An Obstacle-Edging Reflex for an Autonomous Lawnmower

Kathryn A. Daltorio, Amaury D. Rolin, Jonathan A. Beno, Bradley E. Hughes, Alexander Schepelmann, Michael S. Branicky, Roger D. Quinn
Departments of Mechanical & Electrical Engineering
Case Western Reserve University
Cleveland, Ohio, USA
rdq@case.edu

James M. Green
MTD Products, Inc
Cleveland, Ohio, USA
jim.green @mtdproducts.com

Abstract—We developed a controller that allows our prototype lawnmower, CWRU Cutter, 1st place winner of the 2009 ION Autonomous Lawnmower Competition, to follow pre-defined paths and reflexively edge around obstacles before returning to the path. CWRU Cutter is equipped with localization sensors (GPS) and obstacle detecting sensors (LIDAR and camera).

The best human lawnmower drivers mow parallel lines, interrupted smoothly as necessary to follow the contours of obstacles. To get this quality of cut from an autonomous mower, we developed a low-level obstacle avoidance reflex that keeps the robot close to obstacles when required to circumvent them. Our method requires less processing than common planning solutions for two reasons. (1) The environment is represented by a 1-dimensional Polar Freespace array rather than a 3-dimensional (x,y,theta) configuration space. (2) The robot only plans on the local environment recently perceived rather than planning all the way to the goal. Other methods for local obstacle avoidance often assume cylindrical and/or holonomic robots. These are not good assumptions for our rectangular, wheeled lawnmower. Instead, we pre-calculate the polar ranges associated with the robot's footprint and the areas crossed by the footprint during constant curvature stops. These swept area ranges are easily compared with the Polar Freespace array that represents the environment.

First, the robot uses GPS to generate initial velocity and angular velocity commands that steer the robot to a path of parallel lines covering the field to be mown. Data from a camera and LIDAR go into a 1-dimensional array of ranges (the Polar Freespace) that represents the local environment. If the initial command puts the robot on a collision course, the swept area ranges will be greater than the Polar Freespace ranges. To avoid obstacles, a reflex searches the velocity – angular-velocity space to find a safe command reachable from the current speed and as close as possible to the initial path command.

Key reflex search parameters are examined in a MATLAB simulation assuming perfect localization and LIDAR data. A velocity-dependent extension factor is calculated that allows obstacle avoidance as opposed to halting in front of obstacles. The search resolution is adjusted to trade-off calculation speed and clearance. For example, by checking an average of 5 velocity/ angular-velocity command pairs per time-step (maximum of 14 checked commands per time-step) our 66cm by 100cm robot skirts a 2m-diameter obstacle with 1.3cm clearance. To compare, if we had used a previously published algorithm, such as Curvature-Velocity Method, that assumes the footprint was circular, the robot would not be able to pass any nearer than 54cm to the side of obstacles.

We tested this controller on our robot, CWRU Cutter, with the blades on. The software was written in LABVIEW and running on an NI-cRIO at 10 Hz. We observed that the mower was able to edge boxes, soccer balls, and picket fences with side clearance of a few centimeters, and then smoothly return to following parallel paths after passing the obstacle. When rapid changes in the environment occur, such as a person walking in front of the mower, the robot stops until the environment stabilizes. In the future this method could be extended to allow the robot to back-up and try again with relaxed clearances if the robot becomes stuck in a tight corner. Alternatively, these reflexes could be adapted for low-level safety checking on other non-holonomic non-cylindrical robots operating in cluttered environments.

Fig. 1. The autonomous lawnmower CWRU Cutter has a 66cm by 100cm footprint. It is equipped with LIDAR, camera, GPS, IMU, and wheel-encoders

Keywords-reflexive obstacle avoidance; domestic robot; hierarchial control

CWRU Cutter is sponsored by MTD, www.mtdproducts.com. Students were supported by a National Defense Science and Engineering Graduate (NDSEG) Fellowship, and several Case Western Reserve University SOURCE grants. The 2008 and 2009 annual robotic competitions were sponsored by the Institute of Navigation (ION).

I. INTRODUCTION

Autonomous lawnmowers currently available [1][2][3] cut random patterns, backing away from buried boundary wires or any obstacle encountered, much like indoor cleaning robots[4]. To achieve the aesthetic of straight parallel path lines and the safety of predictable paths, prototype lawnmowers [5][6][7] are being designed to use non-random paths but plan to stay far away from obstacles. With our prototype lawnmower, we are interested in safely cutting the grass close to trees, structures and landscaping, which requires a safe obstacle-edging behavior.

Many types of autonomous service robots require collision avoidance, which can be done with range finders such as sonar[8][9][10] and LIDAR[11][12][13]. Complete planning algorithms for obstacle avoidance such as RRT [14] or Probabilistic Roadmap [15] can require extensive processing for precision maneuvers in a partially-observable 3-dimensional (x,y,theta) configuration space. Other methods for local obstacle avoidance, such as Vector Field Histogram [16] or the Curvature-Velocity Method [17], require cylindrical and/or holonomic robots. For our rectangular, wheeled robot these are not good assumptions. Controlling for feedback on tactile sensors requires hard, smooth, unbroken obstacles [18] [19].

The Case Western Reserve University prototype autonomous lawnmower, CWRU Cutter (pronounced, "crew cutter"), a non-holonomic non-cylindrical vehicle with sensors for localization and range-finding (fig. 1), needs a simple real-time, deterministic control algorithm to edge as close to irregularly-shaped obstacles as possible without collision and without unnecessary deviation from linear paths. We wanted a reflexive controller that responds to the local, recently perceived environment. An egocentric 1D polar representation is preferred because the environment is perceived with range-finders, in addition to cameras, and because it allows us to avoid searching a 3-dimensional (x,y,theta) configuration space. Instead of making assumptions about the shape of the robot or the directions of motion, we pre-calculate the polar ranges associated with the robot's footprint and the areas crossed by the footprint during constant curvature stops. These swept area ranges are easily compared with the Polar Freespace array that represents the environment. In addition, the mower's environment is only partially observable but we wanted to limit the amount of internal environmental state represented. Table I. lists some established obstacle avoidance methods and shows how our edging controller fits a new combination of criteria.

We used a hierarchical reflex architecture [28] and extended [29] to write a path driver. The result is a new complete control system that runs in real time in LabVIEW on an NI cRIO to solve the autonomous lawnmowing problem posed by the 2008 ION robotic lawn mower competition. At this competition, CWRU Cutter, in its first year, placed third in the advanced dynamic competition by cutting straight paths and waiting if obstacles were encountered. Without obstacle avoidance, the mower's paths were limited by the accuracy of our absolute position estimates and we mowed only areas at least a meter away from stationary obstacles. Then we added a new obstacle avoidance strategy that considers the surroundings in polar coordinates like while guaranteeing safety of an arbitrary-shaped robot around irregular obstacles. Two safety reflexes, one that stops and one that veers left, allows the robot to mow within centimeters of irregular obstacles such as picket fences. These reflexes could be added to any other robot with accurate range and position sensors to protect against collisions, with no effect on existing safe behaviors. In the 2009 ION competition, CWRU Cutter won first place. The ability to safely maneuver close to obstacles while following predefined paths is important not only for lawnmowers but also to other future vehicles such as autonomous delivery systems, self-parking cars, and inconspicuous surveillance robots.

TABLE I. COMPARISION OF EXISTING CONTROL STRATEGIES

	Requirements							
					Handle:			
Methods	***Collision avoidance override goal-seeking only when necessary***	***Avoids calculating configuration space (e.g. x,y,θ)***	***Check clearances to allow close maneuvering***	***Take advantage of ego-centric polar coordinate system (preferred)***	***Non-holonomic motion***	***Arbitrary non-cylindrical robot shape***	***Dynamic obstacles (e.g. dog)***	***Discontinuous, irregular obstacles (e.g. fence)***
RRT/PRM Planning[a]	+		+		+	+		+
Potential Fields[b]		+			+		+	+
Highspeed Vehicular Control[c]	+				+			+
Integrated Method[d]	+		+		+	+	+	+
Transform to ARM[e]			+		+	+		+
Curvature-Velocity Method[f]	+	+	+		+			
VFH[g]	+	+		+				+
Tactile Control[h]		+	+		+			
Our Edging Controller	+	+	+	+	+	+	+	+

a. Full planning solutions [20] [21][22]

b. Velocity and direction can be controlled by active and repulsive virtual forces[23]

c. Reflexive terrain search level for controlling autonomous cars such as [24] [12] and [25]

d. Searches portions of 5 dimensional configuration space. [26].

e. Transform to Arc-Reachable Manifold is described in [27]. This is not an obstacle avoidance method, but a clever way of calculating configuration space by assuming arc-paths. Robot shapes made of straight lines or arcs can be used. Non-holonomic paths are approximated by arcs.

f. Divides the possible headings into regions bounded by arc-shaped collision paths. [17]

g. In the Vector Field Histogram method, a polar histogram represents the environment. The heading is determined by finding a 'valley' of sufficient threshold in the histogram [16]

h. In [18], a piano wire whisker allows wall following at 3cm. In [19] an active antenna is used for wall-following and corner tracking.

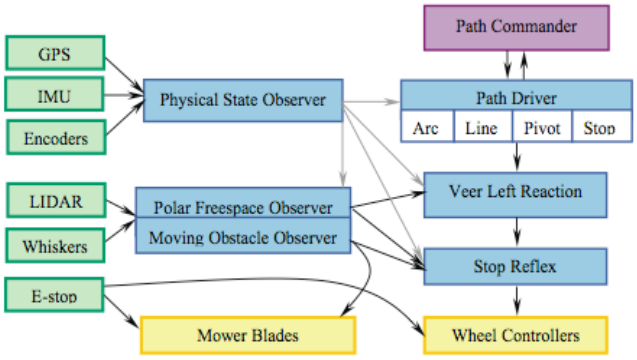


Fig. 2. Hierarchical obstacle-edging and path-following architecture.

II. Control of An Autonomous Mower

A. Problem Formulation

Speed and turning commands can be considered in the velocity-angular velocity plane, as in fig. 5. Given a maximum individual wheel-speed, we can determine a diamond-shaped boundary for acceptable (v,ω) pairs such that $|\omega \cdot track/2| < |v - v_{wheel}|$, where v and ω are the mower's velocity and angular velocity, respectively, v_{wheel} is the maximum desired wheel velocity, and the track is the distance between the two independently driven wheels. Acceptable commands are further constrained by traction acceleration limits, so a slew rate is enforced on v and ω. At every instant, a controller must select the v,ω) pair that will keep the robot moving along the path, deviating by the minimum required to avoid collision with obstacles.

B. Software Architecture

Our architecture, fig. 2., uses reflexive behaviors that are transparent to upper levels, except when immediate action is required [30] [28]. Thus, the robot will follow the nominal path, unless there is an obstacle in the path. If possible, the obstacle is avoided; otherwise, the robot stops.

A common alternative strategy is to have the robot plan paths to avoid any obstacles, which is done for non-holonomic vehicles in [22][12] and [5]. However, if during the time required for real-time re-planning, the robot develops an error in its absolute position or deviates from the path, the robot is susceptible to collisions. Since a reflex is based on recent relative information, it is more robust and asserts safety conditions quickly.

There are two main observers that process and filter the sensor data. The Physical State Observer determines the location, orientation, angular velocity, and velocity of the robot. To do this CWRU Cutter uses a Kalman filter [31][32] on data from wheel-encoders, a Global Positioning System (GPS), and an Inertial Measurement Unit (IMU). The second filter is the Polar Freespace Observer. Its purpose is to filter together any obstacle information, for example from laser range finders, sonar, or touch sensors, into a single array that represents the length of clear space 360° around the robot. So far, we've used LIDAR scans and camera data.

The physical state observer informs the path driver, which calculates (v,ω) to keep the robot on the predetermined path. Then reflexes check the (v,ω) pair with data from the polar freespace observer to determine if a collision will occur. If no collision is predicted, (v,ω) is commanded to the motor controllers. If the reflex detects a problem, a better (v,ω) is selected.

III. Polar Free-Space Observer

In order to maneuver precisely around nearby obstacles, the robot must know what space around it is clear of barriers, even any unobservable areas such as the blind spot behind the range sensor. One way to do this would be to use an occupancy grid, but this would require collision checks across a 3D (x,y,θ) configuration space. We reduce the environmental representation to a single 1D array. The Polar Freespace Observer, fig. 3, operates by observing the current freespace and filtering that with the previous, shifted freespace. The observable freespace consists of the body footprint plus the ranges observed by the range finders. In addition or instead of range finders, we have also extracted free ranges from camera data [33]. The shifted freespace is the previous freespace estimation, shifted according to the change in position and orientation of its center. In CWRU Cutter's case, this was at the center of the LIDAR. The location of the LIDAR is measured from the center of the GPS phase antenna, assuming the LIDAR is rigidly mounted to the chassis. The shift, S, is accomplished by converting each range from polar to Cartesian coordinates, adding the change in position to the result, and converting back to polar coordinates $\left(\phi^p, f^p\right)$. Ranges corresponding to angles ϕ spaced evenly around the robot (at the same resolution as the LIDAR scans) are then interpolated. Then the scans are rotated by the change in orientation.

$$\vec{\phi}_t^{\,p} = \operatorname{atan2}\left(\vec{f}_{t-1}\sin(\vec{\phi}_{t-1}) + \Delta y, \vec{f}_{t-1}\cos(\vec{\phi t}) + \Delta x\right) + \Delta\theta \qquad (1)$$

$$\vec{f}_t^{\,p} = \sqrt{(\vec{f}_{t-1}\sin(\vec{\phi}_{t-1}) + \Delta y)^2 + (\vec{f}_{t-1}\cos(\vec{\phi}_{t-1}) + \Delta x)^2} \qquad (2)$$

$$S(\vec{f}_{t-1}, \Delta x, \Delta y, \Delta\theta) = \text{Interpolation of } \vec{f}_t^{\,p}(\vec{\phi}_t^{\,p}) \text{ at } \vec{\phi}_{t-1} \qquad (3)$$

The observable freespace, a collection of ranges z, and the remembered freespace, a collection of ranges f_{t-1} that have been shifted by the amount the robot moved in the last timestep, are combined element by element according to their variances. The standard deviation, or square root of variance, of the new observation, σ_z, is small where the LIDAR can see or the whisker is activated, but large in the back where the body boundary is assumed because it cannot be observed by the sensors. The standard deviation of the shifted value, σ_{ft-1} is the standard deviation of the previous total scans, plus a factor based on the change in position. Thus at each angular increment around the body, the following update is performed at each timestep:

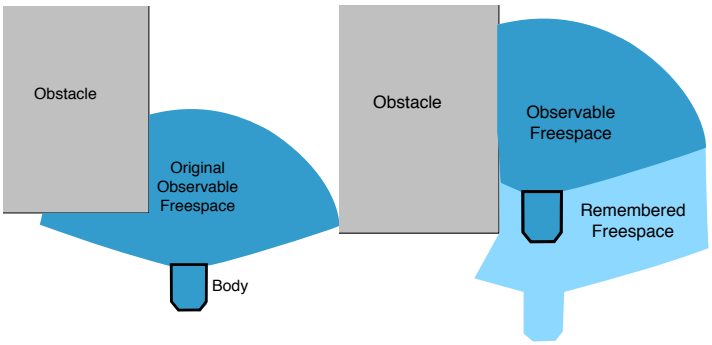


Fig. 3. When the mower starts up, the original polar freespace consists of the area swept out by the range sensors and the area occupied by the body. After the robot moves, the freespace should consist of the new observable freespace and the part of the previous freespace that can be captured in a singled-valued polar function with origin at the new front of the robot.

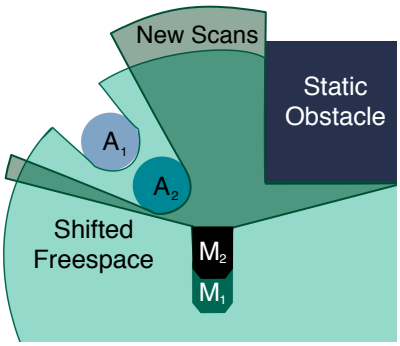


Fig. 4. The polar freespace observer may be used as a moving obstacle detector by comparing the expected position of the environment to the new scans. In this example, the range scans when the mower was at M_1 and the moving obstacle was at A_1 are shifted and compared to the scans observed when the mower is at M_2 and the moving obstacle is at A_2. There is a large difference in the magnitude near the moving obstacle but none near the static obstacle.

$$f_t = \frac{z \cdot \sigma^2_{f_{t-1}} + S(f_{t-1}, \Delta x, \Delta y, \Delta\theta) \cdot \sigma^2_z}{\sigma^2_z + \sigma^2_{f_{t-1}}} \quad (4)$$

$$\text{where } \sigma^2_f = \frac{\sigma^2_z \cdot \sigma^2_{f_{t-1}}}{\sigma^2_z + \sigma^2_{f_{t-1}}} \quad (5)$$

This results in a very accurate map in front of the robot where the LIDAR is mounted and a less accurate map in directions that have not been observed by LIDAR for several time steps. If the robot sits facing forward long enough, eventually the total freespace will converge to the observable freespace.

IV. Moving Obstacle Detector

Instead of tracking any moving obstacles and attempting to project their paths, we want the robot to wait for any moving elements in the environment to stand still or pass. The Polar Freespace Observer lends itself to the detection of unexpected changes in the perception of the environment, since it is already shifting the previous sensor values.

$$S(f_{i-1}, \Delta x, \Delta y, \Delta\theta) - z > \text{Threshold} \quad (6)$$

Where the threshold is related to the uncertainty of the motion and obstacle detection. Our threshold was equal to .5cm flat ground. Therefore the Moving Obstacle Detector, fig. 4, operates within the Polar Freespace Observer and compares the new scans with the shifted scans. If several adjacent scans detect motion over the threshold and include zs that are sufficiently close to the robot, a flag is sent to decelerate and wait a minute to resume. If motion is still detected the robot will keep waiting. If a dynamic obstacle holds still long enough, the robot will recognize it as static and go around it.

V. Obstacle-Avoiding Reflexes

Safe (v,ω) commands are those at which the vehicle can stop before impacting an obstacle. Any command for which each of the Swept Area ranges is less than the Polar Freespace range at its respective angle is safe. It has been demonstrated [34][35] that the common maze-solving heuristic – following a wall to the left or to the right – is sufficient to get around any finite dimensioned obstacle. Therefore we chose that our robot should turn left from the path as much as necessary to avoid the obstacle. The robot will take the longer way around the obstacle about half of the time, but the advantage is that the robot's behavior is very predictable. In future versions, left or right could be set by a higher level as a flag to modulate the low level control until that particular obstacle has been cleared. Without a flag, the robot could oscillate between trying to choose left or right as it passes a single obstacle.

A. Generating Swept Areas

Since we cannot assume a cylindrical robot footprint, a look-up table of swept areas was generated for representative (v,ω) couples at given maximum decelerations. For intermediate (v,ω) couples, the values were determined by two-dimensional linear interpolation.

First it was assumed the mower continued for two timesteps at the given (v,ω). This takes into account that by the time the mower has decided what command to use, the information is already one timestep old. The positions of the mower at these times are not included in the determination of the swept volume, because by the time actions take affect this area has already been covered. Allowing the swept area to be shifted in the direction of the proposed motion allows the robot to move away from obstacles that impinge slightly upon its current footprint.

The mower's position as it slowed with the maximum allowable deceleration along a constant curvature path was determined at every tenth of a timestep until the mower came to a stop. For each determined body position, the polar ranges of the body with respect to some origin fixed on the original body are determined In other words, we simulate what a laser range finder, kept at its original location, would see at each of the footprint boundary edges. The maximum polar value at each angle of every range was kept. This is a polar representation of the area the footprint would cover if it were stopping as fast as possible. This process could be extended to

three-dimensional shapes for robots that sense in multiple planes.

B. *Safe Stopping*

The stop reflex, fig. 5, is composed of three phases that check the commanded velocity and angular velocity before they reach the wheel-controllers.

In phase 1, speed and angular speed limits are enforced. These limits can be visualized in the velocity diamond in fig. 5. Each (v, ω) pair is a point on the plane. The legal (v, ω) pairs are within the shaded diamond defined by a maximum angular velocity (at zero forward speed) and a maximum linear velocity (with no angular velocity). If a velocity outside of this range is requested by software levels, the linear and angular velocities are scaled proportionally (constant curvature, ω/v) to the edge of the diamond. In phase 2, a slew rate is enforced. Only (v, ω) pairs within a box defined by maximum accelerations around the current pair are permitted. If the (v, ω) from phase 1 is not within the box, the closest value on the edge of the box is chosen.

In phase 3, the maximum deceleration is applied if the area swept by (v, ω) after phase 2 is not completely contained by the polar freespace or if a moving obstacle is detected. Since the resolution of both the freespace and the swept areas in polar coordinates are the same, this operation is simple regardless of the complexity of the footprint:

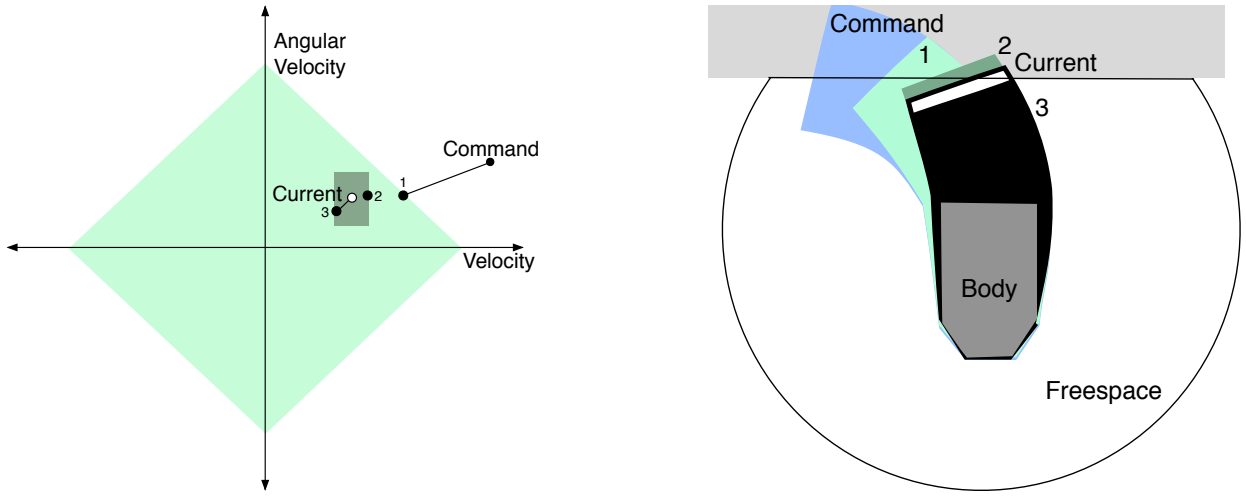


Fig. 5. The stop reflex consists of three checks on a $(v,\omega)_{Command}$ from a higher level. First, velocities are checked to make sure they are within the diamond of legal velocities. If not, a point along the boundary edge is chosen that has the same curvature, such as $(v,\omega)_1$. Second, $(v,\omega)_1$ is slew rated to $(v,\omega)_2$ insure there are no jerky motions. Third, if the swept volume corresponding to $(v,\omega)_2$ is not contained in the freespace, a decelerated speed with the same curvature $(v,\omega)_3$ is passed to the low level controls.

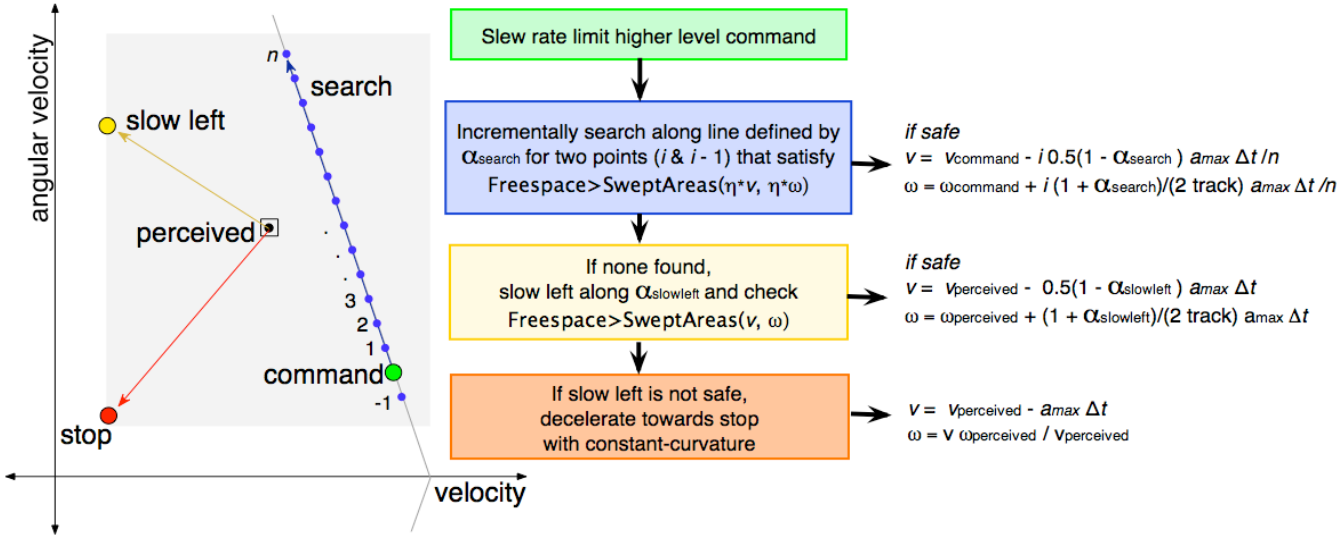


Fig. 6. The veer left reflex, as show in the v/ω plane. After the command (v,ω) from the path driver is slew-rated into the vicinity of the current perceived (v,ω), the command is incrementally shifted to find a safe command such that the projected swept areas are within the perceived freespace. First the original command is checked, then points along the search line with resolution determined by n, then a maximum deceleration slow left is considered, and finally if none of the above are safe, the robot begins a constant-curvature stop, as in the Stop Refelx.

$$f \geq \text{SweptAreas}\,(v, \omega) + m \quad \forall \phi \tag{7}$$

where η is a safety factor (we often used η = 1.2) and *m* is a safety margin. These parameters could be included in the footprint a priori, but adding them here allows on-the-fly adjustments, for example relaxing the safety if the robot gets stuck.

These checks ensure that from any (v, ω) sent to the wheel controllers, the vehicle can stop before collision under the following conditions: (1.) The freespace represented by the polar freespace observer is conservative (there are no obstacles closer than the scan output by the polar freespace) and the mower body definition used in the swept volumes is at least as large as the actual body. (2.) Real-time operation is sustained. (3.) The reflex starts before the obstacle is too close to decelerate from. (For example, in fig. 5, if an obstacle is in the black area 3, the maximum allowable deceleration will be insufficient to prevent collision.) This condition can be satisfied by starting the reflex when the mower is at zero velocity in a position that does not touch any obstacles. (4.) The robot is capable of exerting the requested deceleration and the deceleration rate is no less than that used in the swept area calculations. To ensure the feasibility of our deceleration rates, we tested the largest decelerations the robot can execute without slipping on grass or pitching forward.

C. *Veering Left Reflex*

Instead of decreasing both *v* and ω, proportionally to veer left, we search for a safe command, detailed in fig. 6. As demonstrated in fig. 7, in general, to turn requires more clearance than to stop. Thus to avoid backtracking, the turn should start before the stop starts. To turn with additional clearance, we can use an extension factor, η, such that further along the constant curvature arc is considered. Thus a command (v,ω) is safe if

$$f > \text{SweptAreas}(\eta v, \eta\omega) + m \quad \forall \phi \tag{8}$$

Where the choice of η is critical. If η is too small (Fig. 7b), the robot cannot pass the obstacle. If η is too large (Fig. 7f) the robot begins slowing down much too soon. Moreover, at lower velocities larger η is required to pass obstacles. By approximating the extension factor required for a simple turn, an expression for a velocity-dependent η^* can be developed based on the geometry of the robot and its turning characteristics.

1) *Turn Characterization*

A left turn requires increasing the curvature and may be accompanied by a speed decrease. For our differential steer robot, this means that the left wheel speed is decreased by an amount $a\Delta t$ and the right wheel speed is increased by $\alpha\, a\Delta t$ where *a*, the wheel acceleration, is determined by the resolution of the search and the physical limits of the robot. So the velocity command $(v_t\ \omega_t)$ at time *t* will be:

$$v_t = v_{t-1} - \tfrac{1}{2}(1-\alpha)\, a\Delta t \tag{9}$$

$$\omega_t = \omega_{t-1} + (\alpha+1)\, a\Delta t \,/\, \text{track} \tag{10}$$

where Δt is the timestep and track is the distance between the wheels. Thus, α parameterizes whether to turn without slowing down ($\alpha = 1$) or slow down without turning ($\alpha = -1$).

2) *Anticipating Turning Clearance*

The forward distance, *y*, crossed during a turn can be approximated as:

$$y = \max\left(\int_{t=0}^{t} v(t)\cos\big(\theta(t)\big)dt \right) \forall t \tag{11}$$

where

$$v(t) = v - \tfrac{1}{2}(1-\alpha)at \tag{12}$$

$$\theta(t) = \omega t - \tfrac{1}{2}(1+\alpha)at^2 \,/\, \text{track} \tag{13}$$

After substitution, the solution to this integral involves Fresnel Integrals, $C(x)$ and $S(x)$ which can be approximated as a ramp up to a constant :

$$C(x) = 0.77989 \min(x,1) \tag{14}$$

$$S(x) = 0.71397 \min(x\sqrt{2},1) \tag{15}$$

We found that assuming $\omega = 0$ in the following equations is a simplification that has a negligible affect on the robot's behavior. Then the maximum value, *y,* will be either when *v(t)* = 0, (when the robot comes to stop) at time, t_{stop}, or when the vehicle has turned completely to the left, when θ = π/2 at time, t_{turn}, whichever happens first.

$$t_{stop} = \frac{2v}{a(1-\alpha)} \tag{16}$$

$$y_{stop} = 0.78 v t_{stop} + 0.25 a(1-\alpha) t_{stop}^2 \tag{17}$$

Combining terms results in:

$$y_{stop} = \frac{2.56 v^2}{a(1-\alpha)} \tag{18}$$

Similarly for t_{turn}:

$$t_{turn} = \sqrt{\frac{\pi\,\text{track}}{a(\alpha+1)}} \tag{19}$$

$$y_{turn} = 1.38 v \sqrt{\frac{\text{track}}{a(\alpha+1)}} + \text{track}\,\frac{(\alpha-1)}{2(\alpha+1)} \tag{20}$$

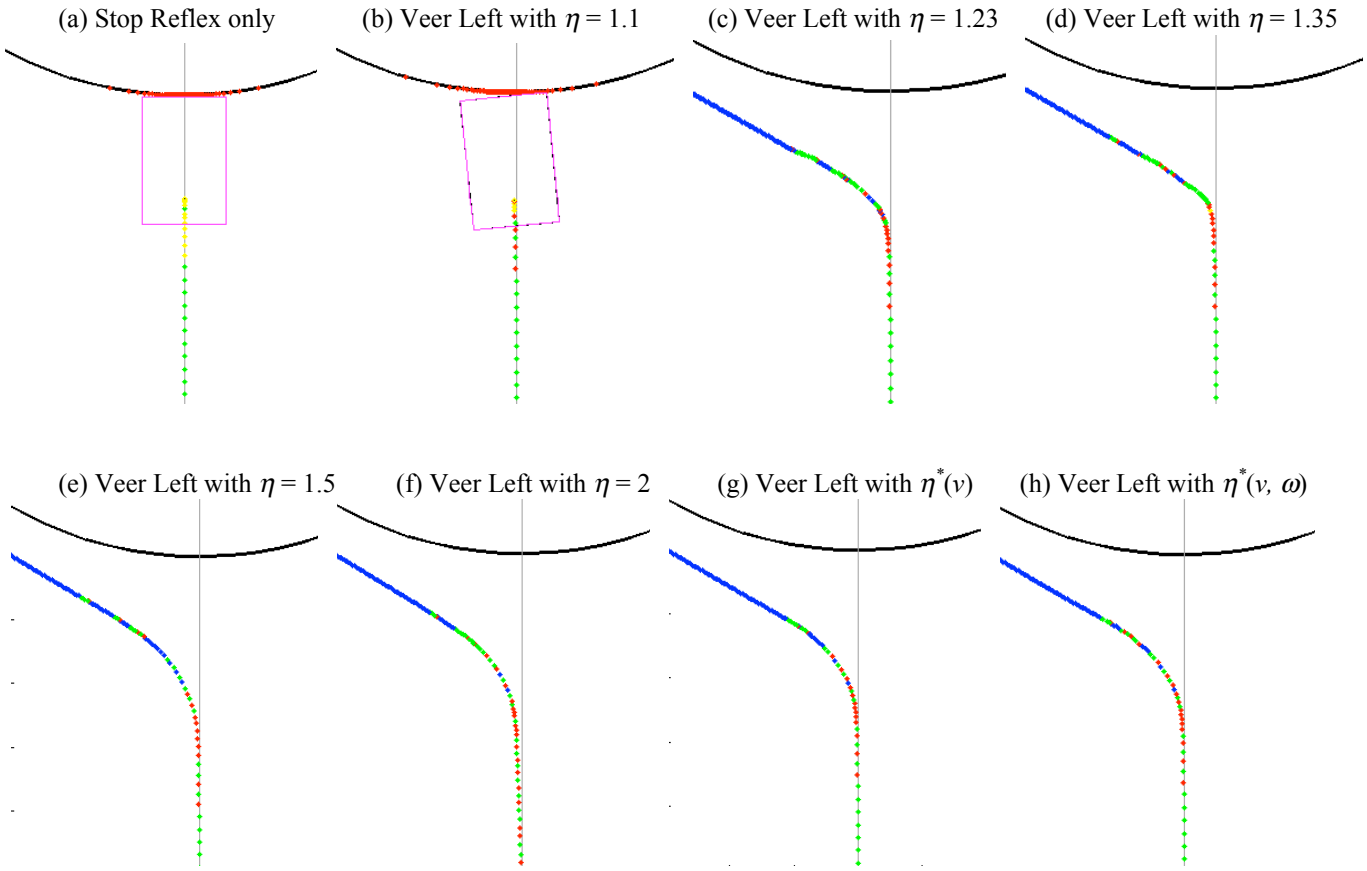


Fig. 7. Simulated trials of a robot with initial velocity 1m/s encountering an obstacle with radius 3m demonstrate the importance of using a good extension factor. The robot is outlined in (a) and (b), the heavy line is the obstacle, and the path of the robot is traced.
greendots: robot followed path command (reflexes considered path command safe)
blue dots: robot reflexively turns with α_{search}
red dots: robot turns with $\alpha_{slowleft}$
yellowdots: robot decelerates with constant-curvature

If the robot comes to a stop without turning (a) the robot will have to back up to make a left turn. If the extension factor is set to too small a constant (b), the robot can get stuck by turning too late. If the extension factor is too large, the robot is decelerated far before the obstacle (e-f). Using a variable extension factor η^* based on either v and ω (h) or v alone (g) eliminates the trial and error and provides a smoother curve. For these conditions, η^* ($v = 1$, $\omega = 0$, track = .6, α_{search} = .5, a_{max} =1) = 1.3405 (d), which turns out to be 9% greater than the minimum passable value of $\eta = 1.23$ (c). With these types of turning the tightest clearance of the body with the front of the obstacle is 2.5cm.

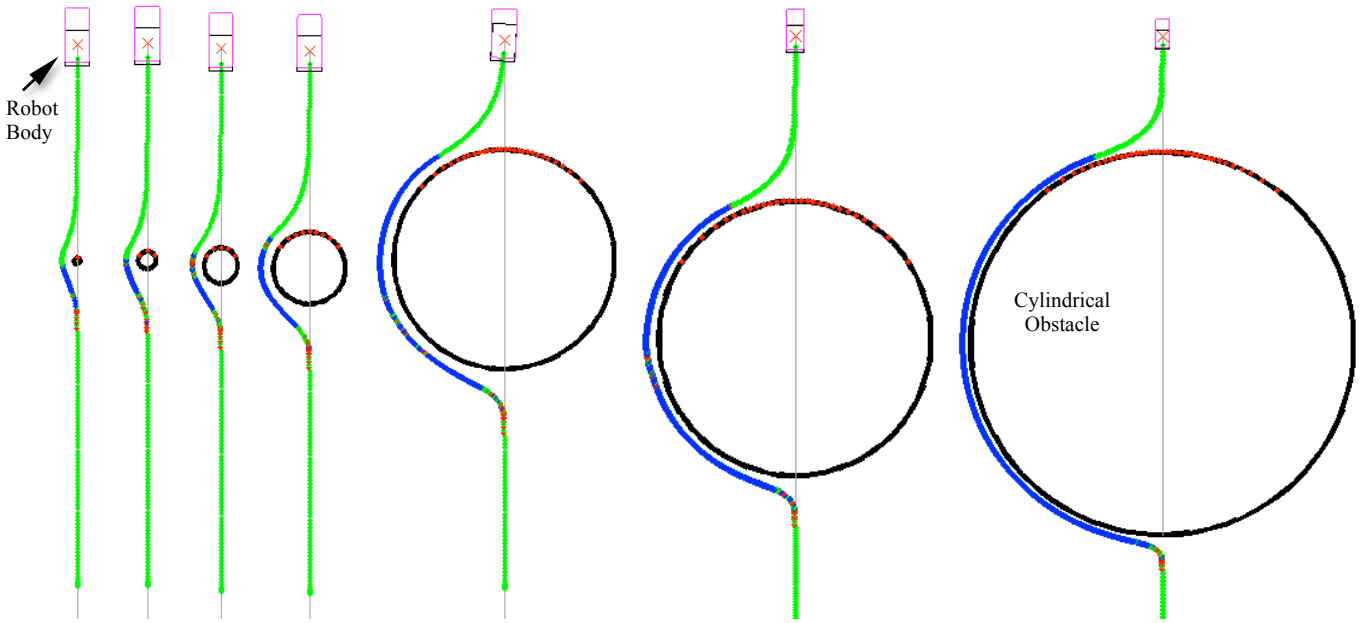


Fig. 8. The simulated rectangular vehicle (shown at top of each run) passing cylindrical obstacles (such as trees) with radii = 0.1m, 0.25m, 0.5m, 1m, 3m, 5m, and 10m using n = 10, α_{search} = .5, and $\alpha_{slowleft}$ = 0. In each case, the clearance on the front (diagram bottom) of the obstacle is less than 25cm. The rear (diagram top) clearance, is determined by the path driver and increases with diameter from 30cm to 160cm.

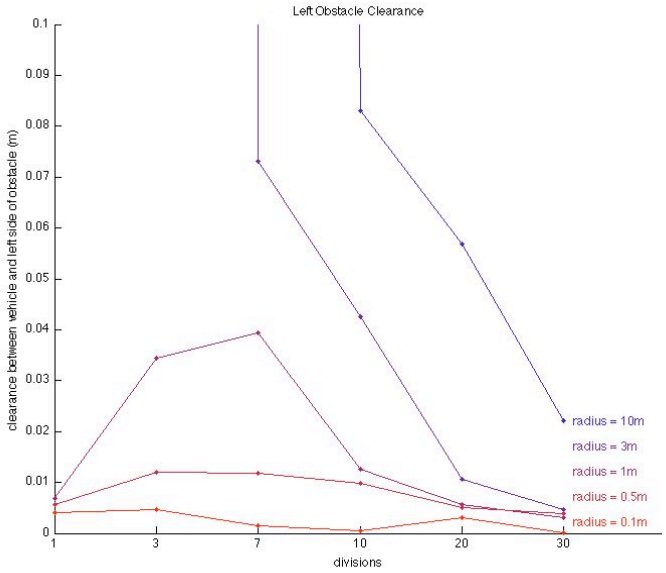


Fig. 9. The minimum clearance between the body of the vehicle and the obstacle at the leftmost edge of the obstacle. Obstacle diameter and the number of divisions, *n*, affect the clearance. Also, for some *n* the vehicle cannot pass the obstacle. For the smallest radii, large *n* doesn't provide sufficient clearance and the robot can get stuck. For radii>bodylength, small *n* doesn't provide sufficient search resolution to make the initial turn onto the obstacle and thus in the plot the clearance is infinitely large. *n*=10 allowed the robot to pass each obstacle.

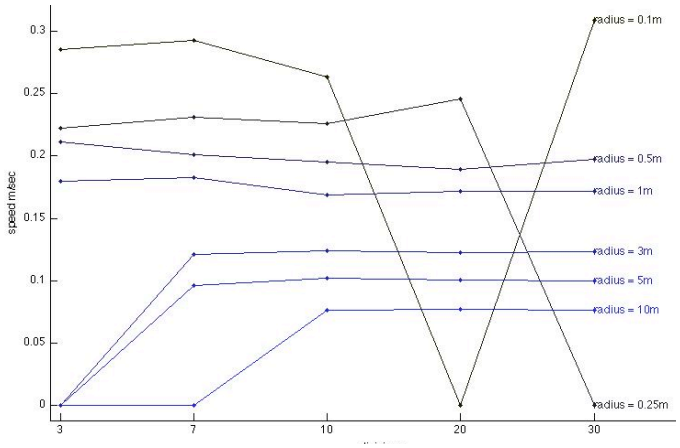


Fig. 10. The average speed during the pass around the obstacle increases with obstacle diameter because the path driver commands greater turning and lower velocities the further from the nominal straight-through path the vehicle is.

To get the forward distance, *y*, crossed by the center of mass:

$$y = \min(y_{turn}, y_{stop}) \qquad (21)$$

To allow clearance for edges further from the center than the front, we add an offset factor equal to the largest proscribed radius less the distance to the front. For example for a square robot the offset is 0.21 *w*, where *w* is the width of the square.

3) Extension Factor

To extend the distances covered by the constant-curvature stop we need extension factor η^*:

$$\eta^* = \frac{1}{v}\sqrt{\frac{2a_{\max}}{y(v, a_{\max}, \alpha_{search}, track) + 0.21w}} \qquad (22)$$

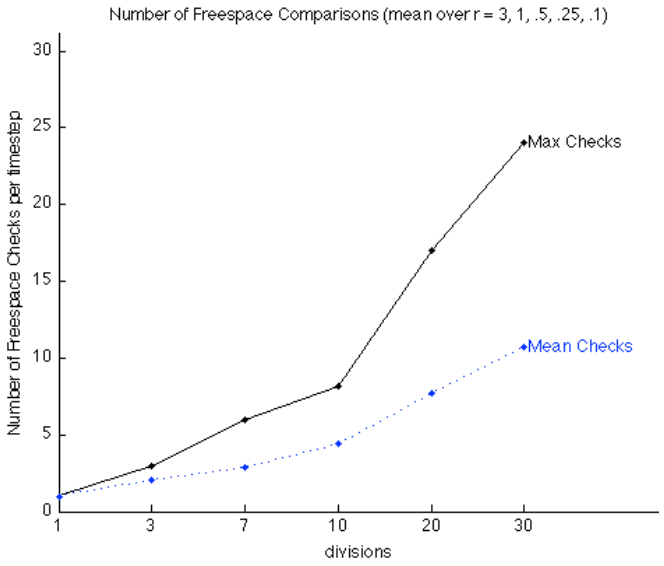


Fig. 11. The larger the divisions, *n,* the more points checked in the search for a safe command, and thus the higher the computation time per timestep.

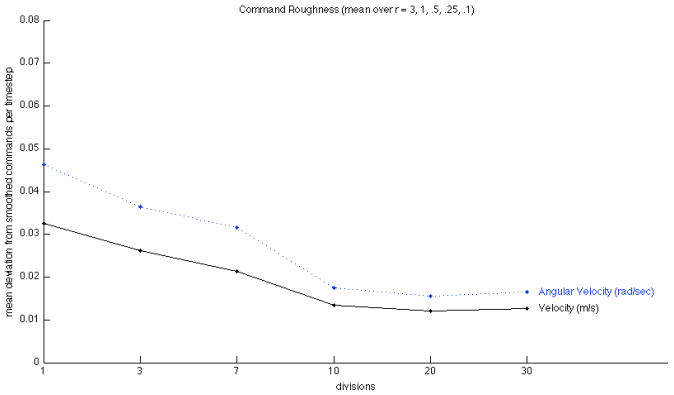


Fig. 12 Jerkiness of the control was determined by comparing commands with smoothed commands. Increasing *n* makes the velocity and angular velocity increase and decrease more smoothly.

If η^* is calculated at each timestep based on the current velocity, *v*, and the maximum acceleration of the vehicle, a_{max}, figure 7g is the result. The vehicle decelerates before the obstacle, followed by turning to follow the contour of the obstacle. As is demonstrated in fig 8, with a good extension factor, the robot can pass many different radii obstacles.

4) Search Resolution Sensitivity

The search resolution, controlled by the number of divisions of the max acceleration, *n,* affects the smoothness of the control, the calculations required at each time step and, indirectly, the closeness of the vehicle to the obstacle. The affect of *n* is plotted in figs (9-12). The safety margin in the *v*-ω space is equal to the distance between two of the points on the α_{search} line in fig. 6 because the second safe point on that line is selected. (Note that choosing a margin independent of *n* would double the number of freespace comparisons used in the search.) Figure 9 suggests that *n* = 10 is best for this system, allowing the vehicle to pass the widest range of obstacles. Figure 10 demonstrates that the radius has a greater affect on

speed than does n. Figure 11 shows the computational trade off and fig. 12 shows the smoothness trade off for other n.

For an arbitrary robot, we can use the estimation of y to transform the margin in v-ω space to a margin in length. One would expect:

$$\frac{dy}{dv}\Delta v \propto v\Delta t \qquad (23)$$

and thus an approximate upper bound for determining n for other vehicle parameters could be in the form:

$$C_{res} > \frac{nv}{1-\alpha}\sqrt{\frac{\alpha+1}{a_{\max}\text{track}}} \qquad (24)$$

where C_{res} is a dimensionless proportionality constant. From this example, with n=10, v=1m/sec, a_{max}=1m/sec^2, α = α_{search}=0.5, and track = 0.66m, we estimate C_{res} to be approximately 30.

5) Alpha Sensitivity

The parameter α that characterizes turning should be in the range of -1 to +1 in order to maintain safety by not accelerating from higher commands and in order to maintain wheel-acceleration limits, a_{max}. The slope of line in v-ω space along which safe commands are searched for is determined by α_{search}. If a safe command along that line is not found, a more extreme measure is required to pass the obstacle. So shifting the current velocity by the $\alpha_{slowleft}$ with the maximum possible acceleration is checked. If even that isn't considered safe, we slow the robot as much as possible (alpha = -1). (Fig. 6). To consider the effect of α_{search} and $\alpha_{slowleft}$ we tested the space of possible values (-1 to 1) in four environments (obstacles with radius .5, 1, 3, 5). The success of passing an obstacle is sensitive to α, especially for the smallest radius, fig. 13. Excluding the small radius data since it is so limited, we can examine the affects of α selection on speed (fig. 14), roughness (fig. 15), and clearance (fig. 16). Surprisingly higher α do not correlate with higher overall speeds. We generally choose α_{search} = .5 and $\alpha_{slowleft}$ = 0, which is successful for the range of radii and represents a good trade-off of other parameters.

VI. Path Driver

A priority for our lawnmower is to mow straight and parallel paths rather than random paths driven by other commercial autonomous mowers [1][2][3]. In addition to line segments, the mower also needs to be able to either pivot or arc to get between lines. Pivoting (attempting to maintain a zero net velocity while turning) is especially important in tight spaces. In addition to making complex cutting shapes possible, arcs are necessary for turning in thick grass where pivots were found to cause stalls. Our path driver follows strings of four different types of path segments: lines, arcs, pivots, and stops. For our mower, we generate these paths offline after measuring the GPS coordinates of the field corners. Many methods of path-tracking involve targeting a series of configuration states [36]. For our application, the line-following presented in [29] is a more natural choice because it provides a steering function which smoothly returns the moving vehicle to the line if obstacles or perturbations take the robot away from it.

The line-following steering function given in [29] asymptotically approaches the desired path when the robot begins close to the path and the adjustment of a single parameter, σ, determines the "smoothness" of the path. However, it is noted that when the initial state of the robot is very different from the states along the path, undesirable instabilities occur which may not converge to the desired line. For example: in our simulation testing, when the robot was far from the path and facing the wrong direction it spiraled many times before starting to approach the path. This is because there is a discontinuity at $\Delta\theta = \pm\pi$ and, if the Δd term is large, it can cause the robot to rotate beyond an orientation perpendicular to the path. We fixed this by including a $\cos(\Delta\theta)$ factor into the third term of the equation thus changing the steering function provided to:

$$d\kappa/ds = -3\kappa/\sigma - 3(\Delta\theta)/\sigma^2 - \Delta d\cdot\cos(\Delta\theta)/\sigma^3 \qquad (25)$$

Where κ is the curvature defined as $\kappa = d\theta/ds = \omega/v$ and $d\kappa/ds$ is the path derivative of curvature κ is controlled with linear feedback turns on κ, $\Delta\theta$ - the difference in orientation between the desired line and the robot, and Δd - the perpendicular distance between the path and robot. (Note that adding a cosine term does not affect linearized stability calculations of [29].

The desired curvature, κ, can be determined from $\kappa = \kappa_{\text{current}} + d\kappa/ds\cdot|v_{current}|\cdot\Delta t$. Once the desired curvature is determined, we pick a velocity by choosing the minimum of :

The desired speed specified for the path, $v_{desired}$

The velocity required to stop by the end of the path, $\sqrt{(2\cdot a_{md}\cdot d_{eop})}$ where a_{md} is the maximum deceleration and d_{eop} is the distance to the end of the path

A slew rate limit on the current velocity, $v_{current} + a_{md}\cdot\Delta t$, where a_{md} is the maximum acceleration, Δt is the timestep and $v_{current}$ is the current velocity

A relationship to the curvature required that allows larger velocities at larger curvatures, but limits the speed at large curvatures. We used $\sqrt{(a/\kappa)}$ where a *is a constant*.

Once the velocity is determined, is the angular velocity is $\omega = \kappa\cdot v$. After this calculation, the resulting angular velocity should be checked to ensure it is within a reasonable range, since infinite values for κ are not invalid.

The path driver continues to the next segment whenever the remaining progress to the endpoint reached a set threshold. If the threshold is small, then the robot comes to a stop at the end of the segment. A large threshold allows the robot to smoothly go between segments.

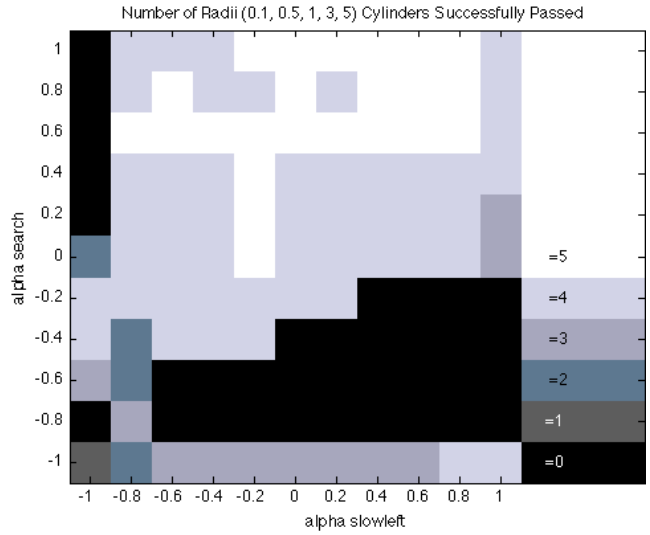


Fig. 13. Many combinations of α >0 permitted passage of obstacles >0.1, but fewer combinations, mostly around α_{search} = .6 were successful on all five obstacles.

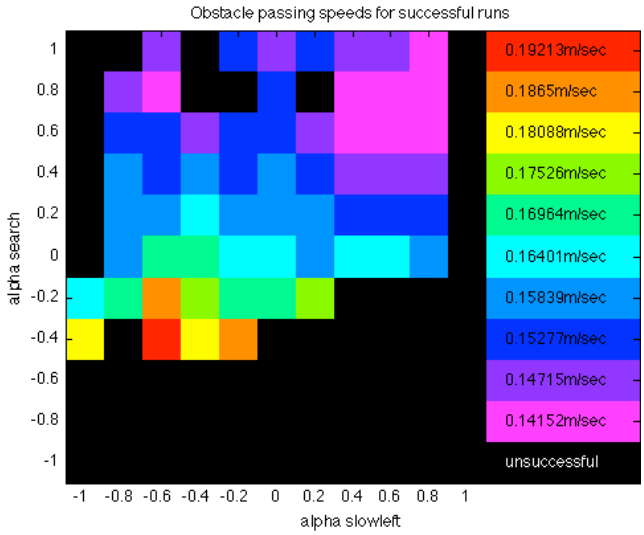
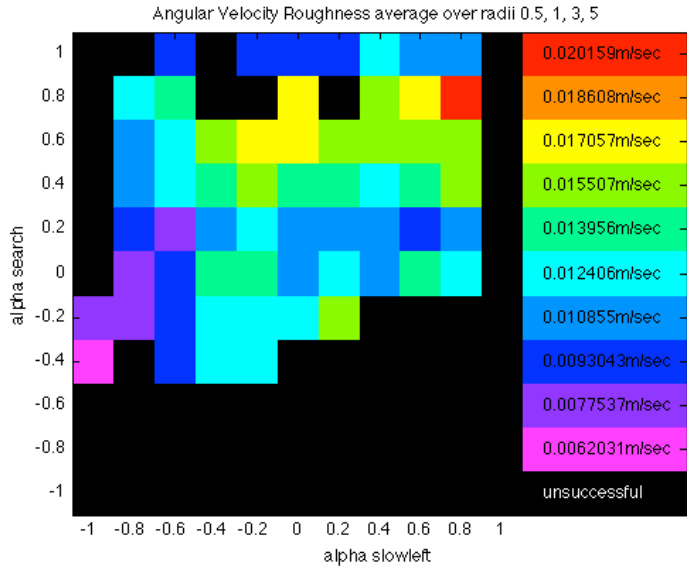


Fig. 14. Speeds tended to be slightly lower at larger α, perhaps because more safe commands were found at lower α. Low speed about 74% highest speed.



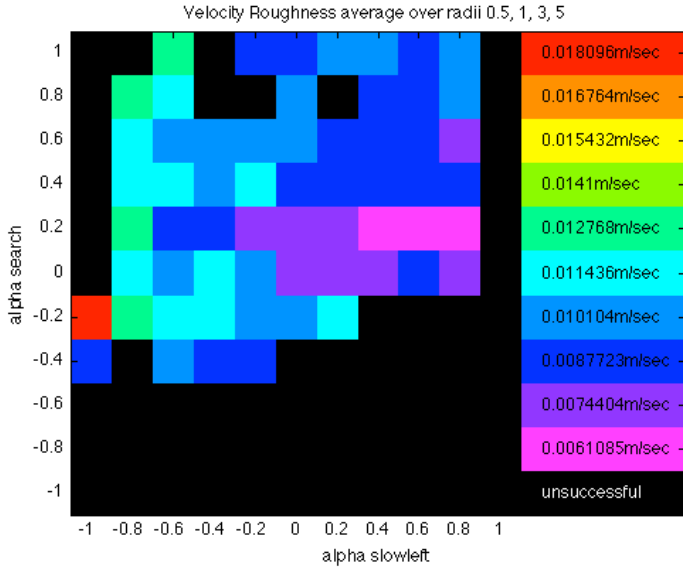
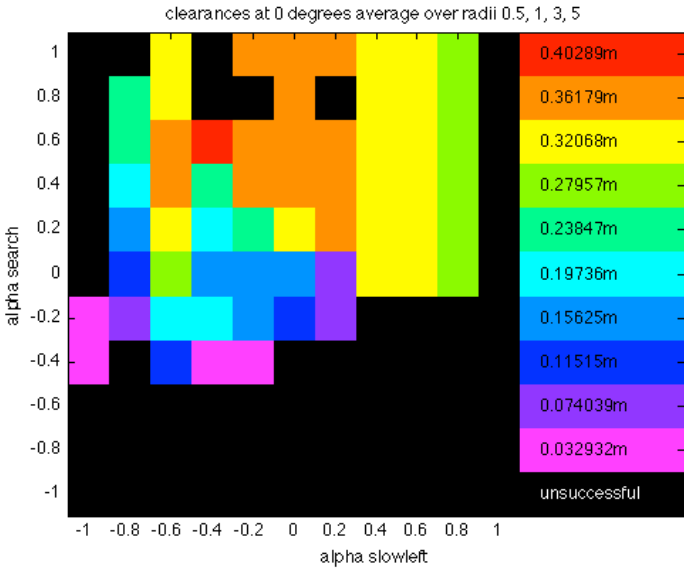


Fig. 15. Average difference between smoothed commands and actual commands, a measure of roughness, as a function of α_{search} and $\alpha_{slowleft}$ for trials that were successful for all four radii (0.5m, 1m, 3m, 5m). Lower values are smoother.



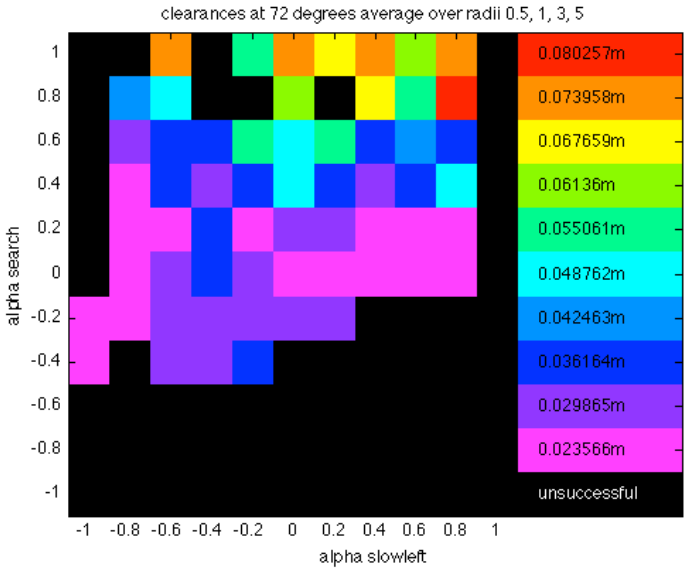


Fig. 16. Minimum clearance averaged over all four radii at front and side. Front clearance seems to be affected more strongly by $\alpha_{slowleft}$ while side clearance affected more by α_{search}.

For pivots, the path driver specifies $v = 0$ and controls ω with a PID on orientation. For our robot, requesting a pivot sometimes caused the motors to stall in thick grass, since the trailing rear wheel would often be facing the wrong way. Therefore, we converted the line-following algorithm to an arc-following algorithm by changing the κ in the first term to a $\Delta\kappa$, where $\Delta\kappa$ is the difference between the current curvature and the desired arc curvature. An important difference from line-following is the end condition. Since the primary purpose of our arcs and pivots is to re-orient the robot, the path driver moves onto the next segment when the orientation is within a certain tolerance, rather than the linear progress. If long arcs are given, both types of thresholds can be enforced. An example of the path driver and the reflexes working together to direct the robot through a cluttered environment is shown in Fig. 17.

VII. PERFORMANCE

A. CWRU Cutter Hardware

Our controller drove the robot CWRU Cutter, which is 126cm tall (including the GPS antenna) 66cm wide and 100 cm long. The robot is built on top of MTD's Troy-Bilt 48V Cordless Lawnmower's deck and powered by a bank of lead acid batteries. The drive wheels are two 20" bicycle wheels, which are each directly driven by a Maxon motor with Dimension Engineering 25A Sabertooth motor controllers. The rear wheel is smaller and trails behind passively. A Sick LIDAR unit is mounted low on the front to see 180º in the front of the robot. Tactile sensors and cameras may be added in future. A Novatel GPS receiver with Omnistar HP, a Crista IMU and quadrature encoders on the drive motors provide localization data.

A National Instruments cRIO 9074 embedded 500 MHz microcontroller real-time controller and field-programmable gate array (FPGA) backplane run the lowest level control. The main control loop executes on the cRIO Real Time MCP processor and has a loop period of 100ms (10Hz). All CWRU Cutter's software was written using National Instruments LabVIEW, since LabVIEW is easy to learn and supports FPGA and Real Time targets. The real-time loop: reads sensors (10 ms), physical state observer computes current x,y,θ,v,ω (15 ms), path driver determines v,ω (5 ms), polar freespace is computed (20 ms), reflexes find v,ω to edge obstacles if necessary (20 ms), and v,ω sent to motor controllers (5 ms).

For testing and transportation, the path driver block can be by-passed and replaced with a signal from an RC controller that allows direct manual input of (v,ω). The reflexes can be used with manual control or can be by-passed.

B. Simulation Results

The Path Driver, the Polar Freespace Observer, and the Reflexes were tested and debugged in a Labview simulation environment (fig. 18). The simulation consisted of obstacles made of line segments that provided the sharp corners most difficult for the polar freespace observer. In the front of mower and along the sides, the freespace is accurate. But in the rear, the freespace occasionally degrades and cuts through the corner of the boundary. This is a problem that could be solved with close-range sensors in the rear.

Even when random white noise of up to 10% of the velocity and signal delay of up to 2 timesteps were added, the reflexes were effective in preventing the robot from impacting the wall. It was impossible for the user to drive the robot into the edge of the freespace. When the Path Driver was bypassed, a human operator was able to drive the robot into an alcove so tightly that the robot couldn't get back out, because of the uncertainty of the freespace in the unobservable area. Adding sensors in the rear or by modulating reflex safety factors if the robot became stuck would solve this problem. However, when the path driver drove the robot in a typical back and forth lawn-mowing pattern, this situation rarely arose.

The path driver allowed very straight lines to be drawn, although sometimes its use in tandem with the veer-left reflex did create unexpected deviations. This occurred when the robot was driving fast toward an obstacle and started to turn away. The turn slowed the robot, which allowed it to return to the path, before turning away a second time to edge around the obstacle. In the worst case, the path driver might not hold the robot against the wall in a u-shaped obstacle for example. We can solve this problem in most simulation environments by having the path driver command a constant v and ω to the right when the robot is sufficiently far from the path. The bug algorithms solve this problem completely by using a wall-following behavior and remembering some environmental data [34].

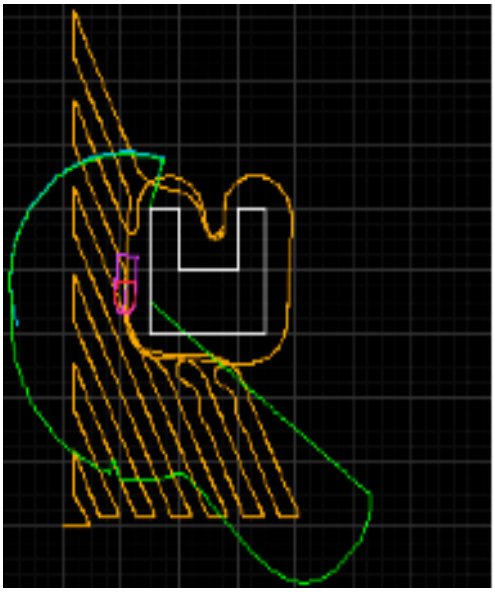


Fig. 18. Results of a simulated trial on a straight-sided obstacle. The orange parallel lines are the path traced by the robot, in red. The swept area of the current velocity is in pink and the polar freespace is in green.

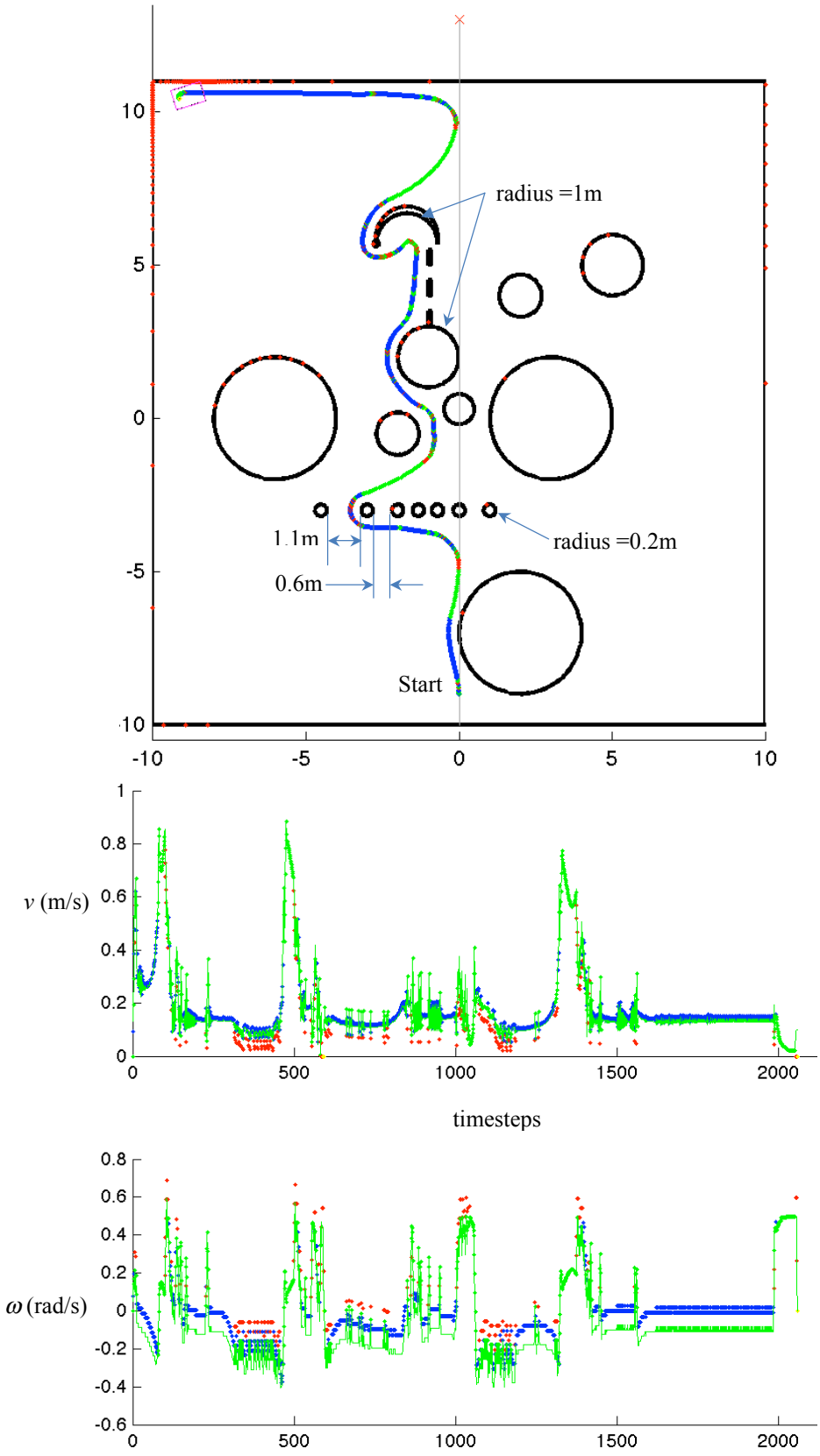


Fig. 17. The robot threading through a cluttered field of obstacles, demonstrating that the robot can follow discontinuous walls and pass through gaps, and even exit limited concave obstacles. At the end of the run, when the robot does get trapped in a corner, the robot stops before impacting the walls. Velocity and angular velocity selected by the path driver (in green) and reflexes (blue for veering left and red for stopping) are shown.

C. CWRU Cutter Performance

Without reflexes, localization inaccuracies required all a priori paths to be over a meter away from obstacles, and still unintended collisions stationary obstacles occurred due to drift and variation of the Physical State estimate. Adding the reflexes allowed CWRU Cutter to edge closely along fences or skirt centimeters away from boxes, people, or walls in its path. In 20 trials, there was only once that CWRU Cutter touched the fence, but the contact was so slight that no damage was done either to the mower or to fence. Moving obstacle detection worked when indoors and the localization estimates were smooth.

The LIDAR must be correctly adjusted for the robot to operate. If the LIDAR is too low, high grass will be treated as an obstacle. If it is too high, low obstacles might be overrun. Some common obstacles, like shoes, are below the grass line and need to be sensed in a different way. Vision[25] and range scan filtering[13] are promising solutions to these problems. The clearance between obstacle and mower can be controlled by safety factors, but is also influenced by the vehicle's speed, allowing a closer cut at lower speeds.

Unlike random-path commercial robots, CWRU Cutter cut lines that were parallel to the eye. The user is required to determine the coordinates of the boundary to be mowed, and then back-and-forth paths across the whole area are generated. The robot safely skirted stationary and moving obstacles without requiring tactile contact.

VIII. Conclusions and Future Work

This method of filtering and reacting to sensed obstacle data allows robots of any shape to approach obstacles closely and narrowly edge around many obstacles. We can use these reflexes with raw LIDAR data, LIDAR data filtered with a polar freespace, 3D range finder data, and even polar free ranges extracted from texture and color analysis of camera data[33][37]. These different types of data do not need to be put into a high-resolution configuration space, but can be considered in the form of a polar single-valued function which can be compared with robot footprint simply. While the choice of path driver affects the speed and final path of the robot, the reflexes are robust to types of path driving and higher-level commands.

Like any reactive behavior, the robot can get stuck. Using the above formulation allows many typical lawn obstacles to be avoided, but in future work higher-level controllers need to be used to control reflex parameter to allow the robot to back up and try again. Additionally, if the higher-level reflex recognizes that the robot is against an obstacle, it would prevent the robot from looping back when non-convex obstacles are encountered.

During the many tests of our robot, we were often seeking a good trade off between low clearances and ease of manual control. For example, indoors the operator often preferred to drive with the reflexes off in order to drive with increased speeds in cluttered rooms or corridors, trusting that the increased traction would allow for greater accelerations and decelerations. Future work could allow safer maneuvering around even tighter obstacles by perceiving the environment with additional close-range sensors in critical directions, and real-time learning and adjustment of critical reflex parameters to reflex perceived performance.

Acknowledgment

The authors would like to thank Todd Jacobs, Kyle Layton, Wyatt Newman, Arkady Polinkovsky, and Matthew Voss.

References

[1] "The lawnbott evolution revolution," 2005[Online]. http://www.bamabots. com/Kerry/112805ambrogio.htm

[2] "Robomower review," 2005 [Online]. http://www.bam-abots.com/rl1000review. htm

[3] Husqvarna Automower [Online] www.automower.com/

[4] R. Ulrich, F. Mondada, J.-D. Nicoud. "Autonomous vacuum cleaner." *Robotics and autonomous systems*, 1997.

[5] N. Nourani-Vatani, M. Bosse, J.Roberts, and M. Dunbabin, "Practical path planning and obstacle avoidance for autonomous mowing." *Australasian Conference of Robotics and Automation*, 2006.

[6] J. Smith, S. Campbell, J. Morton "Design and Implementation of a Control Algorithm for an Autonomous Lawnmower" *Midwest Symposium on Circuits and Systems*, 2005.

[7] M.C. Garcia-Alegre. A. Ribereiro, L. Garcia-Perez, R. Martinez. D. Guinea. "Autonomous Robot in Agriculture Tasks". *European Conf. On Precision Agriculture*, France, 2001

[8] A. Bemporad, M. Di Marco, and A. Tesi, "Sonar-based wall- following control of mobile robots," *Journal of Dynamic Systems, Measurement, and Control*, vol. 122, no. 1, pp. 227-230, March 2000.

[9] R. Carelli and E. 0. Freire, "Corridor navigation and wall-following stable control for sonar-based mobile robots," *Robotics and Autonomous Systems*, vol. 45, pp. 235-247, 2003.

[10] T. Yata, L. Kleeman, and S. Yuta, "Wall following using angle information measured by a single ultrasonic transducer," *in IEEE International Conference on Robotics and Automation*, 1998, pp. 1590-1596.

[11] *Journal of Field Robotics, Special Issue on the 2007 DARPA Urban Challenge*, Part I, Issue Edited by M. Buehler, K. Lagnemma, S. Singh, Vol. 25 Issue 8 (August 2008)

[12] W. Newman et. al, "Team CASE and the 2007 DARPA Urban Challenge," Darpa Grand Challenge Tech Report online at http://www.darpa.mil/ June 2007.

[13] B.-H. Schäfer, A. Hach, M. J., Proetzsch, K. Berns. "3D Obstacle Detection in Vegetated Off-Road Terrain." *2008 IE EE International Conference on Robotics and Automation (ICRA)*, May 19-23, 2008, Pasadena, CA.

[14] Rapidly-exploring random trees: Progress and prospects. S. M. LaValle and J. J. Kuffner. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293--308. A K Peters, Wellesley, MA, 2001.

[15] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, June 1996.

[16] J. Borenstein and Y. Koren, "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots." *IEEE Transactions on Robotics and Automation*, Vol. 7, No. 3, June 1991.

[17] R. Simmons "The Curvature-Velocity Method for Local Obstacle Avoidance," *International Conference on Robotics and Automation,* April, 1996.

[18] D. Jung and A. Zelinsky, "Whisker-based mobile robot navigation," in Proceedings of the IEEE/RSJ International Conference on Intel- ligent Robots and Systems (IROS), vol. 2, November 1996

[19] A. G. Lamperski, O. Y. Loh, B. L. Kutscher and N. J. Cowan. “Dynamical Wall Following for a Wheeled Robot Using a Passive Tactile Sensor.” *IEEE International Conference on Robotics and Automation*. Barcelona, Spain, April 2005.

[20] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *AIAA Journal of Guidance and Control*, 25(1):116–129, 2002.

[21] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *Int. J. Robot. Res.*, 17:840–857, 1998.

[22] H. Noborio, I. Yamamoto, and T. Komaki, "Sensor-based path- planning algorithms for a nonholonomic mobile robot," *International Conference on Intelligent Robots and Systems*, 2000, pp. 917-924.

[23] O. Khatib. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. In *Proc. IEEE Intl. Conference on Robot- ics and Automation*, St. Louis, MO, March 1985, pp. 500- 505.

[24] A. Kelly, “An Intelligent, Predictive Control Approach to the High-Speed Cross-Country Autonomous Navigation Problem” PHD Thesis Carnegie Melon 1995.

[25] S. Thrun, “Stanley: The Robot that Won the DARPA Grand Challenge.” *Journal of Field Robotics* 23(9), 661–692 (2006).

[26] C. Stachniss and W. Burgard. “An Integrated Approach to Goal-directed Obstacle Avoidance under Dynamic Constraints for Dynamic Environments” IROS 2002

[27] J Minguez, L Montano and J Santos-Victor. “Abstracting Vehicle Shape and Kinematic Constraints from Obstacle Avoidance Methods” *Autonomous Robots*. Jan 2006. Vol 20. No 1.

[28] T. S. Wikman, M. S. Branicky, W. S. Newman. “Reflexive Collision Avoidance: A Generalized Approach.” *IEEE ICRA* 1993.

[29] Y. J. Kanayama and F. Fahroo, “A New Line Tracking Method for Nonholonomic Vehicles.” *IEEE International Conference on Robotics and Automation*. April 1997. Albuquerque, New Mexico.

[30] C. Chen, R. Quinn, and R. Ritzmann, "A Crash Avoidance System Based Upon the Cockroach Escape Response Circuit," *IEEE International Conference on Robotics and Automation (ICRA)* Albuquerque, NM, USA, 1997.

[31] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems." *Transactions of the ASME--Journal of Basic Engineering* D ser. 82 (1960): 35-45.

[32] Grewal, M., et. al. *Global Positioning Systems, Inertial Navigation, and Integration*, 2. Hoboken, NJ: Wiley, 2008, pp. 394.

[33] A. Schepelmann, H. Snow, B. Hughes, J. Green, F. Merat, R. Quinn. Vision-Based Obstacle Detection for the CWRU Cutter Autonomous Lawnmower. *In Proceedings of the IEEE International Conference on Technologies for Practical Robot Applications (2009 IEEE TePRA),* Woburn, MA, Nov. 2009, pp. 74-79.

[34] V. J. Lumelski and A. A. Stepanov, “Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment,” *IEEE Transactions on Automatic Control*, Vol. AC-31, No. 11, 1986.

[35] R. Chatterjee and F. Matsuno. "Use of single side reflex for autonomous navigation of mobile robots in unknown environments," *Robotics and Autonomous Systems,* Vol 35, No 2, 2001.

[36] Y. Kanayama, A. Nilipour, and C. A. Lelm. “A locomotion control method for autonomous vehicles.” *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1315-1317, Philadelphia. Pennsylvania, April 1988.

[37] A. Schepelmann, R. Hudson, F. Merat, R. D. Quinn. Visual Segmentation of Lawn Grass for a Mobile Robotic Lawnmower. *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 18-22 October 2010. *(Submitted)*