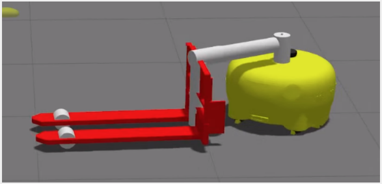
Motion Planning for Autonomous Vehicles

Path planning

Geesara Kulathunga

March 17, 2023



Path planning

Contents

- **Explicit representation** of **configuration space** is **not necessary**;
- , however, given the robot's position, collision detection should be possible
- **Single-query** and **multi-query-based** techniques are available
- **Completeness**: able to find a path in a bounded time, **Probabilistically complete**: if a solution exists, the planner will eventually find it, with no constraints on the time limit.

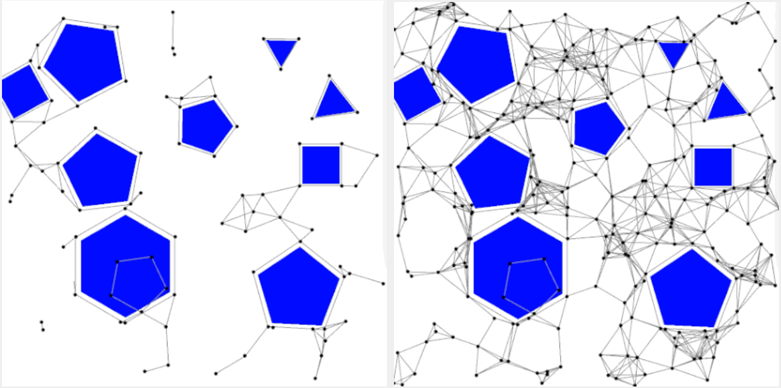
Probabilistic Road Map (PRM)

Uses graph structure

- The **learning phase** First, **generate n number of points** in the configuration space and discard points that lie in the obstacle zones. Second, **connect to the nearest points** and get only the **obstacle-free segments** while discarding the segments that are not collision-free, namely **roadmap**. In **Lazy RoadMap**, does not check the collision checking
- The **quarrying phase** Use the constructed roadmap to find a path between the start and goal node using any graph-based path-searching algorithm, e.g., A*, JPS

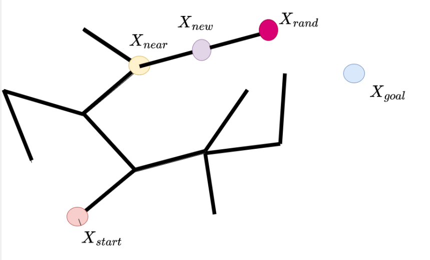
Probabilistic Road Map (PRM)

PRM is **probabilistically complete**,

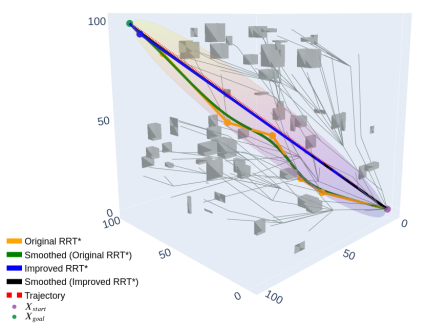


however, required to complete **2 point boundary value problem**
https://en.wikipedia.org/wiki/Probabilistic_roadmap

Sampling-based techniques are well suited for working with **high-dimensional search spaces** due to its computationally efficiency



Construct a tree by generating X_{next} as part of the tree by executing **random control**



Example of such a tree

Kulathunga, G., Devitt, D., Fedorenko, R., Klimchik, A. (2021). Path planning followed by kinodynamic smoothing for multirotor aerial vehicles (mavs). Russian Journal of Nonlinear Dynamics, 17(4), 491-505.

Rapidly-exploring Random Tree

Algorithm 7 RRT algorithm

procedure RRT($M, X_{start}, X_{goal}, dist$) ▷ The map M, starting node x_{start}, goal node x_{goal}, minimum allowable residual $dist$

  $V \leftarrow X_{start}; E \leftarrow 0;$

  for i=1,...,n **do**

    $x_{rand} \leftarrow getFreeSample(M)$

    $x_{near} \leftarrow getNearest(G = (V, E), x_{rand});$

    $x_{new} \leftarrow GetSteerPose(x_{near}, x_{rand});$

    if $ObstableFree(M, x_{near}, x_{new})$ **then**

      $V \leftarrow V \cup \{x_{near}\}$

      $ConnectShortestPath(x_{new}, x_{near})$

      if $|x_{new} - x_{goal}| < dist$ **then**

        return G = (V,E)

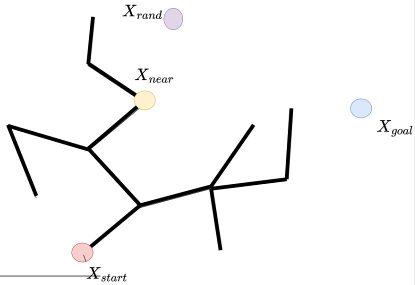
      end if

    end if

  end for

  return G = (V,E)

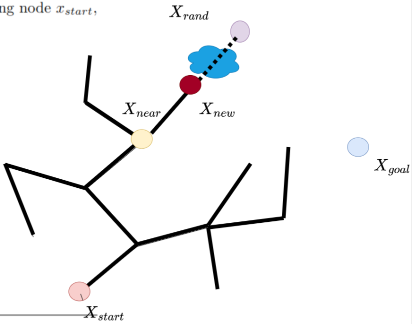
end procedure



Rapidly-exploring Random Tree

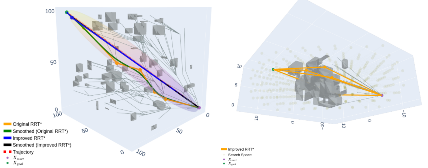
Algorithm 7 RRT algorithm

procedure RRT$(M, X_{start}, X_{goal}, dist)$ ▷ The map M, starting node x_{start}, goal node x_{goal}, minimum allowable residual $dist$
    $V \leftarrow X_{start}; E \leftarrow 0;$
    for i=1,...,n **do**
        $x_{rand \leftarrow getFreeSample(M)}$
        $x_{near} \leftarrow getNearest(G = (V, E), x_{rand});$
        $x_{new} \leftarrow GetSteerPose(x_{near}, x_{rand});$
        if $ObstableFree(M, x_{near}, x_{new})$ **then**
            $V \leftarrow V \cup \{x_{near}\}$
            $ConnectShortestPath(x_{new}, x_{near})$
            if $|x_{new} - x_{goal}| < dist$ **then**
                return G = (V,E)
            end if
        end if
    end for
    return G = (V,E)
end procedure



Algorithm 7 RRT algorithm

procedure RRT($M, X_{start}, X_{goal}, dist$) ▷ The map M, starting node x_{start}, goal node x_{goal}, minimum allowable residual $dist$
  $V \leftarrow X_{start}$; $E \leftarrow 0$;
  for i=1,...,n **do**
    $x_{rand} \leftarrow getFreeSample(M)$
    $x_{nearest} \leftarrow getNearest(G = (V, E), x_{rand})$;
    $x_{new} \leftarrow GetSteerPose(x_{nearest}, x_{rand})$;
    $E_i \leftarrow Edge(x_{new}, x_{nearest})$;
    if $ObstableFree(M, x_{nearest}, x_{new})$ **then**
      $X_{near} \leftarrow GetNearByVertices(x_{new})$
      $V \leftarrow V \cup \{x_{near}\}$
      $ConnectShortestPath(x_{new}, x_{near})$
      if $|x_{new} - x_{goal}| < dist$ **then**
        return G = (V,E)
      end if
    end if
  end for
  return G = (V,E)
end procedure

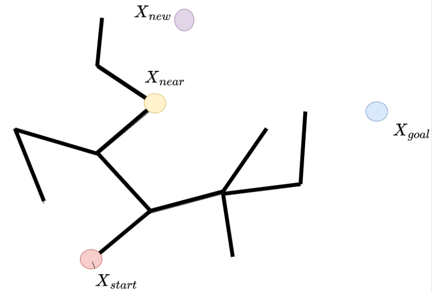


- Pro:
  - Convergence rate towards the x_{goal} is higher than PRM
  - Probabilistically complete
- Cons:
  - **Does not** provide an **optimal solution**
  - Not efficient: **Kd-tree** can be used to construct the graph G, **Bidirectional RRT** (grow a tree from both start and goal locations and path connect when two paths are connected)
  - **Sampling** in the **whole space** makes **no sense**

Algorithm 8 RRTStar algorithm

procedure RRTS($M, X_{start}, X_{goal}, dist$) ▷ The map M, starting node x_{start}, goal node x_{goal}, minimum allowable residual $dist$

    $V \leftarrow X_{start}; E \leftarrow 0;$
    for i=1,...,n **do**
        $x_{rand} \leftarrow getFreeSample(M)$
        $x_{near} \leftarrow getNearest(G = (V, E), x_{rand});$
        $x_{new} \leftarrow GetSteerPose(x_{near}, x_{rand});$
        if $ObstableFree(M, x_{nearest}, x_{new})$ **then**
            $X_{near} \leftarrow getNearByVertices(x_{new})$
            $x_{min} \leftarrow getParent(x_{new}, X_{near}, x_{near})$
            $V \leftarrow V \cup \{x_{min}\}$
            $ConnectShortestPath(x_{new}, x_{min})$
            if $IsEdge(x_{new})$ **then**
                $Rewine()$
            end if
            if $|x_{new} - x_{goal}| < dist$ **then**
                return G = (V,E)
            end if
        end if
    end for
    return G = (V,E)
end procedure

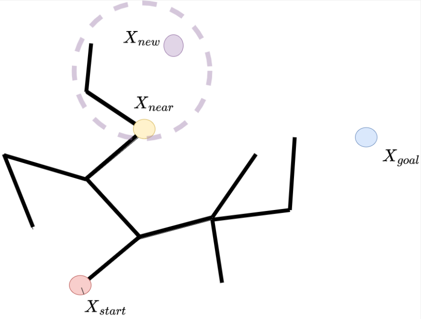


This step is similar to RRT

Rapidly-exploring Random Tree*

Algorithm 8 RRTStar algorithm

procedure RRTS($M, X_{start}, X_{goal}, dist$) ▷ The map M, starting node x_{start}, goal node x_{goal}, minimum allowable residual $dist$
  $V \leftarrow X_{start}; E \leftarrow 0;$
  for i=1,...,n **do**
    $x_{rand} \leftarrow getFreeSample(M)$
    $x_{near} \leftarrow getNearest(G = (V, E), x_{rand});$
    $x_{new} \leftarrow GetSteerPose(x_{near}, x_{rand});$
    if $ObstableFree(M, x_{nearest}, x_{new})$ **then**
      $X_{near} \leftarrow getNearByVertices(x_{new})$
      $x_{min} \leftarrow getParent(x_{new}, X_{near}, x_{near})$
      $V \leftarrow V \cup \{x_{min}\}$
      $ConnectShortestPath(x_{new}, x_{min})$
      if $IsEdge(x_{new})$ **then**
        $Rewine()$
      end if
      if $|x_{new} - x_{goal}| < dist$ **then**
        return G = (V,E)
      end if
    end if
  end for
  return G = (V,E)
end procedure

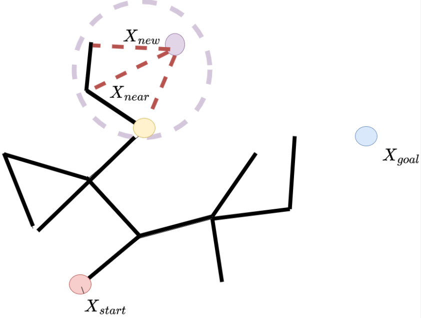


The **number of selected near vertices** can be varied by selecting different **radius** for searching

Rapidly-exploring Random Tree*

Algorithm 8 RRTStar algorithm

procedure RRTS($M, X_{start}, X_{goal}, dist$) ▷ The map M, starting node x_{start}, goal node x_{goal}, minimum allowable residual $dist$
  $V \leftarrow X_{start}; E \leftarrow 0;$
  for i=1,...,n **do**
    $x_{rand} \leftarrow getFreeSample(M)$
    $x_{near} \leftarrow getNearest(G = (V, E), x_{rand});$
    $x_{new} \leftarrow GetSteerPose(x_{near}, x_{rand});$
    if $ObstableFree(M, x_{nearest}, x_{new})$ **then**
      $X_{near} \leftarrow getNearByVertices(x_{new})$
      $x_{min} \leftarrow getParent(x_{new}, X_{near}, x_{near})$
      $V \leftarrow V \cup \{x_{min}\}$
      $ConnectShortestPath(x_{new}, x_{min})$
      if $IsEdge(x_{new})$ **then**
        $Rewine()$
      end if
      if $|x_{new} - x_{goal}| < dist$ **then**
        return G = (V,E)
      end if
    end if
  end for
  return G = (V,E)
end procedure

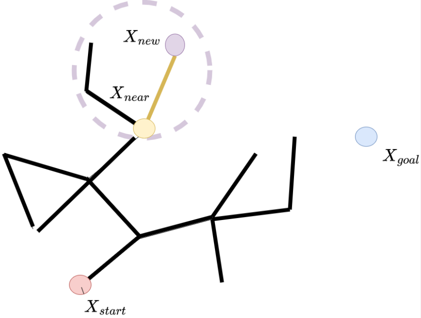


The **parent** can be selected based on defined criteria, e.g., the shortest distance from the start position

Rapidly-exploring Random Tree*

Algorithm 8 RRTStar algorithm

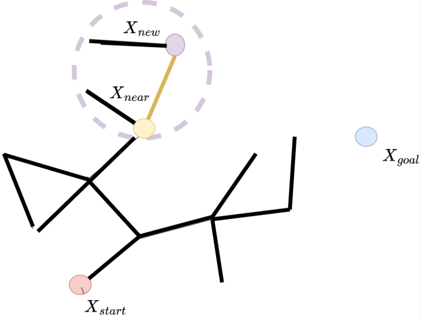
```
procedure RRTS(M, X_start, X_goal, dist)    ▷ The map M, starting node x_start, goal node x_goal, minimum allowable residual dist
    V ← X_start; E ← 0;
    for i=1,...,n do
        x_rand ← getFreeSample(M)
        x_near ← getNearest(G = (V, E), x_rand);
        x_new ← GetSteerPose(x_near, x_rand);
        if ObstableFree(M, x_nearest, x_new) then
            X_near ← getNearByVertices(x_new)
            x_min ← getParent(x_new, X_near, x_near)
            V ← V ∪ {x_min}
            ConnectShortestPath(x_new, x_min)
            if IsEdge(x_new) then
                Rewine()
            end if
            if |x_new − x_goal| < dist then
                return G = (V,E)
            end if
        end if
    end for
    return G = (V,E)
end procedure
```



Rapidly-exploring Random Tree*

Algorithm 8 RRTStar algorithm

procedure RRTS($M, X_{start}, X_{goal}, dist$) ▷ The map M, starting node x_{start}, goal node x_{goal}, minimum allowable residual $dist$
  $V \leftarrow X_{start}; E \leftarrow 0;$
  for i=1,...,n **do**
    $x_{rand} \leftarrow getFreeSample(M)$
    $x_{near} \leftarrow getNearest(G = (V, E), x_{rand});$
    $x_{new} \leftarrow GetSteerPose(x_{near}, x_{rand});$
    if $ObstableFree(M, x_{nearest}, x_{new})$ **then**
      $X_{near} \leftarrow getNearByVertices(x_{new})$
      $x_{min} \leftarrow getParent(x_{new}, X_{near}, x_{near})$
      $V \leftarrow V \cup \{x_{min}\}$
      $ConnectShortestPath(x_{new}, x_{min})$
      if $IsEdge(x_{new})$ **then**
        $Rewine()$
      end if
      if $|x_{new} - x_{goal}| < dist$ **then**
        return G = (V,E)
      end if
    end if
  end for
  return G = (V,E)
end procedure



Cons of RRT*

- **Problem 01**: if the search space is bigger, the number of attempts to find the connection between X_{start} and X_{goal} is higher

Kulathunga, G., Fedorenko, R., Kopylov, S., Klimehik, A. (2020, July). Real-time long range trajectory replanning for mavs in the presence of dynamic obstacles. In 2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS) (pp. 145-153). IEEE.

- **Problem 01**: if the search space is bigger, the number of attempts to find the connection between X_{start} and X_{goal} is higher
- **Solution:**

$$P(x \in X_{free}) \leqslant P(x \in X_{reduced}) = \frac{\lambda(X_{reduced})}{\lambda(X_{free})}$$
$$\frac{\lambda(X_{reduced})}{\lambda(X_{free})} = \frac{3}{4}\pi d^2 \frac{|X_{goal} - X_{start}|}{\lambda(X_{free})} \tag{1}$$

$\lambda(.)$ denotes the volume of the search space. $\frac{\lambda(X_{reduced})}{\lambda(X_{free})}$ depicts the selecting a sample from $X_{reduced}$ always has higher probability

Kulathunga, G., Fedorenko, R., Kopylov, S., Klimehik, A. (2020, July). Real-time long range trajectory replanning for mavs in the presence of dynamic obstacles. In 2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS) (pp. 145-153). IEEE.

Cons of RRT*: how to generate the random points within $X_{reduced}$?

- **Three-dimensional Gaussian bell** can be represented as an ellipsoid where the size and orientation of the ellipsoid are described by the covariance matrix Σ

Cons of RRT*: how to generate the random points within $X_{reduced}$?

- **Three-dimensional Gaussian bell** can be represented as an ellipsoid where the size and orientation of the ellipsoid are described by the covariance matrix Σ
- Thus, defining the $X_{reduced}$ can be seen as an analogy for defining a three-dimensional Gaussian bell, with R standing for rotation matrix and variances as a diagonal matrix $diag(\sigma_{xx}, \sigma_{yy}, \sigma_{zz})$

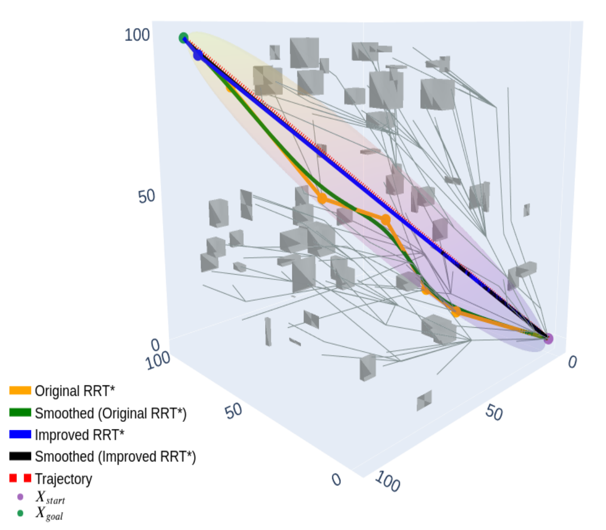
Cons of RRT*: how to generate the random points within $X_{reduced}$?

- **Three-dimensional Gaussian bell** can be represented as an ellipsoid where the size and orientation of the ellipsoid are described by the covariance matrix Σ
- Thus, defining the $X_{reduced}$ can be seen as an analogy for defining a three-dimensional Gaussian bell, with R standing for rotation matrix and variances as a diagonal matrix $diag(\sigma_{xx}, \sigma_{yy}, \sigma_{zz})$
- The relationship between $R, diag(\sigma_{xx}, \sigma_{yy}, \sigma_{zz})$ and Σ

$$\Sigma = R\,\mathbf{diag}(\sigma_{xx}, \sigma_{yy}, \sigma_{zz})\,R^{\top} \tag{2}$$

where $\sigma_{xx} = |X_{goal} - X_{start}|$ and $\sigma_{yy} = \sigma_{zz} = d$. The rotation matrix R aligns **z** (0,0,1) to $\mathbf{X_{goal}} - \mathbf{X_{start}}$

Cons of RRT*: Applying Improved RRT* on $X_{reduced}$



Kulathunga, G., Devitt, D., Fedorenko, R., Klimchik, A. (2021). Path planning followed by kinodynamic smoothing for multirotor aerial vehicles (mavs). Russian Journal of Nonlinear Dynamics, 17(4), 491-505.

Cons of RRT*:

- **Problem 02**: we do not have any control over how random points are being generated within the $X_{reduced}$; Thus, it is necessary to have a technique for generating points such that it helps fast convergence of RRT*.

Cons of RRT*:

- **Problem 02**: we do not have any control over how random points are being generated within the $X_{reduced}$; Thus, it is necessary to have a technique for generating points such that it helps fast convergence of RRT*.
- **Solution:** We can generate points deterministic way within $X_{reduced}$ ensuring map constraints $X_{reduced}$ is defined as

$$(x - c_x)^2/(r_x)^2 + (y - c_y)^2/(r_y)^2 + (z - c_z)^2/(r_z)^2 = 1 \quad (3)$$

where c is the middle pose in between X_{start} and X_{goal} pose and $\mathbf{r} = \mathbf{x_{goal}} - \mathbf{x_{start}} =< r_x, r_y, r_z >$ is defined as follows:

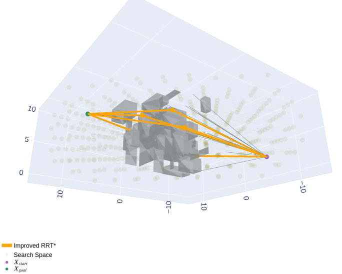
$$\mathbf{r}_x = max(x_{max}, \mathbf{r}_x),\ \mathbf{r}_y = max(y_{max}, \mathbf{r}_y)\ \mathbf{r}_z = max(z_{max}, \mathbf{r}_z) \quad (4)$$

Cons of RRT*:

- **Problem 03**: How do we decide which path is the optimal path?
- **Solution**

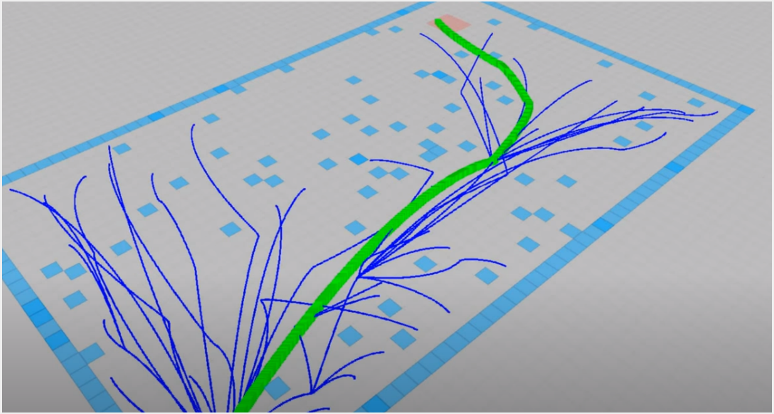
$$Cost = \|(\mathbf{P}_{\mathrm{M}} - \mathbf{P}_{\mathrm{goal}})\| + \sum_{m=1}^{M-1} \|(\mathbf{P}_{\mathrm{m}} - \mathbf{P}_{\mathrm{m+1}})\| \tag{5}$$

where $\mathbf{P}_{\mathrm{m}}$ depicts the *mth* waypoint of the selected path in which it consists of M number of waypoints.



Optimal Rapidly-exploring Random Tree*

By changing *getSteerPose*() function different optimality conditions can be achieved: e.g., **Hybrid RRT***, **Kinodynamic-RRT*, Informed RRT*, and Anytime-RRT***



https://www.youtube.com/watch?v=RB3g_GP0-dU