Comparison of Two Path Planning Methods for Mobile Robots

Jesús Savage, Wolf Schaarschmidt
Laboratorio de Biorobotica
Facultad de Ingeniería, UNAM
savage@servidor.unam.mx, wolfms@ieee.org

Abstract— A mobile robot that navigates in closed environments like houses or offices it needs to find paths that it can use to cross the rooms. A planner usually is a system that given an origin, a goal and a model of the environment it finds the best or a path that the robot needs to follow to reach a destination. We used two approaches for path planning, one uses artificial potential fields and the other a greedy planner augmented by oracles. In the potential field approach the robot is considered as a particle whose movements depends on two forces, one is the attraction force that is due to the goal and the second one is the repulsion force that is due to the addition of all the repulsion forces that the known obstacles generate. One of the main problems using this technique for planning the robot's movements is that the planner can get stuck in a local minimum where the attraction and repulsion forces cancel each other, thus the movement of the Robot is zero or it oscillates around a certain path.

We propose a method to eliminate this by putting additional attraction forces in the space. An expert system uses the knowledge of known obstacles to put intelligently the additional attraction forces in places that will take the robot out of the place where it is stuck, specifically in some of the corners of the obstacles.

In the greedy approach we use a greedy planner augmented by oracles. At any given point, the basic planner chooses the legal step that most reduces the distance to the goal. When its preconditions are satisfied, any of a number of oracles may intervene and supersede the choice of the planner. An oracle recognizes a particular configuration of planning state and obstacles for which it is specialized and makes a better choice than the greedy planner. The preconditions of the oracles either have to be mutually exclusive, or a hierarchy among oracles must be established.

We make a comparison of these two methods addressing the weakness and strengths of each one.

Keywords: Mobile robots, path planning, greedy algorithms, potential fields.

I. *Introduction*

There are several architectures to control mobile robots [1], [2], in our approach, the ViRbot system [3], the mobile robot's operation it is formed by several layers, see figure 1, each one having a specific function that in whole control the behavior of the robot.

In the following sections we present the planning module in which we describe two methods for path planning.

II. *Planner*

The basic problem of planning the robot's movements is reduced to: Given the initial position and heading of the robot A in space W, a path τ specifying a continuous sequence of positions and headings of the robot A must be found in order to avoid collisions with the obstacles $B_i's$, beginning in the initial position and heading and finishing in the goal position and heading. If there is not such a path, the impossibility to solve the problem is reported.

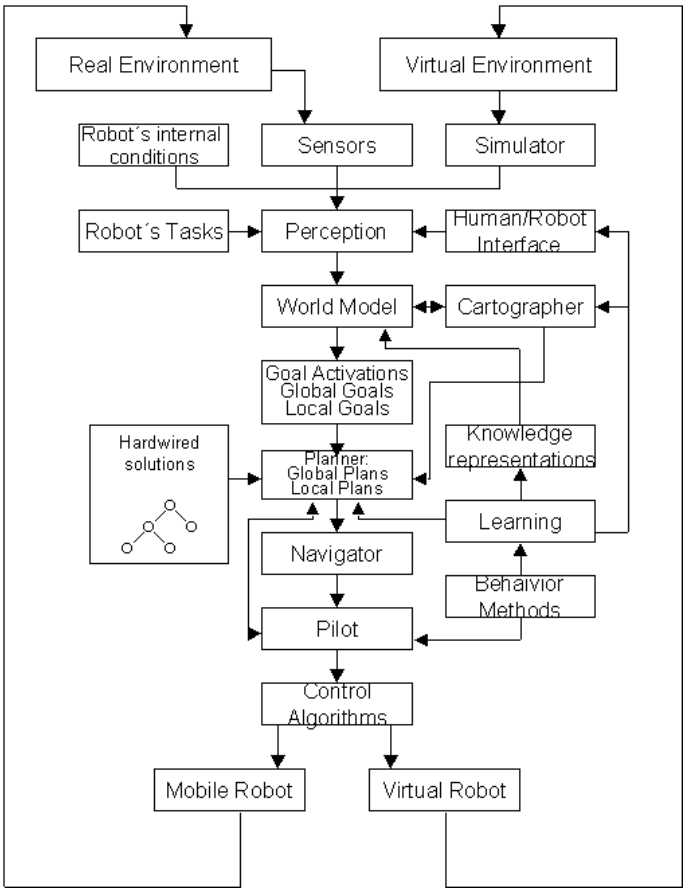

Fig. 1. The ViRbot System

Planning is defined as a procedure or guide for accomplishing an objective or task. This requires searching a space of configurations to find a path that corresponds to a set of the operations that will solve the problem.

The planner takes as an input an initial and a final state and an environment description, and produces an overall plan that will take the robot from the initial to the final state. In the case that the command is to move the robot from one room to another the planner finds the best sequence of movements between rooms until it reaches the final destination. Inside each room it finds also the best movement path taking into account the known obstacles, that represent some of the objects in the room. Each of the objects in the environment has a data base represen-

tation that includes a polygonal description shape of the object[4]. Thus, the planner uses this information to find the best path avoiding the polygons that interfere with the goal.

A. Potential field navigation

Under potential field navigation the robot it is considered as a particle that is under the influence of an artificial potential field U whose local variations reflects the free space structure and it depends on the obstacles and the goal point that the robot needs to reach [5].

The potential field function is defined as the sum of an attraction field that push the robot to the goal and a repulsive field that take it away from the obstacles. The movement planning is done by iterations, in which an artificial force is induced by

$$\vec{F}(q) = -\vec{\nabla}U(q) \tag{1}$$

that forces the robot to move to the direction that the potential field decrees, where $\vec{\nabla}$ is the gradient in q and $q = (x, y)$ represents the coordinates of the robot position.

The potential field is generated by adding the attraction field U_{atr} and the repulsive field U_{rep}

$$U(q) = U_{atr}(q) + U_{rep}(q)$$

thus

$$\vec{F}(q) = \vec{F}_{atr}(q) + \vec{F}_{rep}(q)$$

where

$$\vec{F}_{atr}(q) = -\vec{\nabla}U_{atr}(q)$$
$$\vec{F}U_{rep}(q) = -\vec{\nabla}_{rep}(q)$$

A.1 *Attractive potential field*

The attractive potential field creates an attraction force through the goal configuration of the robot. It can be considered a parabolic field of the following form

$$U_{atr}(q) = \frac{1}{2}\varepsilon_1 \|q - q_{destination}\|^2$$

where $q_{destination}$ represents the coordinates of the destination.

A.2 *Repulsive potential field*

The goal of the repulsive potential field is to create a force that takes away the robot from the obstacles, this is obtained using a potential value that tends to infinite in the surface of the obstacle and decreases as the robot goes away from it. The following equation shows a field with the previous characteristics

$$U_{rep}(q) = \frac{1}{2}\eta \frac{1}{\|q - q_{obstacle}\|^2} \tag{2}$$

where $q_{obstacle}$ represents the coordinates of the obstacle. For several obstacles, the total field potential is the superposition of the individual potential field of each obstacle,

$$U_{rep}(q) = \sum_{i=1}^{k} U^k_{rep}(q) \qquad \text{where } k = \text{obstacle number}$$

The forbidden areas are used to find the repulsion forces that act on the robot. There are several approaches to find the repulsion force that a forbidden area acts on the robot, one of them is to find the centroid of the forbidden area and consider that all the mass is concentrated in that point.

A.3 *Path Generation Using Potential Fields*

There are several methods for planning using potential fields, one of them is to use the gradient vector to guide the robot from the initial position to the goal. Defining the unitarian vector pointing to the gradient direction

$$\vec{f}(q) = \frac{\vec{F}(q)}{\|\vec{F}(q)\|}$$

in this way the movement in discrete times is defined by

$$q_{i+1} = q_i + \delta_i \vec{f}(q) \tag{3}$$

Where δ_i is the movement size that robot can execute.

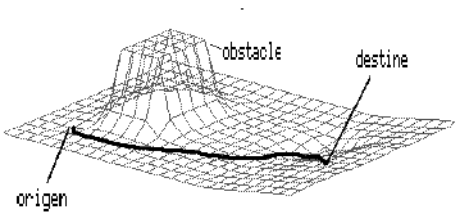

Fig. 2. Path found using potential field

As we can see from figure 2 the growing structure represents an obstacle that generates a repulsive field, the goal generates an attractive field in a parabolic form close to it and a form linear far away that it is represented by an inclination in all the generated surface. The output of the planner is a set of points $(x_1, y_1, x_2, y_2 ... x_n, y_n)$ that it takes the robot from an origin to a destination.

One of the main problems using this technique for planning the robot's movements is that the planner can get stuck in a local minimum where the attraction force and the repulsion forces cancel each other, thus the movement of the Robot is zero or it oscillates around a path.

As we can see from figure 3 when the planner found an obstacle in the middle of the path between the origin and the goal it got stuck in it, oscillating back and forth, due to the repulsion and attraction forces. First the repulsion forces repealed the robot from the obstacle, and when the robot was a little far away from it the attraction force pushed it back to the obstacle, and then the repulsion force acted again repeating the whole process.

We propose a method to eliminate this by putting additional attraction forces in the space. An expert system uses the knowledge of known obstacles to put intelligently these additional attraction forces in places that will take the planner out of the place where is stuck, specifically in some of the vertices of the obstacles, as is shown in figure 4.

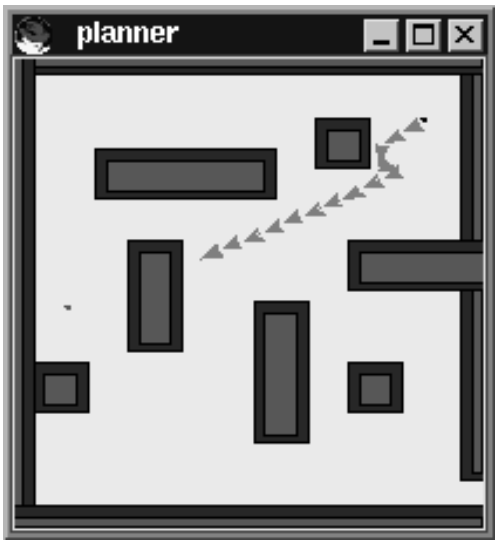

Fig. 3. Planner stuck in a local minimum

Basically the expert system finds the obstacle in which the planner got stuck, then using its vertices $V = \{v_1, v_2, ..., v_N\}$ it selects the vertices $\{v_i, v_{i+1}, ..., v_{k-1}, v_k\}$, where v_i is the closest vertice from the stuck point, v_{i+1} is the clockwise vertice from v_i and v_k is the closest vertice to the goal. Using these selected vertices the expert system first puts a new goal to reach at $v_i + \delta$ disabling the original goal, after the goal in $v_i + \delta$ is reach a new goal is issued at the next selected vertice and so on until $v_k + \delta$ is reached. Finally the original goal is set again.

In the figure 4 we can see that three additional attraction points were put in the corners of the obstructing obstacle to take the robot away from the stuck point and direct it to the destination.

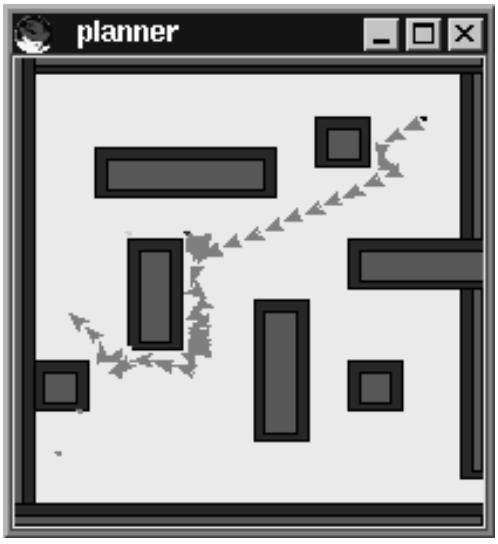

Fig. 4. Path found setting attraction forces in the vertices of the obstructed obstacle

III. *Greedy$^+$ Planner*

The Greedy$^+$ planner uses Euclidean distance and a grid of planning points in an 8-connexity. The mesh size and connexity model are parameters. Obstacles are represented as polygon lists. For higher efficiency, the grid points inside a rectangular bounding box around the obstacles are marked as obstacle points. Horizontal and vertical steps are legal if they do not touch obstacle points. Diagonal steps are legal if none of the points in the square bounding box of the step are obstacle points. If a step is generated by an oracle, it is marked as 'distance to goal is zero' (or even negative, if there is a hierarchy of oracles involved) and thus automatically selected among the candidate steps by the greedy planner.

Greedy$^+$ Algorithm

1. from the current position, generate 8 candidate steps, see figure 5.

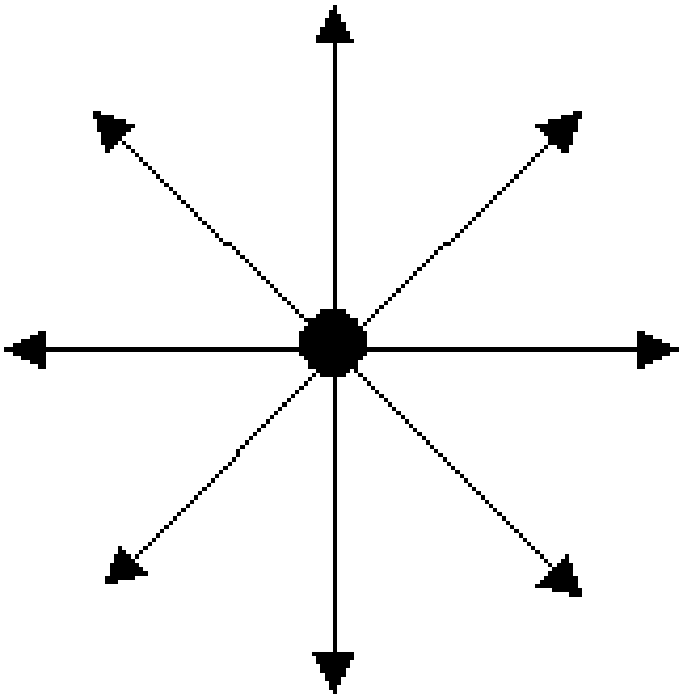

Fig. 5. Eight candidate steps

2. for each step, calculate distance to goal resulting from taking this step, figure 6 shows the distances.
3. depending on oracle preconditions, generate oracle steps; oracle steps are marked with resulting distance zero and thus take precedence.
4. check steps for legality; steps are legal if they do not touch obstacles
5. execute the legal step marked with the lowest resulting distance; since an oracle steps is marked zero, it will be executed if it is legal
6. if goal not yet reached, goto 1.

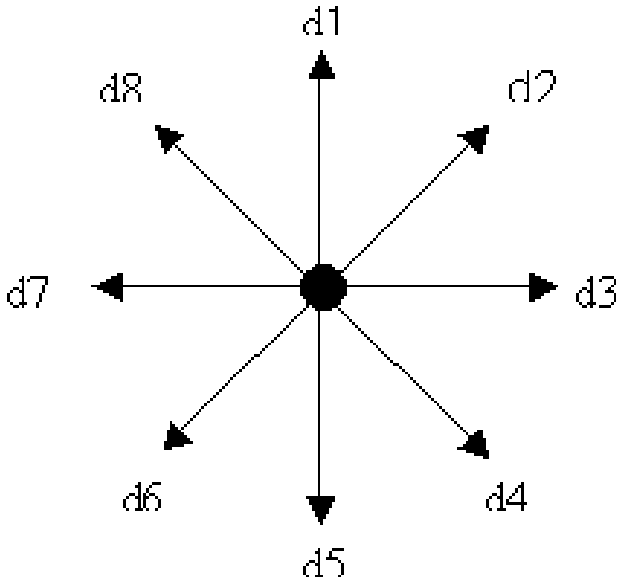

Fig. 6. distance to goal for each step

Sample oracle
Oracle Name: oracle-north-left
Oracle Rule:
IF the goal is north of the robot
AND there is a close obstacle north of the robot,
AND the obstacle continues until the left wall
THEN go east

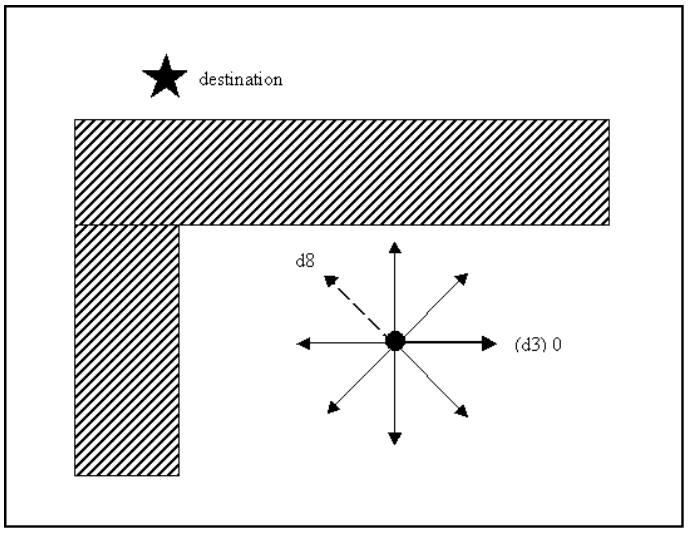

Fig. 7. oracle-north-left

Figure 7 shows the eight standard steps with resulting distances to to the goal. If there were no oracle step, the north west step would be chosen, since it its marked with the lowest resulting distance. But since an oracle step to the east was in fact generated, it takes precedence and it will be executed (if it is legal).

There are 12 active oracles, all very similar to the "oracle-north-left" described above. The oracles only become active if there is an obstacle "in the way". The full list of oracles in this research are: oracle-north, oracle-north-right, oracle-north-left, oracle-south, oracle-south-right, oracle-south-left, oracle-east, oracle-east-right, oracle-east-left, oracle-west, oracle-west-right, oracle-west-left.

Greedy$^+$ comes with no guarantees. It is unlikely to find the optimal path, and it might not find any path, even if several exist. Greedy$^+$ only explores a single path, often without any backtracking. This means that it can be considered an on-line or anytime planner. A step generated by the planner can be executed immediately, at the cost of non-optimality.

While Greedy$^+$ is non-optimal, it will produce a reasonable plan very quickly in a wide range of situations. The power of Greedy$^+$ depends heavily on the oracles, which can use a number of AI techniques and are a means of integrating symbolic reasoning with the numeric greedy planner.

Figure 8 shows an example of a path found by the greedy algorithm.

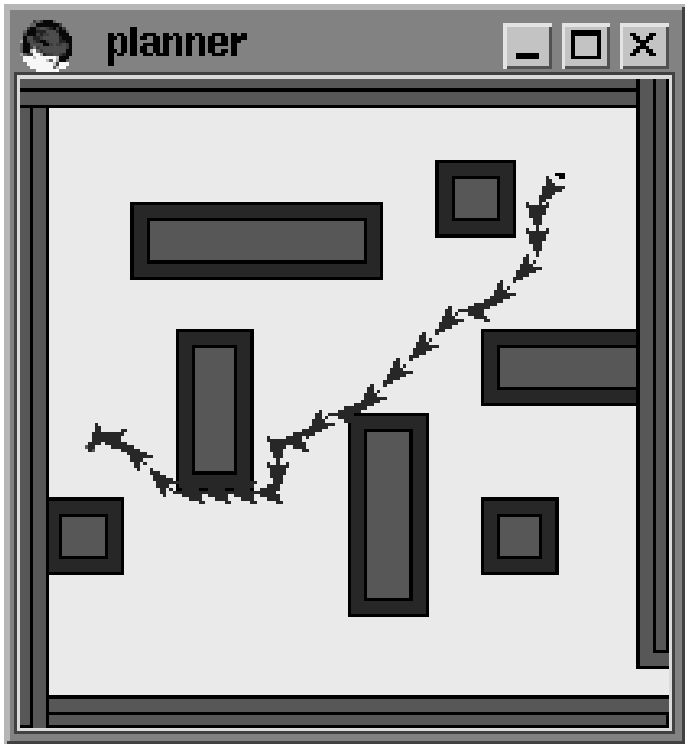

Fig. 8. Path found using the Greedy$^+$ Algorithm

IV. *Experiments and results*

The Potential Field and Greedy$^+$ planners were implemented in ClipsTK [6], a version of CLIPS, an expert system shell developed by NASA [7]. ClipsTK supports Tcl/Tk-style graphics output.

One of the disadvantages with the potential field planner was the tuning of the constants of the equations for the attractive and repulsive forces, that is, to find the values of ϵ, η_1 and δ_i. The values of these constants depend on

the dimensions of the obstacles as well as the movement steps that the robot can execute. There are several ways to calculate these constants, one is trial and error, and another is using genetic algorithms, in our experiments we used the first approach.

After we tuned the system and modify some rules in the expert system, the potential field planner was able to fix the times the planning procedure got stuck in a local minimum. The potential filed approach also does not guarantee that it would find a path given any environment description, but by modifying the constants and putting new rules the system would be able to find reliable movement paths for a mobile robot.

One of the disadvantages with the Greedy$^+$ planner was that for some environments the oracles did not work properly, but after modifying them and creating new ones the Greedy$^+$ planner also found the paths.

V. *Conclusions*

We proposed two approaches for mobile robots' path planning, one uses artificial potential fields and the other a greedy planner augmented by oracles. For the potential fields approach we proposed a method when the planner gets stuck in a local minimum. We used an expert system that uses the knowledge of the obstacles by putting additional attraction forces in the space, specifically in some of the corners of the obstacles.

In the greedy approach we use a greedy planner augmented by oracles. An oracle recognizes a particular configuration of planning state and obstacles for which it is specialized and makes a better choice than the greedy planner.

We make a comparison of these two methods addressing the weaknesses and strengths of each one.

References

[1] J.P. Muller, *The Design of Intelligent Agents: a layered approach,* Springer-Verlag, 1996.

[2] R.C. Arkin and R. Murphy. *Autonomous Navigation in a Manufacturing Environment,* IEEE Transaction on Robotics and Automation 6(4): 445-452 1990

[3] J. Savage, M. Billinhurst, A. Holden *The VirBot: a virtual reality robot driven with multimodal commands,* Expert Systems with Applications 15 (1998) 413-419, Pergamon Press.

[4] Toms Lozano-Prez, *An algorithm for planning collision-free paths among polyhedral obstacles,* Communications of the ACM, vol. 22, pp.560-570, october 1979.

[5] J.-C. Latombe, *Robot Motion Planning.* Massachussets, USA: Kluwer Academic Publishers, 1991.

[6] J. Savage, Emmanuel Hernandez, Gabriel Vazquez, *ClipsTk, Description Manual,* Internal Document, Laboratorio de Interfaces Inteligentes, FI-UNAM (2002).

[7] J. Giarratano, *CLIPS User's Guide.* Software Technology Branch, NASA, 1994.