AD-A282 787

(1)

A Feedforward Control Approach to the Local Navigation Problem for Autonomous Vehicles

Alonzo Kelly

CMU-RI-TR-94-17

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

May 2, 1994

DTIC
S ELECTE D
AUG 01 1994
G

©1994 Carnegie Mellon University

DTIC QUALITY INSPECTED 8

This research was sponsored by ARPA under contracts "Perception for Outdoor Navigation" (contract number DACA76-89-C-0014, monitored by the US Army Topographic Engineering Center) and "Unmanned Ground Vehicle System" (contract number DAAE07-90-C-RO59, monitored by TACOM).

The views and conclusions expressed in this document are those of the author and should not be interpreted as representing the official policies, either express or implied, of the US government.

[illegible] public release [illegible]

94-24159

7/18

94 7 29 098

Abstract

This report describes the state space model which forms the core technology of an integrated autonomous navigation system incorporating perception, control, and position estimation called RANGER. The high speed local navigation problem is formulated as an optimal control problem in state space. This report concentrates on the trajectory tracking, obstacle avoidance and sensor stabilization, aspects of the system. These algorithms form the basis of RANGER's Controller object.

Accesion For
NTIS CRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification ....
By ....
Distribution /
Availability Codes

Dist	Avail and / or Special
A-1	

List of Figures

Figure 1 Basic Pure Pursuit 11
Figure 2 Adaptive Pure Pursuit 12
Figure 3 Feedforward Pure Pursuit 13
Figure 4 Performance of Feedforward Pure Pursuit 14
Figure 5 Feedforward Simulator 17
Figure 6 Reference Points 18
Figure 7 Propulsion Plant Model 20
Figure 8 Propulsion Servo 20
Figure 9 Steering Servo 21
Figure 10 Heaa Servo 22
Figure 11 Wheel Collision 24
Figure 12 Body Collision 24
Figure 13 Static Stability 25
Figure 14 Hazard Signals 25
Figure 15 Merit Servo 28
Figure 16 Performance of Speed Planning 29
Figure 17 Linear System Block Diagram 30
Figure 18 State Space Model 31
Figure 19 Control Laws 32

Table of Contents

1 Introduction 1

1.1 High Speed Autonomy 2

1.2 Commentary 2

1.3 Acknowledgments 3

1.4 Differences from Important Historical Systems 4

1.4.1 Hughes ALV 4

1.4.2 JPL's Robby 4

1.4.3 JPL's Rocky Series 5

1.4.4 CMU's FastNav 5

1.4.5 CMU UGV Systems 6

1.4.6 ALVINN 6

1.4.7 VITA 6

1.5 Requirements 7

1.5.1 Rough Terrain 7

1.5.2 Continuous, High Speed 7

1.5.3 Ackerman Steer Vehicle 7

1.5.4 Robustness 7

2 Analytical Basis of the Concept 8

2.1 Guaranteed Safety 8

2.2 Adaptive Regard 8

2.3 Fidelity Adaptive Planning 8

2.4 Feedforward Path Generation 9

2.5 Feedforward State Space Controller 9

2.6 Continuity Assumption 9

2.7 Terrain Smoothness Assumption 9

2.8 Unknown Hazard Assumption 9

3 Hierarchical Planning and Arbitration 10

3.1 Strategic Goal 10

3.2 Tactical Goal 10

3.3 Arbitration 10

4 Path Tracking 11

4.1 Adaptive Pure Pursuit 11

4.2 Feedforward Pure Pursuit 13

5 Adaptive Regard 15

6 Real Time Latency Modelling 16

6.1 Localization of Image Pixels 16

6.2 Localization of the Vehicle 16

6.3 Localization of the Feedforward Vehicle 16

7 Feedforward Simulation 17

7.1 State Space Terrain Following Model 17

7.2 Reference Points 18

7.3 Suspension Model 18

7.4 Propulsion Model 20

7.5 Steering Model 21

7.6 Dead Reckoning 22

7.7 Sensor Head Model 22

8 Tactical Planning 23

8.1 Temporal State Interpolation 23

8.2 Hazard Identification 23

8.2.1 Unknown Terrain 23

8.2.2 Terrain-Wheel Collision 23

8.2.3 Terrain-Body Collision 24

8.2.4 Static Stability 24

8.3 Hazard Representation 25

8.4 Hazard Arbitration 26

9 Sensor Stabilization and Pointing 27

9.1 Azimuth Controllers 27

9.2 Elevation Controllers 27

10 Speed Planning 28

11 High Speed Autonomy as an Optimal Control Problem 30
11.1 State Space Controller 30
11.2 Hazard Model 31
11.3 Feedforward Optimal Controller 32
12 Discrete Time Nonlinear Model 34
12.1 State Vector 34
12.2 Command Vector 34
12.3 System Model 34
12.4 Forcing Function and Ackerman Kinematics 35
12.5 Terrain Contact Constraint 35
12.6 Hazard Vector 35
12.7 Implementation in Special Purpose Hardware 36
13 Bibliography 37

1. Introduction

RANGER is an acronym for Real-time Autonomous Navigator with a Geometric Engine. This report describes the "geometric engine" part of RANGER - the state space model, and the associated control algorithms.

A new approach to the high speed autonomy problem is presented which is based fundamentally on the state space representation of a multi-input multi-output linear system and the problem analysis presented in [28]. The system closes the overall perceive-think-act loop for a robot vehicle at relatively high update bandwidth and incorporates both somatic and environmental feedback.

A high fidelity feedforward actuator dynamics and terrain following model is introduced which formulates the local navigation problem as an optimal control problem. Feedforward solves the historically significant clothoid generation problem trivially. C space combinatoric explosion is solved by planning in actuation space. The system feeds steering commands to the state space model, and it uses the terrain map to generate the near clothoid response of the vehicle naturally and directly with none of the algorithmic sensitivities of classical solutions. Obstacle avoidance, path planning, path generation, and path following algorithms are introduced which are based on the state space model. These algorithms are stable and reliable at 10 mph on rough terrain and promise to maintain stability at 20 mph and above.

The system is concerned with the high level coordinated control problem. It is not concerned with the specific control of actuators themselves. Further, while it can accept a specification of a strategic goal, such as a path to follow, or a direction to prefer, it cannot generate its own strategic goals. Therefore, it solves the ***local*** planning problem for autonomous vehicles. The local planning problem includes obstacle avoidance, and goal seeking.

The system can be classified as an ***intelligent, predictive controller*** for autonomous vehicles. It is intelligent because it uses range images that are generated from either a laser rangefinder or a stereo triangulation rangefinder in order to perceive the immediate environment of the vehicle. It is predictive because it reasons based on its knowledge of its own capability to react to hazards in real time as well as the transfer function of the vehicle which relates motion commands to the response to those commands. This transfer function is concerned with the geometry, or ***kinetics***, of vehicle motion. There is no explicit representation of force in the system, so it is basically a geometric one.

The system is a state space controller because it explicitly forms an expression of the vehicle dynamic state vector in order to predict the hazard signals upon which decisions are based. The process which converts terrain shape and command inputs into the associated response of the vehicle is a ***constrained multidimensional differential equation***.

Although the word 'reasons' is used above, the system takes a traditional control systems view rather than an artificial intelligence view of the autonomous navigation problem. The path planning problem is regarded as trivial due to the nonholonomic constraint of Ackerman steering. The reason for this is that the region in space that the vehicle can physically reach which is also within the usable field of view of a sensor decreases very rapidly in size as the speed increases.

As of this writing, the system has achieved 15 Km excursions, average speeds of 15 Km/hr and intermittent speeds of 20 Km/hr and all of these achievements are unprecedented. It owes it success to extensive analysis of the problem it is intended to solve.

1.1 High Speed Autonomy

One of the unavoidable realities of safe, high speed autonomy, is that any vehicle requires significant time and space to react to external events. With vehicle and computational latencies largely fixed, the only practical avenue available is to actively search for hazards at increasing distance from the vehicle. This implies increased demands on sensor resolution and overall system accuracy, and increased sensitivity of planning decisions to small model and sensory errors because the system cannot wait until geometry is more favorable in its assessment of hazards. A real-time autonomous navigator actively manages its field of regard instead of its reaction time because the latter cannot be feasibly altered enough to make a difference.

Unavoidable system latencies and limited maneuverability take on a new level of significance. A high speed vehicle is committed to travelling a certain trajectory for a significant distance before any decisions made in the planner can be enacted in hardware. Further, dynamics and actuator response limit the ability of the vehicle to significantly alter its trajectory after the actuator commands are received. These hard physical limits on braking and steering imply that there is little point wasting computational bandwidth assessing hazards in regions where the vehicle is either already committed to go or where it physically cannot go. The size of the region that the system truly has an option of traversing rapidly decreases in size as speeds increase. This of course implies a reduced ability to avoid obstacles, but that is a fundamental physical cost of higher speeds.

Planning algorithms which directly incorporate knowledge of such dynamics enjoy significant performance improvements. A control systems view of a high speed vehicle leads to complete elimination of the historically significant problems of C space combinatoric explosion and nonholonomically constrained path generation. The problem remains one of search but physical dynamics amounts to an overwhelming constraint. Of all possible trajectories in C space, only a relative handful are both dynamically feasible and spatially distinct enough to warrant consideration.

1.2 Commentary

The traditional hierarchical view of the problem arises perhaps from the axioms of structured programming from software engineering and the artificial intelligence view of hierarchical planning. In this view, planners generate command inputs to controllers which do their best to follow them. While these are excellent tools, there is a point at which hierarchy becomes counterproductive when it is mapped onto contemporary computers and contemporary vehicles. In a real time context, response time and throughput requirements are absolute metrics of success and tightly coupled embedded systems are the traditional practical solution to such problems.

A basic assumption of the hierarchical view is that controllers are empowered to execute commands given to them. For the problem solved here, this is not the case. Indeed, there is very little choice available to a vehicle controller about what the vehicle will do over the short term, and any planner needs to incorporate this knowledge into its model of the vehicle-environment interaction.

When the point is reached when planners must incorporate explicit high fidelity feedforward models of vehicle dynamics and system latencies, the traditional hierarchy has, in fact, been broken because the planner has become a feedforward controller. The question becomes not ***what will I tell*** the vehicle to ***do***, but rather, ***how will it respond*** to this ***request***. A HMMWV is a massive system and its actuators can only generate propulsive and steering forces which are a very small fraction of the total inertial force on the vehicle. Hence, the vehicle goes where it wills and a hierarchical planner-controller approach to the problem of controlling a HMMWV at high speed is an indirect attempt to defy physics.

The only model of a high speed conventional vehicle of practical validity in autonomy is a coupled, constrained, nonlinear, multidimensional differential equation. Idealized kinematic models of vehicle steering are a major cause of the brittleness of kinematic arc based planners which has not been generally recognized.

1.3 Acknowledgments

Barry Brumitt and the author investigated the impact of steering delays on the planning and path generation problems respectively prior to this work. Based on this work, and the precedent of FASTNAV, the idea of generalizing steering dynamics feedforward to a complete state space model ultimately emerged. Barry did initial work on the extent of the dynamically feasible subset of C space. The C space planner constructed by Barry Brumitt and Tony Stentz provided the initial impetus for the feedforward simulator.

R. Coulter was the first to suggest that an optimal control approach to propulsion was worth investigating. R educated me on the need for gravity compensation in propulsion control and collaborated with me in a long joint project to attempt to learn the technological basis of position estimation.

Omead Amidi constructed the CMU vehicle controller which continues to be a central element of all research on the vehicle. Omead first suggested that a tightly coupled real-time implementation of the high level perceive-think-act loop would ultimately be necessary at high speed.

The admissability modules evolved from earlier versions that were designed and constructed by R Coulter and Tony Stentz, and later refined by George Mueller. Earlier still, the Hughes ALV pioneered the field itself with a similar approach.

This system can be considered to be a natural evolution of the ideas of Tony Stentz which formed the basis of the first full geometry off road navigator at CMU since the system implemented earlier by Martial Hebert. This software system was implemented by Barry Brumitt, R Coulter, Al Kelly, Bill Burky and George Mueller under the direction of Tony Stentz.

The only reason that the system discussed here could have been conceived at all was the groundbreaking work of this team in boldly attempting the impossible and along the way identifying the issues which affect high speed performance. Portions of the original code still survive in modified form. The system remains consistent with Tony's original concept of a high speed off road navigator based on high fidelity physical models.

Larry Mathies and Todd Litwin of the Jet Propulsion Labs provided the opportunity to integrate the system with a wide field of view stereo ranging system which provided an excellent forcing function for the software engineering of the code and opportunities to engineer the system to accept a generalized suite of environmental sensor inputs.

Larry educated me on aspects of the physics of remote sensing and his stereo algorithm. Todd collaborated with me on the software engineering of the code. Indirectly, Don Gennery told me the relationship between least squares suspension models and the minimum potential energy principle. Tam Nguyen explained the JPL propulsion controller to me.

1.4 Differences from Important Historical Systems

RANGER differs from all historical precedents known to the author in two ways. First, the system explicitly evaluates vehicle safety in terms of timing, speed, resolution, and accuracy in real-time and employs many adaptive mechanisms which optimize performance. In particular, explicit computation of stopping distance and required minimum throughput is performed and these considerations and others form the basis of an elegant adaptive perception mechanism which makes unprecedented vehicle speeds possible. Second, the system uses relatively high fidelity feedforward models of system dynamics and terrain following and is configured as a tightly coupled controller.

This section summarizes the capabilities of some of the most significant historical off road systems. The list is not exhaustive, nor is it a complete discussion of each system provided. Unfortunately, little has been published to date on some systems, and available publications do not address some of the issues investigated here.

1.4.1 Hughes ALV

The Hughes Autonomous Land Vehicle [9] was one of the first off-road mobile robots ever developed. The system generalized the notion of an obstacle to include any vehicle configuration that is unsafe. This allowed navigation over terrain of any surface shape. Based on the ERIM rangefinder and a highly terrainable vehicle, it achieved speeds up to 3.5 km/hr, over distances exceeding 500 meters, and running times averaging 20 minutes.

The speed of this vehicle was limited by many factors including the complexity of the perception and planning processing, and the speed of communication with off board processing. Robustness was limited by low fidelity modelling of steering constraints, poor terrainability and maneuverability of the vehicle on wet muddy soil, inability to detect small, but dangerous obstacles (steel fence spikes), constrained excursion due to line of sight radio communications, local planning minima and bugs in the high level planner.

The system discussed here differs from the ALV in its real-time minimalist approach to the problem. The adaptive perception algorithms are considerably faster than the Connection Machine implementation of the ALV algorithms. It also discards the costly energy minimization interpolation scheme of the ALV in favor of a temporal interpolation of the hazard vector signal. The system also differs in its high speed design incorporating a differential equation state space model.

1.4.2 JPL's Robby

The JPL rover [46] was the first system to drive autonomously across outdoor terrain using dense stereo data, rather than sonar or laser rangefinder data. This system was demonstrated in a 100 meter cross-country demo in Sept. 1990. It has achieved an average speed of 100 meters in 4 hours (7.0 mm/sec) in a start/stop mode. The total elapsed time in this case is not very meaningful, since it includes downtime while the system was being debugged and numerous stops to evaluate performance[1]. The vehicle speed was mechanically limited to 4 cm/sec and off board communication throughput was limited. Cycle times were on the order of 30 secs for planning and 10 secs for perception. Passive stereo was used in the perception system. Later runs were able to achieve speeds of 4 cm/sec.

---

[1] Larry Mathies, personal communication.

The system discussed here differs from Robby in that it attempts both wide excursion and high speed. It is difficult to assess the degree to which computer evolution alone has made this possible.

1.4.3 JPL's Rocky Series

The Rocky series of Microrovers [43] are small scale prototypes intended as testbeds for simple behavior based control systems. They are intended to produce a flight rover for the MESUR pathfinder mission of 1996. These robots are tested in the context of operations in the vicinity of a planetary lander, so large traverse distance or high speed are not being pursued.

The system discussed here differs from the Rocky series in its deliberative approach to autonomy and its attempt to achieve wide excursions at higher speeds. It borrows the minimalist configuration principle in an attempt to optimize real time performance.

1.4.4 CMU's FastNav

The FastNav project concentrated on achieving high vehicle speeds and robust obstacle detection on fairly benign terrain. The Cyclone single scanline laser rangefinder was used as the basis of the perception subsystem. It was used to check a global path for discrete obstacles, such as trees, rocks, and ditches, and stop accordingly. Obstacle detection was based on detecting deviations from a flat world. Path tracking used a position estimation system based on inertial navigation, dead reckoning, and the GPS satellite navigation system.

FastNav [41] made use of simple tire-soil interaction and vehicle dynamics models to navigate on locally-flat terrain with sparse obstacles. Path tracking was demonstrated at 30 km/hr and obstacle detection was demonstrated at 3 m/s.

Some of the reliability problems that were demonstrated include high sweep rates of the Cyclone scanner due to vehicle pitching, and poor accuracy of the inertial navigation system. Early problems with instability of the tracking algorithm were fixed by incorporating a first order model of the steering dynamics into a feedforward compensator. The longest autonomous run was 1 kilometer[2].

RANGER borrows heavily from the FASTNAV project in three ways. First, steering dynamics feedforward is generalized to a complete state space rough terrain model. Second, the mere fact that such speeds were possible pointed directly to a problem with the way in which nonadaptive image processing was a major cause of poor real-time performance. The line scanner has been an important existence proof that the bare throughput requirement at these speeds was far less severe than it appears to be. Third, the pure pursuit path tracker is an adaptation of the FASTNAV tracker for rough terrain.

RANGER also differs from FASTNAV in that it is a cross country system, not a dirt road system. Full 3D models are employed and the considerable difficulties of rough terrain drive many aspects of the design.

---

[2] Much of this is personal communication with Sanjiv Singh and Jay West. Existing documentation does not cover the issues mentioned completely.

1.4.5 CMU UGV Systems

From September 1990 to the present, CMU researchers have developed two preliminary versions of autonomous cross country navigators. Both navigators are based on the Highly Mobile Multi-Wheeled Vehicle (HMMWV). The performance of the both prototypes has been promising when compared with that of previous systems. Both systems are based on the ERIM laser rangefinder.

The first system [7] is similar in concept to the ALV. It generates a complete geometric description of the terrain in cartesian coordinates by transforming successive range images. It evaluates candidate vehicle trajectories by simulating the vehicle state forward in time in order to predict collisions.

This system was the first to model dynamic constraints on vehicle state in path planning. Path generation was based on an alternative to the spanning arcs approach of the ALV for path generation which is theoretically superior in cluttered environments.

Reliability problems for this system are associated with the poor fidelity of elevation maps, cycle time overruns, and path generation and planning algorithm failures. Nevertheless, this system has achieved its purpose of advancing the state of the art and serving as an experimental testbed.

The most impressive achievement consists of a test run of five kilometers comprised of some thirty orbits of the same circular path on substantially flat terrain with a total of less than ten obstacle avoidance maneuvers. Speed has never exceeded 3 meters per second, and only half of this speed has been achieved while avoiding obstacles.

RANGER borrows heavily from this system because it is basically the next prototype in this series. It differs from its predecessor in its adaptive perception algorithms, the efficiency of its admissability module, its universal 3D models, and it favors a controls feedforward approach over the cluttered environment search based planner.

More recently, a second group [18] have developed a system called Ganesha which processes range images to recover regions of constant terrain gradient. This system has achieved runs over distances of 840 meters, at speeds of 2-3 m/s while avoiding natural obstacles, and it has probably exceeded these results as of this writing.

RANGER differs from Ganesha in that it can be considered to be more of a deliberative system on the spectrum of such systems because it remembers both vehicle state and environmental state from cycle to cycle to a higher level of detail than Ganesha does. RANGER is also theoretically superior to Ganesha on rough terrain at a cost of the extra processing and slower reaction time required to navigate in a fully 3D world model.

1.4.6 ALVINN

ALVINN is a road following system based on neural networks [39]. Although it solves a different problem, ALVINN provides a second precedent for the use of feedforward for the purposes of imparting stability at high speed.

1.4.7 VITA

The Daimler-Benz group have developed impressive road followers based on a control systems approach to road following. Although a different problem is solved by these systems, the control approach has been an important precedent.

1.5 Requirements

The design of the system is based on these top level requirements. They are the highest level design drivers that influence the design to a large extent. Certain of the components would be unjustified for systems with other requirements such as, for example, slow, or start stop vehicles or omnidirectional vehicles.

1.5.1 Rough Terrain

Rugged, sparsely vegetated, mostly dry terrain is the target environment within which the vehicle must navigate safely. A flat world assumption along with discrete obstacle detection is not considered valid in the target terrain.

1.5.2 Continuous, High Speed

Speeds which approach the speeds at which humans can drive the same terrain (say, 20 mph) are targetted.

1.5.3 Ackerman Steer Vehicle

While the Ackerman steering configuration is not ideal for planning purposes, most large vehicles intended for human use employ this configuration, so the restriction to this class of vehicles is not as limiting as it may appear. In any case, the maneuverability of such vehicles is dramatically different from other alternatives, particularly at speed, and the report will show that efficiency demands that a special case solution for this class of vehicles be adopted. Nevertheless momentum and the nonholonomic constraint limits maneuverability of all high speed vehicles, so many of the results here apply generally.

1.5.4 Robustness

The system is intended for relatively long traverses of suitable terrain. While robustness is difficult to quantify, it is the intention that reliability be sufficient for routine autonomous excursions of tens of kilometers over suitably chosen terrain.

2. Analytical Basis of the Concept

RANGER is an attempt to design an optimal system from a systems perspective. A consistent requirements analysis was conducted which identifies the interrelationships of requirements and which attempts to optimize the system performance as a whole. Consistent resolution and accuracy goals are imposed on the design so that all subsystems are equally well and equally poorly able to do their respective jobs. In many cases, ***trade-offs are managed in real time in the code itself.***

This approach arose for historical reasons. Earlier versions of the system had no basis for allocating resources to subproblems so all were optimized at their own level. This led to "near perfect" perception and planning algorithms which did an excellent job of predicting collisions with hazards that they were too busy ***thinking*** about to ***do*** anything about.

The system eliminates the lower levels of control which traditionally attempt to follow a path provided by a higher level agent because the problem is too coupled to solve by hierarchy. The system works in actuation space and incorporates feedforward. For this reason, it is effectively a ***controller*** and not a planner.

The system that emerges from these considerations is a unique one. It is a real time control system which operates on environmental and navigation sensory inputs and incorporates a high fidelity model of both the plant and the environment. It is tightly coupled and it merges the traditional hierarchies into one conceptual layer that closes the perceive-think-act loop at high update bandwidth. The system is based upon a large body of analyses presented in [28]. A brief discussion of the most significant aspects of the design is presented below.

2.1 Guaranteed Safety

The system directly implements the policy of guaranteed safety by reasoning in real time about the four dimensions of safety (i.e. timing, speed, resolution, and accuracy) and adapting its perception and planning subsystems to comply directly with the need for safety.

2.2 Adaptive Regard

Adaptive Regard is a mechanism for ensuring that the system minimizes the spatial extent of the region it perceives based on vehicle maneuverability so that speed is maximized without compromising safety. The principle of adaptive regard is to scan for hazards only in the obstacle detection zone. This is the region of space that the vehicle still has an option of traversing. The system implements adaptive regard by adapting its obstacle avoidance behavior to guarantee that an impulse turn can always be executed should an obstacle be detected.

2.3 Fidelity Adaptive Planning

This principle surrounds hazards by a small buffer region in actuation space in order to compensate for any inaccuracies in the computations. It is implemented here as **Gaussian filtering** of the steering vote vector.

2.4 Feedforward Path Generation

The configuration space of the high speed nonholonomic vehicle is degenerate at high speed and the attendant clothoid generation problem is impossible to solve in practical terms. The "path generator" for the system generates, not candidate paths, but candidate command signals ***expressed in actuation space*** (in terms of curvature and speed as a function of time), and uses a high fidelity simulator to solve the differential equations of motion in order to determine the exact response to these candidate commands. This strategy has the following advantages:

- The paths generated ***meet the mobility constraints of the vehicle by construction***, so the difficult and often impossible problem of conversion from C space to A space is completely avoided. Instead, the reverse process of dead reckoning is used in the simulator. This is a kind of feedforward.
- The paths coarsely span the entire region of space that the vehicle can reach, so no real alternatives are missed.

2.5 Feedforward State Space Controller

The system represents hazards as sampled time signals in a kind of multidimensional hazard space. The essential obstacle avoidance mechanism is the ***minimization of an optimal control functional*** in this space. Interpolation is performed in time along the vehicle state vector rather than in the space of the terrain map.

2.6 Continuity Assumption

The assumption that important aspects of vehicle maneuverability and system computational performance will not change significantly from one cycle to the next is called the **continuity assumption**. This assumption is used to resolve some of circular issues related to the coupling between adaptive perception and adaptive regard.

2.7 Terrain Smoothness Assumption

The **terrain smoothness assumption** is required in order to implement a terrain map representation of the environment. It is also used to justify the interpolation of small unknown regions.

2.8 Unknown Hazard Assumption

The **unknown hazard assumption** involves the belief that large unknown regions in a terrain map are unnavigable by default. This key assumption permits the system to operate robustly near large holes or near the edge of a cliff.

3. Hierarchical Planning and Arbitration

The **local minimum problem** is perhaps the most serious threat to robustness of any local obstacle avoidance strategy for autonomous navigation. In order to achieve robust navigation, a sound strategic plan that avoids these minima is a solution that works well in practice. In order to enhance robustness, the system supports the input and continuous monitoring of strategic goals.

3.1 Strategic Goal

The system defines a **strategic goal** which instantaneously may be any one of:

- follow a predefined path
- drive toward a predefined point
- maintain a fixed compass direction
- maintain a fixed curvature

3.2 Tactical Goal

The **tactical goal** is to simultaneously avoid all hazardous conditions including:

- tipover
- body collision (high centering)
- wheel collision
- entering unknown terrain

and extensions to other hazards are simple to perform.

3.3 Arbitration

In situations where a hazard is detected which lies in the way of achieving the strategic goal, the tactical goal will override the strategic one as long as any hazard exists. The output of the steering arbiter may be very discontinuous if a hazard suddenly appears or disappears and because high steering rates constitute a hazard by themselves, the steering arbiter output is smoothed to remove steering discontinuities except when drastic action is called for. That is, the strategic steering output is smoothed whereas the tactical one is not.

The arbitration scheme might be called a ***prioritized sort***. The principles of this technique are:

- Obstacle avoidance always overrides tracking if it chooses.
- Tracking chooses the best path which obstacle avoidance allows it when it is allowed control.

This approach has the side effect that following of an extended feature will emerge naturally, and the system will immediately take the opportunity to reacquire the strategic goal if an opportunity presents itself. While this describes the emergent behavior, the algorithm itself is to:

- sort the votes of the tactical planner using the corresponding strategic votes as the key
- choose the first path in the sorted list which satisfies obstacle avoidance

Effectively, if obstacle avoidance does not care which direction to choose, the strategic planner is given control. If it does care, it will continually veto the strategic votes in order of strategic preference until a safe one is presented to it. If no safe path exists, a stop command is issued.

4. Path Tracking

The only nontrivial aspect of tracking a strategic goal is high fidelity tracking of a convoluted path over rough terrain.

4.1 Adaptive Pure Pursuit

The strategic controller incorporates an adaptive path tracker which is based on the **pure pursuit** algorithm. The pure pursuit algorithm has been around for some time [41]. It is basically a proportional controller formed on the heading error computed from the current heading and the heading derived from the current vehicle position and a goal point on the path. The goal point is computed by finding the point on the path which is a predetermined distance from the current vehicle position. There are many variations possible on the basic algorithm.

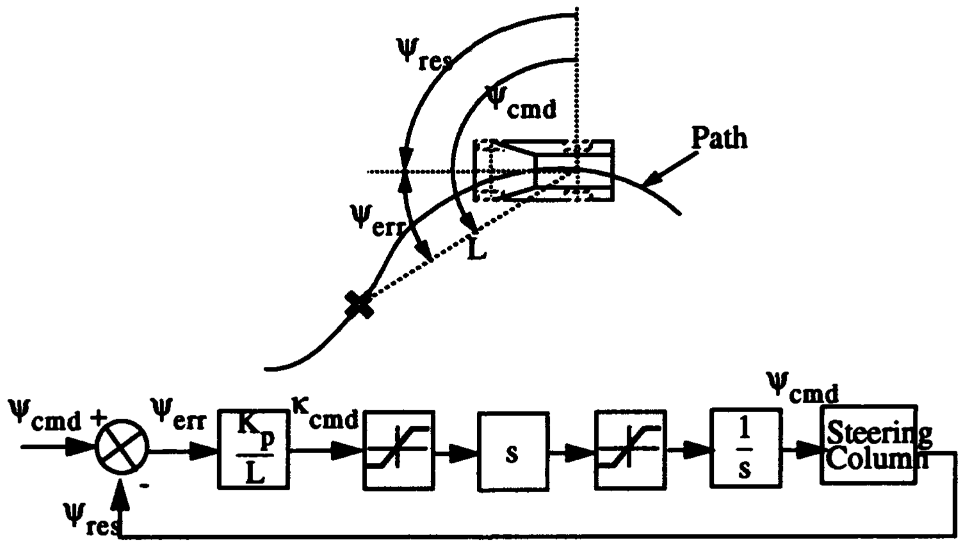


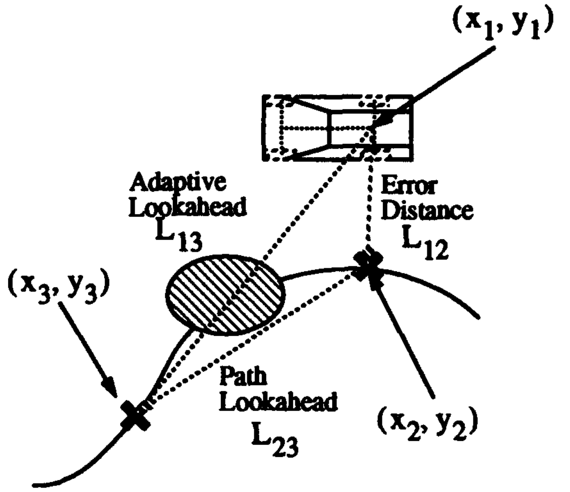
Figure 1 - Basic Pure Pursuit

Heading is measured at the center of the rear axle. The proportional gain is normalized by the lookahead distance L. This can be viewed an adaptive element or, more simply, as a unit conversion from heading error to curvature. Indeed, the ratio ψ_{err}/L is the average curvature required to reacquire the path at the goal point.

A limiter is used which ensures that the curvature is limited at higher speeds to prevent rollover. Another limiter maintains the angular acceleration below a threshold. These measures ensure vehicle safety and directly prevent instability at the cost of an inability to track high curvature paths at high speed. However, vehicle dynamics prevent tracking high curvature paths at high speed anyway.

Large tracking errors or, equivalently, too short a lookahead distance or too high a gain all result in servo instability. This is a well known problem with pure pursuit which can be addressed with feedforward.

A few modifications are introduced to adapt pure pursuit for rough terrain. First, extremely large tracking errors must be acceptable to the servo without causing instability. This is managed by two devices indicated in the following figure.



```
/*
** Adaptive Pure Pursuit
*/
min = HUGE; way_pt = close;
while(L23 < L12 + lookahead)
  {
  x3 = path_x[way-pt];
  y3 = path_y[way_pt];
  if( L13 < min )
    {
    close = way_pt;
    }
  way_pt++;
  }
way_pt = close;
while(L13 < L12 + lookahead)
  {
  x3 = path_x[way-pt];
  y3 = path_y[way_pt];
  goal_pt = way_pt;
  }
```

Figure 2 - Adaptive Pure Pursuit

Instead of using the current vehicle position as the origin of the lookahead vector, the system maintains a running estimate of the point on the path which is closest to the current vehicle position. This is done because it is very expensive to search the entire path each iteration. In doing so, the system is assuming that the vehicle will never have a heading error which exceeds 90 degrees for an extended period of time. This is a **monotone arc length assumption.** This assumption completely eliminates the overwhelming computational cost of simpler implementations of the algorithm.

The lookahead distance is adaptive to the current tracking error - increasing as the error increases as indicated in the accompanying code fragment. The first while loop is responsible for maintaining a running record of the close point, point 2. It searches through an arc length window which adapts to the path tracking error. As the error gets larger, this loop will cause the close point to jump over high curvature kinks in the path as they become less relevant at the resolution of the tracking error.

The second while loop computes the goal point in an identical manner. It basically moves point 3 forward until it falls outside a circle centered at the vehicle whose radius is the sum of the error distance and the nonadaptive lookahead. In this way, when an obstacle avoidance maneuver causes significant path error, the algorithm will search to reacquire the path on the other side of the obstacle instead of causing a return to the front of the obstacle.

Notice that under normal circumstances when the vehicle is tracking the path, the close point is at the vehicle position, the error distance is zero, and the adaptive lookahead is the nonadaptive lookahead. In plain terms, the algorithm gracefully degenerates to classical pure pursuit when obstacle avoidance is not necessary.

4.2 Feedforward Pure Pursuit

An upcoming section will present a high fidelity feedforward simulator which models many aspects of vehicle kinetics. A feedforward option in the tracking algorithm incorporates the output of this simulator into the tracker. The cost of a feedforward simulator must be borne, so an additional feedforward element in the tracker is available for free. The basic idea is as follows. First, at each point in the simulation, evaluate the distance from the vehicle to the goal point. Second, the candidate command which comes closest to the goal point becomes the vote of the strategic controller. Such an algorithm provides excellent path following in three dimensions over rough terrain.

The concept is indicated in the following figure. During a high curvature turn at speed, the feedforward simulator skews the response curves toward the current steering direction.

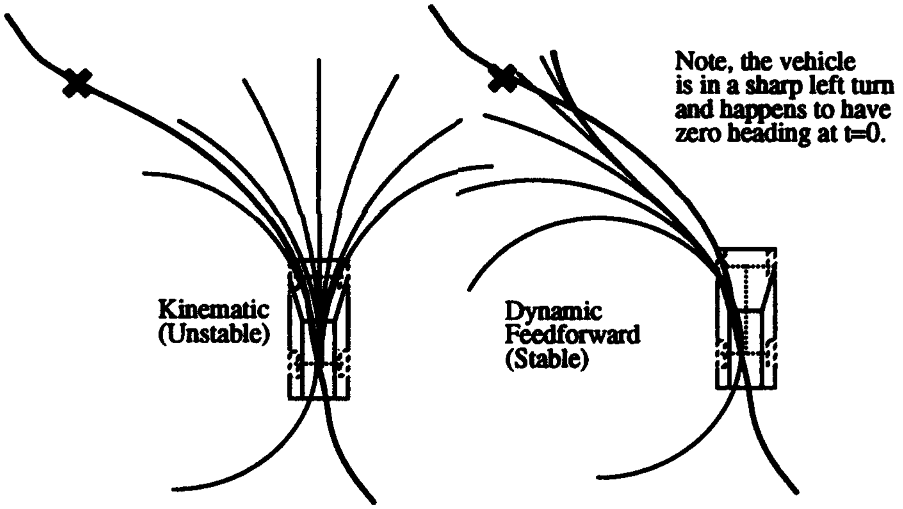


Figure 3 - Feedforward Pure Pursuit

A kinematic tracker would issue a hard right command in the situation depicted above whereas a dynamic one would recognize that such a command would actually ***increase*** the tracking error. A dynamic tracker would issue a zero curvature command and would correctly acquire the goal point as the steering wheel slowly turned toward zero. The following figure illustrates the performance of the feedforward algorithm relative to the kinematic one.

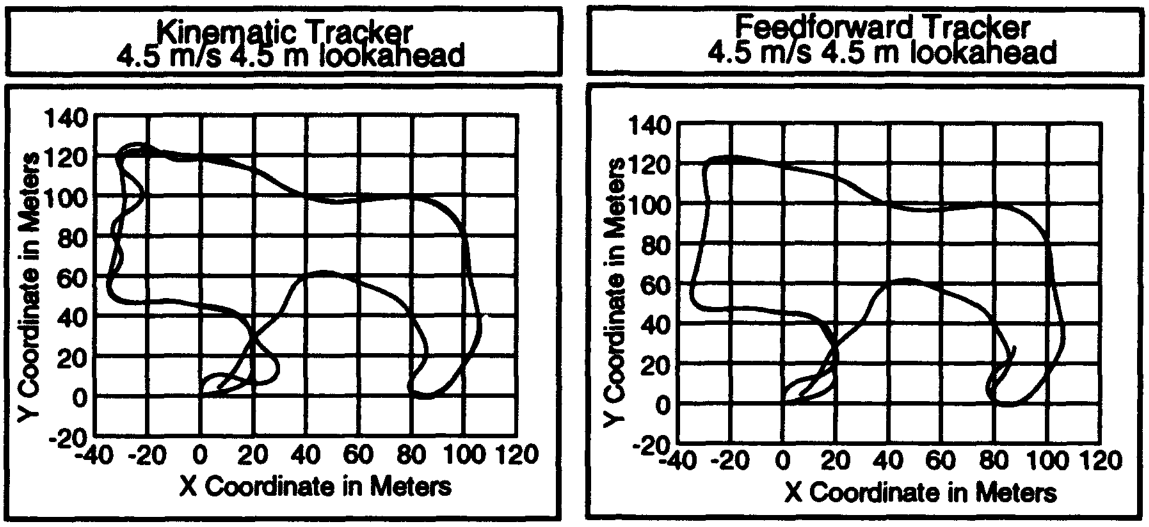


Figure 4 - Performance of Feedforward Pure Pursuit

Each graph shows the target commanded trajectory and the actual trajectory followed under the algorithm used. Both paths are indistinguishable most of the time. Note, however, that the kinematic tracker goes unstable in the top left figure whereas the feedforward one does not.

5. Adaptive Regard

The adaptive regard principle can be expressed as follows: There is no need to process data in regions where:

- the vehicle is already committed to going (dead zone)
- the vehicle cannot go (free zone)
- subsequent images will provide better measurements (beyond detection zone)

because in all cases, ***there is no useful decision that a planner can make***.

In the first case, if an obstacle enters the dead zone, system failure is assured, so they must be detected and avoided before they get this close. In the second case, processing such data is a complete waste of resources. In the third case, processing can be postponed until a future cycle.

If the vehicle cannot avoid obstacles at close quarters, not only is there no need for processing image geometry there, there is also no need for a planner to evaluate safety in its immediate vicinity in the map. The system planner evaluates trajectory safety only within the **planning window**. The algorithm for computing this window based on an impulse turn maneuver is given below.:

```
/*
** Plan Window
*/
Pmax = speed * treact + rhomin;/* adaptive lookahead */
Pmin = Pmax - imaging_density * speed * cycle_time /* adaptive sweep */
```

An impulse turn is a turn from zero curvature to the maximum allowed curvature. It is the maneuver necessary to avoid a large obstacle without slowing or stopping. The imaging density is the average requested density of images on the groundplane. It must be maintained above 1.0 at all times to ensure adequate coverage of the environment. Later, the perception module will map this planning window into image space to allow it to extract the requested data.

6. Real Time Latency Modelling

As a real time system, a reliable measurement of time is required for the system to function correctly. In particular, the system needs to match images with vehicle states which may arrive intermittently and it needs to remember the commands that were issued to all actuators in previous iterations of the main control loop because the system cycles faster than the output latencies. This is accomplished with the following mechanisms.

- all input and output is time stamped
- all input and output is stored in internal FIFO queues
- all sensor latencies are modelled
- all actuator latencies are modelled

It is important to recognize that these FIFO queues do not introduce artificial delay. Their function is to ***model*** delay which already exists in the hardware. Conceptually, all i/o goes through these queues.

6.1 Localization of Image Pixels

The smearing of the environment model which arises from incorrect models of the motion of the sensor during the image digitization process is called the **motion distortion problem.** All incoming vehicle state information is stored in a FIFO queue until it is needed to process images which arrive later. The state queue and the time tags are used to effectively register the ray through every pixel with a unique vehicle pose before its value is converted to world coordinates.

6.2 Localization of the Vehicle

While temporal registration of images and vehicle positions is a resolution matter, the raw delay of the vehicle position estimate is a separate accuracy concern. When the system uses the latest vehicle state as its estimate of position, the error involved in doing so is the velocity times the latency. Clearly a delay of only one second causes 5 meters of localization error in the vehicle position at 5 m/s and pretty well guarantees collisions with obstacles solely because the system does not know how close it is to the obstacle[3]. For this reason, an estimate of position sensor latency is used and fed forward just like any other parameter[4].

6.3 Localization of the Feedforward Vehicle

All output commands are stored in a FIFO queue and are used for the next several iterations in implementing command feedforward. When the system is generating the current command, there may be quite a few previous commands ***still en route to the hardware*** so a proper model must feed these pending commands into the model before the current one is used and make decisions based on the last command to go out, because the system still has control over only that current command.

---

[3] This is an almost intolerable situation, particularly when steering response is added to the picture, and it argues persuasively for tight system coupling. To ignore these issues at 5 m/s it to overestimate obstacle clearance by 5 meters, underestimate the distance required to turn by as much or more, and drive straight into anything in the way. However, the best that can be done is to model the latency if it cannot be reduced. Roughly speaking, a 2 second delay from image to steering actuator makes 10 m/s speed impossible.

[4] This is a practical measure because extant vehicle positioning systems and computing hardware have no consistent time standard. Ideally, the latency could be simply measured from the current time and the time tag of the image.

7. Feedforward Simulation

In order to evaluate safety, the tactical control module computes the response to candidate commands simulated forward in time until the detection zone is reached. The basic justification for this approach is an attempt to meet guaranteed response and localization simultaneously. In order to make sound decisions, the system must know ***now*** where the vehicle will be ***many seconds later*** if a particular command is issued now. It cannot "wait until it gets there" to measure its response because then it will be too late to react.

The solution of this problem is mathematically involved because it involves solution of a coupled set of eight nonlinear differential equations which form the vehicle state space model. The system mechanizes these equations in real time. The equations are coupled because:

- position at step i depends on attitude and speed at step i-1
- attitude at step i depends on steering response, propulsion response, and attitude step i-1
- steering response at step i depends on steering response and steering command at step i-1
- propulsion response at step i depends on attitude, propulsion response and propulsion command at step i-1
- attitude at step i depends on suspension state at step i-1

7.1 State Space Terrain Following Model

By solving the equations in their true coupled form, the system can correctly simulate a vehicle driving along the side of a hill, for example. It correctly simulates the dependence of actual trajectory on the terrain itself, the speed, the actuator response characteristics, and the initial conditions.

The basic simulation loop can be written as follows. At each time step:

- simulate suspension - determine attitude from terrain geometry and position
- simulate propulsion - determine new speed from command, state, and attitude
- simulate steering - determine angular velocity from steering and speed
- simulate position - dead reckon from linear and angular velocity and time step

A mechanization diagram is shown below for these equations:

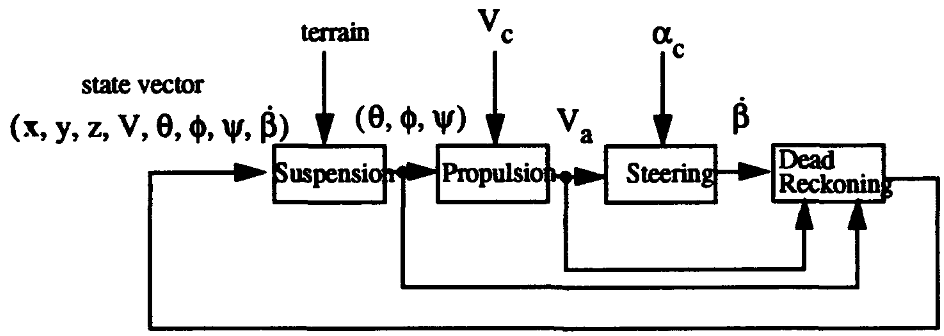


Figure 5 - Feedforward Simulator

In the figure, (x, y, z) represents the position of the vehicle in the navigation frame, (θ, ϕ, ψ) represents its attitude in terms of pitch, roll, and yaw angles respectively, V is the vehicle speed along the body y axis, α is the steer angle of the front wheels, $\dot{\beta}$ is the angular velocity of the body projected onto the body z axis, and the subscripts c and a represent commanded and actual quantities respectively.

7.2 Reference Points

The positions of distinguished points on the body, called **reference points**, are maintained in navigation coordinates throughout the simulation. The kinematics transforms involved in doing this are documented elsewhere [26]. These points are at the same elevation as the bottom of the wheels and are positioned at the wheel contact points and the center of the axles. The processing of the reference points proceeds as follows for each cycle of the simulation:

- convert coordinates from body to nav frame based on the vehicle position and attitude
- replace their z coordinates with those of the terrain at the same (x,y)

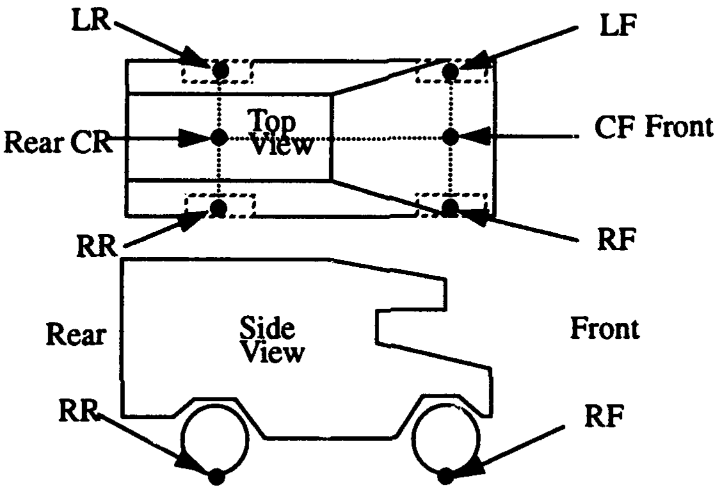


Figure 6 - Reference Points

All aspects of both system state and hazardous conditions can be computed from the reference points and the terrain elevation under them. In this way, coordinate system transformation operations are reduced to an absolute minimum.

7.3 Suspension Model

Using the reference points only, a proper model would:

- compute the elevation "under" each wheel
- compute the deflections of each wheel from the minimum potential energy principle
- compute the positions of three points on the body
- compute the body attitude from these three points

which is a lot of somewhat costly work to do. Therefore, the suspension model is based on:

- a **rigid terrain assumption**
- a **rigid suspension assumption**
- a **locally planar terrain assumption**

The first assumption results from using the terrain map elevations measured by a sensor when the vehicle was not loading the terrain. The second occurs because the positions of the wheels relative to the body are taken as fixed. The third occurs because the elevations under the four wheels are used in forming two vectors even though they do not necessarily lie on the same plane.

The last problem in the process can be expressed as that of recovering an unknown rotation matrix from a few points which are transformed using it. The inverse RPY transform was computed in [26] for this purpose. The displacement transform from body coordinates to world coordinates for a z-x-y Euler angle sequence can be expressed as:

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \begin{bmatrix} (c\psi c\phi - s\psi s\theta s\phi) & -s\psi c\theta & (c\psi s\phi + s\psi s\theta c\phi) \\ (s\psi c\phi + c\psi s\theta s\phi) & c\psi c\theta & (s\psi s\phi - c\psi s\theta c\phi) \\ -c\theta s\phi & s\theta & c\theta c\phi \end{bmatrix} \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix}$$

Yaw and pitch can be determined from any vector known to be aligned with the body y axis:

$$\begin{aligned} \psi &= \mathit{atan}2\,(r_{22}, -r_{12}) \\ \theta &= \mathit{atan}2\,(r_{32}, -r_{12}s\psi + r_{22}c\psi) \end{aligned}$$

Roll can be derived from the world coordinates of a vector known to be aligned with the body x axis.

$$\phi = \mathit{atan}2\,(s\theta\,[-r_{11}s\psi + r_{21}c\psi] - r_{31}c\theta,\ (r_{11}c\psi + r_{21}s\psi))$$

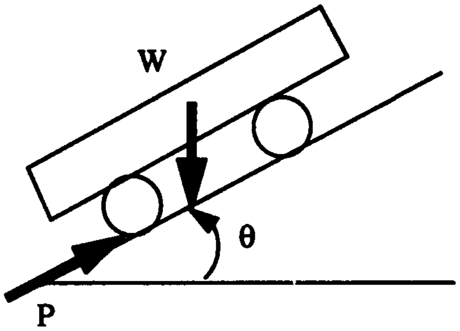
The above equations are an exact inverse kinematic solution. However, the system makes a valid **small pitch assumption** currently in order to reduce the trigonometric computations. The vectors are formed from the reference points. The algorithm is simply:

```
/*
** Simulate Suspension
*/
zleft = (lf[z] + lr[z]) / 2.0;
zrght = (rf[z] + rr[z]) / 2.0;
zfrnt = (rf[z] + lf[z]) / 2.0;
zrear = (rr[z] + lr[z]) / 2.0;
z = zrear + tire_radius;
pitch = atan2(zfrnt - zrear)/wheelbase;
roll = atan2(zleft - zrght)/width;
```

Using a linear deflection model, it can be shown that, since strain energy is squared in strain, the minimum potential energy principle of solid mechanics implies that a least squares solution to the deflections of the four wheels of an independent suspension is also the solid mechanics solution. The simple suspension model seems to work well enough as it is so a more refined model was not implemented. Such a model however, would permit the representation of suspension overextension as a hazard. Also, the deviation of the four wheel contact points from a plane is an equivalent model under a rigid suspension assumption and this would be simpler to implement.

7.4 Propulsion Model

Propulsion is currently modelled as a proportional controller with gravity compensation[5]. Under an assumption that the torque at the wheels can be directly actuated and that the terrain can generate any necessary shear reaction (no slip) Newton's law can be written along the body y axis as follows:



$$\sum \vec{F} = M\vec{a}$$

$$P - W\sin\theta = Ma$$

$$\therefore a_{res} = \frac{P}{M} - g\sin\theta = a_{cmd} - g\sin\theta$$

Figure 7 - Propulsion Plant Model

which gives the plant model. A proportional controller using velocity feedback is configured as in the following block diagram[6]:

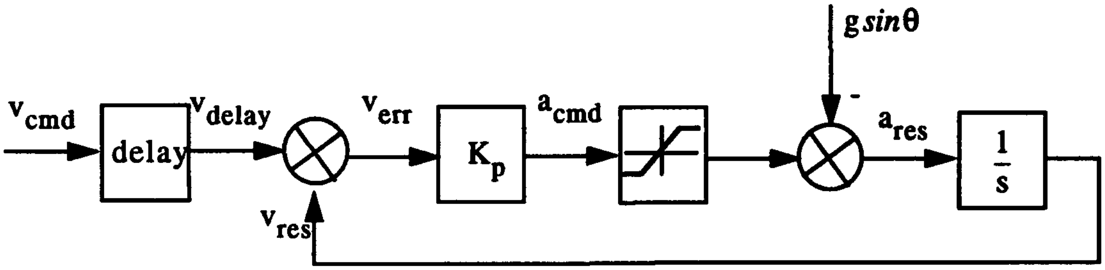


Figure 8 - Propulsion Servo

The inverse of the proportional gain K_p is the time constant. A time constant of 5 seconds corresponds to the generation of 0.1 g command for a 5 m/s speed error. A limiter is added to reflect the fact that the output torque has some limit. One way to compute this limit is to choose a pitch angle corresponding to some grade that is the highest grade that the vehicle can climb. Under this model, the computer algorithm is the finite difference version of the system differential equation, which is simply[7]:

---

[5] This model is similar to the JPL speed controller except that throttle is actuated and not torque.

[6] If, like most of us, the reader has forgotten Laplace Transforms, 1/s is an integral and s is a derivative.

```
/*
** Simulate Propulsion
*/
now = read_clock();
vdelay = queue_lookup(prop_queue,now - prop_delay);
verr = vdelay - vres;
acmd = verr / prop_time_constant;
if (fabs(acmd) > amax) acmd = amax * acmd / fabs(acmd);
ares = acmd - g * sin(pitch);
vres += ares * dt;
```

This model has several advantages over an ideal one. It will correctly cause the simulated vehicle to take several seconds to respond to a step input command and it will even cause the simulated vehicle to back up on a steep grade and generally slow down or speed up depending on the grade.

7.5 Steering Model

The steering model is involved because it must account for the nonlinear relationship between curvature and steer angle. This model is also a primary area of coupling between speed and attitude rate, so it is one of the most important elements of the model. The bicycle model of the steering kinematics is given in [26]. Typically, the steering wheel itself is actuated, so the loop must be closed back at the steer angle. This assumes only that the steering wheel to steer angle transform is linear. Either position or speed could be the controlled variable, but a proportional position loop is assumed[8]. Both the position and the speed are limited[9]. The complete dynamics can be expressed with the following block diagram:

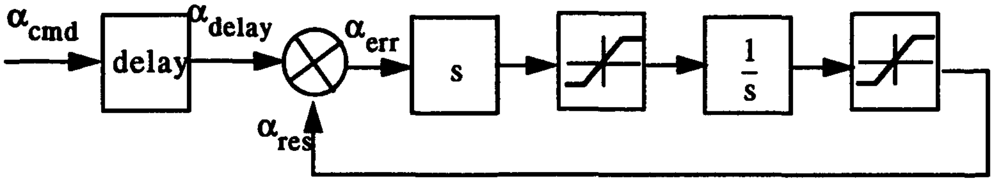


Figure 9 - Steering Servo

---

[7] If we distinguish control algorithms from complete controllers, many actuator control algorithms can be implemented in just a few lines of code. The above situation is the rule, not the exception. Code is provided in order to illustrate that however sophisticated dynamic models may sound, implementing them is easy.

[8] At this moment, a position loop is used because a speed loop requires speed feedback which is not available on some vehicles. It could be generated internally by differentiating the position feedback, but the speed limit is considered to be the overriding element of the delay anyway.

[9] There may also be a trapezoidal command velocity profile. This amounts to an acceleration limit on the command.

Again the coding is straightforward:

```
/*
** Simulate Steering Dynamics
*/
now = read_clock();
alpdelay = queue_lookup(steer_queue,now - steer_delay);
alperr = alpdelay - alpres;
if (fabs(alperr/dt) > alpdotmax)
  alperr = alpdotmax * dt * alperr / fabs(alperr);
ares += alperr;
```

These few lines of code make an enormous difference in the high speed stability of the system.

7.6 Dead Reckoning

The 3D dead reckoning equations also form the system model in the Kalman filter which is documented elsewhere [30]. The attitude rate about the body y axis is available from the instantaneous velocity and the instantaneous steer angle (or curvature) as follows:

$$\frac{d\beta(t)}{dt} = \frac{1}{L}\tan[\alpha(t)]\frac{ds}{dt} = \kappa(t)\frac{ds}{dt}$$

7.7 Sensor Head Model

A controllable pan/tilt sensor head is modelled by two differential equations which are similar to those used for the other actuators. It is likely that the system would not have access to speed feedback, so again, a position loop is used to model the dynamics. A loop which is identical to the steering model can be used to reflect the limited angles of excursion and the limited maximum speed. For the tilt (or pitch) axis, the servo would be configured as follows:

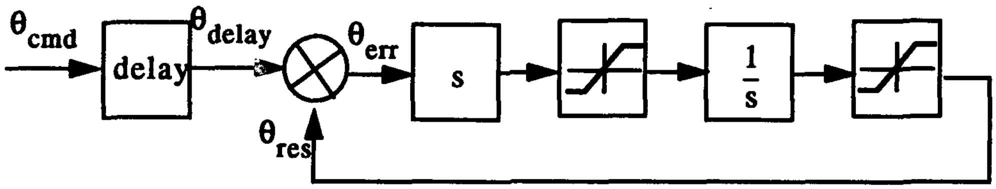


Figure 10 - Head Servo

8. Tactical Planning

The system adopts the basic assumption that all hazardous conditions can be computed from the reference points. Hazards are represented as sampled time signals which are parameterized by the commands to which they correspond. At any point in forward projected time, the system is designed to be robust to the absence of information because the **sampling** and the **occlusion problems** are intrinsic and inevitable.

8.1 Temporal State Interpolation

Field testing of the system on typical rough terrain indicates that, ***if occlusions are distinguished from poorly sampled regions*** of the environment, and the need for sophisticated terrain map interpolation is questionable. Specifically, the degree of undersampling depends on the local incidence angle of pixels to the terrain and the need for interpolation directly implies that the terrain is mostly flat anyway. The reverse argument is more compelling. If a small vertical step in the terrain exists, then it must span several pixels due to its near normal incidence. Therefore, it cannot be undersampled.

These observations imply that it is unnecessary to interpolate an entire terrain map before it is used for planning computations. It can be done ***on the fly***, and more importantly, it can be done ***on the vehicle state vector.*** The state vector alone supplies the information necessary to propagate the system state forward in time, and the terrain contact constraint can be modelled as a "disturbance" which modifies the vehicle attitude whenever the terrain under the wheels is known (which, in practice, is almost always).

The interpolation algorithm used in the system is an interpolation of both the state vector and the hazard signals in time, rather than an interpolation of the terrain map in space.

8.2 Hazard Identification

Three principal forms of hazards are incorporated:

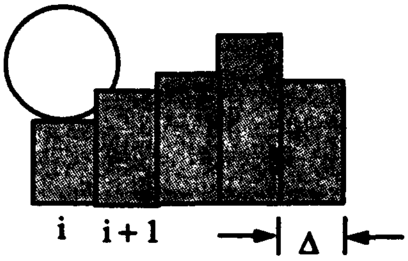
- unknown terrain
- collision of the terrain with the body
- static instability

8.2.1 Unknown Terrain

Unknown map cells may arise from undersampling or terrain self occlusion. An overall information density merit is computed for each candidate command which is based on the number of wheel elevations which are known at each time step.

8.2.2 Terrain-Wheel Collision

Wheel collision is computed as the slope of the elevations under each wheel as the body is propagated forward.



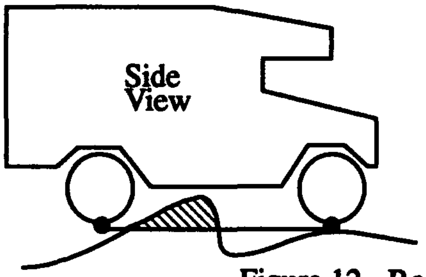
$$\text{merit}(i) = \frac{\frac{z_i - z_{i-1}}{\Delta}}{\left(\frac{\partial z}{\partial s}\right)_{max}}$$

Figure 11 - Wheel Collision

In practice, this signal can be noisy if the sensor cannot supply the angular resolution and relative range accuracy necessary. At times, it is not used at all.

8.2.3 Terrain-Body Collision

Body collision is computed as the lost volume under the body normalized by the area under the body, so it is an expression of the average height under the body. The result is normalized by the number of map cells which were actually known.

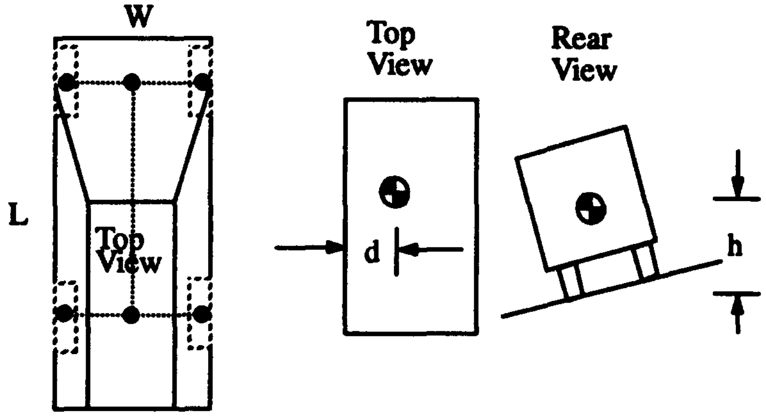


$$\text{merit}(i) = \frac{\frac{1}{n}\sum_i z_i}{h_{max}}$$

Figure 12 - Body Collision

8.2.4 Static Stability

Static stability measures the proximity of the vehicle to tipover. The **static stability margin** can be expressed as the remaining angle between the gravity vector and the support polygon formed by the wheel contact points (which is a rectangle in this case). Graphically, the margin is the minimum distance to any side of the support rectangle normalized by the height of the center of gravity above the terrain.



$$\mathrm{merit}(i) = \max\left(\frac{\Delta z_W/W}{h/W/2}, \frac{\Delta z_L/L}{h/L/2}\right) = \max\left(\frac{2\Delta z_W}{h}, \frac{2\Delta z_L}{h}\right)$$

Figure 13 - Static Stability

The margin can be expressed as the maximum of the tangent of simple approximations to roll and pitch normalized by some maximum. The distinction between measurement as an angle or its tangent is irrelevant for small angles.

One of the implications of this model is that the vehicle will turn into a steep grade so as to increase the stability margin. The system actually computes pitch and roll margins individually, and performs the maximum outside the hazard evaluator. This is a measure used to permit easy debugging. The system can navigate successfully over smooth terrain based on this hazard alone.

8.3 Hazard Representation

In general, each hazard is represented on some normalized scale where the maximum value indicates certainty of danger and the minimum value indicates certainty of safety. A typical hazard signal might look like the following:

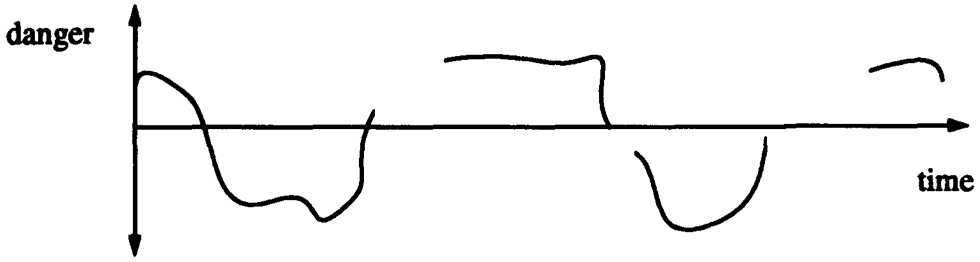


Figure 14 - Hazard Signals

8.4 Hazard Arbitration

Conceptually, the evaluation of hazards amounts to, first, the generation of a 3D field of the form ***merit(hazard,command,time)*** because safety is different for each command, different for each hazard, and a time signal. The obstacle avoidance hazard arbiter conceptually integrates this 3D field into a single decision. The first step of the process is to collapse the time dimension. This is done by computing the signal norm as follows:

$$\boxed{\text{merit} = \left(\sum_i (\text{merit}(i))^{\alpha}\right)^{1/\alpha}}$$

for some power α. In practice, however, a straightforward maximum was found to produce acceptable results. This step reduces the decision field to merit(hazard,command).

The next step is to collapse the hazard dimension. Hazards are considered independent because, for example, body collision is completely irrelevant to static stability. Therefore, a straightforward maximum applied across this dimension gives the worst unsafe condition for all dimensions of safety considered.

The final step is to reduce the command dimension, and this has been discussed in the context of joint arbitration of tactical and strategic votes. However, before the tactical vote vector is passed to the arbiter concerned, it is sometimes smoothed by **Gaussian filtering** in order to increase reliability. The idea is that, because the environment exhibits a degree of smoothness, it is appropriate at times for poorly rated trajectories to degrade the ratings of their spatial neighbors. In this manner, the impact of errors in both feedback and actuation will be less relevant because this mechanism will cause the vehicle to give obstacles a wide berth. At times, this mechanism is disabled depending on the spacing of the candidate trajectories investigated.

9. Sensor Stabilization and Pointing

Sensor head control algorithms are provided that incorporate both somatic and environmental feedback. These algorithms can also be used to control the field of view of real time programmable sensors.

9.1 Azimuth Controllers

The pan axis ***must move with the vehicle*** and because it is mounted to the vehicle, any response on this axis is useful.

The first azimuth controller is the **steering regulator**. This controller generates pan commands which match the commands to the steering wheel. It would have been possible to servo to the steering feedback, but that option is considered to incorporate a delay and provide no benefit in return.

The second azimuth controller is the **detection zone tracker**. The steering feedforward simulator provides an intelligent basis for pointing the pan axis because the centroid of the region which is reachable by the vehicle can easily be computed. This controller computes the average angle subtended at the initial position by the response curve endpoint. In a sentence, this servo looks where the vehicle is going.

9.2 Elevation Controllers

The tilt axis ***must move against the vehicle*** and therefore must be able to respond significantly faster that the vehicle itself in order to be of any use.

The first elevation controller is the **pitch regulator**. This servo drives the elevation degree of freedom to the opposite of the vehicle pitch. Note that if the vehicle pitch is indicated by an inertial sensor, the servo gives effective inertial stabilization of the head without the cost of extra gyroscopes.

The second elevation controller is the **range window tracker**. In rough terrain, the field of view can only be pointed by a loop closed around the perception system because the body attitude and terrain roughness affect the projection of the field of view onto the groundplane and only the high level control loop knows this information.

The adaptive sweep algorithm provides an excellent measure of the deviation of the sensor tilt from the ideal as the average deviation of the range window from the vertical center of the image. This error signal is used to drive the head, and in the process, a rough terrain adaptive sensor head controller results. This controller provides impressive stabilization on rough terrain and it can also drive a roll axis with little modification.

10. Speed Planning

Just like everything else, the speed which should be commanded of the vehicle depends on the local environment. It makes sense, for instance, for the vehicle to slow down when it encounters challenging terrain and likewise increase speed when benign terrain is encountered.

The motivation for speed planning is that there is reason to expect that

- there is a way to measure system robustness
- there is a way to relate robustness to speed in rough terms

The principle used for a speed planner is to close a proportional control loop around the average predicted merit of all candidate commands because this average merit is low in dense hazard environments and high in regions free of hazards. This principle is complemented by the fact that higher speeds tend to lead to reduced maneuverability and hence lower reliability **for the same local terrain**. That is, robustness can be altered in a small region by adjusting speed alone.

Many aspects of performance are implicitly figured in the algorithm. For example, occlusion, resolution, map quantization noise, etc. are all implicitly expressed in the hazard signals, so the system will eventually reach equilibrium with respect to the most constraining sensor characteristic and an appropriate speed will emerge naturally.

When a planner gets confused it tends to stay confused for a time and when it is relatively stable, it also tends to stay stable. This implies that there is a low frequency component to system robustness which is long enough in time constant to exceed the vehicle response limits. Hence, a vehicle may be able to respond fast enough to change things before its too late. It should also be noted that the tactical controller is designed so that a panic stop (slamming on the brakes) is always guaranteed to avoid vehicle damage. The speed planner uses more graceful measures to improve performance.

A simple loop is configured as follows:

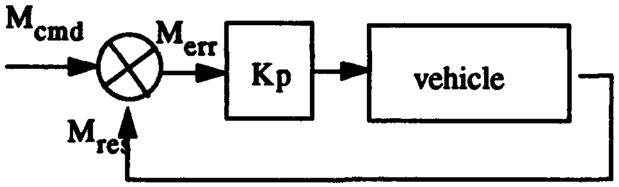


Figure 15 - Merit Servo

This loop is implemented with a time constant that is relatively large so that the speed loop will respond only to the low frequencies in the overall merit. The results for a run of the system in a dense obstacle field at high speed are indicated below.

The vehicle was driven in simulation for 500 cycles at a nominal speed of 4.5 m/s at 3 Hz cycle time. When speed planning was on, the excursion was 703 meters versus 687 meters when it was off. The planner was considered to have failed when the merit fell below a threshold of 0.4 With speed planning on, "failure" occurred 4% of the time, whereas it occurred 6% of the time with

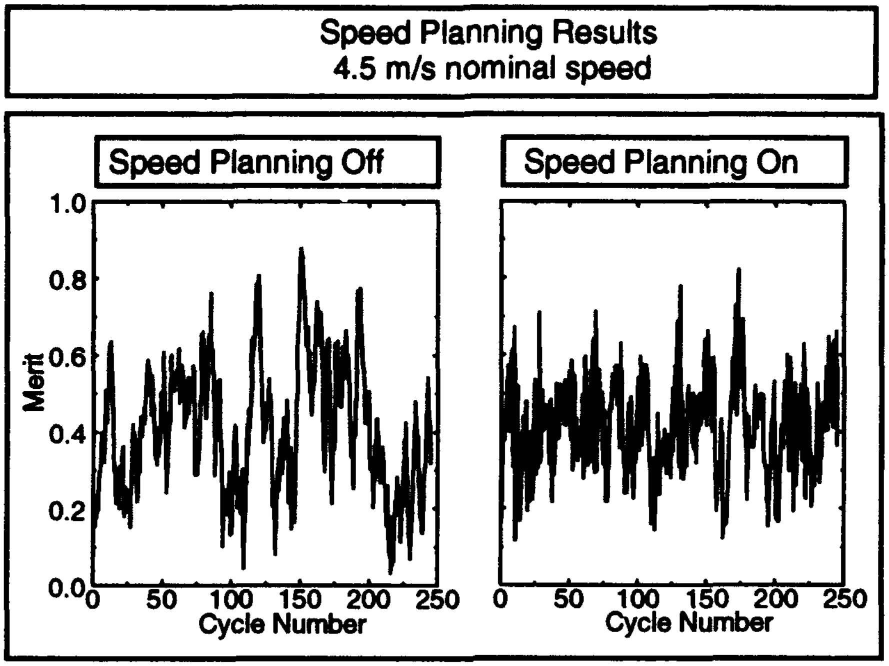


Figure 16 - Performance of Speed Planning

speed planning off. More important, the seriousness of the failures with planning off were much worse than those with planning on. The system attempts to maintain a constant merit of 0.4 when the loop is closed. The figure shows that the deviation from 0.4 is much higher in the left figure. The peaks and valleys are wider in the left figure indicating sustained periods of confusion. In plain terms the algorithm makes it possible to drive farther more robustly.

An extra benefit of the speed planning loop is that it will automatically back the vehicle up and drive off in another direction when a panic stop is issued. The path planner supports this as well because it plans correctly for negative velocities, so backs up "smartly" avoiding obstacles until a new alternative presents itself.

11. High Speed Autonomy as an Optimal Control Problem

This section casts the problem of high speed autonomy as an optimal control problem. The lowest level element of the system is the **control laws**. The control laws are responsible for coordinated control of all actuators. The control laws are implemented in a tightly coupled hierarchy. At this level of conceptualization, the system can be viewed as a multi-input, multi-output, feedforward state space controller.

11.1 State Space Controller

For ***a linear system***[10], the conventional state space model of a system is the venerated two matrix equations:

$$\frac{d\underline{\mathbf{x}}}{dt} = \mathbf{A}\,\underline{\mathbf{x}} + \mathbf{B}\,\underline{\mathbf{u}}$$
$$\underline{\mathbf{y}} = \mathbf{C}\,\underline{\mathbf{x}} + \mathbf{D}\,\underline{\mathbf{u}}$$

Note in particular that the first equation is a differential one. The **command vector $\underline{\mathbf{u}}$** includes vehicle steering and speed as well as demand signals to any sensor heads. The **state vector $\underline{\mathbf{x}}$** includes the vehicle steering, speed, and the state of motion of the vehicle body and the sensor heads[11]. The **output vector $\underline{\mathbf{y}}$** can be any function of both state and inputs, but in our case, it can be considered to be the state itself (so C is the identity and D is zero). These equations are often indicated in a block diagram as shown below:

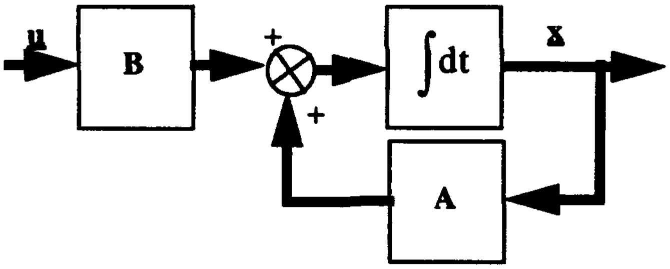


Figure 17 - Linear System Block Diagram

In this model so far, ***the system transition matrix of the last section can be identified directly with the A matrix*** and the equation which employs it to propagate the system state forward in time is the integrator. The transition matrix includes the steering model and the propulsion model. The B matrix can be identified with anything which modifies commands generated by the system before they get to the physical hardware. That is, the ***B matrix can be identified directly with the delay queues***.

---

[10] Our system is not linear. No real system is perfectly linear. Consider this a model for now, and the validity of the model to be an open question.

[11] The state vector includes more information than the commands because the number of state variables is dictated by the order of the differential equation being modelled. The system equation is a matrix equation, so its order is the dimension of the A matrix, and this need have no relationship to the length of u. This corresponds to the fact that every integral requires an initial condition. On a conventional automobile the command vector u is of order 2 whereas the state vector may have ten or more elements.

11.2 Hazard Model

Sometimes, it is useful to further distinguish the signals going to the plant from the information which is ultimately of interest. In our case, the primary quantity of interest is vehicle safety. This can be indicated with a **reference input** or demand vector $\underline{\mathbf{d}}$. Also, there may be quantities which influence the state of the system over which there is no control at all. Let these **disturbances** be called $\underline{\mathbf{u}}_d$. Disturbances can be placed anywhere that makes sense in the diagram and they may or may not be modelled. Further, the C matrix can be reintroduced and the output considered to be the safety of the vehicle expressed in terms of the hazard signal vector. Our system model has now become:

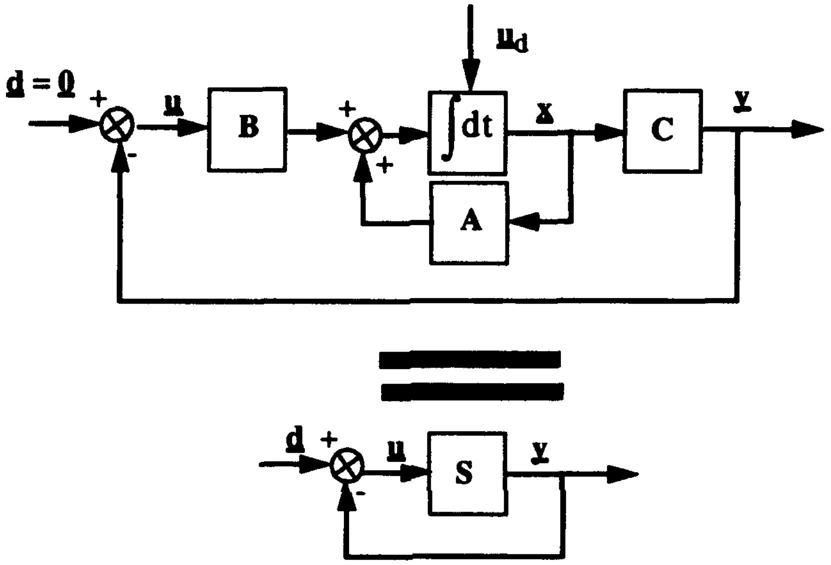


Figure 18 - State Space Model

where a single box for the **state space model** S has been generated by grouping certain elements together. Such a system implemented without feedforward would be a regulator which attempts to keep the hazard vector at 0.

The disturbances can be considered to be anything that affects the system but that is generated outside the system. In our case, the requirement that the vehicle remain on the terrain surface arises from the fact that the terrain is incompressible (by assumption) and it generates forces that prevent the vehicle from sinking. Thus, ***the disturbances can be identified directly with the suspension model.*** The ***output vector is the hazard signal vector*** and the ***C matrix is the hazard evaluation functions*** which map vehicle states and the terrain geometry onto estimate of hazards. Notice that the hazards are time signals and that there are many kinds. Each kind of hazard is one element of the output vector.

This model maps very directly onto the tactical controller. Other control loops can be considered to be subsumed in the diagram as well. Sensor head control is a regulator where the reference input is the deviation of the range window from image dead center. The strategic controller is a regulator which attempts to maintain the heading error against the global path at zero. A complete diagram is far too detailed to draw because:

- the state vector is of order close to 10
- many of the matrices are divisible into blocks
- coordinate system transforms are complicated and are a major source of coupling and nonlinearity.

11.3 Feedforward Optimal Controller

Feedforward is employed for two distinct reasons. First, it imparts stability to many of the algorithms - particularly the strategic controller and the tactical controller[12]. Second, the process by which safety is predicted is also a feedforward process.

The system attempts to guarantee safety by ***predicting*** into the future the relationship between future safety and current command options. Then it acts now based on future consequences. It does this by using a **state space simulator**. This feedforward element can be represented by duplicating the state space model and indicating that it is an estimate with a super hat diacritical mark. The fact that an optimal trajectory is chosen can be represented by replacing the summing point with a more complex minimum transformation:

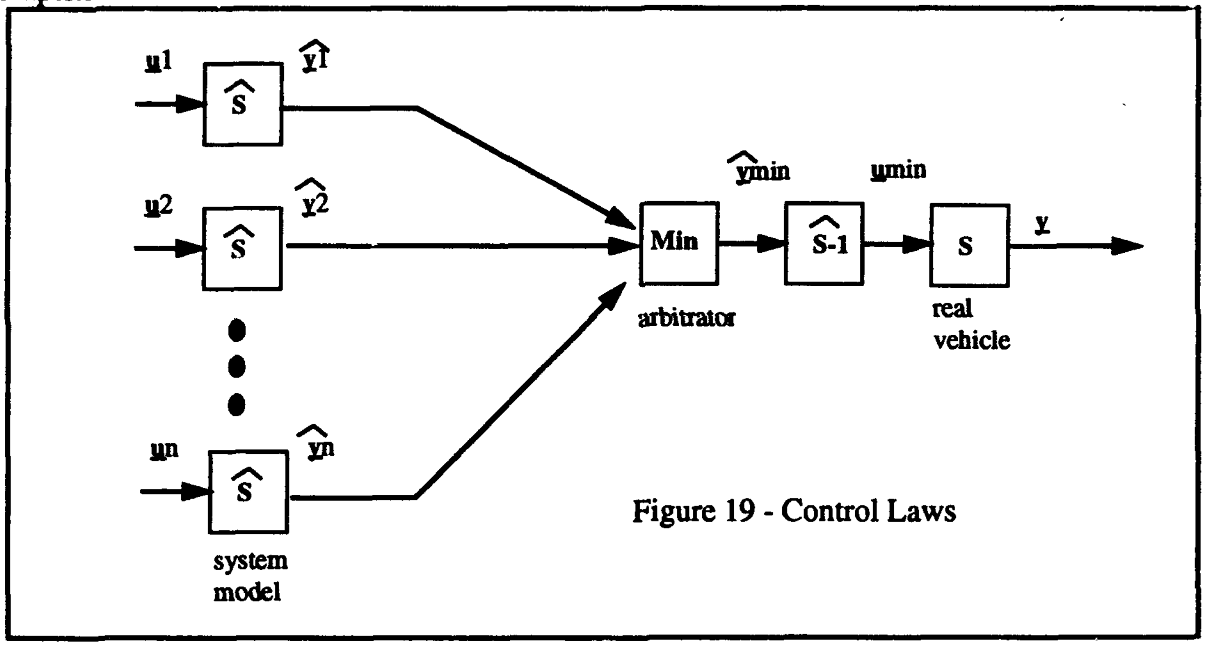


Figure 19 - Control Laws

The state space model maps easily onto the tactical controller. The tactical controller is a kind of suboptimal controller which ***attempts to minimize the norm***[13] ***of the hazard vector***. This is **guaranteed safety** again in control's language.

---

[12] It has not been mentioned yet, but feedforward prevents "waffling" in the steering arbitrator because once a slight turn is commanded once, it affects all downstream computations by biasing all subsequent decisions to reflect the distance the steering wheel turns for that cycle. Each cycle's output is a partial commitment, so the tactical controller can only sit on the fence with respect to a particular obstacle for one cycle. Feedforward leads to a more confident controller because it recognizes when a partial commitment has been made to turn in a certain direction.

[13] The norm is computed by integrating first over time and second over the dimensions of hazard space.

Formally, the system is a suboptimal solution to the following optimal control problem:

$$\min\left[L(\bar{y}(t))\right] = \sqrt{\left[\sqrt{\int_0^t y_i(t)^2}\right]^T \left[\sqrt{\int_0^t y_i(t)^2}\right]}$$

subject to:

$$\frac{d\underline{\mathbf{x}}}{dt} = \mathbf{A}\,\underline{\mathbf{x}} + \mathbf{B}\,\underline{\mathbf{u}}$$

$$\underline{\mathbf{y}} = \mathbf{C}\,\underline{\mathbf{x}} + \mathbf{D}\,\underline{\mathbf{u}}$$

where A models the steering and propulsion dynamics, B models the communications delays, and C models the terrain contact constraint as well as the mapping from system state onto hazards. The rms time integral of the hazards is generalized to the signal α norm given by:

$$\sqrt{\int_0^t y_i(t)^2} \Rightarrow \left(\int_0^t y_i(t)^\alpha\right)^{1/\alpha}$$

and the vector length is generalized to an α norm over hazard space:

$$L_\kappa(y) \Rightarrow \left[\sum_i (y_i)^\alpha\right]^{1/\alpha}$$

The formal solution to an optimal control problem is obtained from the **Pontriagin Maximum Principle** (of which the more familiar **calculus of variations** is a special case). In general, the optimal control problem generates a set of simultaneous partial differential equations with boundary conditions which are very difficult to solve in real time.

The system model amounts to a complicated differential equation constraint. The satisfaction of this constraint is generally very difficult to achieve. The set of trajectories which satisfies the system model equations and which maintain the vehicle in contact with rigid terrain is called, in optimization lingo, the **feasible set**.

The system satisfies this constraint ***by construction*** through feedforward. The inverse system model in the previous diagram is never evaluated explicitly - the system simply remembers the correspondence of commands to response trajectories and inverts this list of ordered pairs. In this manner, the system ***plans in actuation space***.

The system solves the optimization problem, effectively, by sampling the feasible set of trajectories at some practical resolution that ensures adequate coverage of the set, and then by choosing the trajectory with the best value of the functional $[L(\bar{y}(t))]$.

12. Discrete Time Nonlinear Model

The last section gave a linear systems model of a MIMO system. The system model actually used is a nonlinear model, but it can still be expressed as a set of matrix equations if the matrices themselves are allowed to vary with the state vector.

12.1 State Vector

The **state vector** includes the position and attitude of the body and the linear velocity along body y and the angular velocity around body z.

$$\bar{x} = \begin{bmatrix} x\ y\ z\ V\ \theta\ \phi\ \Psi\ \dot{\beta} \end{bmatrix}^T$$

12.2 Command Vector

The **command vector** includes the commanded speed and commanded "yawrate":

$$\bar{u} = \begin{bmatrix} V_c\ \dot{\beta}_c \end{bmatrix}^T$$

12.3 System Model

The system model is based on a **low dynamics assumption**. This is the assumption that velocity can be considered constant for a small period of time. The model is given below:

$$\begin{bmatrix} x \\ y \\ z \\ V \\ \theta \\ \phi \\ \psi \\ \dot{\beta} \end{bmatrix}_{K+1} = \begin{bmatrix} x \\ y \\ z \\ V \\ \theta \\ \phi \\ \psi \\ \dot{\beta} \end{bmatrix}_{K} + \begin{bmatrix} -Vs\psi c\theta \\ Vc\psi c\theta \\ Vs\theta \\ 0 \\ \dot{\beta}s\phi \\ -\dot{\beta}t\theta c\phi \\ \dot{\beta}c\phi/c\theta \\ 0 \end{bmatrix}_{K} dt$$

The terms in the bottom of the vector account for the nonlinear dependence of the attitude rate on the attitude of the body. Angular velocity kinematics are given in [26]. This model will correctly account for motion out of the plane even so far as to correctly turn the vehicle around the body z axis instead of the gravity vector. This aspect of the model can be expressed as a nonlinear transition matrix, if desired.

12.4 Forcing Function and Ackerman Kinematics

The relationship between the yawrate and the steer angle is:

$$\dot{\beta} = \kappa V = \frac{\tan\alpha}{L} V$$

If we ignore the nonlinear limiters for the purpose of illustration, and model the actuator loops as first order systems, the relationship between the command vector and the state derivative vector can be written as:

$$\begin{bmatrix} V \\ \dot{\beta} \end{bmatrix}_{K+1} = \begin{bmatrix} V \\ \dot{\beta} \end{bmatrix}_{K} + \begin{bmatrix} K_{prop}\,(V_{cmd} - V) \\ \dfrac{\tan\,[K_{steer}\,(\alpha_{cmd} - \alpha)]\,V}{L} \end{bmatrix} dt$$

12.5 Terrain Contact Constraint

Under a **small pitch assumption,** the terrain contact constraint alters the attitude each cycle. it can be written as:

$$\begin{bmatrix} \theta \\ \phi \end{bmatrix}_{K+1} = \frac{1}{2} \begin{bmatrix} \frac{1}{L} & \frac{1}{L} & -\frac{1}{L} & -\frac{1}{L} \\ \frac{1}{W} & -\frac{1}{W} & \frac{1}{W} & -\frac{1}{W} \end{bmatrix} \begin{bmatrix} z_{lf} \\ z_{rf} \\ z_{lr} \\ z_{rr} \end{bmatrix}$$

12.6 Hazard Vector

The hazard vector can be written as a 4 vector which includes the wheel collision, body collision, and stability margin:

$$\bar{y} = \begin{bmatrix} \overline{y_{wc}} & y_{bc} & \overline{y_{sm}} \end{bmatrix}^T$$

The wheel collision output relationships are:

$$\overline{y_{wc}}\Big|_K = \frac{1}{\Delta\left(\frac{\partial z}{\partial s}\right)_{max}} \left(\begin{bmatrix} z_{lf} \\ z_{rf} \\ z_{lr} \\ z_{rr} \end{bmatrix}_K - \begin{bmatrix} z_{lf} \\ z_{rf} \\ z_{lr} \\ z_{rr} \end{bmatrix}_{K-1} \right)$$

The static stability output relationships are:

$$\overline{y_W}\Big|_K = \frac{2}{h}\begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{bmatrix}\begin{bmatrix} z_{lf} \\ z_{rf} \\ z_{lr} \\ z_{rr} \end{bmatrix}_K$$

12.7 Implementation in Special Purpose Hardware

The entire system including perception, planning, and control, runs in less than 50 msecs on a SPARC 10 platform and this is already a very high cycle rate for the entire perceive-think-act loop. However, higher performance can be achieved by porting the system to special purpose hardware. In principle, the control laws are implementable solely in terms of matrix algebra, trigonometric functions and a few signal operators - because they really are just one big differential equation. For example, a graphics rendering pipeline would perform all of the necessary geometric transformations on image sensory data to generate the map data structure. The system model is literally a discrete matrix differential equation, and the hazard model can be implemented as a matrix. The arbitrator could then be implemented in a digital signal processor.

13. Bibliography

[1] O. Amidi, "Integrated Mobile Robot Control", Robotics Institute Technical Report CMU-RI-TR-90-17, Carnegie Mellon University, 1990.

[2] M. G. Bekker, "The Theory of Land Locomotion", The University of Michigan Press, 1956.

[3] M. G. Bekker, "Off-the-Road Locomotion", The University of Michigan Press, 1960.

[4] R. Bhatt, L. Venetsky, D. Gaw, D. Lowing, A. Meystel, "A Real-Time Pilot for an Autonomous Robot", Proceedings of IEEE Conference on Intelligent Control, 1987.

[5] W. L. Brogan, "Modern Control Theory", Quantum Publishers, 1974

[6] R. A. Brooks, "A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control", Proceedings of IEEE International Conference on Robotics and Automation, 1987.

[7] B. Brumitt, R. C. Coulter, A. Stent, "Dynamic Trajectory Planning for a Cross-Country Navigator", Proceedings of the SPIE Conference on Mobile Robots, 1992.

[8] T. S. Chang, K. Qui, and J. J. Nitao, "An Obstacle Avoidance Algorithm for an Autonomous Land Vehicle", Proceedings of the 1986 SPIE Conference on Mobile Robots, pp. 117-123.

[9] M. Daily, "Autonomous Cross Country Navigation with the ALV", Proceedings of the 1988 IEEE International Conference on Robotics and Automation, pp. 718-726.

[10] E. D. Dickmanns, "Dynamic Computer Vision for Mobile Robot Control", Proceedings of the 19th International Symposium and Exposition on Robots, pp. 314-27.

[11] E. D. Dickmanns, A. Zapp, "A Curvature-Based Scheme for Improving Road Vehicle Guidance by Computer Vision", Proceedings of the SPIE Conference on Mobile Robots, 1986.

[12] J. C. Dixon, "Linear and Non-Linear Steady State Vehicle Handling", Proceedings of Instn Mechanical Engineers, Vol. 202, No. D3, 1988.

[13] R. T. Dunlay, D. G. Morgenthaler., "Obstacle Avoidance on Roadways Using Range Data", Proceedings of SPIE Conference on Mobile Robots, 1986.

[14] D. Feng, "Satisficing Feedback Strategies for Local Navigation of Autonomous Mobile Robots", Ph.D. Dissertation at CMU, May, 1989.

[15] D. Feng, S. Singh, B. Krogh, "Implementation of Dynamic Obstacle Avoidance on the CMU Navlab", Proceedings of IEEE Conference on Systems Engineering, August, 1990.

[16] U. Franke, H. Fritz, S. Mehring, "Long Distance Driving with the Daimler-Benz Autonomous Vehicle VITA", PROMETHEUS Workshop, Grenoble, December, 1991.

[17] J. Gowdy, A . Stentz, and M. Hebert, "Hierarchical Terrain Representation for Off-Road Navigation", In Proc SPIE Mobile Robots 1990.

[18] D. Langer, J. K. Rosenblatt, M. Hebert, "A Reactive System For Off-Road Navigation", CMU Tech Report

[19] M. Hebert and E. Krotkov, "Imaging Laser Radars: How Good Are They", IROS 91, November 91.

[20] M. Hebert. "Building and Navigating Maps of Road Scenes Using an Active Sensor", In Proceedings IEEE conference on Robotics & Automation, 1989; pp.36-1142.

[21] M. Hebert, T. Kanade, and I. Kweon. "3-D Vision Techniques for Autonomous Vehicles", Technical Report CMU-RI-TR-88-12, The Robotics Institute, Carnegie Mellon University, 1988

[22] B.K Horn and J. G. Harris, "Rigid Body Motion from Range Image Sequences", Image Understanding, Vol 53, No 1, January 1991, pp 1-13

[23] R. Hoffman, E. Krotkov, "Terrain Mapping for Outdoor Robots: Robust Perception for Walking in the Grass", Submitted to IEEE International Conference on Robotics and Automation, 1993.

[24] D. Keirsey,D. Payton, J. Rosenblatt, "Autonomous Navigation in Cross-Country Terrain", proceedings of Image Understanding Workshop, 1988.

[25] A. Kelly, A. Stentz, M. Hebert, "Terrain Map Building for Fast Navigation on Rough Terrain", Proceedings of the SPIE Conference on Mobile Robots, 1992.

[26] A. J. Kelly, "Essential Kinematics for Autonomous Vehicles", CMU Robotics Institute Technical Report CMU-RI-TR-94-14.

[27] A. J. Kelly, "Modern Inertial and Satellite Navigation Systems", CMU Robotics Institute Technical Report CMU-RI-TR-94-15.

[28] A. J. Kelly, "A Partial Analysis of the High Speed Autonomous Navigation Problem", CMU Robotics Institute Technical Report CMU-RI-TR-94-16.

[29] A. J. Kelly, "Adaptive Perception for Autonomous Vehicles", CMU Robotics Institute Technical Report CMU-RI-TR-94-18.

[30] A. J. Kelly, "A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles", CMU Robotics Institute Technical Report CMU-RI-TR-94-19.

[31] A. J. Kelly, "An Intelligent Predictive Controller for Autonomous Vehicles", CMU Robotics Institute Technical Report CMU-RI-TR-94-20.

[32] A. J. Kelly, "Concept Design of A Scanning Laser Rangefinder for Autonomous Vehicles", CMU Robotics Institute Technical Report CMU-RI-TR-94-21.

[33] In So Kweon, "Modelling Rugged Terrain by Mobile Robots with Multiple Sensors", CMU PhD Thesis, 1990

[34] T. Lozano-Perez, and M. A. Wesley, "An Algorithm for Planning Collision Free Paths Among Polyhedral Obstacles". Communications of the ACM, Vol. 22, Num. 10, October 1979, pp. 560-570.

[35] M. Marra, R. T. Dunlay, D. Mathis, "Terrain Classification Using Texture for the ALV", Proceedings of SPIE Conference on Mobile Robots, 1988.

[36] L. S. McTamaney, "Mobile Robots: Real-Time Intelligent Control", IEEE Expert, Vol. 2, No. 4, Winter, 1987.

[37] H. P. Moravec, "The Stanford Cart and the CMU Rover", Proceedings of the IEEE, Vol. 71, Num 7, July 1983, pp. 872-884.

[38] K. Olin, and D. Tseng, "Autonomous Cross Country Navigation", IEEE Expert, August 1991, pp. 16-30.

[39] D. A. Pomerleau, "Efficient Training of Artificial Neural Networks for Autonomous Navigation", Neural Computation, Vol. 3, No. 1, 1991, pp88-97.

[40] J. Rosenblatt,D. Payton, "A Fine-Grained Alternative to the Subsumption Architecture", Proceedings of the AAAI Symposium on Robot Navigation, March, 1989.

[41] S. Singh et. al, "FastNav: A System for Fast Navigation,", Robotics Institute Technical Report CMU-RI-TR-91-20, Carnegie Mellon University, 1991.

[42] S. Shafer, A. Stentz, C. Thorpe, "An Architecture for Sensor Fusion in a Mobile Robot", Proceedings of IEEE International Conference on Robotics and Automation", April, 1986.

[43] H. C. Stone, "Design and Control of the Mesur/Pathfinder Mocrorover", JPL

[44] D. H. Shin, S. Singh and Wenfan Shi. "A partitioned Control Scheme for Mobile Robot Path Planning", Proceedings IEEE Conference on Systems Engineering, Dayton, Ohio, August 1991

[45] A. Thompson, "The Navigation System of the JPL Robot", Proceedings of the International Joint Conference for Artificial Intelligence, 1977, pp749-757.

[46] B. Wilcox et al. "A Vision System for a Mars Rover. SPIE Mobile Robots II", November 1987, Cambridge Mass., pp. 172-179.

Index

A
adaptive regard 8
C
calculus of variations 33
command vector 30, 34
continuity assumption 9
control laws 30
D
detection zone tracker 27
disturbances 31
F
feasible set 33
G
Gaussian filtering 8, 26
guaranteed safety 32
L
local minimum problem 10
locally planar terrain assumption 18
low dynamics assumption 34
M
monotone arc length assumption 12
O
occlusion problem 23
output vector 30
P
pitch regulator 27
planning window 15
Pontriagin Maximum Principle 33
pure pursuit 11
R
range window tracker 27
reference input 31
reference points 18
rigid suspension assumption 18
rigid terrain assumption 18
S
sampling problem 23
small pitch assumption 35
state space model 31
state space simulator 32
state vector 30, 34
static stability margin 24

steering regulator 27
strategic goal 10

T

tactical goal 10
terrain smoothness assumption 9

U

unknown hazard assumption 9