Practical Path Planning and Obstacle Avoidance for Autonomous Mowing

Navid Nourani-Vatani[1,2], Michael Bosse[1], Jonathan Roberts[1] and Matthew Dunbabin[1]
[1]Autonomous Systems Laboratory, CSIRO ICT Centre
PO Box 883, Kenmore, Qld 4069, Australia
Email: Firstname.Lastname@csiro.au
[2]Ørsted - DTU, Automation, Technical University of Denmark
Elektrovej, Bldg. 326, DK-2800 Kgs. Lyngby, Denmark

Abstract

There is a need for systems which can autonomously perform coverage tasks on large outdoor areas. Unfortunately, the state-of-the-art is to use GPS based localization, which is not suitable for precise operations near trees and other obstructions. In this paper we present a robotic platform for autonomous coverage tasks. The system architecture integrates laser based localization and mapping using the *Atlas Framework* with *Rapidly-Exploring Random Trees* path planning and *Virtual Force Field* obstacle avoidance. We demonstrate the performance of the system in simulation as well as with real world experiments.

1 Introduction

Currently, it is labor intensive to maintain parks, sporting fields, golf courses and driving ranges. The Brisbane area (where our lab is located), with one million people, contains over 2000 parks covering an area of approximately 13000 ha [BCC, 2006]. Typical maintenance tasks include grass mowing, fertilizer and herbicide spreading and in the case of driving ranges, golf ball collection. These operations are not only time consuming, but can also be hazardous to the operators and the general public. Additionally, they generally occur during day light hours, limiting and interrupting the time people can use the facility.

There is therefore a need for an automated system that could perform these tasks, providing that it can be cost-effective. The benefits of such a system are reduced labor cost, and allowing after-hours maintenance leading to more daylight hours access and in the case of golf courses, increased green fees.

A number of autonomous systems have been developed for broad-acre harvesting such as the BEELINE system

Figure 1: Our tractor lawn mowing vehicle autonomously cutting long grass.

[Beeline Technologies, 2006] which uses GPS guidance to harvest crops, but there are no commercial systems targeted or suitable for mowing in park or golf course areas. Notable research work has been performed by [Batavia *et al.*, 2002] and [Jarvis, 2001]. [Batavia *et al.*, 2002] showed impressive results and demonstrated autonomous mowing of a golf course for over 20 hours covering an area of 82 ha. They used high-quality RTK-GPS as the localization sensor and also showed an obstacle detection system based on a movable 2D laser scanner (although obstacle detection was not used in all 20 hours of autonomous operation). [Jarvis, 2001] suggests a tri-level control strategy. The two lower levels of control, the reactive and the anticipatory one, provide local support for obstacle collision avoidance. The third, top level, is operated manually and provides the planning.

Over the past 15 years, a number of household lawn mowing robots which are capable of mowing small areas of domestic lawn have been developed. There is even a robot lawn mower competition in the USA[ION, 2006] with the 2006 competition attracting five university entries. Early work by Noonan, et al.[Noonan *et al.*, 1993] described an autonomous lawn mower that used metallic reference markers placed beneath the lawn to guide the

lawn mower. The robot lawn mower drove along paths pre-stored on-board. The Israeli company, FriendlyRobotics, have been making autonomous lawn mowers for ten years. Their Robomower RL1000 product[FriendlyRobotics, 2006] is a fully autonomous lawn mower that detects the edge of the mowing area using a perimeter wire, buried around the edge of the lawn. A very similar robot lawn mower is produced by Mowbot of the UK[Mowbot, 2006]. Both these robots are designed for domestic (small) garden type situations and can mow areas of 100-200m^2 per hour.

Importantly, they are also designed with automation in mind and can turn on the spot. The programmed cutting patterns makes use of this maneuverability. For example, the Robomower uses a so-called W-shaped cutting pattern to cover the area to be cut which contains 90-degree turns.

1.1 Our Previous Work

In previous work [Dunbabin *et al.*, 2004], we described our first attempts at autonomous mowing of golf course and park sized areas. Figure 1 shows our tractor lawn mower robot autonomously mowing long grass at our laboratory test facility. Here, a human driver first drove the tractor around the perimeter of the area to be cut, while the system recorded its position using GPS. A simple inward spiral algorithm was then used to precompute a trajectory to cover the mowing area. The tractor then autonomously navigated along the spiral trajectory while cutting the grass. Like [Batavia *et al.*, 2002] we used high-quality RTK-GPS for localization. However, GPS does not work well in areas around and underneath trees and buried-wires are unsuitable for mowing large areas such as parks and golf courses (common in typical mowing areas in our target mowing applications).

1.2 Laser-Based Navigation

This paper describes a navigation and obstacle avoidance system developed for an under-actuated mowing robot (our tractor) that does not use buried-wires or GPS. We have instead based our system on a 2D laser scanner. We achieve localization and mapping together using the *Atlas Framework* [Bosse, 2004; Bosse *et al.*, 2003; 2004].

We also constraint ourselves to only use forward motion commands.

1.3 Paper Outline

The remainder of this paper is structured as follows. Section 2 describes the architecture of our system including details on the perceptive and executive modules. Section 3 then shows some results from a simulated environment (Gazebo-based) and from the real tractor. Finally, Section 4 lists some conclusions and proposes further work.

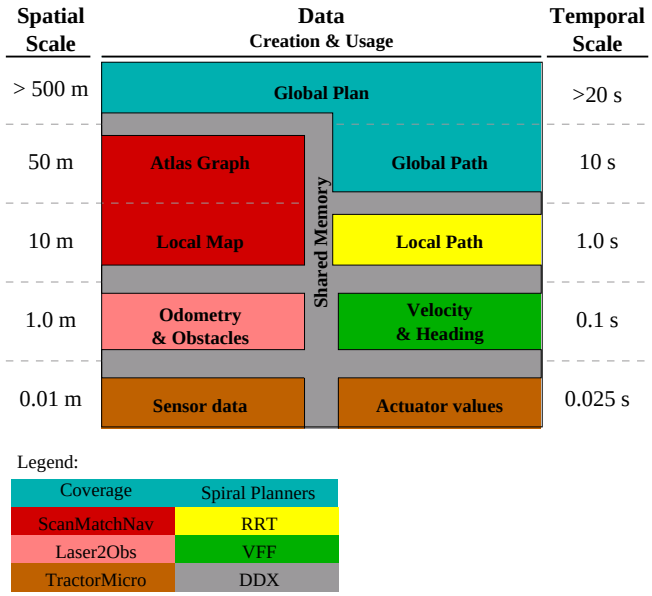


Figure 2: The software architecture is a hierarchy of five levels. Sensor reading and actuator control are processed at the lowest level. As we climb the hierarchy ladder, a more complete picture of the robot and its environment is built. This information is made available for the decision making modules through a shared memory configuration. Some modules create or access data over more than one level. This is shown by highlighting each module in a separate color.

2 System Architecture

2.1 Robotic Testbed

The experimental platform is a Toro ride-on mower which has been retro-fitted with actuators, a control system, and a computer, enabling control over the vehicles operations. Generally the low-level vehicle control and sensor processing occurs on the on-board computer, whereas higher-level routines are run on a notebook computer connected through a LAN connection. The vehicle is used as a testbed for many applications and is fitted with an array of sensors including a wheel encoder, a turn-angle sensor, a GPS receiver, a 6-DOF IMU, a laser scanner and video cameras (see Figure 1 for a photograph of the vehicle). For the algorithms described in this paper, we need only employ the wheel encoder, turn-angle sensor and the laser scanner.

2.2 Software Architecture

Figure 2 shows our layered software architecture. The left column shows the spatial scale, the right column the temporal scale, and the data abstraction is shown in between with the shared memory separating the modules. The system architecture is arranged in a hierarchical manner. The raw sensor measurements and actuator

control values are depicted on the lowest level. Each higher layer increases the abstraction of the data and processes the data at a lower rate.

The system modules can be put into two categories. In the first category are modules that perceive the world. These, indicated on the left side of Figure 2, process sensor readings to estimate the robot's pose and its surroundings. The second category are executive modules, that act on the world. These modules, on the right side of Figure 2, determine which motions the robot should be commanded to achieve its goals. Each of the executive modules needs an appropriate level of information about the robot and its surroundings. This information is provided by the perceptive module at the same abstraction level. In order to find solutions for complex tasks, several levels of abstraction are needed. Abstraction is achieved by minimizing details while maximizing understanding. While it takes time and processing to go up one level of abstraction, decisions made at higher levels do not need to be repeated as often as the lower-level abstractions. Our system architecture is based on this layered design of building spatial awareness step-by-step and separating planning from control.

2.3 Inter-modular Communication

Communication between modules is carried out using Dynamic Data eXchange (DDX) [Corke *et al.*, 2004]. DDX allows a coalition of programs to share data and commands through an efficient shared memory mechanism.

2.4 The Perceptive Modules

The perceptive sub-system is made up of the following three modules: *TractorMicro, Laser2Obs* and *ScanMatchNav.* Together these modules implement localization and map making for the robot. The lower-level modules work with the raw sensor data to create a quick estimate of the robot's pose and its environment. The higher-level modules combine the information that has been gathered and calculated by the lower-level modules to create a more complete picture of the robot's pose and its surroundings.

TractorMicro

The *TractorMicro* module, attached to the rear of our mower tractor, interfaces with an HC12 unit which reads all the sensors and publishes their values into shared memory via DDX [Usher, 2005]. The turn angle sensor readings and the wheel encoder's odometer values are combined and integrated to compute the odometry based robot pose estimate.

Laser2Obs

The *Laser2Obs* module is a simple process that combines the odometry based pose estimates with the most recent raw laser scans. The module publishes a list of current obstacle points from the current laser view with respect to the odometry pose.

ScanMatchNav

The *ScanMatchNav* module is an implementation of the *Atlas Framework* [Bosse, 2004; Bosse *et al.*, 2003; 2004]. This module covers two levels of abstraction. Its lower-level abstraction contains local maps generated by laser scan matching. These maps are small and contain only a few laser scans. The higher-level abstraction is a graph where nodes contain the local maps and edges contain the coordinate transformations between neighboring maps.

Instead of maintaining a single global coordinate frame, coordinates from distant maps are related by composing the transformations along a path through the graph. The paths through the graph are computed using the Dijkstra Shortest Path algorithm with the transformation uncertainty measure as a distance metric [Bosse *et al.*, 2004]. This approach allows us to process environments with large loops in real-time, since at any time, only a small number of local maps are active, and no global optimizations are necessary during loop closures.

The primary outputs from the *ScanMatchNav* module are the corrections to the odometry based poses, the local maps containing obstacle information and updates to the graph of map frames. The advantage of publishing a pose correction instead of an updated pose is that a correction is less susceptible to synchronization errors and processing latencies. The corrections computed from past data can be applied to the most recent odometry measurements since they vary less with time than the motion of the tractor.

2.5 The Executive Modules

The executive sub-system implements task and path planning, obstacle avoidance, path control and actuator control. The goals and commands progressively become more detailed at lower layers in the architecture. The top most goal is to mow an area. Achieving this goal is accomplished by commanding global paths which are divided up into local paths that avoid obstacles. A path is a sequence of way-points which is driven by pure-pursuit to a point on the path at a fixed distance ahead of the vehicle. At the next level down, the desired steer angles and velocities are used as inputs to PID loops which control the actuators.

In this way we separate decision making from the actual robot control. The low-level modules compensate for the low update rate and less detailed view of the

higher-level modules by having some reacting behavior built into them.

The sub-system modules consist of *Global Planner,* Rapidly-exploring Random Tree (RRT) based *Local Path Planner*, *Path Controller* and *TractorMicro.*

Global Planner

We have shown in our previous work [Dunbabin *et al.*, 2004] that we can carry out the task of covering a given area by driving inward spiral paths. However simply spiraling inwards is not sufficient to completely cover an area. The *Global Planner* needs a different strategy when the coverage area is too narrow to turn in, and it needs a method to recover any gaps from previous passes. Describing the solution to these issues are out of the scope of this paper and will be reported in future publications. This paper will focus on the lower-level modules necessary for planning local paths and obstacle avoidance which can be used with a variety of global path planners.

Local Path Planner

The *Local Path Planner* is used to generate short paths (~ 10 m) that take into account the vehicles kinematics and any obstacles in view. This simplifies the task of the global planners by reducing their concern for dynamic constraints.

A Rapidly-exploring Random Trees (RRT) planner is employed as a local path planner. We use an RRT planner because it can take into account the vehicle kinematic constraints to generate feasible paths that avoid obstacles [LaValle and Kuffner, 1999].

The down side of using an RRT planner is that it is random and hence in its basic form the path produced cannot be controlled. If the requirement is only to get the robot from A to B this is not a major issue, but when the requirement is to cover an area then it is very important that the path follows the generated way-points *unless* these way-points are infeasible or obstructed by obstacles. Another problem with this planner is that it is not guaranteed to find a trajectory when allowed a limited number of iterations. We overcome these limitations by producing paths by the higher-level *Global Planner*, using RRT only when obstacles are present. When RRT is incapable of producing a better path avoiding obstacles, we fall back on the way-points generated by the *Global Planner*.

One possibility of improving the RRT path planner in order to produce more controlled paths lies in combining it with a Road Map planner [Acar and Choset, 2002]. We are considering using this implementation in our future work.

Controller

The control module implements both path following and obstacle avoidance in the same module. While the high-level modules plan paths around obstacles, there is a need for additional obstacle avoidance. This need is, of course, due to the dynamic environment and the errors in the estimated robot poses. We have determined that a lightweight reactive controller fulfills our requirements.

The basic method for driving the planned paths uses the pure pursuit algorithm [Wallace *et al.*, 1995]. We have extended the controller with Virtual Force Field (VFF) [Borenstein and Koren, 1989] as the base for our obstacle avoider.

One of the major flaws with VFF is that it represents the robot as a point and does not take the vehicle's shape into consideration. In our approach we calculate the obstacle's repulsive forces on the robot body rather than on the robot center-of-motion (COM). If an object is closer than a specified maximum distance it will apply a force perpendicularly on to the nearest body point. We have four force regions:

$$F_i = \begin{cases} \infty & dist_i < l \\ k/(d-b) \cdot (b^2/dist_i + d - 2 \cdot b) & dist_i < b \\ k \cdot (dist_i - d)/(b-d) & dist_i < d \\ 0 & dist_i > d \end{cases}$$

where F_i is the force from the ith obstacle, $dist_i$ is the distance to the nearest robot body point, k is a constant, l is the virtual wall distance where the force goes to infinity, b is the obstacle boundary distance and d is the maximum distance where the obstacle has an influence on the robot. Region d-b is linear to resemble the effect of a spring. When the vehicle is closer than b, the force increases exponentially, but if the distance is less than l, the robot will be halted immediately. The force on the body point creates a torque on COM by:

$$\tau_i = \vec{b_i} \times \vec{F_i}$$

where τ_i is the torque from obstacle i and $\vec{b_i}$ is the vector from the body point to COM. The torques from each obstacle point are summed and added to the desired turning rate to calculate the commanded turning rate for the robot. See Figure 3.

It is also desirable to slow the vehicle when it drives towards an obstacle, in order to gain more time to avoid it. To do so we use the x-component of the repulsive force, x_F, to calculate a negative forward velocity. Using the x-component ensures maximum influence from obstacles the robot approaches and zero influence from the obstacles on the side and not in the direct path, since the force vector of these obstacles has no x-component.

To ensure that the sum of the obstacle forces do not push the robot faster than it is commanded or make it

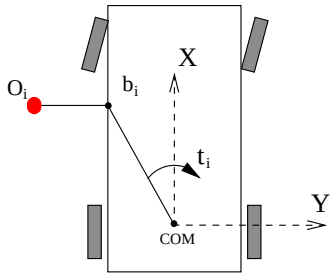


Figure 3: Effect of the obstacle force on the robot. The obstacle repulsive force is calculated on the nearest body point. The effect of this force is transformed to a change of heading by calculating the torque it creates on the center-of-motion.

move backwards we map these forces via an exponential.

$$v_{\text{cmd}} = v_{\text{desired}} \cdot \exp\left\{-\sum x_F\right\}$$

In this manner, when the force is zero, there is no change to the desired velocity, and as the repulsive forces tend toward infinity, the commanded forward velocity tends towards zero.

We multiply the desired vehicle velocity with the exponential of the negative repulsive force to get the commanded forward velocity.

The values for l, b and d are not chosen randomly. It is not necessary to have the l limit if the robot has tactile bumpers or other touch sensors and it is acceptable for the robot to touch objects; this is, for instance, the case for the vacuum cleaning robots which have to touch walls to be able to clean the edges. We are not concerned with touching the obstacles and have therefore chosen $l = 0.1\,\text{m}$. The virtual obstacle boundary limit is the desired distance to an obstacle and has been chosen to be $b = 0.5\,\text{m}$. d is vehicle specific and depends on the vehicle's maximum velocity, brake distance and turn radius. When traveling at $1\,\text{m/s}$, our tractor needs $1.5\,\text{m}$ of distance to slow down and make a minimum radius turn. We have hence set $d = 1.5 + b = 2.0\,\text{m}$. Figure 4 illustrates the repulsive force curve with these parameters.

Our mower has a car-like shape which gives us four sides and four corners. However we have added a virtual point to the front of the vehicle's shape which is necessary to ensure that the robot has enough space to turn before hitting an obstacle in front. Obstacles already behind the vehicle are ignored since we only command forward motions. There are therefore seven body regions as depicted in Figure 5. An obstacle exerts force on the vehicle at the closest body point if it is in any of the seven regions and closer than d.

We are left with one tuning parameter only, k, which is the repulsive force constant. We have empirically tuned

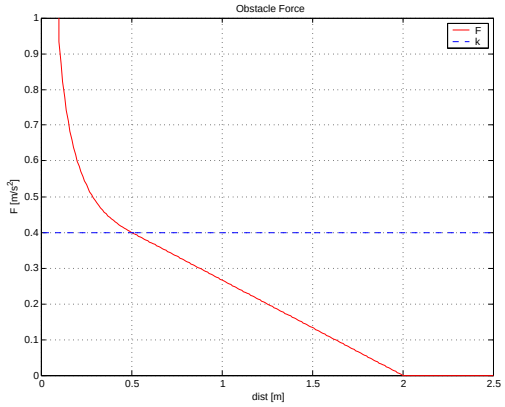


Figure 4: Obstacle force curve for $l = 0.1\,\text{m}$, $b = 0.5\,\text{m}$, $d = 2.0\,\text{m}$ and $k = 0.4$.

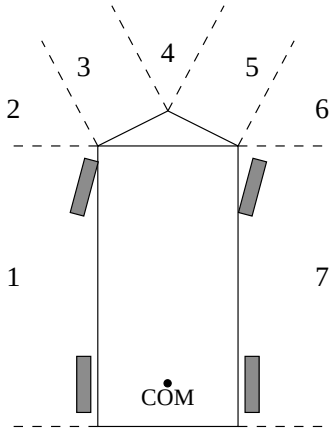


Figure 5: The seven body regions.

this value to $k = 0.4$.

TractorMicro

The TractorMicro module is represented both on the perceptive and executive sides of the system architecture, Figure 2, since the module interfaces to the robot actuator drivers, converting velocity, heading and other tractor control variables.

2.6 The Coverage Task

The input to the Task Planner is the circumference of the area to be mowed. The input is given by driving the mower manually on the perimeter of the area to be covered. During this run the robot creates a boundary map of the environment. This map is created using laser scans as input to the *Atlas Framework*.

The *Global Planner* passes way-points for an inward spiral path to the *Path Controller*. The spiral way-points are generated by offsetting each way-point inwards by

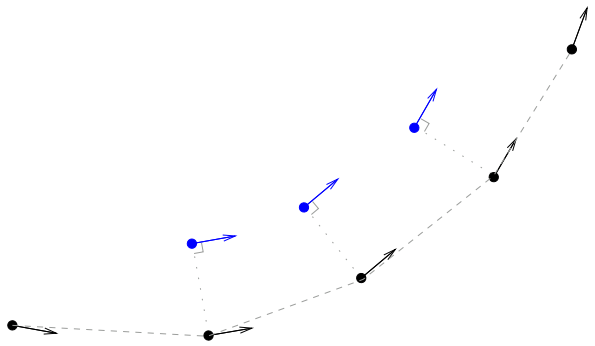


Figure 6: The spiral points (in blue) are generated by offsetting the way-point from the previous path (in black).

the mower's width (Figure 6). If way-points are obstructed by obstacles these way-points are passed to the RRT *Local Path Planner* which calculates a path around the obstacles.

3 Results

As part of our development we have found it useful to initially test the sub-systems using a simulator as this increases productivity considerably. We have implemented an extension to the Gazebo simulator [Howard, 2006] for our robot.

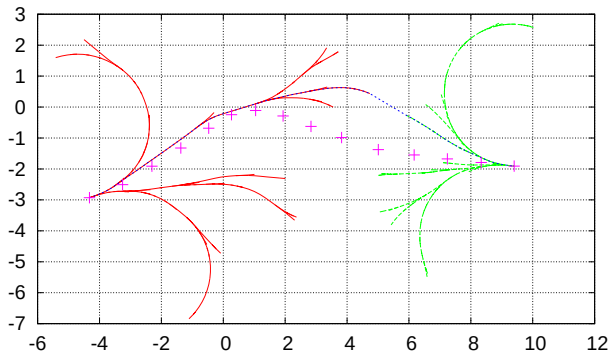
In our layered architecture the Gazebo simulator takes the place of the TractorMicro module. This does not affect or require changes to the other modules in the architecture as the modules are integrated via DDX.

Our experience has shown that the simulator matches the real world response closely. The major discrepancies have shown to be less noisy sensor data and different latencies in the system response. However, this does not decrease our system stability because of the abstraction levels.

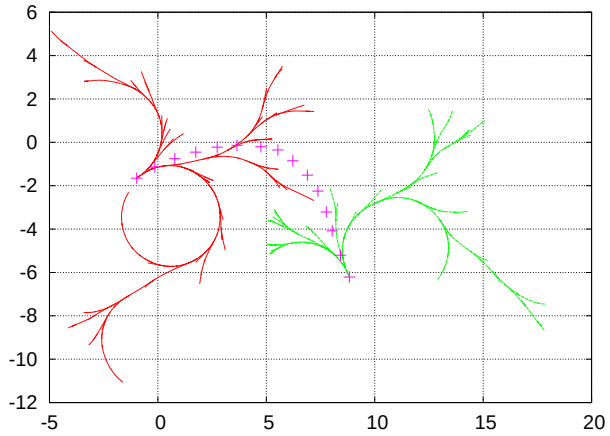
3.1 Path Planning

We have tested the RRT path planner extensively and have noticed a few drawbacks. Because of the randomness of the algorithm, the trajectories created by planner do not follow the desired way-points, which influences the efficiency of the coverage. Furthermore, we limit the number of iterations RRT has to grow its trees to ensure real-time performance. As a consequence, RRT occasionally does not complete a path connecting the way-points.

These issues are illustrated in Figure 7, where in (A), the generated path diverges from the desired way-points, and in (B), RRT fails to find a path in time.



(A)



(B)

Figure 7: The desired way-points are depicted with magenta crosses, the forward and reverse trees are in red and green, and the resulting path is in blue. (A) RRT created path differs slightly from the way-points but still avoids obstacles. (B) RRT failed to compute a path given a limited number of iterations.

3.2 Obstacle Avoidance

We tested the performance of the obstacle avoidance using VFF alone versus using both VFF and RRT generated paths in this experiment.

We commanded the robot to drive $11\,m$ passing a $1\,m \times 4.5\,m$ wall. Starting at a distance of $2.5\,m$ from one end of the wall and shifting down by $1\,m$ towards the wall for every loop. In the first test the RRT path planner was disabled. The path given to the controller to follow was a straight line containing 11 way-points with $1\,m$ separation. In the second test the RRT path planner was used to calculate a path around the obstacle. Figure 8 depicts the results from the two runs.

Using only reactive obstacle avoidance the robot was able to drive 4 loops. Without using RRT, the robot is

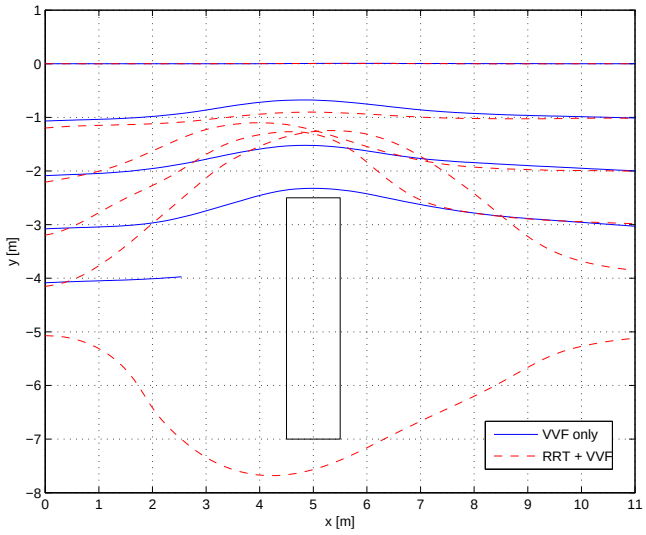


Figure 8: Robot path passing a wall using reactive obstacle avoidance only (solid blue lines), and using paths calculated by RRT (red dashed lines).

unable to navigate around the obstacle on the fifth run and hence stops in front of it. With RRT, the robot is able to successfully navigate around the obstacle in all passes, however, the paths are more random and they adhere less to the desired path.

Figure 9 shows a snap shot of the virtual forces from the obstacles acting on the robot as it is passing a wall. In this case, forces from the 9 objects are being exerted on robot body regions 5, 6 and 7 (See Figure 5).

3.3 Integration Test

We have tested the complete system outdoors in our test site with our robotic mower. The test site is an industrial compound with an area of approximately 30 by 40 meters containing obstacles of various sizes.

The outermost path was driven manually and the waypoints for subsequent loops were generated as described in Section 2.6. Sections of the path which were determined to be too close to obstacles detected from the laser scans, were sent to the local planner. The VFF was used continuously to ensure that sufficient clearance to the obstacles was achieved.

Figure 10 displays the resultant maps from the experimental run. The laser scan points and path sections are colored based on the local map in which they are contained. The tractor's coverage width is depicted as a transparent strip so that gaps and overlaps are visible. For illustrative purposes only, an occupancy grid has been created from the data from *all* the maps over which the results are plotted. It should be noted that even though the plot presents everything in a single coordinate frame, the internal representation uses separate

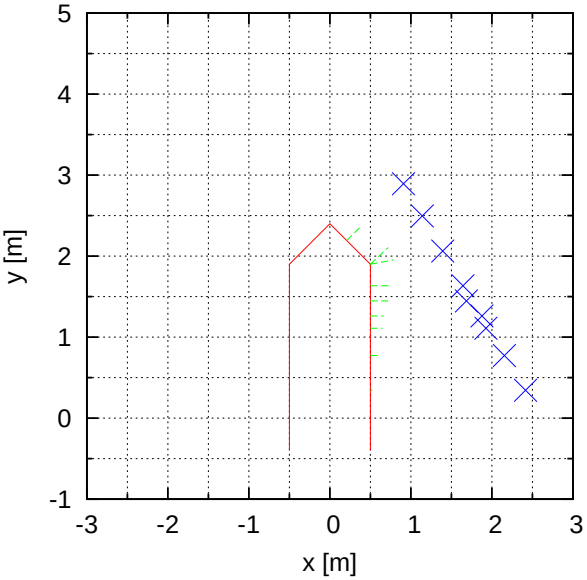


Figure 9: Obstacles in direct vicinity of the robot are shown with blue crosses. The repulsive forces, their magnitude and point of exertion on the body points, are shown shown with green lines. The robot and its virtual front point is shown in red.

coordinate frames for each local map.

For this run, a total of 6 local maps were generated. Each map stores a maximum of 12 laser scans at 1 meter intervals. We can see how the generated paths avoid the obstacles in the environment, but occasional gaps are introduced in the coverage from the randomness of the RRT generated paths. The system is quite stable in the presence of the increased amount of noise present in real data as opposed to our simulator results.

See Video 1 for a short clip showing the tractor navigating around the obstacles during its coverage task.

4 Conclusions and Future Work

In this paper, we have described a layered system architecture suitable for autonomous coverage tasks. The architecture integrates SLAM with motion planning using perceptive and executive modules that communicate via a network transparent shared memory interface.

The perceptive modules build up increasing levels of abstractions from the data describing the robot's environment, whereas, the executive modules break down the task coverage goal into planned motion paths, obstacle avoidance and actuator control.

We have shown how to integrate pure-pursuit path following with virtual force field obstacle avoidance. The robotic testbed has been demonstrated in simulation and with real world experiments, generating laser-based maps and avoiding obstacles while planning spiral paths

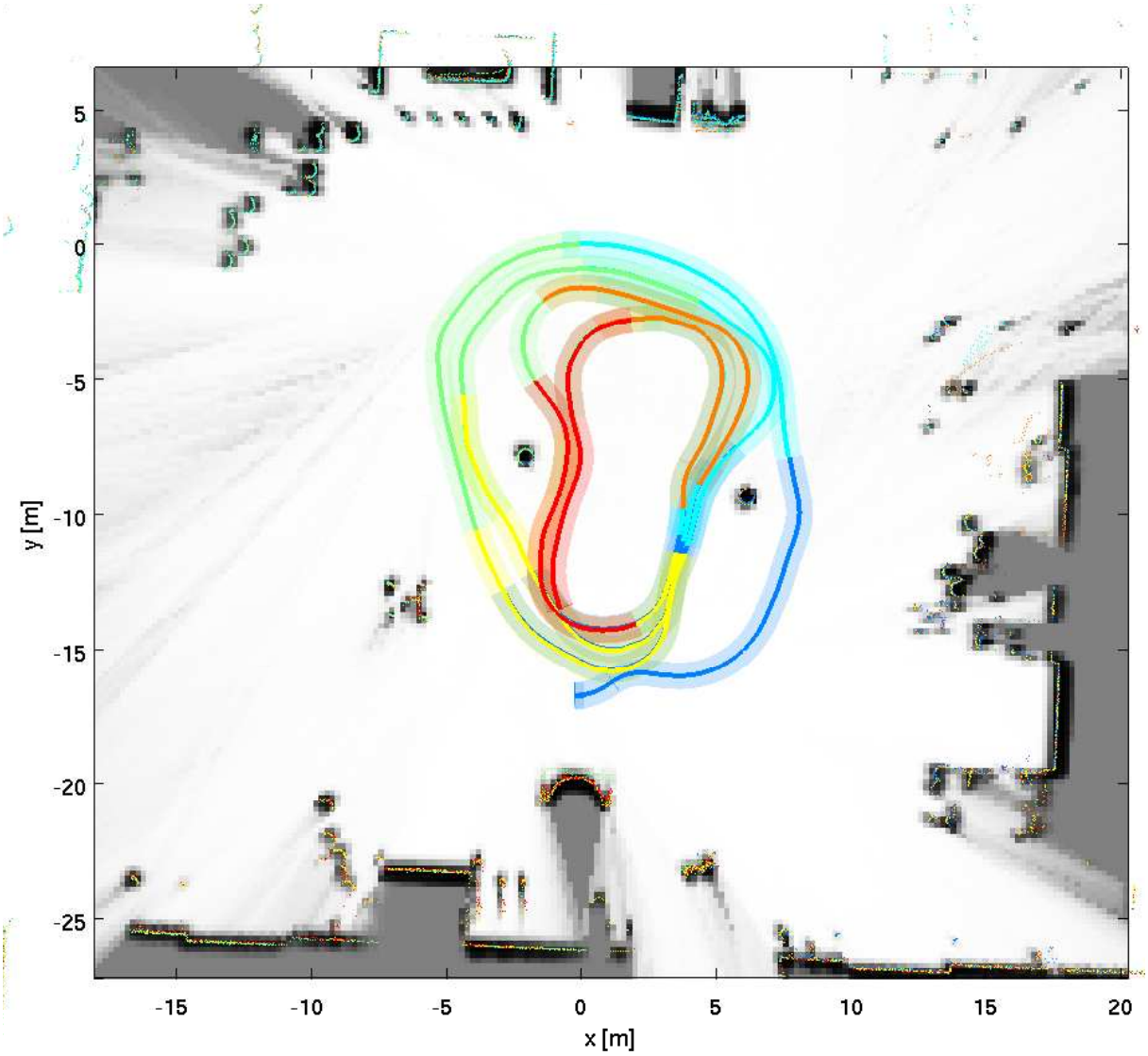


Figure 10: Resultant coverage paths and laser maps. The occupancy map is post generated and is a combination of all the coordinate frames for each of the local maps.

in real-time.

Future work will comprise of improving the RRT-based local path planner to follow the desired paths more closely, and to extend the global path planner to generate paths that can completely cover a given area while taking into account all of the vehicle's kinematic constraints.

Acknowledgments

This work was funded by the CSIRO ICT Centre under the ROVER and Dependable Field Robotics projects. The authors would like to thank the Autonomous Systems Laboratory team for their support of this work. Special thanks go to Polly Alexander, Stephen Brosnan, Peter Corke, Elliot Duff, Paul Flick, Leslie Overs, Cedric Pradalier, Ashley Tews, Kane Usher, John Whitham and Graeme Winstanley who all contributed to the development of our experimental autonomous tractor.

References

[Acar and Choset, 2002] E. U. Acar and H. Choset. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *International Journal of Robotics Research*, 21:345–366, April 2002.

[Batavia *et al.*, 2002] Parag Batavia, Stephan A. Roth, and Sanjiv Singh. Autonomous Coverage Operations in Semi-Structured Outdoor Environments. In *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '02)*, October 2002.

[BCC, 2006] BCC. Brisbane city council budget 2006-07. Internet: http://www.brisbane.qld.gov.au/bccwr/lib179/budget0607_natural_environment_sustainability_parks.pdf, 2006.

[Beeline Technologies, 2006] Beeline Technologies. Beeline Arro. Internet: www.beeline.com.au, 2006.

[Borenstein and Koren, 1989] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. In *IEEE Transactions on Systems, Man, and Cybernetics*, volume 19, pages 1179–1187, Sep/Oct 1989.

[Bosse *et al.*, 2003] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An Atlas Framework for scalable mapping. In *International Conference on Robotics and Automation*, pages 1899–1906, Taipei, Taiwan, September 2003.

[Bosse *et al.*, 2004] M. Bosse, P. Newman, J. Leonard, and S. Teller. Simultaneous localization and map building in large-scale cyclic environments using the Atlas Framework. *International Journal of Robotics Research*, 23(12):1113–1139, December 2004.

[Bosse, 2004] M. C. Bosse. *ATLAS: A Framework for Large Scale Automated Mapping and Localization*. PhD thesis, Massachusetts Institute of Technology, Feb 2004.

[Corke *et al.*, 2004] Peter Corke, Pavan Sikka, Jonathan Roberts, and Elliot Duff. DDX: A distributed software architecture for robotic systems. In *Proc. Australian Conf. Robotics and Automation*, Canberra, December 2004.

[Dunbabin *et al.*, 2004] Matthew Dunbabin, Jonathan Roberts, Kane Usher, and Peter Corke. In the Rough: In-Field Evaluation of an Autonomous Vehicle for Golf Course Maintenance. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3339–3344, Sendai, Japan, Sep/Oct 2004.

[FriendlyRobotics, 2006] FriendlyRobotics. Robomower RL1000. Internet: www.friendlyrobotics.com, 2006.

[Howard, 2006] Andrew Howard. Gazebo 3D Robot Simulator. Internet: player-stage.sourceforge.net/gazebo/gazebo.html, 2006.

[ION, 2006] ION. ION Autonomous Lawnmower Competition. Internet: www.automow.com, 2006.

[Jarvis, 2001] Ray Jarvis. A Tele-Autonomous Heavy Duty Robotic Lawn Mower. In *Proceedings of the Australian Conference on Robotics and Automation*, Nov 2001.

[LaValle and Kuffner, 1999] S. M. LaValle and J. J. Kuffner. Random kinodynamic planning. In *IEEE International Conference on Robotics and Automation*, pages 473–479, 1999.

[Mowbot, 2006] Mowbot. Mowbot - The Robotic Mower. Internet: www.mowbot.com, 2006.

[Noonan *et al.*, 1993] T. Noonan, J. Fisher, and B. Bryant. Autonomous Lawn Mower. US Patent No. 5204814, April 1993.

[Usher, 2005] Kane Usher. *Visual homing for a car-like vehicle*. PhD thesis, Queensland University of Technology, Brisbane, Australia, June 2005.

[Wallace *et al.*, 1995] R. Wallace, K. Matsuzaki, and Y. Goto. Progress in robot road following. In *Proceedings of the International Conference of Robotics and Robotics*, pages 1615–1621, 1995.