A Framework for Mobile Robot Concurrent Path Planning & Execution in Incomplete & Uncertain Environments

John S. Zelek
Engineering Systems & Computing,
School of Engineering, University of Guelph,
Guelph, ON, N1G 2W1, Canada
e-mail: jzelek@uoguelph.ca
www: http://131.104.80.10/webfiles/zelek/

Abstract

Sensor-based discovery path planning is problematic because the path needs to be continually recomputed as new information is discovered. A process-based client-server approach is presented that permits concurrent sensor-based map and localization-correction updates as well as concurrent path computation and execution. Laplace's equation is constantly solved (i.e., a harmonic function) by using an iteration kernel convolved with an occupancy-grid representation of the current free space. The path produced (i.e., by steepest gradient descent on the harmonic function) is optimal in the sense of minimizing the distance to the goal as well as a hitting probability. This helps alleviate the influence of uncertainty on path planning. An initial heuristic estimate provides the path planner with instantaneous response (i.e., reactive), but with some deliberation it is able to produce optimal paths. In addition, the computation time for generating the path is insignificant provided that the harmonic function has converged. On a regular grid, the computation of the harmonic function is linear in the total number of grid elements. A quad-tree representation is used to minimize the computation time by reducing the number of grid elements and minimally representing large spaces void of obstacles and goals.

Introduction

Sensor-based discover path planning is the guidance of an agent - a robot - without a complete a priori map, by discovering and negotiating with the environment so as to reach a goal location while avoiding all encountered obstacles. A robot must not only be able to create and execute plans, but must be willing to interrupt or abandon a plan when circumstances demand it. In traditional AI planning, a smart planning phase constructs a plan which is carried out in a mechanical fashion by a dumb executive phase. The use of plans regularly involves rearrangement, interpolation, disambiguation, and substitution of map information. Real situations are characteristically complex, uncertain, and immediate. These situations require that the planning and the execution phases function in parallel, as opposed to the serial ordering found in traditional AI planning.

Path planning can be categorized as either being *static* or *dynamic*, depending on the mode of available information (Hwang & Ahuja 1992). A *static* path planning strategy is used when all the information about the obstacles is known a priori. Most path planning methods are static. Dynamic path planning is also referred to as being *discovery and on-line planning* (Halperin, Kavraki, & Latombe 1997). In this problem space, the workspace is initially unknown or partially known. As the robot moves, it acquires new partial information via its sensors. The motion plan is generated using partial information that is available and is updated as new information is acquired. Typically this requires an initial planning sequence and subsequent re-planning as new information is acquired.

A path planning framework has been proposed and implemented as part of the SPOTT mobile robot control architecture (Zelek 1996). Within SPOTT's framework there are two concurrently executing planning modules. Planning is defined along a natural division: (1) local, high resolution, and quick response, versus (2) global, low resolution, and slower response. The *local* planning module executes a path based on what is currently known about the environment as encoded in an occupancy grid (Elfes 1987). A potential field technique using harmonic functions is used, which are guaranteed to have no spurious local minima (Connolly & Grupen 1993). A harmonic function is the solution to Laplace's equation. Any path - determined by performing gradient descent - has the desirable property that if there is a path to the goal, the path will arrive there (Doyle & Snell 1984). In addition, the steepest descent gradient path is also optimal in the sense that it is the minimum probability of hitting obstacles while minimizing the traversed distance to a goal location (Doyle & Snell 1984). This optimality condition permits modeling uncertainty with an error tolerance (i.e., obstacle's position and size, robot's position) provided that the robot is frequently localized and obstacle positions and size are validated via sensor information.

The computation of the harmonic function is expensive, being linear in the number of grid points. The presented technique is a *complete algorithm* (i.e., also referred to as *exact algorithm*): guaranteed to find a path between two configurations if it exists. A complete algo-

rithm usually requires exponential time in the number of degrees-of-freedom (Halperin, Kavraki, & Latombe 1997). The approach is confined to a 2D workspace, with the emphasis being on obtaining real-time[1] performance in this dimensional space with future prospects of adaptation to higher dimensionalities.

A client-server process model is used to concurrently compute the potential field (i.e., compute the potential paths to the goal) and execute the path. The *global* planning module uses a search algorithm (i.e., Dijkstra's algorithm) on a graph structure representation of the map. The role of the global planning module is to provide global information to the local path planner. This is necessary because the spatial extent of the local path planner is limited due to its computational requirements increasing proportionally with the number of grid elements.

A typical operational scenario within SPOTT (Zelek 1996) can be presented in the following manner. The goal specification is given as input by the user. If an a priori CAD map exists, it is loaded into the *map database*. The role of the control programs (i.e., behavioral controller) is to provide the local path planner (i.e., potential field) with the necessary sensor-based information for computation, such as obstacle configurations, current robot position, subgoals if necessary, in addition to performing reactive control. The behavioral controller continually computes mapping and localization information by processing sensor data. As the path is being computed, another module is responsible for concurrently performing steepest gradient descent on the potential function in order to determine the local trajectory for the robot.

Background

A type of *heuristic algorithm* for performing dynamic path planning is the potential field technique (Khatib 1986). This technique offers speedup in performance but cannot offer any performance guarantee. In addition, these algorithms typically get stuck in local minima. Techniques have been devised for escaping these local minima such as the *randomized path planner* (Barraquand & Latombe 1991) which escapes a local minimum by executing random walks at a cost of computational efficiency.

A complete algorithm for performing dynamic path planning is the D^* algorithm (Stentz 1995). D^* (i.e., dynamic A^*) performs the A^* algorithm initially on the entire workspace. As new information is discovered, the algorithm incrementally repairs the necessary components of the path. This algorithm computes the shortest distance path to the goal with the information given at any particular time. One problem with the D^* algorithm is that no allowances are made for uncertainty in the sensed obstacles and position of the robot.

[1]Real-time is defined as responding to environmental stimuli faster than events change in the environment.

An approach that approximates being a complete algorithm is *probabilistic planning* (Barraquand *et al.* to appear). Probabilistic planning is generally used for multi-link manipulators and when the configuration space has more than five degrees of freedom. In this approach, the first step is to efficiently sample the configuration space at random and retain the free configurations as milestones. The next step checks if each milestone can be circumvented by a collision free path, producing a probabilistic roadmap. A harmonic function can be computed using the milestones as grid elements. It is argued in this paper that a quad-tree representation can permit the use of a complete approach without resorting to probabilistic techniques even if it is just for 2D planning.

Local Path Planning

SPOTT's local path planner is based on a potential field method using harmonic functions, which are guaranteed to have no spurious local minima (Connolly & Grupen 1993). A harmonic function on a domain $\Omega \subset R^n$ is a function which satisfies Laplace's equation:

$$\nabla^2 \phi = \sum_{i=1}^{n} \frac{\delta^2 \phi}{\delta^2 x_i^2} = 0 \qquad (1)$$

The value of ϕ is given on a closed domain Ω in the configuration space C. A harmonic function satisfies the *Maximum Principle*[2] and *Uniqueness Principle*[3] (Doyle & Snell 1984). The *Maximum Principle* guarantees that there are no local minima in the harmonic function.

Iterative applications of a finite difference operator are used to compute the harmonic function because it is sufficient to only consider the exact solution at a finite number of mesh points[4]. In addition, gradients for path computation are solicited during the iterative computation without concern for obstacle collision[5]. Linear interpolation is used to approximate gradients when the position of the robot is in between mesh points. The obstacles and grid boundary form boundary conditions (i.e., Dirichlet boundary conditions are used) and in the free space the harmonic function is computed using an iteration kernel, which is formed by taking advantage of the harmonic function's inherent averaging property, where a point is the average of its neighboring points. A five-point iteration kernel (Hornbeck 1975) for solving Laplace's equation is given as follows:

$$U_{i,j} = \frac{1}{4}(U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1}) \qquad (2)$$

[2]A harmonic function $f(x,y)$ defined on Ω takes on its maximum value M and its minimum value m on the boundary.

[3]If $f(x,y)$ and $g(x,y)$ are harmonic functions on Ω such that $f(x,y) = g(x,y)$ for all boundary points, then $f(x,y) = g(x,y)$ for all x, y.

[4]As opposed to finite element methods which compute an approximation of an exact solution by continuous piecewise polynomial functions.

[5]Obstacles are not in the free space and are not valid positions for the robot.

A more accurate nine-point iteration kernel can also be used (van de Vooren & Vliegenthart 1967):

$$U_{i,j} = \frac{1}{5}(U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1}) + \frac{1}{20}(U_{i+1,j+1} + U_{i-1,j+1} + U_{i-1,j-1} + U_{i+1,j-1}) \quad (3)$$

The computation of the harmonic function can be formulated with two different types of boundary conditions. The *Dirichlet* boundary condition[6] is where u is given at each point of the boundary. The *Neumann* boundary condition is where $\frac{\delta u}{\delta n}$, the normal component of the gradient of u, is given at each point of the boundary. In order to have flow, there has to be a source and a sink. Thus, the boundary of the mesh is modeled as a source, and the boundary of the goal model is modeled as a sink. The boundaries of the obstacles are modeled according to the type of boundary condition chosen: Dirichlet or Neumann. The path - determined by performing steepest gradient descent on a harmonic function using Dirichlet boundary conditions - is also optimal in the sense that it is the minimum probability of hitting obstacles while minimizing the traversed distance to a goal location (Doyle & Snell 1984). Neumann boundary conditions cause such a path to graze obstacle boundaries. This optimality condition permits modeling uncertainty with an error tolerance (i.e., obstacle's position and size, robot's position) provided that the robot is frequently localized using sensor data. The robot is modeled as a point but all obstacles are compensated for the size of the robot in addition to an uncertainty tolerance[7]. The computation of the harmonic function is performed over an occupancy grid representation of the local map. The computation of the harmonic function is executed as a separate process from the path executioner. In order to reduce the number of iterations to be linear in the the number of grid points, "*Methods-of-Relaxation*" (Ames 1992) techniques such as Gauss-Seidel iteration with Successive Over-Relaxation, combined with the "*Alternating Direction*" (ADI) method were used. A local window is used for computing the potential field to further limit the computation time, and because of the rapid decay of the harmonic function, especially in in narrow corridors[8]. A global path planner (discussed later) provides information about goals outside the local bounds. The process computing the iteration kernels and servicing requests from the executioner and sensor updates uses the following algorithm:

1. Loop 1: Load initial map if available.
2. If the goal is outside the bounds of the potential field, project the goal onto the border.
3. Loop 2:
   (a) Compute an averaging iteration kernel over the entire grid space.
   (b) Poll to see if a new robot position estimate is available. If so, correct the accumulated error.
   (c) Poll to see if any newly sensed data is available. If so, update the map.
   (d) Update the robot's position based on the odometer sensor.
   (e) Poll to see if the execution process module requests a new steering direction for the robot. If so, compute it by calculating the steepest descent gradient vector at the current estimate of the robot's position.
   (f) If the robot is near the edge of the border of the potential field window, move the window so that the robot is at the center and go to Loop 1, else go to Loop 2.

This computation has the desirable property that map features can be added or subtracted dynamically (i.e., as they are sensed) alongside with the correction of the robot's position. In addition, these events are independent of the computation and the execution of the path.

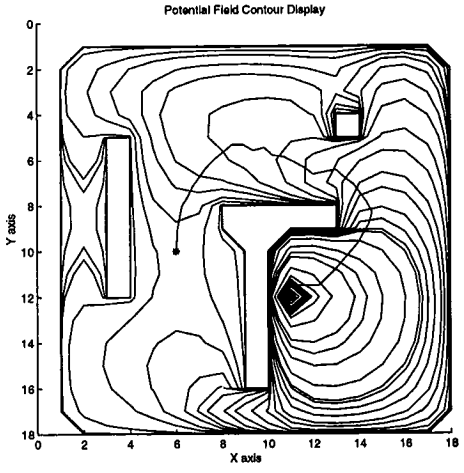

Figure 1: **Equal-potential contours.** of a computation converging towards the harmonic function solution. A path executed at this instance shows some irregularities from a smooth curve. After convergence, the path is smooth, albeit at the resolution of the discretization.

Another approach to speed up performance is to provide an initial guess to the solution, namely the heuristic potential field (Khatib 1986). This makes the local path planner an iterative-refinement anytime approach (Zilberstein 1996) where after a short period after a map change, response is reactive but not optimal or guaranteed; and after convergence is achieved, response is optimal (see Figure 1). Achieving fast convergence will not necessarily be possible or realistic if the path

[6]This inherently makes all applicable boundary points into sources (i.e., in terms of sources and sinks for modeling liquid flow).

[7]This may appear to be heuristic, but it is based on not permitting the robot's position error to grow beyond a certain fixed limit (i.e., by re-localizing).

[8]On a thirty-two bit computer, the harmonic function can only be accurately represented when the length-to-width ratio for a narrow corridor is less than 7.1 to 1 (Zelek 1996).

planning is done in more dimensions than two. A possible approach to take in this scenario is to extend the approach to include *probabilistic planning* (Barraquand *et al.* to appear).

There is no need to compute the harmonic function at a fine grid sampling over the local window especially in large open spaces that are void of any obstacle or goal models. The most efficient irregular grid sampling to implement on a computer is the quad-tree representation. Pyramids and quad-trees are a popular data structure in both graphics and image processing (Pavlidis 1982) and are best used when the dimensions of the matrix can be recursively evenly subdivided until one grid point represents the entire image. Approximately 33% more nodes are required to represent the image but usually the algorithm will be a lot faster, especially if there are large open spaces. Quad-trees have also been previously used for path planning (Chen, Szczerba, & Uhran 1997). Most approaches have usually applied the A* search strategy to find the shortest distance between two points but application of A* in a quad-tree representation does not necessarily produce the shortest distance path. This problem was overcome by making a slight modification to the quad-tree representation by using a framed-quad-tree representation (Chen, Szczerba, & Uhran 1997).

Computing the harmonic function with a quad-tree representation bears some resemblance to the multi-grid methods (Ames 1992) for computing solutions to differential equations with the slight difference that in this approach there is no need to compute all of the grid locations at their highest resolution. To take into account the irregular grid spacing in the quad configuration, the iteration kernel is redeveloped from basic principles (i.e., Taylor series expansion resulting in forward and difference equations) and can be rewritten as follows (given that h_r is the distance from $U_{i,j}$ to $U_{i+1,j}$, h_l is the distance from $U_{i,j}$ to $U_{i-1,j}$, k_d is the distance from $U_{i,j}$ to $U_{i,j-1}$, and k_t is the distance from $U_{i,j}$ to $U_{i,j+1}$) (see Figure 2a):

$$U_{i,j} = \frac{k_d k_t (h_l U_{i+1,j} + h_r U_{i-1,j}) + h_r h_l (k_t U_{i,j+1} + k_d U_{i,j-1})}{k_d k_t (h_r + h_l) + h_r h_l (k_t + k_d)} \quad (4)$$

An alternative method of deriving the iteration kernel for non-uniform grid sampling is to analyze the grid network as a resistor network (Doyle & Snell 1984) and to also draw this interpretation into the language of a random walker on a collection of grid points. The distances between grid points are resistor values and the voltage to the circuit is applied across the obstacle and goal locations. The interpretation of the voltage in terms of a random walker is (Doyle & Snell 1984):
When a unit voltage is applied between a (i.e, obstacles) and b (i.e., goal(s)), making $v_a = 1$ and $v_b = 0$, the voltage v_x at any point x represents the probability that a walker starting from x will return to a before reaching b.

Steepest gradient descent on the collection of voltage points corresponds to minimizing the hitting probability. An interpretation of the current in terms of a random walk can be given as follows (Doyle & Snell 1984):
When a unit current flows into a and out of b, the current i_{xy} flowing through the branch connecting x to y is equal to the expected number of times that a walker, starting at a and walking until he(she) reaches b, will move along the branch from x to y.

Figure 2a illustrates the resistor circuit equivalent used to derive Equation 4. The equation was derived by applying Kirchoff's Current Law[9]. Figure 2b illustrates that this method can be applied to any irregular grid configuration; for example, the sampled configuration space used in probabilistic planning (Barraquand *et al.* to appear). For any irregular grid sampling, the iteration kernel can be rewritten as shown in Equation 5, where U_n are neighboring points to $U_{i,j}$, R_n distance away.

$$\sum_{n=-d}^{n=d} \frac{U_{i,j}}{R_n} = \sum_{n=-d}^{n=d} \frac{U_n}{R_n} \quad (5)$$

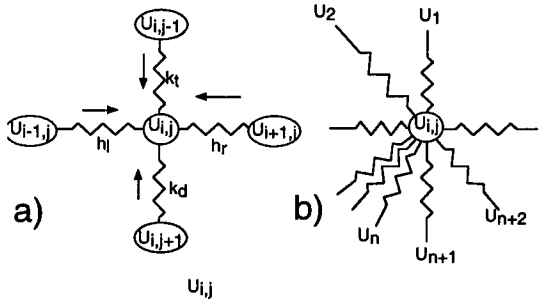

Figure 2: **Resistor Circuit Equivalent.** (a) The resistor circuit equivalent used to derive Equation 4. (b) This method can be applied to any irregular grid sampling by applying Kirchoff's Current Law and equating resistor values with the corresponding distance values (see Equation 5).

Again, linear interpolation is used to approximate gradients when the position of the robot is in between grid points. This introduces some error in the minimal hitting path but the benefit in significantly less computations provides adequate justification for this trade-off. In addition, the introduced error has minimal effect in increasing the chance of collision because the error is in the large grid locations which are void of obstacles and goals (see Figure 3). The quad-tree representation for a particular configuration (see Figure 3) resulted in a thirty-six-fold savings in computational time in achieving convergence.

[9]Kirchoff's Current Law says that the total current flowing into any point other than the points where the voltage is applied across (i.e., a or b) is 0. Current (i.e., I) is calculated as $\frac{V_i - V_j}{R_{ij}}$, where V_i and V_j are the voltages on either side of the resistor R_{ij}.

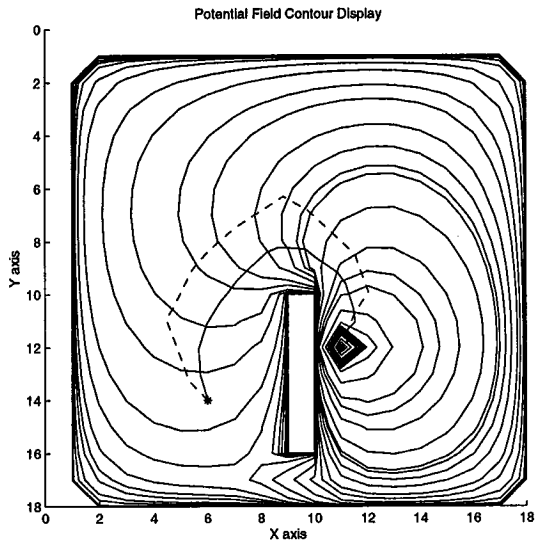

Figure 3: **Path for Quad Representation.** The quad interpolated path (dotted) differs slightly from the path produced on a regular-spaced grid.

Global Path Planning

The global planning module performs search (i.e., deliberation) using Dijkstra's algorithm (Aho, Hopcroft, & Ullman 1983) through a graph structure of nodes and arcs (see Figure 4 and 5). This is only possible if a CAD map is available a priori. Automating the creation of the graph map from the CAD map and the creation of a CAD map from sensor data are both left as future research endeavors. Initially it is assumed that all doors are open until this assumption is refuted by sensor data. A graph abstraction of the CAD map is obtained by assigning nodes to rooms and hallway portions and edges to the access ways (i.e., doors) between nodes. The global graph planner determines a path based on start and stop nodes defined by the current location and desired destination of the robot. If the current goal is outside the extent of the local path planner, then the global path planner projects the goal onto the local map's border as follows:

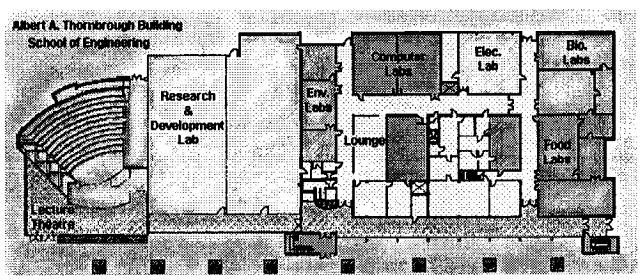

Figure 4: **CAD map** of the first floor of the School of Engineering. The accompanying graph map is in Figure 5.

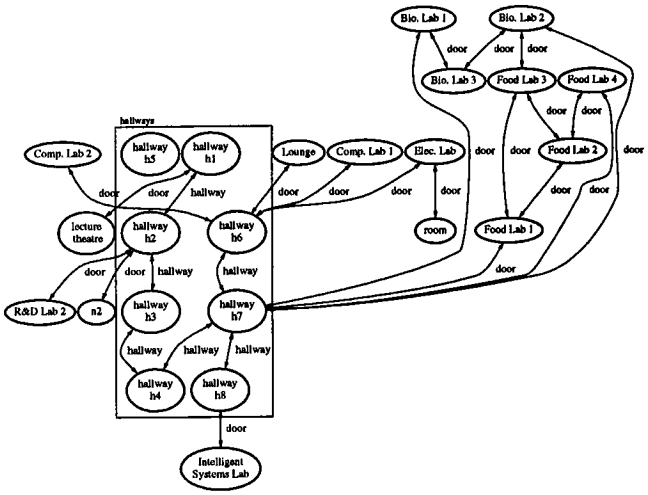

Figure 5: **Floor Plan as Graph.** The graph shows a partial section of the CAD map in Figure 4.

Let $p_g = (x_g, y_g)$ define the position of the goal in an anchored x-y spatial coordinate system (i.e., SPOTT's experiments used cm. as a unit of measure). Let $p_r = (x_r, y_r)$ define the current position of the robot. The local map is a collection of grids (i.e., nodes) but the references into this grid are always continuous rather than discrete. For example, the gradient at a particular position is calculated by interpolating amongst the neighboring grid elements. Let n_r be the node in the global map where the robot is located (i.e., p_r) and n_g be the node where the goal is located (i.e., p_g). Let $b_{pf} = ((x_1, y_1), (x_2, y_2))$ define the top-left and bottom-right points designating the spatial extent of the potential field. By definition, initially p_r is in the center of the region defined by b_{pf}. The global path produced by the global path planner is given in the following form: $< n_r, e_i, n_j, ..., n_{m-1}, e_{l-1}, n_m, e_l, n_g >$. The following is the algorithm (see Figure 6) for projecting the global goal onto the border of the potential field:

1. If p_g is within b_{pf}, and n_g is the same as n_r then do nothing.
2. If p_g is within n_r and p_g is not within b_{pf}, then the goal is projected onto the portion of the side of b_{pf} that is entirely contained within n_r.
3. Let e_k be the first edge within the global path (i.e., moving from n_r towards n_g) that is not completely within b_{pf}. Let p_k be the intersection of b_{pf} with the line connecting the center of the edge e_k with the center of the node (i.e., n_k) last traversed before reaching that edge. The goal is projected onto the portion of the side of b_{pf} that contains both the node n_k and the point p_k.

In each of the three cases there are situations where the goal will not be reachable. In the third case, the goal will not be reachable if a doorway is closed or a node is blocked by an obstacle. In this case, the global path planner replans the global path. In the first two cases, the goal may not be reachable within the spatial extent of the potential field due to newly discovered obstacles, but the goal may actually be physically reachable. The easiest way to deal with this situation is to make the size

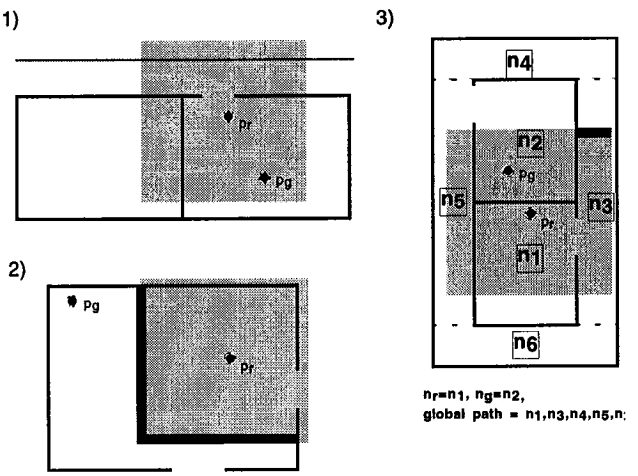

Figure 6: **Projecting Global Goal onto Local Limits.** The three cases as presented by the algorithm in the text are presented. The shaded area corresponds to the extent of the local path planner: b_{pf}. The thick line corresponds to the projection of the goal onto the local path planner's borders. The thin lines represent wall features. In (1), the goal and robot are in the same node and b_{pf}. In (2), the robot and goal are in the same node but not the same b_{pf}. (3) is the most general case.

of the potential field such that the node is completely within the potential field's spatial extent.

Discussion

Experiments were performed using a Nomadics 200 mobile robot. The map was updated with sonar (and laser) range lines at a regular frequency. The sonar range lines were produced by fusing sonar data points and removing outliers (Mackenzie & Dudek 1994). It was found that the sensing range of the sensors (i.e., approximately 5 m) permitted the robot to move at speeds up to 50 cm/sec. By this time the harmonic function had converged and the path was optimal.

In addition, it was found that in highly structured environments (e.g., narrow hallways) it is computationally advantageous to use just behavioral rules for navigation (e.g., move forwards and stay centered in the hallway) as opposed to requiring the full computational power of the local path planner. Typically if an obstacle is discovered in a narrow hallway, the hallway is blocked from passage and a new global path needs to be recomputed, thus negating the need for the local path planner (i.e., the obstacle cannot be circumvented anyways, the hallway is blocked).

The technique provides a viable method for concurrently computing and executing the path for a mobile robot, as well as mapping and localizing the robot via sensor information at the same time. A significant reduction in required computational time was achieved by using the quad-tree representation for computing the harmonic function (i.e., for path planning). The results are encouraging for continued use of the described client-server process model for real-time motion control of a mobile robot. Actual deployment of this technique requires satisfying the assumption that adequate perceptual routines are available (i.e., based on sonar, range, and vision sensors) or developed. The presented technique also appears adaptable for use in path planning using the probabilistic planning methodology by using the sampled configuration space as grid elements.

Future Issues

Future work includes addressing the representation of moving obstacles and goals, sensor fusion, as well as an extension to a multi-dimensional configuration space.

An occupancy grid representation simplifies the sensor fusion problem (Elfes 1987) at the expense of limiting accuracy to the grid's resolution. With the foresight of potentially including map building capabilities into the SPOTT architecture (i.e., the architecture in which the presented path planning method has been tested), an equivalent vector representation of the occupancy grid is maintained and cross-referenced (at the expense of complicating sensor fusion). In addition, the associated vectors (i.e., typically range line segments) may not directly correspond to physical objects. Addressing the issue of sensor fusion within the given framework (if detailed map building functionality is required) may be difficult because fusing data together will typically take away computation time from the main processing loop in which the harmonic function is computed.

A similar problem may hold for updating moving grid elements. Typically a collection of grid elements will move together as a rigid body. In order to limit the affect of updating moving obstacles has on the iterative computation, certain courses of action such as limiting the refresh rate of moving obstacles and how many obstacles can move are possible avenues of exploration. This assumes that adequate perceptual capabilities are present so as to update and predict obstacle (and potentially goals for tracking) trajectories.

References

Aho, A. V.; Hopcroft, J. E.; and Ullman, J. D. 1983. *Data Structures and Algorithms.* Addison-Wesley Publishing Co.

Ames, W. F. 1992. *Numerical Methods for Partial Differential Equations.* Academic Press Inc.

Barraquand, J., and Latombe, J.-C. 1991. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research* 10:628–649.

Barraquand, J.; Kavraki, L.; Latombe, J.; Li, T.; Motwani, R.; and Raghavan, P. to appear. A random sampling scheme for path planning. *International Journal of Robotics Research.*

Chen, D. Z.; Szczerba, R. J.; and Uhran, J. J. 1997. A framed-quadtree approach for determining euclidean shortest paths in a 2-d environment. *IEEE Transactions on Robotics and Automation* 13(5):668–681.

Connolly, C. I., and Grupen, R. A. 1993. On the applications of harmonic functions to robotics. *Journal of Robotic Systems* 10(7):931–946.

Doyle, P. G., and Snell, J. L. 1984. *Random Walks and Electric Networks.* The Mathematical Association of America.

Elfes, A. 1987. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation* 3:249–265.

Halperin, D.; Kavraki, L.; and Latombe, J.-C. 1997. Robotics. Boca Raton, FL: CRC Press. chapter 41, 755–778.

Hornbeck, R. W. 1975. *Numerical Methods.* Quantum Publishers Inc.

Hwang, Y. K., and Ahuja, N. 1992. Gross motion planning - a survey. *ACM Computing Surveys* 24(3):220–291.

Khatib, O. 1986. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research* 5(1):90–98.

Mackenzie, P., and Dudek, G. 1994. Precise positioning using model-based maps. In *IEEE International Conference on Robotics and Automation*, volume 2. San Diego, CA: IEEE. 11615–1621.

Pavlidis, T. 1982. *Algorithms for Graphics and Image Procesiing.* Computer Science Press.

Stentz, A. 1995. The focussed d* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence.*

van de Vooren, A., and Vliegenthart, A. 1967. On the 9-point difference formula for laplace's equation. *Journal of Engineering Mathematics* 1(3):187–202.

Zelek, J. S. 1996. *SPOTT: A Real-time Distributed and Scalable Architecture for Autonomous Mobile Robot Control.* Ph.D. Dissertation, McGill University, Dept. of Electrical Engineering.

Zilberstein, S. 1996. Using anytime algorithms in intelligent systems. *Artificial Intelligence Magazine* 73–83.