Dynamic Informed RRT* Path Planner

Hritvik Choudhari
ENPM 661
University of Maryland
College Park, USA
hac@umd.edu

Abhimanyu Saxena
ENPM 661
University of Maryland
College Park, USA
asaxena4@umd.edu

***Abstract*—RRTs, or rapidly exploring random trees, are used often in motion planning because they are effective at solving single-query issues. RRTs are extended to the challenge of finding the best solution by optimum RRTs (RRT*s), which asymptotically determine the best route from the starting state to each state in the planning domain. This behavior is not only ineffective but also at odds with the single-query nature of their architecture. The subset of states that can enhance a solution for issues attempting to decrease path length can be characterized by a prolate hyperspheroid. We demonstrate that, in big worlds or high state dimensions, the chance of improving a solution becomes arbitrarily tiny unless this subset is explicitly sampled in this subspace.**

Informed RRT* works using the above heuristic approach guarantying a more optimal path compared to normal RRT* by improving the convergence rate. But this approach works best in the case of static environment. In case of dynamic obstacle space, this method won't be able to plan locally a path around it. In this project we will introduce a new approach that incorporates a local planner on top of global planner, both using informed RRT* to find an optimal path in both cases.

I. Introduction

A discretization of the state space is not necessary for stochastic searches, such as Rapidly-exploring Random Trees (RRTs), Probabilistic Roadmaps (PRMs), and Expansive Space Trees (ESTs). Instead, sampling-based techniques are used. This enables them to directly take into account Kino dynamic limitations and scale with issue size more efficiently; nevertheless, the end consequence is a less-strict completeness guarantee. RRTs are probabilistically complete, which ensures that as the number of iterations approaches infinity, the likelihood of discovering a solution if one exists, approaches unity. These sampling-based algorithms did not previously assert that the answer was optimum. RRTs are not asymptotically ideal because the expansion of the current state graph is biased. By incorporating gradual rewiring of the graph, RRT* gets around this. Not only are new states added to a tree, but they are also taken into account as potential parents for surrounding states that already exist in the tree. By asymptotically determining the best routes from the starting state to each state in the issue area, this method with uniform global sampling asymptotically discovers the best solution to the planning problem. This contradicts the single-query nature of them and becomes costly for large size. Informed RRT* focused optimal planning problem as it relates to the minimization of path length in Rn by restricting the exploration space to an ellipsoidal subset of the planning space. After a first solution is discovered, informed RRT* continues to operate like RRT* but only samples from the subset of states determined by an acceptable heuristic to perhaps enhance the original solution. This subset automatically strikes a balance between exploitation and exploration and doesn't call for any further assumptions or adjustments (i.e., no more parameters). Heuristics may not always enhance the search, but their prevalence in real-world planning shows its potential. But in real world situations where the environment is dynamically changing and complex. In that case using just a global planner like Informed RRT* is not sufficient as we need to plan a path around the obstacle that hinders the global path obtained from the planner. To make this possible, we introduce a local planner along with the global Informed RRT* planner such that it retains the useful parts of the tree (the data structure storing the motion plan information) after a dynamic obstacle invalidates the solution path. We then employ two greedy heuristics to repair the solution instead of running the whole motion planning process from scratch. In the reminder paper we focus on some prior work in this field and then introduce the math behind Informed RRT* technique along with the local planner heuristics. Later we present the visual results obtained using it and lastly, we show the quantitative results on comparison between our algorithm and the normal RRT* algorithm.

II. Literature Review

By weighing the sample of X with a heuristic estimate of each state, heuristic-biased sampling seeks to enhance the likelihood of sampling X_f . By choosing states with a probability inversely proportional to their heuristic cost, Urmson and Simmons employ the Heuristically Guided RRT (hRRT) to enhance the quality of a conventional RRT. The usage of RRTs makes the solution virtually certainly suboptimal, despite the fact that the hRRT has been demonstrated to find better solutions than RRT. In their f-biasing method, we employ a two-stage procedure to produce an RRT* heuristic. To give a heuristic, the planning issue is first addressed as a crude abstraction, each separate state's cost. RRT* then chooses a discrete state at random and samples inside of it using a continuous uniform distribution. States that belong to the abstracted solution have the highest likelihood of selection due to the bias in the discrete sampling. This method offers a heuristic bias throughout the whole RRT* process, but it

retains a nonzero probability of choosing each state to take into consideration the discrete abstraction. States that can't make the existing solution better are thus still sampled.

Path biasing is used by Akgun and Stilman in their dual tree implementation of RRT*. The algorithm refines the current solution for a user-specified portion of its iterations after discovering an initial solution. This is accomplished by explicitly sampling from a state's Voronoi area after randomly choosing a state from the solution path. This raises the likelihood of enhancing the existing course at the price of investigating more homotropy classes. Additionally, their technique uses sample rejection while examining the state space. Path biasing and smoothing are combined in the RRT*-Smart method by Nasir et al. RRT*- Smart first smooths and shrinks the path to the smallest number of states after finding a solution, then uses these states as biases for more sampling. This increases the planner's complexity while maintaining the need for global sampling to prevent local optima. While the path smoothing immediately lowers the cost of the present solution, by lowering the number of bias points about which samples are drawn, it may also lower the likelihood of discovering an alternative homotropy class and further violates the RRT* assumption of uniform density.

Kim et al. use a visibility analysis to generate an initial bias in their Cloud RRT* algorithm. This bias is updated as a solution is found to further concentrate sampling near the path. The paper titled "A Novel RRT*-Based Algorithm for Motion Planning in Dynamic Environments" presents a new algorithm for motion planning in dynamic environments. The authors propose an extension of the Rapidly-exploring Random Tree (RRT*) algorithm to handle obstacles that are moving with time.

III. Analysis of the Ellipsodal Informed Subset

A. Equations

Euclidean distance is a valid heuristic for both components in issues that aim to reduce the path length in $\mathbb{R}^n$ (even when motion restrictions are included). The cost of the existing solution, c_{best}, may therefore be written in closed form in terms of this informed subset of states that could enhance the current solution, $X_{\hat{f}} \supseteq X_f$.

$$X_{\hat{f}} = \{x \in X \mid ||x_{start} - x||_2 + ||x - x_{goal}||_2 \leq c_{best}\}$$

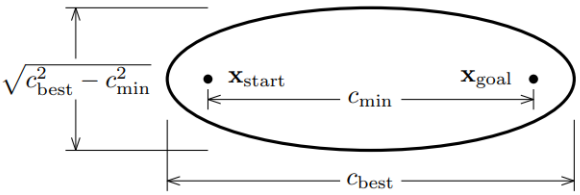


Fig. 1. The heuristic sampling domain, $\hat{X_f}$

It is an n-dimensional prolate hyper-spheroid's general equation (i.e., a unique hyper-ellipsoid). The transverse diameter is c_{best}, the conjugate diameters are $\sqrt{c_{best}^2 - c_{min}^2}$

B. Direct Sampling of an ellipsoidal subset

Uniformly distributed samples in a hyper-ellipsoid, $x_{ellipse} \sim U(X_{ellipse})$, can be generated by transforming uniformly distributed samples from the unit n-ball, $x_{ball} \sim U(X_{ball})$.

$$x_{ellipse} = Lx_{ball} + x - centre$$

where $x_{centre} = (x_{f1} + x_{f2})/2$ is the centre of the hyper-ellipsoid in terms of its two focal points, x_{f1} and x_{f2}, and $X_{ball} = \{x \in X \mid ||x||2 \leq 1\}$. This transformation can be calculated by Cholesky decomposition of the hyper-ellipsoid matrix, $S \in \mathbb{R}^{nxn}$

$$LL^T \equiv S$$

where

$$(x - x_{centre})^T S(x - x_{centre}) = 1$$

with S having eigenvectors corresponding to the axes of the hyper-ellipsoid, $\{ai\}$, and eigenvalues corresponding to the squares of its radii, r_i^2. The transformation, L, maintains the uniform distribution in $X_{ellipse}$

For prolate hyperspheroids, such as Xfb, the transformation can be calculated from just the transverse axis and the radii. The hyperellipsoid matrix in a coordinate system aligned with the transverse axis is the diagonal matrix

$$S = diag\left[\frac{c_{best}^2}{4}, \frac{c_{best}^2 - c_{min}^2}{4}, ..., \frac{c_{best}^2 - c_{min}^2}{4}\right]$$

with a resulting decomposition of

$$L = dia\left[\frac{c_{best}}{2}, \frac{\sqrt{c_{best}^2 - c_{min}^2}}{2}, ..., \frac{\sqrt{c_{best}^2 - c_{min}^2}}{2}\right]$$

where $diag\{\cdot\}$ denotes a diagonal matrix. The rotation from the hyper-ellipsoid frame to the world frame, $C \in SO(n)$, can be solved directly as a general Wahba problem. It has been shown that a valid solution can be found even when the problem is underspecified. The rotation matrix is given by

$$C = Udiag\{1, ..., 1, det(U)det(V)\}V^T$$

where $det(\dot{})$ is the matrix determinant and $U \in \mathbb{R}^{nn}$ and $V \in \mathbb{R}^{nn}$ are unitary matrices such that $U\sum V^T \equiv M$ via singular value decomposition. The matrix M is given by the outer product of the transverse axis in the world frame, a_1,and the first column of the identity matrix. 1_1

$$M = a_1 1_1^T$$

where

$$a_1 = \frac{(x_{goal} - x_{start}}{||x_{goal} - x_{start}||_2}$$

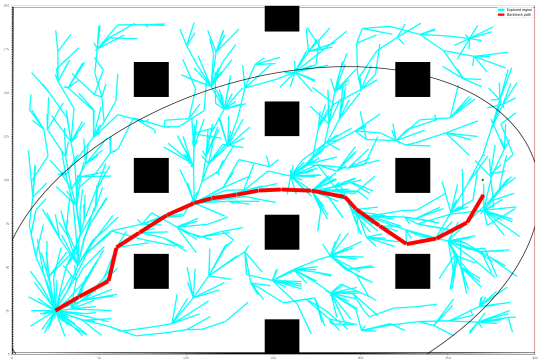
A state uniformly distributed in the informed subset, $x_{\hat{f}} \sim U(X_{\hat{f}})$, can thus be calculated from a sample drawn uniformly

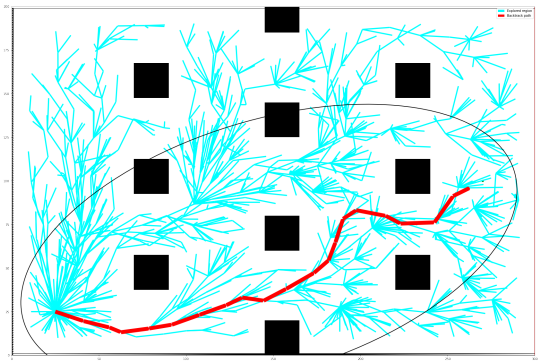
from a unit n-ball, $x_{ball} \sim U(X_{ball})$, through a transformation, rotation, and translation

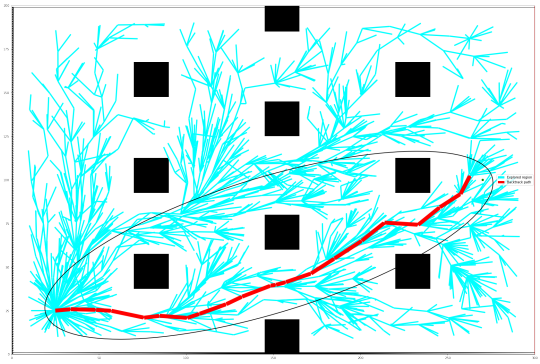
$$x_{\hat{f}} = CLx_{balll} + x_{centre}$$

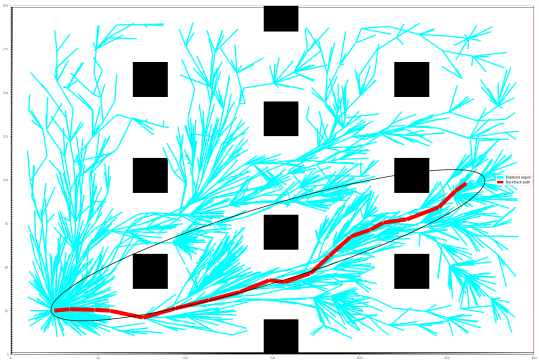
This procedure is presented in algorithm.

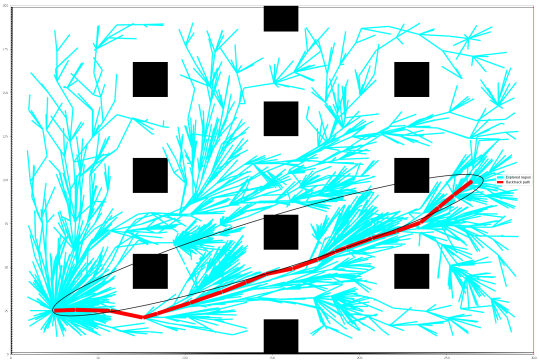
Implementation of Informed RRT* in our custom map











PSEUDO CODE OF OUR ALGORITHM

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

$V \leftarrow \{\mathbf{x}_{\text{start}}\}$;
$E \leftarrow \emptyset$;
$X_{\text{soln}} \leftarrow \emptyset$;
$\mathcal{T} = (V, E)$;
for iteration $= 1 \dots N$ **do**
  $c_{\text{best}} \leftarrow \min_{\mathbf{x}_{\text{soln}} \in X_{\text{soln}}} \{\texttt{Cost}(\mathbf{x}_{\text{soln}})\}$;
  $\mathbf{x}_{\text{rand}} \leftarrow \texttt{Sample}(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}, c_{\text{best}})$;
  $\mathbf{x}_{\text{nearest}} \leftarrow \texttt{Nearest}(\mathcal{T}, \mathbf{x}_{\text{rand}})$;
  $\mathbf{x}_{\text{new}} \leftarrow \texttt{Steer}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{rand}})$;
  if $\texttt{CollisionFree}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$ **then**
    $V \leftarrow \cup\{\mathbf{x}_{\text{new}}\}$;
    $X_{\text{near}} \leftarrow \texttt{Near}(\mathcal{T}, \mathbf{x}_{\text{new}}, r_{\text{RRT}^*})$;
    $\mathbf{x}_{\min} \leftarrow \mathbf{x}_{\text{nearest}}$;
    $c_{\min} \leftarrow \texttt{Cost}(\mathbf{x}_{\min}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{nearest}}, \mathbf{x}_{\text{new}})$;
    for $\forall \mathbf{x}_{\text{near}} \in X_{\text{near}}$ **do**
      $c_{\text{new}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{near}}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})$;
      if $c_{\text{new}} < c_{\min}$ **then**
        if $\texttt{CollisionFree}(\mathbf{x}_{\text{near}}, \mathbf{x}_{\text{new}})$ **then**
          $\mathbf{x}_{\min} \leftarrow \mathbf{x}_{\text{near}}$;
          $c_{\min} \leftarrow c_{\text{new}}$;
    $E \leftarrow E \cup \{(\mathbf{x}_{\min}, \mathbf{x}_{\text{new}})\}$;
    for $\forall \mathbf{x}_{\text{near}} \in X_{\text{near}}$ **do**
      $c_{\text{near}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{near}})$;
      $c_{\text{new}} \leftarrow \texttt{Cost}(\mathbf{x}_{\text{new}}) + c \cdot \texttt{Line}(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})$;
      if $c_{\text{new}} < c_{\text{near}}$ **then**
        if $\texttt{CollisionFree}(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})$ **then**
          $\mathbf{x}_{\text{parent}} \leftarrow \texttt{Parent}(\mathbf{x}_{\text{near}})$;
          $E \leftarrow E \setminus \{(\mathbf{x}_{\text{parent}}, \mathbf{x}_{\text{near}})\}$;
          $E \leftarrow E \cup \{(\mathbf{x}_{\text{new}}, \mathbf{x}_{\text{near}})\}$;
    if $\texttt{InGoalRegion}(\mathbf{x}_{\text{new}})$ **then**
      $X_{\text{soln}} \leftarrow X_{\text{soln}} \cup \{\mathbf{x}_{\text{new}}\}$;

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

if $c_{\max} < \infty$ then
  $c_{\min} \leftarrow \left|\left|\mathbf{x}_{\text{goal}} - \mathbf{x}_{\text{start}}\right|\right|_2$;
  $\mathbf{x}_{\text{centre}} \leftarrow \left(\mathbf{x}_{\text{start}} + \mathbf{x}_{\text{goal}}\right)/2$;
  $\mathbf{C} \leftarrow$ RotationToWorldFrame $\left(\mathbf{x}_{\text{start}}, \mathbf{x}_{\text{goal}}\right)$;
  $r_1 \leftarrow c_{\max}/2$;
  $\{r_i\}_{i=2,\ldots,n} \leftarrow \left(\sqrt{c_{\max}^2 - c_{\min}^2}\right)/2$;
  $\mathbf{L} \leftarrow \text{diag}\{r_1, r_2, \ldots, r_n\}$;
  $\mathbf{x}_{\text{ball}} \leftarrow$ SampleUnitNBall;
  $\mathbf{x}_{\text{rand}} \leftarrow (\mathbf{C}\mathbf{L}\mathbf{x}_{\text{ball}} + \mathbf{x}_{\text{centre}}) \cap X$;
else
  $\mathbf{x}_{\text{rand}} \sim \mathcal{U}(X)$;
return $\mathbf{x}_{\text{rand}}$;

Algorithm 3 $p_{min} \leftarrow$ ChooseParent$(P_{near}, p_{nearest}, p_{new})$

$p_{min} \leftarrow p_{nearest}$
$c_{min} \leftarrow$ Cost $(p_{nearest})$ + $c(p_{new}, p_{nearest})$
for $p_{near} \in P_{near}$ **do**
  if ObstacleFree (p_{near}, p_{new}) **then**
    $c' =$ Cost (p_{near}) + $c(p_{new}, p_{near})$
    if $c' < c_{min}$ **then**
      $p_{min} \leftarrow p_{near}$
      $c_{min} \leftarrow c'$
    end if
  end if
end for
return p_{min}

$\tau \leftarrow$ RRT*FN (p_{init}) {Grow phase}
$p_{current} \leftarrow p_{init}$
$\sigma \leftarrow$ SolutionPath $(\tau, p_{current})$
InitMovement ()
while $p_{current} \neq p_{goal}$ **do**
  $D \leftarrow$ UpdateObtacles ()
  if DetectCollision $(\sigma, p_{current})$ **then**
    StopMovement ()
    $\tau \leftarrow$ SelectBranch $(p_{current}, \tau)$
    $p_{separate} \leftarrow$ ValidPath (σ)
    ReconnectFailed $\leftarrow$ true
    $P_{near} \leftarrow$ Near$(\tau, p_{separate})$
    for $p_{near} \in P_{near}$ **do**
      if ObstacleFree $(p_{near}, p_{separate})$ **then**
        $\tau \leftarrow$ Reconnect$(p_{near}, p_{separate}, \tau)$
        ReconnectFailed $\leftarrow$ false
        break
      end if
    end for
    if ReconnectFailed = true **then**
      $\tau \leftarrow$ Regrow $(\tau, p_{separate},$ SetBias $(\sigma_{separate}))$
    end if
    $\sigma \leftarrow$ SolutionPath $(\tau, p_{current})$
    ResumeMovement ()
  end if
  $p_{current} \leftarrow$ NextNode (σ)
end while

IV. Results

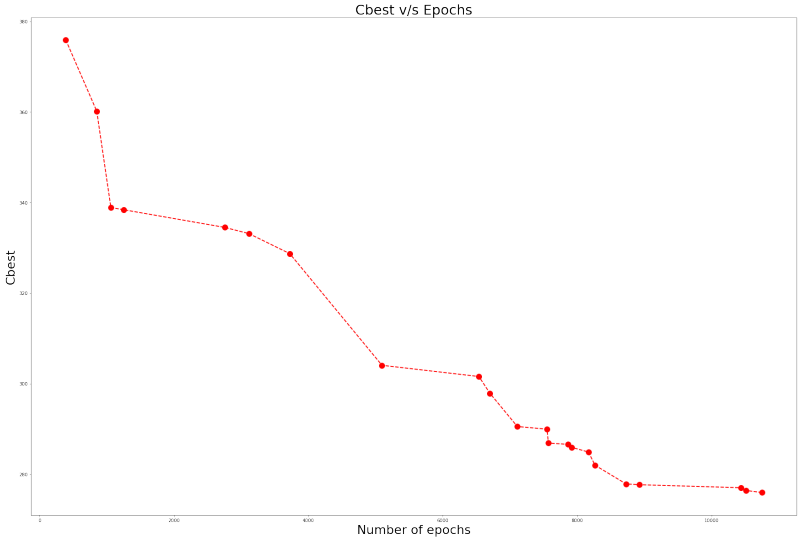


Fig. 2. C_{best} v/s Epochs

Below is the algorithm implementation of short-range code in a custom map.

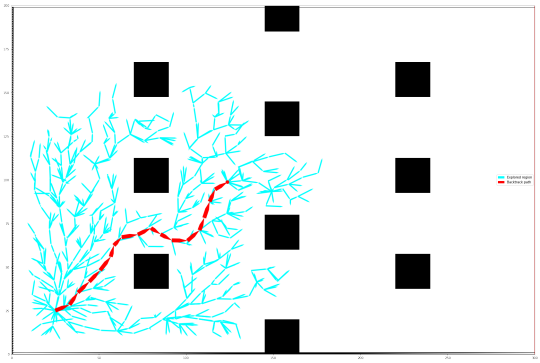


Fig. 3. initial path

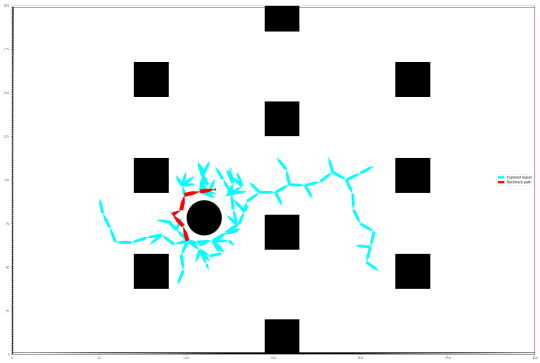


Fig. 4. path around obstacle

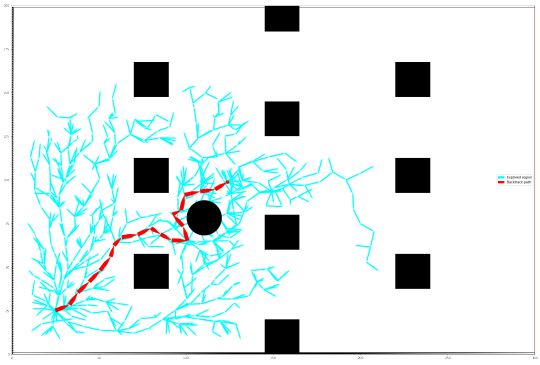


Fig. 5. final path

Below is the algorithm implementation of long-range code in a custom map.

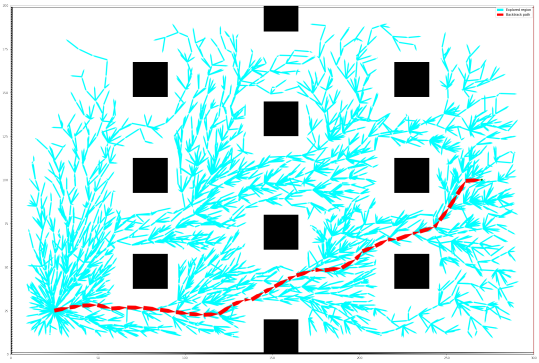


Fig. 6. initial path

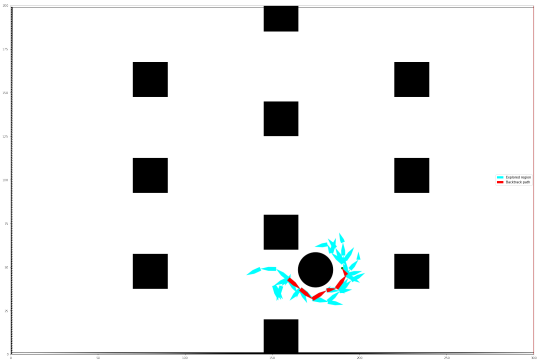


Fig. 7. path around obstacle

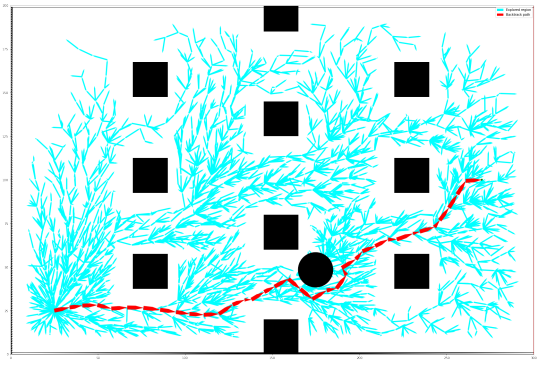


Fig. 8. final path

TABLE I
TIME COMPARISION

	Short range goal	Long range goal
Time Taken by Dynamic Informed RRT*	33.11 sec	65.95 sec
Time Taken by RRT* (Without obstacle)	5.9 sec	7.8 sec

TABLE II
PATH COST COMPARISION

	Short range goal	Long range goal
Path cost by Dynamic Informed RRT*	80	273
Path cost by RRT* (Without obstacle)	146	398

V. CONCLUSION

From the experiment, we can conclude that the Informed RRT* sampling-based approach is not a viable option in case of a dynamically changing environment. We can use it in combination with a local planner that only explores the region around the obstacle keeping the past information of the global path thus making it possible to use the Informed RRT* in such situations thus guarantying an optimal path in all situations with some expense of time compared to normal RRT*. The tradeoff between the time taken and the optimal path is better in the case of our algorithm when compared to RRT*. Further improvements can be done to this approach to make it more robust and general.

REFERENCES

[1] J. D. Gammell, S. S. Srinivasa and T. 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, Chicago, IL, USA, 2014, pp. 2997-3004, doi: 10.1109/IROS.2014.6942976.

[2] O. Adiyatov and H. A. Varol, "A novel RRT*-based algorithm for motion planning in Dynamic environments," 2017 IEEE International Conference on Mechatronics and Automation (ICMA), Takamatsu, Japan, 2017, pp. 1416-1421, doi: 10.1109/ICMA.2017.8016024.

[3] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," IROS, 2: 1178–1183, 2003.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," IJRR, 30(7): 846–894, 2011.

[5] M. Otte and N. Correll, "C-FOREST: Parallel shortest path planning with superlinear speedup," TRO, 29(3): 798–806, Jun. 2013.

[6] S. Kiesel, E. Burns, and W. Ruml, "Abstraction-guided sampling for motion planning," SoCS, 2012.

[7] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation.

[8] D. Kim, J. Lee, and S. Yoon, "Cloud RRT*: Sampling cloud based RRT*," ICRA, 2014.

[9] R. Alterovitz, S. Patil, and A. Derbakova, "Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning," ICRA, 3706–3712, 2011.

[10] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," IROS, 2640–2645, 2011.

[11] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation of RRT*".