Path Planning Using Laplace's Equation

C. I. Connolly J. B. Burns
R. Weiss
Computer and Information Science Department,
University of Massachusetts at Amherst *†

February 7, 1994

Abstract

A method for planning smooth robot paths is presented. The method relies on the use of Laplace's Equation to constrain the generation of a potential function over regions of the configuration space of an effector. Once the function is computed, paths may be found very quickly. These functions do not exhibit the local minima which plague the potential field method. Unlike decompositional and algebraic techniques, Laplace's Equation is very well suited to computation on massively parallel architectures.

1 Introduction

The use of potential functions was introduced by Khatib [10] for robot path planning. The potential field method of path planning (as proposed by Khatib) envisions every obstacle as exerting a repelling force on an effector, while the goal exerts an attractive force. Other authors [2, 12, 17, 14, 1,

*Office of Naval Research URI N00014-86-K-0764, DARPA contract DACA 76-85-C-0008

†In Proc. 1990 IEEE Conference on Robotics and Automation. Figures were unavailable for this copy. Please see the proceedings for figures.

16, 11] have used a variety of potential functions, all based on this underlying scheme. The speed and facility of this method make it a useful tool for constructing paths for robots. The usual formulation of potential fields for path construction does not preclude the spontaneous creation of local minima other than the goal. The robot may "fall" into these minima and achieve a stable configuration short of the goal. Several authors have mentioned this problem ([2, 10, 1]).

Using geometrical arguments, Koditschek [11] showed that, at least in certain types of domains, there always exist potential functions which will guide an effector from almost any (i.e., all but a set of measure zero) starting point to a given goal point. Methods are given for constructing such functions, but only in spherical domains in which spherical obstacles are embedded.

This paper describes a global method which generates smooth collision-free paths. The method computes solutions to Laplace's Equation in arbitrary n-dimensional domains, and results in a weak form of what Rimon and Koditschek define as *navigation functions.*

2 Harmonic functions

A harmonic function on a domain $\Omega \subset R^n$ is a function which satisfies Laplace's equation:

$$\nabla^2 \phi = \sum_{i=1}^{n} \frac{\partial^2 \phi}{\partial x_i^2} = 0$$

. In the case of robot path construction, the boundary of this region consists of the boundaries of all obstacles and goals in a configuration space representation. Harmonic functions satisfy the min-max principle (see [20, 21]): Spontaneous creation of local mimima within the region is impossible if Laplace's equation is imposed as a constraint on the functions used.

The absence of local minima may be intuitively demonstrated by considering the two-dimensional version of Laplace's Equation:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = 0$$

Consider the two curves which are the x and y cross-sections of ϕ at some point p_0. If the second derivatives of ϕ are not zero at p_0, the two curves in question must have second derivatives of opposite sign. Assuming that $\phi \in C^2$, this implies that one curve must be concave upward, and the other must be concave downward. Thus we have that either ϕ is planar at p_0, or that there is a direction outward from p_0 in which ϕ decreases, and another in which ϕ increases. Therefore, in any open region where Laplace's equation holds, local extrema of ϕ cannot exist. More formally, if a function ϕ satisfies Laplace's equation on some region $\Omega \subset R^n$, then ϕ attains its minimum and maximum values *only* on the boundary $\partial\Omega$ of Ω. See, for example [20, chapter 12, exercise 4]. For a rigorous mathematical treatment, the reader is referred to any text on Fourier Analysis or Complex Variables [6, 7].

If a function satifies Laplace's equation in some region, then any critical points of the function in the interior of that region must be saddle points, since local extrema of the function are not possible there. If the effector reaches a saddle point, and it is not near the goal, then there must be a way out. This exit from this critical point may be found by performing a search in the neighborhood of the critical point. In the context of [18], the set from which no path can be generated from start to goal consists of these saddle points. Any perturbation from these points, however, results in a path (sometimes referred to as a *streamline*).

Rimon and Koditschek ([18]) define a *navigation function* as satisfying four properties. Every harmonic function ϕ defined on a compact region $\bar{\Omega} = \partial\Omega \cup \Omega$ satisfies three of the four properties for navigation functions. This may be seen as follows:

- Analyticity: Every harmonic function is analytic [3].
- Polar: Select a point q_d to be the goal point, constrain $\phi(q_d) = 0$ and set all obstacle boundary points p to some constant $\phi(p) = c$. Since all harmonic functions satisfy the min-max principle, ϕ is polar. In other words, q_d will be the point at which ϕ attains its minimum value on $\bar{\Omega}$. Hence, all streamlines of ϕ lead to q_d.
- Admissibility: If we simply set the constant $c = 1$ above, then ϕ will be admissible in the sense of [18]. This is a simple normalization of ϕ.

The fourth property is that a navigation function be Morse (i.e., there are no degenerate critical points) [15]. Here we simply note that every critical point of ϕ in Ω must be an isolated saddle point, from which streamlines may easily be found.

Harmonic functions can be combined using superposition to produce various flows. One such function, which represents a point source (whose potential is infinite at the point), is $\log(r(x,y))$, where $r(x,y)$ is the distance from the source point x_0, y_0. The gradient for this function represents the vector field which would drive an effector away from the obstacle point:

$$\log(r(x,y)) = \log(\sqrt{(x-x_0)^2 + (y-y_0)^2}) \tag{1}$$

$$\frac{\partial}{\partial x}\log(r(x,y)) = \frac{x-x_0}{r(x,y)} \tag{2}$$

$$\frac{\partial}{\partial y}\log(r(x,y)) = \frac{y-y_0}{r(x,y)} \tag{3}$$

Thus, the gradient for this function at a given point is always a unit vector in a direction *away* from the source point. Note also that the second partial derivatives with respect to x and y vanish everywhere, so Laplace's equation is satisfied. Since harmonic functions describe a variety of physical phenomena, many examples of such functions can be found in physics texts [8].

3 Superposition and Obstacle Avoidance

The superposition of harmonic functions presents problems. One such problem is that there is no guarantee that the obstacles will be avoided in complex or dynamic environments. The potential in the neighborhood of a given obstacle is a function not only of that obstacle's potential, but also of every other obstacles' (or goals') potentials. By changing the configurations or strengths (which can easily happen in dynamic environments) the path of the robot can be led arbitrarily close to the obstacle. In a sense, the only structure that can be safely modelled this way is a point itself since the potential goes to infinity at the point source, and thus superposition of fields associated with sources at a finite distance cannot affect this.

Extending the models of the obstacle to an infinitely dense collection of points (in the hope of avoiding paths "between points") also fails. For example, if the logarithmic expression for a point obstacle is integrated over a line, the potential along that line will be finite. This allows a sufficiently strong goal point to "wash over" the line obstacle, making it penetrable.

The problem with using superposition to construct control surfaces is not restricted to the integration of these particular point potentials, as can be

Figure 1: Four mazes with differing wall and door configurations. P is the effector position and arrows are desired directions of motion.

seen by the following argument. The task of getting out of a maze from any point within it is modelled with potentials by placing some sink (goal) external to the maze and requiring that the walls of the maze be sources (obstacles). Consider the gradient of the potential at a point P in four different mazes (A, B, C, and D) with the following relationships (see also Figure 1). Mazes A and B (similarly for C and D) have the same walls (and wall source strength) but their "doors" (just pieces of wall with open or closed states) are in different positions: the right is open in A, while the left is open in B. A and C (similarly B and D) have different walls but exactly the same door positions *relative to the effector's position P.*

Clearly, from the figure, the desired direction of movement for the effector at P is rightward for A, leftward for B, leftward for C and rightward for D. Using a coordinate frame where $(1, 2)$ means (one unit right, two units up) and, for simplicity, assuming unit-sized steps for the effector, the desired step taken in each maze would be $(1, 0), (-1, 0), (-1, 0)$ and $(1, 0)$ for A, B, C and D respectively. This means that $(s_a - s_b) \cdot (s_c - s_d) < 0$ holds where s_i is the step in maze i.

Using superposition, the gradient of the potentials for each maze is:

$$\nabla\phi_a = \nabla\phi_{wallsa} + \nabla\phi_{leftclosed} + \nabla\phi_{rightopen} \quad (4)$$
$$\nabla\phi_b = \nabla\phi_{wallsb} + \nabla\phi_{leftopen} + \nabla\phi_{rightclosed} \quad (5)$$
$$\nabla\phi_c = \nabla\phi_{wallsc} + \nabla\phi_{leftclosed} + \nabla\phi_{rightopen} \quad (6)$$
$$\nabla\phi_d = \nabla\phi_{wallsd} + \nabla\phi_{leftopen} + \nabla\phi_{rightclosed} \quad (7)$$

where $\nabla\phi_{wallsi}$ is the gradient of the potential for the walls of maze i and $\nabla\phi_{leftopen}$ is the gradient of the potential of the open left door (and so on). From the construction we know that $\nabla\phi_{wallsa} = \nabla\phi_{wallsb}$ and $\nabla\phi_{wallsc} = \nabla\phi_{wallsd}$, so that

$$\begin{aligned} \nabla\phi_a - \nabla\phi_b &= \nabla\phi_{leftclosed} - \nabla\phi_{leftopen} + \nabla\phi_{rightopen} && (8) \\ &\quad -\nabla\phi_{rightclosed} && (9) \\ \nabla\phi_c - \nabla\phi_d &= \nabla\phi_{leftclosed} - \nabla\phi_{leftopen} + \nabla\phi_{rightopen} && (10) \\ &\quad -\nabla\phi_{rightclosed} && (11) \\ &= \nabla\phi_a - \nabla\phi_b && (12) \end{aligned}$$

But this means that $(\nabla\phi_a - \nabla\phi_b) \cdot (\nabla\phi_c - \nabla\phi_d) > 0$, no matter what underlying point potentials are used. Thus the potentials constructed by superposition cannot have the relationship that the desired potentials *must* have. The only way that superposition might work in these examples is by carefully, dynamically adjusting the strengths of the walls and the goal so that the effector moves in the correct direction at each point. This would require solving for the entire maze to determine this, subverting the purpose of superposition in the first place.

4 Harmonic functions and arbitrary boundary conditions

From the arguments posed in the previous section, it appears that superposition is not practical for solutions to Laplace's equation where the obstacles are of nonzero extent and must remain impenetrable. However, numerical methods are well suited to the task of computing solutions with such arbitrary boundary conditions.

Numerical solutions for Laplace's equation are readily obtained from Finite Difference methods [4]. Let $\phi(x, y)$ be a function which satisfies Laplace's Equation, and let $u(x_i, y_j)$ represent a discrete regular sampling of ϕ on a grid. A central difference formula for the second derivatives of ϕ can be derived using a Taylor series expansion:

$$\phi_{xx}(x_i, y_j) = \frac{u(x_{i+1},y_j)-2u(x_i,y_j)+u(x_{i-1},y_j)}{h^2} \quad (13)$$
$$-\frac{h^2 u_{xxxx}(\xi_i,y_j)}{12} \quad (14)$$

$$\phi_{yy}(x_i, y_j) = \frac{u(x_i,y_{j+1})-2u(x_i,y_j)+u(x_i,y_{j-1})}{k^2} \quad (15)$$
$$-\frac{k^2 u_{yyyy}(x_i,\eta_j)}{12} \quad (16)$$

where h and k are the step sizes to be used in approximating the derivatives in each direction. The the fourth order error terms are usually negligible and can be discarded. If the step sizes are all equal, Laplace's Equation may be written in discrete form:

$$h^2\phi(x_i, y_j) = u(x_{i+1}, y_j) + u(x_{i-1}, y_j) + u(x_i, y_{j+1}) \quad (17)$$
$$+u(x_i, y_{j-1}) - 4u(x_i, y_j) \quad (18)$$

This expression can be extended to higher dimensions in the obvious way. With the introduction of boundary conditions to fix those $u(x_i, y_j)$ which fall on $\partial\Omega$ (including obstacle boundaries), the linear system described by this equation can be solved to obtain values of ϕ sampled at points on a grid. Since this system is usually quite large, techniques such as Gauss-Seidel iteration or Successive Overrelaxation may be used [4, 21]. In the examples provided here, Gauss-Seidel iteration is used. For Laplace's Equation, this method simply consists of repeatedly replacing each grid element's value with the average of its neighbors, using a 2^n-neighborhood for n dimensional regions. Those grid elements which represent boundary conditions (obstacles and goals) are held fixed.

The numerical representation used for these experiments is important. The iteration technique is terminated when there is no change of any grid node from one iteration to the next. If 32 or even 64-bit floating point representations are used, this is not sufficient to avoid flat areas in the resulting potential function. This problem is remedied as follows:

Let ϕ be the desired solution to Laplace's Equation, u be a grid of values of ϕ, c be a small integer, and let k be the number of bits of precision to be used in solving for ϕ.

1. Solve $\nabla^2\phi$ using Gauss-Seidel iteration using k bit integer arithmetic and array u.

2. Set n to be the number of flat regions in u.

3. If $n > 0$ then shift all values of u left by c bits and go to 1.

4. Terminate. The array u contains a sampling of ϕ where every non-b.c. node has a neighbor with a smaller value.

Once the array u has been computed to the necessary precision, streamlines may be computed simply by following the gradient from the start point. These streamlines must end at goal sets. Note that there may be more than one goal point. These need not even be points, but can be sets. Multilinear interpolation may be used to interpolate between grid nodes, since functions generated thereby are harmonic.

Solving for streamlines requires the generation of a solution over the entire region. Therefore, in a static situation, where the goal point and obstacles are fixed, the solution may be computed and then reused as often as desired. It need not be recomputed unless the obstacles and/or goal point change position. In this case, path generation can proceed very quickly, since it only involves evaluation of the gradient of a precomputed potential function. In a dynamic situation, the solution must be recomputed whenever the configuration of the environment changes.

5 Experiments

All experiments shown here were implemented in Lisp using Gauss-Seidel iteration on a Texas Instruments Explorer II. The somewhat jagged nature of the paths is due to the fact that no interpolation was performed here. Solid circles denote start points, while solid squares denote goal points. The paths shown here were constructed by performing a steepest descent from each start point to the goal, using only the grid points. Interpolation of the gradient would provide smoother paths. In every example presented, the actual computation of the paths required very little computation time (< 1 second). While the computation time of the entire path is proportional to the path length, interpolation of the gradient for control purposes can be performed in constant time.

Figure 2 illustrates paths that were generated from a numerical solution of Laplace's equation. One solution over the region interior to the maze was used to generate all paths. The maze represents an area of approximately 200x200 units. The solution was computed using a 55x44 grid. The solution for this maze was computed using 64-bit integer arithmetic, and required 639 iterations to converge to a zero error solution in 74 seconds. The boxes denote the two goal points which were placed in the maze, and solid circles denote starting points.

Figure 2: Sample paths in a maze

Figure 3 shows another example with three paths and one goal point. This example was computed in 303 iterations and 23 seconds on a 30 by 41 grid.

Figure 4 shows a more complicated domain, whose potential function was computed in 188 seconds with a 50x50 grid. Figure 5 shows another complicated domain with a large interior obstacle. This example used a 70x70 grid and was computed in 176 seconds. Both computations required no more than 80 bits per grid node for the potential function.

Figure 3: Sample paths in a region with obstacles

Figure 4: 50 by 50 grid, 188 seconds

Figure 5: 70 by 70 grid, 176 seconds

Figures 6 and 7 show two views of a three dimensional example with a toroidal freespace which is basically a cube with a hole through it. There are two starting points and one goal point. The goal point is denoted by a small box.

It is estimated that the UMass CAAPP [19] would be capable of solving Laplace's equation on a 512x512 grid via a central difference scheme in 10-20 milliseconds. This makes path construction via harmonic functions in a dynamic environment more practical, at least for a robot with a simple configuration space [1].

6 Summary

Harmonic functions offer a fast method of producing paths in a robot configuration space. The use of harmonic functions prevents the spontaneous creation of local minima at some cost in speed for truly general configurations. However, given that massively parallel processors are well-suited to quickly solving Laplace's equation using standard numerical methods, the authors feel that this is not a serious drawback.

Thanks go to Gerald Pocock of the University of Lowell and T. V. Subramanian of JPL for encouraging and stimulating discussions.

Figure 6: Sample paths in a three-dimensional domain

Figure 7: Another view of the 3D domain

References

[1] Ronald C. Arkin. Towards cosmopolitan robots: Intelligent navigation in extended man-made environments. Technical Report 87-80, COINS Department, University of Massachusetts, September 1987.

[2] Jérôme Barraquand, Bruno Langlois, and Jean-Claude Latombe. Numerical potential field techniques for robot path planning. Technical Report STAN-CS-89-1285, Department of Computer Science, Stanford University, October 1989.

[3] I. N. Bronshtein and K. A. Semendyayev. *Handbook of Mathematics*. Van Nostrand Reinhold Company, New York, 1985.

[4] Richard L. Burden, J. Douglas Faires, and Albert C. Reynolds. *Numerical Analysis*. Prindle, Weber and Schmidt, Boston, 1978.

[5] John Francis Canny. *The Complexity of Robot Motion Planning*. PhD thesis, Massachusetts Institute of Technology, 1987.

[6] Ruel V. Churchill, James W. Brown, and Roger F. Verhey. *Complex Variables and Applications*. McGraw-Hill Book Company, 3rd edition, 1976.

[7] Ruel V. Churchill and James Ward Brown. *Fourier Series and Boundary Value Problems*. McGraw-Hill Book Company, 3rd edition, 1978.

[8] David Halliday and Robert Resnick. *Physics*. John Wiley and Sons, New York, 1962.

[9] S. S. Iyengar, C. C. Jorgensen, S. V. N. Rao, and C. R. Weisbin. Robot navigation algorithms using learned spatial graphs. *Robotica*, 4(2):93–100, April-June 1986.

[10] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pages 500–505. IEEE, March 1985.

[11] Daniel E. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of the 1987*

IEEE International Conference on Robotics and Automation, pages 1–6. IEEE, April 1987.

[12] Bruce H. Krogh. A generalized potential field approach to obstacle avoidance control. In *Robotics Research: The Next Five Years and Beyond*. Society of Manufacturing Engineers, August 1984.

[13] Tomás Lozano-Pérez. Robotics (correspondent's report). *Artificial Intelligence*, 19(2):137–143, 1982.

[14] Damian Lyons. Tagged potential fields: An approach to specification of complex manipulator configurations. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, pages 1749–1754. IEEE, April 1986.

[15] J. Milnor. *Morse Theory*, volume 51 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, New Jersey, 1970.

[16] John K. Myers. Multiarm collision avoidance using the potential field approach. In *Space Station Automation*, pages 78–87. SPIE, 1985.

[17] W. S. Newman and N. Hogan. High speed robot control and obstacle avoidance using dynamic potential functions. Technical Report TR-86-042, Philips Laboratories, November 1986.

[18] Elon Rimon and Daniel E. Koditschek. Exact robot navigation using cost functions: The case of distinct spherical boundaries in E^n. In *Proceedings of the 1988 IEEE International Conference on Robotics and Automation*, pages 1791–1796. IEEE, April 1988.

[19] Charles C. Weems. Some sample algorithms for the image understanding architecture. In *Image Understanding Workshop*, pages 127–138. Defense Advanced Research Projects Agency, April 1988.

[20] Robert Weinstock. *Calculus of Variations with Applications to Physics and Engineering*. Dover Publications, Inc., 1974.

[21] E. C. Zachmanoglou and Dale W. Thoe. *Introduction to Partial Differential Equations with Applications*. Dover Publications, Inc., New York, 1986.