RECITATION 10

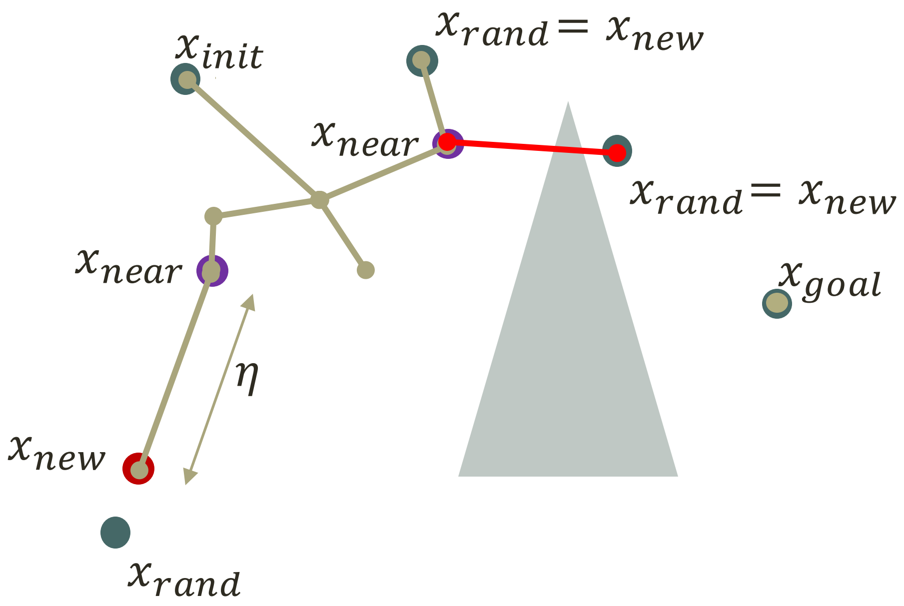
Michal Kleinbort

RRT: RAPIDLY-EXPLORING RANDOM TREE

Algorithm 1 RRT ($x_{\texttt{init}} := s, x_{\texttt{goal}} := t, n, \eta$)

$V = \{x_{\text{init}}\}$
for $j = 1$ to n **do**
  $x_{\text{rand}} \leftarrow$ SAMPLE-FREE( )
  $x_{\text{near}} \leftarrow$ NEAREST(x_{rand}, V)
  $x_{\text{new}} \leftarrow$ STEER($x_{\text{near}}, x_{\text{rand}}, \eta$)
  if COLLISION-FREE($x_{\text{near}}, x_{\text{new}}$) **then**
    $V = V \cup \{x_{\text{new}}\}$
    $E = E \cup \{(x_{\text{near}}, x_{\text{new}})\}$
return $G = (V, E)$

RRT: DEMONSTRATION



Algorithm 1 RRT ($x_{\texttt{init}} := s, x_{\texttt{goal}} := t, n, \eta$)

```
V = {x_init}
for j = 1 to n do
    x_rand ← SAMPLE-FREE( )
    x_near ← NEAREST(x_rand, V)
    x_new ← STEER(x_near, x_rand, η)
    if COLLISION-FREE(x_near, x_new) then
        V = V ∪ {x_new}
        E = E ∪ {(x_near, x_new)}
return G = (V, E)
```

RRT*

Algorithm 2 $\mathtt{RRT^*}\ (x_{\mathtt{init}} := s, x_{\mathtt{goal}} := t, n, r, \eta)$

$V = \{x_{\text{init}}\}$
for $j = 1$ to n **do**
  $x_{\text{rand}} \leftarrow$ SAMPLE-FREE( )
  $x_{\text{near}} \leftarrow$ NEAREST(x_{rand}, V)
  $x_{\text{new}} \leftarrow$ STEER$(x_{\text{near}}, x_{\text{rand}}, \eta)$
  if COLLISION-FREE$(x_{\text{near}}, x_{\text{new}})$ **then**
    $X_{\text{near}} =$ NEAR$(x_{\text{new}}, V, \min\{r(|V|), \eta\})$
    $V = V \cup \{x_{\text{new}}\}$
    $x_{\text{min}} = x_{\text{near}}$
    $c_{\text{min}} =$ COST$(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\|$
    for $x_{\text{near}} \in X_{\text{near}}$ **do**
      if COLLISION-FREE$(x_{\text{near}}, x_{\text{new}})$ **then**
        if COST$(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\| < c_{\text{min}}$ **then**
          $x_{\text{min}} = x_{\text{near}}$
          $c_{\text{min}} =$ COST$(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\|$
    $E = E \cup \{(x_{\text{min}}, x_{\text{new}})\}$
    for $x_{\text{near}} \in X_{\text{near}}$ **do**
      if COLLISION-FREE$(x_{\text{new}}, x_{\text{near}})$ **then**
        if COST$(x_{\text{new}}) + \|x_{\text{near}} - x_{\text{new}}\| <$ COST(x_{near}) **then**
          $x_{\text{parent}} =$ PARENT(x_{near})
          $E = E \cup \{(x_{\text{new}}, x_{\text{near}})\} \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$
return $G = (V, E)$

[Karaman and Frazzoli, 2011]

RRT*

Algorithm 2 RRT* $(x_{\texttt{init}} := s, x_{\texttt{goal}} := t, n, r, \eta)$

$V = \{x_{\text{init}}\}$
for $j = 1$ to n **do**
  $x_{\text{rand}} \leftarrow$ SAMPLE-FREE( )
  $x_{\text{near}} \leftarrow$ NEAREST(x_{rand}, V)
  $x_{\text{new}} \leftarrow$ STEER$(x_{\text{near}}, x_{\text{rand}}, \eta)$
  if COLLISION-FREE$(x_{\text{near}}, x_{\text{new}})$ **then**
    $X_{\text{near}} =$ NEAR$(x_{\text{new}}, V, \min\{r(|V|), \eta\})$
    $V = V \cup \{x_{\text{new}}\}$
    $x_{\min} = x_{\text{near}}$
    $c_{\min} = \text{COST}(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\|$
    for $x_{\text{near}} \in X_{\text{near}}$ **do**
      if COLLISION-FREE$(x_{\text{near}}, x_{\text{new}})$ **then**
        if $\text{COST}(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\| < c_{\min}$ **then**
          $x_{\min} = x_{\text{near}}$
          $c_{\min} = \text{COST}(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\|$
    $E = E \cup \{(x_{\min}, x_{\text{new}})\}$
    for $x_{\text{near}} \in X_{\text{near}}$ **do**
      if COLLISION-FREE$(x_{\text{new}}, x_{\text{near}})$ **then**
        if $\text{COST}(x_{\text{new}}) + \|x_{\text{near}} - x_{\text{new}}\| < \text{COST}(x_{\text{near}})$
  then
          $x_{\text{parent}} = \text{PARENT}(x_{\text{near}})$
          $E = E \cup \{(x_{\text{new}}, x_{\text{near}})\} \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$
return $G = (V, E)$

Find best parent for x_{new} among its neighbors (within radius of r(n))

[Karaman and Frazzoli, 2011]

RRT*

Algorithm 2 RRT* $(x_{\texttt{init}} := s, x_{\texttt{goal}} := t, n, r, \eta)$

$V = \{x_{\text{init}}\}$
for $j = 1$ to n **do**
  $x_{\text{rand}} \leftarrow$ SAMPLE-FREE( )
  $x_{\text{near}} \leftarrow$ NEAREST(x_{rand}, V)
  $x_{\text{new}} \leftarrow$ STEER$(x_{\text{near}}, x_{\text{rand}}, \eta)$
  if COLLISION-FREE$(x_{\text{near}}, x_{\text{new}})$ **then**
    $X_{\text{near}} =$ NEAR$(x_{\text{new}}, V, \min\{r(|V|), \eta\})$
    $V = V \cup \{x_{\text{new}}\}$
    $x_{\min} = x_{\text{near}}$
    $c_{\min} = \text{COST}(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\|$
    for $x_{\text{near}} \in X_{\text{near}}$ **do**
      if COLLISION-FREE$(x_{\text{near}}, x_{\text{new}})$ **then**
        if $\text{COST}(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\| < c_{\min}$ **then**
          $x_{\min} = x_{\text{near}}$
          $c_{\min} = \text{COST}(x_{\text{near}}) + \|x_{\text{new}} - x_{\text{near}}\|$
    $E = E \cup \{(x_{\min}, x_{\text{new}})\}$
    for $x_{\text{near}} \in X_{\text{near}}$ **do**
      if COLLISION-FREE$(x_{\text{new}}, x_{\text{near}})$ **then**
        if $\text{COST}(x_{\text{new}}) + \|x_{\text{near}} - x_{\text{new}}\| < \text{COST}(x_{\text{near}})$ **then**
          $x_{\text{parent}} = \text{PARENT}(x_{\text{near}})$
          $E = E \cup \{(x_{\text{new}}, x_{\text{near}})\} \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$
return $G = (V, E)$

Rewiring: Set x_{new} as the parent of neighboring nodes (within radius of r(n)), if this improves their cost

[Karaman and Frazzoli, 2011]

ROBUSTLY FEASIBLE MOTION-PLANNING PROBLEM

Definition 2. Let $(\mathcal{F}, s, t)$ be a motion-planning problem. A path $\sigma \in \Sigma_{s,t}^{\mathcal{F}}$ is *robust* if there exists $\delta > 0$ such that $\mathcal{B}_\delta(\sigma) \subset \mathcal{F}$. We also say that $(\mathcal{F}, s, t)$ is *robustly feasible* if there exists such a robust path.

Definition 3. The *robust optimum* is defined as

$$c^* = \inf \left\{ c(\sigma) \middle| \sigma \in \Sigma_{s,t}^{\mathcal{F}} \text{ is robust} \right\}.$$

[Solovey et al, 2019]

ASYMPTOTIC OPTIMALITY OF RRT*

Theorem 1. *Suppose that* $(\mathcal{F}, s, t)$ *is robustly feasible, fix* $\eta > 0, \varepsilon \in (0,1), \theta \in (0, 1/4), \mu > 0$*, and define the radius of* RRT* *to be*

$$r(n) = \gamma \left(\frac{\log n}{n}\right)^{\frac{1}{d+1}}, \tag{2}$$

$$\gamma > (2+\theta)\left(\frac{(1+\varepsilon/4)c^*}{(d+1)\theta(1-\mu)} \cdot \frac{|\mathcal{F}|}{\zeta_d}\right)^{\frac{1}{d+1}}, \tag{3}$$

where ζ_d *is the volume of a unit d-dimensional hypersphere,* c^* *is the robust optimum. Then*

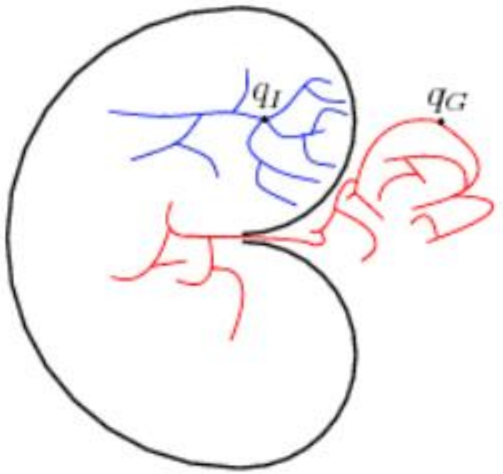
$$\lim_{n\to\infty} \Pr[c(\sigma_n) \leq (1+\varepsilon)c^*] = 1.$$

[Solovey et al, 2019]

GOAL BIAS

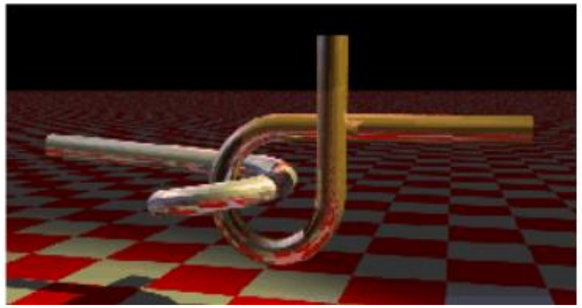
- In each iteration with probability p sample a random point x_{rand} and with probability $1 - p$ set x_{rand} to be a sample in the goal region.
- This forces RRT to occasionally attempt to make a connection to the goal
- If the bias is too strong, then RRT becomes too greedy
- If the bias is not strong enough, then there is no incentive to connect the tree to the goal region

BIDIRECTIONAL RRT



- Grow two trees T_s, T_g from start and goal
- In every iteration select one of the trees for expansion:
  - Balanced Bi-RRT: expand from the tree having less vertices
- Force the trees to meet
  - Every once in while grow them towards the same sample
- Balancing the trees could be helpful when one of the trees is having trouble exploring (balancing allows to put more energy on it)
- Bi-RRT* exists as well

(Figures from LaValle's book)



HRRT: HEURISTICALLY BIASING RRT GROWTH

- Prefer to extend lower cost paths heading towards the goal, while maintaining a reasonable bias towards exploration
- hRRT: the probability to select a node depends on both:
  - the size of its Voronoi region (a bias toward exploration) ,and
  - the quality of the path to that node (a bias toward exploiting known good parts of the space)
- More details in: [Urmson and Simmons, 2003]

HRRT: SELECT_NODE

```
SELECT_NODE(T)
do
        x_rand ← RANDOM_STATE();
        x_near ← NEAREST_NEIGHBOR(x, T);
        m_quality ←1- (x_near.cost –T.opt_cost ) /
                (T.max_cost-T.opt_cost);
        m_quality ← min ( m_quality, T.prob_floor );
        r ←RANDOM_VALUE();
 while ( r > m_quality);
return { x_rand, x_near };
```

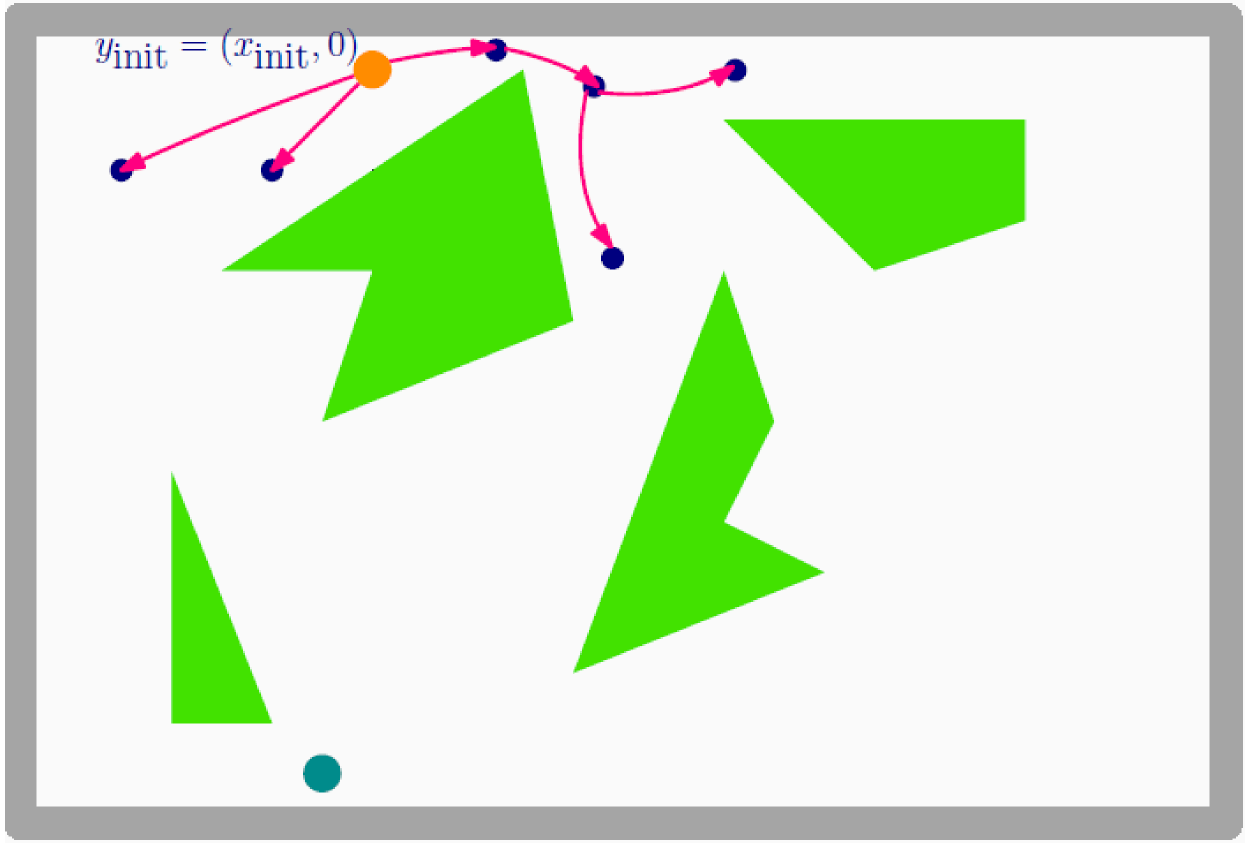
HEURISTIC GUIDED RRT*

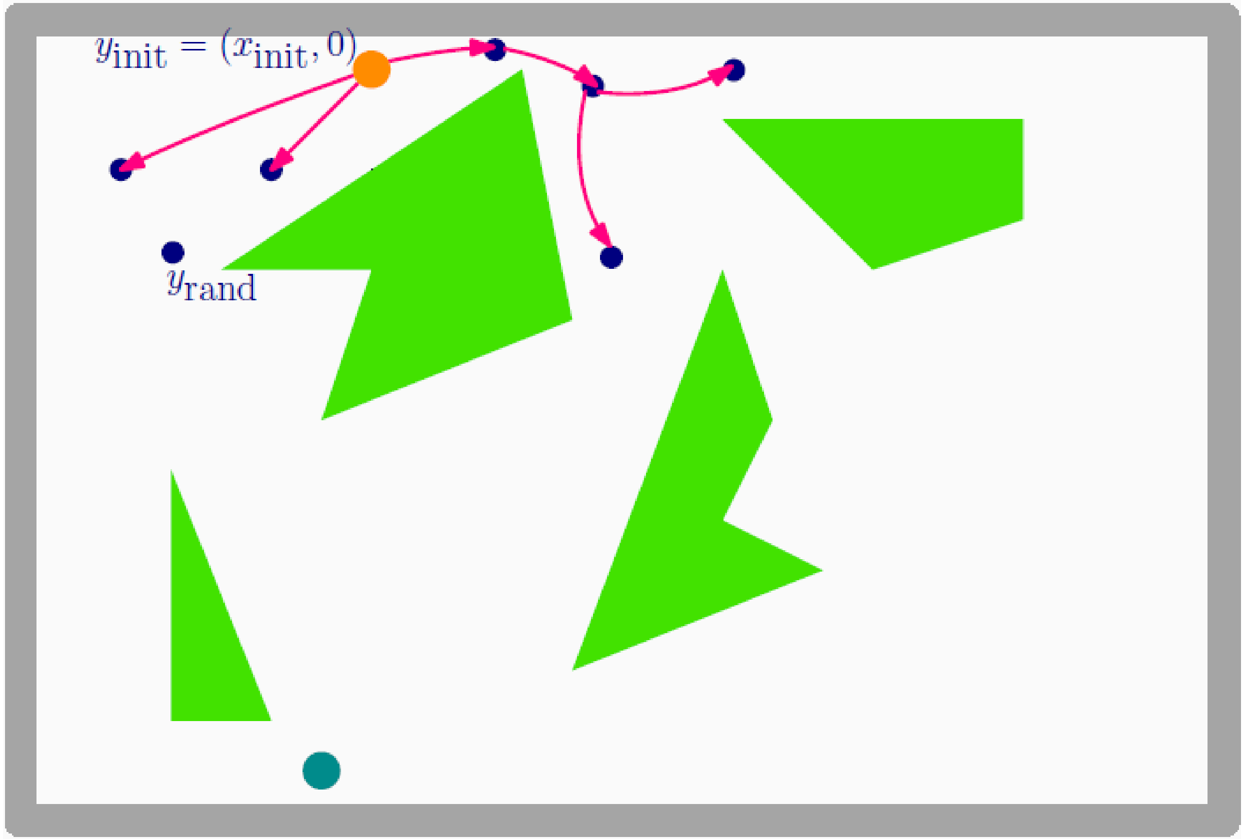
- Estimate the "cost to go" for every sample using Dijkstra on a grid of a specified resolution
- Maintain the leaf node x_{best} with least cost (cost-to-come) + heuristic cost (estimated cost-to-go)
- Sampling distribution is a Gaussian centered at x_{best}
- The implicit assumption: the solution of the reduced motion planning problem lies in the same proximity as the solution to the original problem
- The sampling could misguide the tree growth into "bad" regions
- More details in: [Vemula et al, 2014]

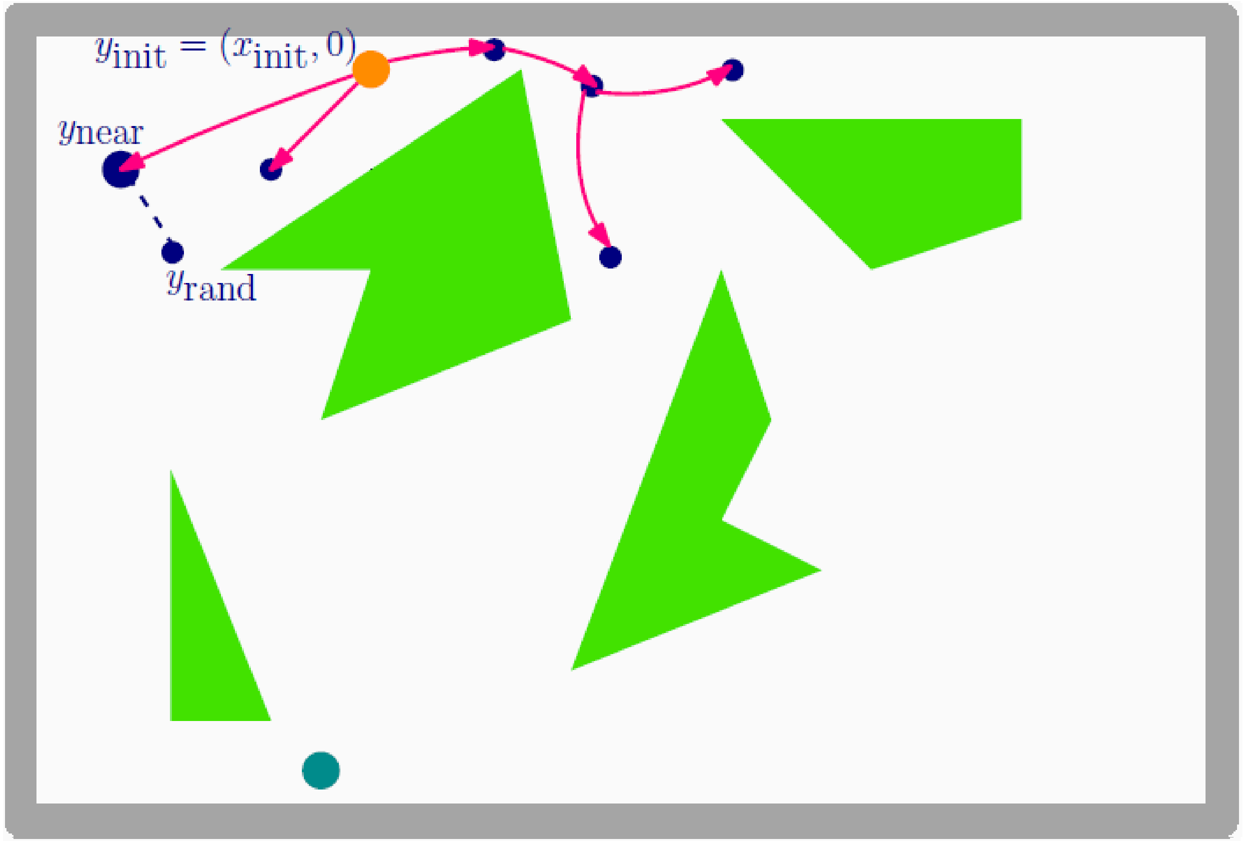
AO-RRT2: AO KINODYNAMIC PLANNER

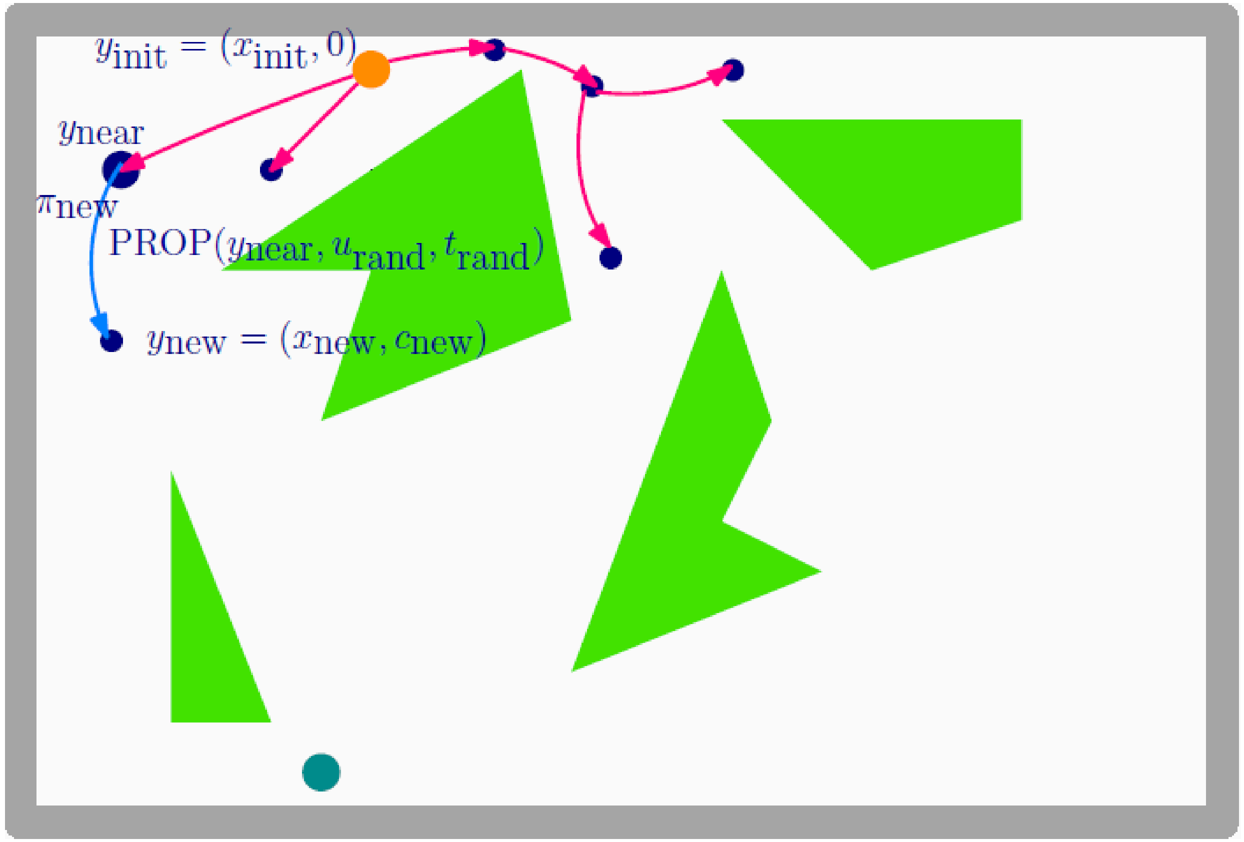
- Run RRT in State+cost space: a (d+1)-dimensional space, where every point $y \in X \times R_+$ is a pair (x, c) s.t. x is a state and $c \geq 0$ is the cost from x_{init} to x over the tree
- Running in this augmented space allows for "corrections" that are often not available in the vanilla RRT
- May prune nodes whose cost-to-come is higher than the best cost found so far from start to goal

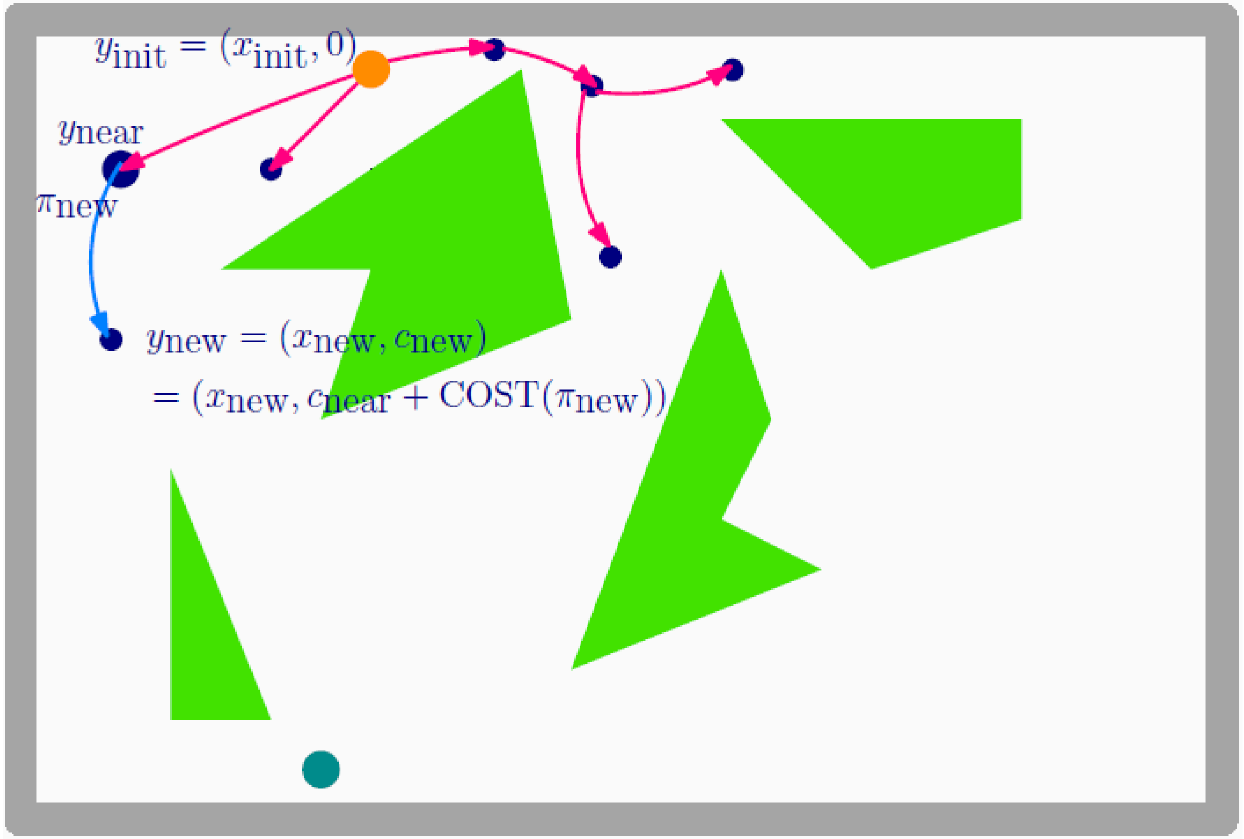
[Kleinbort et al, 2019]

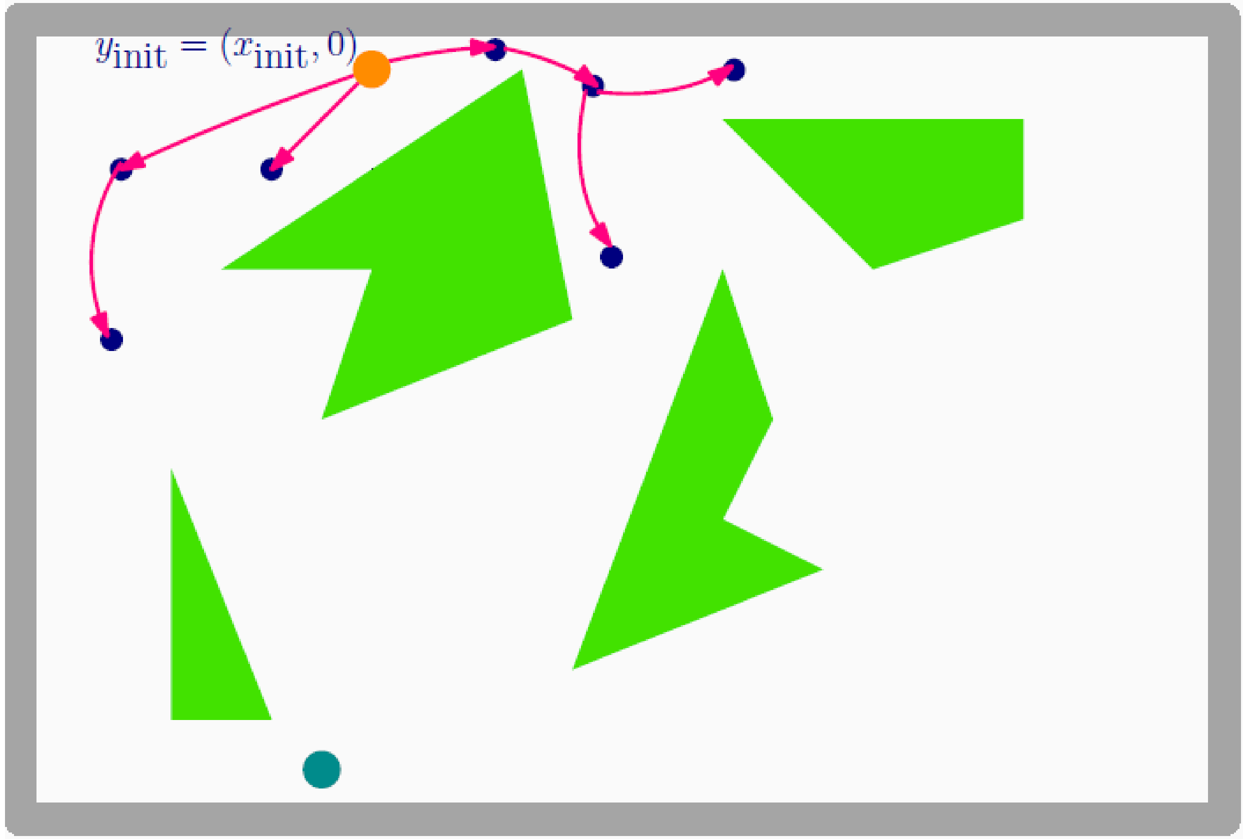

$y_{\text{init}} = (x_{\text{init}}, 0)$


$y_{\text{init}} = (x_{\text{init}}, 0)$
y_{rand}


$y_{\text{init}} = (x_{\text{init}}, 0)$
y_{near}
y_{rand}


$y_{init} = (x_{init}, 0)$
y_{near}
π_{new}
$PROP(y_{near}, u_{rand}, t_{rand})$
$y_{new} = (x_{new}, c_{new})$


$y_{\text{init}} = (x_{\text{init}}, 0)$
y_{near}
π_{new}
$y_{\text{new}} = (x_{\text{new}}, c_{\text{new}})$
$= (x_{\text{new}}, c_{\text{near}} + \text{COST}(\pi_{\text{new}}))$


$y_{\text{init}} = (x_{\text{init}}, 0)$

AO-RRT2: AO KINODYNAMIC PLANNER

Algorithm 1 $\texttt{AO} - \texttt{RRT2}(x_{\text{init}}, \mathcal{X}_{\text{goal}}, k, T_{\text{prop}}, \mathcal{U}, c_{\max})$

$y_{\text{init}} \leftarrow (x_{\text{init}}, 0); \mathcal{T}(\mathcal{Y}).\text{init}(y_{\text{init}}); y_{\min} = (\text{NULL}, \infty)$
for $i = 1$ **to** k **do**
  $x_{\text{rand}} \leftarrow \text{SAMPLE}(\mathcal{X})$ ▷ sample state
  $c_{\text{rand}} \leftarrow \text{SAMPLE}([0, c_{\max}])$ ▷ sample cost
  $t_{\text{rand}} \leftarrow \text{SAMPLE}([0, T_{\text{prop}}])$ ▷ sample duration
  $u_{\text{rand}} \leftarrow \text{SAMPLE}(\mathcal{U})$ ▷ sample control
  $y_{\text{near}} \leftarrow \text{NEAREST}(y_{\text{rand}} = (x_{\text{rand}}, c_{\text{rand}}), \mathcal{T}(\mathcal{Y}))$
  $(x_{\text{new}}, \pi_{\text{new}}) \leftarrow \text{PROPAGATE}(x(y_{\text{near}}), u_{\text{rand}}, t_{\text{rand}})$
  $c_{\text{new}} \leftarrow c(y_{\text{near}}) + \text{COST}(\pi_{\text{new}})$
  if $\text{COLLISION-FREE}(\pi_{\text{new}})$ **then**
    $\mathcal{T}(\mathcal{Y}).\text{add\_vertex}(y_{\text{new}} = (x_{\text{new}}, c_{\text{new}}))$
    $\mathcal{T}(\mathcal{Y}).\text{add\_edge}(y_{\text{near}}, y_{\text{new}}, \pi_{\text{new}})$
    if $x(y_{\text{new}}) \in \mathcal{X}_{\text{goal}}$ and $c(y_{\text{new}}) < c(y_{\min})$ **then**
      $y_{\min} \leftarrow y_{\text{new}}$
return $\text{TRACE-PATH}(\mathcal{T}(\mathcal{Y}), y_{\min})$