A Temporal Potential Function Approach For Path Planning in Dynamic Environments

Vamsikrishna Gopikrishna, Manfred Huber
Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, Texas, USA
vamsi.g.krish@gmail.com, huber@cse.uta.edu

***Abstract*—A Dynamic environment is one in which either the obstacles or the goal or both are in motion. In most of the current research, robots attempting to navigate in dynamic environments use reactive systems. Although reactive systems have the advantage of fast execution and low overheads, the tradeoff is in performance in terms of the path optimality. Often, the robot ends up tracking the goal, thus following the path taken by the goal, and deviates from this strategy only to avoid a collision with an obstacle it may encounter. In a path planner, the path from the start to the goal is calculated before the robot sets off. This path has to be recalculated if the goal or the obstacles change positions. In the case of a dynamic environment this happens often. One method to compensate for this is to take the velocity of the goal and obstacles into account when planning the path. So instead of following the goal, the robot can estimate where the best position to reach the goal is and plan a path to that location. In this paper, we propose a method for path planning in dynamic environments that uses a potential function which indicates the probability that a robot will collide with an obstacle, assuming that the robot executes a random walk from that location and that time onwards. The robot plans a path by extrapolating the object's motion using current velocities and by calculating the potential values up to a look-ahead limit that is determined by calculating the minimum path length using connectivity evaluation and then determining the utility of expanding the look-ahead limit beyond the minimum path length. This paper will discuss how the potential values are calculated and how a suitable look-ahead limit is decided. Finally the performance of the proposed method is demonstrated in a simulated environment.**

***Keywords*—robotics, dynamic environment, path planning, potential function, harmonic function**

I. INTRODUCTION

There are two approaches to determining how a robot should move in a given environment. In the path planning approach the robot first calculates, based on the obstacle locations, what the best path to the goal location is and then follows it. Path planning generally assumes that the planner has all pertinent information about the world at execution time. If the world suddenly changes, then there is no other option than to scrap the plan generated and plan a new one from scratch. In a reactive control system the robot sets off towards the goal and then modifies it's movements to avoid any obstacles it may encounter. This makes sure that the robot does not need to spend time re-planning in case the environment changes. However the tradeoff is in performance. There is a chance that the robot will end up just tracking the goal, ignoring more optimal ways to reach it.

The environment that a robot is expected to work in can be either static or dynamic. In a static environment the state of the world, i.e. the locations of the goal and obstacle states are fixed. An example for such a static environment is a robot having to navigate in a room with no people in it. The only obstacles are the furniture whose locations are fixed. In a dynamic environment the goal and obstacle locations can change as in the case of a robot having to navigate a busy hallway. The people in the hallway can be considered as obstacles, each having their own velocity and trajectory.

In dynamic environments, robot control is usually achieved by means of a reactive system. This is because a static path planning system would have to constantly re-plan as the state of the environment keeps changing. This adds significant overhead to the execution, frequently making the use of a reactive system more efficient. However, the tradeoff is in accuracy. Since the robot only knows the location of the robot, the goal, and the obstacles at that instant, it could end up just tracking the goal, following it around the world instead of reaching it.

This paper proposes a method for path planning in dynamic environments. The method has its roots in harmonic functions and the potential field approach to path planning. It expands the world in the time dimension to find a reachable goal and plans a path to it. The path is obtained by following the decreasing gradient of the potential value where the potential value is the probability of the robot colliding with an obstacle if it begins a random walk from that point at that time. The amount of look-ahead is determined by observing the properties of randomly generated worlds. In the following sections we shall take a look at some of the background information on path planning in dynamic environments, followed by the proposed method. We will also look at some experimental results

II. BACKGROUND AND RELATED WORK

A. Dynamic motion planning

The dynamic motion planning problem is one in which the robots has to navigate an environment where one or more of the obstacles in the system and/or the goal are moving [1]. Unlike in the static environments the problem cannot be solved by merely constructing a geometric path. Instead a continuous function of time specifying the robot's configuration at each instance needs to be generated. One approach is to add a dimension – time – to the robot's configuration space [1]. But unlike the other two, the time dimension is irreversible. The algorithms used for planning in static environments can be used in dynamic environment with some modifications and extensions, which are aimed at taking the specificity of the time dimension into account.

B. *Path planning vs. Reactive Navigation*

In a path planning system the path to the goal is usually calculated before the robot sets off. The robot has to follow the given plan to reach the goal. In a dynamic environment the velocities of the obstacles and the goal are taken into account when calculating the goal. If any of the variables describing the environment change during the planning process, the path has to be re-planned. This could result in significant overhead if the planning algorithm is used in an environment where the dynamics constantly change.

In a reactive system, there is no actual planning taking place. The robot moves towards the goal, only changing course to avoid any obstacles it may encounter. In a static environment this will result in the robot reaching the goal. However if the environment is dynamic then the robot may end up tracking the goal. One solution would be to predict when the robot can reach the goal and the location of the goal at that time. This however, cannot be done by a reactive system. A path planning system, on the other hand, can be easily modified to handle this.

C. *Existing Approaches*

Current research in path planning for dynamic environments consists of two approaches. One approach is to build an effective reactive system that moves towards the goal, deflecting from the path when an obstacle comes towards it. One such method was proposed by Ge and Cui of The University of Singapore [2]. Their proposed method used a potential field approach, where the attractive potential was a function of the relative distance and velocity between the robot and the goal and the repulsive potential of each obstacle is a function of the relative velocity between it and the robot and the shortest distance between its body and the robot provided that the robot is heading in its general direction and the robot is within the obstacle's influence range. The above described method has a few disadvantages. For example the robot can be trapped in local minima. In that case, the robot just waits at that position until the obstacle moves away. If that does not happen for a long time, the planner uses conventional local minimum recovery methods like wall following to escape local minima. Also, there is a possibility that the goal may sometimes be in an obstacle's influence radius. In that case the robot will never reach it. One solution is to modify the repulsive potential function to include the relative position and velocity between the goal and the robot. A similar approach has been proposed by Poty, Melchior and Oustaloup of Université Bordeaux [3].

Another approach to path planning in dynamic environments was proposed by Wu, QiSen, Mbede and Xinhan [4]. This method calculates potential field values using harmonic functions and then follows the path of negative potential gradient, using fuzzy rules to avoid obstacles.

In another approach [5], the world is represented as a probabilistic roadmap using the static portion of the planning space. An initial trajectory is planned over this roadmap taking into account the dynamic obstacles. This trajectory is continuously modified or improved as and when needed.

In case the robot is capable of 3-Dimensional motion the method proposed in [6] can be used where the environment is represented as an octree and the path is planned by using the potential field generated from each obstacle in the octree.

Finally space-time planes have been proposed which include time as a constrained dimension of the c-space. One such method was proposed by Fabio Marchese and Marco Dal Negro of Università delgi Studi di Milano – Bicocca [7]. This method used Multilayered Cellular Automata to plan paths for multiple robots. The many robots are represented as static obstacles in Space-Time 4D space. Another method proposed by Charles W. Warren of The University of Alabama [8] uses a similar world representation. This method calculates the path of a high priority robot and then plans the paths of other robots around it by extending the real space of the robot into a configuration-space-time. Both these methods consider robots to be the only moving objects in the world. The proposed method uses a similar world representation where the extension of the world into the time dimension essentially converts the moving obstacles into static ones at each time step. The proposed method calculates the potential value as the probability that the robot will collide with an obstacle if it takes a random walk. This is represented as a harmonic function. We shall now look at potential fields and harmonic functions in a little detail.

D. *Potential Fields*

Potential fields have been proposed for obstacle avoidance by Khatib [9]. In the potential field approach the robot in configuration space is acted upon by imaginary forces. The goal produces an attractive force and the obstacles produce a repulsive force. The path the robot follows can be found by calculating the resultant vector of these forces.

Planners that use potential fields have the chance of getting stuck in local minima instead of reaching the goal. A number of approaches have been suggested to solve the local minima problem. One solution is to use vector field histograms (VFH) [9] [10] [11]. We can also place an artificial obstacle at local minima [12]. There has also been some research into designing new potential functions that will avoid the creation of local minima [13] [14]. One such method is to design the potential function as a harmonic Function.

E. *Harmonic Functions*

A harmonic function is a real function with continuous second partial derivatives which satisfies Laplace's Equation [15]. Laplace's equation [16] is given by Equation (1).

$$\nabla^2\phi = 0 \tag{1}$$

Where ϕ stands for the harmonic function and ∇^2 stands for the sum of the second partial derivatives of ϕ. Solutions to Laplace's equations can be computed using certain conditions called boundary conditions. The commonly used ones are restricted forms of the Dirichlet and Neumann conditions [17]. Techniques like Jacobi iteration, Gauss-Seidel iteration or successive over-relaxation can be used to calculate the numerical value of the harmonic function at each grid location [17]. These methods require the environment to be discretized, for example by representing it as a grid based map. Although these methods compute function values on a grid, multi-linear functions are used to interpolate between grid points since such functions are harmonic and smooth.

Harmonic functions have many varied applications in robot control. The most popular application is their use in path planning. If the c-space of the environment can be modeled as a harmonic function, then all the robot has to do is to follow the negative potential gradient to reach the goal and avoid the obstacles. The absence of local maxima and minima and the continuity and differentiability of the harmonic functions guarantees that a planner using harmonic functions generates paths that are correct and complete. Since the gradient is smooth, the paths generated by the planner are well behaved. Using harmonic functions we can build a system that combines the characteristics of both a path planning system (since it calculates the potential values of the locations in the world) and a reactive system (since the robot simply follows the path of decreasing potential and avoids the obstacles) [17] [18] [19].

III. Temporal Potential Function Approach

A. *Internal Representation of the Environment*

Before we look at the proposed method we have to discuss how the world is internally represented inside the system. The world is represented as a three dimensional grid of real numbers in the interval [0, 1]. The x and y axes represent the x, y coordinates of the world. The z axis represents time. The number at any grid location (x_1, y_1, z_1) represents the probability of the robot colliding with an obstacle if it takes a random walk from location (x_1, y_1) at time z_1. The collision probability is one for locations containing obstacles and zero for locations containing goals. Building an initial map that only contains these "boundary points" is easy provided we know the velocities of the obstacles and the goal. We simply start from time step zero, setting a value of zero for goal locations and one for obstacle locations. For each time step, the location of the obstacles and the goal are calculated based on the velocity information provided and the values set to one and zero respectively. The other locations are filled by the method that will be described in Section III-C.

B. *Overview of Proposed Method*

Using the representation introduced in the last section, the proposed method is based on the harmonic function approach and the potential field method of path planning. The basic structure of the proposed method is shown in Fig. 1.

The planner first extrapolates the world to a certain look-ahead time using estimates of the obstacle and goal movements. The look-ahead time is essential to limit the complexity of the extrapolation and of the following potential field calculation and is determined by the following process. The time taken to reach the first goal is found by forward chaining through the space-time map. If the dynamics changes during the forward chaining due to a solution not being found for a long time, the process is repeated. Using the time taken to reach the first goal, the look-ahead value is estimated based on observation of the potential value curves of a set of randomly generated worlds with the same properties as the given world. In particular, the look-ahead time is estimated as the time horizon at which, based on experiences in worlds with similar properties in terms of size, obstacle density, and obstacle dynamics, the expected value of a solution of the corresponding length is significantly lower than the already found, shortest path. The corresponding expected value curves are generated during the first execution of the planner.

After finding the look-ahead limit the planner calculates the potential values of all locations in the world at all time steps from the start to that look-ahead time. The robot reaches the goal by simply following the negative potential gradient at each time step. If the robot ends up at an obstacle due to an error in navigation, the potential values are recalculated. If the dynamics of the environment changes during runtime, the process is repeated form the start.

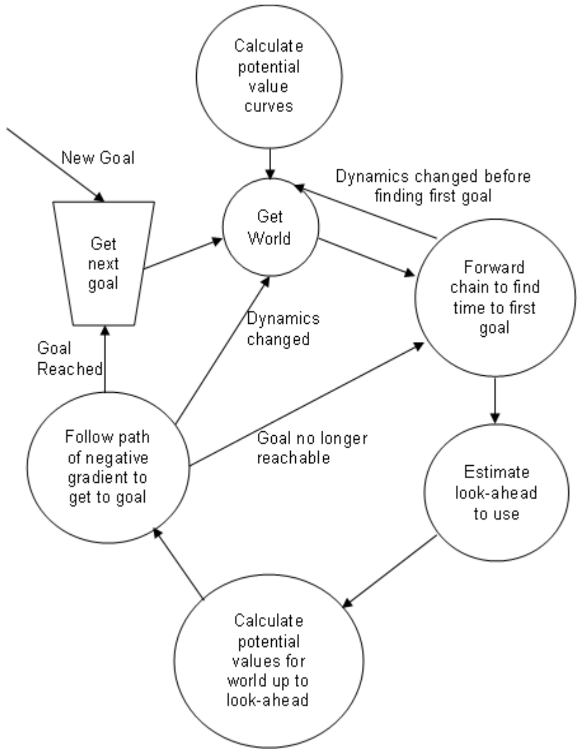
Calculate potential value curves
New Goal
Dynamics changed before finding first goal
Get World
Get next goal
Forward chain to find time to first goal
Dynamics changed
Goal Reached
Goal no longer reachable
Follow path of negative gradient to get to goal
Estimate look-ahead to use
Calculate potential values for world up to look-ahead

Figure 1. Temporal Potential Function Approach.

C. Potential Value Calculation

In the potential field method each location in the world is assigned a potential value. In the approach presented here, the potential value follows the calculation of the harmonic function and represents the probability of the robot colliding with an obstacle prior to reaching the goal under the assumption that it performs a random walk from this position. To achieve this, the potential value of a particular location at time t is given by the average of the potential values of that location's successors at time t+1, as shown in Equation (2).

$$p(x, y, t) = \frac{1}{|S(x,y)|}\left[\sum_{s\in S(x,y)} p(s, t+1)\right] \tag{2}$$

The set S is the set of the successors of a particular state i.e. the locations that can be reached in the next time step. For a robot moving at a fixed velocity they are the locations that it can move to in the next time step. The current location itself will become a neighbor in the next time step as the option of just staying at the current location without moving is open to the robot. Thus the set of neighbors or successors S of a location x, y can be defined as the set of the locations that the robot can reach by moving from the current state at a predetermined velocity and the current state itself.

We need the values of a location's future neighbors before we can calculate the current probability. This implies that the potential values have to be propagated from the future backwards. As a result a maximum duration or look-ahead value has to be selected in order to facilitate the potential function calculation. For the entire world, we work backwards from the look-ahead time step, calculating the potential value for each location in the world from the values calculated in the previous time step until the initial step is reached.

The selection of the look-ahead value is an important step as without an accurate look-ahead the planner may settle for a sub-optimal solution or not find a solution at all.

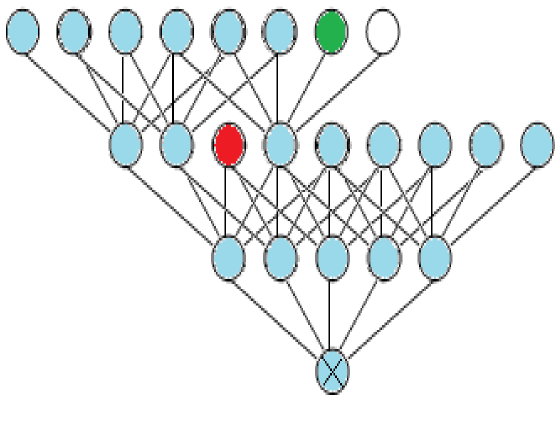

Figure 2. Forward propogating to find 1st reachable goal

D. Look-ahead value determination

The initial look-ahead value is determined by checking graph connectivity where the world is considered to be a directed acyclic graph. This is done by starting at the initial location and moving ahead one time step at a time to the current location's neighbors. If any of the neighbors is a goal then the number to steps it took to reach it is noted. If the neighbor is a goal or a node that has already been visited then there is no need to expand it. Otherwise the neighbors of the node are expanded next. The nodes are expanded in a manner similar to a breadth first search [20]. Only after all the nodes in a current time step are processed are its descendents processed. This ensures that the first possible solution is obtained. This process is illustrated in Fig. 2. The blue circles are the explored points in the space-time array, the white are the unexplored, the red circles are obstacles and the green are goals.

Using the minimum solution depth that we have obtained we determine the potential of the start state by the method described in Section III-B. This potential value is then compared to the expected potential value curve derived for a set of randomly generated environments with the same obstacle density as the current world. From this we find the look-ahead value, la_1, where the current potential value would be obtained in a random world. We also find the look-ahead value, la_2, where the likelihood of obtaining a lower potential value drops below a certain threshold. This threshold value can be a fixed value for the entire curve or one that increases as the look-ahead increases.

We then calculate the look-ahead offsets α and β which are given by Equation (3) and Equation (4)

$$\alpha = la_2 - la_0 \tag{3}$$

$$\beta = la_2 - la_1 \tag{4}$$

Here la_0 is the number of steps after which the first solution was found by forward chaining. The new look-ahead is simply the minimum solution depth incremented by α or β, whichever is greater. An example for la_0, la_1, la_2 is shown in Fig. 3.

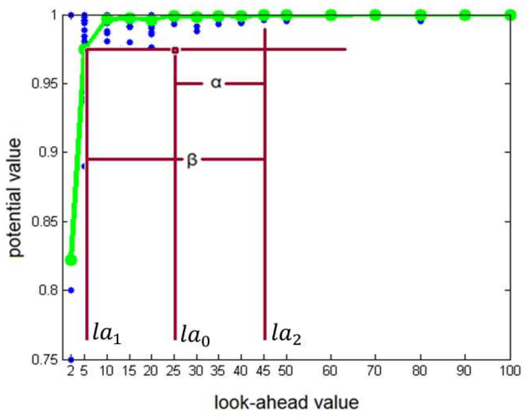

Figure 3. Finding Look-ahead offsets.

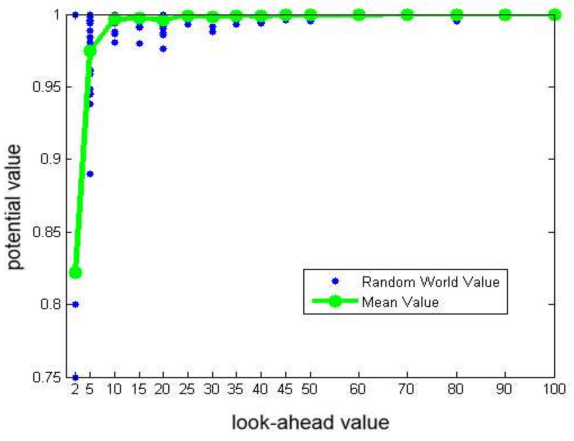

Figure 4. Potential value curve for a set of randomly

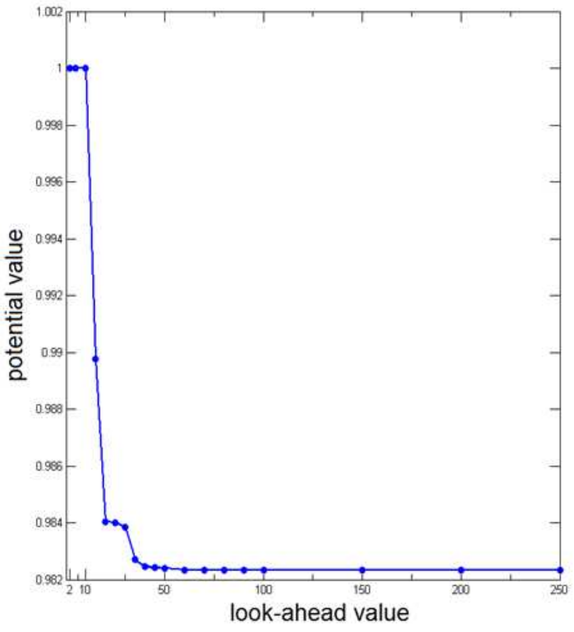

Figure 5. Potential values of start state in given world

E. *Generating Potential Value curves*

The potential value curves have to be generated before the planner can be used. These curves have to only be generated once and can then be used for all runs of the planner, provided that the overall size and characteristics of the world do not change. Given an average obstacle density, a set of random worlds is generated by creating random obstacles until the required obstacle density is obtained. For each of these worlds, and each look-ahead value, one of the non obstacle locations in the final time step before the look-ahead is converted into a goal and the potential values of the world are calculated. One of the initial states from which the goal is reachable is chosen as the start state and its potential value is noted. The process is repeated for a number of randomly generated worlds and for various look-ahead values. The mean values for the potential values and the corresponding standard deviations for each look-ahead value are calculated and the mean values are joined by a curve. These curves and corresponding standard deviation tables are generated for various obstacle densities and are used by the planner as needed. One such curve is shown is Fig. 4. The points represent the potential values of the random worlds. The line is the potential value curve joining their means.

IV. Implementation and Observations

The proposed method was implemented as a MatLab package [21] using MatLab (v7.5.0 R2007b). It also used the Data Structures and Algorithms toolbox [22] and the MatLab pointer library [23]. To study the effect of look-ahead values and obstacle densities and to illustrate the performance of the planning approach, various experiments were performed.

A. *Effect of Look-Ahead on Potential Value*

Consider the potential value of a start state in a given world. The world used here is a known world where the solution occurs at time step 12. As can be seen from Fig. 5, the potential value is 1 until the look-ahead time look-ahead time reaches the minimum solution depth. If we had proceeded with a smaller look-ahead, the planner would have ended in a failure. As the look ahead increases, the potential value of the start state decreases until it reaches a minimum and it does not reduce any further after that value. This emphasizes the importance of selecting the correct look-ahead value since if too high a look ahead is selected, the algorithm may perform a lot of unnecessary calculations.

B. *Effect of Time to Reach Goal on Potential Value*

If we look at the potential value of the start state of a randomly generated world for various look-ahead limits when the goal state is set just at the look-ahead value, we can see that the potential of the start state increases as the distance to reach the first goal increases. Consider the curve given in Fig. 4, which is the curve obtained by averaging the potential values obtained for a number of such worlds. The longer the robot navigates through the world without reaching a goal, the more likely that it would hit an obstacle

C. *Effect of World Size on Potential Value*

The potential value curves generated by worlds of different sizes and a fixed obstacle density (in this case 9%) are shown in Fig. 6. Since the change is greatest in the lower ranges of the look-ahead value, the x-axis (look-ahead value) is logarithmic to improve clarity. Although the curves are generated separately for each different world size, the curves are remarkably similar.

D. *Effect of World Density on Potential Value*

The potential value curves for random worlds of a fixed size and different densities are given in Fig. 7. The potential values tend to get closer to 1 quicker i.e. at lower look-ahead value as the obstacle density increases. As the number of obstacles increases so too does the probability of a robot colliding with it during a random walk.

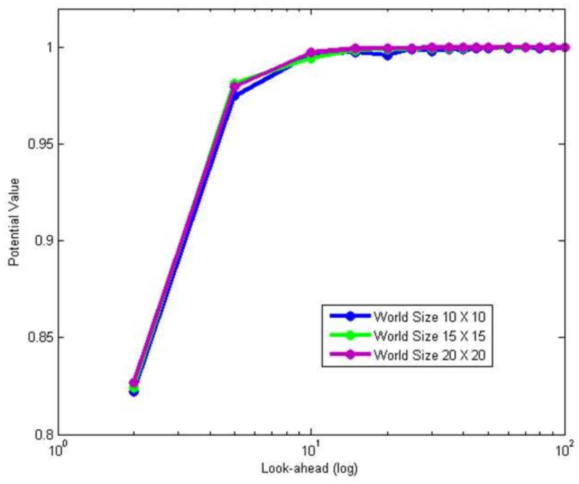

Figure 6. Potential Curves of Random worlds of varying sizes

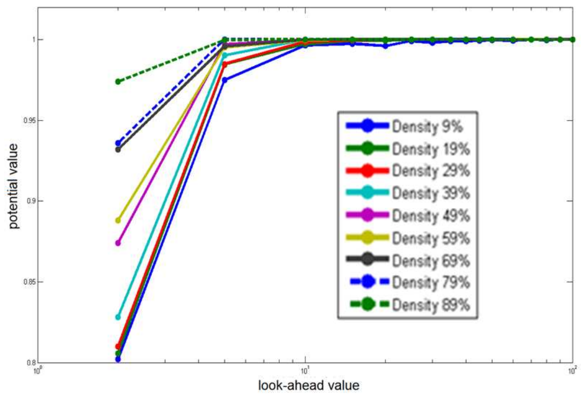

Figure 7. Potential curves of random worlds of varying obstacle density

E. *Sample Execution of the Proposed planner*

In Fig. 8, we can see a simple world created for the purpose of illustrating our algorithm. The arrows at the obstacles and goal show the direction of motion. When running the planner the first step is to check if the set of potential value curves already exists for the given world's size and density. If they do not exist, then the curves are generated for the world's parameters. The next step is to use forward chaining to find the time required to reach the first goal. After obtaining the number of steps required to reach the first solution, the value is compared to the potential value curve of the closest density lower than the world's density and the correct look-ahead is determined. Using this value, the planner builds a three dimensional grid map of the world and extrapolates the goal and obstacle locations up to this look-ahead value. Any position where a goal and obstacle may intersect is also considered as an obstacle. On this grid the potential values for each of the cells are calculated. After calculating the potential value, the planner follows the negative gradient in increasing time, to find the path. The path obtained for the example world is given in Figure 9. The green line shows the path of the goal and the blue line shows the path of the robot.

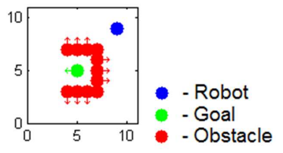

Figure 8. Sample world used

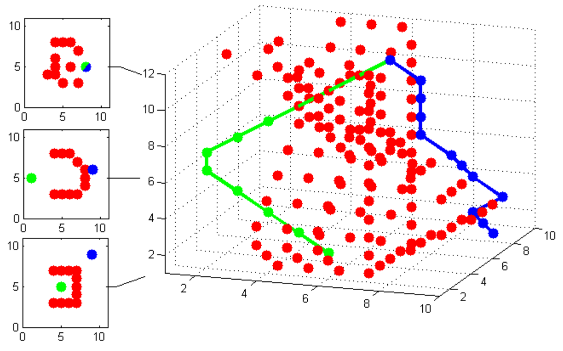

Figure 9. Path traversed by Robot (blue) and Goal (green).

F. *Avoidance of Suboptimal Goals*

The traversal algorithm discussed previously should naturally avoid suboptimal solutions and pick the most optimal solution within the given look-ahead limit. If a reachable goal exists, it will plan a path to it. Here optimality is determined as the minimum collision probability in the case of a random walk. This can be illustrated in a sample world which has been generated especially for this purpose and which contains two possible goals at different times. If the look-ahead is chosen too short, as shown in Fig. 10, then only the first goal, which is surrounded by obstacles, is available. In this case the planner has no other choice but to plan a path to it. But if the proposed method to pick an appropriate look-ahead is used, the second, safer solution is also available as shown in Fig. 11, since the extremely high collision probability of the initial, shortest solution leads to a significant expansion of the look-ahead time until the likelihood of finding a better solution drops below the significance threshold. As a consequence, the planner plans a path to it even though it is technically a longer path.

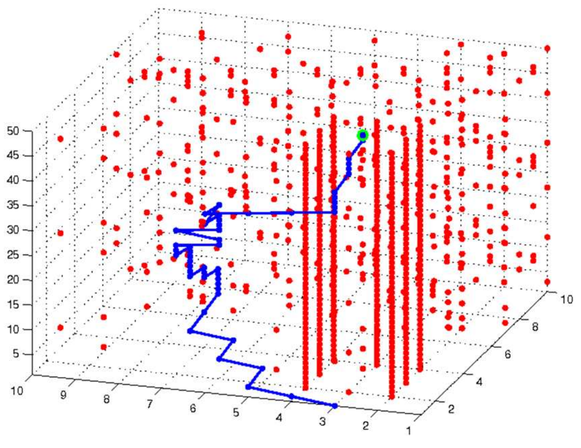

Figure 10. Planner picking only available solution

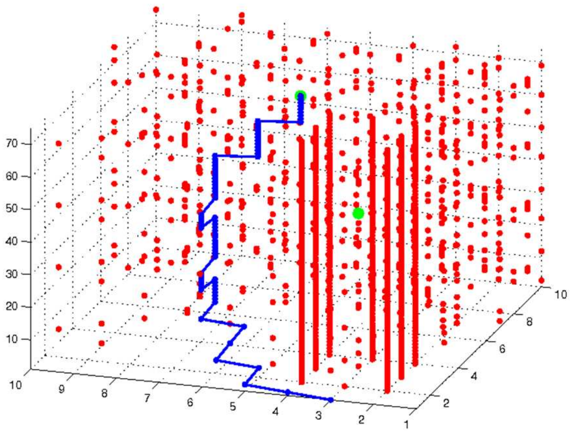

Figure 11. Planner planning path to a safer goal

V. Conclusions and Future Work

This paper introduced a new method for path planning in dynamic environments. Consider for example someone attempting to catch a falling ball, the person automatically calculates where the ball will be when he can catch it and moves his hands to that position instead of trying to move his hands towards the ball. This is in essence the principle behind the proposed approach. The path planner looks ahead by a predetermined amount and calculates the potential values of the world locations from the initial time up to the look-ahead time. The robot then follows the negative potential gradient with respect to time to reach the goal. Since we use forward chaining to find the depth to the first goal we are guaranteed a solution. Also, the usage of potential value curves gives us a good chance of finding a good solution. Since the potential values represent the probability that a robot executing a random walk from that position at that time will collide with an obstacle, following the path of negative potential gradient will mean that the robot will follow the safest path to the goal. Harmonic Functions were also used as the basis for the path planning algorithm because they generate smooth, complete and correct paths. Since the world has been stretched in the time dimension, the multiple iterations of the numerical evaluation of a harmonic function are executed in one pass over the three-dimensional world array.

In the proposed method, if the dynamics of the environment change during runtime, there is no option other than to re-plan from that point on. It remains to be seen if the potential values can be updated on the fly. This could greatly reduce the re-planning time.

References

[1] J.-C. Latombe, Robot Motion Planning, Norwell, MA: Kluwer Academic, 1991.

[2] S. S. Ge, Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," Autonomus Robots, vol. 13, pp. 207–222, 2002.

[3] A. Poty, P. Melchior, A. Oustaloup, "Dynamic Path Planning for mobile robots using fractional potential field," First International Symposium on Control, Communications and Signal Processing, pp. 557-561, 2004.

[4] W. Wu, Z. QiSen, J. B. Mbede, H. Xinhan, "Research on Path Planning for Mobile Robot among Dynamic Obstacles," Joint 9th IFSA World Congress and 20th NAFIPS International Conference. vol. 2, pp. 763–767, 2001.

[5] J. van den Berg, D. Ferguson, J. Kuffner, "Anytime Path Planning and Replanning in Dynamic Environments," Proceedings of the 2006 IEEE International Conference on Robotics and Automation Orlando, Florida, pp. 2366-2371, 2006.

[6] Y. Kitamura, T. Tanaka, F. Kishino, M. Yachida, "3-D path planning in a dynamic environment using an octree and an artificial potential field," IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 2474, 1995.

[7] F. M. Marchese, M. D. Negro, "Path-Planning for Multiple Generic-Shaped Mobile Robots with MCA," ICCS 2006, Part III, Lecture Notes in Computer Science, vol. 3993, pp. 264–271, 2006

[8] C. W. Warren, "Multiple robot path coordination using artificial potential fields," Proceedings of the 1990 IEEE conference on Robotics and Automation, vol. 1, pp. 500–505, 1990.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile tools," International Journal of Robotics Research, vol. 5, no.1, pp. 90-98, 1986.

[10] G. Faria, R. Romero, E. Prestes, M. Idiart, "Comparing harmonic functions and potential fields in the trajectory control of mobile robots," Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics, vol. 2, pp. 762–767, 2004.

[11] J. Borenstein, Y. Koren, S. Member, "The vector field histogram - fast obstacle avoidance for mobile robots," IEEE Journal of Robotics and Automation, vol. 7, pp. 278–288, 1991.

[12] M. G. Park, M. C. LEE, "A new technique to escape local minimum in artificial potential field based path planning," Korean Society of Mechanical Engineers International Journal, vol. 17, n. 12, pp. 1876–1885, 2003.

[13] S. S. Ge, Y. J. Cui, "New Potential Functions for Mobile Robot Path Planning. IEEE Transactions on Robotics and Automation," vol. 16, no. 5, pp. 615–620, 2000.

[14] P.-Y. Zhang, T.-S. Lü, L.-B. Song, "Soccer robot path planning based on the artificial potential field approach with simulated annealing," Robotica, vol. 22, pp. 563–566, 2004.

[15] E. W. Weisstein, "Harmonic Function," MathWorld – A Wolfram Web Resource, http://mathworld.wolfram.com/HarmonicFunction.html.

[16] E. W. Weisstein, "Laplace's Equation," MathWorld – A Wolfram Web Resource, http://mathworld.wolfram.com/LaplacesEquation.html.

[17] C. I. Connolly, R. A. Grupen, "On the Applications of Harmonic Functions to Robotics," Journal of Robotic Systems, vol. 10, pp. 931–946, 1993.

[18] S. M. LaValle, Planning Algorithms, New York, NY: Cambridge University Press, 2006

[19] G. Faria, R. Romero, E. Prestes, M. Idiar, "Comparing harmonic functions and potential fields in the trajectory control of mobile robots," Proceeding of the 2004 IEEE Conference on Robotics, Automation and Mechatronics, vol. 2, pp. 762–767, 2004.

[20] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Englewood Cliffs, NJ: Prentice Hall, 1995.

[21] V. Gopikrishna, "Temporal Potential Function based Path Planner for Dynamic Environments," http://www.mathworks.com/matlabcentral/fileexchange/22346, 2008.

[22] Y. Keren, "Data Structures & Algorithms Toolbox," http://www.mathworks.com/matlabcentral/fileexchange/212, 2001.

[23] N. Zolotykh, "MATLAB Pointer Library," http://code.google.com/p/pointer/, 2007.