Potential and Sampling Based RRT Star for Real-Time Dynamic Motion Planning Accounting for Momentum in Cost Function

Saurabh Agarwal, Ashish Kumar Gaurav, Mehul Kumar Nirala, and Sayan Sinha(✉)

Indian Institute of Technology Kharagpur, Kharagpur, India
{shourabhagarwal,ashishkg0022,mehulkumarnirala,sayan.sinha}@iitkgp.ac.in

Abstract. Path planning is an extremely important step in every robotics related activity today. In this paper, we present an approach to a real-time path planner which makes use of concepts from the random sampling of the Rapidly-exploring random tree and potential fields. It revises the cost function to incorporate the dynamics of the obstacles in the environment. Not only the path generated is significantly different but also it is much more optimal and rigid to breakdowns and features faster replanning. This variant of the Real-Time RRT* incorporates artificial potential field with a revised cost function.

Keywords: Path planning · Robotics · RRT · Potential energy Wavefront · Doppler effect

1 Introduction

Robotics challenges today involve the rapid motion of robots in a highly dynamic environment where agents try to outperform their opponents with superior planning and strategy. At the lowest level of operation, path planning has a major contribution to the performance of an agent. In this paper, we describe the steps and methods involved in the making of a dynamic path planner suited for such challenging environments. We take into consideration the real-time knowledge of the two-dimensional coordinate representing the position of various robots. We also assume that the data available is correct and do not question its accuracy.

Random sampling path planners have been highly used in various fields of robotics, such as aerial robotics [6] and robosoccer [1]. Many variants of Rapidly-exploring Random Trees (RRTs) have been used in such areas. Along with that, path planning based on potential fields has also been popular in robotics and are pretty commonly used in areas such as self-driving cars [3]. In this paper, we describe the use of a variant of the RRT algorithm and discuss incorporation of the concept of artificial potential field (APF) into it.

S. Agarwal, A.K. Gaurav, M.K. Nirala and S. Sinha—Equal contribution.

L. Cheng et al. (Eds.): ICONIP 2018, LNCS 11307, pp. 209–221, 2018.
https://doi.org/10.1007/978-3-030-04239-4_19

2 Previous Works

Extensive research has been done on the path-planning problem in the past few years. Traditional approaches fail to provide estimates before the motion begins, and hence, it becomes difficult to provide proper control on motion of agent. Moreover, their instantaneous nature is prone to making the velocity graph abstract. This makes the motion of the robot unstable, bringing in more error into the scenario, leading to the violation of kinodynamic constraints. Such planners include MergeSCurve [10] and Dynamic Window [4]. Visibility Graphs [7] is another widely used algorithm for obstacle avoidance. But, it generates paths that are very close to the obstacles, and hence is not suitable for a dynamic environment. Other models such as Graph Plan [2] prefer concentrating on fields of robotics with a large number of degrees of freedom, and is difficult to be developed into generic algorithms for path planning.

Path planning algorithm for the dynamic environment needs to be real-time and robust. The paths need to be regenerated or updated from time to time based on various factors. The RRT too is not suitable for dynamic environments. As soon as a new path needs to be generated, a tree growth is performed from the source to the destination, which can be pretty costly. However, various variants of the RRT have been proposed to make them suitable according to specific needs. The variant of RRT suitable in our case is the RT-RRT* [9]. Similarly, the concept of a potential field for path planning was initially applied for static situations only. Though later on, various research work has been performed to introduce dynamism into Artificial Potential Field (APF) [15] algorithms and create ameliorative APF models [11] making them suitable for a rapidly changing environment. In this section, we first discuss the previously proposed algorithms of the RRT. Several graph and potential based algorithms have been discussed.

2.1 Rapidly-Exploring Random Tree

Rapidly-exploring Random Tree (RRT) [8] finds path between $x_{init} \subset X$ and $x_{goal} \subset X$. It is assumed that an obstacle region $X_{obs} \subset X$ is given and we can check if a point lies in this region or not. If any point or edge lies in this region, that should not be used to extend our tree to avoid collision with any obstacle. It is a simple and fast algorithm for finding a path between two points, but it has several limitations. It does not guarantee convergence to an optimal path. Furthermore, no measure is taken to make this algorithm compatible with the dynamic environment. Later, various extensions have been proposed to address these problems. Below is a brief description of RRT and Algorithm 1 presents its pseudo code. In this algorithm, $SampleFree()$ samples a node $x_{rand} \subset X$ randomly in space. Then $x_{nearest}$ is found using $Nearest(V, x)$ which finds a node nearest to x among a list of nodes V. After this $Steer(x_t, x_o)$ is used to give point on path which originates from x_{rand} and terminates to $x_{nearest}$. If $(x_{nearest}, x_{node})$, a path joining $x_{nearest}$ and x_{node}, belong to X_{free} then x_{node} is inserted into V and similarly $(x_{nearest}, x_{node})$, edge joining $x_{nearest}$ and x_{node}, is inserted in E.

Algorithm 1. RRT algorithm

```
1: procedure RRT(x_init)                                  ▷ The source node
2:     V ← {x_init}
3:     E ← φ
4:     for i = 1, 2, ..., n do
5:         x_rand ← SampleFree(i)
6:         x_nearest ← Nearest(V, x_rand)
7:         x_node ← Steer(x_nearest, x_rand)
8:         if ObstacleFree(x_nearest, x_node) then
9:             V ← V ∪ {x_node}
10:            E ← E ∪ {(x_nearest, x_node)}
11:    return G = (V, E)
```

2.2 RRT*

RRT* [5] algorithm proposes a way which tries to minimise the distance of the root from the nodes at each new iteration. After the random sampling of a new node, this algorithm allows rewiring of the tree to reduce the distance from root to child node. It inspects each of the nodes of the tree which are within a neighbourhood of the newly generated child node. This child node is then reconnected to that node, which traces up to the tree root at the shortest distance.

2.3 A Real-Time Path Planning Algorithm Based on RRT*

RT-RRT* [9] algorithm is similar to RRT*, but the tree growth takes place just once, and the nodes are rewired from time to time based on a variety of factors. Since the tree growth does not take place again and again, RT-RRT* is suitable for a dynamic environment as the overhead costs are low. Moreover, rewiring of nodes makes sure the tree has been modified to suit the changes in the environment.

2.4 Potential Guided Directional-RRT*

Potential guided directional-RRT*(PDG-RRT*) [12] is a modification of RRT* which uses Artificial Potential Field [15] for guiding the random sampling more towards the goal. This enhances the rate of convergence and provides a more optimal solution. At each iteration, a random node x_{rand} is generated. This node x_{rand} is then moved a fixed distance α along the direction of the potential gradient to give a new node z. This node z is then added to the tree by adding an edge from this node to another node in the tree such that the distance from the source to z is minimised, and we define $nodes_{max}$ is the maximum number of nodes.

Algorithm 2. Potential Guided Directional-RRT* algorithm

procedure POTENTIAL GUIDED DIRECTIONAL-RRT*(x_{init}) ▷ The source node
  $V \leftarrow \{x_{init}\}$
  $E \leftarrow \phi$
  for $i = 1, 2, 3 \ldots nodes_{max}$ **do**
    $x_{rand} \leftarrow SampleFree(i)$
    $x_{prand} \leftarrow GradientDescent(x_{rand})$
    $x_{nearest} \leftarrow Nearest(V, x_{prand})$
    $x_{node} \leftarrow Steer(x_{nearest}, x_{prand})$
    if ObstacleFree($x_{nearest}$, x_{node}) **then**
      $V \leftarrow V \cup \{x_{node}\}$
      $E \leftarrow E \cup \{(x_{nearest}, x_{new})\}$
  return $G = (V, E)$

3 Proposed Path Planner

The previous works describe various ways in which the RRTs have been modified to improve the optimality, speed and dynamism of path generation. The PGD-RRT* [12] promises more optimal paths, and we attempt to extend it to the dynamic environment. The RT-RRT* is a variant of the RRT* which features dynamism. In this paper, the RT-RRT* has been made more optimal, using concepts from PGD-RRT*. Along with that, some novel concepts have been introduced to bridge the gap, such as a new formula for computing potential and a new cost function.

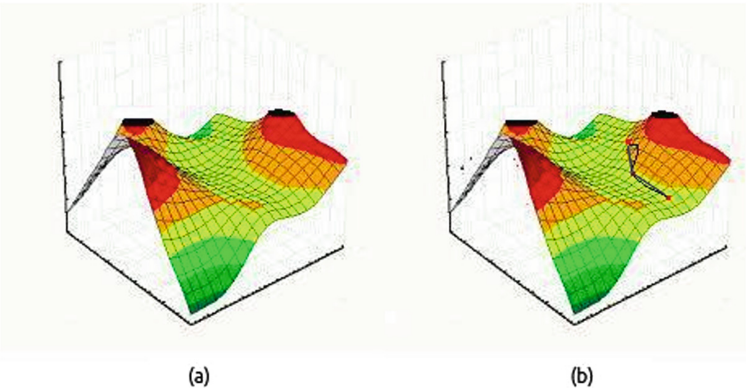

Fig. 1. (a) The graph of superimposition of potential and frequency. (b) Straight line path is taken as cost in other planners marked in grey; actual path taken by our planner marked in black. The black patches mark infinity.

3.1 Tree Growth

The RT-RRT* [9] proposes to grow the tree just once during the lifetime of the path planner, and reuses the nodes and the edges to predict the path. We use a similar concept here, with the difference being in the fact that the random sampling is influenced by the potential field created.

Potential Field: The potential field generated can be compared to an electrostatic field. The obstacles are assumed to be positively charged, and the destination to be negatively charged. This places the obstacles at a relatively higher positive potential. The aim is to descend towards lower potential as much as possible. All charges are taken as point charges. At every point x in the space, the potential is calculated for a particular charge as $V_i = \frac{kQ_i}{r_i}$ where Q_i is the magnitude of the i^{th} point charge (obstacle or destination), and r_i is the distance of the point x from it. The net potential at a point is given by $\sum_{i=1}^{n} V_i = k \sum_{i=1}^{n} \frac{Q_i}{r_i}$. Here k is a proportionality constant, and n is the total number of charges. We determine the values of Q and k by validation.

The sampling method used in the RT-RRT* [9] to generate new nodes is applicable here as well. Briefly, a random number between 0 and 1 is generated, and on the basis of that, it is determined where random sampling is to be performed. $LineTo(x_{goal})$ is invoked when the random number generated is greater than $1 - \alpha$, else if the number is less than $\frac{1-\alpha}{\beta}$ $Uniform(X)$ is invoked. Otherwise $Ellipsis(x_0, x_{goal})$ is called. Here β is a real number used for making a distinction between the last two function calls. Next, taking inspiration from PGD-RRT* [12], the randomly generated point is allowed to descend according to the potential gradient for a certain amount of time. The resulting point is treated as the initially generated node for an RRT*. The best parent to this node is determined within the surrounding grids of the node. From that parent, at a particular step length, the new node is created. The entire procedure, in a nutshell, is provided in the Algorithm 3.

3.2 Hexagonal Grid

The entire area is divided into a set of grids, and all the nodes are mapped to a specific grid. In another way, every node is assigned a particular grid ID, which is developed in such a manner that the entire area gets divided into a set of hexagonal grids. Hexagon ensures most efficient packing in 2D space and is quite popular in path planning domain [13]. The potentials developed correspond to those of the points at the centres of the respective grids.

3.3 Edge Rewiring

The RT-RRT* proposes to rewire edges under three categories. We explain the changes we make to each of them:

Algorithm 3. Random sampling

procedure RANDOM SAMPLING(U, G, α, β, γ, $step\_size$) ▷ Graph G is the existing graph, Field U is the generated field
  $A \leftarrow rand(0, 1)$ ▷ Random number generation between 0 and 1.
  if $A \geq 1 - \alpha$ **then**
    $x_{rand} = LineTo(x_{goal})$
  else
    if $\frac{1-\alpha}{\beta}$ **then**
      $x_{rand} = Uniform(X)$
    else
      $x_{rand} = Ellipsis(x_0, x_{goal})$
  $x_{parent} = BestParent(x_{rand})$ ▷ Best parent is decided as per the recursive cost function
  $x_{random} = NodeAtDist(Angle(x_{parent}, x_{rand}), step\_size)$ ▷ Finding the node on the line joining x_{parent} and x_{rand} at a distance $step\_size$ from x_{parent}
  $x_{final} = x_{random} - \gamma \frac{\delta(U)}{\delta(x)}|_{x=x_{final}}$
  $U = AddToGraph(U, x_{final})$
  if Graph has been saturated **then**
    return
  else
    $RANDOMSAMPLING(U, G, \alpha, \beta, \gamma, \_size)$

Rewire in the Presence of an Obstacle: When a path is being approached by an obstacle, a rewiring is necessary. According to the RT-RRT* algorithm, any path which goes through the obstacles attain infinite cost. Similarly, when obstacles have charge, they provide an infinite potential at their point of existence. Hence, both of them seamlessly integrate with one another. But, here, the infinite potential is assumed for a greater radius, as deemed fit for the situation. Accordingly, rewiring is prioritised in such regions.

Rewire from Root: Rewiring begins at the root and keeps proceeding throughout the tree. Hence, it starts rewiring from the position where the agent lies. Then, it continues to assign nodes which are at greater distance from the agent to their corresponding best-matching parents.

Random Rewiring: This paper proposes to remove random rewiring from RT-RRT* and introduces a way of determining edges which are to be rewired. At every point of the field, these wavefronts would be perceivable. In case the robot is moving, the frequency detected at a point would be different from that of what had been emitted. This is known as the Doppler Effect. The frequency due to Doppler effect is expressed as $f = \left(\frac{v+v_r}{v+v_s}\right) f_0$ where f is the perceived frequency, f_0 is the constant frequency emitted by obstacles, v is the constant velocity of wave in space, v_r is receiver's velocity and v_s is the source's velocity which will be equal to the obstacle's velocity in our case. The frequency at every point

in the field is calculated using $\sin(\omega_1 t) + \sin(\omega_2 t) = 2\sin\left(\frac{\omega_1+\omega_2}{2}t\right)\cos\left(\frac{\omega_1-\omega_2}{2}t\right)$ in a binary fashion. This has been illustrated in Fig. 2(a). Thus, we obtain the net frequency of every point in the field. After plotting this across the field, we obtain the local minima. Finding out the maximum among these local minima, we rewire edges randomly in that grids.

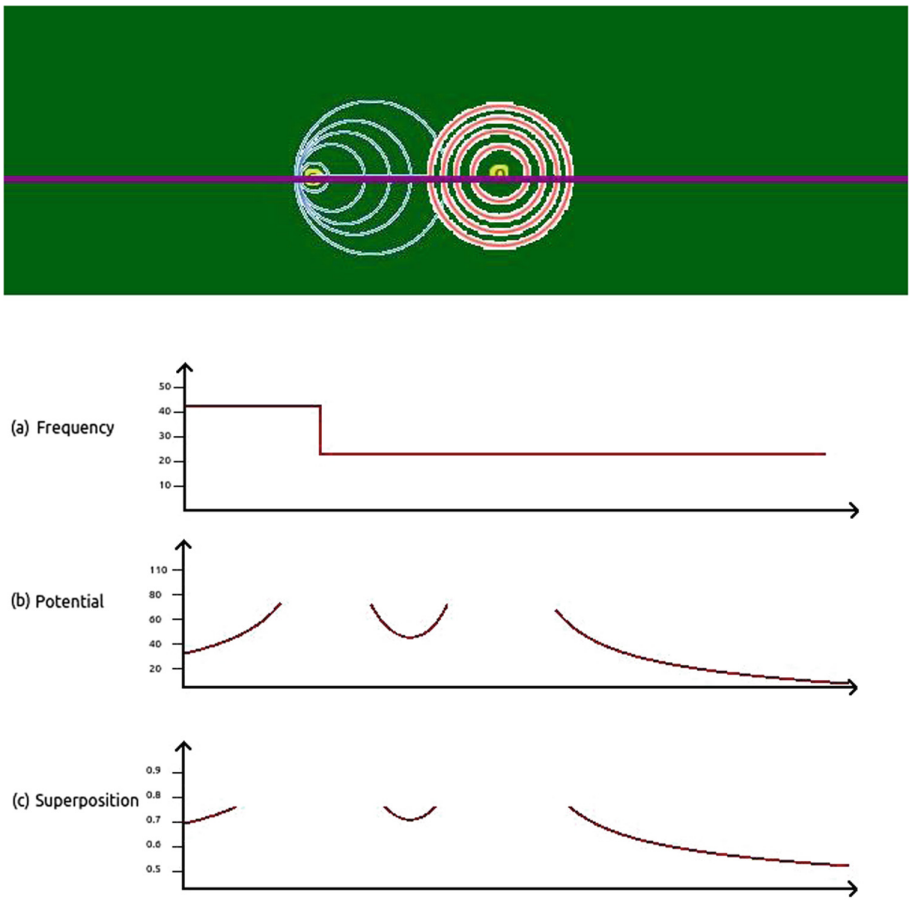

Fig. 2. Wavefronts due to robot while stationary and while moving. This is described mathematically by the Doppler Effect. Along the violet line, the (a) frequency graph, (b) the potential graph and (c) the overall graph have been shown (Color figure online)

3.4 Boundary Conditions

An exponential potential control barrier function is defined in 3D space. Let the boundary of the field/environment be defined by $\mathbf{B(x)}$ and let $\mathbf{x_b}$ be a point on the boundary. The growth of the potential function around boundary is defined

by $B(x) = \frac{1}{1-e^{-||x_b-x||}}$. Thus, the potential field around the boundary of the environment proliferates as,

$$\lim_{x \to x_b} \frac{1}{1 - e^{-||x_b - x||}} \to \infty$$

$||x||$ represents $x^T.x$. A moving body incident on the barrier gets reflected, and hence would restrict path generation.

4 Cost Function

The cost function in the RT-RRT* [9] algorithm is used to find the best parent of a node while performing edge-rewiring. Edge rewiring, as described in previous sections, helps in optimising an already generated path, by providing nodes with better parents. The RT-RRT* defines cost recursively as:

$$cost_{distance}(x_{new}) = cost(x_{closest}) + dist(x_{closest}, x_{new}) \tag{1}$$

Where x_{new} is the node for which we wish to compute the cost, $x_{closest}$ is the node closest to x_{new} and $dist(x_{closest}, x_{new})$ is the distance between x_{new} and $x_{closest}$. This paper proposes to make some changes to the existing cost function, making PGD-RRT* suitable for a dynamic environment. The recursive approach to the cost function remains the same. The difference lies in the way the distance (using $dist$) is computed. Hence $x_{closest}$ shall be modified as well. Nevertheless, $x_{closest}$ remains the closest node via the distance metric, but in this paper, we attempt to replace the commonly used Euclidean distance with geodesic distance. The two-dimensional graph of the area initially developed using an artificial potential field is taken into consideration. The straight line path which gives the Euclidean distance between x_{new} and $x_{closest}$ is projected on this graph. The length of the curve thus obtained is taken as the distance. Thus, the formula remains the same, but the approach to computing the distance is a bit different. It is explained in a better fashion through the illustration provided (Fig. 1).

Let the potential surface be represented as $\boldsymbol{S}$ and the position vectors of the points $x_{closest}$ and x_{new} be $\boldsymbol{a}$ and $\boldsymbol{b}$. The distance is measured along the curve formed by intersection of $\boldsymbol{S}$ with a plane which is parallel to the z axis and passes through the straight line path between $\boldsymbol{a}$ and $\boldsymbol{b}$. We assign $\hat{n}$ to be the cross product between $\boldsymbol{a} - \boldsymbol{b}$ and a unit vector along the z axis. We obtain a plane $\boldsymbol{P}$ as $(\boldsymbol{r} - \boldsymbol{a}) \cdot \hat{n} = 0$, where $\boldsymbol{r}$ is an arbitrary point on the plane. Thus, the curve formed along the path between two nodes becomes $\boldsymbol{C}$: $\boldsymbol{S} - \boldsymbol{P} = \boldsymbol{0}$. $dist(x_{closest}, x_{new})$ is the arclength along $\boldsymbol{C}$ from $\boldsymbol{a}$ to $\boldsymbol{b}$. If we represent $\boldsymbol{C}$ parametrically as $\boldsymbol{p}(t) = < x(t), y(t), z(t) >$ with t_1 and t_2 being the parametric values for $\boldsymbol{a}$ and $\boldsymbol{b}$, then the arclength is given by

$$dist(x_{closest}, x_{new}) = \int_{t_1}^{t_2} \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2 + \left(\frac{dz}{dt}\right)^2}\, dt$$

Doppler Effect: The cost function incorporates the velocity using the Doppler effect. This accounts for the dynamic nature of obstacles. The shift in frequency helps in the prediction of approach/separation of the body concerning obstacles. The frequency at every point in the field is calculated by superimposition of waves from all the obstacles, as illustrated in Fig. 2. This shift in frequency is plotted on a two dimensional axis for all obstacles. The obtained map explains the relative velocity of approach/separation of obstacles with respect to the moving body.

$$\mathbf{F}(\hat{\mathbf{x}}, t) = \sum_{i=0}^{|obstacles|} f_i \tag{2}$$

where, $|obstacles|$ represents the cardinality of the obstacle vector, f_i denotes the observed frequency of $\mathbf{i}_{th}$ obstacle and t is time. The surface so obtained corresponds to obstacles' activity. The minima on the surface correspond to low activity regions of obstacles. To extract the local minima the surface is descended iteratively. $\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0$. where $\mathbf{x_n}$ are points in Cartesian plane with $\mathbf{x_0}$ as initial guess. To ensure convergence step size γ is kept small. However an effective γ via Barzilai-Borwein method [14] yields promising results. The Barzilai-Borwein method is shown below

$$\gamma_n = \frac{(\mathbf{x}_n - \mathbf{x}_{n-1})^T [\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})]}{||\nabla F(\mathbf{x}_n) - \nabla F(\mathbf{x}_{n-1})||^2} \tag{3}$$

The cost is defined as

$$cost_{velocity}(x_{new}) = \lambda |\mathbf{F}(\hat{\mathbf{x}}, t)|_{at\ \mathbf{x}=x_{new}} \tag{4}$$

where λ is a constant close to unity. Thus, low activity regions would be associated with low costs. Effective or total cost is computed as a linear combination of given by,

$$cost(x) = c_0 cost_{distance}(\mathbf{x}) + c_1 cost_{velocity}(\mathbf{x}) \tag{5}$$

Where c_0 and c_1 are two constants which depend on the environment and can be tuned to achieve promising results. Incorporating boundary condition would change effective cost to

$$cost(x)_{effective} = cost(x)B(x)$$

5 Experimentation and Analysis

We conducted an experiment based on tree generation time given a set of obstacles. The experimental setup included ROS based communication nodes and a GUI based interactive platform for setting up the environment. This was performed on Linux 4.4.0-127-generic with 8 GiB memory and 2.3 GHz x 8 CPU. Thus, it can be seen that our planner takes nearly the same time to generate the

Algorithm 4. Rewiring random edges

procedure REWIRE RANDOM(U, G, D, $grid$, ϵ, $root$) ▷ Graph G is the existing graph, Field U is the generated field
  $F = U + \epsilon D$
  $grid_{min} = rand(1, length(grid))$
  $max\_min = grid_{min}$
  for each i; i in $length(grid)$ **do**
    $flag = False$
    if $F[i] < F$[each $neighbour$ of $grid[i]$] **then**
      if $max\_min < F[i]$ **then** $max\_min < F[i]$
  $cost_{org} = cost(Parent(max\_min), max\_min)$
  $node_{final} = max\_min$
  for each node i in current and neighbouring grids **do**
    $cost_{here} = cost(i, max\_min) + cost(i, root)$
    if $cost_{here} < cost_{org}$ **then**
      $cost_{org} = cost_{here}$
      $node_{final} = i$
  $max\_min.setParent = node_{final}$

tree given similar set of environmental conditions. Along with that, our planner provides a more optimised path. The average number of iterations required to find the path to each goal was 10.36 for our method and 18.43 for RT-RRT*.

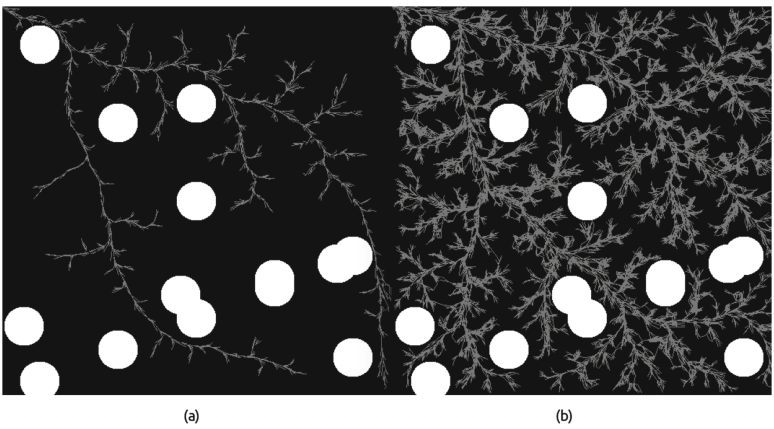

Fig. 3. The simulation environment, showing (a) our path planner and (b) RT-RRT*

Replanning: In a dynamic environment, the replanning of path is an important aspect. Moving obstacles when come along the path, a new path has to be found incorporating the new position of the obstacles. For better performance of agents, this replanning of path should take place in less amount of time.

Table 1. Comparison of time (in milli seconds) in generation of path

Obstacles	Average Traversal Time			
	PGD-RRT*	RRT*	RT-RRT*	Our planner
5	4.1996	3.7078	3.7951	2.7958
7	4.2315	3.7912	3.8105	2.8110
9	4.2555	3.9105	3.8492	2.8862
10	4.2658	3.9824	3.8908	2.8873
12	4.3655	4.1473	4.1335	3.1080
15	4.4662	4.2194	4.3264	3.1866
18	4.6905	4.4505	4.5155	3.4242
20	4.9696	4.9191	4.9583	3.9823

Our proposed path planner outperforms other planners in this, hence making it suitable for a real-time dynamic environment. With respect to the PGD-RRT*, our path planner is very quick in replanning as the tree growth does not take place again and again, but gets rewired when required. With respect to the RT-RRT*, our planner provides a more optimised path, which is clear from the illustration provided (Fig. 3) and Table 1. Our planner is based on a dynamic environment only and can be outperformed by other planners in a static environment (Table 2).

Table 2. Comparison of time (in milli seconds) in replanning of path

Obstacles	Average Replanning Time			
	PGD-RRT*	RRT*	RT-RRT*	Our planner
5	8.1996	8.7078	3.7958	3.7951
7	8.2758	8.8213	3.8501	3.8491
10	8.2783	8.8582	3.8614	3.8519
11	8.2811	8.8607	3.8688	3.8564
15	8.2902	8.9078	3.8891	3.8888
18	8.3215	8.9303	3.9011	3.8997
21	8.3414	8.9489	3.9114	3.9114

Our planner takes lesser tree generation time compared to the RT-RRT*, as it is biased in an intelligent way to reach the destination faster, using the concept of potential fields. Moreover, our planner features faster replanning as compared to PGD-RRT* as the tree growth does not take place again and again and is rewired. A similar advantage is seen over RRT* as well. The path generated is more optimised as compared to the RT-RRT* due to intelligent rewiring, which is evident from Table 1.

6 Conclusion

Our work focuses on a path planner with concepts from a variant of rapidly exploring random trees and artificial potential fields. They are merged innovatively, with concepts from various other domains, and a cost function redefined to suit a dynamic environment. This remains efficient in time with respect to tree growth and requires a lesser number of iterations for path generation, compared to that of similar planners.

Future Work

In the future, we propose to research on how the acceleration of a robot can help in computing a more optimal path. Though the mathematics proposed in this paper can easily be extended to take acceleration into consideration, acceleration is not that easy to measure and a lot of noise comes in the way of the location data of the robots being received. Hence, our research aims to find an efficient solution to denoising the data post which we can take into consideration the acceleration, and if possible, other forms of kinetics.

Acknowledgement. We thank Manjunath Bhatt (manjunathbhat9920@iitkgp.ac.in), Rahul Kumar (vernwalrahul@iitkgp.ac.in) and Shubham Maddhashiya (shubhamsipah@iitkgp.ac.in) for assisting us in this project and supporting us as and when required.

References

1. Bhushan, M., Agarwal, S., Gaurav, A.K., Nirala, M.K., Sinha, S., et al.: KgpKubs 2018 team description paper. In: RoboCup 2018 (2018)
2. Blum, A.L., Furst, M.L.: Fast planning through planning graph analysis. Artif. Intell. **90**(1–2), 279–298 (1997)
3. Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J.: Practical search techniques in path planning for autonomous driving. In: Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08) (2008)
4. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Robot. Autom. Mag. **4**, 23–33 (1997)
5. Karaman, S., Frazzoli, E.: Incremental Sampling-based Algorithms for Optimal Motion Planning. Robotics: Science and Systems. arXiv preprint:1005.0416 (2010)
6. Kim, J., Ostrowski, J.P.: Motion planning of aerial robot using rapidly-exploring random trees with dynamic constraints. In: IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), vol. 2, pp. 2200–2205 (2003)
7. Kunigahalli, R., Russell, J.S.: Visibility graph approach to detailed path planning in CNC concrete placement. In: Proceedings of the 11th ISARC, pp. 141–147 (1994)
8. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Report No. TR 98–11. Computer Science Department, Iowa State University (1998)
9. Naderi, K., Rajamki, J., Hmlinen, P.: RT-RRT*: a real-time path planning algorithm based on RRT*. In: 8th ACM SIGGRAPH Conference on Motion in Games (MIG 2015), pp. 113–118 (2015)

10. Nguyen, K.D., Ng, T.C., Chen, I.M.: On algorithms for planning S-curve motion profiles. Int. J. Adv. Robot. Syst. **5**(1), 99–106 (2008)
11. Qixin, C., Yanwen, H., Jingliang, Z.: An evolutionary artificial potential field algorithm for dynamic path planning of mobile robot. In: International Conference on Intelligent Robots and Systems, pp. 3331–3336 (2006)
12. Qureshi, A.H., et al.: Potential guided directional-RRT* for accelerated motion planning in cluttered environments. In: IEEE International Conference on Mechatronics and Automation, Takamatsu, pp. 519–524 (2013)
13. Sinha, S., Nirala, M.K., Ghosh, S., Ghosh, S.K.: Hybrid path planner for efficient navigation in urban road networks through analysis of trajectory traces. In: 24th International Conference on Pattern Recognition (2018, in Press)
14. Tan, C., Ma, S., Dai, Y., Qian, Y.: Barzilai-Borwein step size for stochastic gradient descent. arXiv preprint:1605.04131 (2016)
15. Vadakkepat, P., Lee, T.H., Xin, L.: Application of evolutionary artificial potential field in robot soccer system. In: Joint 9th IFSA World Congress and 20th NAFIPS International Conference, vol. 5, pp. 2781–2785 (2008)