Motion Planning Networks

Ahmed Qureshi
PhD Candidate, Electrical and Computer Engineering
Contextual Robotics Institute
University of California, San Diego
qureshiahmed.github.io

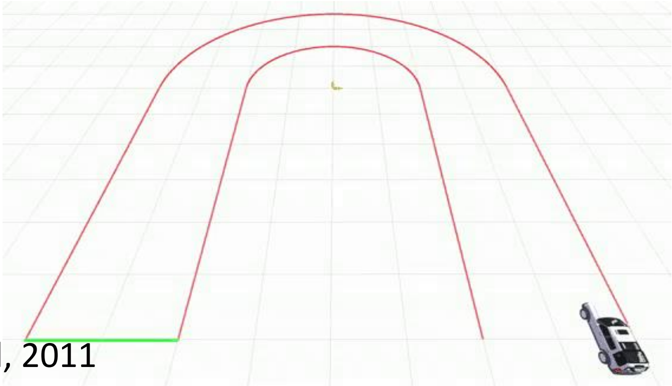
ARClab
advanced robotics and controls

UC San Diego
Jacobs School of Engineering

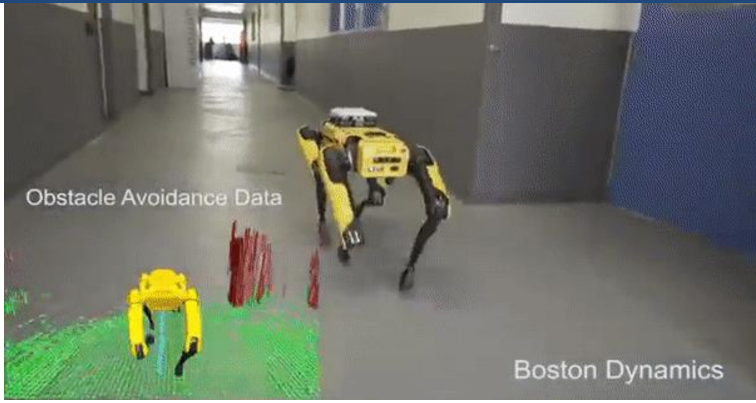
Motion Planning

Find a path that satisfies all constraints between the given start and goal configurations.

- Collision Avoidance
- Dynamics
- Kinematics (e.g., end-effector)

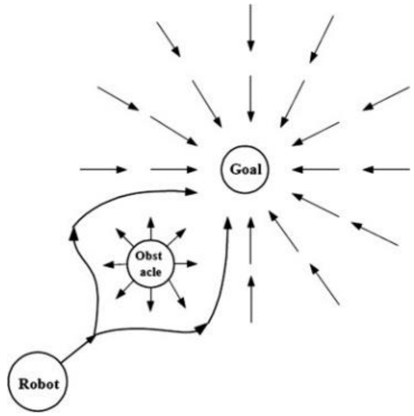


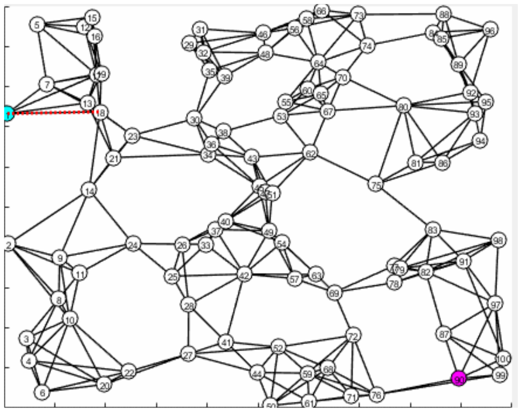
Karaman et. el, 2011



Common Strategies

Resolution Complete Methods





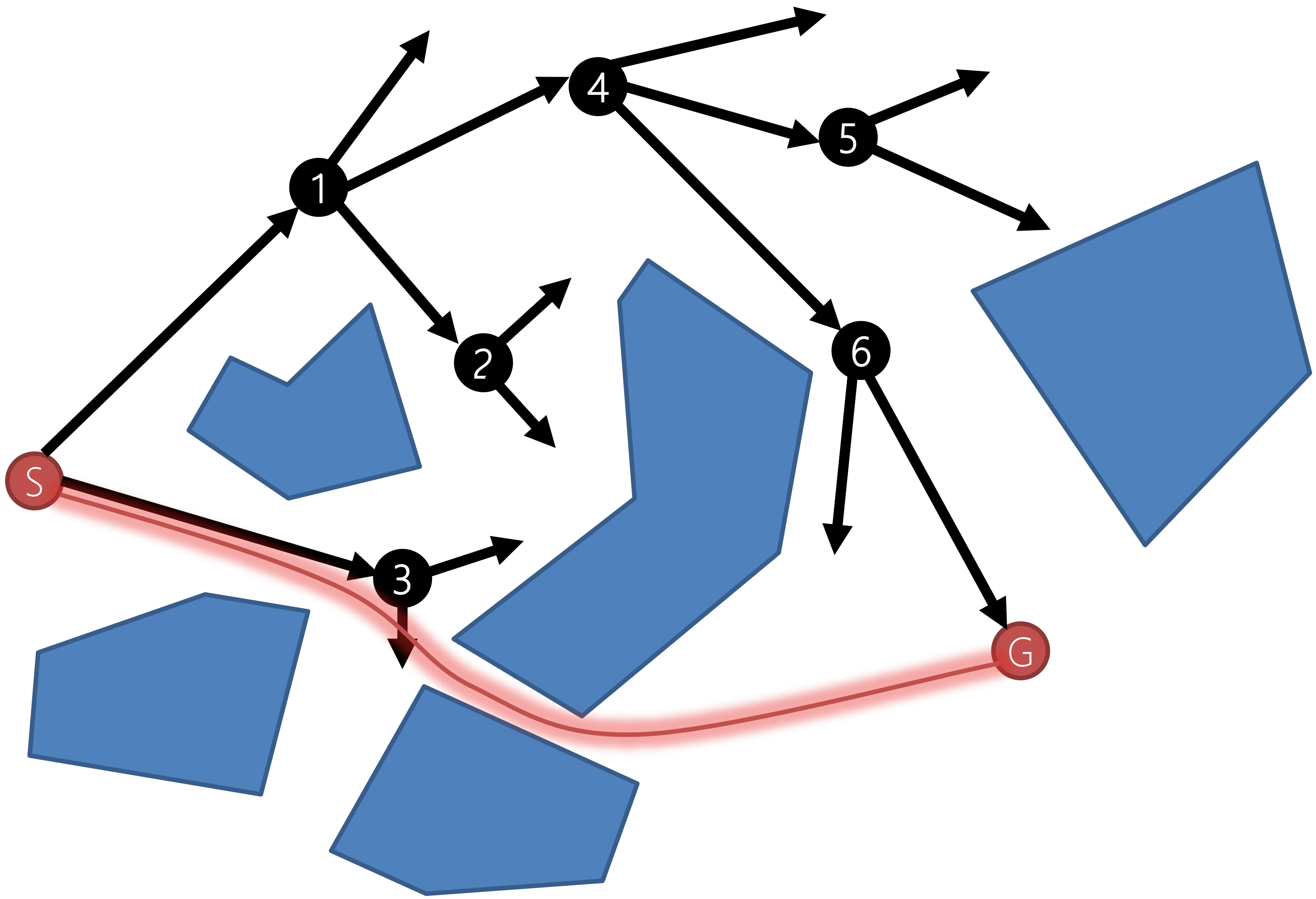
Artificial Potential Fields (Khatib, 86'), Cell Decomposition (Chazelle, 87')

Visibility Graphs (Lozano-Perez, 79')
Probabilistic Roadmaps (Kavraki, 96')

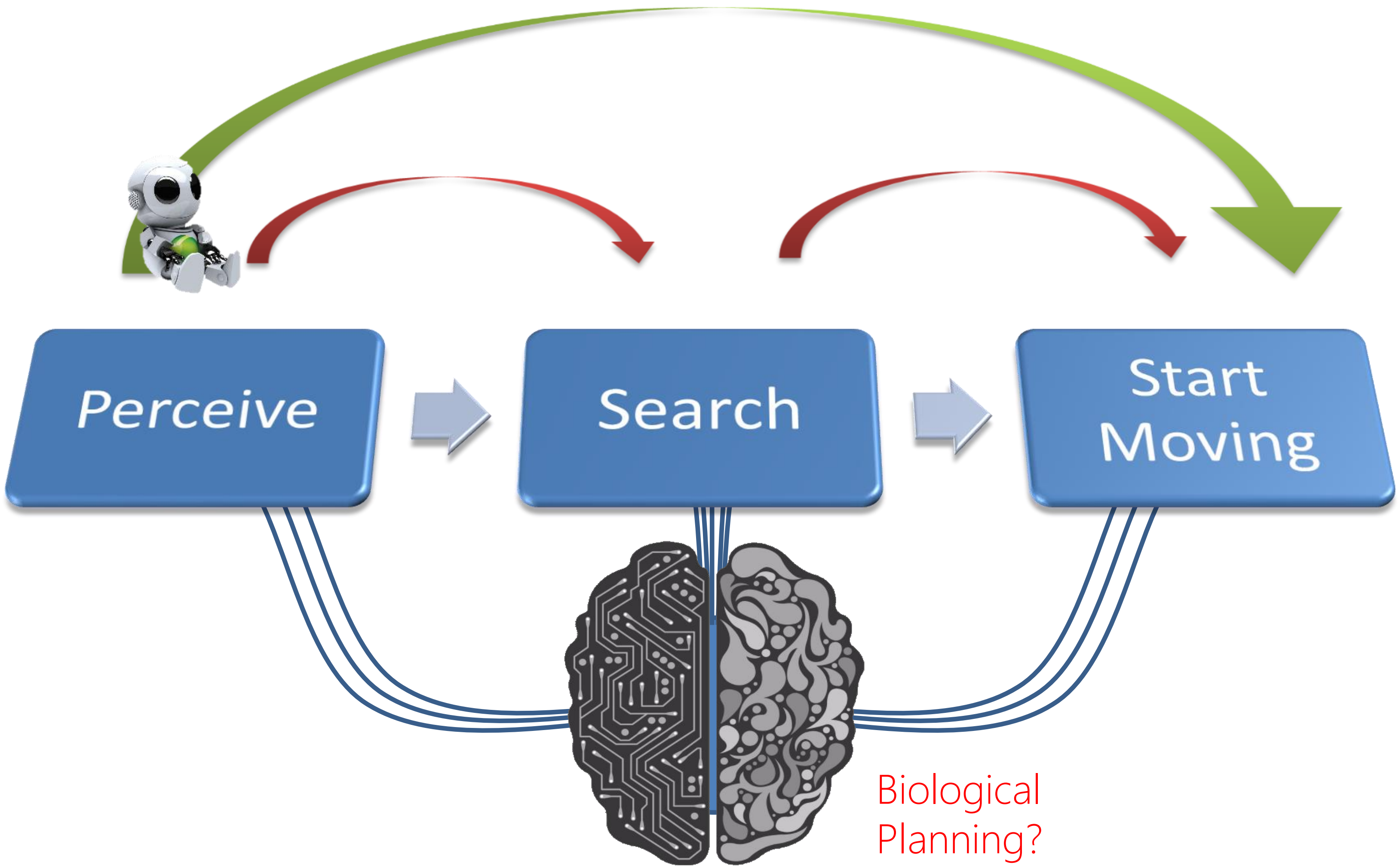
Sample-based Planners

RRT (LaValle, Kuffner, 98'), RRT (Karaman, 2011)*

Motion Planning

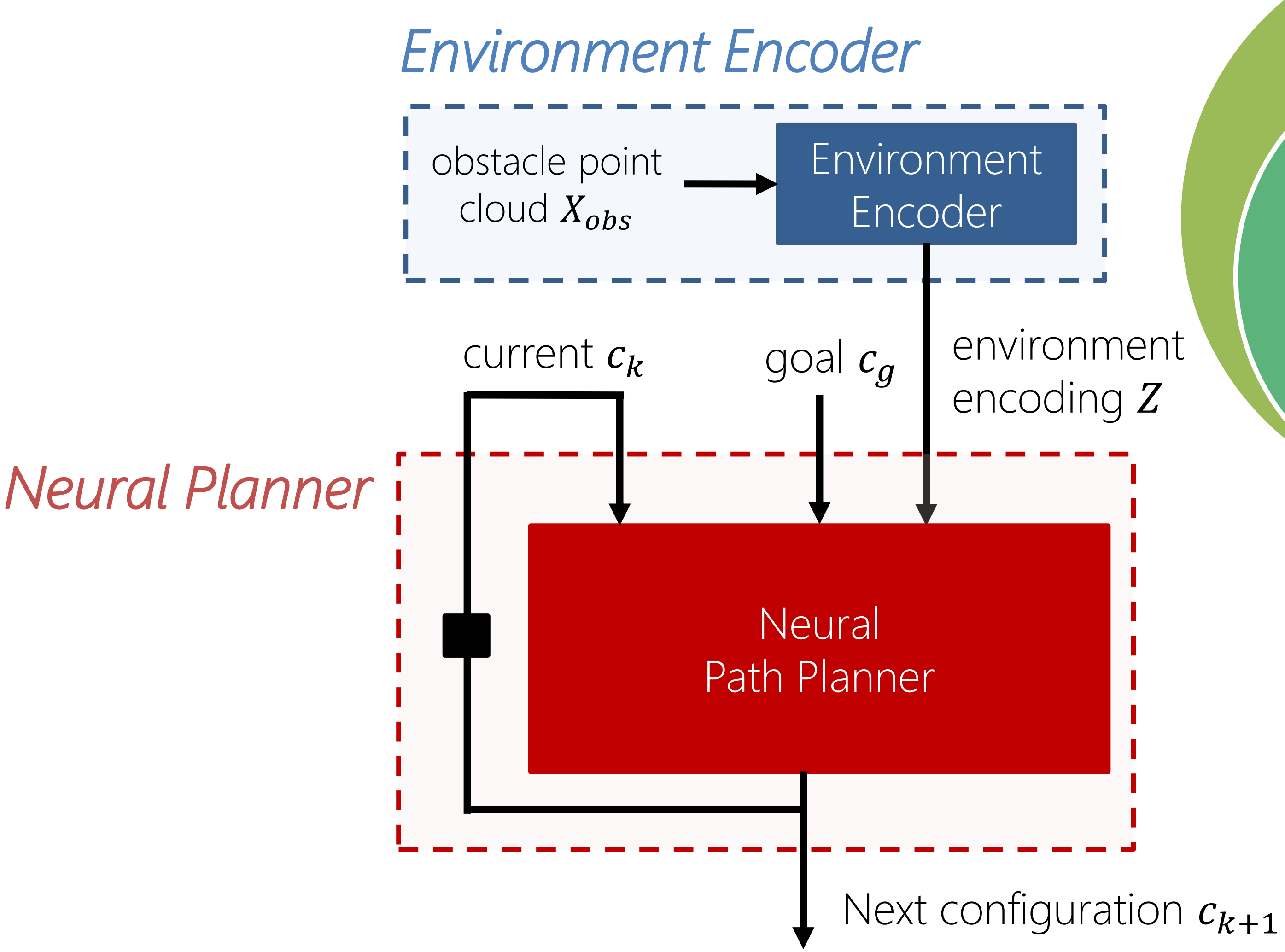


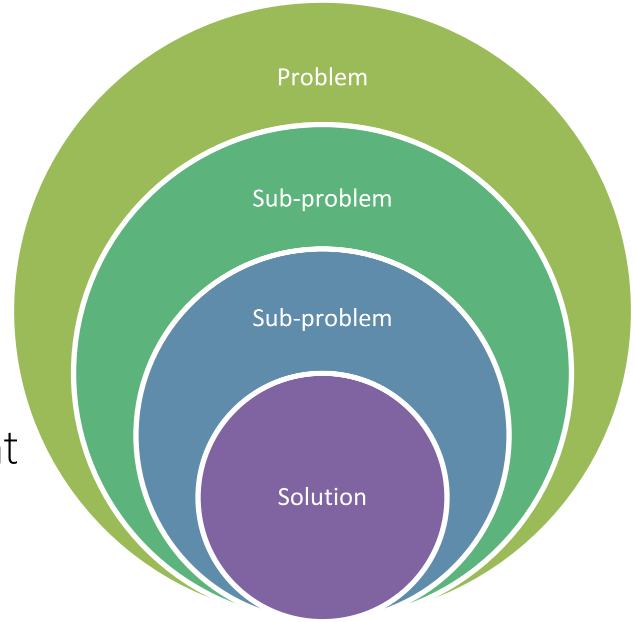
Sequence in Robot Thinking



Neural Motion Planning

- Motion Planning Networks (MPNet)





MPNet: Motion Planning Networks

Encoder Network (Enet):

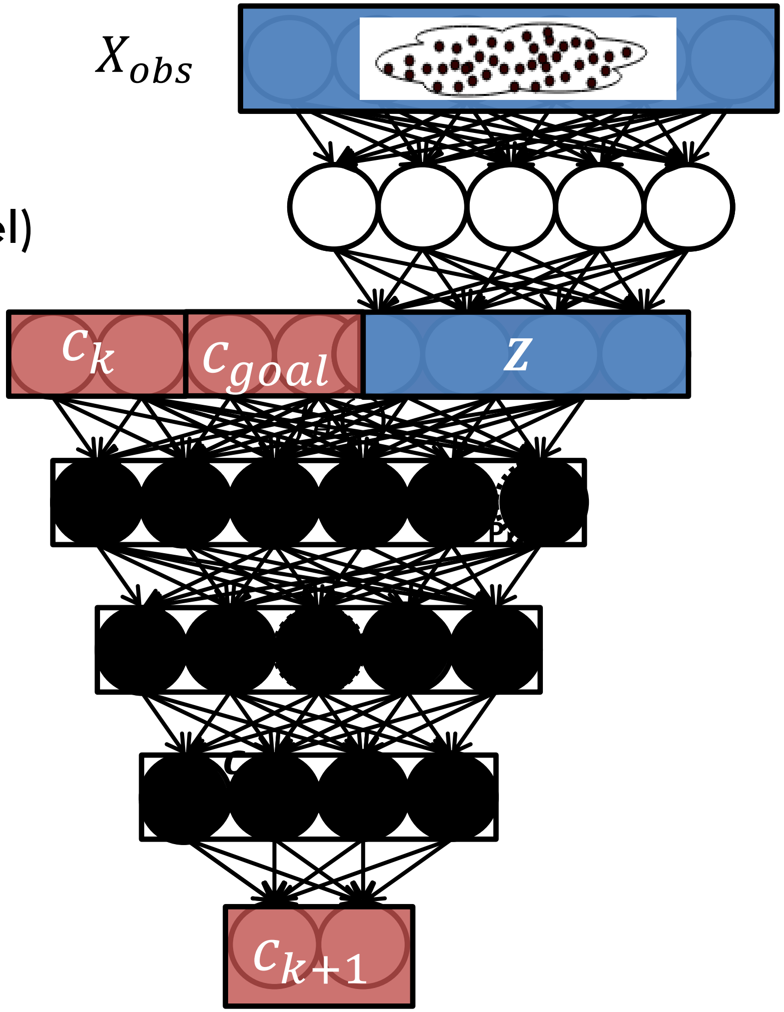
- Input: obstacles point cloud $\boldsymbol{x}_{obs} \in \mathbb{R}^{\boldsymbol{d}}$
- Output: Embedding $Z \in \mathbb{R}^{m}$
- 3D CNN (Preprocess point-cloud to voxel)
- Feed forward neural network

Planning Network (Pnet):

- Input: $\mathbf{Z}$, $\boldsymbol{c}_t, \boldsymbol{c}_T$
- Output: $\hat{\boldsymbol{c}}_{t+1} \leftarrow \mathbf{PNet}(\boldsymbol{c}_t, \boldsymbol{c}_T, \boldsymbol{Z})$
- Stochastic feed-forward neural network

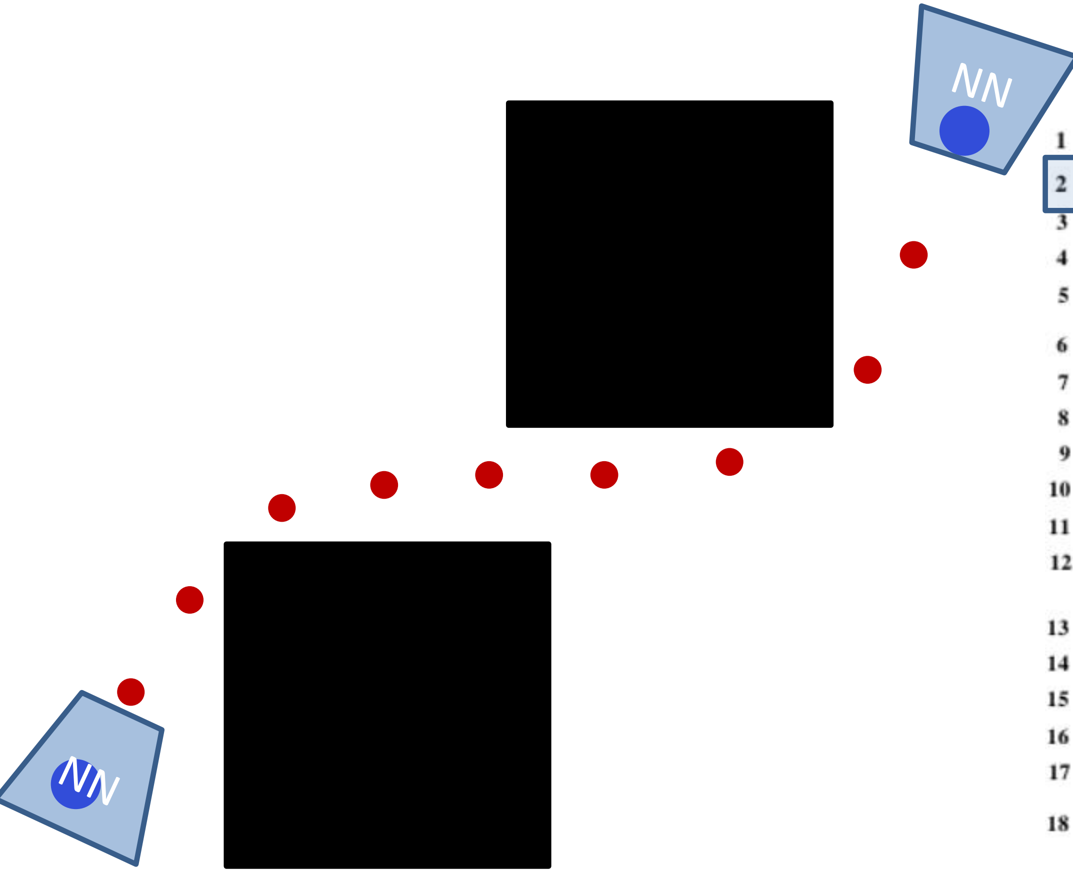
Recursive Planning Algorithm:

- End-to-end paths or informed samples
- Worst-case theoretical guarantees



MPNet: Recursive Bidirectional Neural Planning

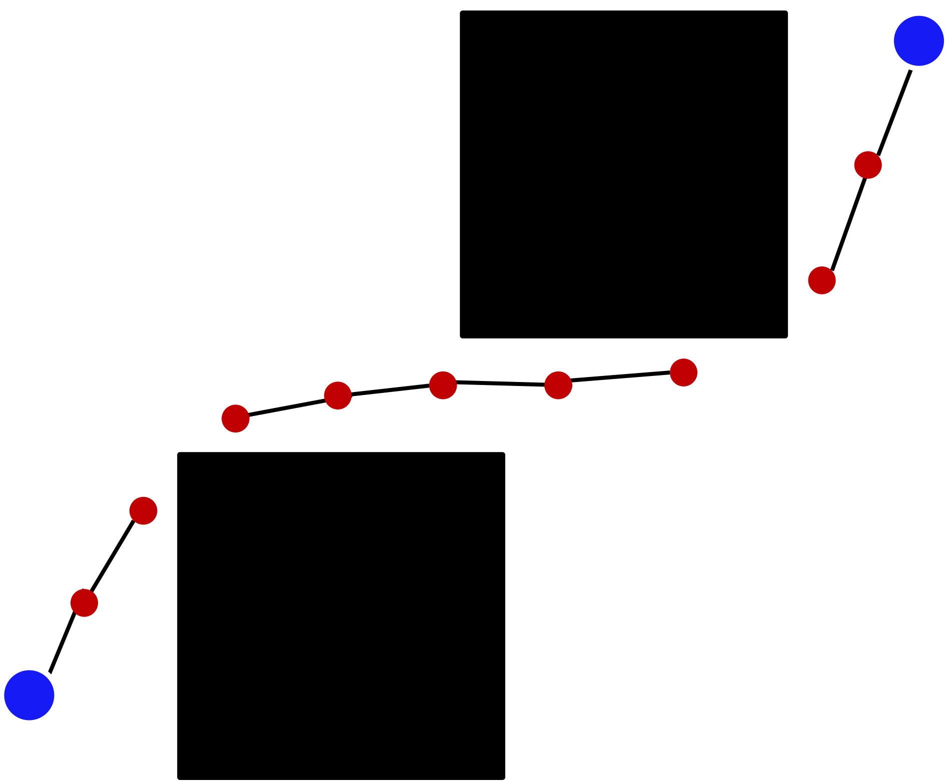
- Find critical states between given start and goal using MPNet.
- Create a coarse plan.




```
Z ← Enet(x_obs)
σ ← BNP(c_init, c_goal, Z)// BidirectionalNeuralPlanner
σ ← LazyStatesContraction(σ)
if IsFeasible(σ) then
    return σ
else
    p = 0// Set replanner to use BNP
    for i ← 0 to N_r do
        σ ← Replan(σ, Z, p)// NeuralReplanning
        σ ← LazyStatesContraction(σ)
        if IsFeasible(σ) then
            return σ
    p = 1// Set replanner to use OraclePlanner
    σ ← Replan(σ, Z, p)// HybridReplanning
    σ ← LazyStatesContraction(σ)
    if IsFeasible(σ) then
        return σ
    return ∅
```

MPNet: Recursive Bidirectional Neural Planning

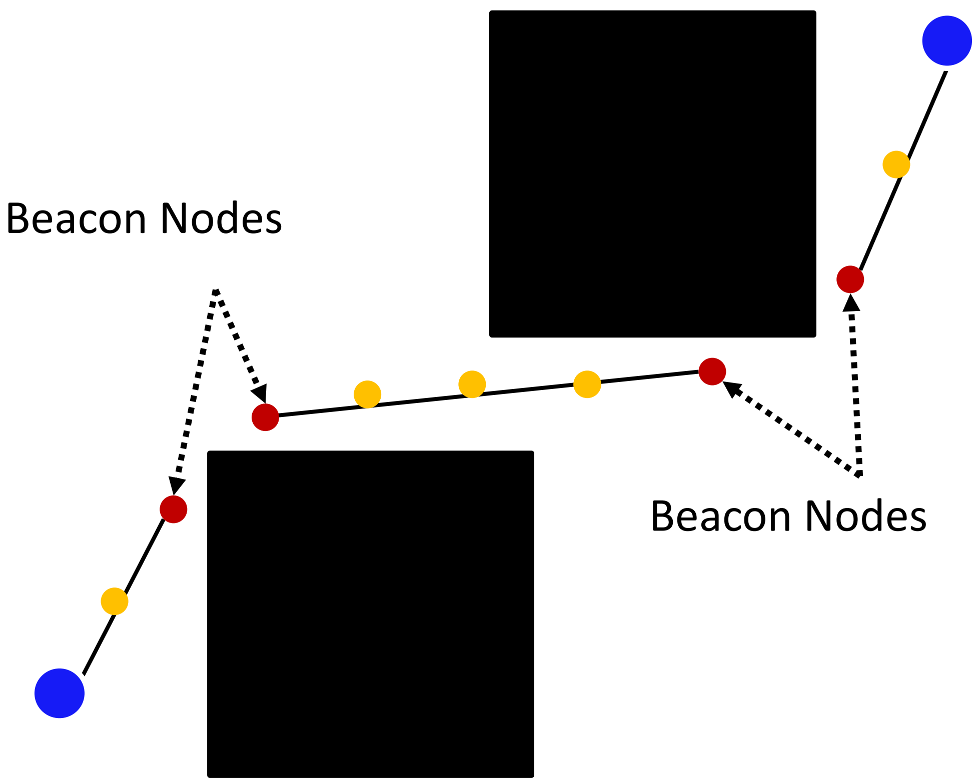
- Find critical states between given start and goal using MPNet.
- Create a coarse plan.



$Z \leftarrow \text{Enet}(x_{\text{obs}})$
$\boxed{\sigma \leftarrow \text{BNP}(c_{\text{init}}, c_{\text{goal}}, Z) \text{// BidirectionalNeuralPlanner}}$
$\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
if IsFeasible(σ) **then**
  return σ
else
  $p = 0$// Set replanner to use BNP
  for $i \leftarrow 0$ ***to*** N_r **do**
    $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$// NeuralReplanning
    $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
    if IsFeasible(σ) **then**
      return σ
  $p = 1$// Set replanner to use OraclePlanner
  $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$// HybridReplanning
  $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
  if IsFeasible(σ) **then**
    return σ
  return $\varnothing$

MPNet: Recursive Bidirectional Neural Planning

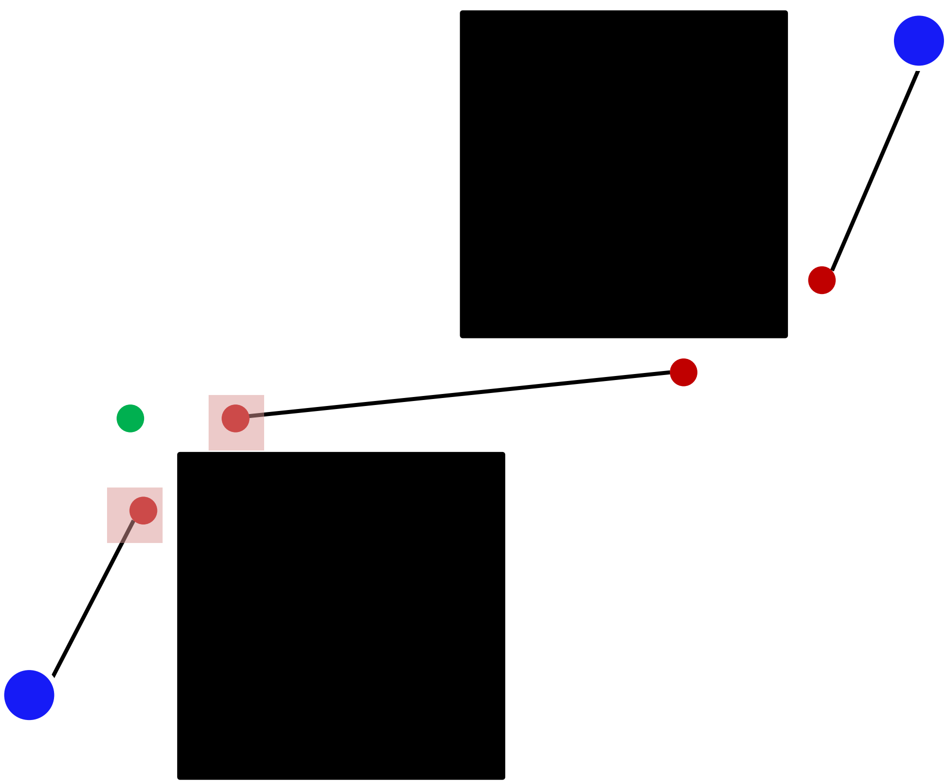
- Remove redundant states (branch-and-bound).
- Identify beacon states.




```
Z ← Enet(x_obs)
σ ← BNP(c_init, c_goal, Z)// BidirectionalNeuralPlanner
σ ← LazyStatesContraction(σ)
if IsFeasible(σ) then
    return σ
else
    p = 0// Set replanner to use BNP
    for i ← 0 to N_r do
        σ ← Replan(σ, Z, p)// NeuralReplanning
        σ ← LazyStatesContraction(σ)
        if IsFeasible(σ) then
            return σ
    p = 1// Set replanner to use OraclePlanner
    σ ← Replan(σ, Z, p)// HybridReplanning
    σ ← LazyStatesContraction(σ)
    if IsFeasible(σ) then
        return σ
    return ∅
```

MPNet: Recursive Bidirectional Neural Planning

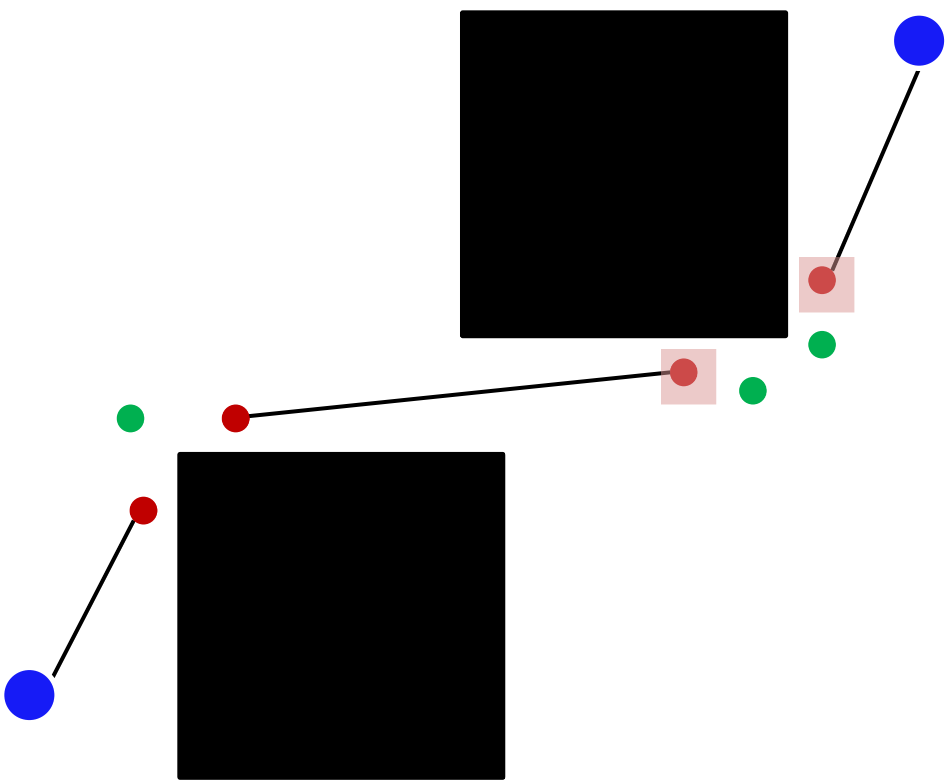
- Replan between beacon states with MPNet (Divide and conquer approach)



$Z \leftarrow \text{Enet}(x_{\text{obs}})$
$\sigma \leftarrow \text{BNP}(c_{\text{init}}, c_{\text{goal}}, Z)$// BidirectionalNeuralPlanner
$\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
if IsFeasible(σ) **then**
  return σ
else
  $p = 0$// Set replanner to use BNP
  for $i \leftarrow 0$ **to** N_r **do**
    $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$// NeuralReplanning
    $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
    if IsFeasible(σ) **then**
      return σ
  $p = 1$// Set replanner to use OraclePlanner
  $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$// HybridReplanning
  $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
  if IsFeasible(σ) **then**
    return σ
  return $\varnothing$

MPNet: Recursive Bidirectional Neural Planning

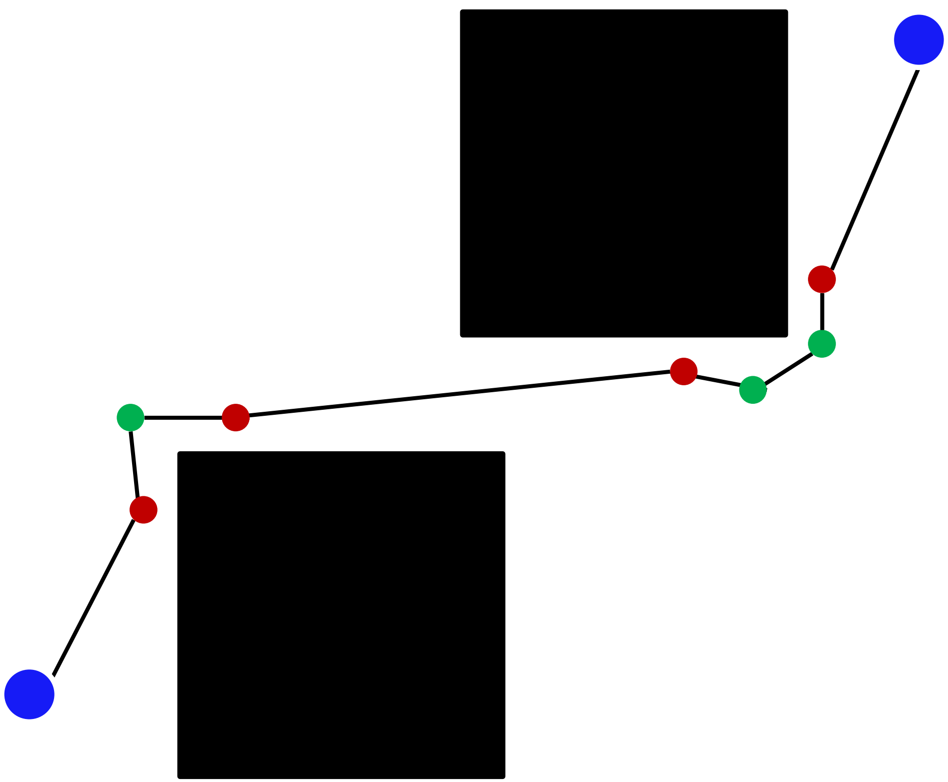
- Replan between beacon states with MPNet (Divide and conquer approach)



```
Z ← Enet(x_obs)
σ ← BNP(c_init, c_goal, Z)// BidirectionalNeuralPlanner
σ ← LazyStatesContraction(σ)
if IsFeasible(σ) then
    return σ
else
    p = 0// Set replanner to use BNP
    for i ← 0 to N_r do
        σ ← Replan(σ, Z, p)// NeuralReplanning
        σ ← LazyStatesContraction(σ)
        if IsFeasible(σ) then
            return σ
    p = 1// Set replanner to use OraclePlanner
    σ ← Replan(σ, Z, p)// HybridReplanning
    σ ← LazyStatesContraction(σ)
    if IsFeasible(σ) then
        return σ
    return ∅
```

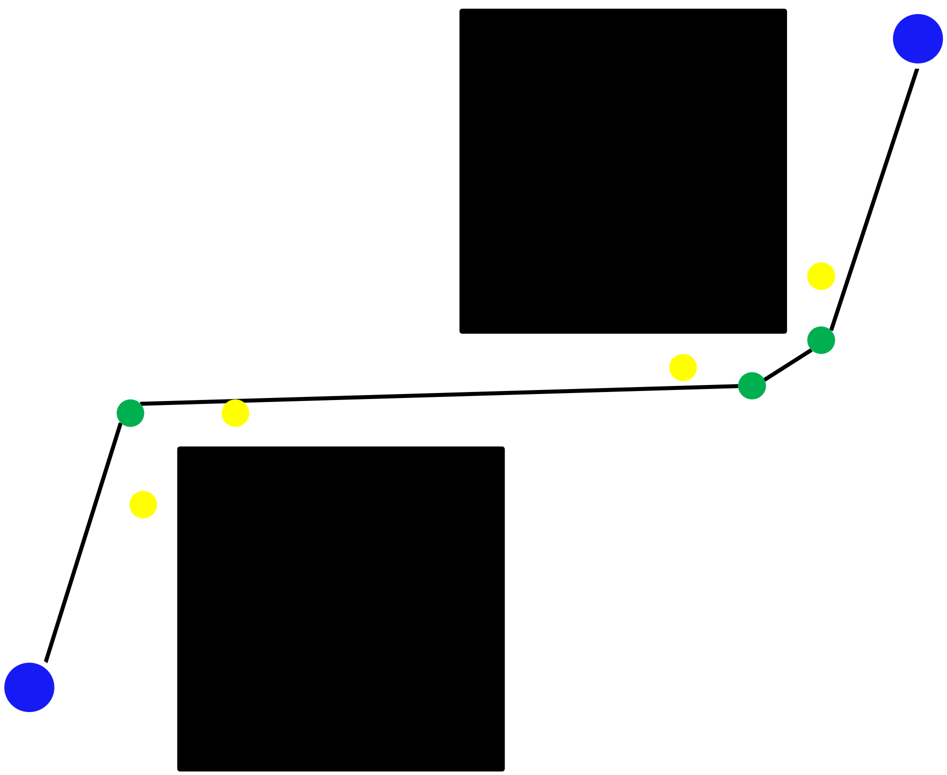
MPNet: Recursive Bidirectional Neural Planning

- Replan between beacon states with MPNet (Divide and conquer approach)



$Z \leftarrow \text{Enet}(x_{\text{obs}})$
$\sigma \leftarrow \text{BNP}(c_{\text{init}}, c_{\text{goal}}, Z)$ // BidirectionalNeuralPlanner
$\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
if IsFeasible(σ) **then**
  return σ
else
  $p = 0$ // Set replanner to use BNP
  for $i \leftarrow 0$ **to** N_r **do**
    $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$ // NeuralReplanning
    $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
    if IsFeasible(σ) **then**
      return σ
  $p = 1$ // Set replanner to use OraclePlanner
  $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$ // HybridReplanning
  $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
  if IsFeasible(σ) **then**
    return σ
  return $\varnothing$

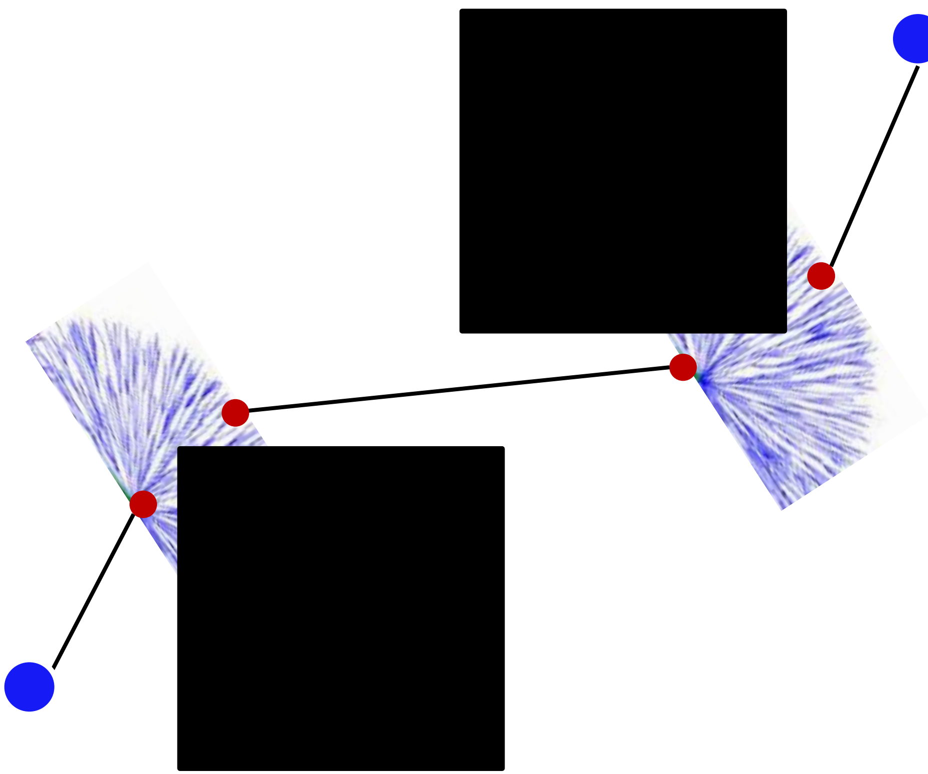
- Replan between beacon states with MPNet (Divide and conquer approach)



$Z \leftarrow \text{Enet}(x_{\text{obs}})$
$\sigma \leftarrow \text{BNP}(c_{\text{init}}, c_{\text{goal}}, Z)$ // BidirectionalNeuralPlanner
$\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
if IsFeasible(σ) **then**
    return σ
else
    $p = 0$ // Set replanner to use BNP
    for $i \leftarrow 0$ ***to*** N_r **do**
        $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$ // NeuralReplanning
        $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
        if IsFeasible(σ) **then**
            return σ
    $p = 1$ // Set replanner to use OraclePlanner
    $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$ // HybridReplanning
    $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
    if IsFeasible(σ) **then**
        return σ
    return $\varnothing$

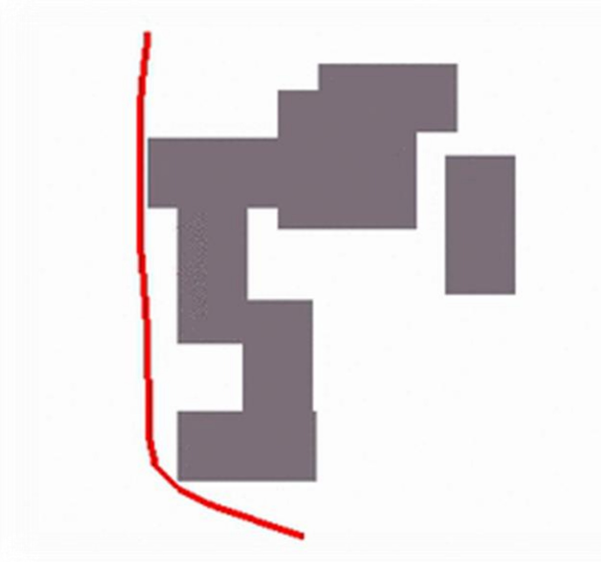
MPNet: Recursive Bidirectional Neural Planning

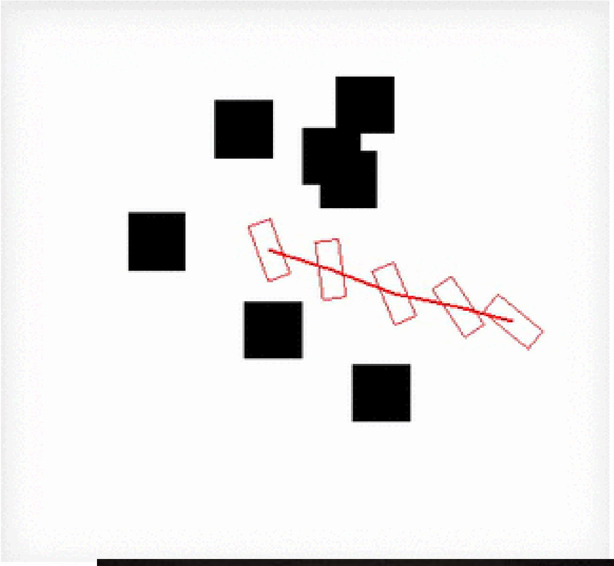
- **Hybrid replanning:** Combines MPNet with classical planners.
  - Outsource a segment of a planning problem to a classical planner.

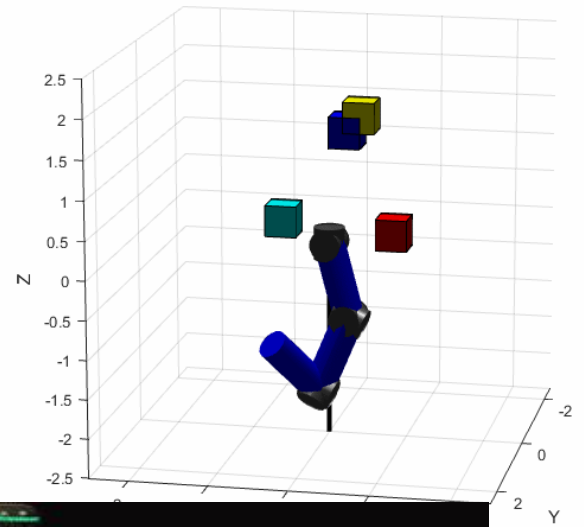


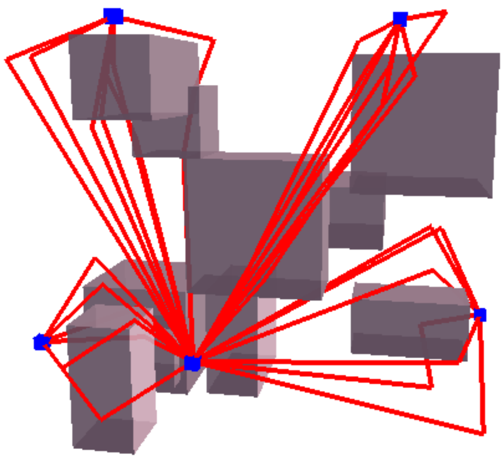
$Z \leftarrow \text{Enet}(x_{\text{obs}})$
$\sigma \leftarrow \text{BNP}(c_{\text{init}}, c_{\text{goal}}, Z)$ // BidirectionalNeuralPlanner
$\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
if IsFeasible(σ) **then**
  return σ
else
  $p = 0$ // Set replanner to use BNP
  for $i \leftarrow 0$ ***to*** N_r **do**
    $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$ // NeuralReplanning
    $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
    if IsFeasible(σ) **then**
      return σ
  $p = 1$ // Set replanner to use OraclePlanner
  $\sigma \leftarrow \text{Replan}(\sigma, Z, p)$ // HybridReplanning
  $\sigma \leftarrow \text{LazyStatesContraction}(\sigma)$
  if IsFeasible(σ) **then**
    return σ
  return $\varnothing$

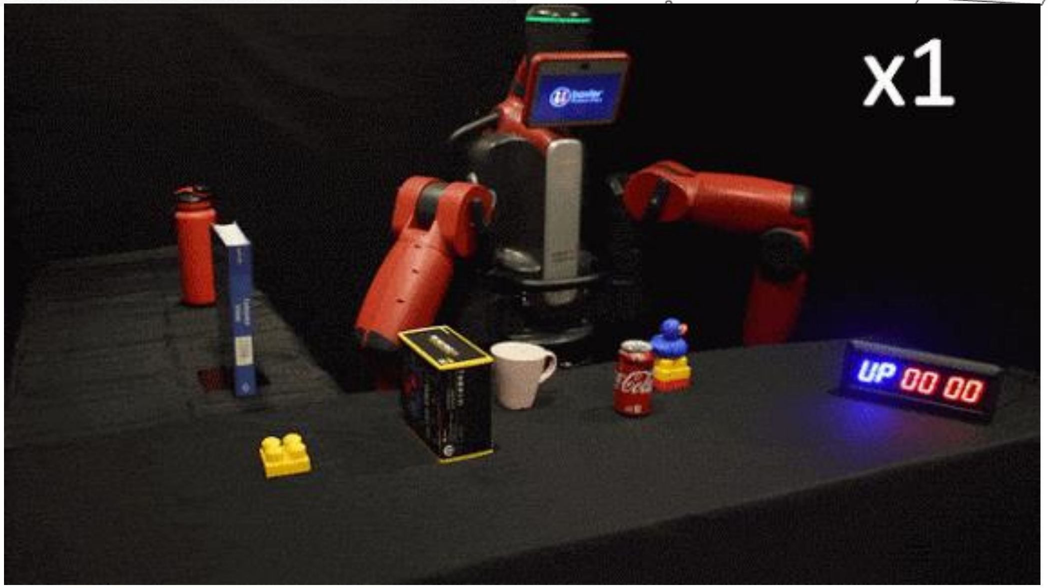
Visualizations





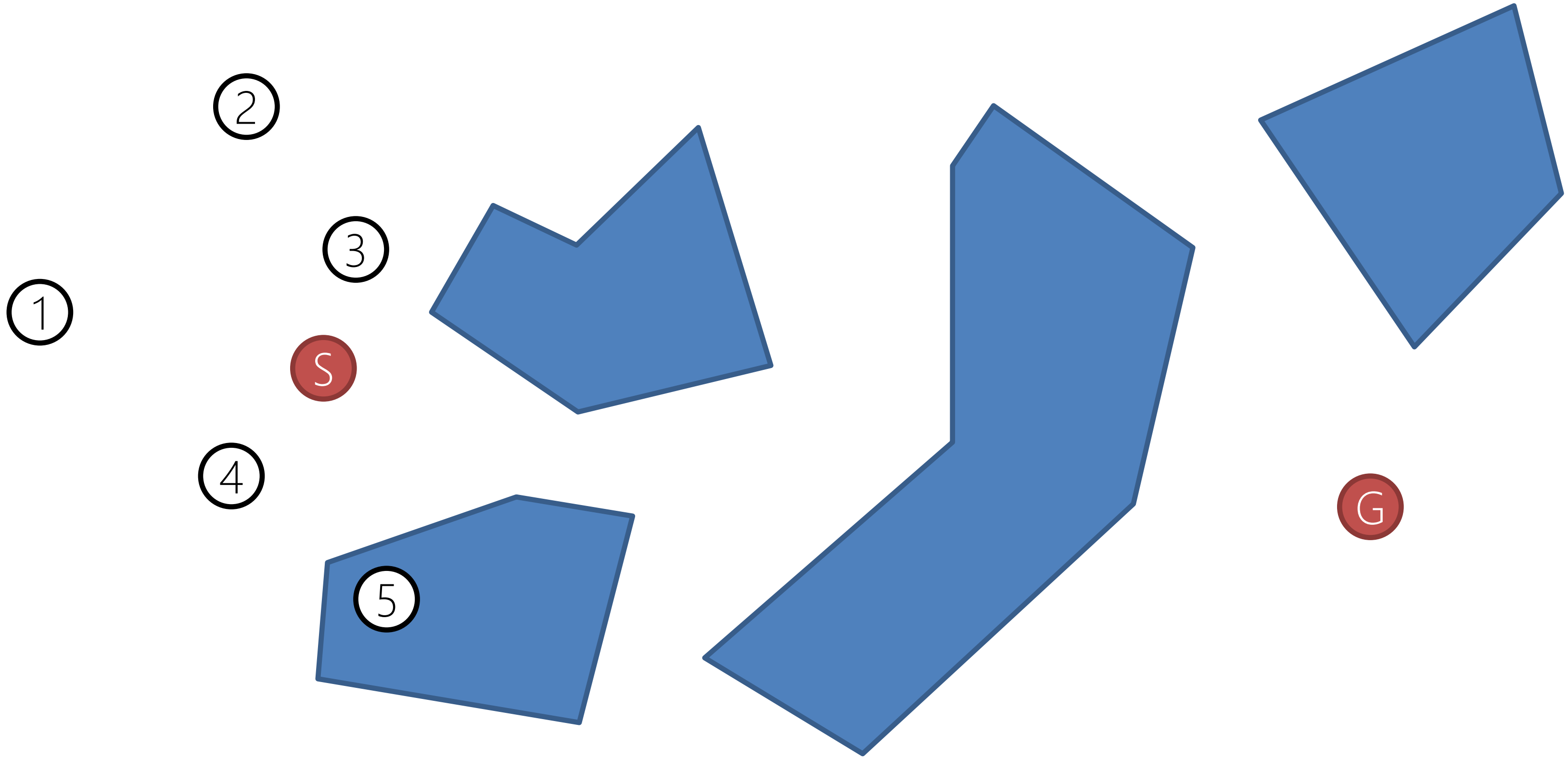




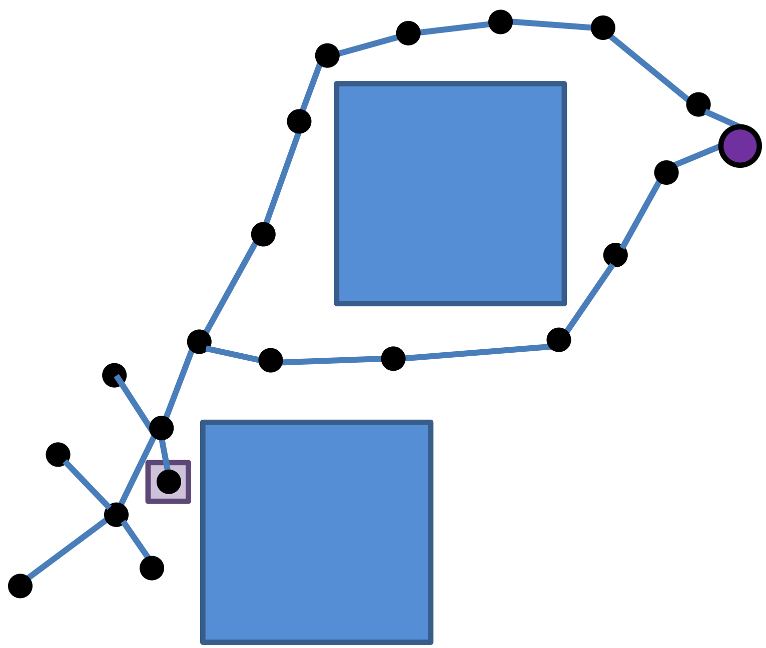


Informed Sampling

But what if we wanted to use this not to *find the solution* but make *informed choices?*

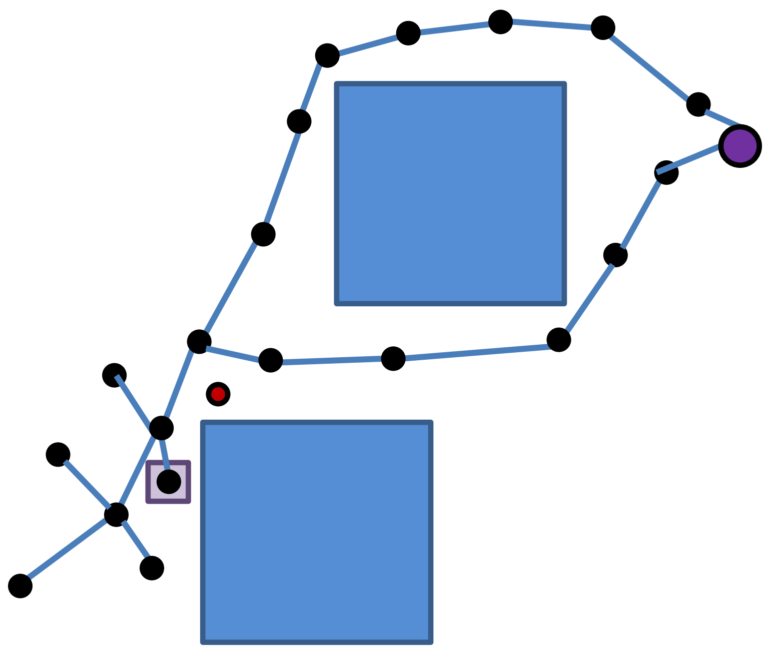


MPNet: Informed Sampling



Initialize $T \leftarrow \mathrm{SMP}\ (c_{\mathrm{init}}, c_{\mathrm{goal}}, \mathcal{X}_{\mathrm{obs}})$
$c_{\mathrm{rand}} \leftarrow c_{\mathrm{init}}$
$Z \leftarrow \mathrm{Enet}(x_{\mathrm{obs}})$
for $i \leftarrow 0$ ***to*** n **do**
  if $i < N_{\mathrm{smp}}$ **then**
    $c_{\mathrm{rand}} \leftarrow \mathrm{MPNet}(Z, c_{\mathrm{rand}}, c_{\mathrm{goal}})$
  else
    $c_{\mathrm{rand}} \leftarrow \mathrm{RandomSampler}()$
  $c_{\mathrm{nearest}} \leftarrow \mathrm{Nearest}(c_{\mathrm{rand}}, T)$
  if $\mathrm{Steer}(c_{\mathrm{nearest}}, c_{\mathrm{rand}})$ **then**
    $C_{\mathrm{near}} \leftarrow \mathrm{Near}(c_{\mathrm{rand}}, T)$
    $T \leftarrow \mathrm{Rewire}(T, C_{\mathrm{near}}, c_{\mathrm{rand}})$
  if $c_{\mathrm{rand}} \in c_{\mathrm{goal}}$ **then**
    $c_{\mathrm{rand}} \leftarrow c_{\mathrm{init}}$
return $\varnothing$

MPNet: Informed Sampling



Initialize $T \leftarrow \mathrm{SMP}\ (c_{\mathrm{init}}, c_{\mathrm{goal}}, \mathcal{X}_{\mathrm{obs}})$

$c_{\mathrm{rand}} \leftarrow c_{\mathrm{init}}$

$Z \leftarrow \mathrm{Enet}(x_{\mathrm{obs}})$

for $i \leftarrow 0$ ***to*** n **do**

  if $i < N_{\mathrm{smp}}$ **then**

    $c_{\mathrm{rand}} \leftarrow \mathrm{MPNet}(Z, c_{\mathrm{rand}}, c_{\mathrm{goal}})$

  else

    $c_{\mathrm{rand}} \leftarrow \mathrm{RandomSampler}()$

  $c_{\mathrm{nearest}} \leftarrow \mathrm{Nearest}(c_{\mathrm{rand}}, T)$

  if $\mathrm{Steer}(c_{\mathrm{nearest}}, c_{\mathrm{rand}})$ **then**

    $C_{\mathrm{near}} \leftarrow \mathrm{Near}(c_{\mathrm{rand}}, T)$

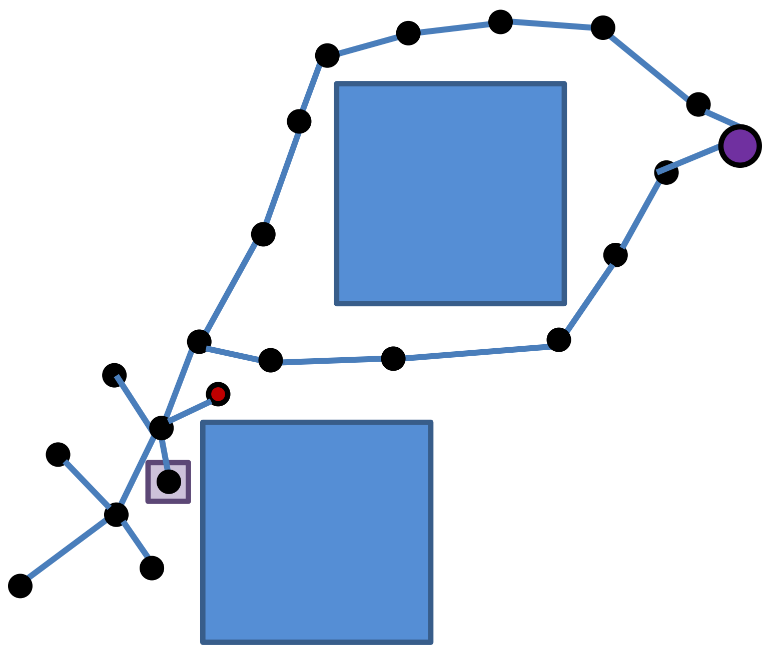
    $T \leftarrow \mathrm{Rewire}(T, C_{\mathrm{near}}, c_{\mathrm{rand}})$

  if $c_{\mathrm{rand}} \in c_{\mathrm{goal}}$ **then**

    $c_{\mathrm{rand}} \leftarrow c_{\mathrm{init}}$

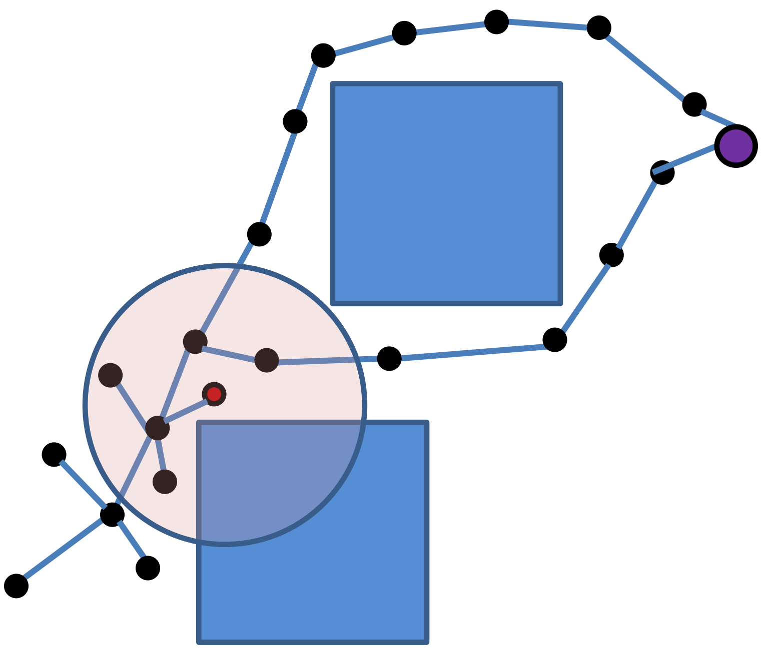
return $\varnothing$

MPNet: Informed Sampling



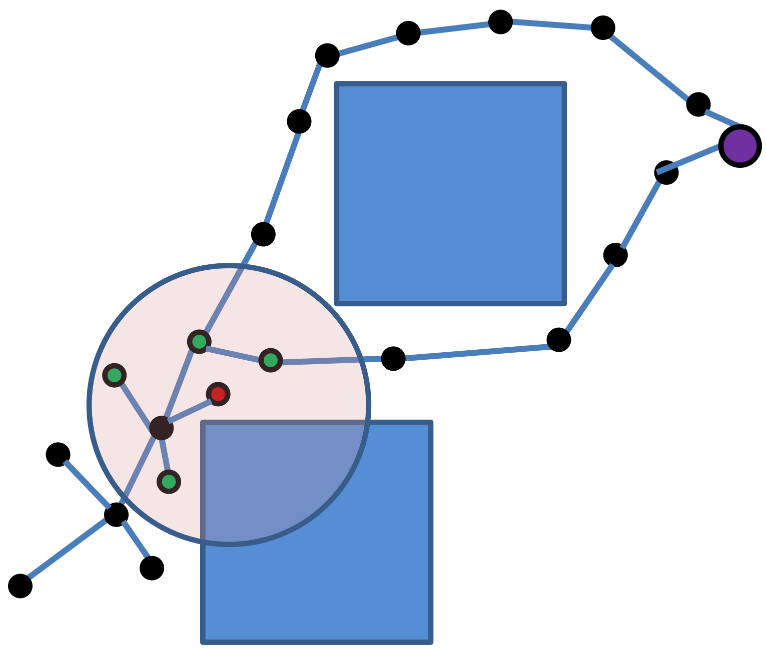
Initialize $T \leftarrow$ SMP $(c_{\text{init}}, c_{\text{goal}}, \mathcal{X}_{\text{obs}})$
$c_{\text{rand}} \leftarrow c_{\text{init}}$
$Z \leftarrow \text{Enet}(x_{\text{obs}})$
for $i \leftarrow 0$ ***to*** n **do**
  if $i < N_{\text{smp}}$ **then**
    $c_{\text{rand}} \leftarrow \text{MPNet}\big(Z, c_{\text{rand}}, c_{\text{goal}}\big)$
  else
    $c_{\text{rand}} \leftarrow \text{RandomSampler}()$
  $c_{\text{nearest}} \leftarrow \text{Nearest}\big(c_{\text{rand}}, T\big)$
  if $\text{Steer}\big(c_{\text{nearest}}, c_{\text{rand}}\big)$ **then**
    $C_{\text{near}} \leftarrow \text{Near}(c_{\text{rand}}, T)$
    $T \leftarrow \text{Rewire}(T, C_{\text{near}}, c_{\text{rand}})$
  if $c_{\text{rand}} \in c_{\text{goal}}$ **then**
    $c_{\text{rand}} \leftarrow c_{\text{init}}$
return $\varnothing$

MPNet: Informed Sampling



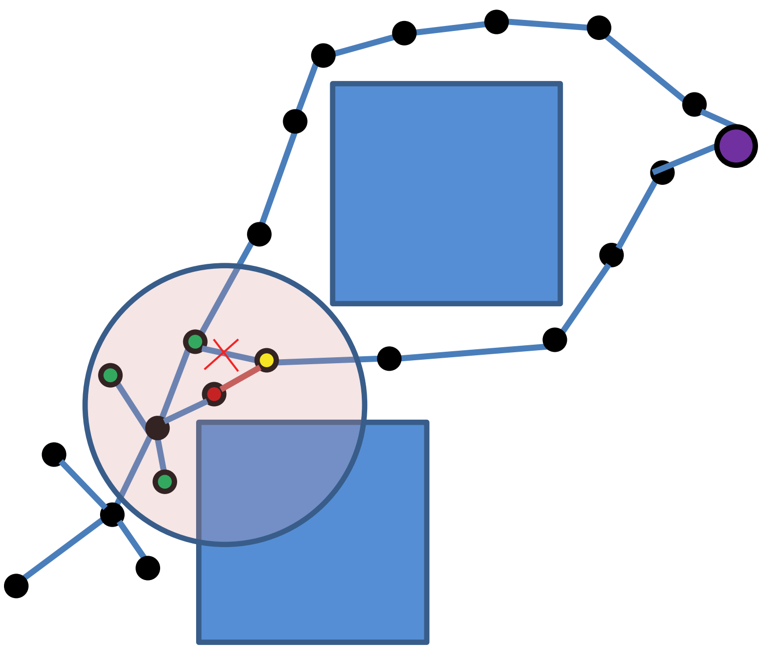
Initialize $T \leftarrow$ SMP $(c_{\text{init}}, c_{\text{goal}}, \mathcal{X}_{\text{obs}})$
$c_{\text{rand}} \leftarrow c_{\text{init}}$
$Z \leftarrow \text{Enet}(x_{\text{obs}})$
for $i \leftarrow 0$ ***to*** n **do**
    if $i < N_{\text{smp}}$ **then**
        $c_{\text{rand}} \leftarrow \text{MPNet}(Z, c_{\text{rand}}, c_{\text{goal}})$
    else
        $c_{\text{rand}} \leftarrow \text{RandomSampler}()$
    $c_{\text{nearest}} \leftarrow \text{Nearest}(c_{\text{rand}}, T)$
    if $\text{Steer}(c_{\text{nearest}}, c_{\text{rand}})$ **then**
        $C_{\text{near}} \leftarrow \text{Near}(c_{\text{rand}}, T)$
        $T \leftarrow \text{Rewire}(T, C_{\text{near}}, c_{\text{rand}})$
    if $c_{\text{rand}} \in c_{\text{goal}}$ **then**
        $c_{\text{rand}} \leftarrow c_{\text{init}}$
return $\varnothing$

MPNet: Informed Sampling



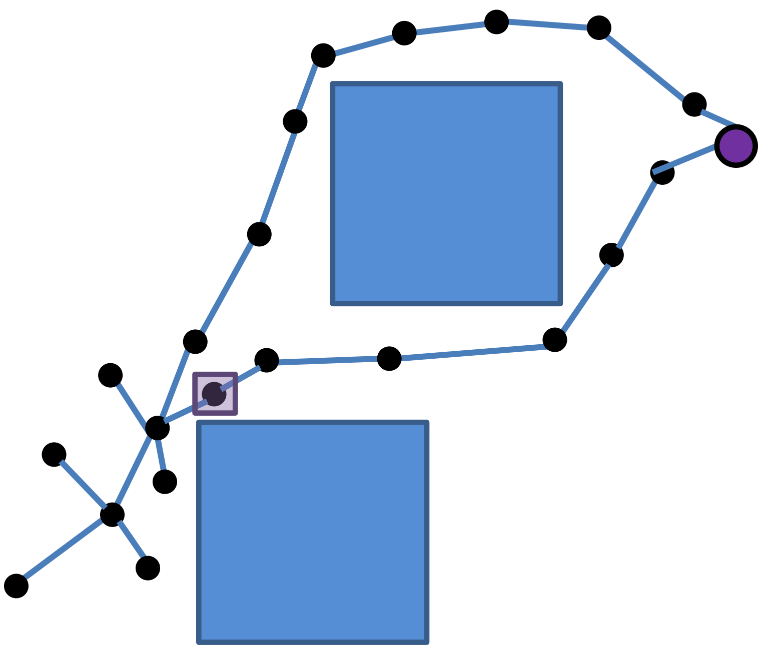
Initialize $T \leftarrow \text{SMP}\ (c_{\text{init}}, c_{\text{goal}}, \mathcal{X}_{\text{obs}})$
$c_{\text{rand}} \leftarrow c_{\text{init}}$
$Z \leftarrow \text{Enet}(x_{\text{obs}})$
for $i \leftarrow 0$ ***to*** n **do**
  if $i < N_{\text{smp}}$ **then**
    $c_{\text{rand}} \leftarrow \text{MPNet}(Z, c_{\text{rand}}, c_{\text{goal}})$
  else
    $c_{\text{rand}} \leftarrow \text{RandomSampler}()$
  $c_{\text{nearest}} \leftarrow \text{Nearest}(c_{\text{rand}}, T)$
  if $\text{Steer}(c_{\text{nearest}}, c_{\text{rand}})$ **then**
    $C_{\text{near}} \leftarrow \text{Near}(c_{\text{rand}}, T)$
    $T \leftarrow \text{Rewire}(T, C_{\text{near}}, c_{\text{rand}})$
  if $c_{\text{rand}} \in c_{\text{goal}}$ **then**
    $c_{\text{rand}} \leftarrow c_{\text{init}}$
return $\varnothing$

MPNet: Informed Sampling



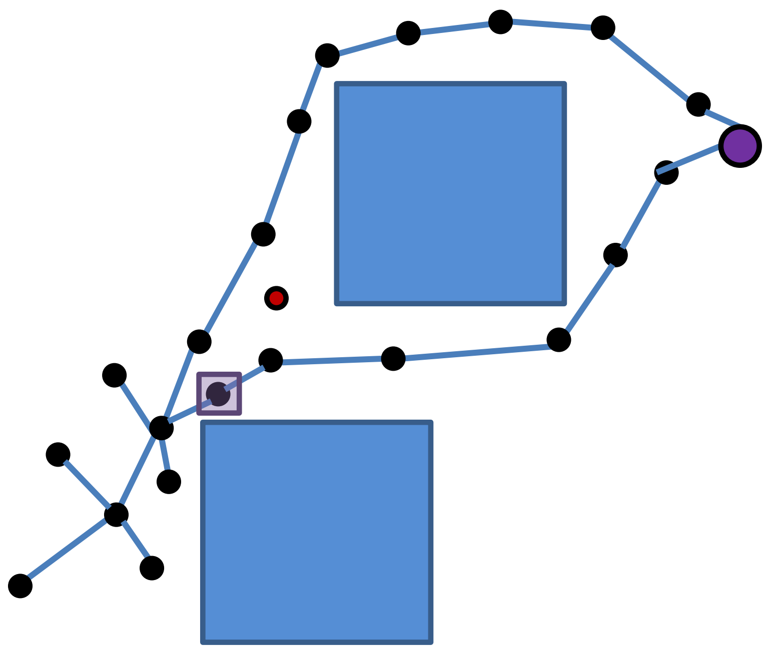
Initialize $T \leftarrow \text{SMP}\ (c_{\text{init}}, c_{\text{goal}}, \mathcal{X}_{\text{obs}})$
$c_{\text{rand}} \leftarrow c_{\text{init}}$
$Z \leftarrow \text{Enet}(x_{\text{obs}})$
for $i \leftarrow 0$ ***to*** n **do**
  if $i < N_{\text{smp}}$ **then**
    $c_{\text{rand}} \leftarrow \text{MPNet}(Z, c_{\text{rand}}, c_{\text{goal}})$
  else
    $c_{\text{rand}} \leftarrow \text{RandomSampler}()$
  $c_{\text{nearest}} \leftarrow \text{Nearest}(c_{\text{rand}}, T)$
  if $\text{Steer}(c_{\text{nearest}}, c_{\text{rand}})$ **then**
    $C_{\text{near}} \leftarrow \text{Near}(c_{\text{rand}}, T)$
    $T \leftarrow \text{Rewire}(T, C_{\text{near}}, c_{\text{rand}})$
  if $c_{\text{rand}} \in c_{\text{goal}}$ **then**
    $c_{\text{rand}} \leftarrow c_{\text{init}}$
return $\varnothing$

Informed Sampling



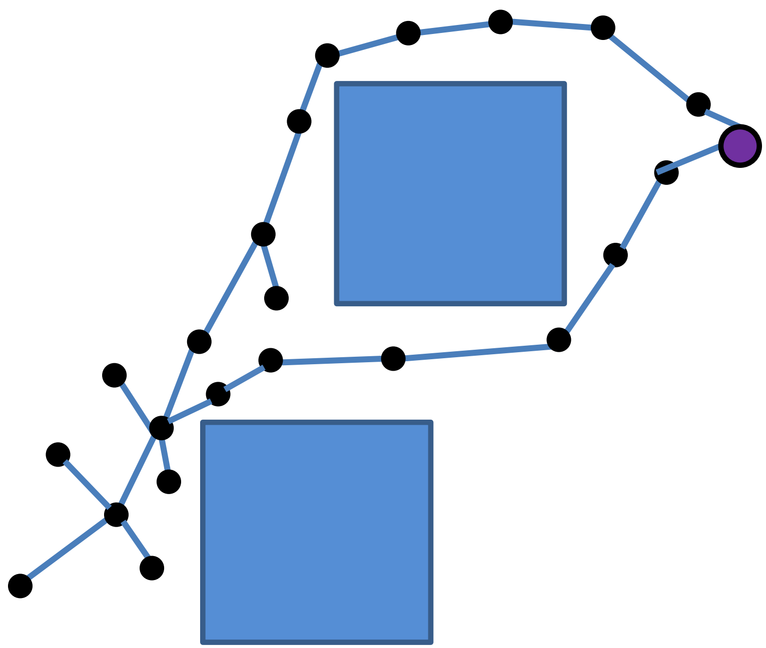
Initialize $T \leftarrow \text{SMP}\ (c_{\text{init}}, c_{\text{goal}}, \mathcal{X}_{\text{obs}})$
$c_{\text{rand}} \leftarrow c_{\text{init}}$
$Z \leftarrow \text{Enet}(x_{\text{obs}})$
for $i \leftarrow 0$ ***to*** n **do**
    if $i < N_{\text{smp}}$ **then**
        $c_{\text{rand}} \leftarrow \text{MPNet}(Z, c_{\text{rand}}, c_{\text{goal}})$
    else
        $c_{\text{rand}} \leftarrow \text{RandomSampler}()$
    $c_{\text{nearest}} \leftarrow \text{Nearest}(c_{\text{rand}}, T)$
    if $\text{Steer}(c_{\text{nearest}}, c_{\text{rand}})$ **then**
        $C_{\text{near}} \leftarrow \text{Near}(c_{\text{rand}}, T)$
        $T \leftarrow \text{Rewire}(T, C_{\text{near}}, c_{\text{rand}})$
    if $c_{\text{rand}} \in c_{\text{goal}}$ **then**
        $c_{\text{rand}} \leftarrow c_{\text{init}}$
return $\varnothing$

Informed Sampling



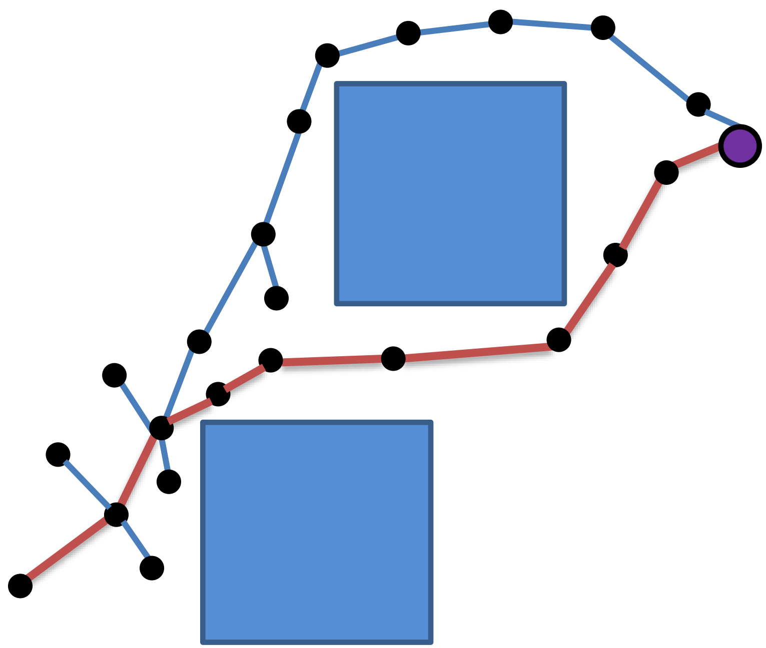
Initialize $T \leftarrow \text{SMP}\,(c_{\text{init}}, c_{\text{goal}}, \mathcal{X}_{\text{obs}})$
$c_{\text{rand}} \leftarrow c_{\text{init}}$
$Z \leftarrow \text{Enet}(x_{\text{obs}})$
for $i \leftarrow 0$ ***to*** n **do**
    if $i < N_{\text{smp}}$ **then**
        $c_{\text{rand}} \leftarrow \text{MPNet}(Z, c_{\text{rand}}, c_{\text{goal}})$
    else
        $c_{\text{rand}} \leftarrow \text{RandomSampler}()$
    $c_{\text{nearest}} \leftarrow \text{Nearest}(c_{\text{rand}}, T)$
    if $\text{Steer}(c_{\text{nearest}}, c_{\text{rand}})$ **then**
        $C_{\text{near}} \leftarrow \text{Near}(c_{\text{rand}}, T)$
        $T \leftarrow \text{Rewire}(T, C_{\text{near}}, c_{\text{rand}})$
    if $c_{\text{rand}} \in c_{\text{goal}}$ **then**
        $c_{\text{rand}} \leftarrow c_{\text{init}}$
return $\varnothing$

Informed Sampling



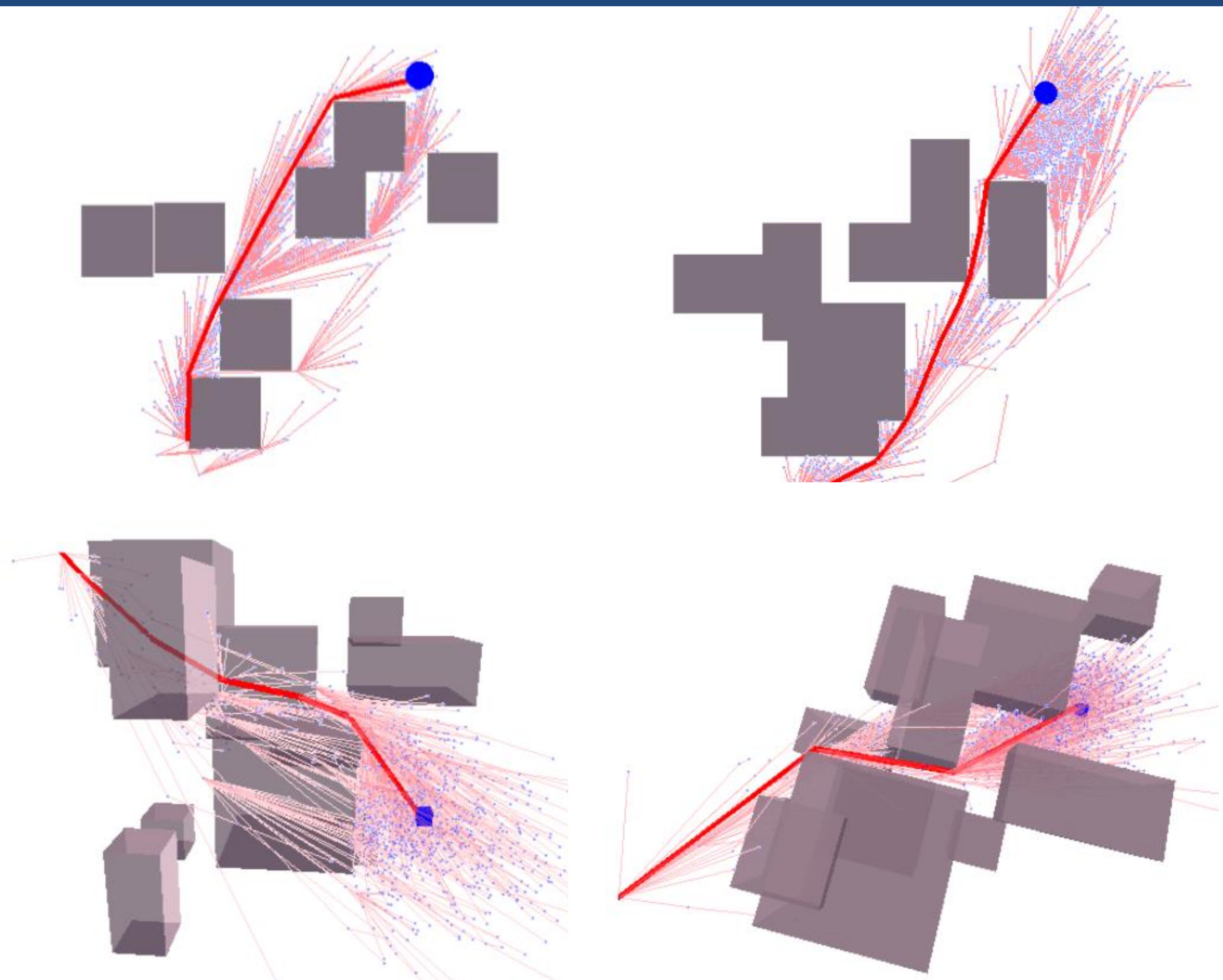
Initialize $T \leftarrow$ SMP $(c_{\text{init}}, c_{\text{goal}}, \mathcal{X}_{\text{obs}})$
$c_{\text{rand}} \leftarrow c_{\text{init}}$
$Z \leftarrow \text{Enet}(x_{\text{obs}})$
for $i \leftarrow 0$ ***to*** n **do**
  if $i < N_{\text{smp}}$ **then**
    $c_{\text{rand}} \leftarrow \text{MPNet}(Z, c_{\text{rand}}, c_{\text{goal}})$
  else
    $c_{\text{rand}} \leftarrow \text{RandomSampler}()$
  $c_{\text{nearest}} \leftarrow \text{Nearest}(c_{\text{rand}}, T)$
  if $\text{Steer}(c_{\text{nearest}}, c_{\text{rand}})$ **then**
    $C_{\text{near}} \leftarrow \text{Near}(c_{\text{rand}}, T)$
    $T \leftarrow \text{Rewire}(T, C_{\text{near}}, c_{\text{rand}})$
  if $c_{\text{rand}} \in c_{\text{goal}}$ **then**
    $c_{\text{rand}} \leftarrow c_{\text{init}}$
return $\varnothing$

Informed Sampling

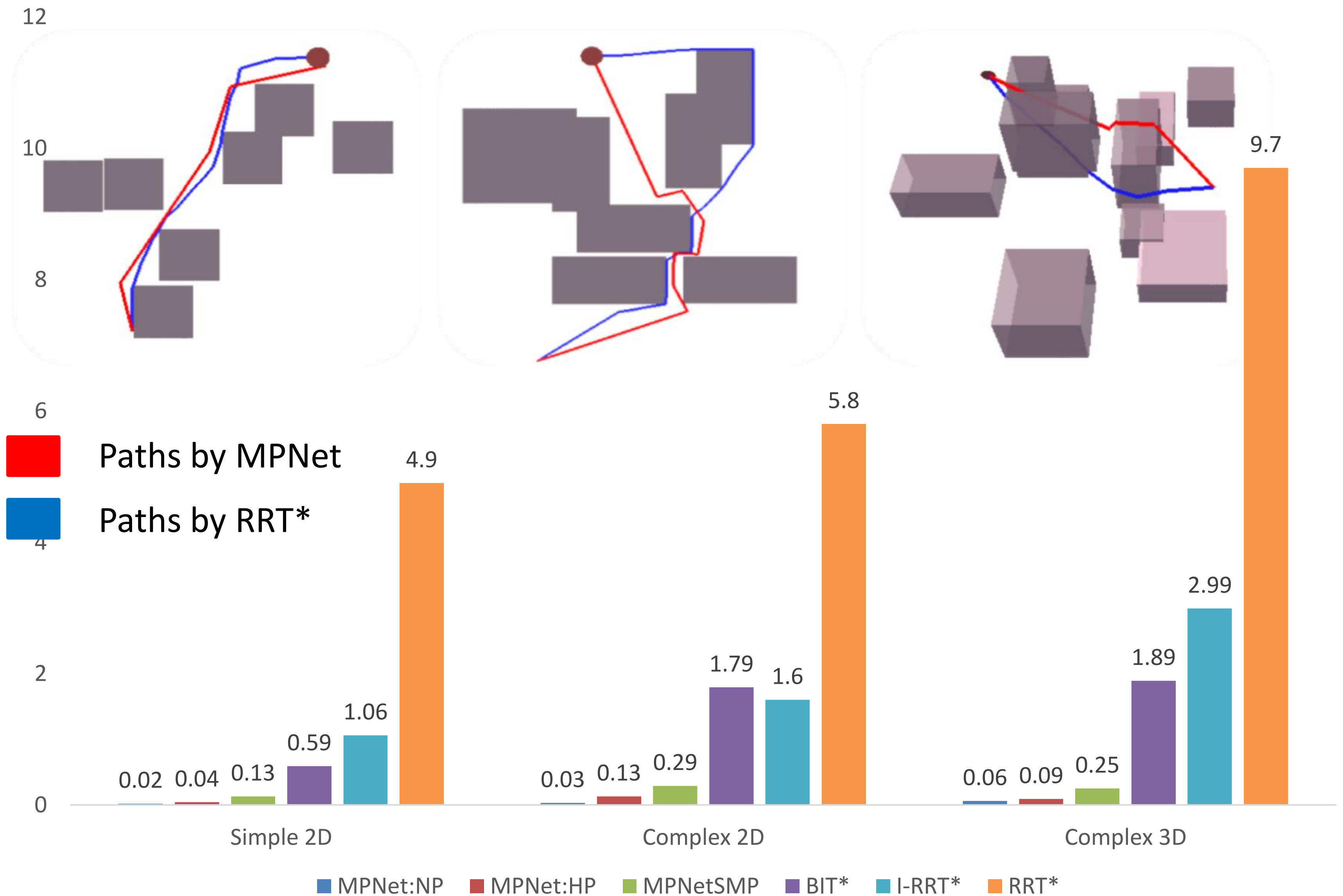


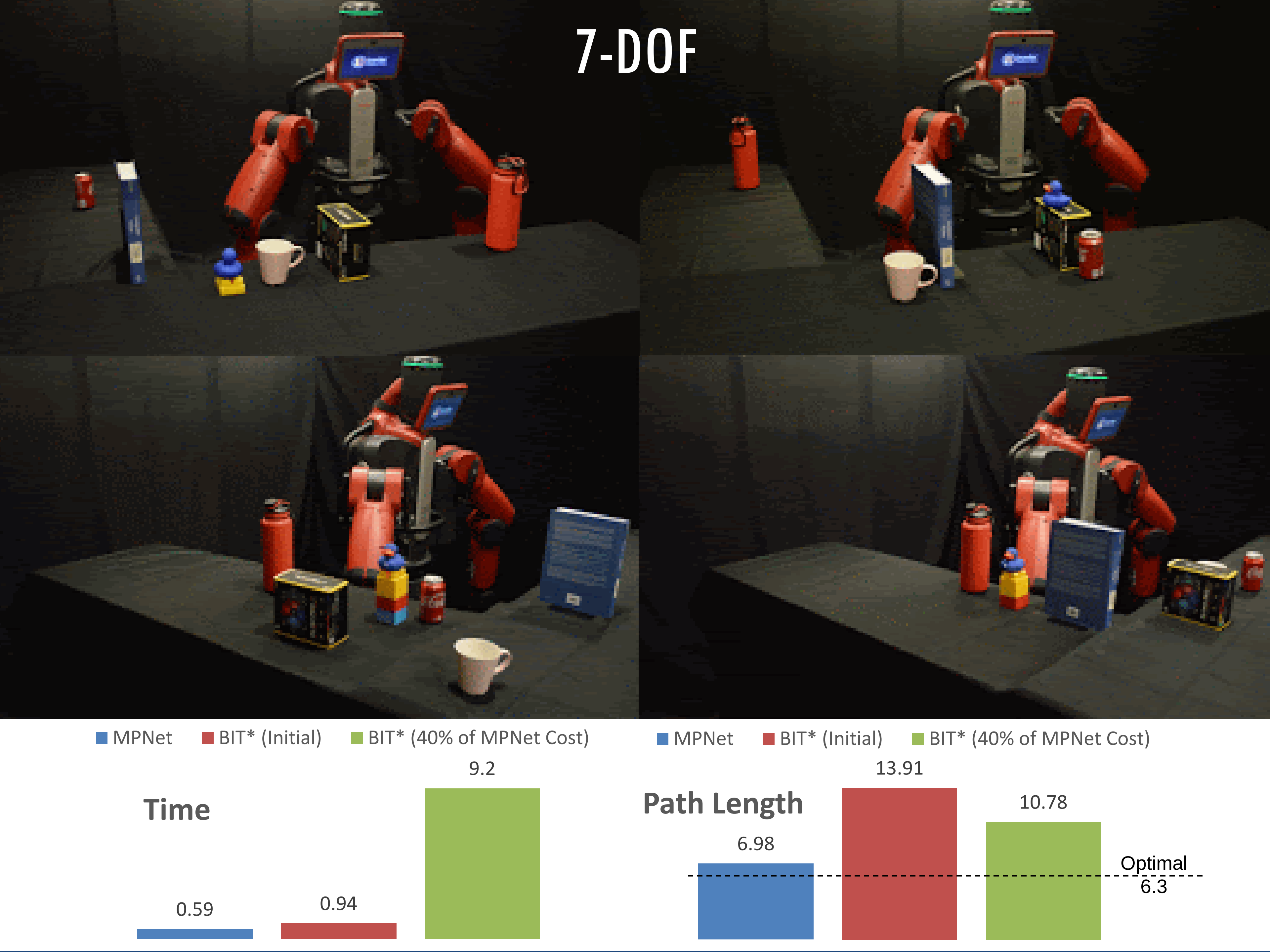
```
Initialize T ← SMP (c_init, c_goal, X_obs)
c_rand ← c_init
Z ← Enet(x_obs)
for i ← 0 to n do
    if i < N_smp then
        c_rand ← MPNet(Z, c_rand, c_goal)
    else
        c_rand ← RandomSampler()
    c_nearest ← Nearest(c_rand, T)
    if Steer(c_nearest, c_rand) then
        C_near ← Near(c_rand, T)
        T ← Rewire(T, C_near, c_rand)
    if c_rand ∈ c_goal then
        c_rand ← c_init
return ∅
```

Informed Sampling Visualization



Results


12
10
8
6
4
2
0
Paths by MPNet
Paths by RRT*
0.02 0.04 0.13 0.59 1.06 4.9
0.03 0.13 0.29 1.79 1.6 5.8
0.06 0.09 0.25 1.89 2.99 9.7
Simple 2D
Complex 2D
Complex 3D
MPNet:NP
MPNet:HP
MPNetSMP
BIT*
I-RRT*
RRT*


7-DOF
MPNet
BIT* (Initial)
BIT* (40% of MPNet Cost)
Time
0.59
0.94
9.2
MPNet
BIT* (Initial)
BIT* (40% of MPNet Cost)
Path Length
6.98
13.91
10.78
Optimal
6.3

Summary of Results

End-to-end collision-free paths with minimal-to-no branching

- Significantly faster than state of art in challenging environments
- Significantly less variance in time-to-completion.
- Near-optimal path length

Decompose planning problems into sub-problems.

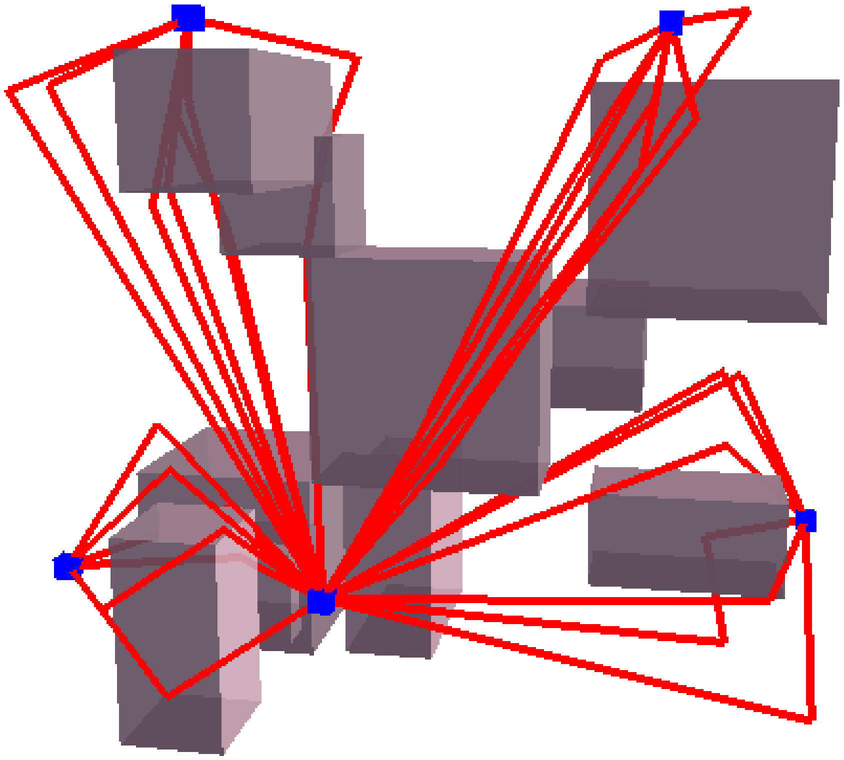
- Allows easy integration with standard planners.
  - retains computational benefits with completeness guarantees.

Can learn from streaming data

- Can actively ask for demonstrations only when needed.
- Reduced data for learning
- minimizes catastrophic forgetting

Validated on variety of environments

- Seen and unseen environments from 2 to 7DOF.

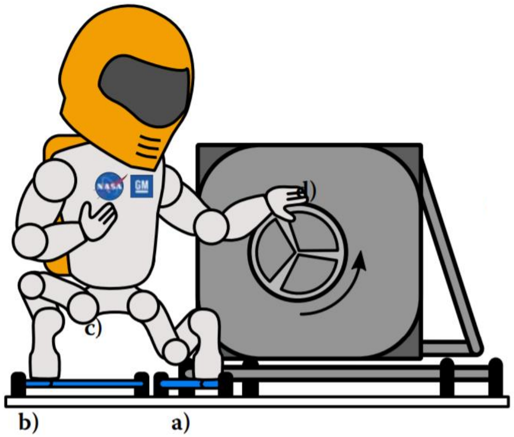


MPNET: EXTENSIONS

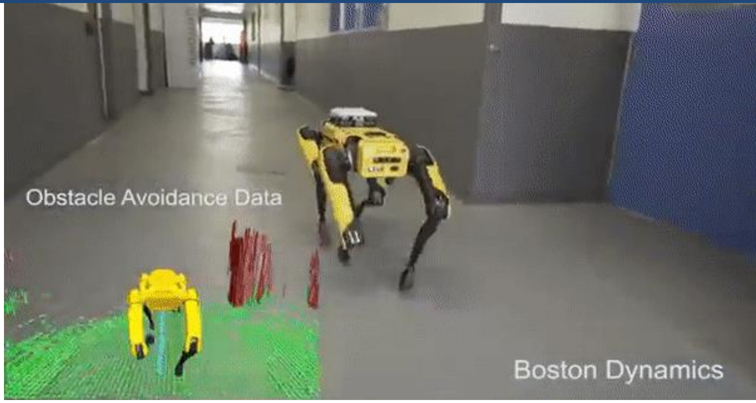
Motion Planning

Find a path that satisfies all constraints between the given start and goal configurations.

- Collision Avoidance
- Kinematics (e.g., end-effector)
- Dynamics

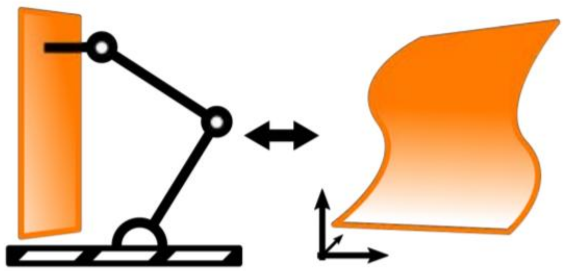


Kingston et. el, 2018

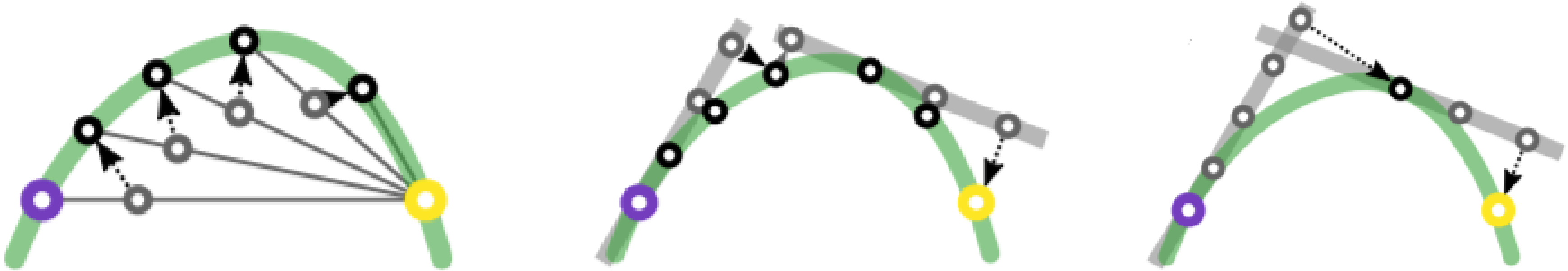


Motion Planning under Kinematic Constraints

- Bi-RRT & Constraint-adherence approaches
  - CBiRRT: Projection [1]
  - Atlas-RRT: Atlas [2]
  - TB-RRT: Tangent Bundle [3]



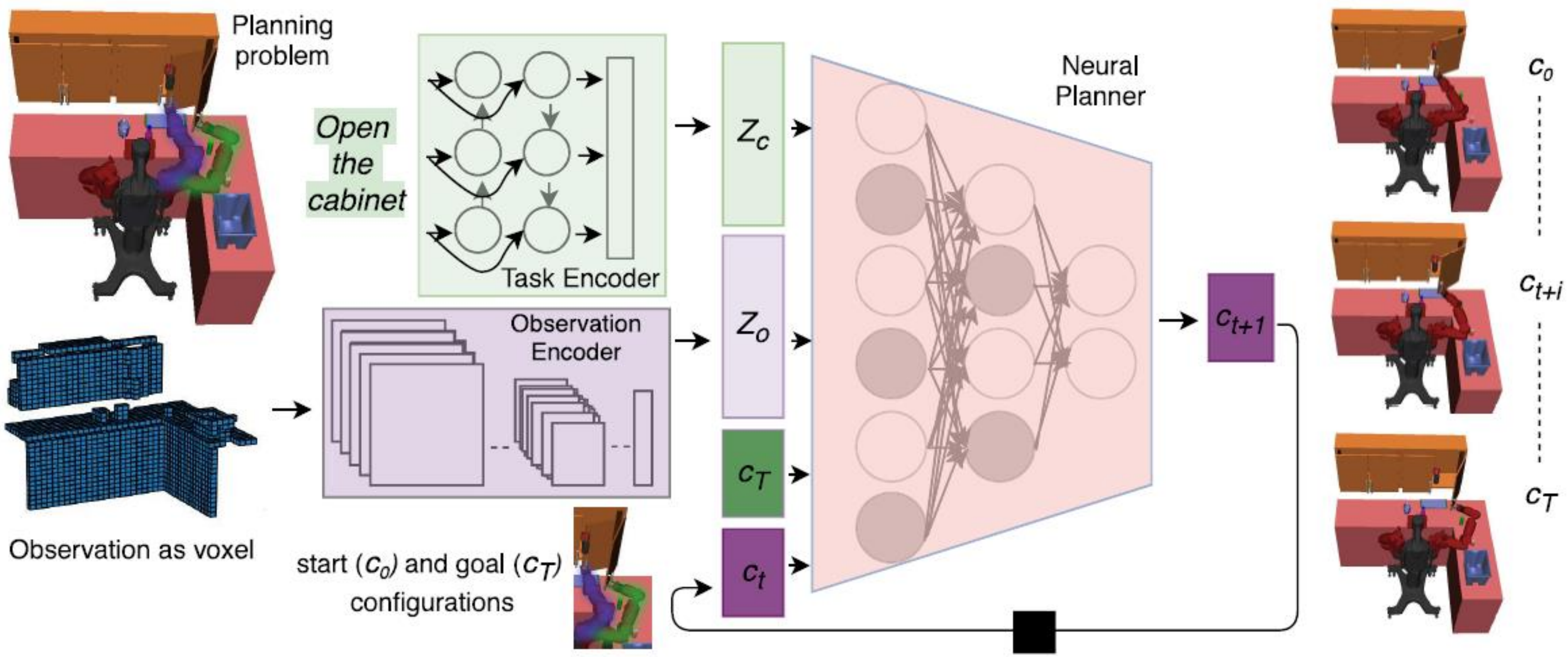
Kingston et. el, 2018, 2019



Projection　　Atlas　　Tangent Bundle

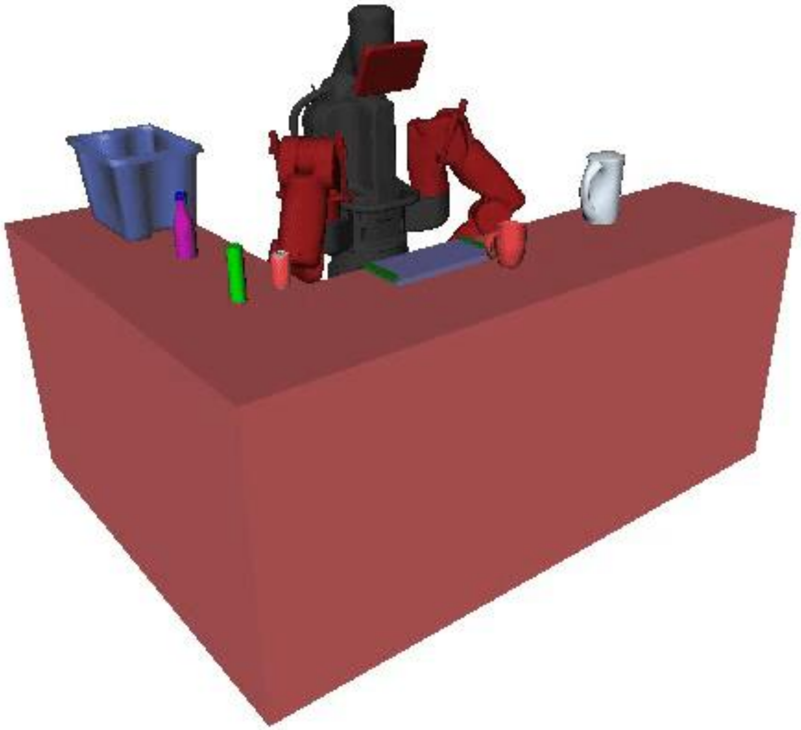
[1] Berenson et. el. (2011). Task space regions: A framework for pose-constrained manipulation planning, IJRR.
[2] Jaillet & Porta (2012). Path planning under kinematic constraints by rapidly exploring manifolds. IEEE TRO.
[3] Kim et. el. (2016). Tangent bundle RRT: A randomized algorithm for constrained motion planning. *Robotica*.

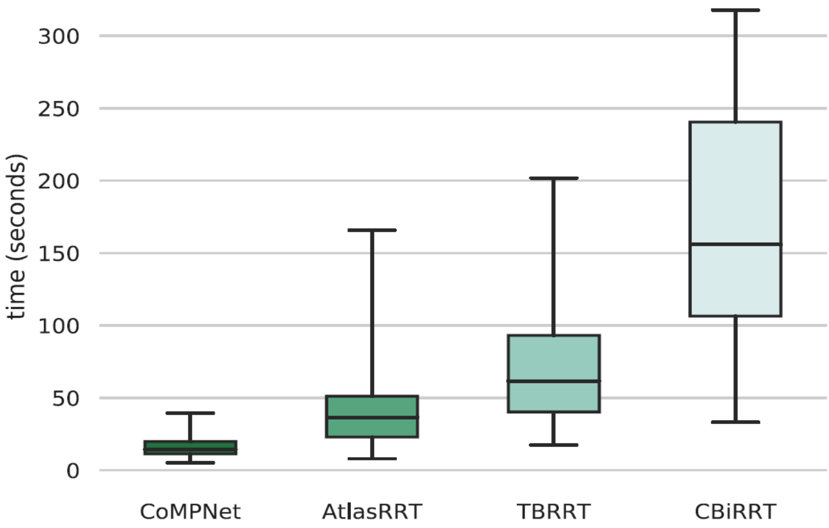
Constrained Motion Planning Networks



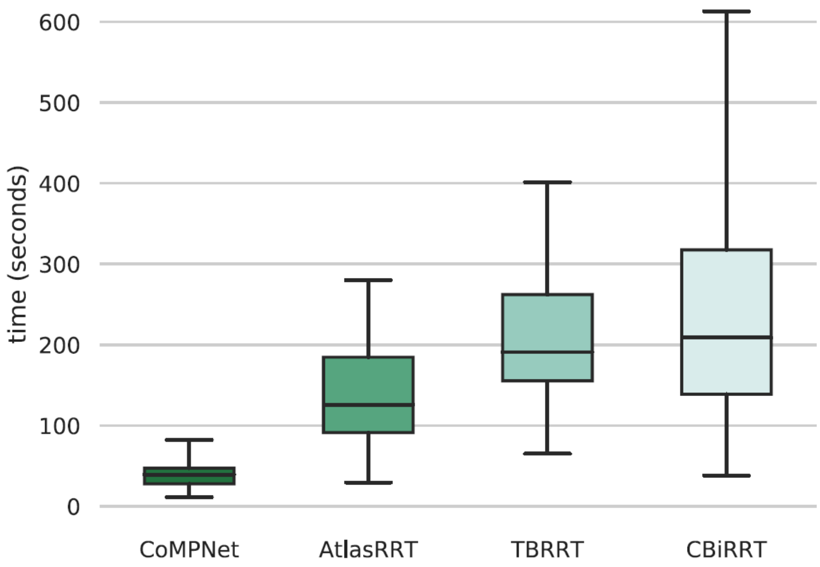
Qureshi et. el, Neural Manipulation Planning on the Constrained Manifolds, RAL 2020.

CoMPNet: Results

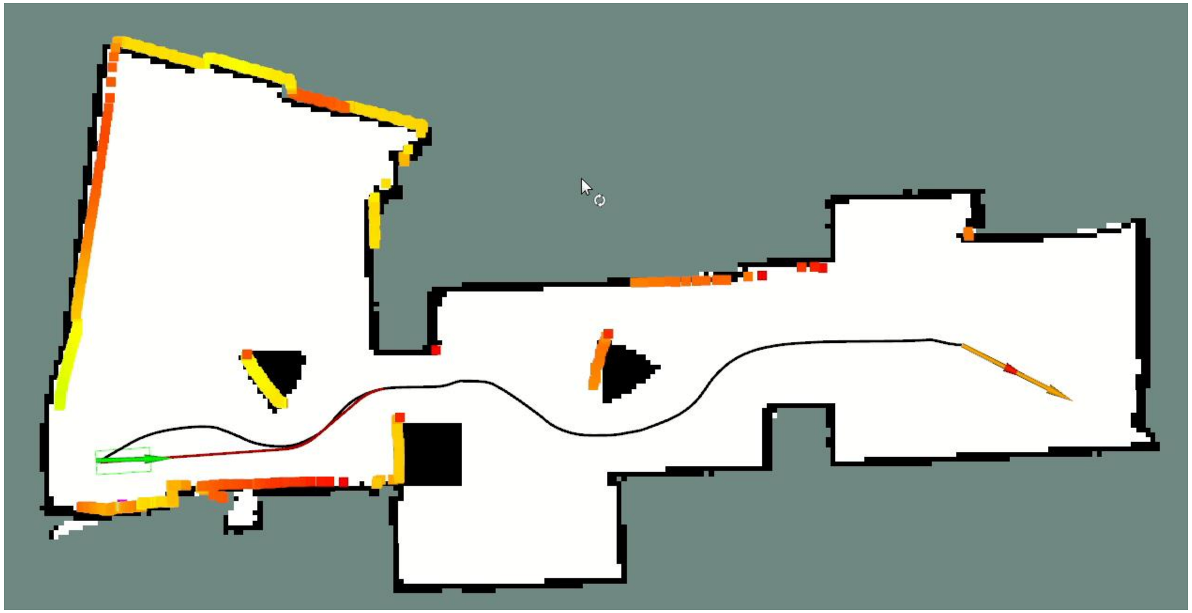




CoMPNet: Results



Dynamically Constrained Motion Planning Networks



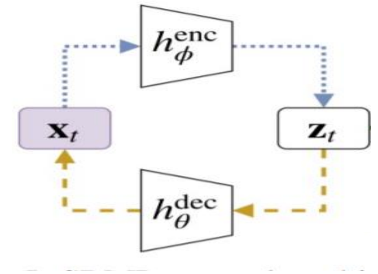
Jacob Johnson

- Real-time planning with egocentric maps
- Generates samples that satisfy non-holonomic constraints
- Provide local planner plugin for ROS navigation stack
  - https://github.com/jacobjj/mpnet_local_planner

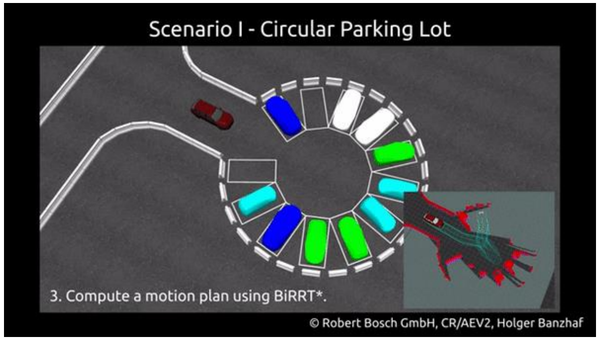
J. Johnson et. el, Dynamically Constrained Motion Planning Networks for Non-Holonomic Robots, IROS2020.

Extensions to Motion Planning Networks / Neural Planning

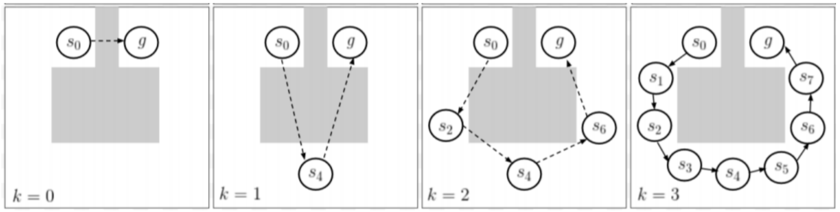
- **Planning in Learned Latent Spaces –** *Ichter and Pavone. RAL 4.3 (2019): 2407-2414.*



- **Sampling-based ego-poses for planning motions of nonholonomic vehicles –** Banzhaf, et al. *RAL* 4.2 (2019): 1053-1060.



- **Subgoal Trees** – Jurgenson, Groshev, Tamar." *ICML* (2019).



- **Harnessing Reinforcement Learning for Neural Motion Planning** – Jurgenson, Tamar." *RSS* (2019).

Publications

[1] **A.H.Qureshi**, Y.Miao, A.Simeonov, and M.C.Yip. “Motion Planning Networks: Bridging the Gap Between Learning-based and Classical Motion Planners”, *IEEE Transactions on Robotics (TRO)*, 2020.

[3] **A.H.Qureshi**, A.Simeonov, M.J.Bency, M.C.Yip. “Motion Planning Networks”, *IEEE/RAS International Conference on Robotics and Automation (ICRA)*, pp. 2118-2124, Montreal, Canada 2019.

[4] **A.H.Qureshi** and Michael.C.Yip . “Deeply Informed Neural Sampling For Robot Motion Planning”, *Proceedings of IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS)*, pp. 6582-6588, Madrid, Spain 2018.

[5] **A.H.Qureshi**, J.Dong, A.Choe, and M.C.Yip. “Neural Manipulation Planning on Constraints Manifolds”, *IEEE Robotics and Automation Letters (RAL)*, 2020.

[5] J.Johnson, L.Jun, F.Liu, **A.H.Qureshi** and M.C.Yip. “Dynamically Constrained Motion Planning Networks for Non-Holonomic Robots”, *IEEE IROS*, 2020.

THANK YOU