INTEGRATING A POTENTIAL FIELD BASED PILOT INTO A MULTIAGENT NAVIGATION ARCHITECTURE FOR AUTONOMOUS ROBOTS

Manikanth Mohan
Kanpur Genetic Algorithm Laboratory(KanGAL)
Indian Institute of Technology, Kanpur, India
mmohan@iitk.ac.in

Dídac Busquets, Ramon López de Màntaras, Carles Sierra
Artificial Intelligence Research Institute(IIIA-CSIC)
Campus UAB, 08193 Belleterra, Barcelona, Spain
{*didac, mantaras, sierra*}*@iiia.csic.es*

Key words: Potential Fields, Obstacle-Avoidance , Pilot, Autonomous Robot Navigation, Multi-Agent Systems, Bidding

Abstract: In this paper we present a new Pilot for a Multi-Agent based control architecture for Autonomous Robots. This Pilot is based on the use of virtual potential fields and is easy to implement yet effective. The Pilot functions as an autonomous agent in a complex Multi-Agent Architecture for the control and navigation of an autonomous robot. In this architecture, various agents are responsible for different tasks, and they might have to compete and cooperate for the successful completion of a particular navigation mission. The interaction between different agents is achieved through the use of a bidding mechanism. In this respect, we also present a way to define the bid for the new pilot, so that the pilot can be integrated easily into the Multi-Agent Architecture. The pilot has been tested both on simulations and navigation experiments involving a real robot and it has given successful and encouraging results. We also deal with some problems of this pilot and ways to get around them.

1 INTRODUCTION

Robot systems for navigating through unknown environments has been a focus area of research for many years now, with many application areas including space-crafts (rovers for Mars, Moon), handling hazardous materials, and search and rescue missions, to name a few. Real time obstacle avoidance is a major concern in Robotic systems (Arkin, 1998; Latombe, 1996) for their critical importance to the success in any real life application of robotics, more so in unstructured unknown environments. (Manz et al., 1991) presents a good comparison of some common real time obstacle avoidance methods. Like in the applications mentioned above, the robot must be able to start in an unknown location and navigate to a desired target. In such a case the inability to bypass or take preventive action when confronted by an obstacle can jeopardize the whole exercise and even endanger the safety of the robot. The use of virtual potential fi elds for obstacle avoidance was introduced in (Khatib, 1985) and since it has been explored and discussed to a great deal (Masoud and Masoud, 2000; Khosla and Volpe, 1988). A modifi ed implementation based on potential fi elds, along with a good analysis of the method can be found in (Koren and Borenstein, 1991). Using this as a base idea we have developed an agent,the *Pilot*, which among other functions, also takes care of real time obstacle avoidance. This agent is a part of the overall Multi-Agent System briefly described in the next section for landmark based navigation of the robot. See (Busquets et al., 2003) for details.

2 THE ROBOT ARCHITECTURE AND MULTI-AGENT NAVIGATION SYSTEM

The overall robot control architecture is composed of three systems: (i) the *Navigation System* (ii) the *Pilot System* and (iii) the *Vision System*. Figure 1 shows these systems and the forms of interaction between them. Each system competes for two available resources of the robot: motion (via control over the wheels) and vision (through control over the camera). The Pilot is responsible for all motions of the robot. It initiates these motions to carry out commands from the Navigation system and (independently) to avoid obstacles. The Vision system is responsible for identifying and tracking landmarks (including the goal).

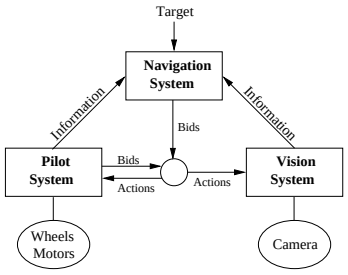

Figure 1: The Robot architecture.

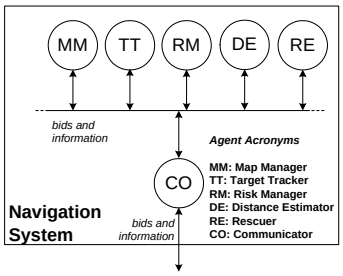

Figure 2: The Multi-Agent Navigation System.

It also implements a short-term memory (called the Visual Memory), which stores the location of known landmarks and obstacles in the environment. Finally, the Navigation system is responsible for choosing higher-level robot motions to move the robot to a specified goal. It does this by the use of a landmark based navigation method. This requires requesting the Vision system to identify and track landmarks, to build a map of the environment, and requesting the Pilot to move the robot toward the goal position or toward some intermediate target position. From this brief description, two observations can be made. First, these three systems must cooperate to achieve the overall task of reaching the target. For instance, the Pilot needs the Vision system to identify obstacles, while it needs the Navigation system to select a path to the goal. Second, the systems can also compete — there are some tradeoffs between them. For example, both the Pilot and the Navigation system compete for the Vision system. The Pilot needs vision for obstacle avoidance, while the Navigation system needs vision for landmark detection and tracking.

To manage this cooperation and competition, we have earlier proposed the use of a competitive coordination system based on a *bidding mechanism*. In the overall robot system, the Navigation and the Pilot systems generate bids for the services offered by the Pilot and Vision systems. These services are to move the robot toward a given direction, and to move the camera and identify the landmarks found on its viewfield, respectively. The service actually executed by each system depends on the winning bid at each point in time. The bids represent the urgency of each system for having a service engaged. The bids are in the range [0,1] with higher bids indicating a greater urgency to have the service engaged. The Navigation system, is implemented as a multi-agent system, where each agent is competent in a specific task (Figure 2). This system is composed of four agents with the following responsibilities:

- *Target Tracker* - keep the target located with maximum precision and reach it,
- *Risk Manager* - keep the risk of losing the target low,
- *Rescuer* - recover from blocked situations, and
- *Map Manager* - keep the information on the map consistent and up-to-date

Depending on its responsibilities and the information received from other agents, each agent proposes which action the Navigation system should take. There is an additional agent, the *Communicator*, which manages the communication between the Navigation system and other systems. Again, we find that the agents must cooperate, since an isolated agent is incapable of moving the robot to the target, but they also compete, because different agents want to perform different actions at the same time. Like the overall system, the Navigation system itself employs a bidding mechanism to coordinate its agents. Each agent bids for the action it wants the robot to perform. These bids are sent to the *Communicator*, which determines the winning action. The selected action is then sent as the Navigation system's bid for the services of the Vision and Pilot systems. See (Busquets et al., 2003) for details on this multi-agent system.

3 THE NEW POTENTIAL FIELD BASED PILOT

The Pilot is responsible for safely commanding the motors that control the robot to move in a given direction. It implements the requests for motion received from the Navigation system (if they are safe), and is responsible for avoiding obstacles and for providing reflex reaction when the robot makes an unexpected collision (maybe due to lack of knowledge or imprecise knowledge of an obstacle). With the information coming from the Vision system and Visual Memory, it uses a Potential Field (PF) based technique, to decide the turn necessary for avoiding collisions. The Pilot also generates a bid, indicating the urgency of the action it is proposing. Since obstacle avoidance is of maximal importance, the bid, in critical conditions, should be higher than the bids coming from the

Navigation system. However, it should not be set to the highest possible value, 1, so that there is the possibility of adding a new system that can override the Pilot (e.g. a manual override). Additionally the Pilot also makes bids for camera control, to look ahead when required (e.g. for gathering additional information about obstacles). This bid is based on a function that raises the bid depending on the distance traveled since the last time the robot looked forward:

$$bid_{look} = \left(\frac{d_l}{d_m}\right)^s \quad (1)$$

where d_l is the distance covered since last look, d_m is the maximum distance allowed to travel without looking ahead and s is a scalar which determines the growth of the function. Even though this is also an important function of the Pilot, in this paper we only focus on the obstacle avoidance module of the Pilot.

3.1 Mathematical Formulation: Point Obstacles and the Target

For developing the PF-method, we follow the usual approach (like in electrostatics) where the magnitude of the virtual force is dependent on the square of the distance between the robot and the obstacle. The direction of the force is that of the line joining the obstacle and the robot. If the body in question is an obstacle, then the force is repulsive in nature meaning that the robot and obstacles hypothetically have *"charges"* of the same sign. The target *"charge"* is of the opposite nature, thereby providing an attractive force on the robot. We can express the virtual force on the robot due to an obstacle o by (we denote vectors in boldface):

$$\mathbf{F_o} = K\frac{Q\,q_o}{d^2}\hat{\mathbf{u}} \quad (2)$$

where,

$\mathbf{F_o}$ = Force vector associated with obstacle o
K = Constant of proportionality
Q = *"Charge"* associated with the robot
q_o = *"Charge"* associated with the obstacle
$d = \|\mathbf{x_r} - \mathbf{x_o}\|$ ($\|.\|$ denotes Euclidean norm)
$\mathbf{x_r}$ = Position vector of the Robot
$\mathbf{x_o}$ = Position vector of the obstacle
$\hat{\mathbf{u}}$ = unit vector in the direction of o, i.e.

$$\hat{\mathbf{u}} = \frac{\mathbf{x_r} - \mathbf{x_o}}{\|\mathbf{x_r} - \mathbf{x_o}\|}$$

We follow the convention that the obstacles and the robot are negatively charged while the target is positive. Therefore, positive forces are repulsive in nature while negative forces are attractive. Instead of assigning values to Q and q_o we take them to be both 1 and adjust the magnitude of the force by manipulating K. Since the information about the position of the obstacles, obtained from the Visual Memory, is in the form of fuzzy coordinates, the obstacles are initially represented as points (for landmarks and simple obstacles) and lines (for linear obstacles), but are later *"grown"* to become circles and rounded rectangles respectively (Figure 3). These areas, hence forth referred to as prohibited areas, represent the obstacles they contain. In our case, the growing size is kept slightly larger than the diameter of the robot. The calculation of the forces is made using crisp coordinates, given by the center of the *"grown"* circles. We represent the radius of this circle by R_{min}. If the robot is inside this prohibited region, (a situation we denote by the name *IN_DISTRESS*) then the force exerted by this body is of constant magnitude away from the obstacle as given in Equation (3). Therefore, the final equation expressing the force due to an obstacle is given as in Equation (4):

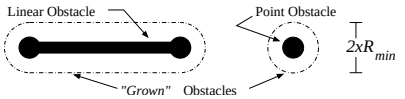

Figure 3: *Growing* of obstacles. Dark areas show actual obstacles and dotted lines represent the *grown* obstacles.

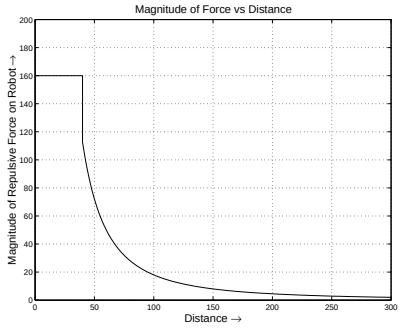

Figure 4: Variation of magnitude of force with distance for a single obstacle instance.

$$\mathbf{F_{max}} = \frac{K}{R_{min}^2}\hat{\mathbf{u}} \quad (3)$$

$$\mathbf{F_o} = \begin{cases} K\frac{Q\,q_o}{d^2}\hat{\mathbf{u}} & \text{if } d > R_{min} \\ \mathbf{F_{max}} & \text{if } d \le R_{min} \end{cases} \quad (4)$$

This way of definition ensures that we have to work only with bounded forces (unbounded forces cause problems during computation and implementation). Additionally, this gives a very easy method for generating the associated bid (described in Section 4). Figure 4 shows the variation of the magnitude of the force with the distance to a single obstacle.

To take into account the direction of the target, and the fact that no matter where the robot is, in normal circumstances, our aim is to reach the target (implying that the target should always play a major role in deciding the angles given by the Pilot) we take the attractive force of the target to be a constant value. We therefore define the target force by:

$$\mathbf{F_t} = A_t \hat{\mathbf{v}} \tag{5}$$

where,

$\mathbf{F_t}$ = Force vector associated with Target t
A_t = Constant of proportionality
$\mathbf{x_t}$ = Cartesian coordinates of the Target t
$\hat{\mathbf{v}}$ = unit vector in the direction of t, i.e.

$$\hat{\mathbf{v}} = \frac{\mathbf{x_r} - \mathbf{x_t}}{\|\mathbf{x_r} - \mathbf{x_t}\|}$$

So far we have defined the forces for the point obstacles and the target, we are yet to define the force for linear obstacles, which we treat in the next section.

3.2 Linear Obstacles

The natural extension for calculating repulsive forces for linear obstacles is by integrating the force (per unit length) over the length of the obstacle. But, this causes a non-uniform distribution of forces, for example, it is stronger at points near A in (Figure 5) which lie near the perpendicular bisector of the line obstacles, compared to the sides (near B). Due to this,

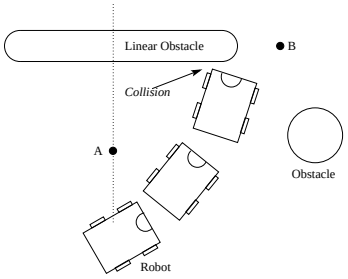

Figure 5: Impending crash due to the effect of a nearby obstacle

the presence of another obstacle nearby can cause the robot to collide into the linear obstacle. So, the magnitude of the repulsive force is calculated by taking the nearest point from the robot to the linear obstacle, while the direction is given from its center (see Figure 6). This new method for dealing with linear obstacles ensures that the repulsive force is high always whenever the robot is near the linear obstacle, independent of the orientation with respect to the robot.

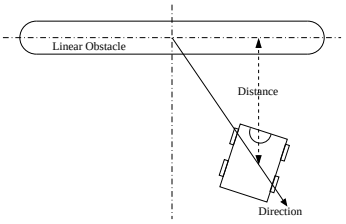

Figure 6: Calculation of force for a Linear Obstacle

3.3 Diverting Targets

Under the overall architecture of the robot, the Navigation system decides and communicates the current target to the Pilot. This may differ from the initial target given at the beginning of the mission because the Navigation system can set up temporary targets, in the form of crossing a virtual line (called diverting target) joining two landmarks to help the robot to escape from a difficult or blocked position. Here, we would like the Pilot to lead the robot across the virtual line (diverting target) without colliding with the landmarks at both ends. Simply putting an attractive force towards the line can cause the robot to crash into one of the landmarks at the end. Another easy way out is by putting an attractive force towards the middle of the diverting target. But this might not be optimal, specially if the diverting line is very long (Figure 7). We therefore approach this problem in a different manner. Consider the line target to be the line AB in Figure 8. We define a corridor perpendicular to this line target, which is at a safe distance from the landmarks A and B at both ends. If the robot is inside this corridor, the attractive force is simply perpendicularly towards the line target; otherwise it is towards the center of the diverting target. So, in Figure 8, if the robot is at C (outside the corridor), then the force is towards the center of the diverting target, while at D (inside the corridor) the force is simply perpendicular towards the line target. This makes sure that the Pilot leads the robot across the lines without risking a collision.

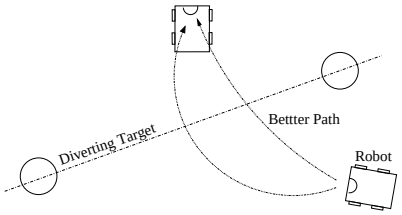

Figure 7: Two possible paths for crossing the diverting target line

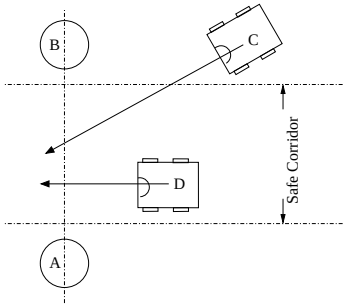

Figure 8: New treatment of diverting target lines, using an imaginary *safe-corridor*

3.4 Putting it all together

Once we have defined the virtual forces exerted by obstacles and targets, we are ready to define how the total force is computed. The total virtual force on the robot is found by vector addition of all obstacle and target forces. So, if $O = \{\text{All known obstacles in the current Visual Memory}\}$, then the net force on the robot $\mathbf{F}_{net}$ is given by:

$$\mathbf{F_{net}} = \sum_{\forall o \in O} \mathbf{F_o} + \mathbf{F_t} \tag{6}$$

We can express this in terms of its x and y components as: $\mathbf{F_{net}} = F_x\mathbf{i} + F_y\mathbf{j}$ and the turn angle θ by

$$\theta = tan^{-1}\left(\frac{F_y}{F_x}\right) \tag{7}$$

The direction of motion of the robot is in the direction of the net force (subject to certain conditions, described in Section 5) and hence Equation (7) defines the turn angle returned by the Pilot. Along with this, the Pilot also returns a bid (Section 4) which in some sense indicates the urgency of taking this turn.

In our implementation the values of the parameters are: $K = 18$, $A_t = 50$ and $R_{min} = 0.40m$. These values were arrived at after considerable experimentation using the simulator and the real robot. However, we could have also got these values analytically. The following relationship can be derived:

$$A_t = \frac{0.76 \cdot K}{d_{min}^2} \tag{8}$$

where d_{min} is half of the minimum distance between two obstacles the robot would go through (i.e. the minimum distance between landmarks A and B in Figure 8). We do not include the whole analysis due to space limitations. With the parameters found experimentally, $d_{min} = 0.52m$, thus, the Pilot is able to drive the robot between two obstacles 1 meter apart.

Additionally, since the positions of landmarks and obstacles are measured and maintained with respect to the robot as the origin, $\mathbf{x_r}$ is always (0,0) while $\mathbf{x_o}$'s and $\mathbf{x_t}$ are the egocentric position vectors (or coordinates) of the obstacles and the target respectively.

The direction of the net forces for an example environment consisting of two points obstacles, two linear obstacles and a point target is shown in Figure 9.

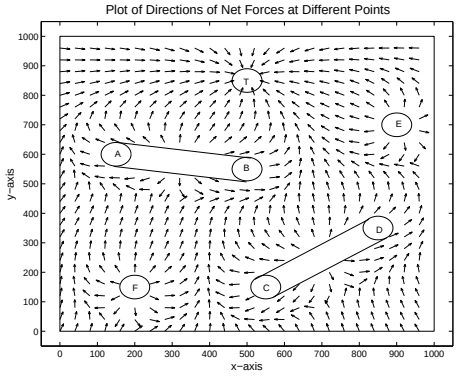

Figure 9: Direction of net forces for a sample environment.

3.5 Reflex Actions for Accidental Collisions

The Pilot is also responsible for providing reflex reactions when the Robot makes an accidental collision with an obstacle. These accidental collisions can happen due to a variety of reasons, like it is possible that the Vision System failed to notice the presence of the obstacle, or that it has some imprecise position data. In either case, due to the lack of knowledge, the robot is unable to avoid it. In the event of an accidental collision, the Pilot communicates this information (obtained from the bumpers) to the Navigation system and takes immediate reflex action. The reflex action taken by the Pilot consists of firstly, stopping the forward motion, asking the cameras to pan the surrounding to get more information of the new obstacle, then backing up a certain safe distance and calculating the new forces, after the inclusion of the new obstacle. If the robot is required to turn a large angle (say, more than 160^o), the Pilot sets the speed to be negative (i.e. starts moving backwards) and turns the complement of the angle. This greatly improves the motion of the robot, and prevents the robot from making unnecessary large turns. But for avoiding moving backwards a lot (which is its blind area) the backward movement is allowed only if the total distance covered backwards is less than a given threshold.

4 INTEGRATION INTO THE MULTI-AGENT SYSTEM: BIDDING

As mentioned in Section 2, the Pilot receives requests from the Navigation system to move the robot towards the target and it also proposes its own actions for real-time obstacle avoidance. Thus, it has to compete with the Navigation system in order to have its proposed actions executed. For this competition, the Pilot also sends a a bid along with its turn angle. It receives a similar bid from the Navigation System via the *Communicator*. The Pilot executes the turn of the winning bid (the higher bidder wins). This competition ensures that if the robot is near an obstacle (or heading towards one), the Pilot's action is executed because its bid will be higher than that of the Navigation system. Similarly, if the robot is in a safe region, then the Pilot's bid is low. This helps us in achieving what we want i.e. that the Pilot should come into play only when the robot might get into trouble and not interfere with the navigation otherwise. Usual PF based methods suffer from this problem, that they interfere with the normal movement of the robot even when it is unnecessary. But, by treating the new Pilot as an agent in a Multi-Agent System, where agents compete and cooperate for the resources of the robot, we are able to work around this problem.

Now we need to come up with a good way to define the bid of the robot, so that it meets the needed requirements (i.e. the bid be high, if the robot is near or heading towards an obstacle and is low otherwise). But achieving this is not difficult and is in-fact very intuitive. We first find the maximum repulsive force generated by any individual obstacle, G_{max}. We then define the bid of the Pilot, bid_p as a function of this maximum force, through Equation (9):

$$\text{Let } G_{max} = max\{|\mathbf{F_o}| : \forall \mathbf{o} \in \mathbf{O}\}$$

$$bid_p = 0.9\left(\frac{G_{max}}{|\mathbf{F_{max}}|}\right) \tag{9}$$

In Equation (9), $\mathbf{F_{max}}$ is as defined in Equation (3). The multiplication factor of 0.9 is to scale the bid within the interval [0, 0.9], so that if necessary, we can incorporate another agent in the future which has the ability to overrule the bid of the Pilot. We see that by limiting the maximum force due to an individual obstacle to G_{max}, the generation and definition of the Pilot's bid is easy. The reason why we use the maximum repulsive force instead of the net force or the total repulsive force is because the presence of obstacles nearby can hamper the magnitude of the force. By considering the maximum repulsive force, it becomes very unlikely that the bid of an agent from the Navigation System will win, if the robot is very near to any obstacle. Figure 10 graphically shows the variation of the bid generated by the Pilot for the environment shown in Figure 9.

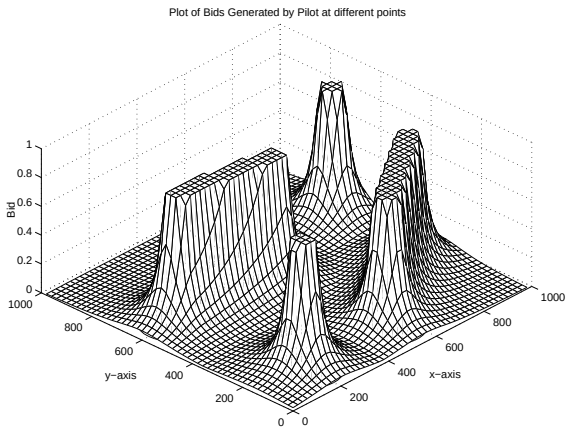

Figure 10: Bids Generated by Pilot in a sample environment (shown in Figure 9).

5 SOME PROBLEMS AND SOLUTIONS

The Pilot at this stage was able to perform the basic duties of avoiding the obstacles and also lead the robot to the target. Over some runs of the robot, some problems were noticed. These are described below, along with the changes made to counter them.

Abrupt turns: Sometimes it is noticed that the robot makes a lot of abrupt turns, like those shown in Figure 11. The presence of an obstacle on either side of the robot causes the forces to change very rapidly in the vicinity of the obstacles, especially in the regions along the line between them. Even though this is expected with PF methods, we try to moderate these quick turns by averaging the obtained angle with that computed in the previous step. This helps in decreasing this problem to some extent.

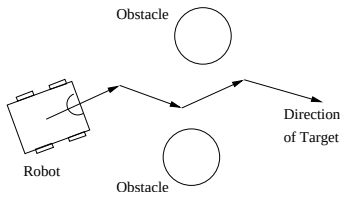

Figure 11: Robot making abrupt oscillating turns while trying to move between two close obstacles.

Crashes in spite of being in IN_DISTRESS condition: When the robot is very near any of the obstacles

(i.e. in a state of IN_DISTRESS), finding the turn angle from the total force can give an angle not sufficient enough for avoiding the obstacle. To take care of this, if the robot is in IN_DISTRESS state, we make some changes. The force of the obstacle which caused the IN_DISTRESS condition is increased to 150% of its actual value to give it more weight. The turn angle is found only from the net repulsive force, i.e. we temporarily ignore the target, because the immediate priority is to avoid the obstacle. Additionally the averaging of the angles (to avoid abrupt turns) is temporarily not used in this situation, because now abrupt turns help avoiding the obstacle.

More than One Obstacle in Same Direction: Consider the situation shown in Figure 12. There are three repulsive forces on the robot, one from each of the obstacles A, B, and C producing a strong repulsive force away from the region. Technically, a force from A

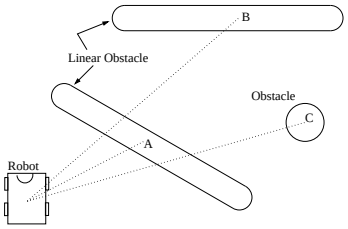

Figure 12: More than one obstacle in same direction

is sufficient at this position of the robot, because the other obstacles (B and C) are anyway shielded by A. We could even have a very bad extreme case like the one shown in Figure 13. In this case it is not correct to compute the force for all the point obstacles which are hidden behind the linear obstacle. To overcome this, we consider only those obstacles which are directly facing the robot. If the line segment from the obstacle (or center of linear obstacles) to the robot intersects any other obstacle, it doesn't contribute any repulsive force. So in the above case (Figure 12), only the force

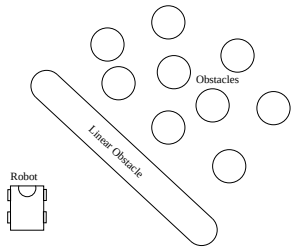

Figure 13: An extreme case of obstacles in the same direction.

from the line segment A would be considered. The line segment joining the center of B to the robot and from C to the robot intersect A, hence these 2 obstacles provide no repulsive force. In the later case, (Figure 13), the point obstacles hidden from the robot would affect the robot only if the line segments from them don't intersect with the line obstacle.

Effect of Obstacles even after they have been passed: The obstacles continue to affect the motion of the robot even after the robot has passed by them (Figure 14). Even though this problem is expected from PF based approaches, we try to reduce this by setting the repulsive force to be zero, if the angle between the net repulsion and the net attraction is less than 90^o. For example, in the case shown in Figure 14, if $\alpha < 90^o$, we set the repulsive force to zero. Even though this doesn't completely solve the problem, it helps turning towards the target easier than without this modification. Other ways that could be used to overcome this is by taking into account the orientation of the robot and its direction of motion.

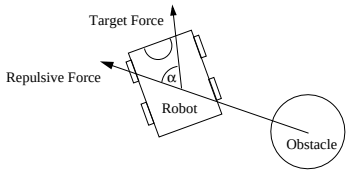

Figure 14: Effect of an Obstacle even after it has been passed

6 EXPERIMENTAL RESULTS

This Pilot was implemented and tested on a real Pioneer 2AT robot and was found to give satisfactory results. To make a comparative study, the performance of the new Pilot was tested against the Pilot, that was used in the architecture till then. The earlier Pilot, called the *Geometric Pilot* was simple and used a nearest obstacle avoidance (using geometric principles) algorithm. Both these pilots were tested on both simulated and real experiments on different environments. We measure the performance of the Pilot in terms of the time taken to reach the target and

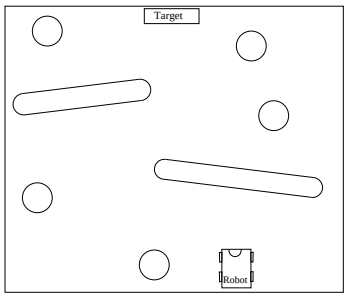

Figure 15: Sample environment used for the experiments.

Table 1: Comparative Results of the Performance of the two Pilots on the same environment. (*GP denotes Geometric Pilot and PFP Potential Field Pilot*)

	Time Taken (in sec)		Path Length (in cm)	
	Avg	Std Dev	Avg	Std Dev
GP	297.33	*81.91*	1532.04	*429.33*
PFP	243.37	*99.05*	1194.92	*198.23*

the length of the path taken. Table 1 shows the averages and the standard deviations of the time taken and path length for 45 reruns on one of the environments (shown in Figure 15), of size 8 by 8 meters. The table clearly shows that the performance of the PF based Pilot is much better than the Geometric Pilot in both of these measures. The PF-Pilot improves the time taken by an average of 18.14%, while it improves the path length by around 22%. Therefore the new PF-Pilot is able to signifi cantly improve both the time and the path taken by the robot. This shows the effectiveness of the new Pilot as a part of the overall Multi-Agent architecture.

7 CONCLUSION AND SCOPE

In this paper, we have presented the development of a new Pilot which uses a virtual potential fi eld strategy to compute the turn angles for avoiding obstacles. The advantage of using a PF method is that it goes in tandem with fuzzy coordinates, because a small imprecision (which are inevitable in real implementations), does not affect the resulting fi eld considerably. By using the PF method to build an agent, as part of a Multi-Agent system, we are able to work around some problems that are inherent to PF methods. The unique way for defi ning the bid, ensures that the Pilot intervenes only when necessary. The maximum bid of the Pilot is kept higher than the other systems to ensure that the Pilot gets higher priority in critical conditions. As shown by the results of the experiments done, the new Pilot provides much better performance and is able to considerably improve the performance of the robot, by saving both time and path-length. We have also found ways to deal with some problems faced by the new Pilot. The Pilot also provides reflex reactions in response to accidental collisions. A simplifi cation that we have assumed here is that obstacles are either linear in shape or are points. This assumption is usually not valid in actual outdoor environments, but most indoor environments can be approximated with such lines and points. The extension of PF methods to deal with arbitrary shapes can be very complex and an approximation to lines and points might be easier and effective rather than an exact method. Through this paper, we have attempted to give a new way to design intelligent agents from simple ideas with real applications. Such ideas, though applied here only to obstacle avoidance, may have many more applications and can be used to model complex, intelligent behaviors and systems with multiple agents.

Acknowledgements

This work has been partially supported by the MCyT's project QualNavEx DPI2003-05193-C02-02 and CIRIT's project CeRTAP.

REFERENCES

Arkin, R. C. (1998). *Behavior-Based Robotics*. MIT Press, Cambridge, MA.

Busquets, D., Sierra, C., and L´opez de Màntaras, R. (2003). A multi-agent approach to qualitative landmark-based navigation. *Autonomous Robots*, 15:129–153.

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation*, pages 500–505, St. Louis, Missouri.

Khosla, P. and Volpe, R. (1988). Superquadric artificial potentials for obstacle avoidance and approach. In *Proc. of the IEEE Conf. on Robotics and Automation*, pages 1778–1784, Philadelphia, USA.

Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. of the IEEE Conf. on Robotics and Automation*, pages 1398–1404, Sacramento, California, USA.

Latombe, J. (1996). *Robot Motion Planning*. Kluwer Academic Publishers, UK.

Manz, A., Liscano, R., and Green, D. (1991). A comparison of real-time obstacle avoidance methods for mobile robots. In *Second Intl. Symposium on Experimental Robotics*, Toulouse, France. Springer-Verlag.

Masoud, S. and Masoud, A. (2000). Constrained motion control using vector potential fields. *IEEE Trans. on Systems, Man, and Cybernetics*, 30(3):251–272.