Gaussain and Informed Sampling based RRT* Motion Planning on a 7DOF Robot

ECE 276 C Robot Manipulation & Control
Sanchit Gupta ECE ISRC Kishore Nukala ECE ISRC

I. Introduction

Motion Planning problem in Robotics is one of the most widely researched topics in Robotics and there are numerous ways of efficiently doing it based on the robot and the environment. One common approach to solve the motion-planning problem involves discretizing the continuous state space. This can be done by using a grid for graph-based searches or by randomly sampling points for stochastic incremental searches. Graph-based searches, like A*, are known for being probabilistic complete, and optimal. They are guaranteed to find the best solution, if one exists, and indicate failure otherwise (within the limits of the discretization). However, these graph-based algorithms do not scale well when faced with larger problem sizes, such as increased state dimensions or problem ranges or complex environments with dynamic obstacles.

Stochastic searches, on the other hand, employ sampling-based methods like Rapidly-exploring Random Trees (RRTs), Probabilistic Roadmaps (PRMs), and Expansive Space Trees (ESTs). These methods avoid the need for discretizing the state space, allowing for better scalability with larger problem sizes and the consideration of kino-dynamic constraints. However, they do not provide a strict completeness guarantee. RRTs, for instance, offer probabilistic completeness, meaning that the probability of finding a solution, if one exists, approaches certainty as the number of iterations increases. In the case of 7 DOF robots like a Panda Robot, search-based sampling is a desired one as the free space of the robot is 3D which makes it difficult to discretize as well as the use cases of Robotic Manipulators require continuous free space instead of discretized one.

Optimal RRTs (RRT*s) extend RRTs to the problem of finding the optimal solution, but in doing so asymptotically find the optimal path from the initial state to every state in the planning domain. This behavior is not only inefficient but also inconsistent with their single-query nature. Moreover, the inherent sampling proposed in the Vanilla RRT* involves sampling a random sample from the entire configuration space which can be very time-consuming for large environments. To study the performance of a heuristic-based sampling for determining the entire trajectory or sampling from a probabilistic subset of the configuration space for rewiring we propose two different sampling techniques. To analyse their performance we use a 7DOF Panda robot and navigate in a complex environment with different obstacles.

II. BackGround

Panda Robot is a sensitive, interconnected, and adaptive robotic arm. The robotic arm is inspired by human agility and a sense of touch. With torque sensors in all seven axes, the arm can delicately manipulate objects, and accomplish programmed tasks. As mentioned, we use 7 DOF Panda robot for our analysis placed in a complex environment with obstacles as shown in fig1 and fig2. Each of the 7 joint angles has its own constraints or range of movements as mentioned below in Table 1:

TABLE I
Joint constraints of the Panda Robot (in degrees)

	q1	**q2**	**q3**	**q4**	**q5**	**q6**	**q7**
$\min_q$	-166.0	-101.0	-166.0	-176.0	-166.0	-1.0	-166.0
$\max_q$	166.0	101.0	166.0	-4.0	166.0	215	166.0

We used a DH model representation of the Panda Robot from the robotics toolkit and the DH values used for the project are given in the following Table 2.

TABLE II
DH Parameters of Panda Robot

a_{j-1}	α_{j-1}	θ_j	d_j
0.0	$0.0°$	q_1	0.333
0.0	$-90.0°$	q_2	0.0
0.0	$90.0°$	q_3	0.316
0.0825	$90.0°$	q_4	0.0
-0.0825	$-90.0°$	q_5	0.384
0.0	$90.0°$	q_6	0.0
0.088	$90.0°$	q_7	0.107

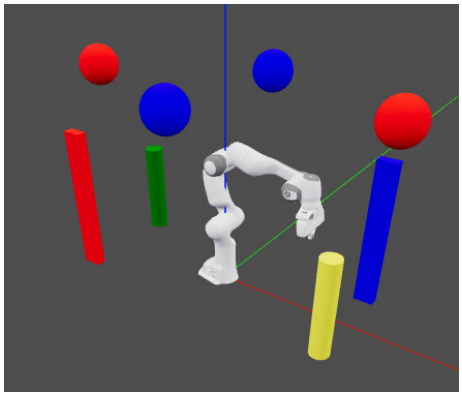

Fig. 1. Isometric view of Robot in workspace

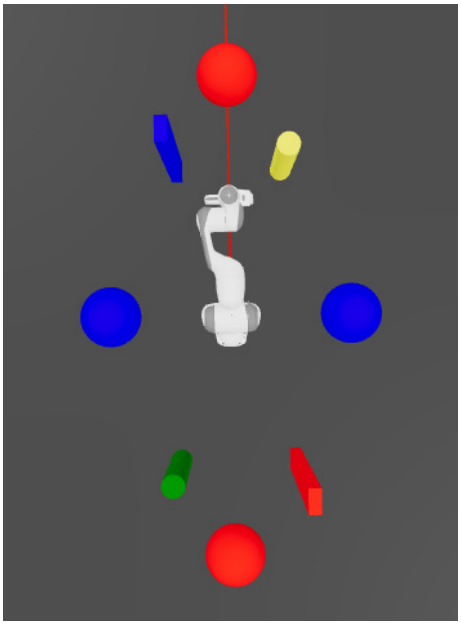

Fig. 2. Top view of Robot in workspace

In order to solve a motion planning problem on a robot, we have to identify the obstacle space, free space, and configuration space of the robot before performing RRT Algorithm. Since the control is allowed on the joint angles we will be considering the configuration space of the joint angles which is a 7-dimensional space of real numbers with the constraint limits as per the joint angles. Let $X \subseteq \mathbb{R}^n$ be the state space of the planning problem. Let $X_{\text{obs}} \subset X$ be the states in collision with obstacles, and $X_{\text{free}} = X \backslash X_{\text{obs}}$ be the resulting set of permissible states. Let $x_{\text{start}} \in X_{\text{free}}$ be the initial state, and $x_{\text{goal}} \in X_{\text{free}}$ be the desired final state. Let $\sigma : [0,1] \to X$ be a sequence of states (a path), and Σ be the set of all nontrivial paths. The optimal planning problem is then formally defined as the search for the path σ^* that minimizes a given cost function $c : \Sigma \to \mathbb{R}_{\geq 0}$, while connecting x_{start} to x_{goal} through free space.

$$\sigma^* = \underset{\sigma \in \Sigma}{\arg\min} \{c(\sigma) \mid \sigma(0) = \mathbf{x}_{\text{start}}\,, \sigma(1) = \mathbf{x}_{\text{goal}}\,, \\ \forall s \in [0,1], \sigma(s) \in X_{\text{free}}\}\,,$$

where $\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers.

We are using the CoppeliaSim as the simulator for visualizing the environment and analyzing the motion of the manipulator. Obstacles of different shapes (2–Cuboids, 2–Cylinders, and 4 –Spheres) are used as the environment. Since it is difficult to visualize a 7 DOF configuration space, we provide all the results in a 2D space (x,y) in the report and the path of the robot as the position of the End Effector. we will be analyzing two different methods of sampling and rewiring on Vanilla RRT* to analyze the performance of the modified RRT* algorithm. Detailed methodology and the algorithm are explained in the next section.

III. Technical Approach

1) RRT::* As previously mentioned, the RRT algorithm does not provide a guarantee of asymptotic optimality. To address this limitation, the RRT* algorithm was introduced. The RRT* algorithm incorporates incremental rewiring of the graph, wherein newly added states are not only connected to the existing tree but also considered as potential replacement parents for nearby states in the tree. By employing uniform global sampling, the RRT* algorithm asymptotically discovers the optimal solution to the planning problem by gradually identifying the optimal paths from the initial state to the goal state. However in the random sampling step, RRT* searches the entire configuration space irrespective of the distance between the start and goal point i.e, no heuristics are involved in the sampling stage. This can be addressed to an extent by implementing a sampling bias in the sampler i.e., in each step the goal point is sampled with a certain small probability. This Sampling bias helps in getting a faster convergence comparatively but still the problem of searching the entire config space remains the same. Moreover, in an RRT* the rewiring procedure results in an algorithm that asymptotically finds the optimal solution to the planning problem by asymptotically finding the optimal paths from the initial state to every state in the problem domain, which is not required in a single query system. So, we propose 2 different modifications over Vanilla RRT* :

- Informed RRT* which restricts the subset of sampling for rewiring to an ellipse region with start and goal as focus points once a potential path to the goal has been identified.
- Introducing a heuristic metric-based sampling using Euclidean distance and multi-variate Gaussian distribution.

To summarise RRT* we can consider the following algorithm:

Algorithm RRT*

$V \leftarrow \{\mathbf{x}_s\}$; $E \leftarrow \emptyset$
for $i = 1 \ldots n$ **do**
  $\mathbf{x}_{rand} \leftarrow \textsc{SampleFree}()$
  $\mathbf{x}_{nearest} \leftarrow \textsc{Nearest}((V, E), \mathbf{x}_{rand})$
  $\mathbf{x}_{new} \leftarrow \textsc{Steer}(\mathbf{x}_{nearest}, \mathbf{x}_{rand})$
  if $\textsc{CollisionFree}(\mathbf{x}_{nearest}, \mathbf{x}_{new})$ **then**
    $X_{near} \leftarrow \textsc{Near}((V, E), \mathbf{x}_{new}, \min\{r^*, \epsilon\})$
    $V \leftarrow V \cup \{\mathbf{x}_{new}\}$
    $c_{min} \leftarrow \textsc{Cost}(\mathbf{x}_{nearest}) + \textsc{Cost}(\mathit{Line}(\mathbf{x}_{nearest}, \mathbf{x}_{new}))$
    for $\mathbf{x}_{near} \in X_{near}$ **do**
      if $\textsc{CollisionFree}(\mathbf{x}_{near}, \mathbf{x}_{new})$ **then**
        if $\textsc{Cost}(\mathbf{x}_{near}) + \textsc{Cost}(\mathit{Line}(\mathbf{x}_{near}, \mathbf{x}_{new})) < c_{min}$ **then**
          $\mathbf{x}_{min} \leftarrow \mathbf{x}_{near}$
          $c_{min} \leftarrow \textsc{Cost}(\mathbf{x}_{near}) + \textsc{Cost}(\mathit{Line}(\mathbf{x}_{near}, \mathbf{x}_{new}))$
    $E \leftarrow E \cup \{(\mathbf{x}_{min}, \mathbf{x}_{new})\}$
    for $\mathbf{x}_{near} \in X_{near}$ **do**
      if $\textsc{CollisionFree}(\mathbf{x}_{new}, \mathbf{x}_{near})$ **then**
        if $\textsc{Cost}(\mathbf{x}_{new}) + \textsc{Cost}(\mathit{Line}(\mathbf{x}_{new}, \mathbf{x}_{near})) < \textsc{Cost}(\mathbf{x}_{near})$ **then**
          $\mathbf{x}_{parent} \leftarrow \textsc{Parent}(\mathbf{x}_{near})$
          $E \leftarrow (E \setminus \{(\mathbf{x}_{parent}, \mathbf{x}_{near})\}) \cup \{(\mathbf{x}_{new}, \mathbf{x}_{near})\}$
return $G = (V, E)$

Fig. 3.

As mentioned in the algorithm at each iteration, we check for the collision-free nature of the new edge being added and subsequent rewiring of the neighboring nodes to update the cost to reach value. Also, parent-child edge details are also updated in each iteration to retrieve the entire path at the end of iterations. A total of 6 different paths were generated using each algorithm and a detailed analysis and comparison of the same is provided in the Results section.

2) Informed RRT:* Informed RRT* retains the same probabilistic guarantees on completeness and optimality as RRT* while improving the convergence rate and final solution quality achieved. Basically to improve the quality of any path planning solution we need to add more states in the nearby vicinity of the problem domain that can put in more information and help in minimizing the path length. From the Informed RRT* we know that once a potential solution has been identified for a path planning problem using RRT*, the quality of the solution can be improved by restricting the sampling domain to a ellipsoid region between the start and the goal region. Informed RRT* follows the same behavior as RRT* initially, where it explores the state space to find a solution. However, once a solution is found, it transitions to a different strategy. It restricts its sampling to a subset of states defined by an admissible heuristic, aiming to enhance the existing solution. This subset of states implicitly maintains a balance between exploitation and exploration, eliminating the need for further tuning. In simpler configurations, there wouldn't be much of a significant difference observed between RRT* and Informed RRT* but the difference becomes evident in complex configurations. And, this is one of the main reasons for us to choose a complex environment. Some advantages as mentioned in the Informed RRT* paper are:

- Better results than RRT* in complex environments.
- Less dependence on the dimension and domain of the planning problem as well as the ability to find better topologically distinct paths sooner.
- Capable of finding solutions within tighter tolerances of the optimum than RRT* with equivalent computation.
- It could also be used in combination with other algorithms, such as path smoothing, to further reduce the search space.

Informed RRT* has been implemented on 7 DOF and similar behavior as mentioned above is observed. To summarise the implementation, we can refer to the following Algorithm 1 and 2 of Informed RRT*.

In Informed RRT*, as solutions are discovered (line 30), they are appended to a list of potential solutions (line 31). The minimum value from this list (line 6) is utilized to calculate and directly sample the state space X_f (line 7). It is customary to consider the minimum of an empty set as infinity.

Given a pose $x \in X_{\text{free}}$, the function InGoalRegion(x) returns True if and only if the state is in the goal region X_{goal}, as defined by the planning problem. One common goal region is a ball of radius r_{goal} centered about the goal, i.e., $X_{\text{goal}} = \{x \in X_{\text{free}} \mid \|x - x_{\text{goal}}\|^2 \leq r_{\text{goal}}^2\}$. The value of r_{goal} is a hyperparameter for our algorithm and has been identified through trial and error.

Algorithm 1: Informed RRT*$(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$

1 $V \leftarrow \{\mathbf{x}_{\text{start}}\}$;
2 $E \leftarrow \emptyset$;
3 $X_{\text{soln}} \leftarrow \emptyset$;
4 $\mathcal{T} = (V, E)$;
5 **for** iteration $= 1 \dots N$ **do**
6 $\quad c_{\text{best}} \leftarrow \min_{\mathbf{x}_{\text{soln}} \in X_{\text{soln}}} \{\texttt{Cost}(\mathbf{x}_{\text{soln}})\}$;
7 $\quad \mathbf{x}_{\text{rand}} \leftarrow \texttt{Sample}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}, c_{\text{best}})$;
8 $\quad \mathbf{x}_{\text{nearest}} \leftarrow \texttt{Nearest}(\mathcal{T}, \mathbf{x}_{\text{rand}})$;
9 $\quad \mathbf{x}_{\text{new}} \leftarrow \texttt{Steer}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{rand}})$;
10 $\quad$ **if** $\texttt{CollisionFree}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$ **then**
11 $\quad\quad V \leftarrow \cup\{\mathbf{x}_{\text{new}}\}$;
12 $\quad\quad X_{\text{near}} \leftarrow \texttt{Near}(\mathcal{T}, \mathbf{x}_{\text{new}}, r_{\text{RRT}^*})$;
13 $\quad\quad \mathbf{x}_{\min} \leftarrow \mathbf{x}_{\text{nearest}}$;
14 $\quad\quad c_{\min} \leftarrow \texttt{Cost}(\mathbf{x}_{\min}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$;
15 $\quad\quad$ **for** $\forall \mathbf{x}_{\text{near}} \in X_{\text{near}}$ **do**
16 $\quad\quad\quad c_{\text{new}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{near}}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})$;
17 $\quad\quad\quad$ **if** $c_{\text{new}} < c_{\min}$ **then**
18 $\quad\quad\quad\quad$ **if** $\texttt{CollisionFree}(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})$ **then**
19 $\quad\quad\quad\quad\quad \mathbf{x}_{\min} \leftarrow \mathbf{x}_{\text{near}}$;
20 $\quad\quad\quad\quad\quad c_{\min} \leftarrow c_{\text{new}}$;
21 $\quad\quad E \leftarrow E \cup \{(\mathbf{x}_{\min}, \mathbf{x}_{\text{new}})\}$;
22 $\quad\quad$ **for** $\forall \mathbf{x}_{\text{near}} \in X_{\text{near}}$ **do**
23 $\quad\quad\quad c_{\text{near}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{near}})$;
24 $\quad\quad\quad c_{\text{new}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{new}}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})$;
25 $\quad\quad\quad$ **if** $c_{\text{new}} < c_{\text{near}}$ **then**
26 $\quad\quad\quad\quad$ **if** $\texttt{CollisionFree}(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})$ **then**
27 $\quad\quad\quad\quad\quad \mathbf{x}_{\text{parent}} \leftarrow \texttt{Parent}(\mathbf{x}_{\text{near}})$;
28 $\quad\quad\quad\quad\quad E \leftarrow E \setminus \{(\mathbf{x}_{\text{parent}}, \mathbf{x}_{\text{near}})\}$;
29 $\quad\quad\quad\quad\quad E \leftarrow E \cup \{(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})\}$;
30 $\quad\quad$ **if** $\texttt{InGoalRegion}(\mathbf{x}_{\text{new}})$ **then**
31 $\quad\quad\quad X_{\text{soln}} \leftarrow X_{\text{soln}} \cup \{\mathbf{x}_{\text{new}}\}$;
32 **return** $\mathcal{T}$;

Fig. 4.

Algorithm 2: $\texttt{Sample}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}, c_{\max})$

1 **if** $c_{\max} < \infty$ **then**
2 $\quad c_{\min} \leftarrow \left|\left|\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}\right|\right|_2$;
3 $\quad \mathbf{x}_{\text{centre}} \leftarrow (\mathbf{x}_{\text{start}} + \mathbf{x}_{\text{goal}})/2$;
4 $\quad \mathbf{C} \leftarrow \texttt{RotationToWorldFrame}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}})$;
5 $\quad r_1 \leftarrow c_{\max}/2$;
6 $\quad \{r_i\}_{i=2,\dots,n} \leftarrow \left(\sqrt{c_{\max}^2 - c_{\min}^2}\right)/2$;
7 $\quad \mathbf{L} \leftarrow \text{diag}\{r_1, r_2, \dots, r_n\}$;
8 $\quad \mathbf{x}_{\text{ball}} \leftarrow \texttt{SampleUnitNBall}$;
9 $\quad \mathbf{x}_{\text{rand}} \leftarrow (\mathbf{C}\mathbf{L}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}) \cap X$;
10 **else**
11 $\quad \mathbf{x}_{\text{rand}} \sim \mathcal{U}(X)$;
12 **return** $\mathbf{x}_{\text{rand}}$;

Fig. 5.

SampleUnitNBall: The function returns a uniform sample, denoted by x_{ball}, from the volume of an n-ball with a unit radius centered at the origin. In other words, $x_{\text{ball}} \sim U(X_{\text{ball}})$. RotationToWorldFrame$(x_{\text{from}}, x_{\text{to}})$ is a function that returns the rotation matrix $C \in \text{SO}(n)$, which transforms coordinates from the hyper ellipsoid-aligned frame to the world frame. This rotation matrix C has been derived in the Informed RRT* paper.

3) Gaussian heuristic based RRT:* As mentioned earlier, one of the main disadvantages of RRT* is with respect to the sampling in which the algorithm samples randomly from the entire configuration space. Even when the path is nearer to the goal region, the sampling is in the entire region. To introduce a goal-informed capability in sampling we introduce a heuristic into the random sampler. A random sample is generated from a multi-variate Gaussian distribution with the current position of the robot as the mean and scaled heuristic metric as the covariance. This process is repeated until convergence. Two main advantages of using a heuristic-based Gaussian sampler are to restrict the region of the sampling as the robot nears the goal and thereby converge with minimum sampling. This particular feature also helps for better sampling in the neighborhood of the end effector in tight spaces. Gaussian-based sampling has also been tested for similar cases of Vanilla RRT*, and Informed RRT*, and the corresponding results are discussed in the next section.

IV. Results

For the analysis of the 3 algorithms, we have selected 6 waypoints in such a way as to challenge the algorithm in avoiding obstacles and finding an optimal path from the initial configuration. All the optimal paths generated for RRT* and Informed RRT* are simulated using the CoppileaSim simulator to better visualise the trajectory.Each of the algorithms has been tested on all the waypoints and it has been observed that the optimal path, number of iterations, and the time taken were different. In each of these cases, the distance used for the steer function and the threshold radius used for computing the neighborhood of a given point are hyperparameters and are identified after a series of trials and errors. These observations and comparisons are discussed in detail below.

TABLE III
Comparison of Algorithms

Algorithm	Time Taken for Optimal Path	Number of Iterations
RRT*	∼450 sec	8000
Informed RRT*	∼200 sec	5000
Gaussian RRT*	∼600 sec	3500

A. Comparison of RRT* and Informed RRT*

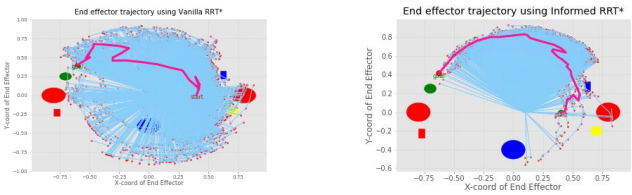

Fig. 6. 2D Visualization of Environment and Optimal Path RRT*(left) Informed RRT* (right)

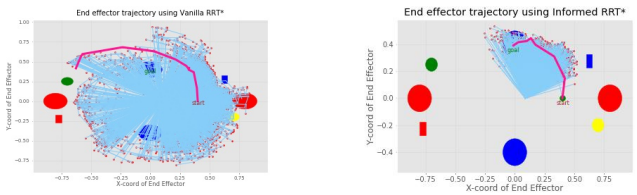

Fig. 7. 2D Visualization of Environment and Optimal Path RRT*(left) Informed RRT* (right)

We have presented a 2D top view of the environment and obstacles and plotted the x-y trajectory of the end effector of the robot. It can be observed that the parts of the robot are overlapped with the obstacles in 2D representation whereas in reality they are passing below/above the obstacles, thus successfully avoiding them.

- It can be observed that the sampling in RRT* is spread all over the place where as in Informed RRT* the sampling is restricted to an ellipsoidal region with start and goal points as its focus points. This ensures faster convergence with a more optimal path compared to RRT* for high DOF robots in complex environments.
- The End effector trajectory in RRT* can be observed to be jittery as compared to the smooth trajectory of End effector generated in Informed RRT*.
- Time taken for the Informed RRT* for finding an optimal path is less than the time taken by RRT* owing to its efficient rewiring and restricted sampling.
- Number of iterations required for the Informed RRT* was observed to be less than that of RRT* for the same hyper parameters of Steering and Neighbourhood radius.

Hence, lower number of iterations and faster convergence along with a more optimal path makes Informed RRT* a better and computationally efficient option over Vanilla RRT* for path planning for high DOF robots in complex environments.

With regard to Gaussian Heuristic-based RRT*, it can be observed that the behavior of the algorithm was inconsistent especially when sampling longer distance goals. As this sampling is completely based on the multi-variate Gaussian being generated, once the sample generated shifts in a direction away from the goal it might drive the rest of the algorithm away from the goal which is not a desired behavior. Gaussian RRT* was also observed to give good results when the goal point was nearer or the random sampling was not driving the algorithm away from the goal. In some cases, Gaussian RRT* was observed to give very fast convergence too but the behavior was not consistent. In the cases where the algorithm was successful, it was clearly observed to be performing very less sampling to obtain an optimal path. These two observations can be clearly understood from the following pictures. This behavior of the algorithm can be further analyzed to achieve consistency and efficiency.

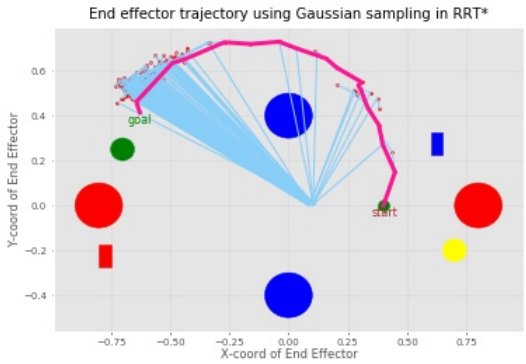

Fig. 8. Optimal Path computed using Gaussian RRT*

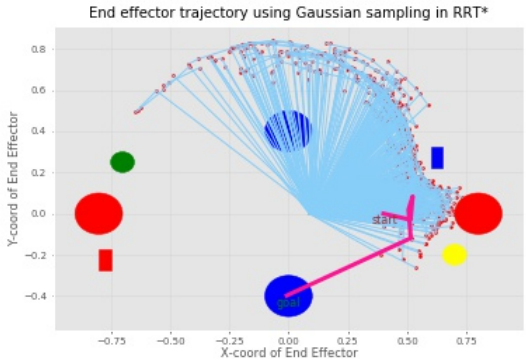

Fig. 9. Failed case using Gaussian RRT*

V. References

1) Jonathan D. Gammell, Siddhartha S. Srinivasa, Timothy D. Barfoot Informed RRT*: Optimal Sampling-based Path Planning Focused via Direct Sampling of an Admissible Ellipsoidal Heuristic 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)
2) Nikhil Das, Michael C. Yip Forward Kinematics Kernel for Improved Proxy Collision Checking
3) S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," IJRR, 30(7): 846–894, 2011.