Conversion and control of an all-terrain vehicle for use as an autonomous mobile robot

John S. Jacob\textsuperscript{a}, Robert W. Gunderson\textsuperscript{b}, R. Rees Fullmer\textsuperscript{a}

\textsuperscript{a}Department of Mechanical and Aerospace Engineering, Utah State University.
\textsuperscript{b}Department of Electrical and Computer Engineering, Utah State University.

ABSTRACT
A systematic approach to ground vehicle automation is presented, combining low-level controls, trajectory generation and closed-loop path correction in an integrated system. Development of cooperative robotics for precision agriculture at USU required the automation of a full-scale motorized vehicle. The Triton Predator 8-wheeled skid-steering all-terrain vehicle was selected for the project based on its ability to maneuver precisely and the simplicity of controlling the hydrostatic drivetrain. Low-level control was achieved by fitting an actuator on the engine throttle, actuators for the left and right drive controls, encoders on the left and right drive shafts to measure wheel speeds, and a signal pick-off on the alternator for measuring engine speed. Closed loop control maintains a desired engine speed and tracks left and right wheel speeds commands. A trajectory generator produces the wheel speed commands needed to steer the vehicle through a predetermined set of map coordinates. A planar trajectory through the points is computed by fitting a two dimensional cubic spline over each path segment while enforcing initial and final orientation constraints at segment endpoints. Acceleration and velocity profiles are computed for each trajectory segment, with the velocity over each segment dependant on turning radius. Left and right wheel speed setpoints are obtained by combining velocity and path curvature for each low-level timestep. The path correction algorithm uses GPS position and compass orientation information to adjust the wheel speed setpoints according the “crosstrack” and “downtrack” errors and heading error. Nonlinear models of the engine and the skid-steering vehicle/ground interaction were developed for testing the integrated system in simulation. These tests lead to several key design improvements which assisted final implementation on the vehicle.

Keywords: automated vehicle, steering control, path correction, engine control

1. INTRODUCTION
The need for large-scale robotic vehicles arises in the exploration of hazardous sites as well as in the emerging field of precision agriculture. Both applications require a vehicle that can operate continuously for several hours, cover large distances at speeds of up to 20 mph, carry a payload of several hundred pounds, and drive at precisely controlled speeds. To meet these needs, the Center for Self-Organizing and Intelligent Systems (CSOIS) at Utah State University has procured two Triton Predator all-terrain vehicles with the intention of converting them to driverless operation for use as autonomous robotic vehicles. In addition to meeting the basic criteria stated above, the Triton Predator was selected for its payload flexibility, its stability on a variety of terrain, its ability to perform point turns, and the estimated ease of automating all onboard functions.

In this work, a control architecture is proposed for the conversion of a motorized all-terrain vehicle into an autonomous robotic vehicle, and the application of this approach to the Triton Predator ATV is outlined.

1.1 Description of the vehicle
The Triton Predator shown in Figure 1 is an 8-wheeled, skid-steering, amphibious all-terrain vehicle powered by a 25 HP v-twin gasoline engine. It is capable of operating for up to five hours on one fueling and has a top speed of about 25 mile per hour. Speeds as low as a few inches per second can also be sustained indefinitely. Payload capacity is about 600 pounds, and a variety of implements can be installed in its cargo bed.
The drivetrain is hydraulic, consisting of two variable-displacement pumps driven in tandem by the engine output shaft and two fixed-displacement hydrostatic motors driving the left and right rear axles, respectively. The four wheels on each side of the vehicle are linked together by drive chains. Each pump-motor combination drives one side of the vehicle, resulting in independent control of the left and right sets of wheels. The vehicle is operated by squeezing the throttle hand lever for engine power and moving the control levers (called “canes”) forward for forward motion and back to stop or reverse. Steering is accomplished by moving the levers counter to each other. The lever positions correspond to wheel speed, so care must be taken to avoid moving the control levers too quickly and stalling the engine. The levers have a dead zone at the neutral position of less than 1 cm width (1° angular width), and must be in this position before the engine is started.

The variable displacement pumps behave like a continuously variable transmission (CVT), allowing the vehicle to be driven at any ratio of wheel speed to engine speed. The ratio of the fixed drive motor displacement to the variable pump displacement determines the wheel speed to engine speed ratio. This displacement ratio is set by the position of each pump's swash plate control lever, which is connected to the control canes in the operator's compartment. The left and right wheels may also be driven at continuously differing speeds, allowing turns of any radius.

Conversion to driverless operation is relatively easy for this vehicle, requiring only one throttle position actuator and two drive control cane position actuators. The hydraulic CVT drive eliminates the need for clutch, gearshift and brake actuators, which enormously simplifies remote operation. The sensors required are an engine tachometer and two wheel tachometers. Additionally, a Global Positioning Satellite (GPS) receiver and an electronic compass will be installed on the vehicle to provide position and orientation feedback to the control system.

1.2 General approach

As a robotic vehicle, the Triton Predator will be required to follow a path described by a series of waypoints and stop at a goal point with a given orientation. Although the vehicle is capable of point-turns, they will not be used to re-orient the vehicle prior to driving or after arriving at a goal point. The accuracy required will not be more than one vehicle width (about four feet), and position corrections after coming to a halt at goal points will not be permitted.

To meet these design goals, a general approach is taken in which the vehicle is first automated through the addition of actuators and low-level feedback control loops. Then, high-level robotics controls are added which perform trajectory generation and path correction functions. The advantages of this division of high and low-level control are 1) the low-level control loops are relatively simple to design and permit the dual-use of remote operation through telepresence; 2) the high-level controls are sufficiently generalized as to be adaptable to other automated vehicles.

The low-level controls consist of three feedback loops around the three components of the drive system: the engine throttle and the two swash plate levers on the hydraulic pumps. Their purpose is to ensure that the wheel speeds track the desired wheel speeds regardless of disturbances such as variations in terrain or payload. The high-level controls combine the path command (a series of waypoints and a final goal point) and sensor data (position and orientation) to produce a sequence of low-level setpoints which moves the vehicle smoothly through the waypoints and brings it to a stop at the goal point. High-level control consists of two major components: trajectory generation and path correction. Trajectory generation develops a time-sequence of position, heading, curvature and velocity based on command waypoints. Each of these values is at least C2 continuous and provides the vehicle's desired position, heading angle, velocity, and turning rate for each moment in time. Path correction monitors the inevitable deviations from the path and adjusts the vehicle's velocity and turning rate to minimize position and orientation error.
The command velocity and turning rate are converted into low-level commands appropriate for the vehicle, which in this case are left wheel speed and right wheel speed.
2. LOW-LEVEL CONTROL AND VEHICLE AUTOMATION

The first step in converting the vehicle into a mobile robot is automating all onboard functions and placing them under closed-loop control. The task is simplified for the vehicle considered here since all drive functions (accelerating, braking and turning) are controlled through the two canes connected to the hydraulic pumps, and the speed is controlled by the engine throttle. Thus, complete automation of the vehicle is accomplished with the addition of just three feedback loops, two of which are identical, and their required sensors and actuators.

2.1 Decoupling the feedback loops

The automated Predator drivetrain is shown schematically in Figure 2. It consists of the engine with a closed-loop speed controller and two sets of hydraulic pump-motor systems each with a closed-loop wheel speed controller. The three "independent" control loops are found to be actually strongly coupled to one another, as is more clearly shown in the control system block diagram in Figure 3. The hydraulic pumps act as a load disturbance on the engine since they must overcome inertia and drag on the vehicle. The engine acts as a speed disturbance to both pumps, since the wheel speed is directly proportional to engine speed. Fortunately, only a slight change to the low-level control architecture is required to decouple the three systems.

Requiring the drive control loops to track a given absolute wheel speed places wheel speed directly under the control of two competing controllers. Instead, the drive controls should only be required to track the wheel speed to engine speed ratio. When one considers that this ratio is the quantity directly influenced by the swash lever to which the actuator is connected, it seems a natural choice. This results in the complete decoupling of the three systems, as shown in Figure 3. Also, since the drive ratio is not a dynamic physical quantity in the sense of having inertia, dampening forces and restoring forces influencing its magnitude, the drive controllers do not need to have a tuned dynamic response. The drive ratio is, to a good approximation, statically determined by the swash lever position, allowing the drive controllers to be something no more complicated than a one-dimensional lookup table.

It may be argued that the engine speed controller is not necessary, and that insistence on one forces the issue of the coupled systems. In fact, relying on the drive controllers exclusively to govern the speeds of the wheels results in incorrect responses to terrain variation disturbances. This problem provides another compelling reason for decoupling the low-level control loops in the
manner described above. When the Predator encounters a hill, the added load reduces the vehicle's speed. Wheel speed drive controllers would respond by moving the swash levers toward higher displacement, since this should result in higher wheel speed and compensate for the disturbance. Actually, this is analogous to shifting an automobile into a higher gear for driving up a steep hill, and is not the correct response. First-time operators of the Predator invariably stall the vehicle on the first slope they encounter, because the operation of the drive controls is somewhat counterintuitive. Moving the controls forward results in less drive force rather than more, since it increases the wheel speed to engine speed "gear" ratio. The correct response is to hold the drive controls steady and apply more engine throttle. On a very steep hill, the drive controls must sometimes even be moved backwards for the vehicle to move faster uphill.

It is apparent that no simple, linear wheel speed controller would be able to respond appropriately to such disturbances. On the other hand, drive ratio controllers would have exactly the right response, allowing the engine speed controller alone to compensate for all such disturbances. On very steep hills, if the engine were unable to maintain a set speed, the speed error could initiate the setting of a higher engine speed setpoint. This change affects the drive ratio controllers by causing them to immediately "downshift" to a lower ratio setting, since the desired wheel speed would then be divided by a larger desired engine speed.

2.2 Implementing the low-level feedback loops

It is beyond the scope of this paper to describe in detail the design of the three feedback control loops shown in Figure 1. It suffices to mention that the engine controller requires an inverse model linearization for its performance to be uniform over the full range of engine operating points (speed and load). The model is based on the empirical torque-RPM curve at full throttle combined with the approximate effects of throttle closure on the output torque. Then, a linear PI controller is designed classically to settle the engine speed in less than 1 second with no steady-state error and less than 5% overshoot.

The drive controllers, similarly, are incidental to this paper. It is worth mentioning, though, that knowledge of the appropriate swash lever position for a given desired drive ratio is not available a priori. Therefore, an adaptive or self-assembling lookup table is desirable for this application, and many numerical techniques are available that would work well.

With low-level controls in place, the Predator is considered an automated vehicle, and may be operated remotely. When the vehicle is made autonomous through the addition of robotic controls, the ability to operate the vehicle remotely through a human operator, or even to operate the vehicle manually, is still desirable and should be retained.

3. HIGH-LEVEL CONTROLS AND ROBOTICS

The automated vehicle is able to respond to commands such as "move forward," "stop" and "turn." A robotic vehicle requires a higher level of autonomy, and must be able to position and orient itself on command without operator assistance or intervention. The autonomous vehicle must be able to move through a set of command waypoints and come to a stop at the goal point with the desired orientation.

The high-level controls proposed to achieve robotic autonomy of the vehicle are shown schematically in Figure 4. They consist of a command processor, a trajectory generator, a path corrector, and a kinematic model of the vehicle. The function of the command processor is simply to set the engine speed based on whether the vehicle is waiting for a new command or in the process of moving to a goal point. It also passes command waypoints sequentially along to the trajectory generator. The trajectory generator forms a smooth path connecting the waypoints and determines the path position, heading,
curvature, and velocity for each timestep between waypoints. The path corrector compares the vehicle's current actual position with that determined by the trajectory generator for the same timestep. The resulting position error vector and heading error are used to determine adjustments to the velocity and path curvature determined by the trajectory generator. These adjustments constitute closed loop control over the position and orientation of the vehicle. Finally, the corrected velocity and path curvature must be converted into left and right wheel speed setpoints for the low level controls through a model of the vehicle turning kinematics. This requires knowledge of how the heading rate of the vehicle is related to wheel speed difference, and depends on parameters such as the transverse spacing between the wheels.

3.1 Trajectory generation

Connecting the command waypoints through a sequence of low-level commands is the purpose of trajectory generation. Specifically, the velocity and the turning rate or path curvature are needed, where only a series of coordinates are given. The first step in bridging this gap is to connect the waypoints with a smooth, continuous function. Then, heading angle and path curvature can be derived from this function. These quantities should also be smooth and continuous. Finally, a velocity profile is constructed based on a trapezoidal acceleration profile.

The path connecting waypoints is obtained by creating a two dimensional cubic spline between three consecutive waypoints. One assumes a piecewise cubic polynomial function for each segment for the X and the Y coordinates of the waypoints as follows:

\[ X(s) = A_1 s^3 + B_1 s^2 + C_1 s + D_1 \]
\[ Y(s) = A_2 s^3 + B_2 s^2 + C_2 s + D_2 \]

The polynomials are functions of a parameter set equal to zero, one, and two at the first, second and third consecutive waypoints considered, respectively. To determine the polynomial coefficients, continuity conditions are enforced at the midpoint, value conditions are enforced at the three waypoints, and derivative conditions are enforced at the endpoints. The endpoint derivative conditions relate to the initial and final orientation of the vehicle. The coefficients are determined analytically as functions of the waypoint coordinates and the initial and final orientations by solving a third order linear system of equations that result from enforcing the above conditions. A different set of coefficients is obtained for \( 0 \leq s < 1 \) and \( 1 \leq s < 2 \).

Distance along the path is obtained by numerically integrating the following expression:

\[ \frac{dD}{ds} = \sqrt{x'(s)^2 + y'(s)^2} \]

The curvature of any point on the path is obtained from the expression

\[ \kappa = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}} \]

The heading, or vehicle orientation, at any point on the path is obtained from

\[ \theta_s = \text{atan2}(y'(s), x'(s)) \]

In this manner, the position, curvature, heading and distance of the path are provided as functions of the path parameter "s". To complete trajectory generation, it is necessary to have these quantities as well as velocity as functions of time. This is accomplished by creating a piecewise first order continuous function representing acceleration as a function of time. Acceleration is then integrated twice to produce velocity and distance as functions of time. Distance is then correlated to distance as a function of the parameter "s" obtained by integrating equation 2. Finally, a lookup function is created correlating "s" with time and vice versa. Creation of the velocity profile is now presented in detail.

The vehicle is expected to drive continuously through path waypoints, stopping only at the last one. The velocity of the vehicle may be determined by the type of mission, the terrain, the curvature of the path, or other criteria, and may vary from one path segment to the next. The initial velocity over one path segment is obtained by measuring the actual vehicle velocity at that point.
(zero for the first segment of a mission), and the final velocity is determined by one of the criteria listed above, or is set to zero for the final path segment.

To avoid placing step load disturbances on the engine speed controller, the acceleration is defined to be first order continuous. For each path segment the acceleration is described with a trapezoidal profile: increasing linearly from zero for the first time interval, holding steady at the maximum allowable value for the second interval, then decreasing to back to zero. The rate of increase, the total time and the time intervals are initially unknown. The piecewise continuous acceleration profile is integrated once and the application of continuity and endpoint conditions to the results produces a velocity profile for the same time intervals. The process is repeated once more, resulting in distance as a function of time. The application of all continuity conditions, initial and final velocity conditions, maximum acceleration, total path segment distance, and the requirement that all time segments be positive and real results in a determination of the initially unknown profile parameters. The analytical results are easily placed in a subroutine which returns acceleration, velocity and distance given the four conditions listed above and the time of interest. Another useful result is the total transit time for the path segment.

Distance as a function of time "t" is then compared to distance as a function of parameter "s", resulting in a lookup table for s(t). For a given time of interest, it is then possible to obtain the position of the vehicle from X(s) and Y(s), the curvature of the path at that point from equation 3, and the desired orientation angle or heading from equation 4. The desired velocity at that point in time is also available, completing the vehicle trajectory.

It should be noted here that the purpose of requiring path curvature is to steer the vehicle. For a vehicle with a steering wheel, there is a one-to-one correspondence between the steering angle and the curvature of the trajectory. For this vehicle, and other track-steering or skid-steering vehicles, the yaw rate or turning rate is of interest, and is related to the difference between the wheel speeds on opposite sides of the vehicle. Yaw rate is related to path curvature $\kappa$ and tangential velocity $V$ by the expression

$$\dot{\theta} = \kappa \cdot V.$$

The temporal components of the trajectory, velocity and curvature, are used to create low-level command sequences that have the best chance of moving the vehicle through the waypoints and bringing it to a halt at the goal point with the desired orientation. The spatial components of the trajectory, position and orientation, are used by path correction to keep the mobile robot close to the desired path, ensuring that it arrives at the desired location with the required accuracy.

### 3.2 Path correction

Since the vehicle's function is to accurately position and orient itself, its position and orientation should be placed under closed-loop control. This is the purpose of the path corrector, to be described in detail here. It compares the actual position and orientation as obtained through a GPS receiver and an electronic compass to the desired position and orientation obtained from the trajectory generator. A position error vector and a heading error are obtained. The position error vector is resolved into perpendicular components aligned with and perpendicular to the current place on the desired path. The aligned component of the error vector is called "downtrack error" and the perpendicular component "crosstrack error."

The presence of a positive downtrack error indicates the vehicle is ahead of the trajectory, and the speed should be reduced. Corrections to the desired velocity are made proportionally to the downtrack error in a linear control law. The presence of a crosstrack error indicates that the vehicle did not follow the desired curvature of the path, and corrections to the steering must be made. Curvature is negative for clockwise-rotating motion along the path, and positive for counter-clockwise. Similarly, crosstrack error is positive to the left of the desired path and negative to the right. Thus, a linear control law can be written which makes corrections to the curvature proportionally to the crosstrack error. Heading error is usually a precursor to crosstrack error, and also requires a correction to the curvature. A term proportional to the heading error is therefore added to the curvature correction law.
3.3 Vehicle steering kinematics

The corrected velocity and curvature trajectories must be converted into left and right wheel speeds for the low-level controls. This conversion is made by considering the circular arcs described by the left and right wheels during a turn. The difference in wheel speed is related to the arc radius, and therefore to the curvature, through the width of the vehicle "w":

\[ V_r - V_l = \frac{wV}{r} = w\kappa V. \]

The tangential velocity \( V \) is the average of the wheel speeds, and is also the centerline velocity during a turn. Solving for left and right wheel speeds results in the following expressions:

\[ V_l = V - 2w\kappa V \]
\[ V_r = V + 2w\kappa V \]

During a turn, the leading and trailing wheels are actually dragged transversely across the path. The path curvature described by the vehicle during a turn will be less in magnitude than predicted by solving equation 6 for \( \kappa \), given the left and right wheel speeds. This steering deadband can be compensated for in the determination of left and right wheel speeds by augmenting the absolute value of the \( 2w\kappa V \) terms in equation 7. The amount of augmentation, however, must be determined empirically and may depend upon the terrain surface, vehicle payload and distribution, tire pressure and wear, and other variables. The steering kinematics equations do not have to be perfect, however, since the vehicle's position is ultimately under closed-loop control.

4. SIMULATIONS

The trajectory generator and path corrector was tested by a simulation of the vehicle. This shows whether the waypoints are connected by a suitable trajectory, whether the correct low-level commands are created from that trajectory, and whether the path corrector can make the necessary corrections to an errant vehicle. The simulation does not show how this system of robotic control will work on the real vehicle, since the vehicle model is an idealization. Instead, the reliability of the algorithms is verified, their function is illustrated, and insights into possible areas of improvement may be gained.

4.1 Modeling the Vehicle

Figure 5 shows a block diagram of the vehicle simulation. Its purpose is to numerically determine the vehicle's position and orientation in the X-Y plane over time given a sequence of left and right wheel speeds. Equation 7 can be inverted to yield forward velocity and yaw rate in terms of the left and right wheel velocities resulting in the following expressions:

\[ V = \frac{V_r + V_l}{2} \]
\[ \dot{\theta} = \frac{V_r - V_l}{2w} \cdot \frac{d}{w} \text{sign}(V_r - V_l). \]

In this expression, "d" is the yaw rate deadband width discussed above in 3.3. Yaw rate is integrated over time to produce heading angle. Heading is defined as radians counterclockwise from the positive X axis, so the positive X and Y axes are defined as pointing along heading 0 and \( \frac{\pi}{4} \) respectively.

The vehicle's response to forward velocity from the wheels (see equation 8) is modeled by the Laplace transform transfer function

\[ T(s) = \frac{1}{\tau s + 1}. \]

The parameter \( \tau \) depends on the terrain surface. Hard, rough surfaces result in the vehicle's speed equaling the wheel speed almost immediately. On loose or slippery surfaces the vehicle's speed may take longer to equal wheel speed. A value for \( \tau \) of 0.0625 gives the vehicle a 1/4 second response time to a unit step wheel speed change.
Forward vehicle velocity is resolved into X and Y components by the cosine and sine respectively of the heading angle. The X and Y velocity components are then integrated separately to yield position on the X-Y plane.

The vehicle simulation must also estimate the torque load on the engine. Engine load is based on vehicle acceleration, vehicle mass, forward velocity and uncorrected yaw rate. For simplicity, and to avoid numerical differentiation wherever possible, it is assumed that $\dot{\theta} = 0$, and that load due to rotational inertia about the vehicle's vertical axis is negligible. Linear acceleration must be determined numerically by differentiating forward velocity. A side benefit of the transfer function in equation 9 is that it eliminates numerical overflow caused by the differentiation of discrete numerical jumps in the simulation.

### 4.2 Modeling the Closed-Loop System

Figure 6 shows the complete, closed-loop simulation block diagram of the Predator control system. It incorporates the vehicle simulation in Figure 5 as a sub-system, as well as the engine control system, the hydraulic drive subsystem, and the feedback sensors, GPS and compass, modeled as sample-and-hold units with a single timestep delay and small amounts of noise added.

The Trajectory Generation/Path Correction subsystem contains the algorithms described in 3.1 and 3.2 above. The block calls a shell program that contains a set of command waypoints and passes data between the control subroutine and the simulation. The control subroutine computes left and right speed setpoints and sets the engine speed.

The GPS sampling period is 0.200 seconds, which is the rate at which the GPS receiver updates position information. The low-level sampling period is 0.050 seconds. The control system shell operates each low-level sampling period, but only accesses the trajectory generation and path correction subroutines once every four iterations, or once for each GPS sampling period. The control system returns four sets of setpoints, so that the high-level controllers receive a new setpoint each timestep. It is felt that the low-level controls should sample as frequently as the computational loads of their algorithms will allow, but that there is no need for the high-level controllers to sample any more frequently than the GPS rate, since their computational loads are high.

The simulation is performed by Simulink (TM) using a fifth-order Runge-Kutta variable timestep integration formula. The maximum and minimum step sizes are set to 1 and .0001 seconds respectively, and the step size control tolerance is .0001.
4.3 Closed-Loop Simulations

Figure 7 shows an X-Y field plot indicating the path taken by the simulated vehicle. Each waypoint is marked on the plot by an asterisk. The small circles indicate time checkpoints, used to visually evaluate cross-track and down-track errors at specific times. The vehicle passes through or close to each waypoint and comes to a stop within two inches of the goal point. The final heading is -87°, close to the prescribed -90° and well within the compass noise. The jagged trace of the vehicle generated by the output of the simulated GPS sensor indicates the amount of noise added through the sensor. The abrupt jumps in the desired path show how subsequent path segments are computed from the vehicle's current position to the next waypoints after a waypoint has been passed.

Repeated simulations with a variety of path commands were performed to determine suitable path correction gains. It was found that down-track and heading gains were the most important ones in determining the vehicle's tracking performance. The down-track error gain determines how well the vehicle "keeps up" with the time-scheduled trajectory. The heading error gain is crucial to keeping the vehicle on the path, more so than cross-track gain, which caused repeating weaving about the desired path when excessively high gains were used.

The trajectory generator used a maximum acceleration of 1/4 g, or 8 ft/sec². The maximum load on the engine (about 20 ft-lbs of torque) occurred during the initial acceleration between the first two waypoints, and resulted in the throttle opening gradually to about 50% for a few seconds. As indicated in Figure 8, the engine speed was held at the prescribed 2200 RPM in spite of the fluctuating load, until the vehicle came to rest at the goal point. There, the command processor properly reduced the engine speed setpoint to an idle at 800 RPM.
5. SUMMARY AND CONCLUSIONS

The function of a mobile robot is to position and orient itself autonomously on command. As with manipulator robots, such devices can open a wide variety of possibilities where previously the need for human attention has been a limiting factor. Though remotely-guided vehicles are not new, the need for greater autonomy in remotely operated vehicles is increasing. The difficulty lies in bridging an information gap between the known goal point location and an evolution of wheel speeds over time that will bring the vehicle to that point. Further, it is necessary for the autonomous vehicle to control interrelated onboard mechanical systems in addition to maintaining closed-loop control over its own position and heading. It was the purpose of this work to describe a control architecture that addresses all of these needs in a unified way, and produce a truly robotic and truly autonomous vehicle control architecture.

The control architecture chosen and developed for this vehicle divides the system into low-level and high-level controls. The low-level controls were tailored to the specific vehicle, both to meet its unique requirements and to take full advantage of its capabilities. The high-level controls relied less on specific information about the vehicle, and thus are easily adaptable to other platforms. Simulations showed that the algorithms functioned as desired and were successful in meeting their design criteria. An integration of the complete system was also accomplished, demonstrating the utility of this approach.

Areas of possible further development are the addition of self-adaptive controls for path correction, kinematics and engine control, and the integration of obstacle avoidance into the command processor. Velocity and curvature setpoints were scheduled to the low-level controls on the basis of elapsed time. A more robust approach for future work may be scheduling the setpoints based instead on distance traveled. In such a scheme, turns would be initiated and completed at the correct locations, even if the vehicle becomes delayed.

The value of this work for future efforts lies in its demonstration of a fully integrated vehicle trajectory, and the separation high and low-level controls along the lines of general vehicle robotics and closed-loop control of specific mechanical drive systems. It suggests a clear approach to the conversion of almost any vehicle to robotic applications, regardless of scale or intended application.