Proceeding of the IEEE
International Conference on Robotics and Biomimetics (ROBIO)
Shenzhen, China, December 2013

# Adaptive Potential Guided Directional-RRT*

Ahmed Hussain Qureshi, Saba Mumtaz, Khawaja Fahad Iqbal, Badar Ali, Yasar Ayaz, Faizan Ahmed, Mannan Saeed Muhammad, Osman Hasan, Whoi Yul Kim and Moonsoo Ra

*Abstract*— **The Rapidly Exploring Random Tree Star (RRT*) is an extension of the Rapidly Exploring Random Tree path finding algorithm. RRT* guarantees an optimal, collision free path solution but is limited by slow convergence rates and inefficient memory utilization. This paper presents APGD-RRT*, a variant of RRT* which utilizes Artificial Potential Fields to improve RRT* performance, providing relatively better convergence rates. Simulation results under different environments between the proposed APGD-RRT* and RRT* algorithms demonstrate this marked improvement under various test environments.**

***Index Terms - RRT*, Artificial Potential Fields, Fast Convergence Rate, Optimal Path, Directional Sampling and Path Planning***

## I. Introduction

Motion Planning is an important feature of autonomous robots. Whatever purpose a robot is designed for, its first task is to learn how to plan its movement in a given work space, avoiding possible obstacles and reaching desired destinations. Robots for planetary exploration, museum tour guides, search and rescue robots and robots in surgery are some examples of robotics applications that require motion planning [1], [2]. Moreover, motion planning is not just restricted to robotics applications anymore. Structural analysis in biology [3] and computer graphics [4] are new areas of research where motion planning algorithms can be of great assistance. Over the years, different techniques for motion planning have been researched, including neural networks , sampling based algorithms[5], [6], potential field methods [7] etc.

Rapidly Growing Random Trees are predominantly utilized in dealing with problems with differential constraints. Their effectiveness has lead to a number of improvements by biasing its sampling techniques [8], [9], [10] and to achieve asymptotic optimality [11]. One of these improvements is the recently proposed RRT* algorithm [11], which finds an initial path as fast as the RRT algorithm but goes on to refine it by continuing to randomly sample the configuration space and eventually giving an optimum path solution. However, the effectiveness of this solution is limited by large memory demands due to utilization of a large number of iterations and slow convergence rate.

A. H. Qureshi, S. Mumtaz, K. F. Iqbal, B. Ali, Y. Ayaz and F. Ahmed are with the RISE lab, School of Mechanical And Manufacturing Engineering, NUST, Islamabad, Pakistan. (10beeaqureshi, 10beesmumtaz)@seecs.edu.pk, fahadiqbal@smme.nust.edu.pk, badarali@live.com, yasar@smme.nust.edu.pk, faizan.ahmed@smme.nust.edu.pk

A. H. Qureshi, S. Mumtaz are also associated with the School of Electrical Engineering and Computer Sciences, NUST, Islamabad, Pakistan.

O. Hasan is associated with the School of Electrical Engineering and Computer Sciences, NUST, Islamabad, Pakistan. osman.hasan@seecs.edu.pk

M. S. Muhammad, W. Y. Kim and M. Ra are with the IE Lab., Department of Electronic Engineering,College of Engineering, Hanyang University, Seoul,Korea. (mannan, wykim, msna)@hanyang.ac.kr

This paper introduces the concept of potentially guided directionalized sampling in RRT* by integrating the Artificial Potential Field Algorithm (APF) [1] with the RRT* algorithm. APF is a reactive approach towards path planning, where robot trajectories are flexible and their construction based on the robot's interactions with the configuration space. APF algorithms are already being used in many practical motion planning applications [12]. As discussed further on, our proposed Adaptive Potential Guided Directional-RRT* (APGD-RRT*) uses Artificial Potential Fields to reduce the dispersion of random samples in the RRT* algorithm by focusing them in the direction of decreasing potential. Directionalized sampling reduces the algorithm's execution time by decreasing the number of iterations required by RRT* to reach optimum solution.

The rest of the paper is organized as follows. Section II outlines RRT* and Artificial Potential Fields implemented in our algorithm to modify RRT*. Section III explains in detail the operation of our Adaptive Potential Guided Directional-RRT*(APGD-RRT*) algorithm. Section IV covers simulation results of APGD-RRT* and RRT* in varying environments while Section V performs a comparative analysis of RRT* and APGD-RRT*. The paper concludes with Section VI briefly summarizing our work and future plans in this research domain.

## II. Related Work

### A. RRT*

RRT* [13] path finding algorithm is used to guide a robot from an initial start point, $X_{init} \in X$, to a goal point, $X_{goal} \in X$. This is done by using RRT* to identify a specific input $u : [0, T] \in U$ which produces an optimal path solution when fed to a dynamic environment characterized by $\dot{x} = f(x(t), u(t))$. Variable $t \in [0, T]$ denotes time taken in translating the robot from the start to goal point. The configuration space is represented as $X \in \mathbb{R}^d$ and $U \in \mathbb{R}^m$ is the input space. $X_{obs} \in X$ are obstacles in the configuration space while $X_{free} = \frac{X}{X_{obs}}$ is the obstacle free region.

Following are some of RRT*'s main processes: *Random Sampling:* Samples the obstacle free configuration space and assigns the random samples to the random state variable $z_{rand} \in X_{free}$.

*Distance:* Determines the Euclidean distance, defined as cost, between two input points.
*Nearest Node:* Locates the node in the random tree, $T$, nearest to the random sample.
*Near Nodes:* This function returns the set of nodes located within the confines of a sphere having a predefined volume $\gamma*(\frac{\log n}{n})^d$ around $z_{rand} \in X$. Any given tree has $n$ vertices and $d$ dimensions while $\gamma$ is a constant.

*Steering:* Defines a control input $u_{new}$ connecting $x_{new}$ to $z_{nearest}$ in the configuration space. $z_{new}$ is a point located at a small incremental distance $\delta$ from $z_{nearest}$, on the trajectory $z_{nearest}$ to $z_{rand}$ .

*Collision checking:* Determines whether a given path cuts across any obstacle in the configuration space.

*Insert Node:* Connects the new node $z_{new}$ to the tree by joining it with its parent node $z_{parent}$. This function also keeps track of the cost of $z_{new}$ and the cost already incurred by $z_{parent}$.

```
1: $T \leftarrow InitializeTree()$;
2: $T \leftarrow InsertNode(\theta, z_{init}, T)$;
3: $i \leftarrow 0$;
4: while $i \leq N$ do
5:     $z_{rand} \leftarrow Sample(i)$;
6:     $z_{nearest} \leftarrow Nearest(T, z_{rand})$;
7:     $(x_{new}, u_{new}, T_{new}) \leftarrow Steer(z_{nearest}, z_{rand})$;
8:     if $Obstaclefree(x_{new})$ then
9:         $Z_{near} \leftarrow Near(T, z_{new}, |V|)$;
10:        $Z_{min} \leftarrow$
           $ChooseParent(Z_{near}, z_{nearest}, z_{new}, x_{new})$;
11:        $T \leftarrow InsertNode(z_{min}, z_{new}, T)$;
12:        $T \leftarrow Rewire(T, Z_{min}, Z_{near}, z_{new})$;
13:    end if
14:    $i \leftarrow i + 1$;
15: end while
16: $returnT$
```

**Algorithm 1:** $T = (V, E) \leftarrow RRT * (z_{init})$

## B. Artificial Potential Fields

Artificial Potential Fields utilize artificial potential represented as $U \in \mathbb{R}$, in a configuration space $Q \in \mathbb{R}^d$ to direct a robot from its starting position $Q_{init} \in Q$ to its goal position $Q_{goal} \in Q$. Negative gradient of potential $-\bigtriangledown U \in \mathbb{R}$ is a vector quantity guiding the robot's movements. $U_{att} \in U$ are attractive potentials, $U_{rep} \in U$ are repulsive potentials and their gradients are $\bigtriangledown U_{att}$ and $\bigtriangledown U_{rep}$ respectively.

In order to move the robot towards the goal and away from any obstacles $Q_{obs} \in Q$ in the configuration space, the goal region is made to generate an attractive potential given by Equation (1) while obstacles are made to generate repulsive potentials using equations (5) $\rightarrow$ (8). The gradient of the attractive potential is computed using Equations (3) and (4).

$$U_{att} = \frac{1}{2} * K_a * d^2(q, q_{goal}), d(q, q_{goal}) < d^*_{goal} \quad (1)$$

$$U_{att} = d^*_{goal} * K_a * d(q, q_{goal}) - \frac{1}{2} * K_a * (d^*_{goal})^2, \\ d(q, q_{goal}) > d^*_{goal} \quad (2)$$

$$\bigtriangledown U_{att} = K_a * d(q, q_{goal}), d(q, q_{goal}) < d^*_{goal} \quad (3)$$

$$\bigtriangledown U_{att} = \frac{(d^*_{goal} * K_a * (q - q_{goal}))}{d(q, q_{goal})}, d(q, q_{goal}) < d^*_{goal} \quad (4)$$

As depicted by Equations (1) and (2), attractive potential varies quadratically only if the Euclidean distance of a given test point $q$ is larger than the constant distance $d^*_{goal}$ and conically otherwise. This allows a robot to advance swiftly when it is far away from the goal and slowly when it comes close, helping to avoid a situation where the robot mistakenly overshoots the goal.

$$d_i(q) = \min_{c \in O_i} d(q, c) \quad (5)$$

$$U_{rep} = \frac{1}{2} * K_r * (\frac{1}{d_i(q)} - \frac{1}{Q^*_i}), d_i(q) \leq Q^*_i \quad (6)$$

$$U_{rep} = 0, d_i(q) \geq Q^*_i \quad (7)$$

$$U_{rep} = \sum_{i=0}^{N} U_{rep_i} \quad (8)$$

The gradient of repulsive potentials is then calculated using Equations (9) $\rightarrow$ (12). The variable $d_i(q)$ is the Euclidean distance to the nearest obstacle $O_i$ calculated using Equation (5). The minimum operator outputs the shortest distance $d(q, c)$, where $c$ represents the point on the nearest obstacle closest to the test point.

$$\bigtriangledown d_i = \frac{(q - c)}{d(q, c)} \quad (9)$$

$$\bigtriangledown U_{rep_i} = K_r * (\frac{1}{Q^*_i} - \frac{1}{d_i(q)}) * \frac{\bigtriangledown d_i(q)}{d_i^2}, d_i(q) \leq Q^*_i \quad (10)$$

$$\bigtriangledown U_{rep_i} = 0, d_i(q) \geq Q^*_i \quad (11)$$

$$\bigtriangledown U_{rep} = \sum_{i=0}^{N} \bigtriangledown U_{rep_i} \quad (12)$$

If the distance between the robot and the nearest obstacle is greater than the constant $Q^*_i$, the algorithm dictates that the repulsive potential be considered as zero. This tells the robot that the obstacle is at a safe distance from the robot's current position and that it should continue moving quickly towards the goal. $K_a$ and $K_r$ are scaling factors that determine the size of the attractive and repulsive potentials as per current configuration space conditions. Algorithm 2 illustrates the gradient descent procedure used by Artificial Potential Fields where $\delta$ represents a small incremental distance.

```
1: $q(0) \leftarrow q_{init}$;
2: while $\bigtriangledown U \neq 0$ do
3:     $\bigtriangledown U \leftarrow PotentialGradient(q_i)$;
4:     $q_{i+1} \leftarrow q_i + \delta * (\frac{\bigtriangledown U}{|\bigtriangledown U|})$;
5:     $i \leftarrow i + 1$;
6: end while
```
**Algorithm 2:** $V \leftarrow GradientDescent(q_{init})$

## III. Adaptive Potential Guided Directional-RRT*

This section presents our Adaptive Potential Guided Directional-RRT* (APGD-RRT*) algorithm, a modification of the RRT* algorithm which uses Artificial Potential Fields to directionalize its random sampling. The term adaptive points to the algorithm's automatic tuning of its attractive ($K_a$) and repulsive ($K_r$) scaling factors.

$z_{prand} \in X$ represents the new potentially guided random sample. The random state $z_{rand} \in X$ is provided an offset in the direction of decreasing attractive potential field gradient according to the step size $\alpha$. The gradient of the attractive potential decreases when the distance between the random sample and the goal region decreases. The direction denoted by $\vec{d}$ is therefore from $z_{rand} \in X$ to the goal point, i.e., the direction of decreasing attractive potential fields.

Algorithm 3 outlines working of Adaptive Potential Guided Directional-RRT*. Difference between RRT* and APGD-RRT* occurs in line 6 where the Randomized Gradient Descent planning incorporates APF with the RRT* algorithm.

```
1: $T \leftarrow InitializeTree()$;
2: $T \leftarrow InsertNode(\theta, z_{init}, T)$;
3: $i \leftarrow 0$;
4: while $i \leq N$ do
5:     $z_{rand} \leftarrow Sample(i)$;
6:     $z_{prand} \leftarrow RGD(z_{rand})$;
7:     $z_{nearest} \leftarrow Nearest(T, z_{prand})$;
8:     $(x_{new}, u_{new}, T_{new}) \leftarrow Steer(z_{nearest}, z_{prand})$;
9:     if $Obstaclefree(x_{new})$ then
10:        $Z_{near} \leftarrow Near(T, z_{new}, |V|)$;
11:        $Z_{min} \leftarrow$
           $ChooseParent(Z_{near}, z_{nearest}, z_{new}, x_{new})$;
12:        $T \leftarrow InsertNode(z_{min}, z_{new}, T)$;
13:        $T \leftarrow Rewire(T, Z_{min}, Z_{near}, z_{new})$;
14:    end if
15:    $i \leftarrow i + 1$;
16: end while
17: $returnT$;
```
**Algorithm 3:** $T = (V, E) \leftarrow APGD - RRT * (z_{init})$

APGD-RRT* computes $z_{prand} \in X$ using Randomized Gradient Descent (RGD) Planning described in Algorithm 4. Following are few of its select procedures:

*Attractive Potential:* Returns the attractive potential and its gradient which are then used to calculate the direction $\vec{d}$ from $z_{rand}$ to $X_{goal}$.

*Find Obstacles:* Uses the variables direction $\vec{d}$ and obstacle region $X_{obs}$ to find the obstacles $Z_{obs} \in X_{obs}$ occurring in the direction of $\vec{d}$.

*Repulsive Potential:* Calculates repulsive potentials produced by obstacles returned by the Find obstacles function. *Computing Step Size:* This process uses the calculated repulsive potentials and set of obstacles along $\vec{d}$ to compute a step size $\alpha$ used to give an an offset to $z_{rand}$, creating $z_{prand}$.

```
1: $U_{att}, \bigtriangledown U_{att} \leftarrow AttractivePotential(X_{goal}, z_{rand})$;
2: $\vec{d} \leftarrow FindDirection(z_{rand}, U_{att}, \bigtriangledown U_{att})$;
3: $Z_{obs} \leftarrow FindObstacles(\vec{d}, X_{obs})$;
4: $U_{rep}, \bigtriangledown U_{rep} \leftarrow RepulsivePotential(Z_{obs}, z_{rand})$;
5: $\alpha \leftarrow ComputeStepSize(z_{rand}, U_{rep}, \vec{d})$;
6: $z_{prand} \leftarrow (z_{rand} + \vec{d} * \alpha)$;
7: $returnz_{prand}$;
```
**Algorithm 4:** $z_{prand} \leftarrow RGD(z_{rand})$

Gradient Descent planning computes the next state using information about its previous state and works iteratively as shown in Algorithm 2 until $\bigtriangledown U \rightarrow 0$. However, in Randomized Gradient Descent Planning the next state is independent of the previous state and a random sample $z_{rand} \in X$ is moved in the direction of decreasing potential by a step size $\alpha$ to generate a new point $z_{prand} \in X$. Once this is done, Lines 6 $\rightarrow$ 13 of Algorithm 3 execute on $z_{prand}$ and the next random state $z_{rand}$ is provided to this function.

Potential gradient and step size $\alpha$ are explained below.

### A. Finding Potential Gradient

Equations (13) $\rightarrow$ (18) compute the artificial potential and potential gradients from the random sample $z_{rand}$. No tuning of scaling factors is required. Moreover, $Q^*$ and $d^*_{goal}$ are ignored so that potentials may be calculated in the entire configuration space. Only quadratic, not conical, variation in attractive potential is calculated to determine the direction as illustrated in Equation (13).

$$U_{att} \propto d^2(z_{rand}, X_{goal}) \tag{13}$$

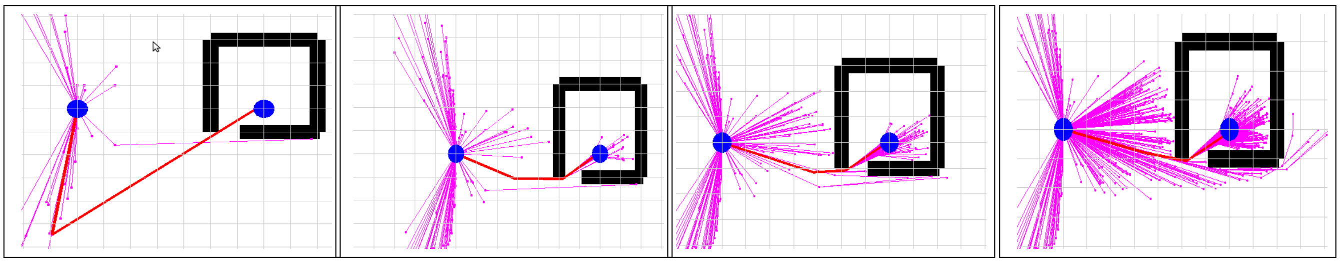
Not considering $d^*_{goal}$and assigning $K_a$ a value of one, Equation (1) simplifies to Equation (14). The potential gradient is computed by Equation (15).

$$U_{att} = \frac{1}{2} * d^2(z_{rand}, X_{goal}) \tag{14}$$

$$\bigtriangledown U_{att} = d(z_{rand}, X_{goal}) \tag{15}$$

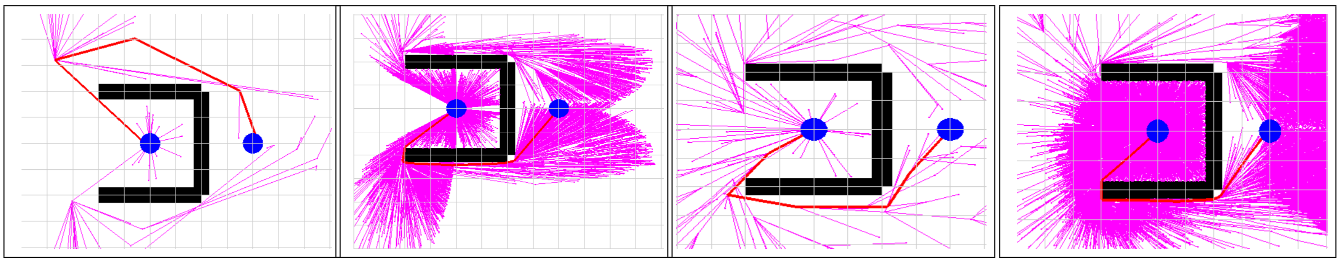
Variation in repulsive potential with respect to distance is shown in Equation (16), clearly specifying that as the distance $d(z_{rand}, X_{obs}) \rightarrow 0$, the repulsive potential $\bigtriangledown U_{rep} \rightarrow \infty$.

$$U_{rep} \propto (\frac{1}{d(z_{rand}, X_{obs})^2}) \tag{16}$$



(a) n=41,t=0.01,C=75.81 (b) n=200,t=0.03 and C=36.96 (c) n=500,t=0.34 and C=35.92 (d) n=1600,t=1.57 and C=35.82

Fig. 1: APGD-RRT* performance in Narrow Passage Environment



(a) APGD-RRT*-Initial Path: n=106, t=0.01, C=75.11 (b) APGD-RRT*-Final Path: n=20000, t=6.63, C=51.03 (c) RRT*-Initial Path: n=2000, t=1.27, C=57.46 (d) RRT*-Final Path: n=850000, t=140.6, C=50.65

Fig. 2: APGD-RRT* and RRT* performance in Local Minima Environment

Calculation of Repulsive potential and repulsive potential gradient is done using Equations (17) and (18). Not considering both $Q^*$ and $K_r$, Equation (6) reduces to (17) while the gradient of Equation (17) is calculated as shown in Equation (18).

$$U_{rep} = \frac{1}{2} * \left(\frac{1}{d(z_{rand}, Z_{obs})^2}\right) \tag{17}$$

$$\bigtriangledown U_{rep} = \left(\frac{1}{d(z_{rand}, Z_{obs})}\right) * \frac{\bigtriangledown (z_{rand}, Z_{obs})}{d(z_{rand}, Z_{obs})^2} \tag{18}$$

### B. Step Size

The set of obstacles in $Z_{obs} \in X_{obs}$ as determined by the Find Obstacles procedure determine the value of the step size $\alpha$. The obstacle closest to the test point $z_{rand}$, easily be recognized in the set space as the obstacle creating the strongest repulsive field, is assigned to $O_i$. Equation (19) then determines the distance $d(z_{rand}, c)$ between the test point and the point on obstacle $O_i$ in the direction of $\vec{d}$ that is closest to the test point. $d_i(z_{rand})$ is simply the distance between the closest obstacle and the test point.

$$d_i(z_{rand}) = \min_{c \in O_i} d(z_{rand}, c) \tag{19}$$

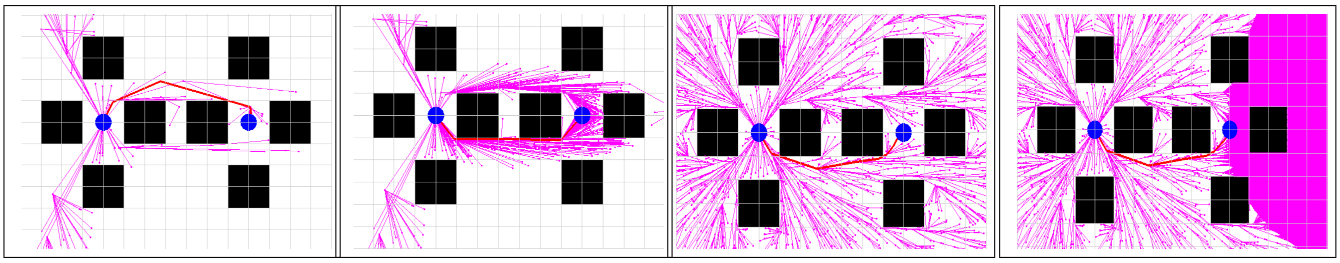
Once the minimum separation between the test point and the nearest obstacle is determined, this distance is divided into 5 equal sized lengths $s_1, s_2 ... s_5$, each length representing a node. Step size $\alpha$ gives $z_{rand}$ an offset to any one of these randomly selected nodes and calls it $z_{prand}$.

## IV. Experimental Results

Performance results of APGD-RRT* in different environments are now presented. Black shapes in the configuration space represent obstacles while blue circles are the initial and goal points. The purple network of lines is the growing rapidly exploring random tree while the dark red line is the chosen optimal path given by APGD-RRT*. The purple coating in some figures is due to large number of random samples taken from the configuration space and all figures represent the small sections of whole configuration space.

Figure 1 is a narrow passage environment to which APGD-RRT* is applied. Performance of the algorithm is detailed in terms of cost, time and number of iterations required to reach optimal solution. Cost is the Euclidean distance between successive nodes from start to goal points, time is the number of seconds utilized by the algorithm to run the given number of iterations while 'n' is the number of iterations that were run. The sub-figures depict the eventual convergence of the APGD-RRT* algorithm to an optimal/near optimal path solution in a mere 1600 number of iterations. This is significant since APF alone is unable to perform in a narrow passage environment such as this. This is due to the repulsive potential generated by obstacles on all 4 sides of the starting point which prevent the robot from approaching the narrow passage opening. APGD-RRT* on the other hand, resolves this issue by using APF to directionalize random samples towards the goal region and then using them to quickly find a path leading to the goal point. Figure 1 shows the potential guided random samples concentrated in the direction of the goal instead of wastefully dispersed in the entire configuration space.

Figure 2 illustrates the performance of both APGD-RRT*



(a) APGD-RRT*-Initial Path: n=200,t=0.02 and C=43.09 (b) APGD-RRT*-Final Path: n=3000,t=2.14 and C=38.68 (c) RRT*-Initial Path: n=16063, t=0.57, C=39.62 (d) RRT*: n=10mil, t=238.68, C=39.46

Fig. 3: Performance of APGD-RRT* and RRT* in Cluttered Environment

and RRT* in a local minima environment. It can be seen that introducing randomness in APF algorithms allow them operate in this environment and lead APGD-RRT* towards an initial solution much faster than RRT*, i.e, in n=106 iterations as compared to RRT*'s n=2000 iterations. Consequently, the final optimum path is achieved by APGD-RRT* in only n=20000 iterations instead of RRT*'s n=850000 iterations.

Figure 3 show the performance of APGD-RRT* and RRT* in the same cluttered environment. APGD-RRT* can be seen to locate the initial path in much fewer number of iterations (n=200) as compared to RRT*(n=16063). Predictably, it also takes less iterations (n=3000) to determine the optimal/near optimal path from start to goal points while RRT* takes an extremely large number of iterations (n=10million) and still fails to achieve the optimal solution. Since RRT* was unable to determine the optimal path even after 10 million iterations , the cost of the path that RRT* finds after 10 million iterations (C=39.46) is also greater than the cost of the optimal path determined by APGD-RRT* (C=38.68).

## V. Statistical Comparison

This section presents comparisons of performance of APGD-RRT* and RRT* in different environments. Comparisons are made on the basis of iterations required and time consumed to achieve initial and optimized path in 10 different environments. Furthermore, statistical t-test is performed to compare convergence rate of APGD-RRT* and RRT* in more than 50 environments. Convergence rate is governed by the equation 20.

$$\frac{CostOfInitialPath - CostOfOptimizedPath}{TimeToGetOptimizedPath - TimeToGetInitialPath} \tag{20}$$

Figure 4 clearly shows that in all test environments APGD-RRT* manages to find the initial path very quickly as compared to RRT*. This is due to the directionalized sampling feature of APGD-RRT* as explained previously. As expected, time consumed by APGD-RRT* in all test environments is also significantly less than that of RRT* as can seen in Figure 5.

This trend continues in the optimal path solutions. Figure 6 shows that once again, APGD-RRT* converges to the

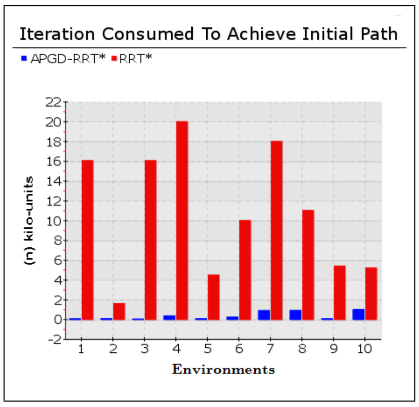


Fig. 4: Iterations Utilized To Find Initial Path

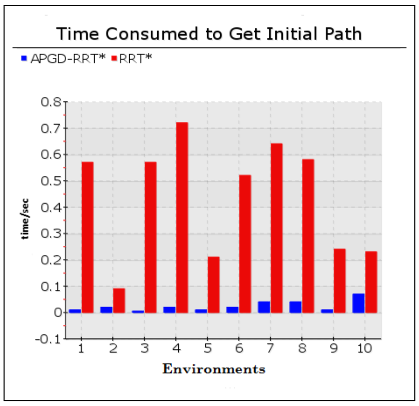


Fig. 5: Time Consumed To Find Initial Path

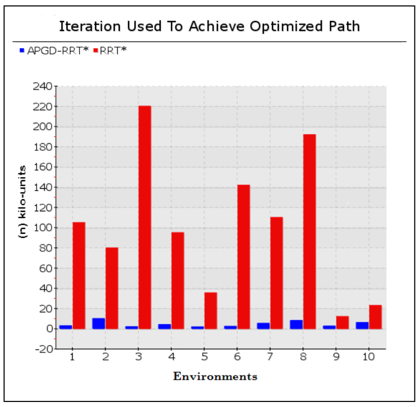


Fig. 6: Iterations Utilized To Find Final Path

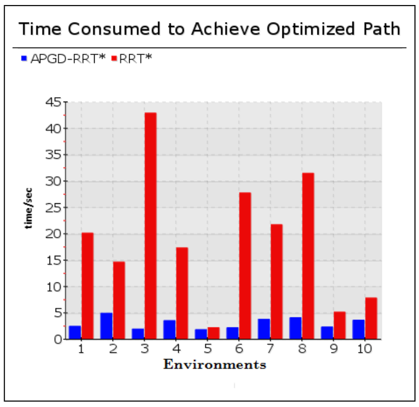


Fig. 7: Time Utilized To Find Final Path

| | APGD-RRT* | RRT* |
|---|---|---|
| Mean | 4.61 | 1.28 |
| Standard Deviation | 2.30 | 1.13 |
| Highest Value | 11.4 | 4.92 |
| Lowest Value | 1.28 | $2 * 10^{-3}$ |
| Median | 4.17 | 1.04 |
| Average Absolute Deviation From Median | 1.74 | 0.791 |

TABLE I: t-test: Comparison of APGD-RRT* and RRT*

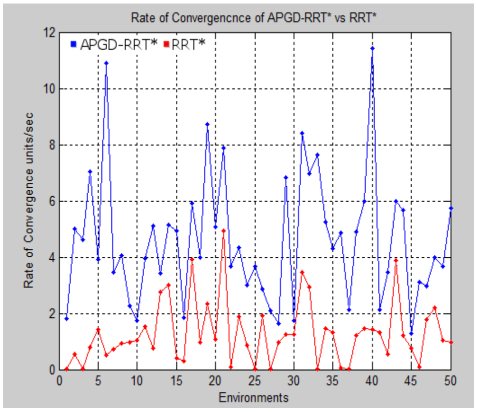


Fig. 8: Rate of Convergence in fifty different environments

optimal path solution much faster as compared to RRT*. Consequently, as shown in Figure 7, it also requires much less execution time.

Figure 8 compares convergence rate of APGD-RRT* and RRT* in fifty different environments and the convergence rate of APGD-RRT* remains significantly higher as compared to RRT*, which proves fast converging abilities of APGD-RRT*. Table I demonstrate the results of the t-test performed. The null hypothesis states that the convergence rate of the two algorithm is same. The t-value is 9.18 and the critical value is 1.98 with 98 degree of freedom. Therefore, on the basis of critical value and t-value we can reject the null hypothesis, i.e., mean convergence rate of APGD-RRT* is significantly greater ($p < 0.05$) than that of RRT*. Moreover, Table I also shows the highest and lowest value of convergence rate achieved by each algorithm.

## VI. Conclusions

While RRT* is a proficient optimal path finding algorithm in environments whose obstacle geometry may be unknown, it still falls short in terms of memory efficiency and fast convergence rate. Adaptive Potential Guided Directional-RRT* deals with this shortcoming by integrating Artificial Potential Field Algorithm with RRT*, resulting in significantly reduced memory requirements and faster convergence to optimal solution. This claim has been tested in various different environments and APGD-RRT* has been proven to give much better performance than RRT* in each of them. In continuation of this work, we shall attempt to implement APGD-RRT* algorithm on Poineer 3AT mobile robots and perform detailed hardware analysis.

## VII. Acknowledgments

This work is supported by the collaborative research and funding of RISE Lab. (Dept. Robotics and AI, SMME, NUST, Pakistan) and IE Lab. (Dept. of EE, College of Engineering, HYU, Korea, under the grant number NIPA-2013-H0301-13-1011 of MSIP, Korea)

## References

[1] H. M. Choset, *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[2] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[3] J. Cortés, T. Siméon, V. R. De Angulo, D. Guieysse, M. Remaud-Siméon, and V. Tran, “A path planning approach for computing large-amplitude motions of flexible molecules,” *Bioinformatics*, vol. 21, no. suppl 1, pp. i116–i125, 2005.

[4] Y. Liu and N. I. Badler, “Real-time reach planning for animated characters using hardware acceleration,” in *Computer Animation and Social Agents, 2003. 16th International Conference on*. IEEE, 2003, pp. 86–93.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[6] S. M. LaValle and J. J. Kuffner, “Randomized kinodynamic planning,” *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[7] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *The international journal of robotics research*, vol. 5, no. 1, pp. 90–98, 1986.

[8] S. R. Lindemann and S. M. LaValle, “Incrementally reducing dispersion by increasing voronoi bias in rrts,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, vol. 4. IEEE, 2004, pp. 3251–3257.

[9] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle, “Dynamic-domain rrts: Efficient exploration by controlling the sampling domain,” in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 3856–3861.

[10] S. Khanmohammadi and A. Mahdizadeh, “Density avoided sampling: an intelligent sampling technique for rapidly-exploring random trees,” in *Hybrid Intelligent Systems, 2008. HIS’08. Eighth International Conference on*. IEEE, 2008, pp. 672–677.

[11] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[12] P. Vadakkepat, T. H. Lee, and L. Xin, “Application of evolutionary artificial potential field in robot soccer system,” in *IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th*. IEEE, 2001, pp. 2781–2785.

[13] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, “Anytime motion planning using the rrt*,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1478–1483.