-249.

[28] Juidette, H., Youlal, H., "Fuzzy Dynamic Path Planning Using Genetic Algorithms," *Electronics Letters*, 2000, v 36, no 4, p374-376.

[29] Howard, A., Seraji, H., "Real-Time Assessment of Terrain Traversability for Autonomous Rover Navigation," *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, v 1, p 58-63.

[30] Margolis, D., "Semi-Active Control of Wheel Hop in Ground Vehicles," *Vehicle System Dynamics*, 1983, v 12, p 317-330.

[31] Karnopp, D., "Active Damping in Road Vehicle Suspension Systems," *Vehicle System Dynamics*, 1983, v 12, p 291-316.

[32] Schuetze, K.T., Beno, J.H., Weldon, W.F., Sreenivasan, S.V., "A Comparison of Controller Designs for an Active, Electromagnetic, Off-Road Vehicle Suspension System Traveling at High Speed," *SAE*, 1998, 980924, p 16-26.

[33] Gopalasamy, S., Osorio, C., Hendrick, K., Rajamani, R., "Model Predictive Control for Active Suspensions – Controller Design and Experimental Study," *Proceeding of the ASME Dynamic Systems and Control Division*, 1997, v 61, p 725-733.

[34] Iagnemma, K., "Rough-Terrain Mobile Robot Planning and Control, with Application to Planetary Exploration," *Ph.D. Thesis*, Massachusetts Institute of Technology, Cambridge, MA, 2001.

[35] Huntsberger, T., Baumgartner, E., Aghazarian, H., Cheng, Y., Schenker, P., Leger, P., Iagnemma, K., Dubowsky, S., "Sensor Fused Autonomous Guidance of a Mobile Robot and Applications to Mars Sample Return Operations," *Proceedings of the SPIE Symposium on Sensor Fusion and Decentralized Control in Robotic Systems II*, 3839, 1999.

[36] Sreenivasan, S., Waldron, K., "Displacement Analysis of an Actively Articulated Wheeled Vehicle Configuration With Extensions to Motion Planning on Uneven Terrain," *ASME Journal of Mechanical Design*, 1996, v 118, no 2, p 312-317.

[37] Farritor, S., Hacot, H., Dubowsky, S., "Physics-Based Planning for Planetary Exploration," *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, Belgium, May 1998.

[38] Iagnemma, K., Dubowsky, S., "Vehicle Wheel-Ground Contact Angle Estimation: with Application to Mobile Robot Traction Control," *7th International Symposium on Advances in Robot Kinematics, ARK '00*, 2000, p 137-146.

[39] Eckhardt, H., "Kinematic Design of Machines and Mechanisms," McGraw-Hill, New York, 1989.

[40] Craig, J., "Introduction to Robotics: Mechanics and Control," Second ed, Addison-Wesley, Reading, MA 1989.

[41] Papadopoulos, E., Rey, D., "A New Measure of Tipover Stability Margin for Mobile Manipulators," *Proceedings of the IEEE International Conference on Robotics and Automation,* 1996.

[42] Baeder, B.T., Osborn, C.T., Rhea, J.L. "Low Cost Navigation Technology Investigation For The Unmanned Ground Vehicle Pogram," *IEEE Location and Navigation Symposium*, 1994, p 574-580.

[43] http://www.chenowth.com/ viewed 20 July, 2001.

[44] Farritor, S., Hacot, H., and Dubowsky, S., "Physics-Based Planning for Planetary Exploration," *IEEE Internatl. Conference on Robotics and Automation*, 1998, p 278-83.

[45] Hait, A., Simeon, T., "Motion Planning on Rough Terrain for an Articulated Vehicle in Presence of Uncertanties," *IEEE/RSJ International Symposium on Intelligent Robots and Systems*, 1996, p 1126-1133.

[46] http://www.dakar.com/2002/presentationus/ viewed 5 July, 2001.

[47] Shin, K.G., McKay, N.D., "Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints," *IEEE Transactions on Automatic Control*, June 1985, v AC-30, no 6, p 531-541.

[48] Iagnemma, Karl. Personal interview. 6 June 2001.

[49] German, O.V., Ofitserov, D.V., "Problem Solving: Methods, Programming and Future Concepts," Elsevier Science B.V, Amsterdam, 1995.

APPENDIX A

SRR KINEMATICS

In order to use the Force-Angle stability measure, the kinematics of the Sample Return Rover (SRR) have to be well understood. This section reviews the derivation of the kinematics by using Denavit-Hartenberg parameters [40].

The following is the frame attachment procedure as described in [40]

1. Identify the joint axes and imagine infinite lines along them. For steps 2 through 5, consider two of these neighboring lines (at axes i and $i+1$).
2. Identify a common perpendicular or a point of intersection of the two axes. At the point of intersection or at the point where the common perpendicular meets the ith axis, assign the link frame origin.
3. Assign the $\hat{Z}_i$ axis pointing along the ith joint axis.
4. Assign the $\hat{X}_i$ axis pointing along the common perpendicular, or if the two axes intersect, assign $\hat{X}_i$ to be normal to the plane containing the two axes
5. Assign the $\hat{Y}_i$ axis to complete a right-hand coordinate system.

Figure 46 shows the SRR rover with the left side reference frames assigned. The $\hat{Y}_i$ axes have been omitted for clarity. The right side reference frames are assigned in a similar fashion.

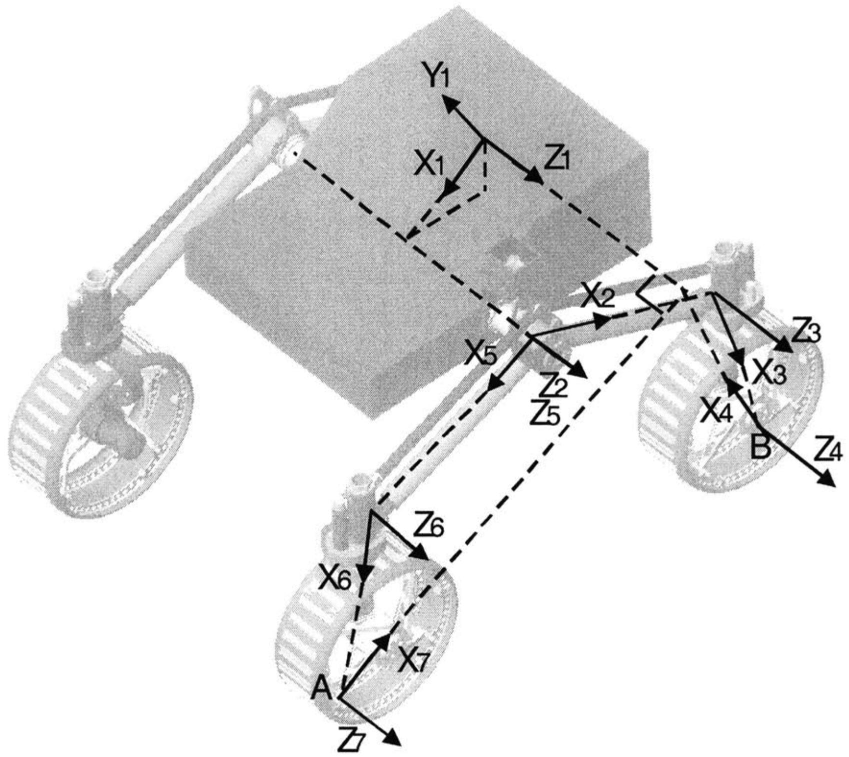

Figure 46: SRR Rover with assigned left side reference frames.

Once the frames have been assigned, it is easy to determine the following link parameters (as described in [40]):

a_i = the distance from $\hat{Z}_i$ to $\hat{Z}_{i+1}$ measured along $\hat{X}_i$

α_i = the angle between $\hat{Z}_i$ and $\hat{Z}_{i+1}$ measured about $\hat{X}_i$

d_i = the distance from $\hat{X}_{i-1}$ to $\hat{X}_i$ measured along $\hat{Z}_i$

θ_i = the angle between $\hat{X}_{i-1}$ and $\hat{X}_i$ measured about $\hat{Z}_i$

Inspection of the frame assignments yields two kinematic loops per side of the SRR: 1-2-3-4 and 1-5-6-7, on the left side. The two resulting sets of parameters are shown in Table 2, where β_1 is the angle between $\hat{X}_1$ and $\hat{X}_5$ measured along $\hat{Z}_5$.

i	α_i	a_i	θ_i	d_i
1	0	a_1	θ_1	$-d_1$
2	0	a_2	θ_2	d_2
3	0	a_3	θ_3	0
4	0	a_4	θ_4	0

i	α_i	a_i	θ_i	d_i
1	0	a_1	β_1	$-d_1$
5	0	a_5	θ_5	d_5
6	0	a_6	θ_6	0
7	0	a_7	θ_7	0

Table 2: Left side Denavit-Hartenberg parameters [40].

In order to use the Force-Angle stability measure, it is necessary to know the location of the wheel-ground contact point. This may be determined without writing full loop closure equations. A method for estimating the contact angles, and points, is presented in [38]. These points were chosen to be the origin of frames {4}, point B, and {7}, point A, on the left hand side, with parallel construction on the right hand side. To simplify the discussion, the rest of this development will only consider the left side of the SRR. The locations of the contact points are:

$$^{6}\vec{P}_A = \begin{pmatrix} a_6 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

and

$$ {}^{3}\vec{P}_B = \begin{pmatrix} a_3 \\ 0 \\ 0 \\ 1 \end{pmatrix} $$

where the preceding superscript denotes the describing frame. Homogenous transformation matrices are used to describe these points in the body fixed frame {1}. The general form of these matrices is:

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_i)d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Thus,

$$ {}^{1}\vec{P}_A = {}^{1}_{5}T\,{}^{5}_{6}T\,{}^{6}\vec{P}_A $$

and

$$ {}^{1}\vec{P}_B = {}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}\vec{P}_B $$

Table 2 lists all link parameters needed to determine ${}^{1}\vec{P}_A$ and ${}^{1}\vec{P}_B$; of the parameters, only a_2 and a_5 are constants (they describe the length of the SRR rocker suspension). Additional knowledge is required to determine the remaining parameters. The SRR is equipped with a passive differential joint, ϕ, between its left and right side rocker, as well as active joints that determine the spread of the rockers, shown as ψ in Figure 47. The variable parameter angles may be expressed in terms of ϕ, ψ, and constants. Thus:

$$\theta_2 = \gamma + \phi + \psi_{left}$$

$$\theta_3 = \zeta - \psi_{left} + \lambda_B$$

$$\theta_5 = \gamma + \phi - \psi_{left}$$

$$\theta_6 = \psi_{left} - \zeta + \lambda_A$$

where γ and ζ are geometry-determined constants and λ_A and λ_B are wheel-ground contact angles. A method for estimating the contact angles is presented in [38].

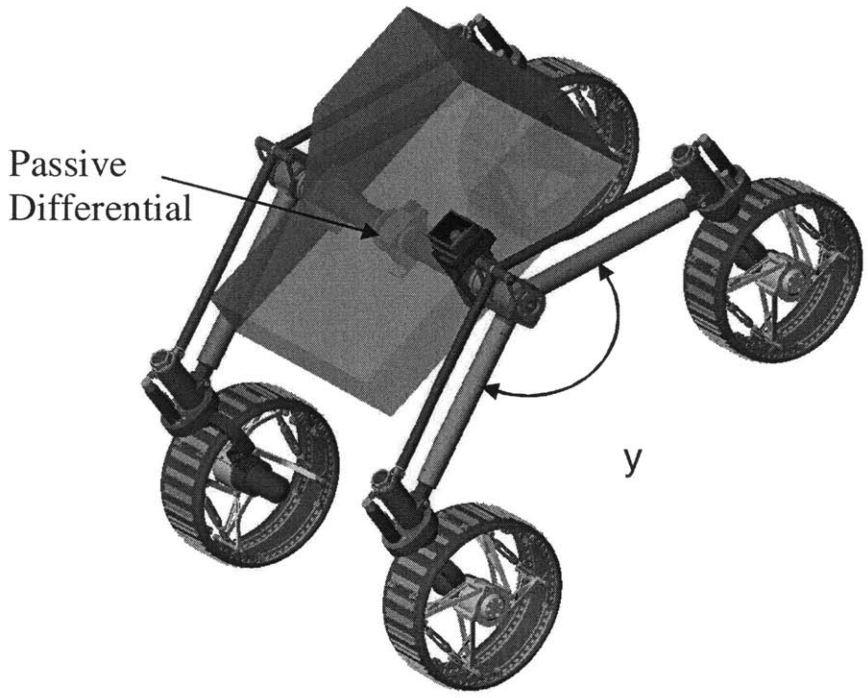

Figure 47: SRR with actively actuated rocker joint.

The location of frame {1}, which coincides with the vehicle center of mass, is determined by knowing the state of the vehicle (ϕ, ψ_{left}, ψ_{right}, and the manipulator configuration). Thus, all link parameters are known and one can determine the vector locations of the contact points relative to the center of mass.

APPENDIX

B

KINEMATIC RECONFIGURABILITY SAMPLE CODE

This Program is designed to control the active reconfiguration of SRR shoulders.

Variable parameter names correspond to those introduced in Appendix A.

INPUTS:

Roll
Pitch
Differential Angle reading
Current Shoulder angles

OUTPUTS:

Shoulder angles

```
#include  <stdio.h>
#include  <stdlib.h>
#include <math.h>
//#include "reconfig.h"
/************ Global Constants ****************/
#define PI 3.1416
        // the following nominal values were gotten from a MATLAB program
        //                  and correspond to 135 degrees between arm links
#define theta_1_nom 0.7260 // [radians] = 41.5943 degrees
#define theta_2_nom 0.7141 // [radians] = 40.9162 degrees
#define theta_3_nom 0.7260 // [radians] = 41.5943 degrees
#define theta_4_nom 0.7984 // [radians] = 45.7463 degrees
#define psi_nom 1.1781 // [radians] = 67.5 degrees; nominal shoulder angle (1/2 of 135)
#define step 0.01 // step size for derivative

/************ Global Variables *************/
float unity[3][3] = {{1.0 , 0.0 , 0.0} ,
                     {0.0 , 1.0 , 0.0} ,
                        {0.0 , 0.0 , 1.0}};
// These are the weights for the stability measure
float A1 = 1.0;
float A2 = 1.0;
float A3 = 1.0;
float A4 = 1.0;
float A5 = 10.0;

/*-------------------------------------------------------*/
        This is the start of the Functions
/*-------------------------------------------------------*/
void mul3_1(float a[3], float b[3], float c[3][3])
{
        c[0][0] = a[0] * b[0];
        c[1][0] = a[1] * b[0];
        c[2][0] = a[2] * b[0];
        c[0][1] = a[0] * b[1];
        c[1][1] = a[1] * b[1];
        c[2][1] = a[2] * b[1];
        c[0][2] = a[0] * b[2];
        c[1][2] = a[1] * b[2];
        c[2][2] = a[2] * b[2];
}

//////////////////////////////////////////////////////////////////////

void sub3_3(float a[3][3], float b[3][3], float c[3][3])
{
        c[0][0]=a[0][0] - b[0][0];
        c[1][0]=a[1][0] - b[1][0];
        c[2][0]=a[2][0] - b[2][0];
        c[0][1]=a[0][1] - b[0][1];
        c[1][1]=a[1][1] - b[1][1];
        c[2][1]=a[2][1] - b[2][1];
        c[0][2]=a[0][2] - b[0][2];
        c[1][2]=a[1][2] - b[1][2];
        c[2][2]=a[2][2] - b[2][2];
}

//////////////////////////////////////////////////////////////////////

void mul3_3_1(float a[3][3], float b[3], float c[3])
{
        c[0]=a[0][0] * b[0] + a[0][1] * b[1] + a[0][2] * b[2];
        c[1]=a[1][0] * b[0] + a[1][1] * b[1] + a[1][2] * b[2];
        c[2]=a[2][0] * b[0] + a[2][1] * b[1] + a[2][2] * b[2];
```

```
}

//////////////////////////////////////////////////////////////////

float magnitude(float a[3])
{
        return( sqrt( a[0] * a[0] + a[1] * a[1] + a[2] * a[2] ) );
}

//////////////////////////////////////////////////////////////////

void kinema_calc( float psi_l, float psi_r, float phi_r, float eta[3], float a1[3], float a2[3],
float a3[3], float a4[3], float l1[3], float l2[3], float l3[3], float l4[3])
{
        int i;

        float theta_10 = 0.1350;  // additional rotation angle
        float a_2 = 0.2544;       // meters, Suspension arm length
        float a_3 = 0.1555;       // meters,
        float d_2 = 0.2119;       // meters, half-width of the rover
        float r = 0.1008;         // meters, wheel radius

        float res3_3[3][3]; // intermediate dummy variable
        float a_3_3[3][3];  // intermediate dummy variable
        float mag_a;
        float mag_l;

        // These are the wheel-ground contact angles
        float gamma_w1 = (-PI/2);    float gamma_w2 = (-PI/2);    float gamma_w3 = (-
PI/2);  float gamma_w4 = (-PI/2);

        float gPa[3];           float gPb[3];           float gPc[3];           float gPd[3];


        // Pointer vectors from C of M to each one of the wheels
        gPa[0] =  r*cos(-phi_r+gamma_w1)-sin(phi_r+theta_10)*a_3-
1.074*sin(phi_r+psi_l)*a_2+0.003313+0.01398*cos(eta[1]-0.212-
eta[0])+0.01398*cos(eta[1]-0.212+eta[0])+0.007320*sin(eta[0]+eta[2]+eta[1]-
0.212)+0.007320*sin(-eta[0]+eta[2]+eta[1]-0.212)-0.07445*sin(phi_r-
psi_l)*a_2+0.07445*sin(phi_r+psi_r)*a_2+0.07445*sin(phi_r-psi_r)*a_2;
   gPa[1] =  r*sin(-phi_r+gamma_w1)-cos(phi_r+theta_10)*a_3-
0.9256*cos(phi_r+psi_l)*a_2-0.02642-0.02795*sin(eta[1]-
0.212)+0.01464*cos(eta[2]+eta[1]-0.212)+0.07445*cos(phi_r-
psi_l)*a_2+0.07445*cos(phi_r+psi_r)*a_2+0.07445*cos(phi_r-
psi_r)*a_2+0.04589*cos(phi_r);
```

```
   gPa[2] = d_2+0.01398*sin(eta[1]-0.212+eta[0])-0.01398*sin(eta[1]-0.212-
eta[0])+0.007320*cos(-eta[0]+eta[2]+eta[1]-0.212)-0.007320*cos(eta[0]+eta[2]+eta[1]-
0.212);


        gPb[0] = r*cos(-phi_r+gamma_w2)+sin(-phi_r+theta_10)*a_3-1.074*sin(phi_r-
psi_l)*a_2+0.003313+0.01398*cos(eta[1]-0.212-eta[0])+0.01398*cos(eta[1]-
0.212+eta[0])+0.007320*sin(eta[0]+eta[2]+eta[1]-0.212)+0.007320*sin(-
eta[0]+eta[2]+eta[1]-0.212)-
0.07445*sin(phi_r+psi_l)*a_2+0.07445*sin(phi_r+psi_r)*a_2+0.07445*sin(phi_r-
psi_r)*a_2;
   gPb[1] = r*sin(-phi_r+gamma_w2)-cos(-phi_r+theta_10)*a_3-0.9256*cos(phi_r-
psi_l)*a_2-0.02642-0.02795*sin(eta[1]-0.212)+0.01464*cos(eta[2]+eta[1]-
0.212)+0.07445*cos(phi_r+psi_l)*a_2+0.07445*cos(phi_r+psi_r)*a_2+0.07445*cos(phi
_r-psi_r)*a_2+0.04589*cos(phi_r);
   gPb[2] = d_2+0.01398*sin(eta[1]-0.212+eta[0])-0.01398*sin(eta[1]-0.212-
eta[0])+0.007320*cos(-eta[0]+eta[2]+eta[1]-0.212)-0.007320*cos(eta[0]+eta[2]+eta[1]-
0.212);


        gPc[0] =
r*cos(phi_r+gamma_w3)+sin(phi_r+theta_10)*a_3+1.074*sin(phi_r+psi_r)*a_2+0.0033
13+0.01398*cos(eta[1]-0.212-eta[0])+0.01398*cos(eta[1]-
0.212+eta[0])+0.007320*sin(eta[0]+eta[2]+eta[1]-0.212)+0.007320*sin(-
eta[0]+eta[2]+eta[1]-0.212)-0.07445*sin(phi_r-psi_l)*a_2-
0.07445*sin(phi_r+psi_l)*a_2+0.07445*sin(phi_r-psi_r)*a_2;
   gPc[1] = r*sin(phi_r+gamma_w3)-cos(phi_r+theta_10)*a_3-
0.9256*cos(phi_r+psi_r)*a_2-0.02642-0.02795*sin(eta[1]-
0.212)+0.01464*cos(eta[2]+eta[1]-0.212)+0.07445*cos(phi_r-
psi_l)*a_2+0.07445*cos(phi_r+psi_l)*a_2+0.07445*cos(phi_r-
psi_r)*a_2+0.04589*cos(phi_r);
   gPc[2] = -d_2+0.01398*sin(eta[1]-0.212+eta[0])-0.01398*sin(eta[1]-0.212-
eta[0])+0.007320*cos(-eta[0]+eta[2]+eta[1]-0.212)-0.007320*cos(eta[0]+eta[2]+eta[1]-
0.212);


        gPd[0] = r*cos(phi_r+gamma_w4)-sin(-phi_r+theta_10)*a_3+1.074*sin(phi_r-
psi_r)*a_2+0.003313+0.01398*cos(eta[1]-0.212-eta[0])+0.01398*cos(eta[1]-
0.212+eta[0])+0.007320*sin(eta[0]+eta[2]+eta[1]-0.212)+0.007320*sin(-
eta[0]+eta[2]+eta[1]-0.212)-0.07445*sin(phi_r-psi_l)*a_2-
0.07445*sin(phi_r+psi_l)*a_2+0.07445*sin(phi_r+psi_r)*a_2;
   gPd[1] = r*sin(phi_r+gamma_w4)-cos(-phi_r+theta_10)*a_3-0.9256*cos(phi_r-
psi_r)*a_2-0.02642-0.02795*sin(eta[1]-0.212)+0.01464*cos(eta[2]+eta[1]-
0.212)+0.07445*cos(phi_r-
psi_l)*a_2+0.07445*cos(phi_r+psi_l)*a_2+0.07445*cos(phi_r+psi_r)*a_2+0.04589*cos
(phi_r);
```

```
   gPd[2] =  -d_2+0.01398*sin(eta[1]-0.212+eta[0])-0.01398*sin(eta[1]-0.212-
eta[0])+0.007320*cos(-eta[0]+eta[2]+eta[1]-0.212)-0.007320*cos(eta[0]+eta[2]+eta[1]-
0.212);

        // Now I continue to determine the tipover axes

        for(i=0;i<3;i++) a1[i] = gPb[i]-gPa[i];
        for(i=0;i<3;i++) a2[i] = gPc[i]-gPb[i];
        for(i=0;i<3;i++) a3[i] = gPd[i]-gPc[i];
        for(i=0;i<3;i++) a4[i] = gPa[i]-gPd[i];

        // Here I make them into unit vectors
        mag_a=magnitude(a1);
        for(i=0;i<3;i++) a1[i]=a1[i]/mag_a;
        mag_a=magnitude(a2);
        for(i=0;i<3;i++) a2[i]=a2[i]/mag_a;
        mag_a=magnitude(a3);
        for(i=0;i<3;i++) a3[i]=a3[i]/mag_a;
        mag_a=magnitude(a4);
        for(i=0;i<3;i++) a4[i]=a4[i]/mag_a;

        // Now I get the perpendicular to each axis vector
        mul3_1(a1, a1, a_3_3);
        sub3_3(unity, a_3_3, res3_3);
        mul3_3_1(res3_3 , gPb, l1);
        mag_l=magnitude(l1);
        for(i=0;i<3;i++) l1[i]=l1[i]/mag_l; // This makes it into a unit vector

        mul3_1(a2, a2, a_3_3);
        sub3_3(unity, a_3_3, res3_3);
        mul3_3_1(res3_3 , gPc, l2);
        mag_l=magnitude(l2);
        for(i=0;i<3;i++) l2[i]=l2[i]/mag_l; // This makes it into a unit vector

        mul3_1(a3, a3, a_3_3);
        sub3_3(unity, a_3_3, res3_3);
        mul3_3_1(res3_3 , gPd, l3);
        mag_l=magnitude(l3);
        for(i=0;i<3;i++) l3[i]=l3[i]/mag_l; // This makes it into a unit vector

        mul3_1(a4, a4, a_3_3);
        sub3_3(unity, a_3_3, res3_3);
        mul3_3_1(res3_3 , gPa, l4);
        mag_l=magnitude(l4);
        for(i=0;i<3;i++) l4[i]=l4[i]/mag_l; // This makes it into a unit vector
}
```

```
///////////////////////////////////////////////////////////////

void force_calc( float alpha, float gamma, float a1[3], float a2[3], float a3[3], float a4[3],
float f1[3], float f2[3], float f3[3], float f4[3], float *zeta, float *beta)
{

        int i;

        float Fg[3];
        float F[3];
        float weight = 115.27;     // Newtons
        float a_3_3[3][3];
        float res3_3[3][3];
        float mag_f;


        // here I compute the force vectors in the g frame

        //first, gravity
        *zeta=atan(-tan(gamma)/tan(alpha));
        *beta=atan(-sin(* zeta)*tan(alpha));


        // This if stucture is here to calculate beta near zeta shift points
        // at 0, 180 and +-90 degrees
        // I have so many if statements here since I had trouble with the
        // absolute value statement
        if ( ( (PI/2.0-0.1) <= *zeta ) && ( *zeta <= (PI/2.0+0.1) ) ){
                *beta=atan(-sin(*zeta)*tan(alpha));
        }
        if ( ( (-PI/2.0-0.1) <= *zeta ) && ( *zeta <= (-PI/2.0+0.1) ) ){
                *beta=atan(-sin( *zeta)*tan(alpha));
        }
        if ( ( -0.1<= *zeta ) && ( *zeta <= 0.1 ) ){
                *beta=atan(cos( *zeta)*tan(gamma));
        }
        if ( ( (PI-0.1) <= *zeta ) && ( *zeta <=(PI+0.1) ) ){
                *beta=atan(cos( *zeta)*tan(gamma));
        }
        if ( ( (-PI-0.1) <= *zeta ) && ( *zeta <=(-PI+0.1) ) ){
                *beta=atan(cos( *zeta)*tan(gamma));
        }

        if ( *beta < 0.0 ) *beta= *beta+PI;  // correct for atan
        Fg[0]=weight* cos( *beta)*sin( *zeta);
        Fg[1]=weight*(-sin( *beta));
        Fg[2]=weight*cos( *beta)*cos( *zeta);

        // The redundant structure exists for easy inclusion of additional forces
        F[0]=Fg[0];
        F[1]=Fg[1];
        F[2]=Fg[2];


        // Now I get the forces about each axis vector
        mul3_1(a1, a1, a_3_3);
        sub3_3(unity, a_3_3, res3_3);
        mul3_3_1(res3_3 , F , f1);
        mag_f=magnitude(f1);
        for(i=0;i<3;i++) f1[i]=f1[i]/mag_f; // This makes it into a unit vector

        mul3_1(a2, a2, a_3_3);
        sub3_3(unity, a_3_3, res3_3);
        mul3_3_1(res3_3 , F , f2);
        mag_f=magnitude(f2);
        for(i=0;i<3;i++) f2[i]=f2[i]/mag_f; // This makes it into a unit vector

        mul3_1(a3, a3, a_3_3);
        sub3_3(unity, a_3_3, res3_3);
        mul3_3_1(res3_3 , F , f3);
        mag_f=magnitude(f3);
        for(i=0;i<3;i++) f3[i]=f3[i]/mag_f; // This makes it into a unit vector

        mul3_1(a4, a4, a_3_3);
        sub3_3(unity, a_3_3, res3_3);
        mul3_3_1(res3_3 , F , f4);
        mag_f=magnitude(f4);
        for(i=0;i<3;i++) f4[i]=f4[i]/mag_f; // This makes it into a unit vector
}

/////////////////////////////////////////////////////////////////////

void stab_calc( float psi_l, float psi_r, float l1[3], float l2[3], float l3[3], float l4[3], float
f1[3], float f2[3], float f3[3], float f4[3], float  *theta_1, float *theta_2, float *theta_3,
float *theta_4, float *S )
{

        *theta_1=( acos( f1[0]*l1[0] + f1[1]*l1[1] + f1[2]*l1[2] ) )/theta_1_nom;
        *theta_2=( acos( f2[0]*l2[0] + f2[1]*l2[1] + f2[2]*l2[2] ) )/theta_2_nom;
        *theta_3=( acos( f3[0]*l3[0] + f3[1]*l3[1] + f3[2]*l3[2] ) )/theta_3_nom;
        *theta_4=( acos( f4[0]*l4[0] + f4[1]*l4[1] + f4[2]*l4[2] ) )/theta_4_nom;

        // Now I get the lumped stability measure

        *S = (A1/ (*theta_1) + A2/ (*theta_2) + A3/ (*theta_3) + A4/ (*theta_4) +
A5*(psi_l-psi_nom)*(psi_l-psi_nom) + A5*(psi_r-psi_nom)*(psi_r-psi_nom));
}

////////////////////////////////////////////////////////////////////

void deriv_calc( float psi, float theta_1_o, float theta_2_o, float theta_3_o, float
theta_4_o, float theta_1, float theta_2, float theta_3, float theta_4, float *dSdpsi )
{
        float dtheta1_dpsi; float dtheta2_dpsi; float dtheta3_dpsi; float dtheta4_dpsi;

        dtheta1_dpsi=(theta_1-theta_1_o)/step;
        dtheta2_dpsi=(theta_2-theta_2_o)/step;
        dtheta3_dpsi=(theta_3-theta_3_o)/step;
        dtheta4_dpsi=(theta_4-theta_4_o)/step;

        *dSdpsi = -A1*dtheta1_dpsi/theta_1_o/theta_1_o-
A2*dtheta2_dpsi/theta_2_o/theta_2_o-A3*dtheta3_dpsi/theta_3_o/theta_3_o-
A4*dtheta4_dpsi/theta_4_o/theta_4_o+2*A5*(psi-psi_nom);
}

/*---------------------------------------------------------*/
        func is the main program
        for stability calculation
/*---------------------------------------------------------*/
float func( float psi[2])
{
        float a1[3];        float a2[3];        float a3[3];        float a4[3];
        float l1[3];        float l2[3];        float l3[3];        float l4[3];
        float f1[3];        float f2[3];        float f3[3];        float f4[3];
        float theta_1;      float theta_2;      float theta_3;      float theta_4;
        float psi_l;        float psi_r;


        float S; /* Lumped stability measure */
        float beta;
        float zeta;
        float alpha; float gamma; float phi_r; float eta[3];

        gamma = (0.0 + 90.0)*(PI / 180.0); //( RoverVector.Roll * PI / 180.0) + (PI / 2.0);
        /* Roll */
        alpha = (0.0 - 90.0)*(PI / 180.0); //-1.0*(( RoverVector.Pitch * PI / 180.0) - (PI /
2.0)); /* Pitch */
        phi_r = 0.0; //-Pots_Data[10] * PI / 180.0; /* Differential Angle (right) */

        eta[0] = 0.0; /* base */
        eta[1] = PI / 2.0; /* shoulder */
        eta[2] = PI / 2.0; /* elbow */

        psi_l = psi[0];
        psi_r = psi[1];

        /*--------------------------------------------*/
        kinema_calc( psi_l, psi_r, phi_r, eta, a1, a2, a3, a4, l1, l2, l3, l4);
        force_calc( alpha, gamma, a1 , a2, a3, a4, f1, f2, f3, f4, &zeta, &beta);
        stab_calc( psi_l, psi_r, l1, l2, l3, l4, f1, f2, f3, f4, &theta_1, &theta_2, &theta_3,
&theta_4, &S );

        return S;

}

/*-------------------------------------------------------*/
                dfunc is the main program
                 for derivative calculation
/*-------------------------------------------------------*/
void dfunc( float psi[2], float ders[2])
{
        float a1[3];            float a2[3];            float a3[3];            float a4[3];
        float l1[3];            float l2[3];            float l3[3];            float l4[3];
        float f1[3];            float f2[3];            float f3[3];            float f4[3];
        float theta_1;          float theta_2;          float theta_3;          float theta_4;
        float theta_1_o;        float theta_2_o;        float theta_3_o;   float theta_4_o;
        float psi_l;            float psi_r;
        float dSdpsi_l;         float dSdpsi_r;

        float S; /* Lumped stability measure */
        float beta;
        float zeta;
        float alpha; float gamma; float phi_r; float eta[3];


        gamma = (0.0 + 90.0)*(PI / 180.0); //( RoverVector.Roll * PI / 180.0) + (PI / 2.0);
        /* Roll */
        alpha = (0.0 - 90.0)*(PI / 180.0); //-1.0*(( RoverVector.Pitch * PI / 180.0) - (PI /
2.0)); /* Pitch */
        phi_r = 0.0; //-Pots_Data[10] * PI / 180.0; /* Differential Angle (right) */
        eta[0] = 0.0; /* base */
        eta[1] = PI / 2.0; /* shoulder */
        eta[2] = PI / 2.0; /* elbow */

        psi_l = psi[0];
        psi_r = psi[1];
        /*-------------------------------------------*/
        kinema_calc( psi_l, psi_r, phi_r, eta, a1, a2, a3, a4, l1, l2, l3, l4);
        force_calc( alpha, gamma, a1 , a2, a3, a4, f1, f2, f3, f4, &zeta, &beta);
        stab_calc( psi_l, psi_r, l1, l2, l3, l4, f1, f2, f3, f4, &theta_1_o, &theta_2_o,
&theta_3_o, &theta_4_o, &S );

        /* Now I recalculate for derivative wrt psi_l */
        kinema_calc( (psi_l + step), psi_r, phi_r, eta, a1, a2, a3, a4, l1, l2, l3, l4);
        force_calc( alpha, gamma, a1 , a2, a3, a4, f1, f2, f3, f4, &zeta, &beta);
        stab_calc( psi_l, psi_r, l1, l2, l3, l4, f1, f2, f3, f4, &theta_1, &theta_2, &theta_3,
&theta_4, &S );
        deriv_calc( psi_l, theta_1_o, theta_2_o, theta_3_o, theta_4_o, theta_1, theta_2,
theta_3, theta_4, &dSdpsi_l );


        /* Now I recalculate for derivative wrt psi_r */
        kinema_calc( psi_l, (psi_r + step), phi_r, eta, a1, a2, a3, a4, l1, l2, l3, l4);
        force_calc( alpha, gamma, a1 , a2, a3, a4, f1, f2, f3, f4, &zeta, &beta);
        stab_calc( psi_l, psi_r, l1, l2, l3, l4, f1, f2, f3, f4, &theta_1, &theta_2, &theta_3,
&theta_4, &S );
        deriv_calc( psi_r, theta_1_o, theta_2_o, theta_3_o, theta_4_o, theta_1, theta_2,
theta_3, theta_4, &dSdpsi_r );

        ders[0] = 0.1 * dSdpsi_l;
        ders[1] = 0.1 * dSdpsi_r;
        return;
}
```

APPENDIX

C

VEHICLE MODELING

The key to high-speed rough terrain traverse is modeling the system with a minimum level of complexity as to capture all important effects. Figure 48 shows a free body diagram for a general, 4-wheel robot. The body and each wheel possess both mass and inertia. The mass and inertia of suspension components may be divided among the body and the wheels. Non-inertial coordinate frames are attached at each body's center of mass.

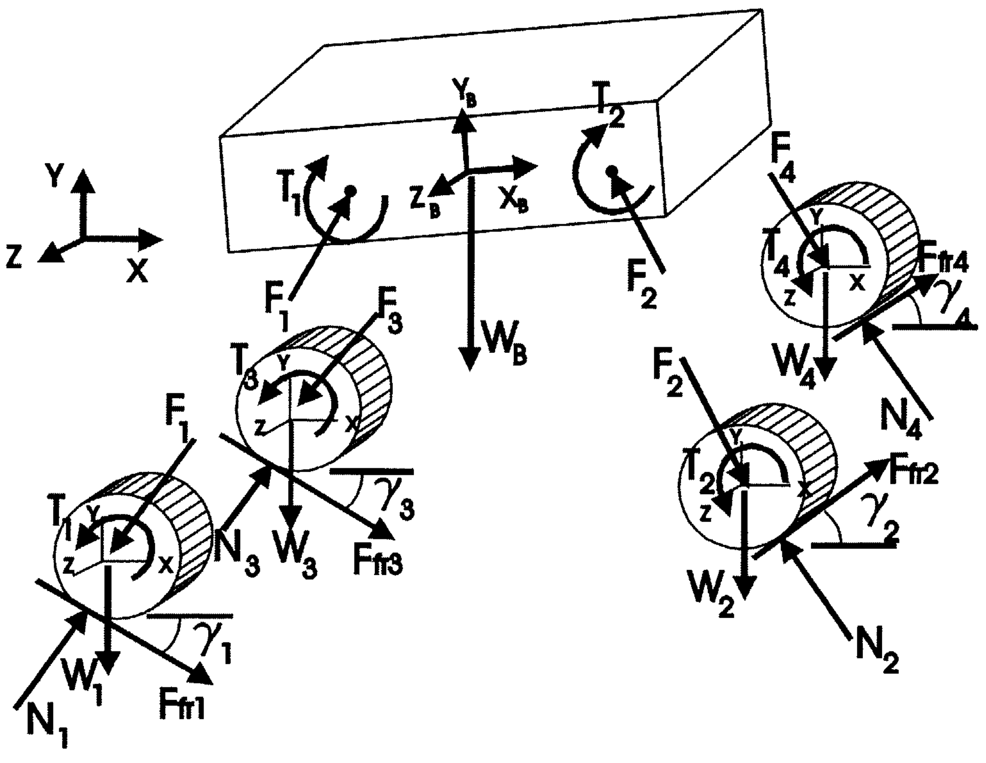

Figure 48: General mobile robot free body diagram.

The problem of generating the equations of motion simplifies greatly for a planar robot. Figure 49 shows a proposed model of a planar vehicle. Suspension and wheel compliance are taken into account. Note that in two dimensions, this becomes a nine degree of freedom system.

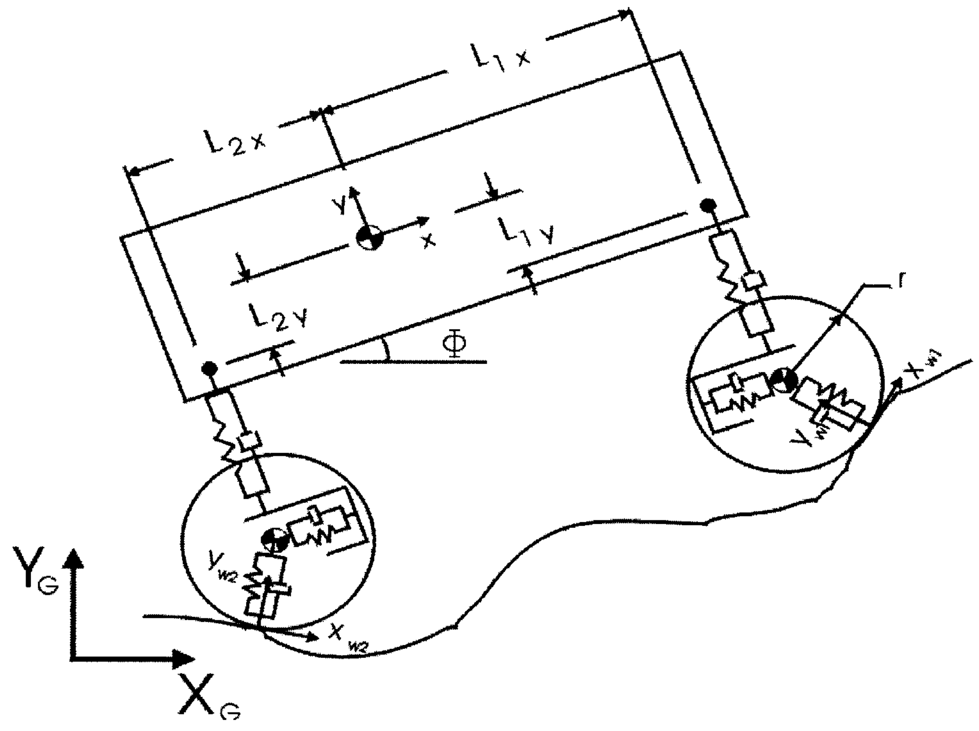

Figure 49: Planar rover model.

Thus, for a planar system, the problem simplifies to 2 inputs:

τ_1 and τ_2, the front and rear wheel torques, respectively,

and 15 outputs:

body: x_B, y_B, ϕ (pitch)

wheel 1: x_1, y_1, θ_1 (rotation angle)

wheel 2: x_2, y_2, θ_2 (rotation angle) ground 1: N_1, F_{fr1}, γ_1 (contact angle)

ground 2: N_2, F_{fr2}, γ_2 (contact angle)

These may be determined by deriving the associated equations of motion and constraint equations:

<u>**Body**</u>

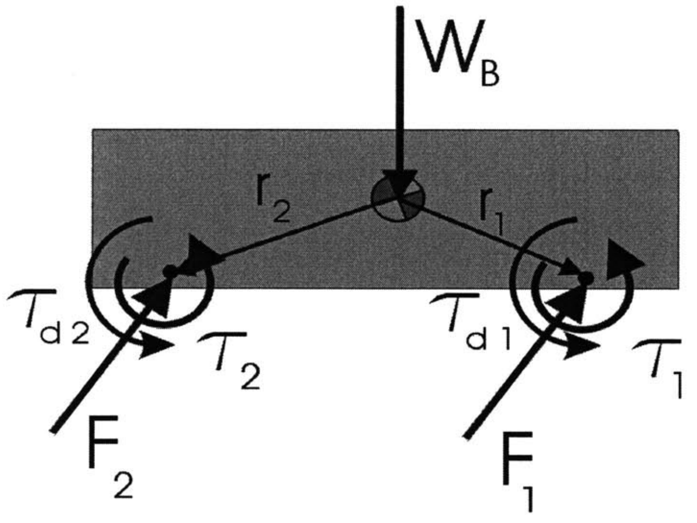

Figure 50: Planar vehicle body free-body-diagram.

Figure 50 shows the free body diagram of the vehicle body. One can readily write the general equations of motion:

$$\sum F = \vec{F}_1 + \vec{F}_2 + \vec{W}_B = m_B \ddot{\vec{X}}_B$$

$$\sum M_{cm} = \vec{\tau}_1 + \vec{\tau}_{d1} + \vec{\tau}_2 + \vec{\tau}_{d2} + \vec{r}_1 \times \vec{F}_1 + \vec{r}_2 \times \vec{F}_2 = I\ddot{\vec{\phi}}$$

where $\vec{W}_b$ is the body weight, I is the body inertia, and $\vec{r}_1$ and $\vec{r}_2$ describe the location of the suspension mounting points relative to the vehicle center of mass.

The damping torques, $\vec{\tau}_{d1}$ and $\vec{\tau}_{d2}$ are:

$$\left|\tau_{di}\right| = C_{bearing}\left|\dot{\theta}_i\right|$$

where $C_{bearing}$ is the wheel bearing friction coefficient.

The vector location of the center of mass is:

$${}^{g}\vec{X}_B = \begin{bmatrix} x_B \\ y_B \end{bmatrix}$$

where the preceding superscript describes the frame in which the points are expressed; the frames are defined in Figure 49.

The suspension forces $\vec{F}_1$ and $\vec{F}_2$ are defined as:

$$\vec{F}_i = K_i \Delta\vec{X}_i + C_i \Delta\dot{\vec{X}}_i$$

where the stiffness matrix is:

$$K_i = \begin{bmatrix} k_1 & k_2 \\ k_3 & k_4 \end{bmatrix}$$

the damping matrix is:

$$C_i = \begin{bmatrix} c_1 & c_2 \\ c_3 & c_4 \end{bmatrix}$$

and

$$\Delta\vec{X}_i = \begin{bmatrix} x_B - x_i \\ y_B - x_i \end{bmatrix}$$

$$\Delta\dot{\vec{X}}_i = \begin{bmatrix} \dot{x}_B - \dot{x}_i \\ \dot{y}_B - \dot{y}_i \end{bmatrix}$$

Note that all quantities must be defined in the same reference frame. Since it is desirable to know the acceleration of the vehicle body in the inertial frame, both the stiffness and damping matrices, which are defined in the body frame {B}, must be transformed accordingly. First, a rotation matrix is defined as:

$$^{g}_{i}R = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$$

where ϕ is the body pitch. The superscript describes the destination frame, the subscript describes the starting frame. The rotation matrix is then used to express the stiffness and damping matrices in the global frame {G}:

$$^{G}K_i = {}^{G}_{B}R\,{}^{B}K_i$$

and

$$^{G}C_i = {}^{G}_{B}R\,{}^{B}C_i$$

The same rotation matrix may be used to redefine the vectors $\vec{r}_1$ and $\vec{r}_2$:

$$^{G}\vec{r}_i = {}^{G}_{B}R\,{}^{B}\vec{r}_i = {}^{G}_{B}R \begin{bmatrix} L_{ix} \\ L_{iy} \end{bmatrix}$$

where L_{ix} and L_{iy} are vehicle parameters and are defined in Figure 49.

In the global frame, the weight vector takes a familiar form:

$$^{G}\vec{W} = \begin{bmatrix} 0 \\ -g \end{bmatrix}$$

Wheels

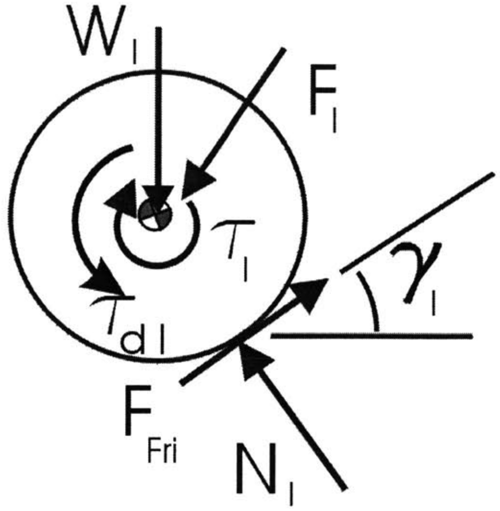

Figure 51: Free body diagram of a planar rover wheel.

The development of the equations of motion of the ith wheel, i={1,2}, is similar to that of the body. The general equations of motion are:

$$\sum F = \vec{F}_i + \vec{F}_{fri} + \vec{N}_i + \vec{W}_i = m_i \ddot{\vec{X}}_{wi}$$

$$\sum M_{cm} = \vec{\tau}_i + \vec{\tau}_{di} + \vec{r}_{wi} \times \vec{F}_i = I_i \ddot{\vec{\theta}}_i$$

where $\vec{r}_{wi}$ is the wheel radius, m_i is the wheel mass, and I_i is the wheel inertia. The vector location of the ith wheel's center of mass is:

$${}^{G}\vec{X}_{wi} = \begin{bmatrix} x_{wi} \\ y_{wi} \end{bmatrix}$$

Here, it is easiest to work in a reference frame with its x and y axis attached to the line of action of the frictional and normal force, respectively. This holds true because it allows for a partial decoupling between the friction and normal force. A rotation matrix that expresses all forces in the wheel-ground contact coordinate frame {wi}, defined in Figure 49, takes the form:

$$ {}^{wi}_{G}R = \begin{bmatrix} \cos(\gamma_i) & \sin(\gamma_i) \\ -\sin(\gamma_i) & \cos(\gamma_i) \end{bmatrix} $$

where γ_i is the wheel-ground contact angle, as shown in Figure 51. The ground contact angles may be estimated as described in [38]. The forces may then be expressed in the new coordinates:

$$ {}^{wi}\vec{F}_i = {}^{wi}_{G}R\,{}^{G}\vec{F}_i $$

and

$$ {}^{wi}\vec{W}_i = {}^{wi}_{G}R\,{}^{G}\vec{W}_i $$

The normal force is computed as:

$$ {}^{wi}\vec{N}_i = \begin{cases} K_{wi}\Delta\vec{X}_{wi} + C_{wi}\Delta\dot{\vec{X}}_{wi} = * & \mathit{if}* > 0 \quad \mathit{and} \quad \Delta X_{wi} < r_{wi} \\ 0 & \mathit{if}* < 0 \quad \mathit{or} \quad \Delta X_{wi} > r_{wi} \end{cases} $$

where K_{wi} and C_{wi} are the wheel stiffness and damping, respectively, r_{wi} is the maximum wheel radius, and $\Delta\vec{X}_{wi}$ and $\Delta\dot{\vec{X}}_{wi}$ describe the distance and rate of change of distance between the wheel-ground contact point and wheel center. Note that the normal force may only pull, and not push, on the wheel.

Then:

$$ {}^{wi}\vec{F}_{fri} = \begin{cases} \dfrac{2}{3}\left(\dfrac{\vec{\tau}_i + \vec{\tau}_{di}}{\vec{r}_{wi}} + {}^{wi}\vec{F}_i + {}^{wi}\vec{W}_i\right) = * & \mathit{if}* < \mu\,{}^{wi}\vec{N}_i \\ \mu\,{}^{wi}\vec{N}_i & \mathit{if}* > \mu\,{}^{wi}\vec{N}_i \end{cases} $$

All forces are then re-expressed in the inertial coordinate frame {G} by multiplying them by the inverse of ${}^{wi}_{G}R$.

Thus, there are 15 equations for the 15 outputs; nine equations of motion, four equations from wheel-ground interaction (ground heights and contact angles), and two friction force constraints. The system of equations may be solved in closed form.