Mobile Robot Path Planning in Dynamic Environments

John P. H. Steele* and Gregory P. Starr†
Department of Mechanical Engineering
University of New Mexico
Albuquerque, New Mexico 87131, USA

Abstract

This paper presents our approach to the problem of controlling a mobile robot which must travel in dynamic environments. A dynamic environment is one which has obstacles, both stationary and moving, of which the robot is unaware at the time the original path plan is developed. There exists a natural decomposition of the path planning problem which provides the basis for dealing with such environments. The decomposition consists of (1) planning a path which avoids all the known static obstacles prior to initiating any motion, and (2) as the path motion is executed, the motion control is updated in real-time to avoid collisions with any unforseen obstacles either stationary or moving. The algorithms for both the static and the dynamic avoidance of obstacles is presented.

Introduction

A good deal of research has been done with regard to path planning for mobile robots in known static environments. That is, if the robot has a complete (in the sense of obstacles) map of its world, several approaches have been shown to be successful at providing the information necessary to drive the robot through such an environment and avoid collisions with the known obstacles[Thompson 77][Giralt et al. 79][Brooks Lozano-Peréz 83] [Thorpe 84][Crowley 84]. All such approaches assume error-free deadreckoning motion control. Thus these methods are only a first step towards a robot control which can deal with a realistic environment such as would be faced by a robot working in a real application. In a real situation, the robot would be faced with unexpected (previously unknown) static obstacles, and quite likely, dynamic obstacles, e.g., humans and other robots doing cooperating work.

The key to dealing with dynamic environments is the decomposition of the planning process into two supporting processes: (1) a premeditated plan for traveling from the present location to some desired goal location and (2) a real-time avoidance control which is capable of redirecting the motion of the robot when unexpected obstacles are encountered.

Because the architecture of the control system has such a strong influence on the behavior of a mobile robot system, no presentation on navigation would be complete without a discussion of the control system structure. The control system architecture defines the relationship between the command input and the actuation of its corresponding activities.

System Concept

We believe that the control system should contain different levels of reasoning. That is, the control system should react to new information at the most appropriate level to maximize the benefit to the system. For example, the robot could spend a great deal of time and energy analyzing the color, make, and model of the oncoming automobile, but a much more appropriate response would be to simply note the movement and get out of the way. We have identified three separate levels of response on our robot: (1) reflex, (2) subliminal, and (3) reasoned. A block diagram of our control system is shown in Figure 1.

The approach we propose is really a hybrid of two other approaches: *classical* and *layered* [Brooks 86]. In this design, the most typical flow of information is modeled after that of the "classical" approach, that is, information is received by sensors, processed, passed up to a reasoning module (supervisor, world modeler, and planner), which digests the information, plans the most appropriate response, and sends action commands out to the effector system. In addition, this "classical" flow path allows for introspection by the control system of its own processes, for example, the ability to analyze the behavior of the sensor, and if the readings are erratic, to decide to disregard the information received from that particular sensor until there is time to run some diagnostic.

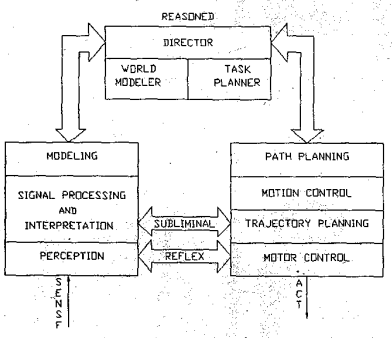

Figure 1: High Level Control Architecture

Although this comprehensive path is available for information flow, not all common activities require such detailed analysis. For example, if the robot is about to run into an obstacle, a complete model of the obstacle is not immediately necessary. What is needed immediately is a command to the locomotion system to "stop", or to divert motion to an avoidance path. With our multi-level hybrid hierarchical control structure, such actions are taken at the appropriate level.

Reflex

Reflex response is that which requires rapid response but with no real thought. A human analogy is putting your finger on a hot stove. The neurons in your fingers sense the heat and command motion of your arm muscles to remove your finger from the hot surface. No thought is given to what it is that is hot or what it looks like or even what is the best path to escape from the heat, our muscles simply respond to the sensory command and action is taken. In our robot, Remolina, this process has been implemented by installing a ring of ultrasonic range sensors around the perimeter of the robot. If an object is detected within this close proximity of the robot then an avoidance action is taken (at present Remolina simply stops, however we hope to implement redirection of motion in a direction "normal" to the direction of the sensed object). Such action is represented by the lower level of direct communication between the sensor subsystem and the effector subsystem in Figure 1. No reasoning, modeling, or analysis is involved, the robot simply reacts to the sensation.

*New address: Advantage Production Technology, 4208 Balloon Park Rd., Albuquerque, NM 87109, USA

†Work supported under DOE contract DE-AC04-76DP00789

CH2556-9/88/0000-0922 $1.00 © 1988 IEEE

Subliminal

Subliminal responses are those which actually require analysis on the part of the control system but are conducted at a level below that of full modeling and reasoning. For these subliminal responses, the behavioral goal is already determined, in fact, the goal is molded by the way the software is written. The level of modeling and analysis is only sufficient to provide the system with information necessary to carry out the desired action and not to provide the control system with any additional knowledge or invoke any high level reasoning. The movement of our limbs to satisfy some higher level goal is an example of this type of action. We reach for the coffee cup but we don't consciously think about the path taken to get there. We alter our path to avoid a chair in the middle of the room but we are thinking about something else. As robots become more sophisticated, more and more of what we will come to consider as their "innate" behavior will come from this subliminal response which we program into their control system. For Remolina, the process of avoiding obstacles while the robot is in motion is an example of such a response. That is, as the robot travels along its intended path, if an obstacle is encountered which prevents the robot form traversing this path, the response of diverting the path traveled to circumvent a collision with the obstacle and the return to the prescribed path constitutes a subliminal response. This is because no detailed modeling of the obstacle is needed. The robot simply needs to compute an alternate path which will allow it to continue motion in the general direction of the goal without hitting the object. This is implemented using an adaptation of the force repulsion model [Khatib 86]. The avoidance of moving obstacles also falls into this category.

Reasoned Response

Reasoned response constitutes all of those control activities which include conscious deliberation — *should I go to class or the ice cream parlor?* These responses require the modeling of the input data to the maximum extent possible by the system, or at least to the level necessary to make an adequate decision. The implementation of expert systems to do rational decision making would be at this level. The analysis of user commands will be at this level. The use of introspection[Georgeff Lansky 87] will be at this level. Such processes are necessarily slower than the two lower levels of response previously mentioned, but this lower bandwidth is acceptable since most life threatening interactions will be dealt with at those lower levels which have the bandwidth to respond in a timely fashion.

Decomposition of the Path Planning Problem

We live in dynamic environments. Almost no configuration of space remains unchanged for very long; furniture gets moved, new objects are added, and old ones disappear. The frequency of such perturbations may be slow but none the less certain. Thus the robots which we will build must be capable of dealing with such dynamic situations and, in turn, the path planning strategy we employ must also be capable of dynamic response. We have chosen to evolve the path performing character of the robot using a human paradigm. That is, given the task of traveling from point A to point B, many of us would *decompose* the problem along the following lines. First we would generate a list of the subgoal locations through which we want to travel on our way to the goal, B. These subgoals are selected from a geographic knowledge base which we have stored. The geographic information would be the most current we have available and may include information about temporary detours or difficulty ratings for certain segments of the path. The particular points are chosen based upon a particular set of criteria under which we are operating at the time, e.g., I'll go to the water fountain by way of Sue's office so I can see if she is there. Any particular set of weighting factors can be used, but the hope is that the particular set we chose somehow makes the path chosen more effective, whatever our purpose. Once the subgoals have been chosen, we set off on our way to the goal. At this point, we may well become aware of changes in the environment which will make the execution of our original path impossible, e.g., we may find a locked door, or we may find a workman and his tools in one hallway, and almost certainly we will find that some furniture has been moved since our last recording of positions or that someone else is on a trajectory which would result in collision if we did not alter our original course. Thus to get to the final location, B, the original path that was planned will have to be altered in *real-time*. This paradigm can be used to implement a path planning / motion executing algorithm on a mobile robot. The key to the effectiveness of the algorithm comes from the decomposition of the problem into (1) the preplanned static subgoal portion and (2) the dynamic avoidance portion. A similar decomposition has been proposed by [Krogh Thorpe 86] for unknown static obstacles.

The static path planning procedure which we are using has been presented elsewhere[Steele Starr 87]. Suffice to say that we use a graph search method which results in lists of subgoal coordinates linking the start and goal position with straight line paths between the subgoals. The paths are designed to allow adequate clearance from nearby known obstacles. However, the path generated offers only a guide for traveling towards the goal since obstacles may obstruct the proposed path and the robot may very well drift as it travels. This leads to our next topic.

Obstacle Avoidance

Khatib modeled objects in a robot manipulator's workspace using a field potential approach[Khatib 86]. In our application, the field potential reduces to a set of force vectors, an attractive force vector for the goal, and a repulsive force vector for each obstacle seen by the sensors.

The attractive force vector has a constant magnitude which is set equal to one. The direction of the attractive force vector is established by the position of the goal relative to the robot expressed in world coordinates. Referring to Figure 2 we can write,

$$\vec{R}_{g_r} = \vec{R}_{g_w} - \vec{R}_{r_w}. \tag{1}$$

The repulsive force has a magnitude which is a function of an inverse cube relationship[1] and has a gain factor, K_d, which is used to tune the repulsive responsive, i.e.,

$$\vec{F}_{repul} = \frac{K_d}{d^3}. \tag{2}$$

If we want the equilibrium point, (or balancing point), for the repulsive force to be 1.2 meters, K_d is set to 1.728. That is, at a distance of 1.2 meters, the repulsive force will result in a vector of magnitude equal to 1.0. The obstacle shown in Figure 2 is a distance of 1.0 meters away and thus the magnitude of the repulsive force is 1.728. The angular relationship of the sensed obstacle is relative to the *robot* and thus must be transformed into world coordinates before we sum the repulsive force, F_{repul}, with the attractive force, F_{at}. We can express the transformation as:

$$F_{repul_w} = T_w^r F_{repul_r}, \tag{3}$$

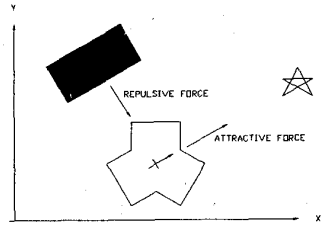

Figure 2: Computing attractive and repulsive forces

[1] A cube relationship gives a crisper path response than does a square relationship.

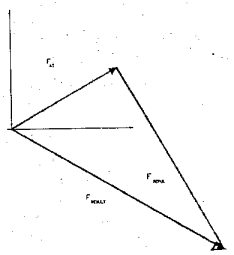

Figure 3: Vector summation of forces

where the transformation matrix is given as:

$$T_w^r = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4)$$

For the example given in Figure 2 the repulsive force relative to world coordinates is given by:

$$\vec{F}_{repul_w} = 0.864\vec{u}_x - 1.496\vec{u}_y. \quad (5)$$

Once the mapping to world coordinates has been done, the vector summation is done as shown in Figure 3. The velocity of the robot is determined by the resultant force vector, F_{result}. The direction of motion is taken from the direction of F_{result} and the magnitude of velocity is determined from the magnitude of F_{result}, except that when the magnitude of the resultant exceeds 1.0, the speed is set to the maximum allowed for the robot, i.e.,

$$|\dot{V}_r| = \begin{cases} K_v|F_{result}| * V_{max} & \text{if } F_{result} < 1.0, \\ 1.0 * V_{max} & \text{otherwise.} \end{cases} \quad (6)$$

K_v is a gain factor which allows us to tune the rate at which the speed decreases as we approach the obstacle, higher K_v's (greater than 1.0) mean the robot maintains its full speed even as it gets within the equilibrium range of the obstacle. A graph of this is shown Figure 4.

Some readers may have already noticed one problem with this algorithm. When the obstacle is directly in line with the robot's path

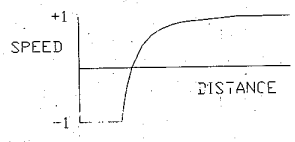

Figure 4: Plot of speed versus separation distance ($K_v = 1$)

to the goal, no lateral repulsion force is generated and the robot will simply roll up to a stop at the equilibrium distance from the obstacle. In reality the chances that the obstacle will lie exactly along the line of action to the goal are very small (except when the sensor has a broad beam and the computer assigns a direction as can be the case with ultrasonic range sensors). However, even an angle close to the direct line of action will result in the robot slowing to an almost complete stop and then very very slowly inching its way out to the side. This is because the speed is attenuated as an inverse function of separation. This creeping motion is very undesirable from both an efficiency and an aesthetic point of view.

This "stalled" motion is a characteristic of the algorithm. We have implemented a fix to the algorithm which gets around this problem, although as we will point out later, it is not a panacea. The fix involves switching the x and y components of the repulsive force vector *if* the angle of incidence of the obstacle is less than 45 degrees. The repulsion is computed in the regular fashion, then the angle is checked against the path. If the angle of incidence is less than 45 degrees then the x component is shifted to y and the y component is shifted to x, with the appropriate sign changes. This results in a much more aggressive move to the side when encountering an obstacle directly in the path. A graphic representation of the resulting force vector computation is shown in Figure 5. This improves the avoidance behavior significantly because the robot speed is more evenly maintained and the move to the outside is more abrupt. The limitation to this fix is that the obstacle is assumed to always be on one side or the other of the 0 degree orientation of the robot. With real sensor data this is not always the case! This is because the readings from the sensors are not always perfect. If the sighting is close to the 0 degree direction, then a slight shift in the centroid of the sighted obstacle or the sighting of a phantom obstacle can shift the repulsive force to the other side of zero. This in turn causes a dramatic shift in the resultant force once the x and y components are interchanged. If the position of the obstacle continues to shift back and forth across the zero heading of the robot, the robot will exhibit harmonic oscillation in front of the obstacle.

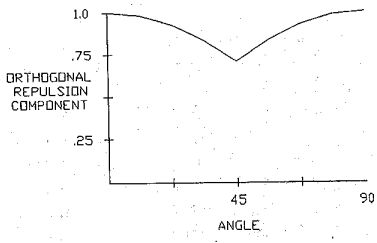

Figure 5: Switching algorithm

Avoiding Moving Obstacles

Avoidance of other moving bodies by a robot in motion begins by estimating the trajectory of the moving obstacle. This estimate is based upon sensor inputs available to the robot control system. An assumption is made that the obstacle will travel in a straight line. This assumption can be justified on the basis that the update rate on the sensor information is rapid enough that the actual path of the obstacle can be approximated as a series of line segments, i.e., a polyline. In any cycle, the robot reacts to potential collisions based upon this assumption of straight line motion. The trajectory of the robot is modeled as straight line motion as indeed it is for Remolina between subgoals of the planned path. These trajectories are represented as lines in three space; x, y, time. These lines are explicitly represented using the positions of two points on each line.

Having gotten an explicit representation for both the robot and the obstacle (note: multiple interactions are simply modeled as a summation in the vectorial sense, of the individual interactions) the next step is to determine how closely these paths interact and at what point in time and space the "*point*" of closest approach occurs. I use *point* in quotes since the interaction is only a point if the two trajectories actually intersect. More typically the interaction is a line. In fact, most typically the trajectories of the robot and the obstacle are represented by two lines skewed in three space. In this case there is a single line which bridges the two lines and represents the closest approach of the two objects. This line is the common normal, CN, a line which is perpendicular to each of the trajectory lines and in addition intersects both lines. To obtain the CN, take the cross product of the two path trajectories in their vector representation. This results in a *vector* representation for the CN.

Once the CN has been found in a vector form, the next step is to compute the actual points of intersection of this CN with the two trajectories. Let $(x, y, t)_{poi_o}$ be the point of intersection of the obstacle trajectory with the common normal and let $(x, y, t)_{poi_r}$ represent the point of intersection of the robot trajectory with the common normal, then the CN line definition can be written as:

$$\begin{aligned} x - x_{poi_o} &= \gamma A_{CN} \\ y - y_{poi_o} &= \gamma B_{CN} \\ t - t_{poi_o} &= \gamma C_{CN}. \end{aligned} \quad (7)$$

Also CN can be expressed as:

$$\begin{aligned} x - x_{poi_r} &= \delta A_{CN} \\ y - y_{poi_r} &= \delta B_{CN} \\ t - t_{poi_r} &= \delta C_{CN}. \end{aligned} \quad (8)$$

Since $(x,y,t)_{poi_r}$ is also on the robot trajectory we can also write

$$\begin{aligned} x_{poi_r} - x_{r_1} &= \epsilon A_r \\ y_{poi_r} - y_{r_1} &= \epsilon B_r \\ t_{poi_r} - t_{r_1} &= \epsilon C_r. \end{aligned} \quad (9)$$

Similarly for the obstacle trajectory,

$$\begin{aligned} x_{poi_o} - x_{o_1} &= \zeta A_o \\ y_{poi_o} - y_{o_1} &= \zeta B_o \\ t_{poi_o} - t_{o_1} &= \zeta C_o. \end{aligned} \quad (10)$$

Now since we have two points on the CN line we can also write

$$\begin{aligned} x_{poi_r} - x_{poi_o} &= \xi A_{CN} \\ y_{poi_r} - y_{poi_o} &= \xi B_{CN} \\ t_{poi_r} - t_{poi_o} &= \xi C_{CN}. \end{aligned} \quad (11)$$

Substituting into Equation 11 from Equation 9 and Equation 10 results in a system of three linear equations as shown in Equation 12,

$$\begin{aligned} \epsilon A_r - \zeta A_o - \xi A_{CN} &= x_{o_1} - x_{r_1} \\ \epsilon B_r - \zeta B_o - \xi B_{CN} &= y_{o_1} - y_{r_1} \\ \epsilon C_r - \zeta C_o - \xi C_{CN} &= t_{o_1} - t_{r_1}. \end{aligned} \quad (12)$$

The A_j, B_j, C_j and the $(x,y,t)_j$ are all known quantities. Thus we have three equations and three unknowns: (ϵ, ζ, ξ).

These equations can be solved directly using Gaussian elimination. The three quantities for which we are solving are a series of weighting factors which represent the location of the POI's on their respective object lines relative to the specified initial points. Then using Equations 9 and 10 the actual locations of the POI's can be determined.

Having determined the POI's, the separation between of the two lines can be computed using the standard metric distance, i.e.,

$$dist = \sqrt{(x_{poi_r} - x_{poi_o})^2 + (y_{poi_r} - y_{poi_o})^2 + (t_{poi_r} - t_{poi_o})^2} \quad (13)$$

With this information in hand, we can now determine just how closely the robot and obstacle will interact if we make an assumption as to the extent of the obstacle. With the incorporation of a MOS, Moving Obstacle Sensor, this extent could be measured and introduced at the time of sighting of the moving obstacle. However, because no MOS is presently in the system, the radius of the obstacle, r_o, is set empirically, based upon our estimate of the x-y extent of most moving obstacles which are expected to be encountered by the robot.

The cross section of the actual swept volume of a circular (foot print) robot is an ellipse in three space,$(x,y,time)$. However, for ease of computation and as an additional safety factor in the avoidance distance selected, we idealize the cross section as a circle. Thus the swept volume would be a cylinder.

We simply transform the CN into a "repulsive force" which is then used to alter the course of the robot in a manner analogous to that used for the static obstacle avoidance. The force repulsion generated by the separation distance of the robot and the obstacle at their point of closest approach, *pca*, is determined using an inverse cube law relationship. That is, the magnitude of the repulsion force is computed as shown in Equation 14,

$$|F_{repul}| = K_f \frac{1}{d^3}. \quad (14)$$

This magnitude is associated with the direction established by the vector formed by moving from the POI for the obstacle to the POI for the robot. The gain, K_f, is set empirically to provide the desired avoidance characteristics with regard to distance of separation.

Summary

We believe the control architecture of a mobile robot must contain multiple levels of response. The design we have implemented has three such levels of response: reflex, subliminal, and reasoned. This allows information to be processed more efficiently to achieve prescribed system goals. We have presented a natural decomposition of the mobile robot navigation problem based upon separation of activities into premeditated goal setting and real-time obstacle avoidance. We use a force repulsion paradigm as the basis for the obstacle avoidance algorithm. The repulsion is based upon sensor readings. We have presented a paradigm for the avoidance of moving obstacles. This model, which maps the motion of the robot and obstacle into a three dimensional space $(x,y,time)$, facilitates the analysis necessary to avoid collisions either by spatial or time avoidance.

References

[Brooks Lozano-Peréz 83] R. A. Brooks and Tomás Lozano-Peréz, "A Subdivision Algorithm in Configuration Space For Find Path With Rotation", *IJCAI-83*, pp. 799-806.

[Brooks 86] R. A. Brooks, "A Robust Layered Control System For A Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol RA-2, No. 1, March 1986, pp 14-23.

[Crowley 84] J. L. Crowley, "Navigation for an Intelligent Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 1, Mar 1985, pp. 31-41.

[Giralt et al. 79] G. Giralt, R. Sobek, and R. Chatila, "A Multilevel Planning and Navigation System For A Mobile Robot", *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, Tokyo, Japan, August, 1979, pp. 335-337.

[Georgeff Lansky 87] Michael P. Georgeff and Amy L. Lansky, "Reactive Reasoning and Planning", *AAAI-87*, Seattle Washington, August 1987, pp. 677-682.

[Kant Zucker 86] Kamal Kant and Steven W. Zucker, "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition", *International Journal of Robotics Research*, Vol. 5, No. 3, Fall 1986, pp. 72-89.

[Khatib 86] Oussama Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *International Journal of Robotics Research*, Vol. 5, No. 1, Spring 1986, pp. 90-98.

[Krogh Thorpe 86] Bruce H. Krogh and Charles E. Thorpe, "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles", *Proceedings of IEEE Robotic and Automation Conference*, 1986, pp. 1664-1669.

[Steele Starr 87] John P. H. Steele and Gregory P. Starr, "Path Planning and Heuristics for Mobile Robots in Real World Environments", *Proceedings of IEEE, Systems, Man, & Cybernetics*, 1987.

[Thompson 77] Alan Thompson, "The Navigation System of the JPL Robot", *JPL Publication 77-20*, July 15, 1977.

[Thorpe 84] C. E. Thorpe, "Path Relaxation: Path Planning for a Mobile Robot", *AAAI 84*.