Terrain aware inversion of predictive models for high performance UGVs

Alonzo Kelly, Thomas Howard, Colin Green
Robotics Institute, Carnegie Mellon University, 500 Forbes Ave, Pittsburgh, PA, 15213;

ABSTRACT

The capacity to predict motion adequately over the time scale of a few seconds is fundamental to autonomous mobility. Model predictive optimal control is a general formalism within which most historical approaches can be cast as special cases. Applications continue to grow in ambition to seek higher precision of motion and/or higher vehicle speeds. Predictions must therefore improve in fidelity and speed simultaneously.

We favor an approach to precision motion control that we call parametric optimal control. It formulates the optimal control problem as one of nonlinear programming - optimizing over a space of parameterized controls encoding all feasible motions. It enables efficient inversion of the solutions to the equations of motion for a ground vehicle. Such an inversion enables a computation of precisely the command signals necessary to drive the vehicle to goal position, heading, and curvature while following the contours of the terrain under arbitrary wheel terrain interactions.

Dynamics inversion is so fundamental that many other mobility behaviors can be constructed from it. Fielded applications include pallet acquisition controls for factory AGVs, high speed adaptive path following for military UGVs, compensation for wheel slip on the Mars Exploration Rovers and full configuration space planning in dense obstacle fields.

Keywords: autonomous mobility, unmanned ground vehicle, model predictive control, trajectory generation.

1. INTRODUCTION

The problems of motion planning and control for unmanned ground vehicles exhibit several important distinguishing characteristics. In challenging terrain, both halves of the expression “intelligent mobility” can be a challenge to achieve. Effective wheeled vehicle mobility depends on the capacity of the terrain to react to propulsive and steering forces generated by the vehicle. Effective control depends on a capacity to predict these forces and/or compensate for related disturbances. Those forces depend on terrain mechanical properties that can be difficult to predict. While terrain mechanical descriptors may not be entirely predictable, they can be predicted somewhat and, one way or another, every model of vehicle motion makes assumptions about how the terrain can be induced to propel the vehicle.

Despite the difficulty of modeling motion well, autonomous mobility presents the compounding difficulty of predicting motion quickly. One rule of thumb is that the mobility prediction module, only one of several compute-intensive components of a typical system, must be three orders of magnitude faster than real time. In very rough terms, a system must replan every tenth of second or so, each plan should probably consider at least ten alternative motions, and each alternative should predict about ten seconds into the future. Hence, in 1/10 second, 100 seconds of simulation must be performed and only a fraction of the CPU is available to do this.

The crowning complexity of this computational house of cards is the desire for autonomous vehicles to travel at reasonably high speeds. As speeds increase, inertial forces increase and higher demands are placed on the terrain to produce higher magnitude reactions. More complex models are therefore necessary to predict when the terrain will fail mechanically - leading to more complex states of motion such as rollover and sliding. While autonomous vehicles will certainly not plan to rollover, a capacity to predict this state of motion is necessary to enable its prevention.

Not only do the models becomes more complex at higher speeds but it becomes necessary to predict motions a longer distance into the future. Time steps cannot be increased proportionally without sacrificing accuracy, so each candidate motion requires more time steps as well as more complication per step. In summary, intelligent mobility produces severe requirements for computation to be both accurate and fast. We will present in this paper an effective approach for precision control in difficult terrain – even for relatively high speed vehicles.

1.1 Vehicle Modeling

A somewhat general description of vehicle dynamics is a nonlinear vector differential equation of the form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{1}$$

$$\mathbf{C}(\mathbf{x}, t) = \mathbf{0} \tag{2}$$

The model may be kinematically driven with velocity inputs or dynamically driven with forces, as the case requires. Let the state vector $\mathbf{x}$ include at least the position (x,y,z) and orientation (ϕ,θ,ψ) of the vehicle expressed relative to a frame of reference fixed to the ground.

Very often, the propulsive forces generated by the vehicle are limited in magnitude and direction for several reasons. Under conditions which enforce terrain contact, roll (ϕ), pitch (θ), and elevation (z) are predetermined by terrain shape so three of the six degrees of freedom available to a rigid body are constrained. Furthermore, nonholonomic constraints (expressing the incapacity of wheels to move sideways) do not entirely apply under conditions of wheel slip but in either case, wheeled vehicles remain underactuated whether the wheels slip or not. Typically, the two degrees of freedom of linear velocity in the forward direction and angular velocity in the local tangent plane are actuated indirectly by adjusting fuel flow rates, wheel/track speeds, steer angles etc. This means one rate degree of freedom out of the remaining three $(\dot{x}, \dot{y}, \dot{\psi})$ is lost.

The impact of these facts on planning and control is that even the simplest useful models of mobility are underactuated differential equations. Furthermore, terrain shape must be known to predict motion because steering takes place in the instantaneous terrain tangent plane. Intuitively, the same steering signal executed on flat and then on nonflat terrain will drive a vehicle to two different places. Nonflat terrain also increases coupling in the model because the projections of the angular velocity vector onto the attitude and heading states depend on those states themselves.

1.2 Problem Definition

In many contexts, the pose of the vehicle (x, y, ψ), not its rate, is the variable of interest and this problem of precision pose control is the one we address here. The task of driving a fork truck to pick up a pallet (Figure 1) illustrates the problem well.

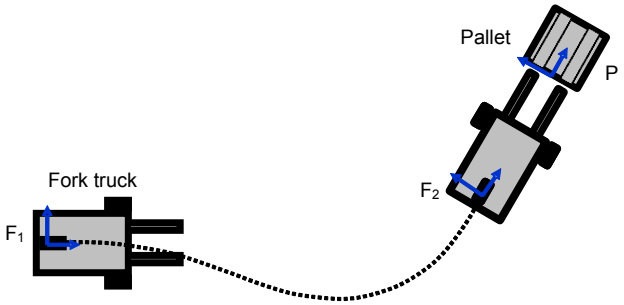

Fig. 1. In order to pick up a pallet, a fork truck must achieve a fairly precise target posture characterized by constraints on position, heading, and zero curvature.

Vehicle mobility constraints assert that the pallet cannot be approached by moving sideways. Therefore, consider solving the problem in reverse by imagining a vehicle at the terminal state which is moving backward toward the start. Consider turn radius and its rate to be limited. It becomes intuitively clear that the initial maneuver in the forward direction must be a turn to the right although the pallet is initially to the left of the fork truck. If the fork truck simply drives left toward the pallet, it will not achieve the right heading. This difficulty arises in part because of underactuation. It is not possible to drive toward the goal and then rotate to the correct heading at the last minute because heading cannot be changed quickly or independently from position.

The task of controlling vehicle pose is that of inverting the system dynamics differential equation to produce the controls which will achieve a goal terminal state. Since the terminal state is an integral of the dynamics, the problem is to select a function $u(t)$, which when placed inside this dynamics integral, will produce the desired terminal state:

$$\mathbf{x}_f = \int_0^{t_f} \mathbf{f}(\mathbf{x}, \mathbf{u}, t) dt \qquad (3)$$

There will usually be an entire continuum of functions $u(t)$ which could solve the problem but only one is needed. Intuitively, a slight change to the beginning of a solution can be compensated by another later while still achieving the desired terminal state. A human driver somehow solves this problem with little effort but techniques to solve it computationally are far from immediately obvious.

1.3 Prior Work

Trajectory generation has ancient roots in the theory of differential equations, the calculus of variations, and Lagrange's variational approaches to dynamics. From basic mathematics, techniques are available to both optimize an objective functional over an unknown curve and to require that boundary conditions on the solution be satisfied. Some of the earliest work related to trajectory generation of direct relevance to robotics is work on curves of minimal length under constraints on curvature [1]. Kanayama's original work on clothoids introduced the idea of using continuous piecewise linear curvature curves for robot trajectory generation [2]. Some work has concentrated on the problem of producing sequences of simple primitives which are optimal overall [3].

In [4], a holonomic geometric path is found in an obstacle field and path segments are smoothed using optimal control. A near real-time optimal control trajectory generator is presented in [5], which solves eleven first-order differential equations subject to the state constraints. Work on the problem of trajectory generation in arbitrary terrain is rare. In [6] a planner is presented which initially assumes flat terrain but then trims the results to accommodate rough terrain.

Our own work on this topic in recent years is summarized in [7][8].

1.4 Approach

Our approach accepts and exploits several aspects of the overall context of intelligent mobility. Such systems are fundamentally deliberative in the sense that they predict the consequences of several candidate actions and then choose one based on some implicit or explicit optimization criterion. Planning systems of capable off road robots already incorporate somewhat high fidelity (forward) predictive models in order to competently perform such tasks as avoiding obstacles and following paths. Our approach can be viewed as one of simply making these models available to lower levels of control and introducing the valuable capacity to solve them in reverse to enable precision control.

It is well known that even simple curvature actuated versions of the dynamics integral cannot be solved in closed form. This fact, as well as the sampled nature of terrain representations produced by terrain sensing, argue for the use of numerical methods to evaluate the dynamics integral. This numerical integratio0n approach has been used more or less universally for off-road robots.

Rather than search the continuum of all possible controls, we instead approximate this space by a space of parameterized controls. Significant efficiency is gained as a result in return for a negligible loss of control expressiveness. The parameterized problem remains nonlinear and we solve it using standard techniques of linearization and iteration. The most significant aspect of our approach is the numerical approach to linearization. We compute Jacobians of the terminal state with respect to parameter variations numerically. This means we can invert models of arbitrary complexity including elements of terrain following, actuator dynamics, wheel terrain interaction, arbitrary kinematics, power plant dynamics etc.

2. FORMULATION AND IMPLEMENTATION

This section presents the overall mathematical formulation as well as the software implementation.

2.1 Architecture

A three level, three loop architecture is used as illustrated in Figure 2. The highest level loop (Trajectory Generation - left side) inverts the dynamics integral. The medium level loop (Motion Prediction – center) integrates the equations of motion. The lowest loop (Vehicle Model – right side) is the differential equation itself. These elements are distinguished in software for reasons of generality. While the vehicle model is clearly vehicle dependent, the solution for integration of

the differential equation and the solution of the nonlinear parametric terminal state equations are entirely vehicle independent.

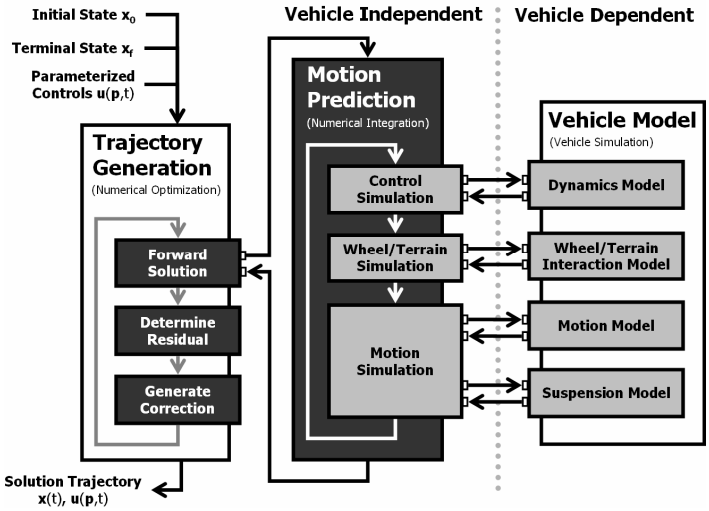

Fig.2 . The system architecture distinguishes three main elements, only one of which is vehicle dependent. Each represents a computational loop where the innermost loop is on the right side.

2.2 Parameterized Controls

A parameterized control is a mapping from a parameter vector (a few numbers) onto a continuous function. The fact that the parameters express a relatively small number of degrees of freedom is key to their capacity to reduce computation. Figure 3 illustrates two typical cases: a trapezoidal linear velocity control, and a polynomial angular velocity control.

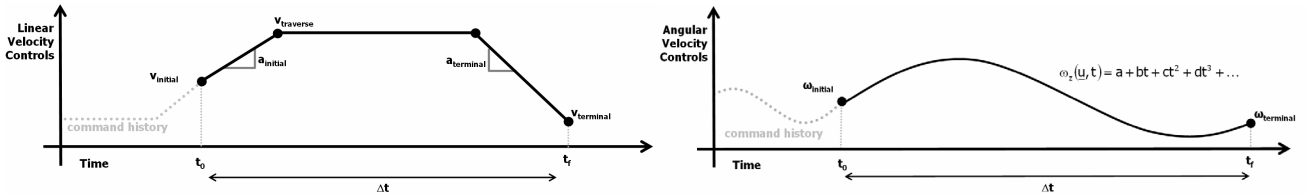

Fig.3. Two examples of parameterized controls. Any parameterization creates a mapping from a few numbers (the parameters) onto a continuous function.

The corresponding parameter vectors are:

$$\mathbf{p}_{|\mathbf{v}|} = \left[v_0\ a_0\ v_{traverse}\ a_f\ v_f\ \Delta t\right]^T \qquad (4) \qquad \mathbf{p}_{\omega_z} = \left[a\ b\ c\ d \ldots \Delta t\right]^T \qquad (5)$$

2.3 Trajectory Generation

While the general mechanism used can optimize an arbitrary functional, the simplest case of terminal constraint satisfaction will be presented for reasons of limited space. The numerical method applied is Newton's method, where the Jacobian of the dynamics integral is inverted to find a correction to the parameters needed to minimize the constraint error $\mathbf{\Delta x_f(p)}$, defined as the difference in the predicted terminal state (for the present parameter values) and the desired terminal state. The iteration proceeds as follows:

$$\left[\frac{\partial \Delta \mathbf{x_f}(\mathbf{p})}{\partial \mathbf{p}}\right] \Delta \mathbf{p} = \left(\mathbf{x_f} - \mathbf{x}(t_f)\right) = -\Delta \mathbf{x_f}(\mathbf{p}) \qquad (6)$$

$$\Delta \mathbf{p} = -\left[\frac{\partial \Delta \mathbf{x_f}(\mathbf{p})}{\partial \mathbf{p}}\right]^{-1} \Delta \mathbf{x_f}(\mathbf{p}) \tag{7}$$

Since the partial derivatives of dynamics integral cannot generally be found analytically, estimates must be found numerically. Forward (eq. 8) or central difference (eq. 9) linearizations of the integrals of the equations of motion can be used to estimate these partial derivatives with respect to the parameters.

$$\frac{\partial \Delta \mathbf{x}_{i,j}(\mathbf{p})}{\partial \mathbf{p}_k} = \frac{\Delta \mathbf{x}_{i,j}(\mathbf{p}_k + e, \mathbf{p}) - \Delta \mathbf{x}_{i,j}(\mathbf{p})}{e} \tag{8}$$

$$\frac{\partial \Delta \mathbf{x}_{i,j}(\mathbf{p})}{\partial \mathbf{p}_k} = \frac{\mathbf{q}_{i,j}(\mathbf{p}_k + e, \mathbf{p}) - \mathbf{q}_{i,j}(\mathbf{p}_k - e, \mathbf{p})}{2e} \tag{9}$$

2.4 Motion Prediction

Any method for integrating a differential equation is potentially applicable at this step. Although Runge-Kutta may be more efficient, we have been able to use Euler's method of integration so far to determine the change in vehicle state over a small time step (Δt):

$$\mathbf{x}(t + \Delta t) = \mathbf{x}(t) + \dot{\mathbf{x}}(\mathbf{x}, \mathbf{u}, t)\Delta t \tag{10}$$

Some formulations enforce constraints such as terrain contact using constraint drift terms. We typically enforce them explicitly at each time step by computationally allowing the vehicle to settle on the terrain to achieve minimum error between wheel contact points and terrain elevations. This procedure also produces suspension deflections as a byproduct in quasi-static cases.

2.5 Vehicle Model

Of course, a real vehicle is at least as complicated as a system of rigid bodies. Representing the general case would require models of not only terrain interaction but also the functioning of the control system. That is, a closed loop model is necessary at the system level. However, the fact that an autonomous vehicle is not a passive system can sometimes be used to simplify the model. Closed loop dynamics may be simpler than equivalent open loop ones because the controller will reject disturbances that might not have been predictable open loop.

Our models in most cases are kinematically actuated – driven by velocities rather than forces. Such models are sufficiently accurate for our purposes but much faster to compute than doubly integrated, dynamically actuated models. Velocities requested by the autonomy layer are mapped onto desired wheel speeds. Effects such as wheel slip can be incorporated at this stage by expressing their dependence on speed and slope to produce the response wheel speeds. These responses are mapped onto response velocities of the entire system. Using an x-y-z Euler angle sequence, the mapping from body frame linear and angular velocities onto their equivalents in the world frame are:

$$\dot{\mathbf{r}}\big|_{\mathbf{world}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & c\psi s\theta c\phi + s\psi s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\psi s\theta c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{11}$$

$$\Psi\big|_{\mathbf{world}} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & t\theta s\phi & -t\theta c\phi \\ 0 & c\phi & s\phi \\ 0 & -\frac{s\phi}{c\theta} & \frac{c\phi}{c\theta} \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{12}$$

These equations are then integrated in equation 10.

3. PERFORMANCE

The algorithm deals with nonlinearity by local linearization and iteration. It therefore becomes natural to wonder about issues of convergence, optimality, and runtime. In all of the applications discussed later we have never had difficulty with either convergence or optimality. Although the algorithm is a local optimization approach which essentially follows the gradient of terminal state error, local minima do not occur in practice within the radius of practical initial guesses. An initial guess for the parameters of the control is needed to start the iterations. We use the trajectory generator itself to generate lookup tables of these initial guesses in an off line process where runtime is not important and each new solution is started from an adjacent one.

3.1 Convergence

We typically require terminal state error to be an order of magnitude less than may really be necessary - within a millimeter in position and a milliradian in heading. In order to illustrate convergence, Figure 4 shows the terminal state error evolution for a case where the initial guess was artificially set to two orders of magnitude higher than is typically the case.

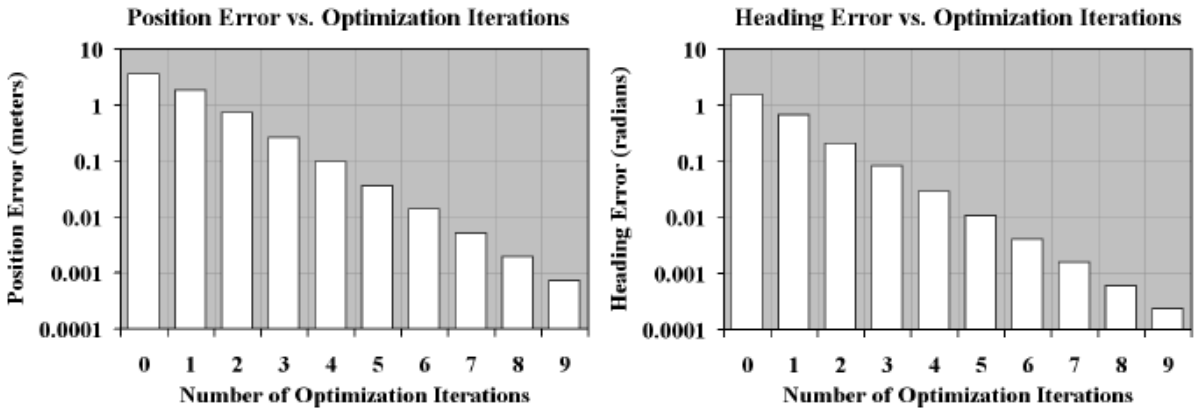

Fig.4. Convergence. Quadratic convergence is typical in Newton's method. Here, every two iterations reduces error by a decade.

Observe that the error in both position and heading decreases steadily and exponentially. Every two iterations reduces the error by a decade. In a realistic case, the error would start near 0.1 m in position and be reduced below a millimeter in 4 iterations.

3.2 Runtime

Runtime depends somewhat on terrain roughness. Over a large number of test cases, a factor of 3 separates the easy cases from the difficult ones. As illustrated in Figure 5, most of this is due to an increase in iterations.

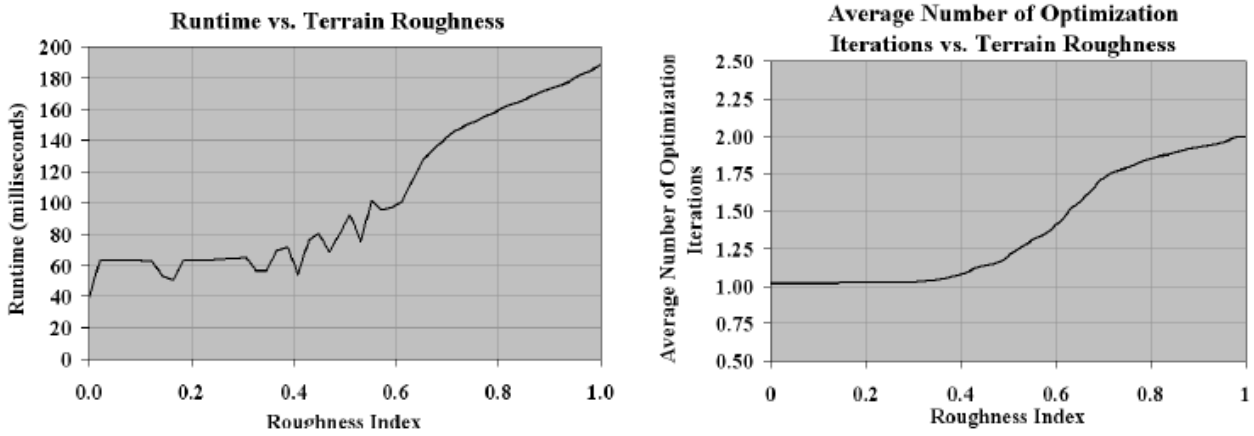

Fig.5. Algorithm Runtime. These results are for several hundred highly complicated test cases run on a 1.8 GHz laptop computer.

The above results are for a highly complicated 12 actuator vehicle model including wheel slip and very rough terrain. Trajectory generation queries for simpler vehicle models on flat terrain are solved in under a millisecond.

4. APPLICATIONS

The above algorithm has been refined on numerous mobile robot research programs over the last decade. The core capacity of contemporary autonomous vehicles enables them to drive over terrain moving from A to B and perhaps avoid obstacles and achieve waypoints. However, vehicles which interact more directly and productively with their environments must often do so from specific poses and configurations that must be occupied fairly precisely. Such applications motivated the original work but the resulting solution has applications well beyond them. Some of the applications that we have addressed are listed below.

4.1 Basic Pose Control

The original motivation was for an automated fork truck as described earlier. Here the core motivation is a desire to compute in real-time a trajectory which permits a pallet to be addressed from the correct position and heading with zero curvature. With reference to Figure 1, the problem is to determine a feasible motion from frame F1 to frame F2 given a measurement of the relationship between frame F1 and frame P.

At a more fundamental level, the trajectories themselves are useful as a description of a desired motion. We have used them to encode guidepath networks in factories. A guidepath is an invisible roadway through the plant along which the Automated Guided Vehicles (AGVs) are confined to move. By using curvature controls that are cubic in distance, sufficient degrees of freedom exist to ensure that curvature is continuous where two primitive motions join. This leads to reduced disturbances and higher performance path following by the AGVs.

A related advantage is the fact that the specification of an entire time or space continuous motion informs lower level controllers about the instantaneous derivatives of the reference signals expressing desired motions. Such derivatives can be used effectively in feedforward control. Likewise, the explicit expression of the future content of the reference signal enables model predictive approaches to control.

4.2 Off Road Path Following

A class of path following algorithms is based on the concept of a corrective trajectory terminating at a reacquisition point somewhere forward on the path. We have implemented an adaptive version of such an algorithm and demonstrated it on the Crusher platform. As illustrated in Figure 6, a search is conducted along the target path to determine the best tradeoff between the aggressiveness of the maneuver and the crosstrack error. Each point of reacquisition has the correct heading and curvature and the optimal control formulation adjusts lookahead automatically as speed, path curvature, and crosstrack error vary over time.

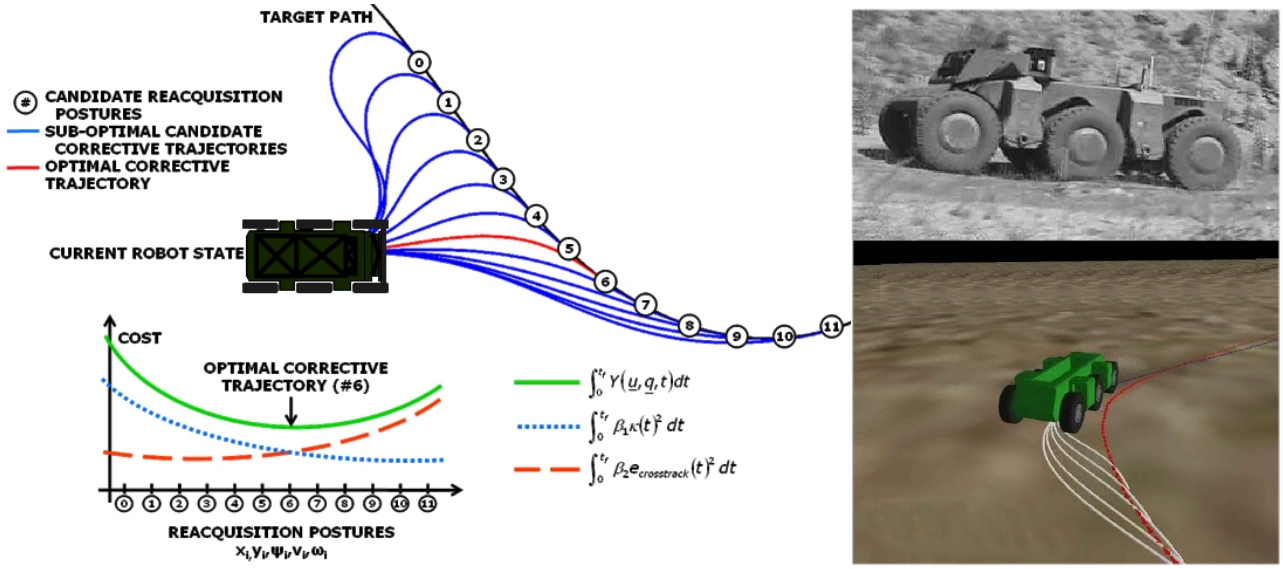

Fig.6. Adaptive Lookahead Path Following. A model predictive optimal controller chooses an acquisition point on the goal path which is the best tradeoff between aggressiveness and crosstrack error.

4.3 Model Predictive Slip Compensation

NASA has funded our group to develop trajectory generation algorithms for potential use on the next generation of Mars rovers. We have used our algorithms to develop mechanisms to predictively compensate for wheel slip. As Figure 7 shows, models of how wheel slip depends on slope can be inverted readily to cause the vehicle to approach a sloped goal on the high side in anticipation of sliding into it from above. Of course, such models will never be perfectly accurate but continuous compensation of this nature is likely to be far more effective than ignoring slip entirely.

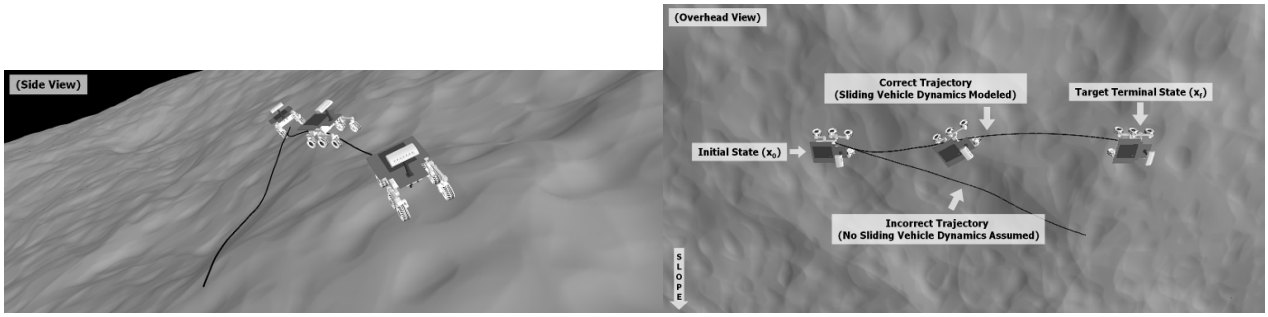

Fig.7. Predictive Slip Compensation. The vehicle model in this cases anticipates a sideslip velocity which is proportional to slope and forward velocity. The trajectory generator automatically compensates by approaching the goal from above.

4.4 Control of Hyperactuated Systems

A secondary goal of the rover work is to exploit the full maneuverability of a vehicle with six steered and driven wheels. As shown in Figure 8, our algorithm can exploit the capacity of this rover to drive sideways in order to orbit around a "science target" (a rock) and deploy instruments most effectively.

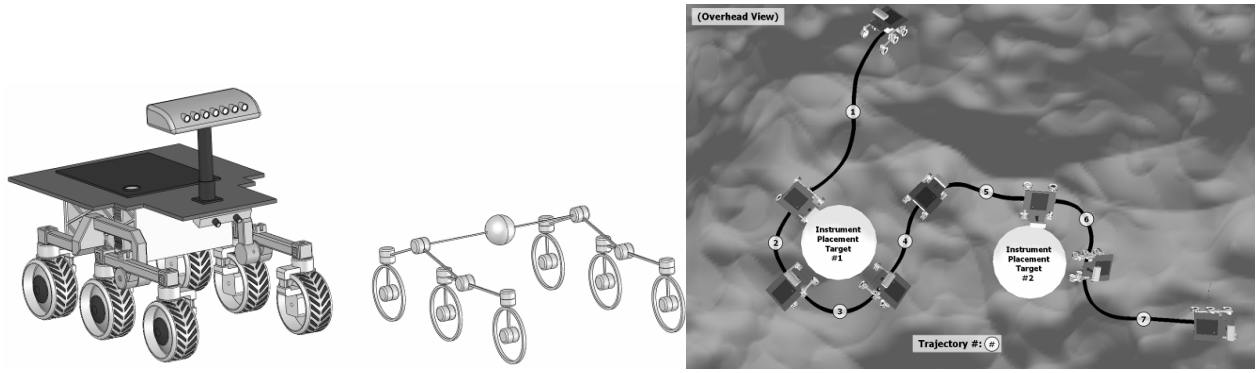

Fig.8. Hyperactuated System Control. The Rocky8 Mars rover prototype has six drive and steered wheels. Exploiting its capacity to move sideways leads to efficient instrument placement.

4.5 On-Road Search Space Generation

Driving and avoiding occasional obstacles on roadways is a difficult problem in part because any trajectories which avoid an obstacle must remain on the road before, during, and after the avoidance maneuver. Lane changes require trajectories with similar properties. This is a case of very highly constrained motion planning because trajectories must satisfy not only the constraints of dynamic feasibility but also those of remaining on the road and avoiding the obstacle. A trajectory generator is an ideal solution to this problem because it can produce motions which are laterally offset from some nominal motion and such motions satisfy all constraints. Our algorithm has been used for this purpose on the DARPA Grand Challenges as illustrated in Figure 9.

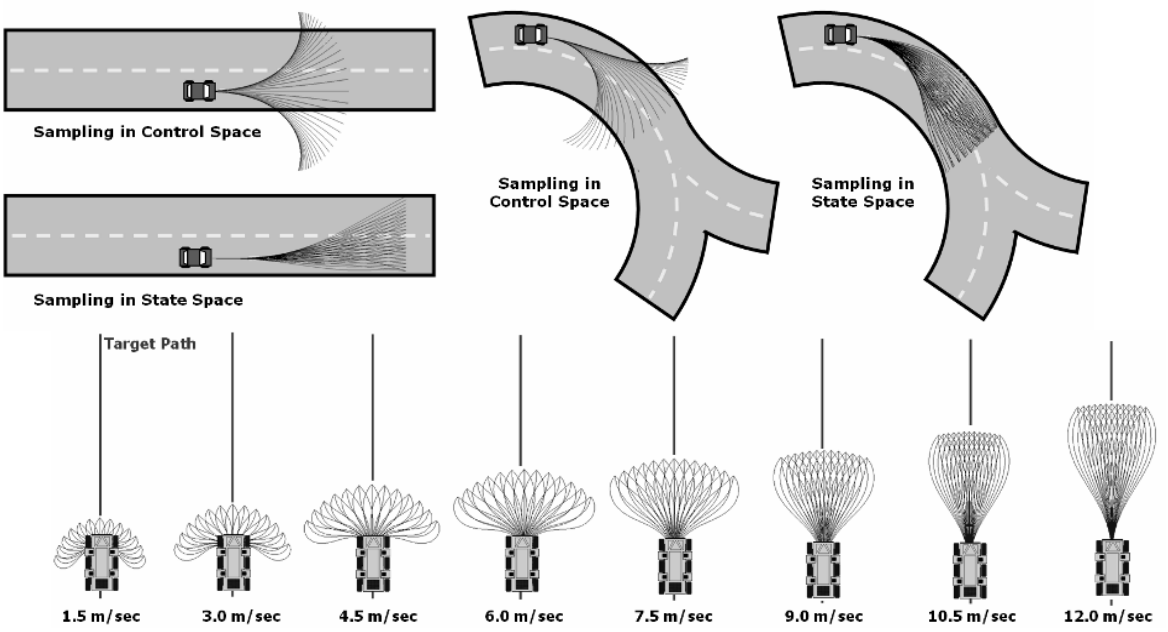

Fig.9. Motion Planning in Constrained Environments. (Top) Uniform sampling in control space for an automobile does not lead to alternatives that remain on a road. Trajectory generation solves this problem readily. (Bottom) Search points adjust in distance and angular span with initial velocity.

4.6 Off-Road Hierarchical Control

Motion planners used in off-road environments often follow a low fidelity global plan using a higher fidelity local planner which enforces dynamic feasibility constraints, avoids obstacles, and follows the global guidance. In the case where the global planner computes a navigation function, a cost field is available everywhere to the local planner whose gradient points in the direction of the optimal path to the goal. We have used the trajectory generator to sample uniformly in the workspace of the vehicle while terminating each candidate motion at the heading desired by global planning as illustrated in Figure 10.

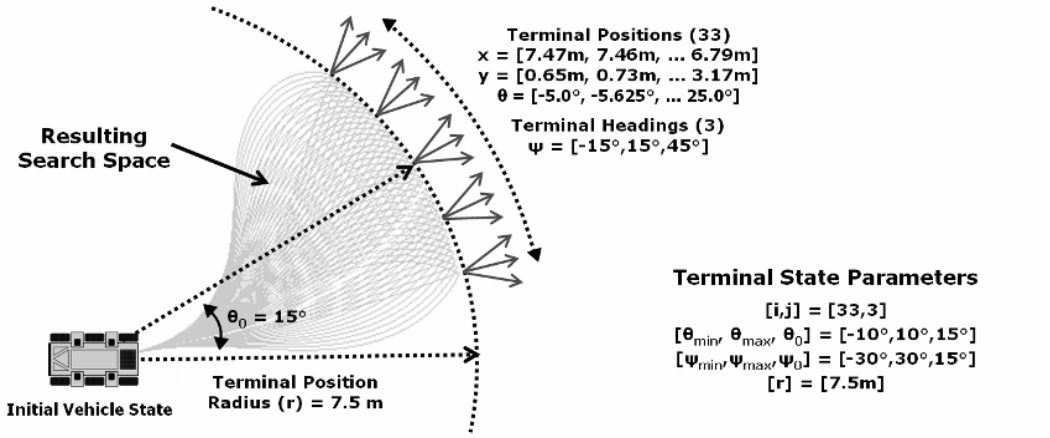

Fig.10. Motion Planning in Cluttered Offroad Environments. In this case, the search points at the ends of the trajectories derive their headings from the global plan.

4.7 Off-Road Global Search Space Design

The capacity to generate feasible motions to arbitrary reachable states makes it possible to construct search spaces for global planners which exhibit useful symmetries, encode only feasible motions, and are continuous where primitive

motions join. The application of search algorithms to such a search space can produce complicated maneuvers like n-point turns automatically. Such a search space is illustrated in Figure 10.

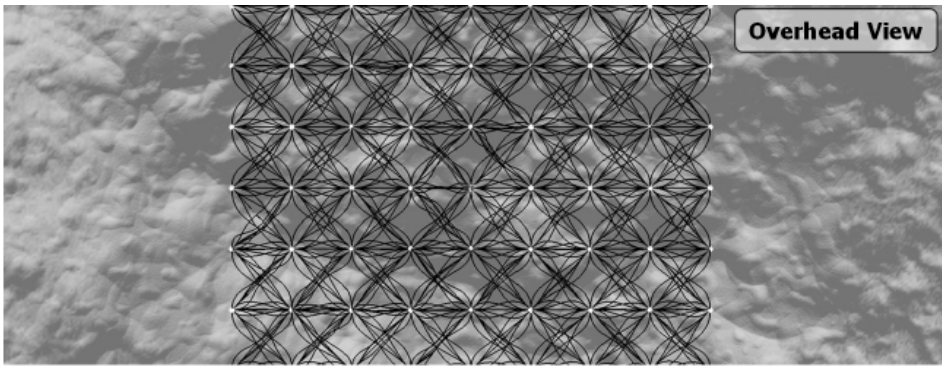

Fig.10. Feasible State Lattice for Global Planning. A regular lattice of states is laid out on terrain and the trajectory generator is used to connect them locally with feasible motions.

5. CONCLUSION

Sampling and searching in the space of controls or actions is a well-established technique in motion planning for ensuring feasible local motion plans. It has been used almost universally in off road ground robotics for some time. The rationale for the use of this technique is that the resulting motions are at least dynamically feasible even if they do not quite take the robot where we would like it to go. A more compelling rationale is that better techniques – ones that produced feasible motions to precisely designated target states - did not exist. We have presented such a technique in this paper.

A capacity to efficiently produce a feasible motion to any reachable point in state space is enabling in several ways. It leads to robots that can interact with their environments in more purposeful and intelligent ways because getting to the precise somewhere where the job needs to be done becomes possible. It leads to robots whose precision understanding of their own mobility enables precision high speed maneuvering in challenging circumstances including those involving complicated terrain interactions, narrow corridors imposed by the rules of the road, and dense obstacle fields. Over the last several years, our group has redefined our approaches to many motion control and planning algorithms in order to maximally exploit this newly available capacity.

REFERENCES

1. L.E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79:497-516, 1957.
2. Y. Kanayama and N. Miyake. Trajectory Generation for Mobile Robots. In *Proc. of the Int. Symp. on Robotics Research*, pages 16-23, Gouvieux, France, 1985.
3. J.A. Reeds and L.A. Shepp. Optimal Paths for a Car that Goes Both Forward and Backward. *Pacific Journal of Mathematics*, 145(2):367-393, 1990.
4. J.P. Laumond. Nonholonomic Motion Planning via Optimal Control. In the *Proc. of the Workshop on Algorithmic Foundations of Robotics*, pages 227-238, San Francisco, California, 1995.
5. J. Reuter. Mobile Robot Trajectories with Continuously Differentiable Curvature: An Optimal Control Approach. in the *Proc. of the IEEE/RSI Conference on Intelligent Robots and Systems*, Victoria, British Columbia, October 1998.
6. M. Cherif. Motion Planning for All-Terrain Vehicles: A Physical Modeling Approach for Coping with Dynamic and Contact Interaction Constraints. In the *IEEE Transaction on Robotics and Automation*, vol. 15, no. 2, pages 202-218, April 1999.
7. Howard, T., Kelly, A., "Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots", to appear *The International Journal of Robotics Research*, Vol. 26, No. 2, February 2007, pp. 141-166.
8. Kelly, A., Nagy, B. "Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control", *The International Journal of Robotics Research*. 2003; 22: 583-601.