Robot Navigation in a Known Environment with Unknown Moving Obstacles

Steven Ratering and **Maria Gini**
Department of Computer Science
University of Minnesota

Introduction

To be useful in the real world, robots need to move safely in unstructured environments and achieve their given goals despite unexpected changes in their surroundings. The environments of real robots are rarely predictable or perfectly known so it does not make sense to make precise plans before moving.

The robot navigation problem can be decomposed into the two problems of getting to a goal and avoiding obstacles. The problem of getting to a goal is a global problem in that short paths to the goal generally cannot be found using only local information. The topology of the workspace is important in finding good routes to a goal. The problem of avoiding obstacles can often be solved using only local information, but for an unpredictable environment it cannot be solved in advance because the robot needs to sense the obstacles before it can be expected to avoid them.

Some have solved the navigation problem by solving these two sub-problems one after the other. A path is first found from the robot's initial position to the goal and then the robot approximates this path as it avoids obstacles. This method is rather restrictive in that the robot is required to stay fairly close to or perhaps on a given path. This would not work well if the path found goes through a passageway which turns out to be blocked by some unforeseen obstacle. Solutions that are only local, such as those produced often by artificial potential fields, often lead the robot into local minima traps.

We propose a much more flexible solution using a common tool, the artificial potential field, in a new form that we call a "hybrid artificial potential field". A hybrid potential field is obtained by combining two different kinds of artificial potential fields, a global discontinuous potential field and a local continuous potential field.

The global potential field covers the whole floorplan and captures the static floorplan information. Since it includes only information about static objects it can be computed *a priori* when given the goal. We represent this field as a two dimensional array of heights so that the robot rolls downhill to the goal while avoiding fixed obstacles. In this potential field all free cells will be assigned the city block distance of the shortest free path from that cell to the goal so this field has no minima other than at the goal.

The local field captures local information obtained by sensors about obstacles and covers only the area around the robot. The purpose of this field is to push the robot away from obstacles that are on its path. This field has to be computed repeatedly as the robot moves in the workspace. We continuously sense the environment with sonar sensors and regularly update the dynamic potential field. For each sensor reading we place a "hill" in the potential field. Since these hills are built as the obstacles are sensed this method works with obstacles that move in unknown, unpredictable paths. Obviously, if obstacles move too fast collisions are sometimes unavoidable.

Unfortunately, when these two types of potential fields are added together, the result may have local minima. We have analyzed the causes of these minima and found ways to deal with them.

Artificial Potential Fields

Artificial potential fields can be used to solve the robot navigation problem by determining the positions the robot should move through or by controlling the forces that move the robot. Khatib [Khatib, 1985] [Khatib, 1986] pioneered the use of potential fields in a force control context. His robot was directed to move as if the goal were generating an attractive force and the obstacles were generating repulsive forces. The attractive force of the goal should extend throughout the environment of the robot, but the repulsive forces should have little or no effect at great distances. Each of these forces could be represented by a surface in three dimensional space in which the negative of the gradient points in the direction of the force. Then the attractive force is represented by a valley and the repulsive forces are represented by hills. The robot then rolls downhill until it reaches a minimum point and then it stops. The primary problem with this version of an artificial potential field is that there may be minima other than the goal so the robot may roll to some spot other than the goal

and stop.

Figure 1 illustrates a simple case where the robot will go to a local minimum other than the goal. The robot is

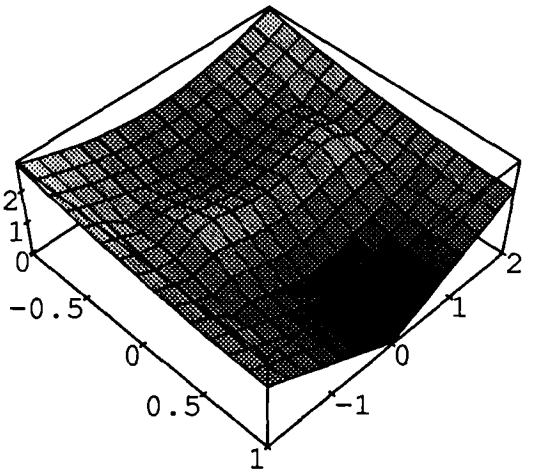

Figure 1: A wall generates a local minimum in a continuous potential field

in the middle of the upper left edge $(-1, 0)$, and the goal is in the middle of the lower right edge $(1, 0)$. There is an obstacle in the shape of a line segment from $(0, -1)$ to $(0, 1)$. The valley for the goal has height x at points whose distance from the goal is x. The hill for the obstacle has height 1 at points in the obstacle, height 0 at points whose distance from the obstacle is greater than $1/2$, and height $1 - 2x$ at points whose distance from the obstacle is x where x is between 0 and $1/2$. The valley for the goal will keep the robot on the line $y = 0$, and the hill for the obstacle will keep the robot from going through the obstacle. The minimum that the robot will reach has height $3/2$ at $(-1/2, 0)$. Walls between the robot and the goal will typically generate local minima. There have been many attempts to get around the problem of local minima.

Koditschek [Koditschek, 1987] showed that no analytic potential field function will direct a point robot in two dimensions with stationary obstacles of arbitrary shape to goal points while avoiding all obstacles. To guarantee that obstacles are avoided, there will necessarily be local minima other than the goal in some environments.

Barraquand, Latombe, and Langlois [Barraquand and Latombe, 1989] [Barraquand *et al.*, 1991] pioneered the use of a very different type of potential field. In this version, the potential field is placed in a discretized workspace and a search is made in configuration space from the starting configuration to the goal configuration. This search is guided by the potential field used as a heuristic function. Thus, the potential field is used only in position control and not force control. The floorplan is represented as a grid of free and occupied cells and the goal cell is given the value zero. The neighbors to the goal (that are free) are given the value one, and their neighbors (that are free and unassigned) are given the value two, and so on. Eventually, all free cells will be assigned the city block distance of the shortest free path from that cell to the goal. We call this a discontinuous version because the potential field is undefined for the occupied cells and there may be "cliffs", sharp discontinuities, generated by walls in the floorplan as illustrated in figure 2. This version of potential field has no minima other than at the goal.

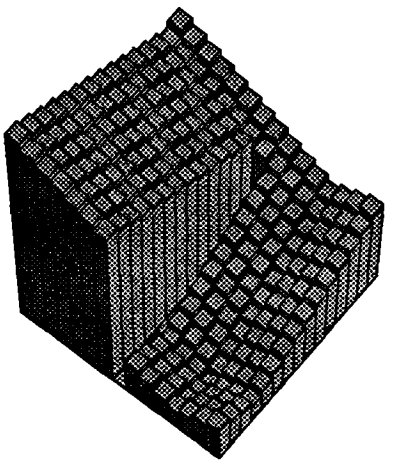

Figure 2: A wall generates a cliff in a discontinuous potential field

Navigation Using Hybrid Potential Fields

The problem with the continuous version of artificial potential fields is that the robot may be pulled to a spot other than the goal where the attractive force of the goal and the repulsive forces of the obstacles cancel each other out and the robot gets stuck in a local minimum. A discontinuous artificial potential field does not have this problem, but this version does not handle moving obstacles and it can take a relatively long time to compute.

The method of navigation we are proposing uses a hybrid between the continuous and the discontinuous versions of artificial potential fields. We derive a discontinuous potential field as a function of the floorplan and a continuous potential field as a function of the sensor readings. The floorplan does not change much, but the sensor readings change frequently, so we refer to the two types of potential fields as static and dynamic.

We first build the static potential field based on the floorplan. The algorithm to compute the static potential field requires the floorplan to be represented by a grid of occupied and free cells. All cells that are fully

or partially occupied are labeled occupied. Then we expand the obstacles by the radius of the robot so that if the center of the robot is in a free cell the rest of it will also be in free cells. The static potential field has only one minimum at the goal and is undefined for cells containing known obstacles. This potential field will pull the robot toward the goal from any free location in the workspace.

For each set of sensor readings, we build a local continuous potential field containing hills for the sensed obstacles. This dynamic potential field centered on the robot will push the robot away from the obstacles, both stationary and moving. These hills can be built as the obstacles are sensed so this works with obstacles that move in unknown paths. We represent this potential field as a discrete grid where the cells are the same size as the cells of the static potential field so that the two fields can easily be added together.

Unfortunately, when the static and dynamic potential fields are added together, the result may have local minima. On the other hand, we have studied what causes these minima and we found ways to deal with them. These local minima are caused by either sensed obstacles not in the floorplan, or by hills that are wider than they need to be, or by false sensor readings. If a local minimum is caused by a false sensor reading or by an obstacle that will get out of the way soon, then the robot should wait until its current location is no longer a minimum. If the minimum is caused by a hill that is too wide, the hill's extent should be reduced. If the minimum is caused by an obstacle that is in the way and will stay in the way, the robot should find another way to the goal.

This way of dealing with local minima is programmed first by detecting when the robot is stuck and then after it is stuck for some time, we reduce the extent of the hills caused by stable obstacles and if the robot remains stuck even longer, we put stable obstacles in the floorplan and recompute the discontinuous static potential field. We consider the robot stuck if it has not moved more than some threshold distance in some period of time. By modifying the parameter that specifies how long the robot will wait in a local minimum, one can modify how "patient" the robot is.

When the static potential field is recomputed, we add obstacles to the floorplan if the obstacles appear stable, and we remove obstacles from the floorplan if they were previously added but they have since moved out of their previous location. It may be the case that the obstacles added to the floorplan block every path from the robot to the goal and no new paths are opened up by removing other obstacles. In that case we go back to the original floorplan and original static potential field and keep trying. This way if all paths to the goal are blocked for some time, and then one or more paths is opened up, the robot still has a good chance of reaching the goal. There may be applications where this "persistent" attitude is not desired and for these situations, the algorithm can easily be modified so that the robot gives up if it has not reached the goal after some period of time.

Results from the Simulated Robot

We designed many scenarios with one, two, three, or four obstacles moving in straight lines with constant velocity. For different situations we found the speed necessary for the obstacles to collide with the robot. When there was only one obstacle, it would have to move at about the same speed as the robot for there to be a collision. When there were two obstacles, they could squeeze the robot between themselves and collide with somewhat slower speeds. When there were three or four obstacles, they could surround the robot at very slow speeds and the potential field hills would prevent the robot from escaping and then the obstacles would close in and crush the robot.

To show that this method of navigation works well in environments with many moving obstacles (not working in concert to crush the robot), we generated many scenarios with between 10 and 50 randomly moving obstacles. Each line of tables 1 and 2 was derived from running 100 random scenarios. The tables give the obstacle speed in millimeters per second (the robot's maximum speed is 500 millimeters per second), the number of obstacles per scenario, the average times to reach the goal, the number of the 100 scenarios where the robot reached the goal without any collisions, and the average number of collisions per scenario. Table 1 was obtained using a floorplan of a single large room bounded by four walls and table 2 was obtained using the five room floorplan with two blocked doorways as shown in figure 3. Each obstacle started in a random cell of the floorplan and moved in a random path consisting

obstacle speed	number of obstacles	average path time	safe runs	average crashes
100	10	72.15	98	0.02
100	20	96.42	94	0.06
100	30	125.55	92	0.09
100	40	148.56	90	0.11
100	50	174.70	84	0.18
300	10	73.66	97	0.03
300	20	94.54	92	0.08
300	30	114.43	86	0.17
300	40	133.58	80	0.24
300	50	147.48	69	0.43
500	10	76.45	95	0.05
500	20	96.69	76	0.32
500	30	109.58	63	0.57
500	40	132.23	37	1.18
500	50	143.55	25	1.61

Table 1: One room statistics

obstacle speed	number of obstacles	average path time	safe runs	average crashes
100	10	914.08	97	0.03
100	20	760.62	90	0.10
100	30	830.59	78	0.30
100	40	829.28	69	0.43
100	50	994.96	61	0.67
300	10	577.99	72	0.35
300	20	536.30	64	0.54
300	30	563.65	36	1.24
300	40	605.38	16	2.00
300	50	648.86	12	3.21
500	10	494.87	31	1.27
500	20	510.80	9	3.04
500	30	548.79	2	5.02
500	40	596.57	0	7.93
500	50	621.53	0	11.25

Table 2: Five room statistics

of short straight line segments. The different obstacles moved in many different directions and changed directions often. A few of of the crashes occurred before the robot had a chance to move because some of the initial random configurations placed an obstacle on top of the robot. When there were 10 obstacles, 2 of the 100 random cases started in a collision situation, and for 20, 30, 40, and 50 obstacles, there were 6, 7, 8, and 9 initial collisions. Most of the collisions in the one room case where the obstacles moved 100 millimeters per second were caused by these initial collisions. Increasing the number of obstacles generally increased the time needed to reach the goal. Increasing the number of obstacles or increasing the speed of the obstacles decreased the number of crash-free runs and increased the average number of crashes per run. For the one room case, only when there were 40 or 50 obstacles moving at 500 millimeters per second was the likelihood of a collision greater than 50%. There were many more collisions in the five room example with blocked doorways. Most of these collisions were by doorways or walls or in the small rooms. The average time to reach the goal is much larger in the five room scenarios and this average is skewed by a few very long paths in some cases. The performance of the robot degrades gracefully and becomes poor only in very difficult situations. Figure 3 shows the path to the goal the robot took in one scenario with 40 obstacles which are not shown.

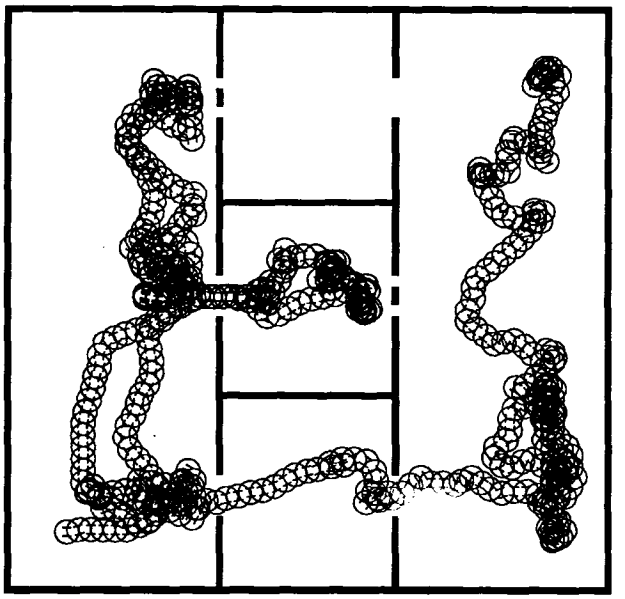

Figure 3: The path taken amidst 40 obstacles

Results from the Real Robot

Our TRC Labmate, Eric the Red, is confined to a lab whose rough floorplan is given in figure 4. Also shown in figure 4 is a path Eric took starting from near the upper right corner of the lab. Eric soon got stuck in a local minimum behind two boxes and then he added

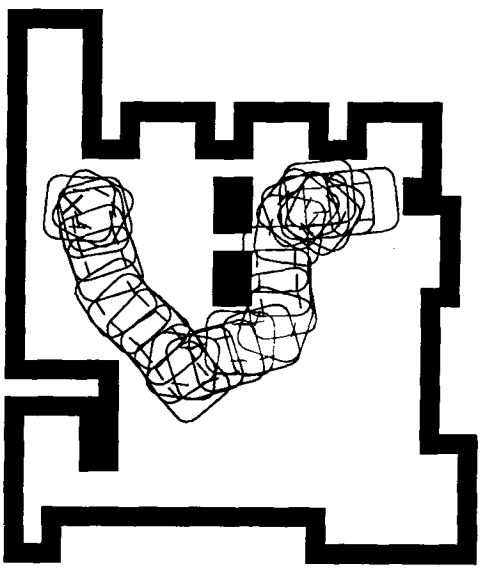

Figure 4: A path Eric took around some boxes

the obstacles to his floorplan and recomputed the static potential field and then moved around the boxes. Ten times Eric took similar paths (same starting and ending positions) and on the average he ended up about half a foot from where he "thought" he was. Figure 5 shows

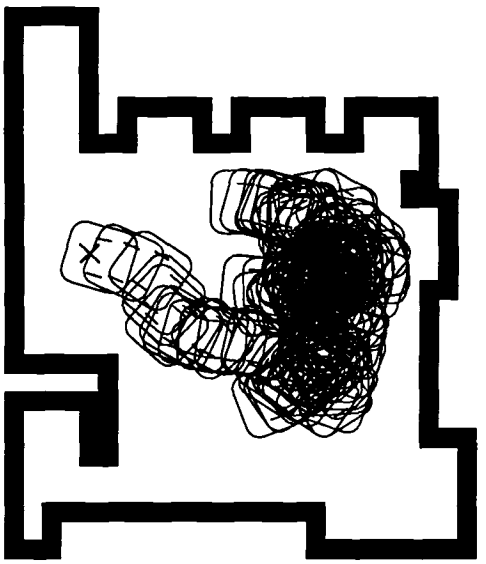

Figure 5: Eric's path when obstacles persistently got in the way

the path Eric took when moving boxes and a moving person continued to block Eric's path for three minutes before getting out of the way. After ten similar paths, it again was off by about half a foot. For the trials mentioned above, Eric regularly adjusted its position and orientation by comparing the sensor readings to the floorplan. This adjustment was usually unnecessary for the first case, but for the second case with the "determined" obstacles, when there was no adjustment, Eric was off by more than three feet on the average.

Our adjustment algorithm not only keeps Eric on track, but it can also get Eric back on track if the initial configuration it is given is wrong. We ran 15 trials with the initial orientation off by 5°, 10°, 15°, 20°, and 25° each three times. Eric stopped about a half a foot from where it thought it stopped on the average when the goal was about 15 feet from the starting position. If the dead reckoning was perfect and there was no correction based on sensor readings, Eric would have been off by 1.3 and 6.5 feet for the initial errors of 5° and 25° respectively.

Contributions

We have addressed the problem of robot navigation in a known environment with unpredictable moving obstacles. We have shown that using our hybrid potential field works well by testing both a simulated robot and a real one. We have not made any assumptions on the number or the motion of the obstacles and this makes the problem very difficult. We have performed a large number of experiments in simulation with up to 50 obstacles moving at various speeds on random trajectories. We also have experimental results with a moving robot, a TRC Labmate, equipped with ultrasonic sensors, to show that our solution works in the real world in real time. Our method will lead to some collisions, but it performs very well even when there are many obstacles moving in unpredictable paths. Its performance degrades gracefully as the speed and number of obstacles increase.

Our main contribution to robot navigation is the idea of combining a potential field which captures the floorplan information with a potential field which captures the sensory data. We use a single tool, a potential field, to solve the two problems of getting to a goal and avoiding obstacles. Since this method computes a static potential field for the entire workspace, the robot is not restricted to any fixed path as in several other methods. This method is fast enough to work in real time because for most iterations it runs in constant time in the number of obstacles and in the size of the workspace.

The main problem with using potential fields for navigation is that they often have local minima other than at the goal. Another contribution is that we have analyzed the causes of local minima in the hybrid potential field and have found ways to deal with each kind of local minimum.

References

[Barraquand and Latombe, 1989] J. Barraquand and J.-C. Latombe. Numerical potential field techniques for robot path planning. Technical Report STAN-CS-89-1285, Stanford University, 1989.

[Barraquand *et al.*, 1991] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. In *Proceedings Fifth International Conference on Advanced Robotics – Robots in Unstructured Environments*, pages 1012–1017, Pisa, Italy, June 1991.

[Khatib, 1985] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation*, pages 500–505, St. Louis, MO, 1985. IEEE.

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

[Koditschek, 1987] D. E. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *IEEE International Conference on Robotics and Automation*, pages 1–6. IEEE, 1987.