Proprioceptive Control for a Robotic Vehicle Over Geometric Obstacles

Kenneth J. Waldron*, Ronald C. Arkin**, Douglas Bakkum**, Ernest Merrill** , Muhammad Abdallah*

*Stanford University
Department of Mechanical Engineering
Stanford, CA 94305

**Georgia Institute of Technology
College of Computing
Atlanta, GA 30332-0280

Abstract

In this paper we describe a software system built to coordinate an autonomous vehicle with variable configuration ability operating in rough terrain conditions. The paper describes the system architecture, with an emphasis on the action planning function. This is intended to work with a proprioceptive algorithm that continuously coordinates wheel torques and suspension forces and positions to achieve optimal terrain crossing performance.

Index terms: autonomy, rough terrain, proprioception, coordination, action planning

1. Introduction

It has long been recognized that an ability to vary the configuration of a vehicle has the potential to improve the mobility of that vehicle when crossing large obstacles [1-3]. However, there are very few examples in which systems have been developed to autonomously vary configuration in response to the terrain the vehicle is traversing. In this paper we describe a software system designed to achieve this in a wheeled vehicle with fully controllable, active suspension mechanisms.

1.1 Architecture for Variable Configuration Vehicles

Terrestrial vehicles operate in a very different environment from the largely isotropic media through which air and water vehicles travel. Particularly when operating off-road, land vehicles must respond to frequent changes in gradient and soil properties, while responding appropriately to the presence of obstacles. This means that trajectory, and vehicle configuration re-planning must take place very frequently. Further, in order to optimally respond to variations in the terrain over which the vehicle is passing, it is necessary to have a coordination process running continuously to optimally translate vehicle trajectory commands into commanded values for the actuators that drive the vehicle and control its suspension.

Based on experience with previous programs, we adopted a layered planning, coordination and control architecture. The upper layer plans vehicle trajectory and configuration in response to exteroceptive sensor data, including data from both imaging and GPS navigation sensors, together with status information passed up from the coordination system. Vehicles with active suspension capability can be configured to optimally meet obstacles. For example, when engaging a large positive obstacle, it is necessary to raise the front wheels so that they contact the obstacle above the "friction height" at which the vehicle can generate sufficient traction for them to roll over the obstacle. In land vehicle operation there is insufficient time to run replanning algorithms every time the terrain conditions change. We adopted an architecture with a relatively simple master program that would select from a library of stored plans, or behaviors, as dictated by environmental conditions. The selection is based on the closeness of fit of the modeled characteristics of the terrain to a set of discriminator statistics for each behavior.

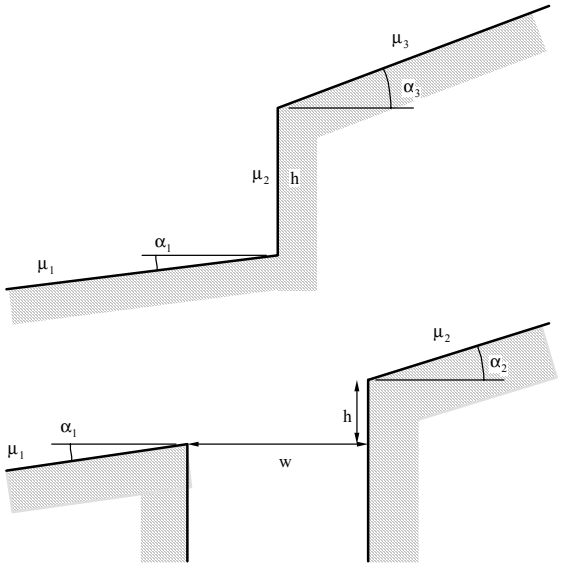

Figure 1: Geometric 0bstacle models in cross-section. The action planner selects from a library of such models. In turn, there is a library of possible action sequences for each model from which the planner selects.

The coordination, or proprioceptive, layer has the task of translating the vehicle velocity and configuration commands, generated by the active behavior, into commanded values of the actuator controllers. It responds to sensors internal to the system, hence the label "proprioceptive". These include actuator feedback sensors, additional sensors on the vehicle and suspension structures, and an inertial measurement unit. It translated the vehicle motion commands into force commands *via* dynamic models of the vehicle and its suspension elements. The system level commanded forces were allocated to the wheel and suspension actuators on the basis of an optimizing principle. It is possible to do this

using efficient, closed-form computations for physically meaningful optimizing principles [7]. This might be thought of as smart traction control in analogy to conventional vehicles. In fact the system does far more than that. The proprioceptive algorithm must run continuously at update rates of the order of 100 Hz. It passes status information to the upper layer, enabling that layer to determine when it is necessary to terminate a segment of a plan and move on to a new segment. The control layer consists of the actuator controllers. Coordination computations that have to be done at high update rates, such as those internal to a wheel-station with an active suspension, are also included in this level. This layer responds to the commanded force/torque, or position commands from the proprioceptive layer, and to the actuator feedback sensors.

2. Planning Layer

When crossing geometric obstacles, or near approximations to such obstacles, use of the variable configuration capability provides significant improvements in performance. Consequently, for obstacles in this category relatively complex configuration plans, called action sequences in the following, have been developed.

For more general obstacles, the proprioceptive algorithm assumes a more important role with its ability to respond continuously to changes in the terrain. In this case relatively simple action sequences are appropriate, such as simply pitching back when engaging a positive obstacle, using the proprioceptive algorithm to effect a crossing of the obstacle, then returning to the normal vehicle attitude.

For purposes of this study, trajectory planning for geometric obstacles consists of making an appropriate selection from among a library of action sequence plans associated with each obstacle map. A basic set of geometric obstacle maps is shown in Figure 1. The generalized step obstacle can be used in either direction to develop plans for both positive and negative obstacles. Each plan consists of a series of segments to be executed in sequence. A segment consists of a set of rules for generating position/velocity commands to the proprioceptive layer. The termination of a segment is determined by satisfaction of an inequality of a function of system variables.

The proprioceptive layer may also notify the planner of inability to complete execution of a segment due to wheel slip, instability, or some other condition. The planner will then search the library of action sequence plans for that obstacle for another valid plan, and may initiate execution of that plan.

3. System Design

Proprioceptive control was implemented using multiple distributed computational processes as shown in figure 2. Each layer in the system was implemented as its own process, although additional layers were included as necessary. Some additional processes used for mission specification and display, that are not germane to this paper, are not depicted.

The Mission Control Unit (MCU) and Vehicle Control Unit (VCU) are separate PC's connected by a backplane. The Robot Executable and HServer are processes running on the MCU. The upper, action-selection, layer was implemented in the Robot Executable, which is a program compiled for each particular mission, generated by the *MissionLab* software system[1] [4]. Vehicle trajectory in the form of waypoints to a specific goal location is compiled into the Robot Executable. In addition, the library of stored behaviors for each specific obstacle type is stored and accessed here. The HServer (Hardware Server) process is used to handle the details of controlling a particular type of robot so that the Robot Executable can be kept as general as possible. The coordination, or proprioceptive layer, was implemented in the VCU, while the control layer was simulated using Visual Nastran. IPT, an inter-process communications package from Carnegie Mellon was used to handle the communication between the Robot Executable and HServer. Low-level shared memory was used for communication between HServer and the VCU due to the real-time requirements of the proprioceptive layer.

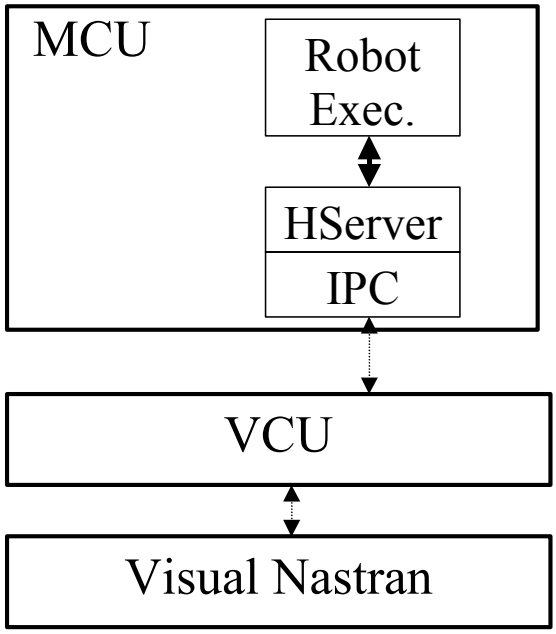

Figure 2: System Architecture

When an obstacle is encountered, a stored behavior is selected by the upper layer in the Robot Executable. The upper layer then sends high-level vehicle control commands for the heading, speed, acceleration and either wheel heights or vehicle attitude based on the current segment of the behavior (Table 1). These values act as biases for the lower layers. For example, the control layer

[1] *MissionLab* v5.0 is freely available at www.cc.gatech.edu/ai/robot-lab/research/MissionLab.html

will compute the best height of each wheel based on traction, with a bias towards the values provided by the upper layer.

In addition, triggering conditions are sent for transitioning to the next behavioral phase. For example, when a positive obstacle is encountered the first phase of the behavior calls for the vehicle to move forward, while pitching the front of the vehicle upward 0.45 radians. The upper layer can also specify that the trigger for moving to the next segment of the plan is for $|\xi_1, \xi_2| < 0.3$ radians, where ξ_1 and ξ_2 are the angles for the contact force line of action on wheels 1 & 2 and the z axis, respectively.

Parameter	Description
use_reverse	Flag to go in reverse
wheel_flags	Flag to indicate wheel heights to be commanded individually, all together (vehicle elevation) or vehicle attitude (roll, pitch) to be used instead.
wheel_1_height	Height of wheel 1.
wheel_2_height	Height of wheel 2.
wheel_3_height	Height of wheel 3.
wheel_4_height	Height of wheel 4.
wheel_5_height	Height of wheel 5.
wheel_6_height	Height of wheel 6.
vehicle elevation	Height of all wheels.
egocentric_heading_angle (yaw)	Commanded heading of vehicle, relative to vehicle coordinate system.
roll_angle	Commanded roll angle.
pitch_angle	Commanded pitch angle.
Speed	Commanded (forward) vehicle speed.
acceleration;	Commanded (forward) acceleration.

Table 1: Upper Layer commands to the Proprioceptive Layer.

The triggering mechanisms are quite flexible. Any number of conditions might be involved. For example, the upper layer could specify a trigger when any of the wheel heights on the left side is greater than any of the wheel heights on the right side. Or, one or more parameters could be compared to a numerical values, such as the $|\xi_1, \xi_2| < 0.3$rad example given above. Multiple triggering conditions could be specified for each state and identified with the request_num field of the message.

The strength in the design lies in its modularity and in the fact that the upper layer can execute at a much slower rate than is required by the lower layers. Since the triggering criteria is known by the proprioceptive layer, it can notify the upper layer of only those events that are of importance to the upper layer in its current state. This allows allows plan reconfiguration and interruption to occur from both the operator and high-level planner as well as due to unexpected interactions with the terrain or unpredicted vehicle performance.

4. Design of Action Sequences

The development of a plan to surmount a given obstacle is a design problem. There are an infinite number of possible solutions. The first decision was to use a segmented plan. This simplifies the problem in several ways. It removes the need for close coupling between the planning and proprioceptive algorithms. Further, it is necessary to change the set of commanded variables from time to time, particularly to accommodate a change of vehicle configuration. That inherently segments the plan into a sequence of actions. Configuration of the vehicle can be addressed on a coarse time-scale. The plan becomes an action sequence in which the commanded values of the vehicle motion and configuration variables are constant on each segment of the plan, but the variables and/or values change at the transition to the next segment.

The essential elements of each plan segment are the set of variables used and the commanded values passed down to the proprioceptive algorithm, together with the conditions that determine the termination of the segment.

It is necessary that the command variables chosen be consistent with the number of degrees of freedom of the vehicle. For a wheeled vehicle the non-holonomic nature of the wheel contact implies that lateral displacement or velocity cannot be directly controlled. Thus, the number of controllable degrees of freedom of the vehicle body is five, given that an active suspension allows controlled motion along the vehicle's vertical axis, and about the pitch and roll axes, as well as motion along the longitudinal axis and about the yaw axis.

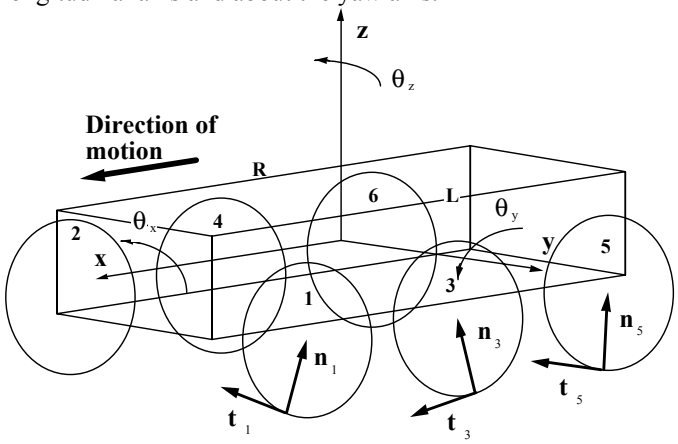

Figure 3: Vehicle coordinate system and motion degrees of freedom. The reference frame is aligned with the horizontal longitudinal axis of the body (x) and the vertical when the vehicle is resting on a level plane with the suspension positions neutral (z). Rotation about the x axis (θ_x) is referred to as roll, about the y axis (θ_y) is referred to as pitch, and about the z axis (θ_z) is yaw. Because of the non-holonomic nature of the wheel-ground contact lateral (y) displacement, or velocity, cannot be directly controlled. The commandable degrees of freedom are, therefore, x, z, θ_x, θ_y, θ_z. In order to control vehicle configuration, suspension positions may be substituted for some of these degrees of freedom.

The basic motion degrees of freedom are shown in Figure 3. However, it is also necessary to command the suspension position to control vehicle configuration. Suspension position commands can be substituted for motion commands, so long as the total number of five commanded variables is maintained. However, it is

necessary also to ensure that the command set is consistent. For example, commanding vehicle body height, by commanding position in the direction of the vehicle z axis is clearly inconsistent. Similarly, directly commanding suspension position on opposite sides of the vehicle determines the roll position. Thus, commanding a roll angle, or rate, in addition to the suspension commands would lead to an inconsistency. Likewise, if two suspension positions were commanded on the same side of the vehicle, commanding a pitch angle, or rate, would be inconsistent. It might be noted that, on uneven terrain these would not, in general, be inconsistencies in the strict kinematic sense. However, the algorithm would be extremely ill conditioned, and the vehicle behavior would be unacceptable from a practical point of view.

The procedure used for designing action plans was first to draw out the vehicle geometry in a series of positions that mark transitions. They are identified by changes in the system mechanics, caused by changes in the contact configuration: either new points of contact with the terrain, or the breaking of old points of contact. At each such transition position a set of commanded values is identified that will take the vehicle to the next position. Also, one or more inequality conditions that identify the next transition position are formulated. Figure 4 shows examples of transition positions together with commanded variables and segment end conditions. Note again that this is a design problem. For a vehicle with given geometry there are multiple possible action plans that could be used to cross a given obstacle.

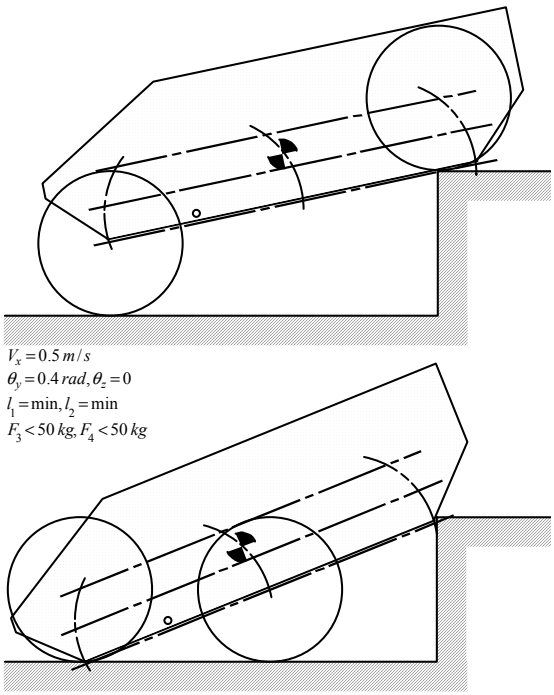

Figure 4: Two successive positions in a crossing of a negative step obstacle, as laid out graphically during the development of an action sequence. Here the front suspension is being retracted to ground the middle wheels. Hence the commanded variables are longitudinal velocity, pitch angle, yaw angle and the positions of the two front suspensions. The segment continues until the middle wheels are grounded as detected by the contact forces on both those wheels exceeding a threshold value.

Each plan was then laid out in the form of a flow chart. This was found to be useful in explicating the interactions between the planner and the proprioceptive algorithm. A small portion of such a chart is shown on Figure 5. In this figure, the arrows represent stimuli, or commands. The circles represent actions of the planner, and the rectangles represent the responses of the proprioceptive software. The rhomboids represent inequality tests.

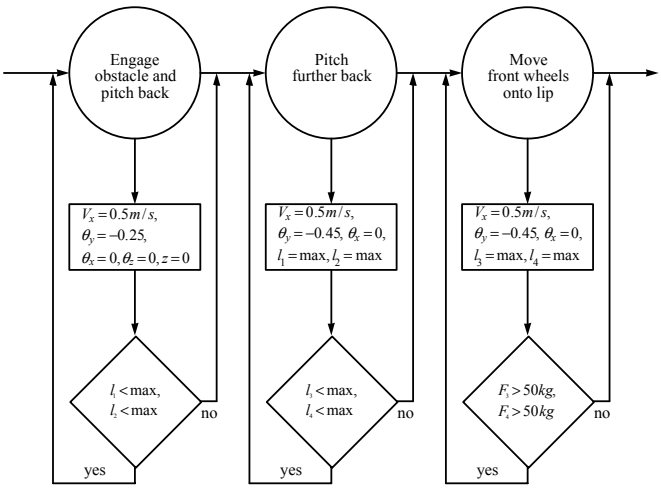

Figure 5: A small part of the flow chart of a plan for surmounting a positive step obstacle. The circles represent the plan segments executed in the planner. The values in the rectangles represent the commanded variables transmitted to the proprioceptive algorithm. The diamonds represent value tests to determine whether the current plan segment should continue or be terminated.

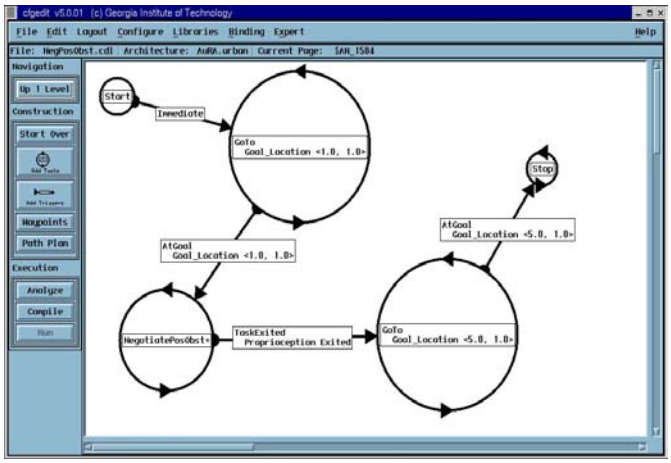

Figure 6: *MissionLab* / Planning Layer GUI. The NegotiatePosObst* contains the action sequence used in the simulation studies.

At this point the plan was in a form closely related to the stimulus/response semantics of *Missionlab* [4]. It was encoded in CDL {5] and implemented in *Missionlab* as

shown in Figures 6 and 7. Traditionally, *MissionLab* has been used in applications in a planar environment by sending commands of two degrees of freedom to the vehicle: heading and speed. However, when using an actuated suspension, three additional degrees of freedom are added: ride height, roll, and pitch. *Missionlab* was modified to accommodate these additional parameters.

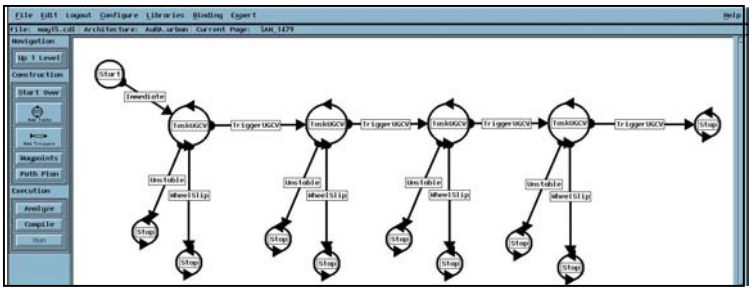

Figure 7: A general action sequence in *MissionLab*, as is used by the NegotiatePosObst* task. The circles represent action segments (command of the five degrees of freedom is specified here). The rectangles represent information returned by the coordination layer. Two failure modes are denoted, but more can be added.

The planning layer implemented using *MissionLab* was chosen to be distinct from the coordination layer, in order to provide a user interface abstracted from the detailed computations involved with the wheel and suspension motors: only high level vehicle commands need be of concern when constructing action sequences. This and a graphical user input of vehicle parameters via slider bars allows simple construction and modification of action sequences for development and testing purposes.

Once the vehicle is in action, the planning layer receives input from the exteroceptive sensors (e.g., vision, ladar) to classify any navigable obstacles or terrain, and then chooses an appropriate action sequence to navigate the terrain. The communication between the planning layer and the coordination layer is bidirectional (Figure 8). After an action sequence is selected, the planning layer sends a set of numerical constant parameters of the type double or float to the proprioception layer. The constant parameters represent both (1) commands specifying the five degrees of freedom for each action segment, and (2) specification of an inequality signifying the timing to transition between action segments. The coordination layer then sends back (1) a Boolean Yes/No command to a requested transition between action segments and (2) an integer specifying the occurrence of a failure mode. Communication rates of 10 Hz are sufficient.

In one sense, the coordination layer acts as a translator between the planning layer and the wheel and suspension motors; therefore, the planning layer is general and can remain the same for subtle changes in vehicle geometries. The coordination layer adjusts the wheel and suspension motor parameters to satisfy the specification on the degrees of freedom: heading, speed, and vehicle attitude (roll, pitch, and ride height; or individual wheel suspension heights).

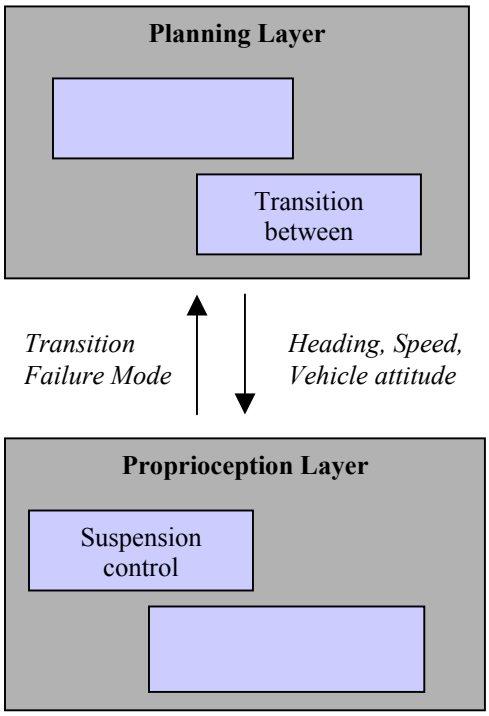

Figure 8: Action sequence communication between the planning and coordination / proprioception layers.

In addition, the planning layer sends the parameters of an inequality (or a set of inequalities) to the coordination layer that once met, signifies satisfactory completion of a segment of the action sequence. The coordination layer uses proprioceptive sensors to monitor the inequality and returns a Boolean Yes/No upon its satisfaction.

The planning layer receives two types of feedback from the coordination layer: the above mentioned signal to transition, and also a vehicle failure notification such as 'wheel slip' or 'unstable'. Depending on the failure mode, a regression to a prior action segment will occur, or a new action sequence will be chosen.

5. Proprioception

As was indicated above, the proprioceptive software translates body motion and configuration commands from the motion planner into force and torque commands to the traction and suspension actuators. The proprioceptive software receives inputs from sensors internal to the vehicle such as traction motor torque sensors, suspension member force sensors, suspension and wheel encoders and/or tachometers, body mounted and suspension-mounted accelerometers etc. This includes an inertial measurement unit strapped down to the vehicle body.

The proprioceptive algorithm must run continuously at update rates of the order of 100 Hz. Therefore, speed of computation is an issue. Its function is very similar to the

corresponding algorithm used for the Adaptive Suspension Vehicle project [6,7], despite the apparently very different locomotion mechanics of a walking machine *versus* a wheeled vehicle with active suspension. The major differences in the software for a wheeled robotic vehicle with active suspension and a walking machine are actually at the action planning level, not the proprioceptive level. In both cases, the fundamental physical principle on which the proprioceptive algorithm is built is minimizing the tendency of the locomotion element to slip. In fact, the exact same algorithm could have been used. However, it does not perform well in extreme terrain.

The algorithm actually used, and the underlying physical and mathematical models, will be described in detail elsewhere [8], as space prevents a complete exposition here. It is based on decomposition of the commanded force system calculated for the vehicle into planar force systems in the central planes of the wheels on either side of the vehicle. In this respect it does require that the wheels on each side of the vehicle remain close to coplanar regardless of suspension movements.

The broad principles of operation of the proprioceptive software are that the commanded values received from the upper layer are compared to the corresponding actual values of the vehicle motion variables as measured by the IMU to generate a rate error system. This is divided by a time interval to create an acceleration system that is multiplied by an inertia matrix to generate an inertial force system. That is combined with the vehicle weight to generate a commanded force system on the vehicle body. That force system is then decomposed into the two planar force systems based on the wheel center planes on either side of the vehicle body. Lateral force is split between those force systems in proportion to the respective resultants.

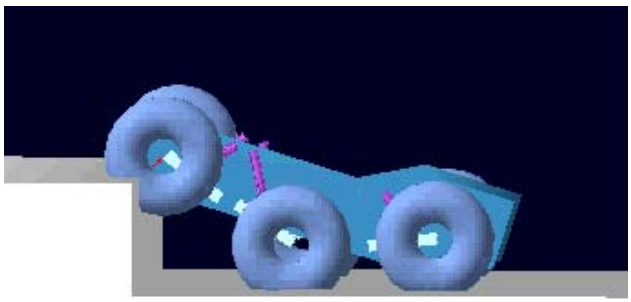

Figure 9: A frame from the ADAMS simulation of the vehicle crossing a 1 m positive step obstacle. Tire deflections were not graphically modeled, so the apparent penetration of the tires is displaying tire deflections.

The algorithm then allocates the in-plane forces to the wheel-ground contacts according to one of two principles. In easy terrain a simple form of the zero interaction force principle is used. This distributes weight and traction force as evenly as possible, and is optimal in weak soil conditions [7]. In strongly uneven terrain a different principle is used to effectively maximize available traction. Heuristically, the way this works is to select the two wheels on a side of the vehicle that are geometrically best located to generate traction. The third wheel is then unweighted to place as much load as possible on the two wheels identified. Basically, tangential contact, or traction force is generated by friction with the normal load being generated by the vehicle weight. It therefore is optimal to place the weight load on those wheels that are best positioned to generate traction force.

Of course, the above is a broad brush description. There are a number of special cases that need to be accommodated, such as those in which several wheels are out of contact with the ground, or those in which one or more suspensions are against the bump stops.

6. Simulation Studies

Simulation models of the vehicle were developed in both Visual Nastran and ADAMS. The former was a high fidelity model including detailed models of the wheel-ground mechanics, and the active suspension mechanism. The latter was a simplified model that used linearized active suspension and tire models, and a Coulomb friction model of the wheel-ground contact. The former model ran much slower than real-time and proved to be very cumbersome for purposes of debugging the coordination software. Thus, although the first version of the action sequence planner was to have been tested on this platform, it did not run successfully.

The proprioceptive software was successfully run on the ADAMS simulation platform, which ran at near real-time speeds with a very simplified action sequence. The simulated vehicle did successfully cross a 1 m step obstacle at 10 m/s. Figure 9 is a frame from this simulation.

7. Discussion

We have described work done on coordination of a robotic vehicle with variable configuration capability in the form of an active suspension system. The system described was fully designed and coded. Initial testing using fully dynamic vehicle models was successful. Work continues on testing of the full system against a high-fidelity simulation model.

Acknowledgments

This research was supported by DARPA's Unmanned Ground Combat Vehicle Program and SAIC Corporation. The authors would also like to thank Mr. Raghavan Madhavan for his work in coding the proprioceptive algorithm.

References

[1] Rayfield, W.P., 1970, "Mars Roving Vehicle Configuration", R.P.I. Technical Report MP-16 to the National Aeronautics and Space Administration.

[2] Hirose, S., Aoki, S., Miyake, J., "Design and Control of Quadry-Truck Crawler Vehicle Helios-II", *RoManSy 8 Proceedings*, Warsaw University of Technology Publications, 1990, pp. 320-329.

[3] Murphy, R.R., 2000, "Marsupial and shape-shifting Robots for Urban Search and Rescue", *IEEE Intelligent Systems* 15(3), pp. 14-19.

[4] MacKenzie, D., Arkin, R.C., and Cameron, R., "Multiagent Mission Specification and Execution", *Autonomous Robots*, Vol. 4, No. 1, Jan 1997, pp. 29-52.

[5] User Manual for MissionLab V5.0, Georgia Tech Mobile Robot Laboratory, College of Computing, January 2002.

[6] Pugh, D.R., Ribble E.A., Vohnout V.J., Bihari, T.E., Walliser, T.M., Patterson, M.R., Waldron, K.J., 1990, "Technical Description of the Adaptive Suspension Vehicle," *International Journal of Robotics Research*, Vol. 9, No. 2, pp. 24-42.

[7] Kumar, V. and Waldron, K. J., "Force Distribution in Closed Kinematic Chains" *IEEE Transactions on Robotics and Automation,* Vol. 4, No. 6, December 1988, pp. 657-664.

[8] Waldron, K.J. and Abdallah, M., "A Traction-Optimizing Coordination Algorithm for Off-Road Operation of Robotic Vehicles", in preparation for *IEEE/ASME Transactions on Mechatronics*.