NAVIGATION AND RETRO-TRAVERSE ON A REMOTELY OPERATED VEHICLE

Karl Murphy

Systems Integration Group
Robot Systems Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

During the retro-traverse function, a computer controlled vehicle automatically retraces a previously recorded path that is stored as a series of x-y points. Two navigation systems were tested: an on-board inertial navigation system and a radio grid system. Two steering algorithm were used. In the state space method, the steering command is proportional to the lateral position error and the heading error. In the pure pursuit method, the vehicle steers toward a goal point on the path a specified distance in front of the vehicle. On a smooth path these methods are identical. On a piecewise linear path, the pure pursuit method is significantly superior in performance. With this method, the vehicle follows paths accurately up to speeds of 70 kph.

Introduction

The U.S. Army Laboratory Command is studying the control of unmanned land vehicles as part of the Defense Department's Robotics Testbed (RT) Program. In the RT scenario, humans remotely operate several Robotic Combat Vehicles (RCVs) from an Operator Control Unit (OCU). Each vehicle contains: actuators on the steering, brake, throttle, transmission, transfer case, and parking brake; an inertial navigation system; a mission package which performs target detection, tracking, and laser designation; and data and video communication links. The OCU contains controls and displays for route planning, driving, operation of the mission package, and control of the communication links.

A typical mission includes a planning phase where the operator plans a route using a digital terrain data base. The operator then remotely drives the vehicle to a desired location as the vehicle records the route using navigation data. The operator activates the mission package for automatic target detection, and when targets are detected, the mission package designates them with a laser. The vehicle then automatically retraces the route, a process termed *retro–traverse*.

The RT control architecture follows the Real-time Control System (RCS) methodology developed at National Institute of Standards and Technology (NIST) [1, 2, 3]. The RCS architecture is a hierarchy of control modules in which each module controls one or more modules at the next lower level. High-level commands are decomposed into simpler commands as they pass down through the levels.

Each control module is composed of task decomposition, world modeling, and sensor processing. Task decomposition implements real-time planning, control, and monitoring functions. It decomposes tasks both spatially and temporally. The sensory processing modules detect, filter, and correlate sensory information. The world modeling modules estimate the state of the external world and make predictions and evaluations based on these estimates

The vehicle used for the Robotics Testbed is the U. S. Army's High Mobility Multipurpose Wheeled Vehicle (HMMWV), a one and one quarter ton, four wheel drive truck. The vehicle actuator package was originally developed by Kaman Sciences for targets and was therefore intended to be low cost and expendable. This package is currently being replaced with a faster, more accurate actuator package designed specifically for the higher performance required for remote vehicle driving.

Two types of navigation sensors were tested. The first, an inertial navigation system contained entirely on the vehicle does not require external aids. The U.S. Army Modular Azimuth and Position System (MAPS), uses an odometer, three accelerometers, and three optical ringlaser gyros. The second navigation system, the Radio Frequency Navigation Grid (RFNG), provides vehicle position data during testing and development. This grid employs radio beacons placed at the corners of the test area which emit timing pulses received by the vehicle. Differences inphase information from each transmitter are used to compute vehicle position. These systems are described in the next section of this paper, Navigation.

This work is funded by the U.S. Army Laboratory Command. This paper was prepared by U.S. Governme employees and is not subject to copyright. Equipment listings do not imply a recommendation by NIST.

The following section, Retro-Traverse, describes the retro-traverse mode of driving, which is analogous to teach programming and playback commonly used by industrial robots. Initially, a path (i.e., a sequence of x-y positions) is recorded as the operator remotely drives the vehicle. Later, the controller turns the vehicle around using an operator selected maneuver and initiates path following. During path following, steering commands are calculated that drive the vehicle along the path using one of two methods. In the state space method, the steering is set proportional to the lateral position error and the heading error. In the pure pursuit method, which was developed by Amidi [4], the steering is set so that the vehicle will steer toward a point a certain distance ahead on the path. These methods are shown to be the same when the path is continuous in curvature. When the curvature is not continuous, the pure pursuit method is considerably more stable.

Navigation

Two different navigation systems were studied for use on the RT vehicle. The MAPS is an inertial navigation / dead reckoning system while RFNG is a non-inertial positioning system that uses radio beacons placed on the corners of the test site.

MAPS

The U.S. Army's Modular Azimuth and Positioning System, MAPS, is an inertial navigation system containing three ring laser gyros, three accelerometers, and a rear axle odometer. MAPS can supply orientation and translation data at a 5 Hz rate with a translation resolution of 1 meter. To obtain faster update and a higher resolution, Alliant Tech developed a Navigation Interface Unit, NIU. The NIU requests orientation data from the MAPS, which is available by itself at 25 Hz, and taps into the odometer for translation data. It then integrates the odometer data, providing position and heading to the mobility controller at 25 Hz.

One shortcoming of using the NIU is that it assumes that the vehicle's velocity is in the same direction that the MAPS is pointing. This is not exactly the case due to mounting errors and vehicle side slip. The MAPS monitors its accelerometers to detect and correct these errors. We plan to modify the NIU so that it will periodically request translation data from the MAPS to correct its position estimate. The positional updates will be made when the vehicle is stopped to avoid interrupting the 25 Hz orientation update rate.

The MAPS/NIU position measurment drifts at a rate slightly more than 0.1% of distance traveled, as determined by driving the vehicle around closed paths of various lengths and returning to the starting position. If the operator drives the vehicle 500 meters out and the vehicle retro-traverses back 500 meters, the MAPS will have drifted 1 meter.

The MAPS/NIU accuracy is 2% of distance from the start point. This was determined by driving the vehicle to several surveyed location and comparing the surveyed data with the MAPS data. This error occurs mainly because the assumed ratio of distance traveled to odometer pulses was slightly off.

RFNG

Kaman Science's Radio Frequency Navigation Grid, RFNG, is a non-inertial positioning system and has no drift. However, it requires ground-base transmitters, limiting RFNG to applications such as vehicle testing or training missions.

Three RFNG transmitters broadcast radio pulses, one after the other. Two transmitters, A and B, are designated as slaves and phase lock their output pulses to the pulses received from the master. A receiver on the vehicle measures the phase differences between the master and A pulses and between the master and B pulses. The phase differences are directly related to the range difference to the transmitters. Imagine the vehicle moves away from the master and toward slave B. The pulse from B now take less time to arrive while the pulse from the master takes longer, increasing the master-B phase difference. The change in phase difference multiplied by the wave length, 12.6 meters, is equal to the change in range difference.

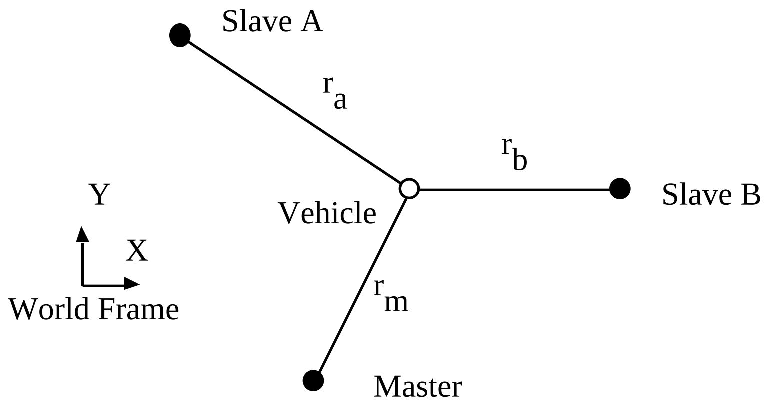


Figure 1. The RFNG transmitters to vehicle ranges.

Computing vehicle position is an iterative process. Consider Figure 1. The range or distance from the vehicle to the three transmitters is

$$\begin{aligned} r_m &= \sqrt{(x - x_m)^2 + (y - y_m)^2} \\ r_a &= \sqrt{(x - x_a)^2 + (y - y_a)^2} \\ r_b &= \sqrt{(x - x_b)^2 + (y - y_b)^2} \end{aligned} \qquad (1)$$

where (x, y), (x_m, y_m), (x_a, y_a), and (x_b, y_b), are the position of the vehicle, the master, the slave A, and slave B transmitters. The range differences are

$$\begin{aligned} r_{ma} &= r_m - r_a \\ r_{mb} &= r_m - r_b \end{aligned} \qquad (2)$$

A closed form solution does not exist for x and y and the equations (1) and (2) must be solved iteratively. Initial estimates for x and y are used to calculate estimated r_{ma} and r_{mb}, which are then used to calculate new estimates for x and y using Newton's method

$$\Delta\bar{\mathbf{x}} = \left(\frac{\partial\bar{\mathbf{r}}}{\partial\bar{\mathbf{x}}}\right)^{-1} \Delta\bar{\mathbf{r}}$$

or

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \frac{\partial r_{ma}}{\partial x} & \frac{\partial r_{ma}}{\partial y} \\ \frac{\partial r_{mb}}{\partial x} & \frac{\partial r_{mb}}{\partial y} \end{bmatrix}^{-1} \begin{bmatrix} \Delta r_{ma} \\ \Delta r_{mb} \end{bmatrix} \qquad (3)$$

where Δr_{ma} and Δr_{mb} are the errors between the estimated and actual range differences, and Δx and Δy are the changes in the x and y estimates to be used for the next iteration. This converges within a few iterations.

The determinate of the partial differential matrix provides a measure of system performance. Recall that the inverse of a matrix is the adjoint divided by the determinate. Poor system performance occurs when the determinate is near zero. In this case the system is less sensitive and large changes in position correspond to only small changes in range difference. Better performance occurs when the determinate is large, small changes in position correspond to large changes in range difference. The determinate is shown in Figure 2. As expected the determinate is the largest and the sensitivity is the greatest in the triangular region between the transmitters.

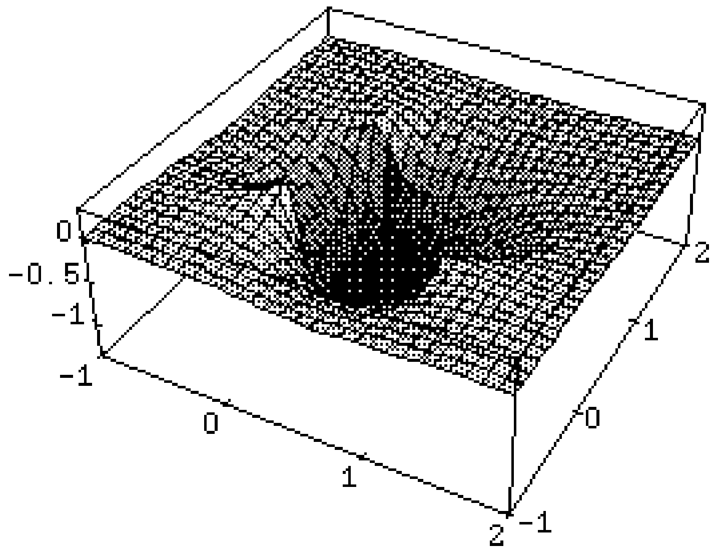


Figure 2. The determinate of the partial differential matrix. Poor system

performance occurs when the determinate is near zero. The transmitters are located at (0, 0), (1, 0), and (0, 1).

The RFNG provides positional updates at 20 Hz. The circular error probability, CEP, is 0.01 meters when the vehicle is stationary and 0.1 meter when the vehicle is moving. Velocity and heading are computed by differencing position.

RETRO-TRAVERSE

During retro-traverse the on board controller drives the vehicle, retracing a previously recorded path. The path is stored as a series of x-y positions. The navigation system measures the vehicle's position and velocity which is used to compute steering, brake, and throttle commands. Velocity is maintained using a PID controller on the throttle and brake similar to a car's cruise control. Two algorithms were used for steering, a state space and a pure pursuit controller. To help analyze these methods, a simple model of the vehicle is developed next, followed by a description of the path.

Bicycle Model

A simplified model for the vehicle is the bicycle model, shown in Figure 3, which assumes no wheel slip. The instantaneous curvature, γ, of the vehicle's path is defined as

$$\gamma = \frac{d\theta}{ds} \qquad (4)$$

where θ is the heading and s is the distance traveled. A path that curves to the left has positive curvature, to the right has negative curvature, and a straight path has zero curvature. The instantaneous radius of curvature, r, is the inverse

$$r = \frac{1}{\gamma} \qquad (5)$$

The incremental change in translation is

$$\begin{aligned} dx &= \cos\theta\, ds \\ dy &= \sin\theta\, ds \end{aligned} \qquad (6)$$

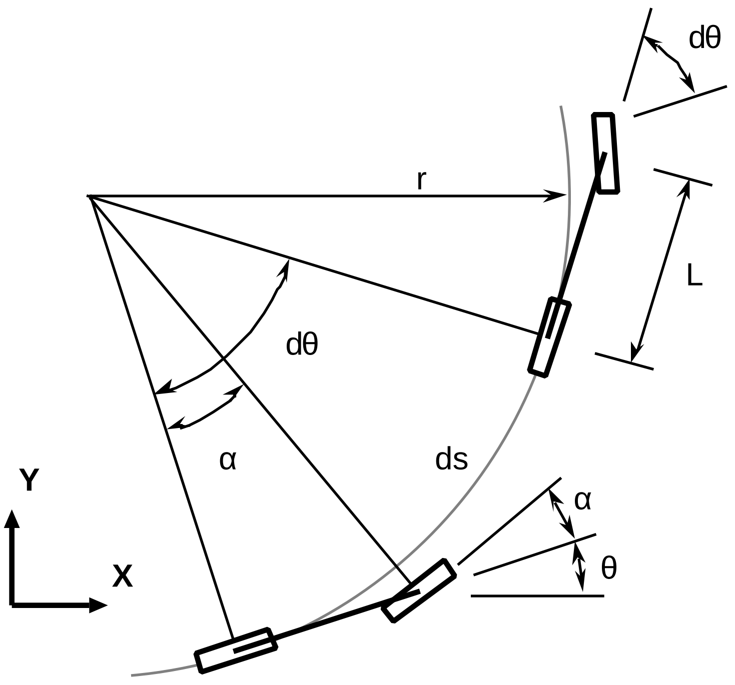


Figure 3. A bicycle model with no wheel slip. The radius of curvature, r, is a function of the front wheel angle, α

The exact relation between the curvature of a vehicle's trajectory and the steering angle, α, is complex and varies with surface conditions, vehicle speed, tire stiffness, etc. For the bicycle model, the curvature is proportional to the steering angle, α, when α is small

$$\alpha = \tan^{-1}\left(\frac{L}{r}\right) = L\,\gamma_P \qquad (7)$$

where L is the wheel spacing.

The RCS servo controller accepts the steering command in units of curvature, m^{-1}, rather than say the position of the steering wheel in degrees. The benefit is that the upper levels do not need to know the details of the vehicle but need only express the steering command as a function of the desired trajectory in terms independent of the particular vehicle.

The Path

The taught path is represented as a series of line segments defined by a set of x-y positions, or knot points, that denote the start and end of each path segment. As the vehicle moves during teaching, the navigation data is monitored and a new end point is recorded after the vehicle has traveled a specified distance, 1 meter for the RT vehicle. During playback, the x–y positions are recalled in sequence as the vehicle moves past each successive segment.

Selecting the segment length depends on the minimum turn radius of the vehicle and the amount of noise in the navigation data. Since the path represents a curved trajectory with straight line segments the greatest error occurs at the minimum turn radius, seven meters for the RT vehicle. Using a segment length of 1 meter, the maximum error is 2 cm.

The knot points could be selected closer and closer together to better and better approximate the trajectory. However, two factors limit how closely spaced the points can be. First, the actual trajectory is not known, only discrete navigation points with a spacing dictated by the navigation update rate and vehicle speed. Second, if the points are too close, noise in the navigation data will result in a jagged path. For example, recording path segments 0.1 meter long from RFNG data with noise of 0.1 meter would result in a rough path.

Besides positional errors, using a piecewise linear path to represent a vehicle's trajectory has problems with the first and second derivatives. The first derivative is related to heading and the second derivative is related to curvature. The trajectory of a vehicle will have a continuous curvature and a continuous slope, if the steering wheel does not have step changes in position. However, at a knot point, the piecewise linear path has a discontinuous slope and an infinite curvature.

A path that better represents the original trajectory would use splines that are continuous in curvature at the knot points. Cubic polynomials that have a continuous first and second derivative at each knot point could be fitted to the series of N points. Solving for the cubic polynomials requires inverting an NxN matrix. The matrix is tridiagonal so the inversion is straight forward [5], though computationally time consuming. If the navigation data is noisy, B-splines [5] could be used because they are not forced to pass through each knot point as are the cubic polynomials. Computing the B-splines also involves inverting a banded matrix.

The added complexity of fitting splines to the path points should be undertaken only if the selected steering method requires a smooth path to follow, such as the state space method described next.

State Space Control

The state space method is a commonly used algorithm that sets the steering angle proportional to the distance the vehicle is off the path and its heading error. The curvature of the path may also be added. Using the notation shown in Figure 4, the steering command is

$$\gamma_{steer} = \gamma_{path} - K_y\, y_{err} - K_\theta \theta_{err} \qquad (8)$$

where the K's are control constants, y_{err} is the lateral error, θ_{err} is the heading error, and γ_{path} is the curvature of the path near the vehicle.

The controller has two parts. The feed forward part, γ_{path}, steers the vehicle so that its trajectory has a curvature close to that of the path. If there were no following errors, this part of the controller would keep the vehicle on the path, curving the vehicle's trajectory as the path curved. The remaining part of the controller is the feedback part and steers the vehicle back to the path when following errors exist. The negative signs on the feedback terms are needed because if either y_{err} or θ_{err} are positive, the vehicle should turn to the right which is a negative curvature.

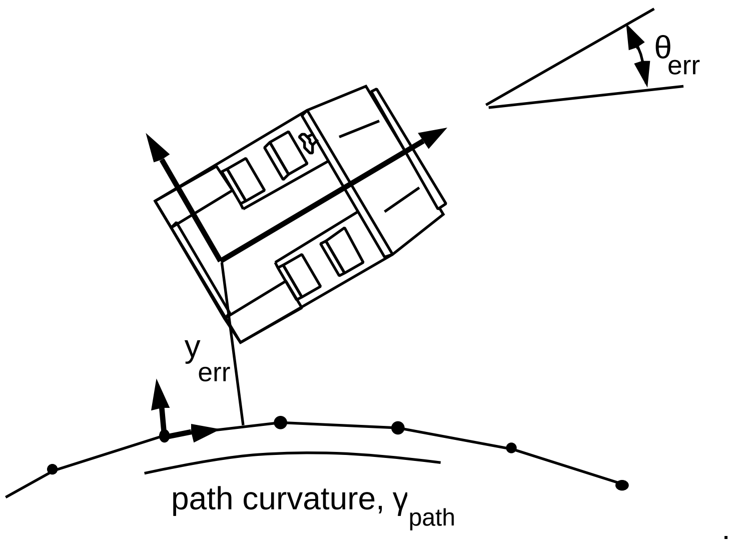


Figure 4. The state space steering method. The steering command is computed from the lateral error, the heading error, and the path curvature.

To analyze the feedback portion of this controller, consider the motion of the vehicle as expressed in the path frame of a straight path ($\gamma_{path} = 0$). Assuming a small heading error, Equation (6) becomes

$$\frac{dy_{err}}{ds} = \theta_{err} \qquad (9)$$

and the trajectory's curvature, Equation (4), becomes

$$\gamma_{steer} = \frac{d\theta_{err}}{ds} = \frac{d^2 y_{err}}{ds^2} \qquad (10)$$

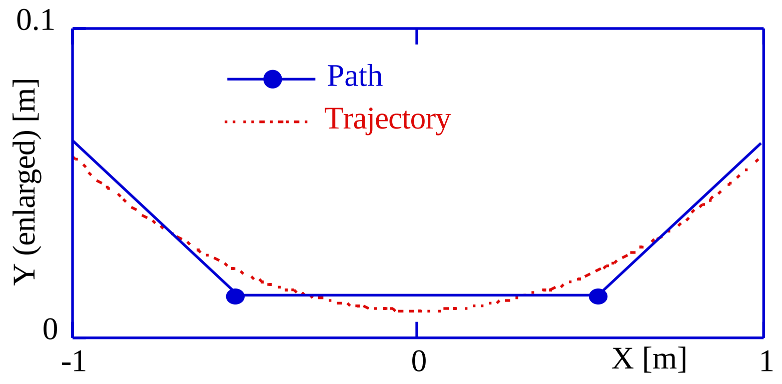
Substituting into Equation (8) yields the dynamics of the lateral error as a function of the distance traveled.

$$\frac{d^2 y_{err}}{ds^2} + K_\theta \frac{dy_{err}}{ds} + K_y\, y_{err} = 0 \qquad (11)$$

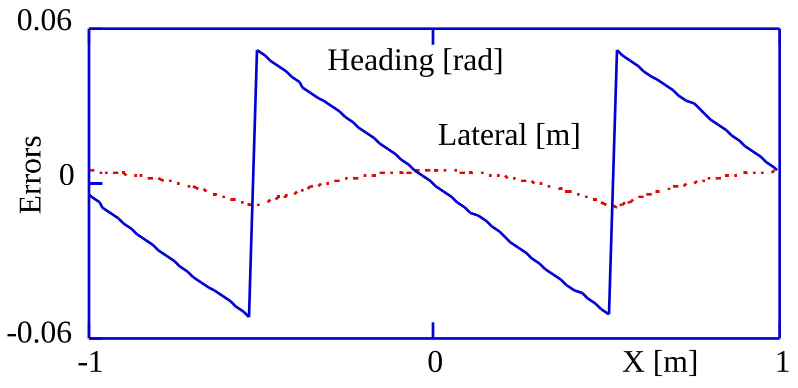
The feedback gains K_y and K_θ can be chosen to provide the desired characteristics such as damping ratio and natural frequency. Note that Equation (11) is not time dependent. The lateral error is a function that depends solely on distance traveled. The vehicle, or at least the ideal bicycle model, will follow the same trajectory on the ground regardless of its speed. The trajectory of the actual vehicle, with the vehicle's control delays, would be affected by velocity and the natural frequency would need to be decreased as control delays increase.

Although this method is stable and has a predictable response, it requires a smooth path. The piecewise linear path is too rough for the vehicle to follow the path exactly. The steering wheel oscillates, turning left and then right, as the vehicle passes the knot points on the path. This becomes unstable at all but the lowest speeds, less than 10 kph.

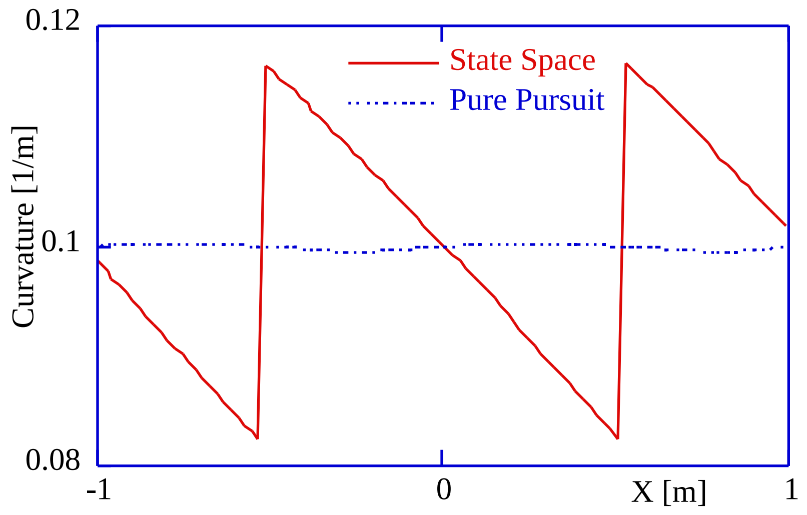
To understand this problem, consider the vehicle traveling a circular trajectory with a turn radius of 10 meters as it passes knot points separated by 1 meter, see Figure 5a. Even though the positional error is never larger than 1 cm, the heading error reaches 0.05 radians, 3 degrees, see Figure 5b. Using feedback gains of $K_y = 0.066\ m^{-1}$ and $K_\theta = 0.333\ m^{-2}$, the steering command that would be issued using Equation (8) oscillates between 0.12 and 0.08 m^{-1}, see Figure 5c. On the RT vehicle, this translates to turning the steering wheel back and forth ±90 degrees, and doing this once every meter traveled. Increasing K_θ, which normally increases damping, would only amplify the oscillations. For comparison, the steering command generated using the pure pursuit method, described in the next section, is also shown in Figure 5c.



(a) Acircular trajectory and piecewise linear path.



(b) Lateral and heading errors.



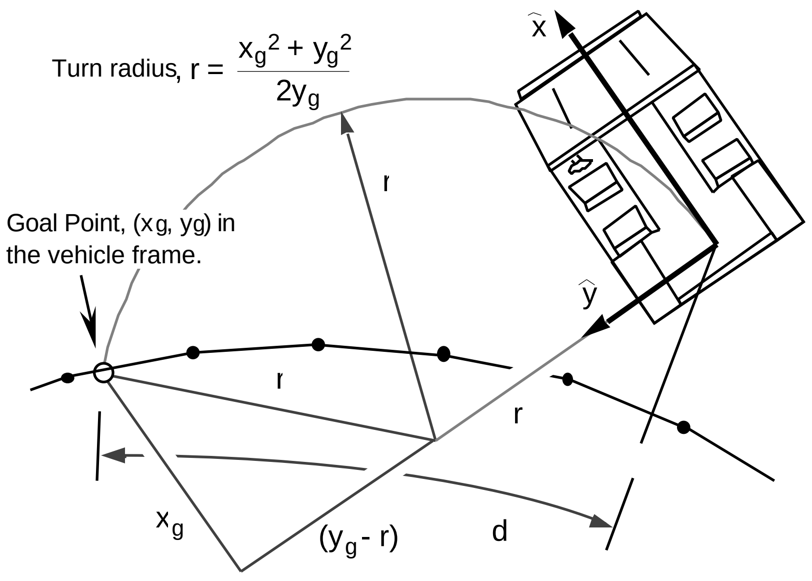
(c) Resulting steering command

Figure 5. As the vehicle travels along a circular trajectory (a), position and heading errors (b) result in a wildly fluctuating steering command when the state space methods is used (c). The steering command for the pure pursuit method using equivalent gains results in a much more stable steering command.

Clearly, a smoother path with a more continuous slope must be used with this steering method. However, to avoid the problem of computing a smooth path the pure pursuit steering algorithm was investigated.

Pure Pursuit

The pure pursuit method, as proposed by Amidi [4], is a very simple and stable steering algorithm that works well with the piecewise linear path. This method, unlike the state space method, does not “look” at the the path next to the vehicle since the vehicle can not move sideways, but instead looks to a point up ahead on the path and steers to that point. The mobility controller repeatedly selects the turn radius so that the vehicle will drive to a goal point on the path a set distanceahead of the vehicle. See Figure 6.



Turn radius, $r = \frac{x_g^2 + y_g^2}{2y_g}$
$\hat{x}$
r
Goal Point, (x_g, y_g) in the vehicle frame.
$\hat{y}$
r
r
x_g
$(y_g - r)$
d

Figure 6. Pure Pursuit Steering. The goal point is a point on the path a specified distance, d, ahead of the vehicle. The turn radius, and hence the steering angle, is selected to drive the vehicle over the goal point. The goal point, turn radius, and steering angle are repeatedly updated as the vehicle moves.

Calculating the turn radius is straight forward. Compute the position of the goal point in the vehicle frame, (x_g, y_g). Then, noting that the center of the turn circle is (0, r), one obtains

$$(x_g - 0)^2 + (y_g - r)^2 = r^2 \tag{12}$$

Solving for r and taking the reciprocal yields the steering curvature for pure pursuit, γ_{pp}, or

$$\frac{1}{r} = \gamma_{pp} = \frac{2y_g}{x_g^2 + y_g^2} \tag{13}$$

Unlike the state space method, the pure pursuit method is not affected by small bumps in the path, see Figure 5c. On a “smooth” path the two methods are identical. Consider the vehicle on a path with zero position and heading error. On a path with a slowly changing curvature, the commanded steering curvature, γ_{pp}, is the same as the path’s curvature, γ_{path}. If the vehicle were displaced a small distance, Δy, the position of the goal point in the new vehicle frame becomes $(x_g, y_g - \Delta y)$. Using Equation 13, and rearranging we find the new steering command is

$$\gamma_{pp} = \gamma_{path} - \frac{2}{d^2}\Delta y \tag{14}$$

where d is the look ahead distance. Likewise, if the heading is also changed a small amount, $\Delta\theta$, then the new steering command becomes

$$\gamma_{pp} = \gamma_{path} - \frac{2}{d^2}\Delta y - \frac{2}{d}\Delta\theta \tag{15}$$

This has the same response as the state space controller, Equation 8. The natural frequency and damping ratio are

$$\omega_n = \frac{\sqrt{2}}{d}$$
$$\zeta = \frac{1}{\sqrt{2}} \approx 0.7071 \tag{16}$$

Notice that the damping ratio is constant and does not depend on the look ahead distance, d. Since the damping ratio is less than 1 and the system is slightly underdamped. The natural frequency ω_n decreases as the look ahead distance increases. Increasing d causes the system to respond slower. The step response for the pure pursuit method using the bicycle model is shown in Figure 7.

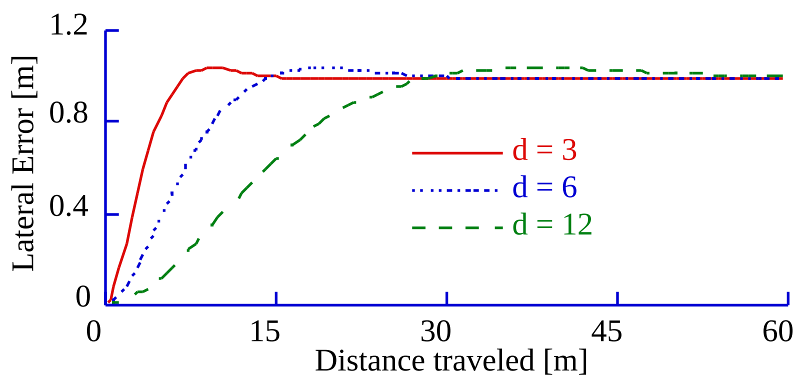


Figure 7. Pure Pursuit Step Response. Simulation of the bicycle model shows the response slows as the look ahead distance, d, increases. The trajectories are slightly underdamped.

The look ahead distance is selected according to vehicle speed. If the look ahead distance is too large, the response will be slow and the vehicle will cut corners. If the distance is too small, the controller will be unstable especially at higher speeds. The RT vehicle used 6 meters for speeds up to 30 kph and 12 meters for speeds up to 80 kph. Results from a test of the RT vehicle are shown in Figure 8. The look ahead distance is 6 meters and the speed ranges from 10 kph in the sharp turns to 30 kph on the straighter portions. Note the vehicle cuts the sharp corners slightly and is 0.5 meters off course during the sharp turns. The average lateral error is 0.1 meter RMS, as measured by the MAPS/NIU. This path following error is well within the drift error for the MAPS/NIU which was 0.5 meters at the end of the round-trip path. In other tests, the pure pursuit method was used to drive the vehicle at speeds of 70 kph.

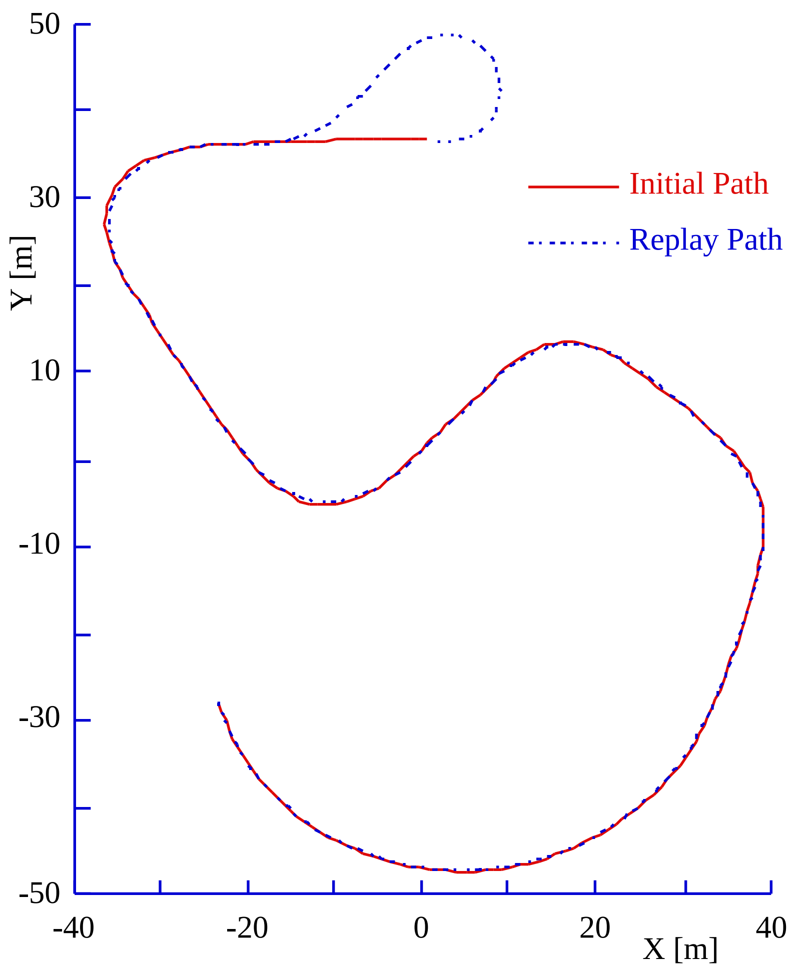


Figure 8. Experimental results for the RT vehicle and the MAPS/NIU using the pure pursuit steering algorithm with look ahead distance of 6 meters at speeds ranging from 10 to 30 kph.

Summary

The U.S. Army Laboratory Command, as part of the Department of Defense Robotics Testbed Program, is developing a testbed for cooperative, real-time control of unmanned land vehicles. The program entails the development and integration of many elements which allow the vehicles to perform both autonomous and teleoperated functions. During the retro-traverse function the vehicle automatically uses navigation data to drive back along a previously recorded path.

The effectiveness of retro-traverse depends on the repeatability of the navigation systems. The MAPS, and an associated Navigation Interface Unit, provides 3-D position and orientation at 25 Hz with drift of 0.1% of distance traveled. The RFNG system provides position updates at 20 Hz with 0.1 meter repeatability and does not drift.

Two steering algorithms were tested. In the state space method, the steering angle is set proportional to the perpendicular distance the vehicle was from the path and the heading error. A feed forward term was also included to account for the curvature of the path. On a smooth path, this method allows the selection of the vehicle response characteristic but becomes unstable on a piecewise linear path.

The second method, pure pursuit, steers the vehicle toward a goal point on the path a specified distance in front of the vehicle. On a smooth path, this method is identical to the state space method. On a piecewise linear path it is significantly superior as it is not disturbed by the slight corners on the piecewise linear path. With this method, the vehicle follows paths accurately at speeds up to 70 kph.

References

[1] S. Szabo, H. A. Scott, K. N. Murphy, and S. A. Legowik, “High-Level Mobility Controller for a Remotely Operated Unmanned Land Vehicle,” Journal of Intelligent and Robotic Systems, to be published 1992.

[2] S. Szabo, H. A. Scott, K. N. Murphy, and S. A. Legowik, Control System Architecture for a Remotely Operated Unmanned Land Vehicle, in Proceedings of the 5th IEEE International Symposium on Intelligent Control, Philadelphia, PA, September, 1990.

[3] S. Szabo, H. A. Scott, and R. D. Kilmer, Control System Architecture for the TEAM Program, in Proceedings of the Second International Symposium on Robotics and Manufacturing Research, Education and Applications, Albuquerque, NM, November, 1988.

[4] O. Amidi, Integrated Mobile Robot Control, M.S Thesis, Carnegie Mellon University, May, 1990.

[5] D. Kahaner, C. Moler, and S. Nash, Numerical Methods and Software, New Jersey; Prentice Hall, 1989, ch. 4.