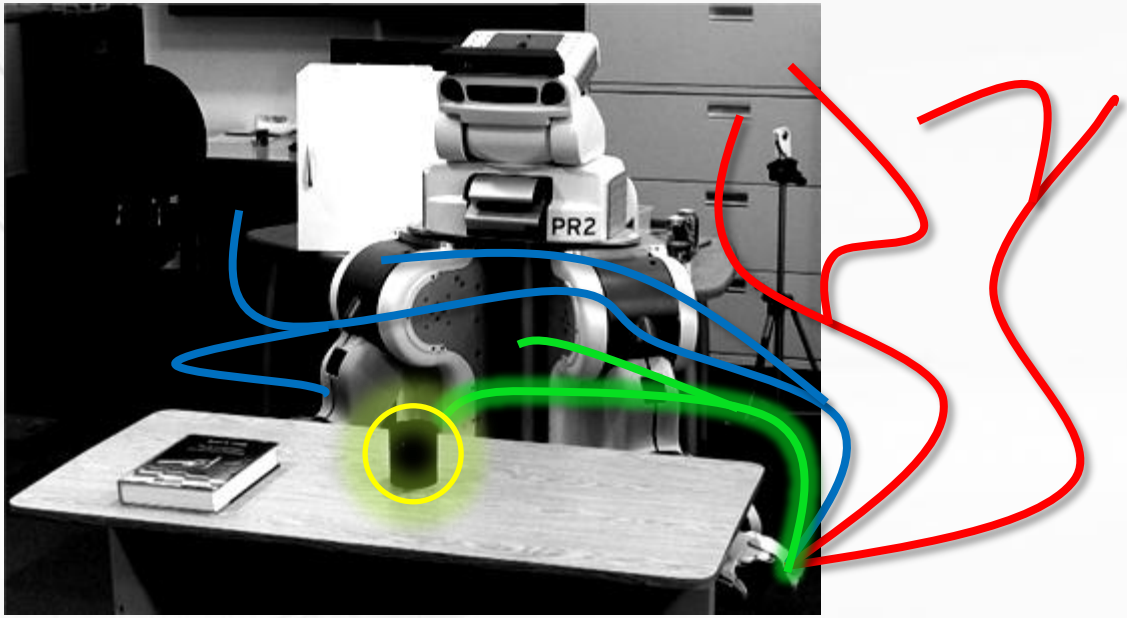
Asymptotically-optimal Manipulation Planning using Incremental Sampling-based Algorithms



Alejandro Perez [1]

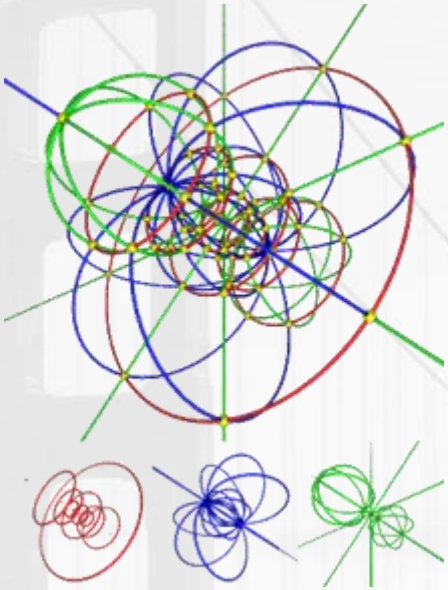
Sertac Karaman [2], Alexander Shkolnik [1],

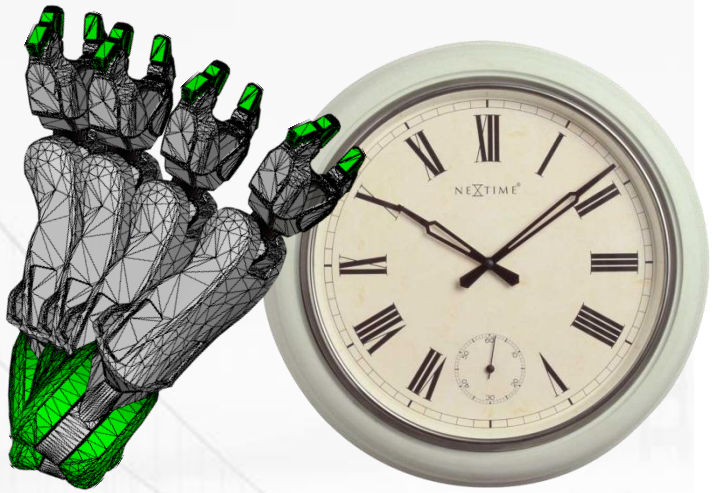
Emilio Frazzoli [2], Seth Teller [1] and Matt Walter [1]

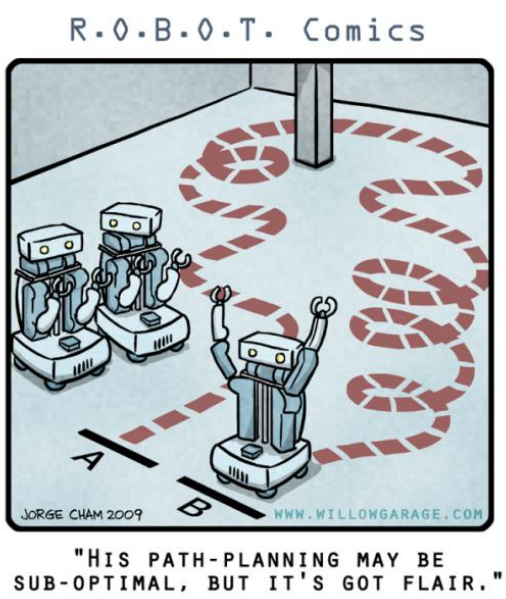
[1] MIT/CSAIL [2] MIT/LIDS

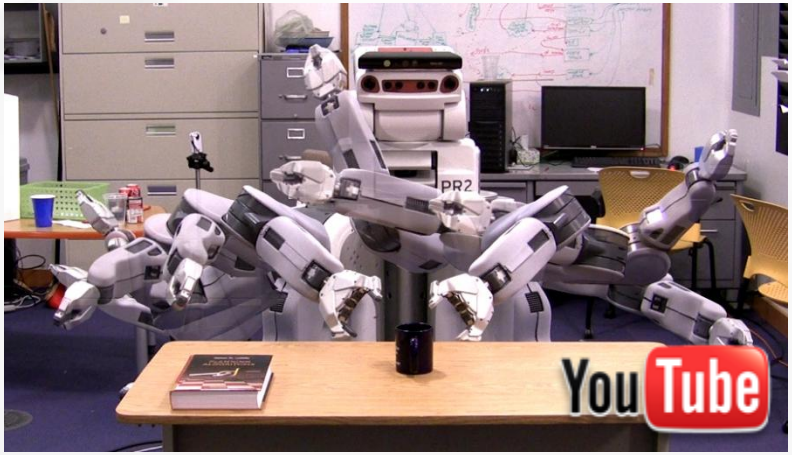
IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

Manipulation Planning

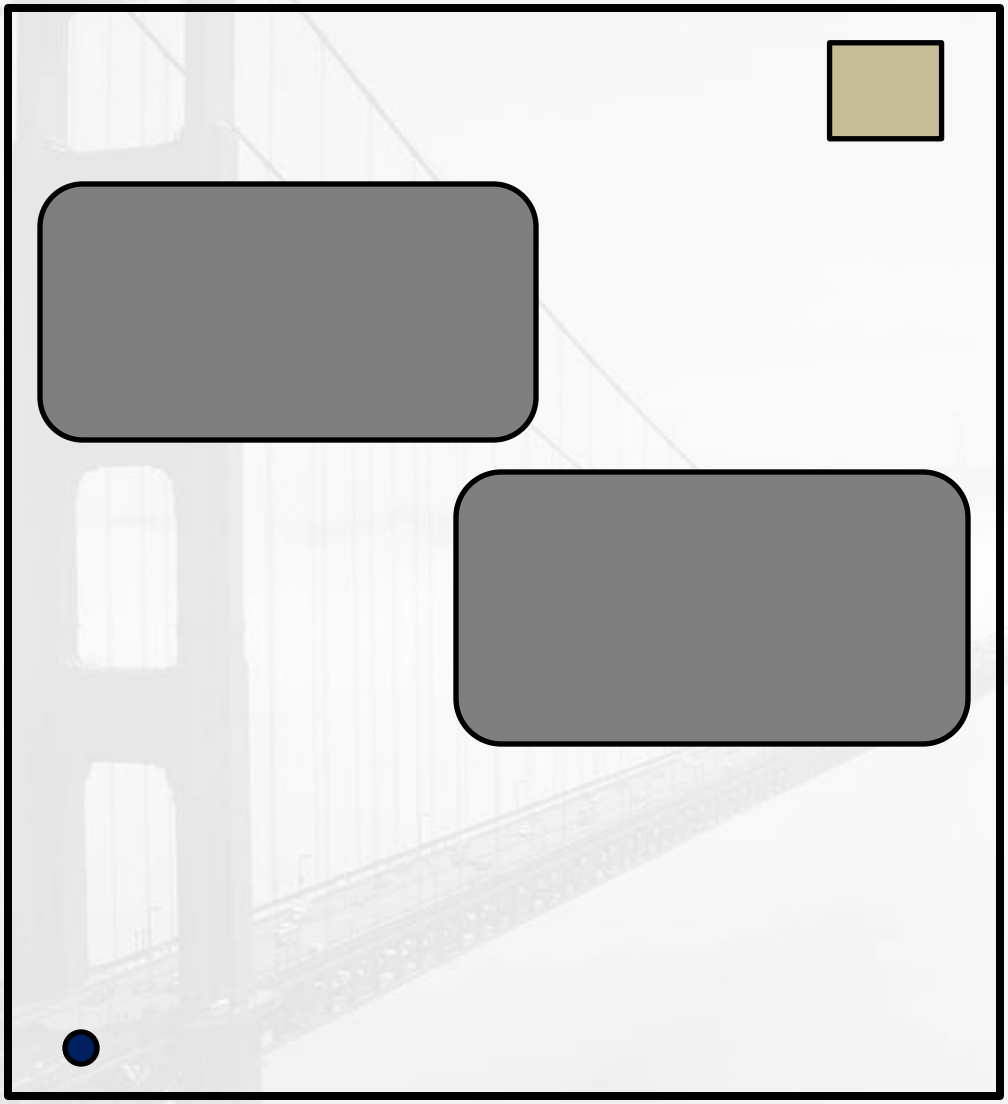








RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

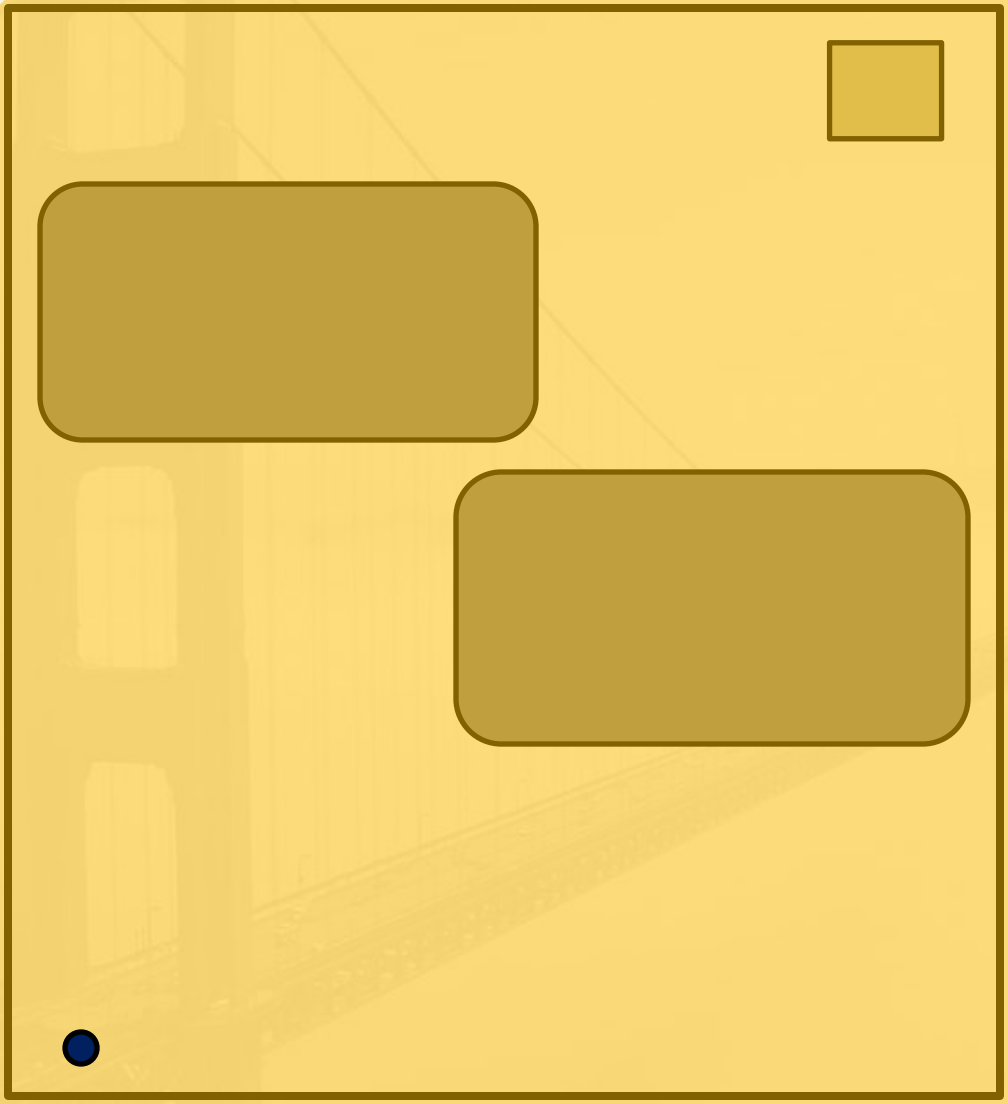
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add($(x_{\text{parent}}, x_{\text{new}})$);
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

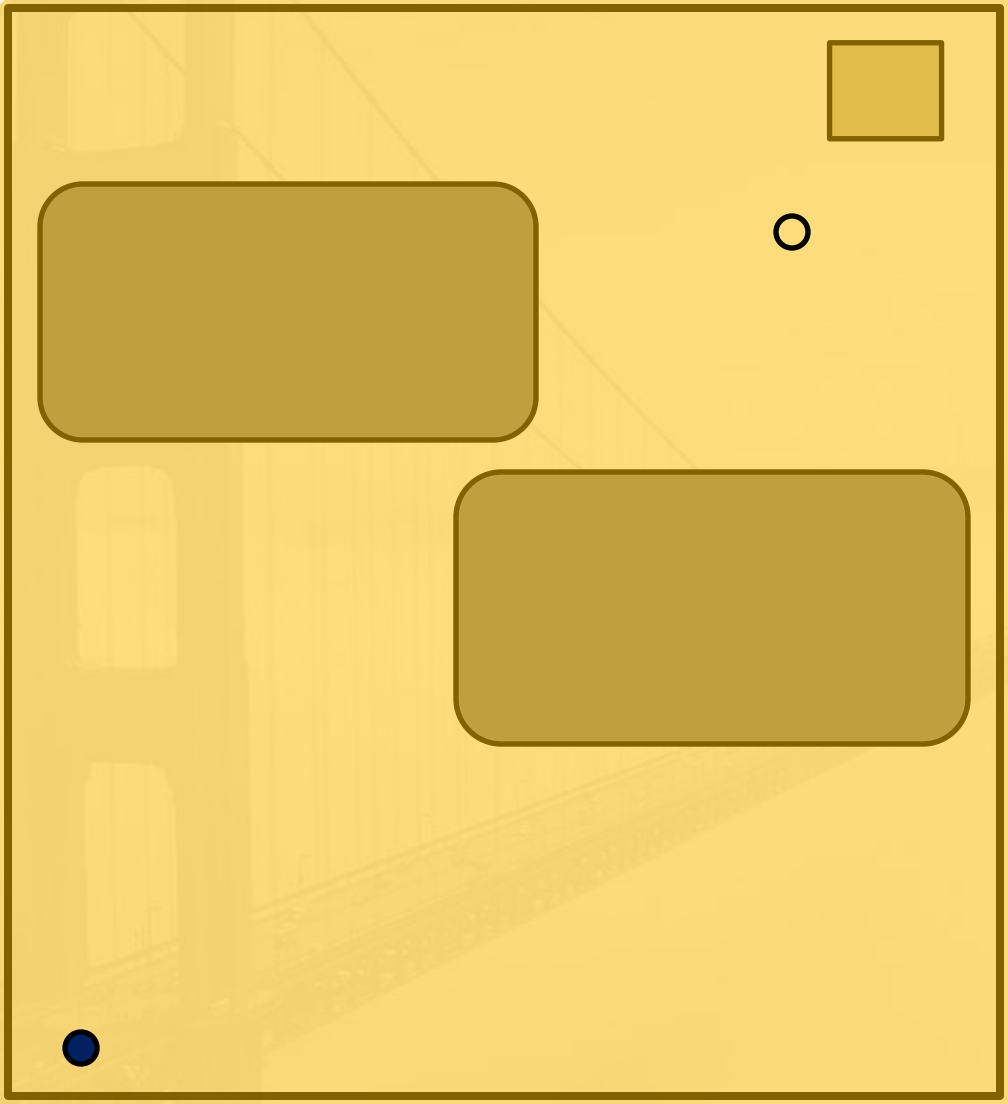
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if $\texttt{InsideBall}(x_{\text{new}}, T)$ **then**
      if !$\texttt{CollisionFree}(x_{\text{new}})$ **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        $\texttt{TrimRadius}\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if $\texttt{CollisionFree}(\sigma)$ **then**
    V.add(x_{new}, r_0);
    E.add($(x_{\text{nearest}}, x_{\text{new}})$);
  else
    $\texttt{TrimRadius}\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



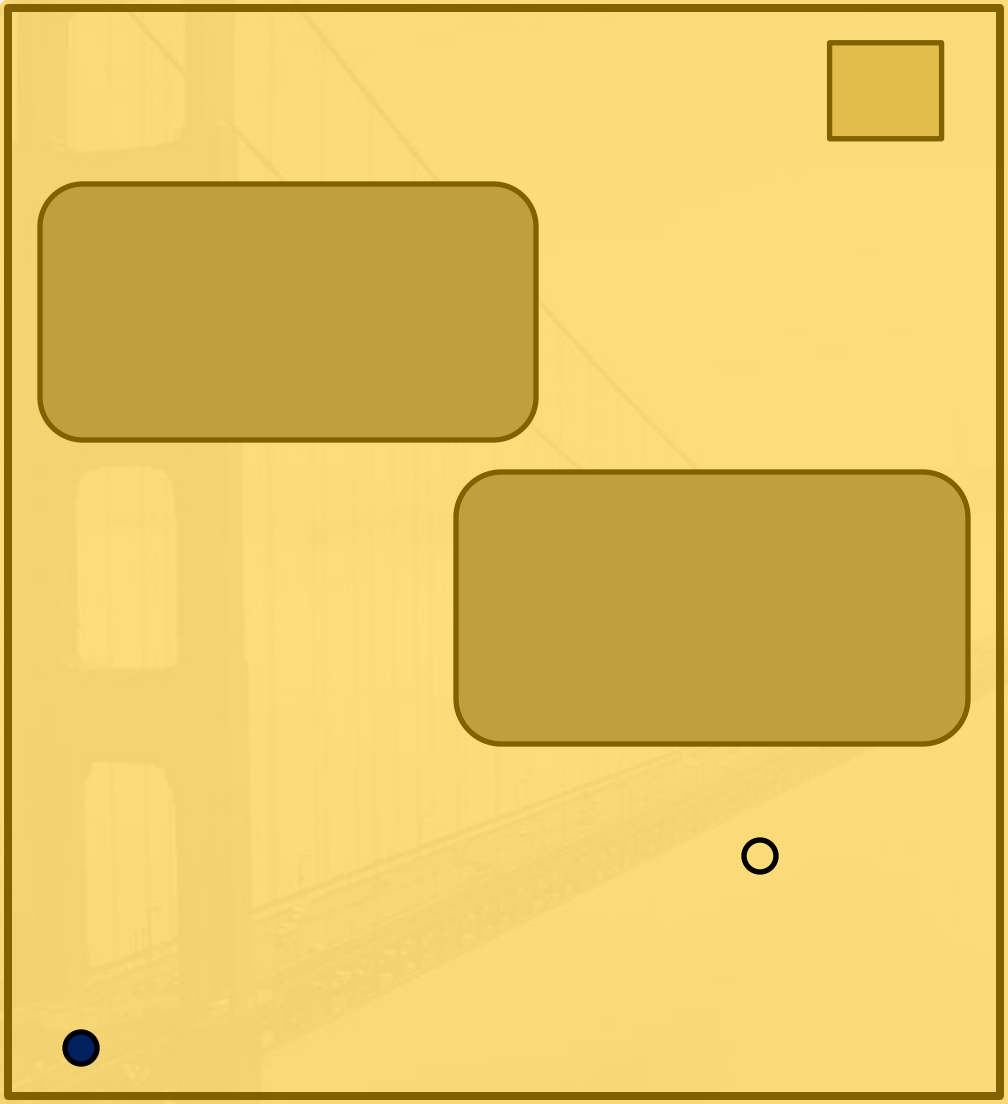
The RRT* Algorithm

```
V ← {x_init}; E ← ∅; T ← (V, E);
for i = 1 to N do
    x_new ← Sample(i);
    X_near ← Near(V, x_new);
    if X_near = ∅ then
        X_near ← Nearest(V, x_new);
    L_near ← PopulateSortedList(X_near, x_near);
    x_parent ← FindBestParent(L_near, x_new);
    if x_parent ≠ NULL then
        V.add(x_new);
        E.add( (x_parent, x_new) );
        E ← RewireVertices(E, X_near, x_new);
return T = (V, E).
```

The Ball Tree Algorithm

```
V ← {x_init, r = 0}; E ← ∅; T ← (V, E);
for i = 1 to N do
    while true do
        x_new ← Sample(i);
        if InsideBall(x_new, T) then
            if !CollisionFree(x_new) then
                x_nearest ← NearestBall(V, x_new);
                TrimRadius (x_nearest, ||x_new − x_nearest||);
        else
            break;
    x_nearest ← NearestBall(V, x_new);
    σ ← Steer(x_new, x_nearest);
    if CollisionFree(σ) then
        V.add(x_new, r_0);
        E.add( (x_nearest, x_new) );
    else
        TrimRadius (x_nearest, ||x_new − x_nearest||);
return T = (V, E).
```

RRT* with Ball Trees



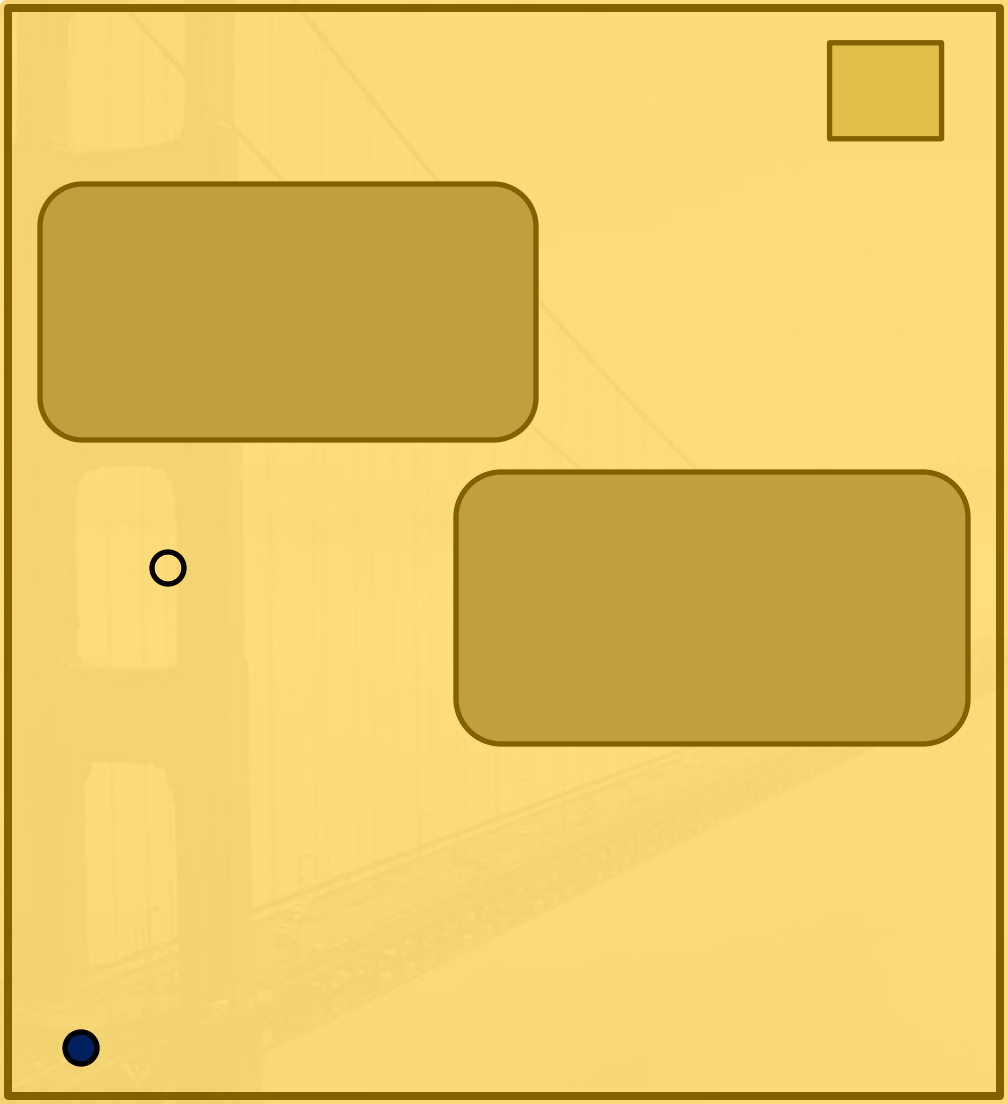
The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add($(x_{\text{parent}}, x_{\text{new}})$);
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if $\texttt{InsideBall}(x_{\text{new}}, T)$ **then**
      if !$\texttt{CollisionFree}(x_{\text{new}})$ **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        $\texttt{TrimRadius}\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if $\texttt{CollisionFree}(\sigma)$ **then**
    V.add(x_{new}, r_0);
    E.add($(x_{\text{nearest}}, x_{\text{new}})$);
  else
    $\texttt{TrimRadius}\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

RRT* with Ball Trees



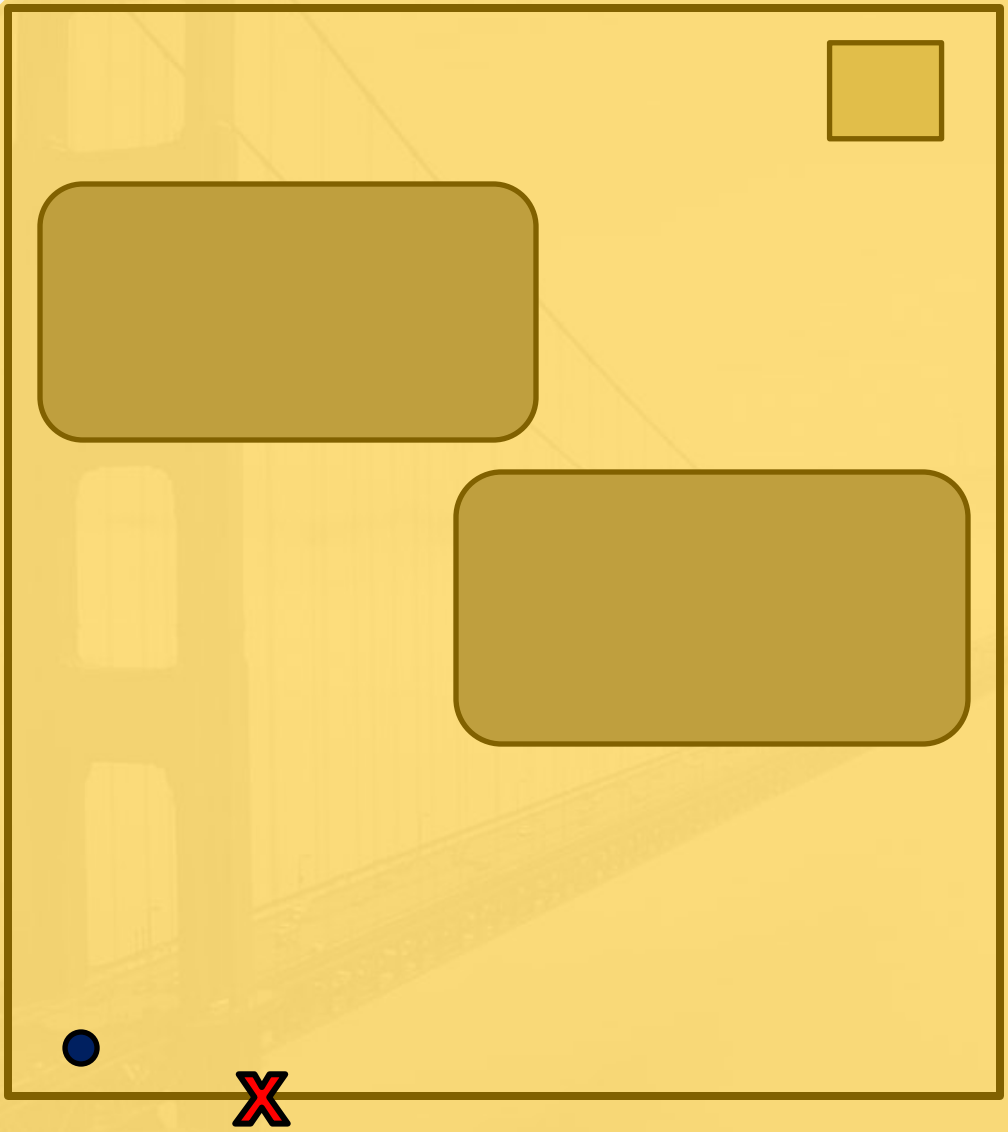
The RRT* Algorithm

```
V ← {x_init}; E ← ∅; T ← (V, E);
for i = 1 to N do
    x_new ← Sample(i);
    X_near ← Near(V, x_new);
    if X_near = ∅ then
        X_near ← Nearest(V, x_new);
    L_near ← PopulateSortedList(X_near, x_near);
    x_parent ← FindBestParent(L_near, x_new);
    if x_parent ≠ NULL then
        V.add(x_new);
        E.add( (x_parent, x_new) );
        E ← RewireVertices(E, X_near, x_new);
return T = (V, E).
```

The Ball Tree Algorithm

```
V ← {x_init, r = 0}; E ← ∅; T ← (V, E);
for i = 1 to N do
    while true do
        x_new ← Sample(i);
        if InsideBall(x_new, T) then
            if !CollisionFree(x_new) then
                x_nearest ← NearestBall(V, x_new);
                TrimRadius (x_nearest, ||x_new − x_nearest||);
        else
            break;
    x_nearest ← NearestBall(V, x_new);
    σ ← Steer(x_new, x_nearest);
    if CollisionFree(σ) then
        V.add(x_new, r_0);
        E.add( (x_nearest, x_new) );
    else
        TrimRadius (x_nearest, ||x_new − x_nearest||);
return T = (V, E).
```

RRT* with Ball Trees



The RRT* Algorithm

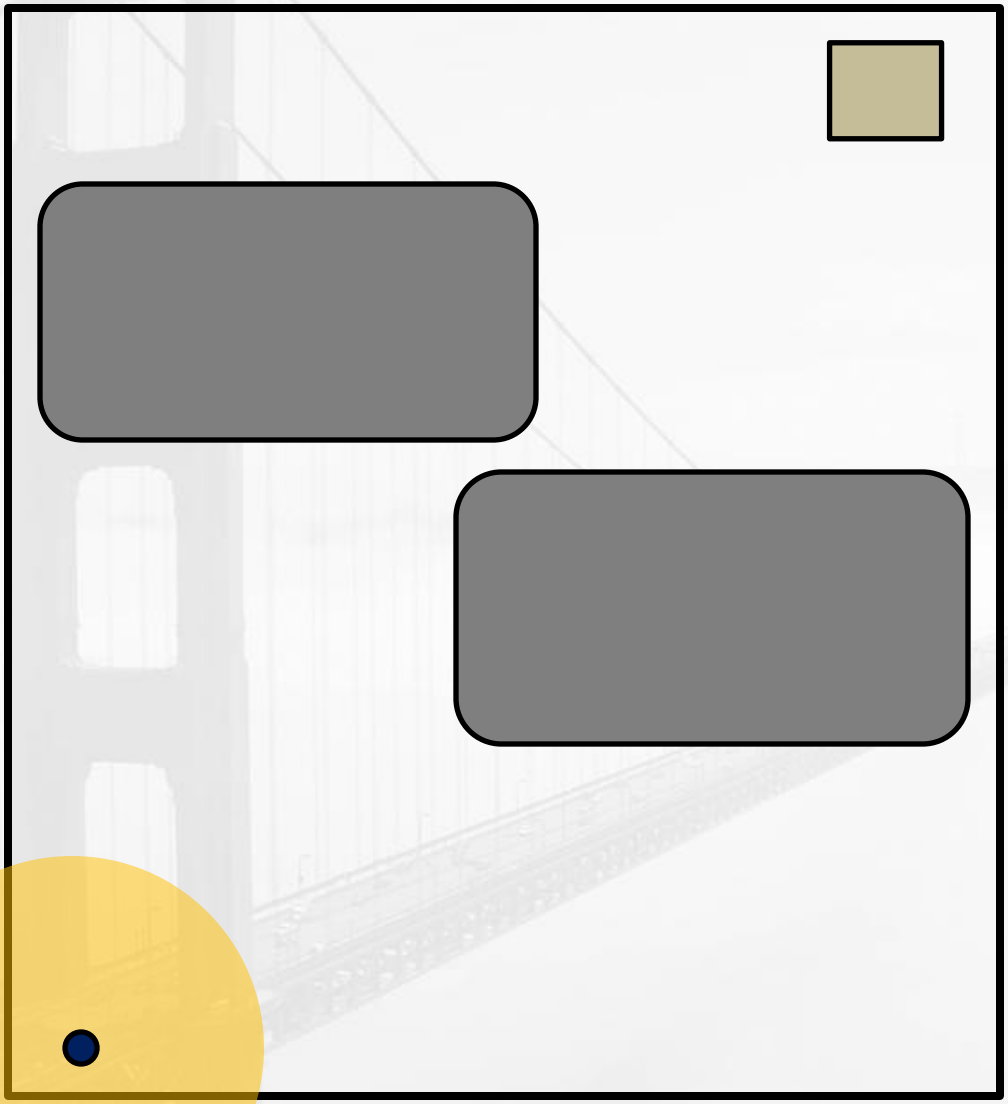
$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add($(x_{\text{parent}}, x_{\text{new}})$);
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if $\texttt{InsideBall}(x_{\text{new}}, T)$ **then**
      if !$\texttt{CollisionFree}(x_{\text{new}})$ **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        $\texttt{TrimRadius}\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if $\texttt{CollisionFree}(\sigma)$ **then**
    V.add(x_{new}, r_0);
    E.add($(x_{\text{nearest}}, x_{\text{new}})$);
  else
    $\texttt{TrimRadius}\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

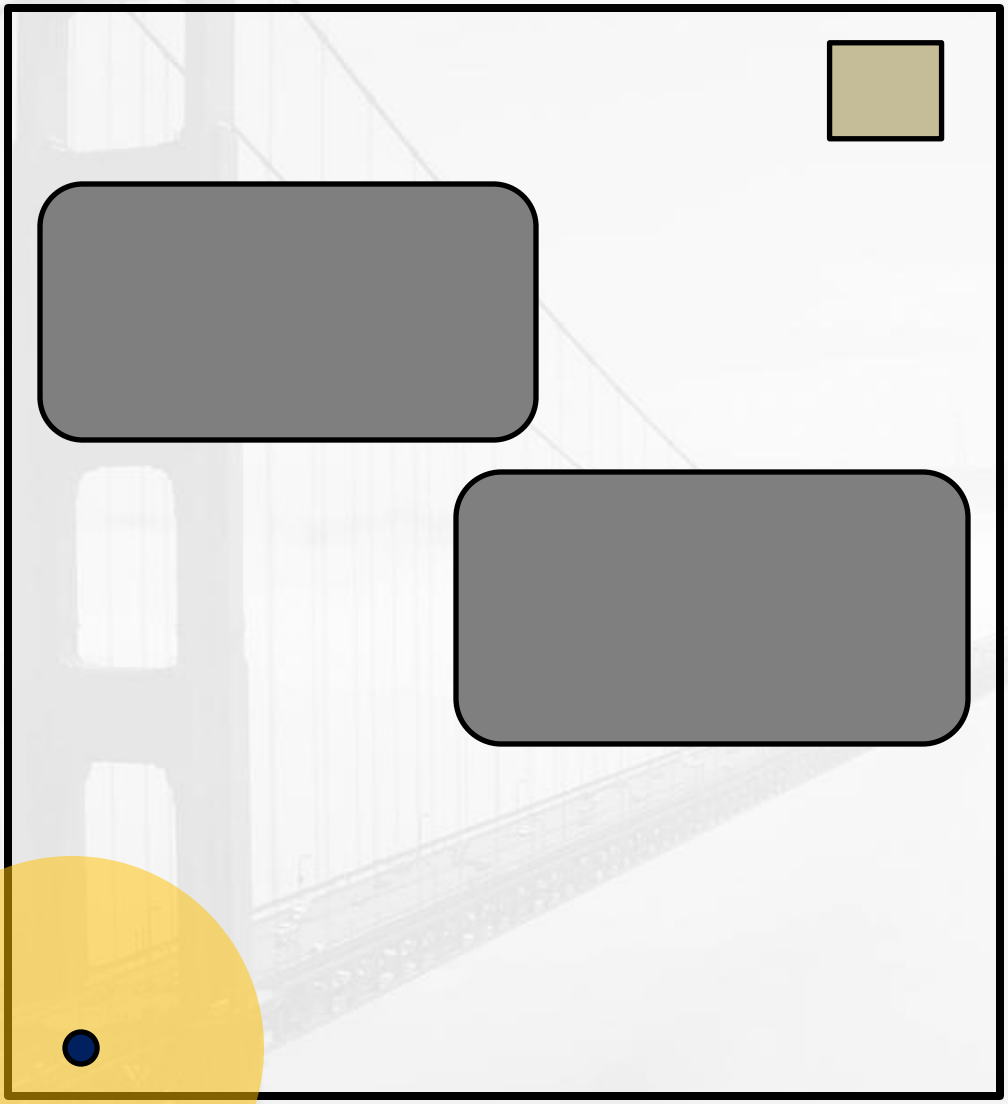
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

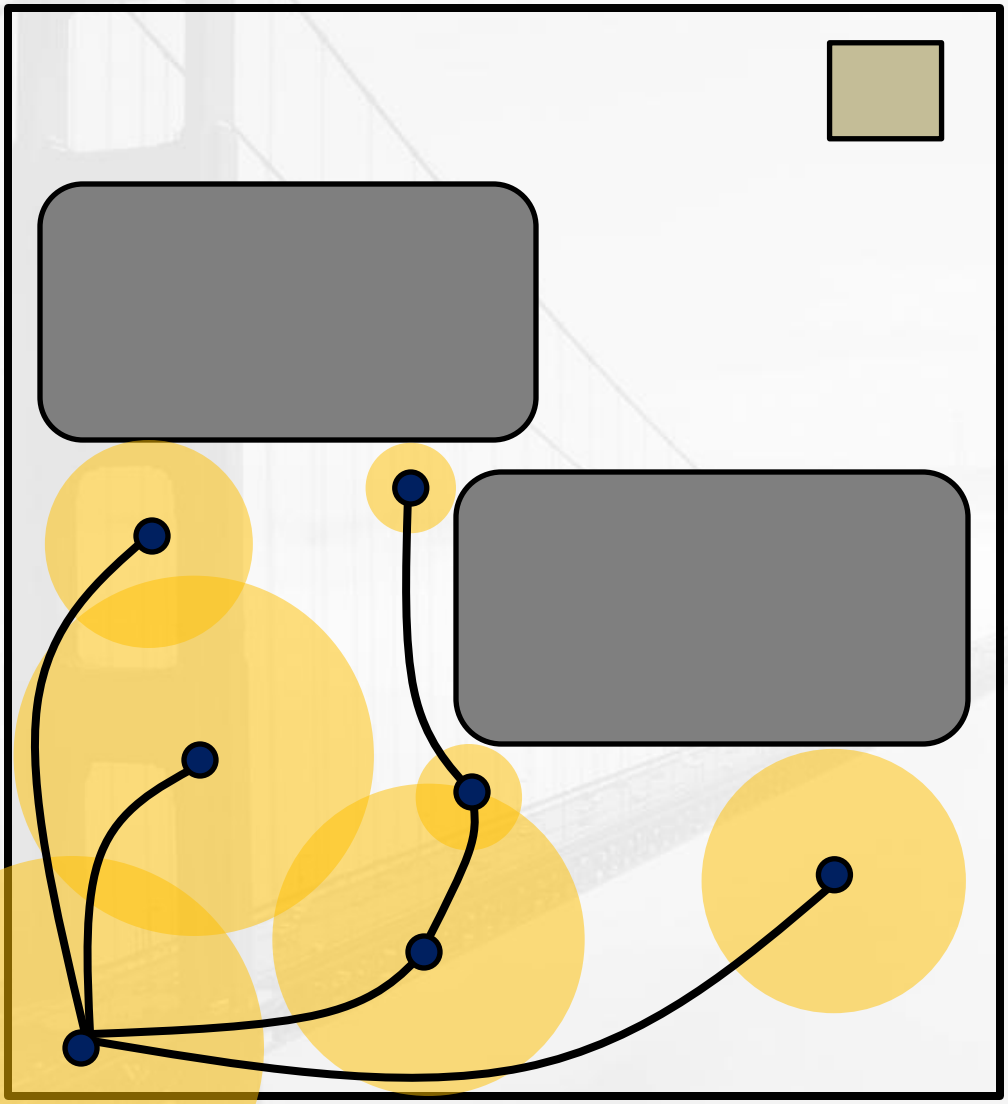
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

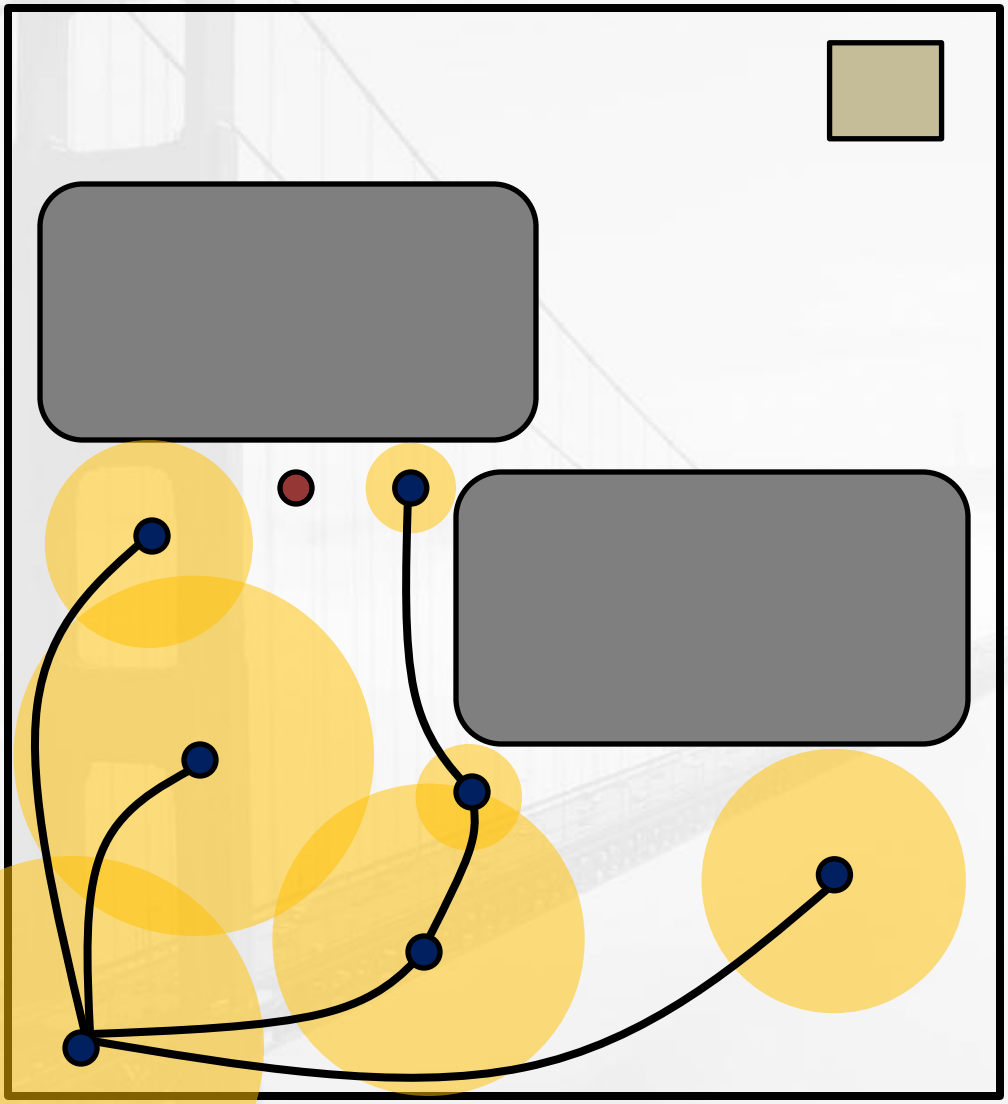
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

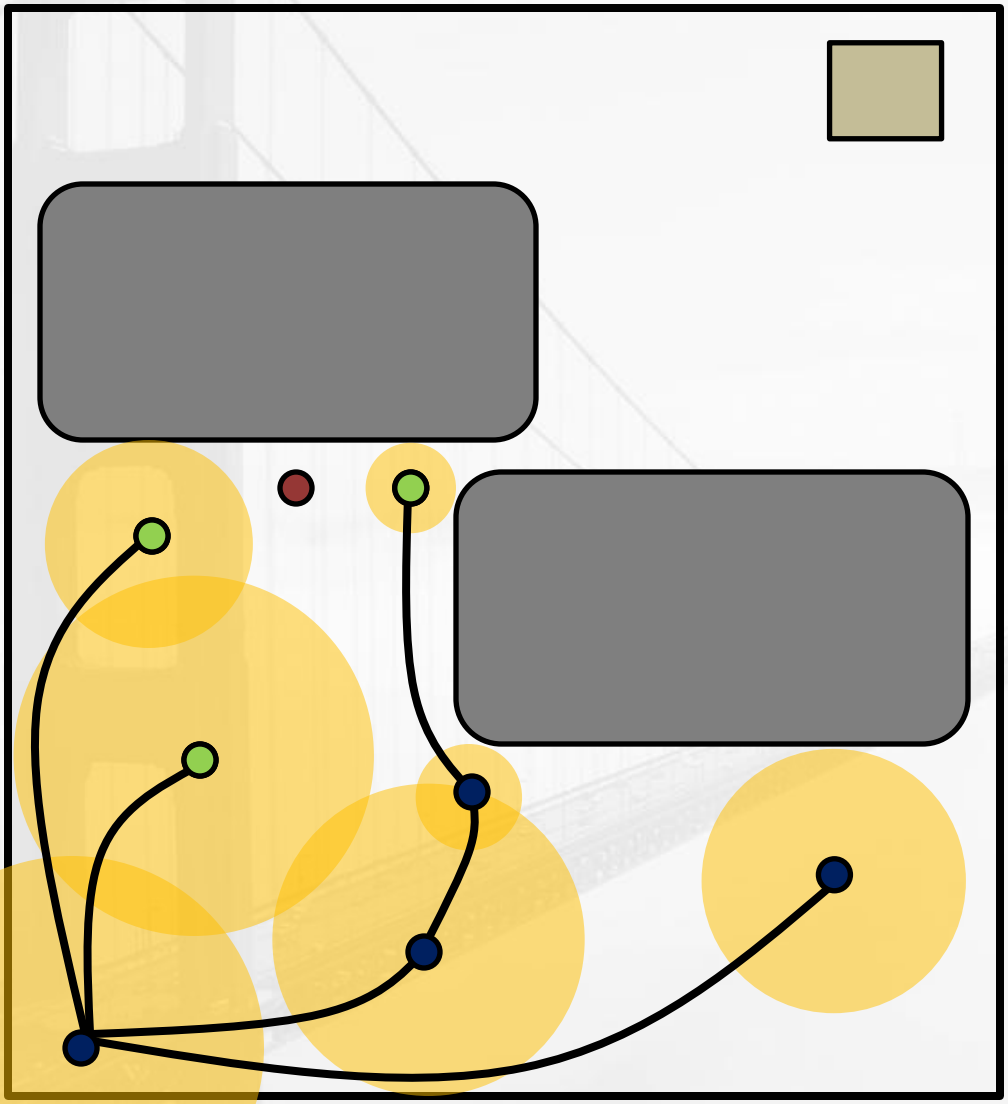
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

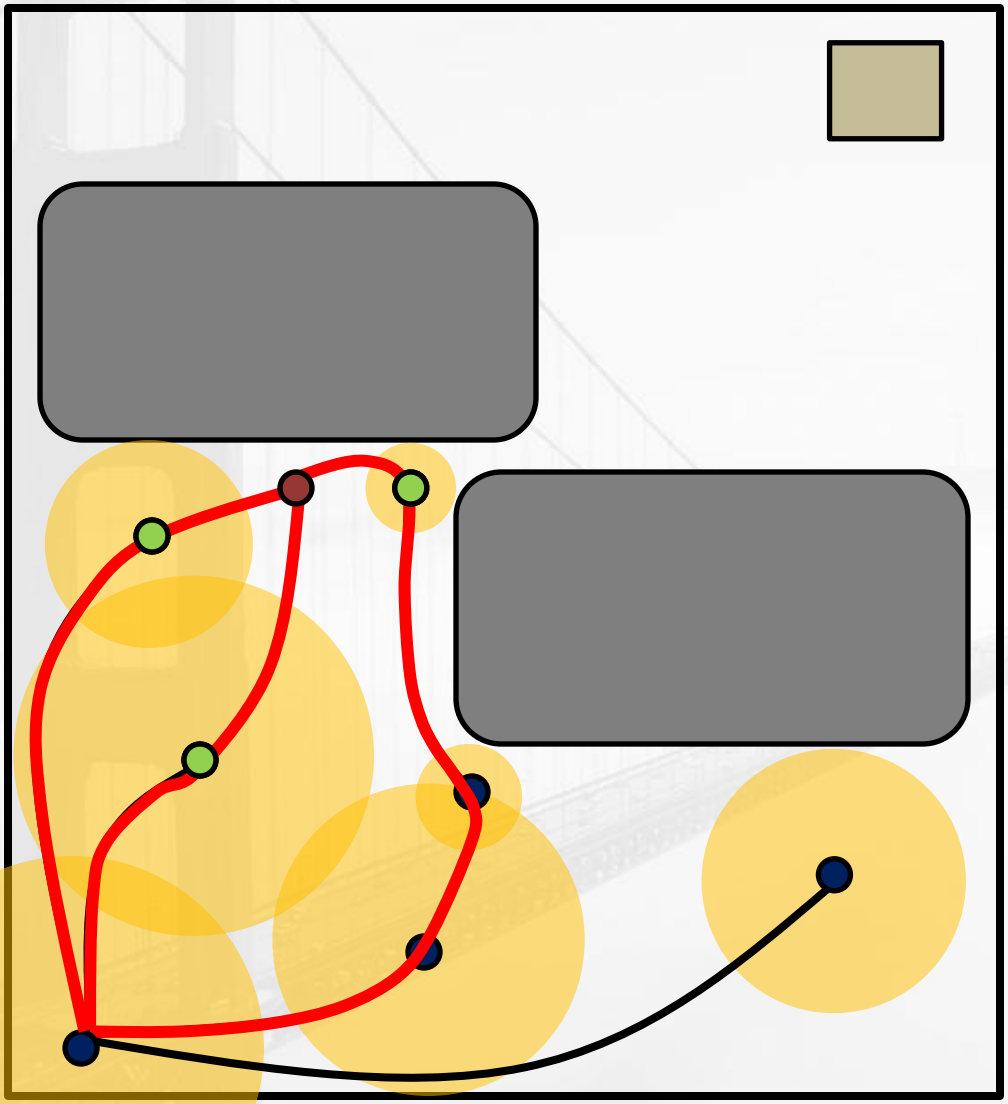
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

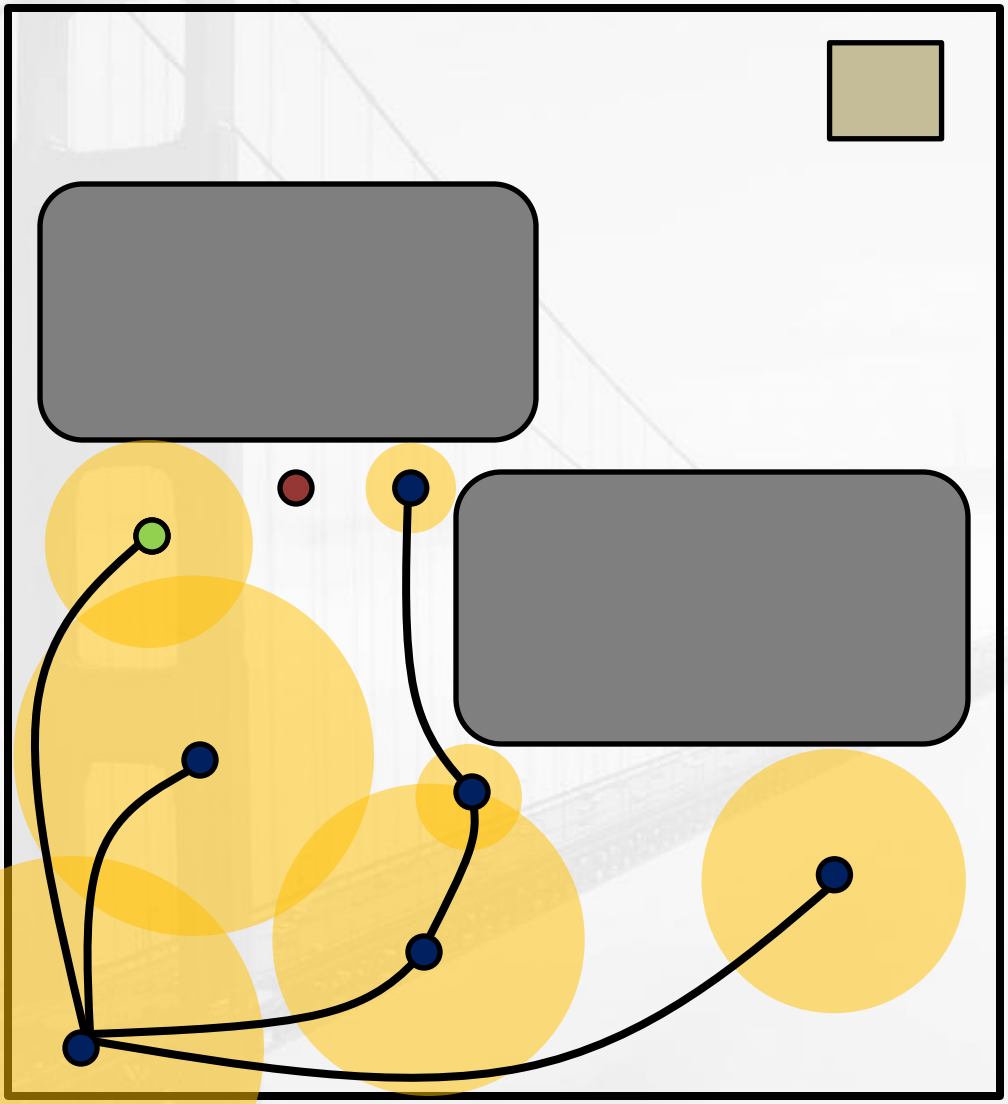
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

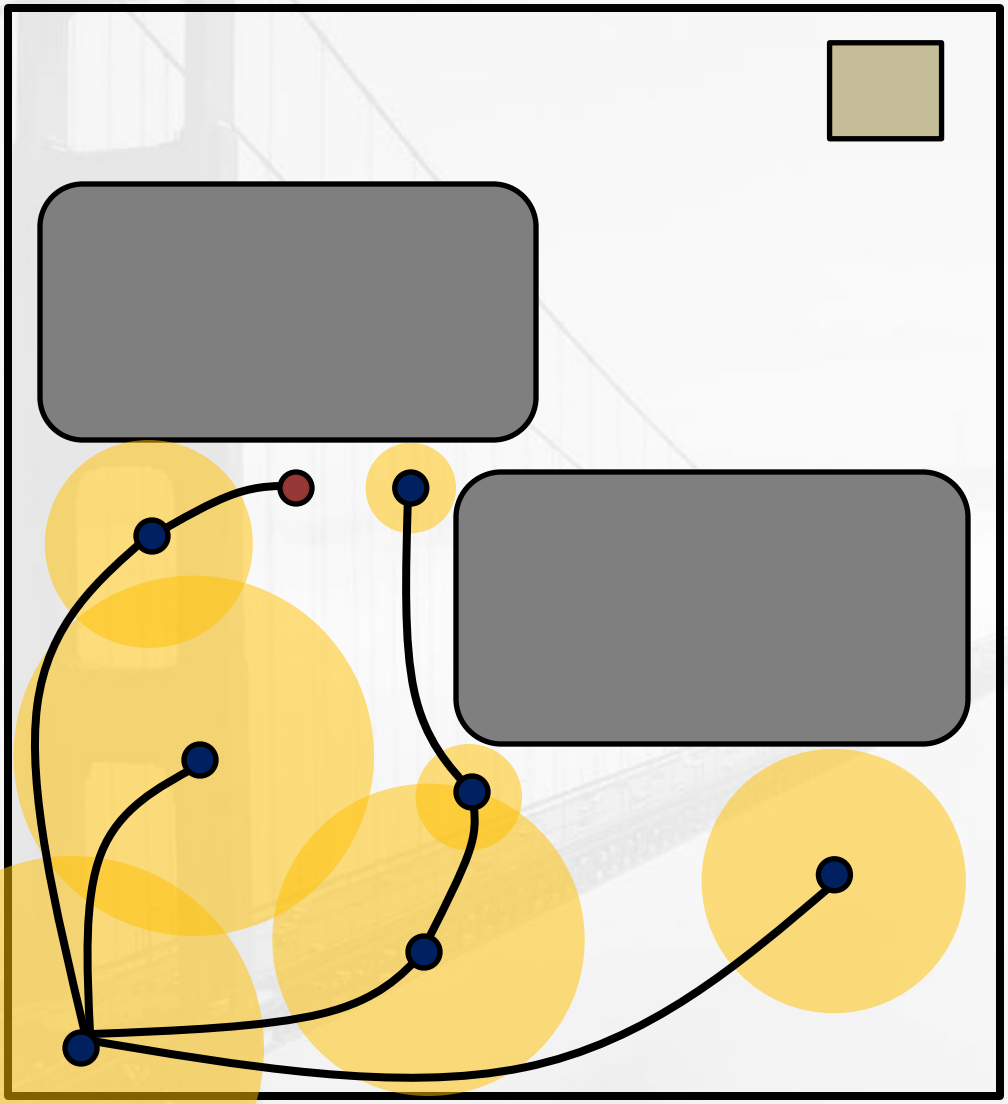
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$((x_{\text{parent}}, x_{\text{new}}))$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

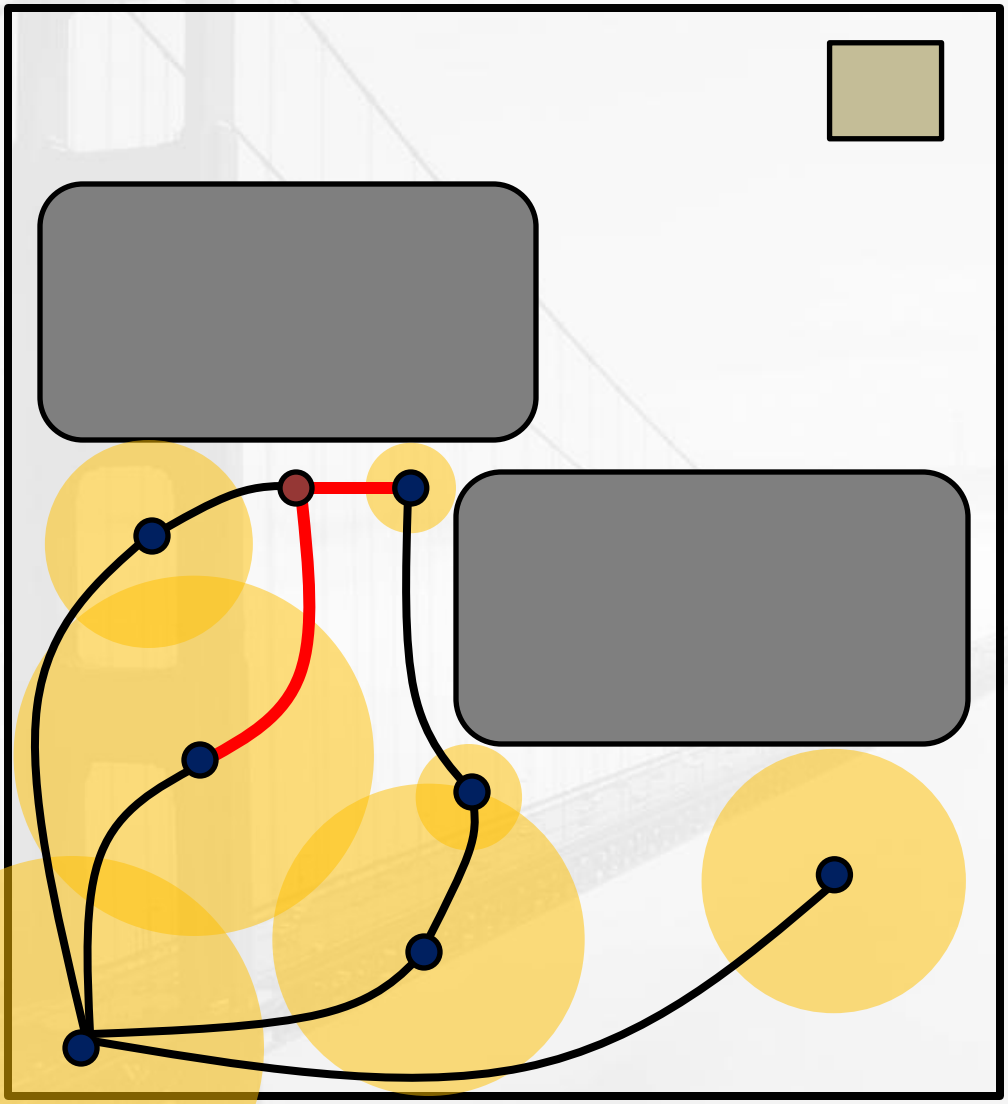
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if $\texttt{InsideBall}(x_{\text{new}}, T)$ **then**
      if !$\texttt{CollisionFree}(x_{\text{new}})$ **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        $\texttt{TrimRadius}(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if $\texttt{CollisionFree}(\sigma)$ **then**
    V.add(x_{new}, r_0);
    E.add$((x_{\text{nearest}}, x_{\text{new}}))$;
  else
    $\texttt{TrimRadius}(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

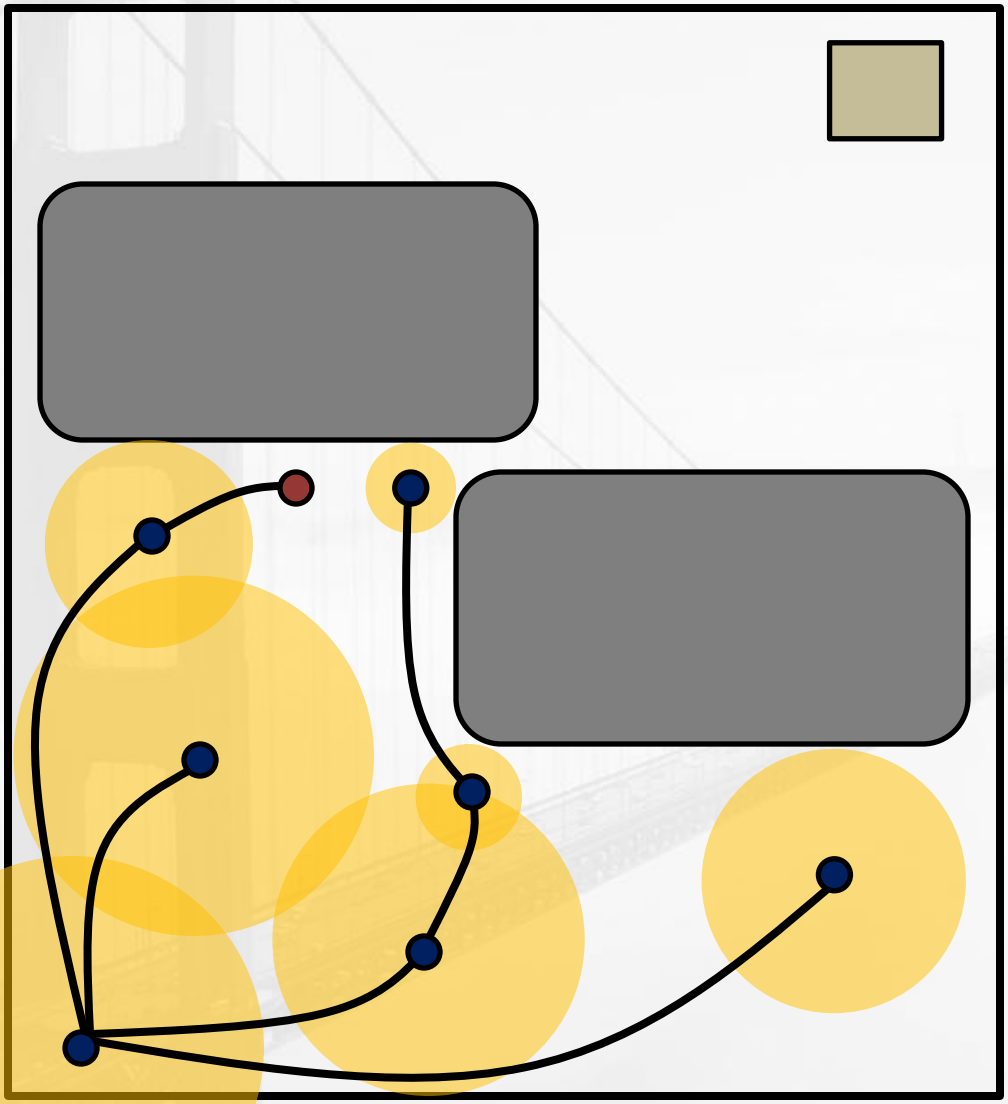
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

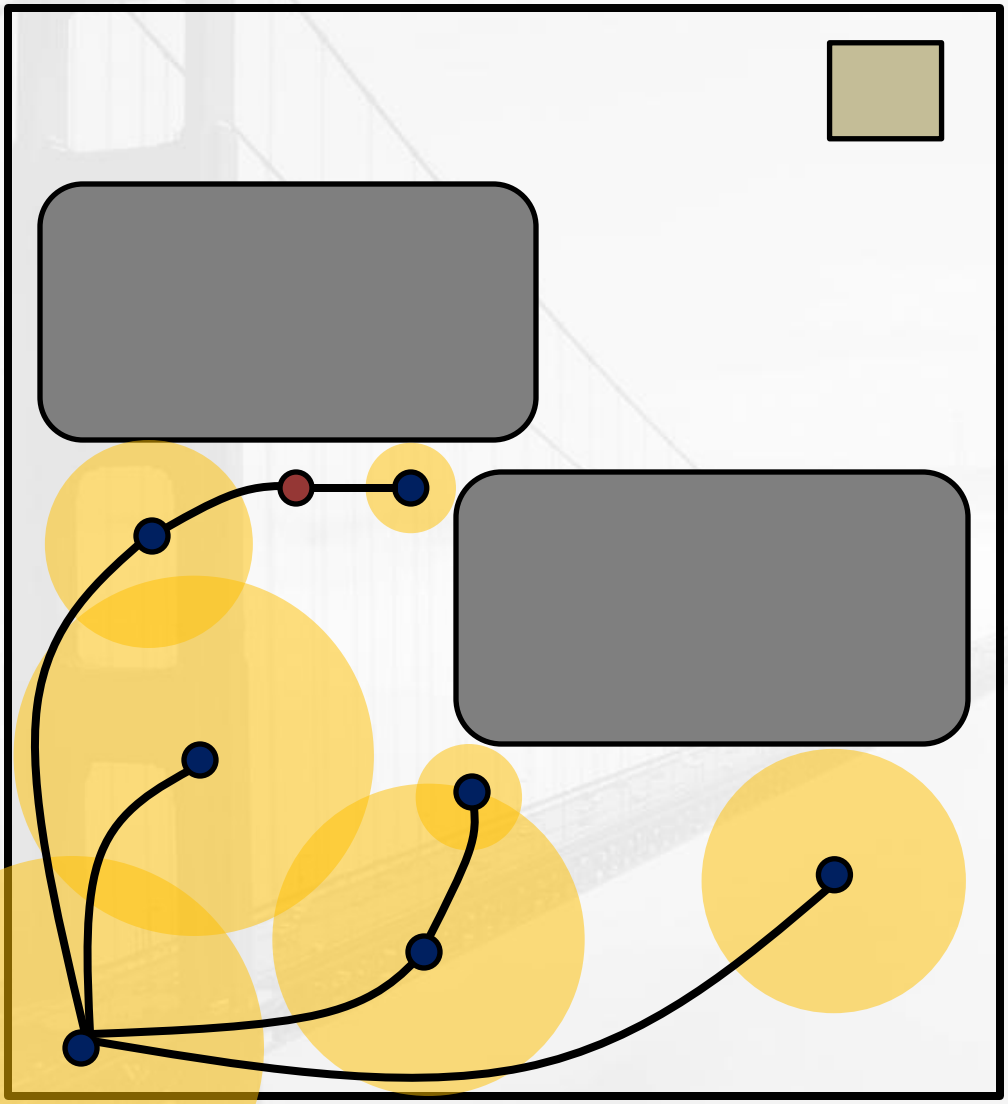
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

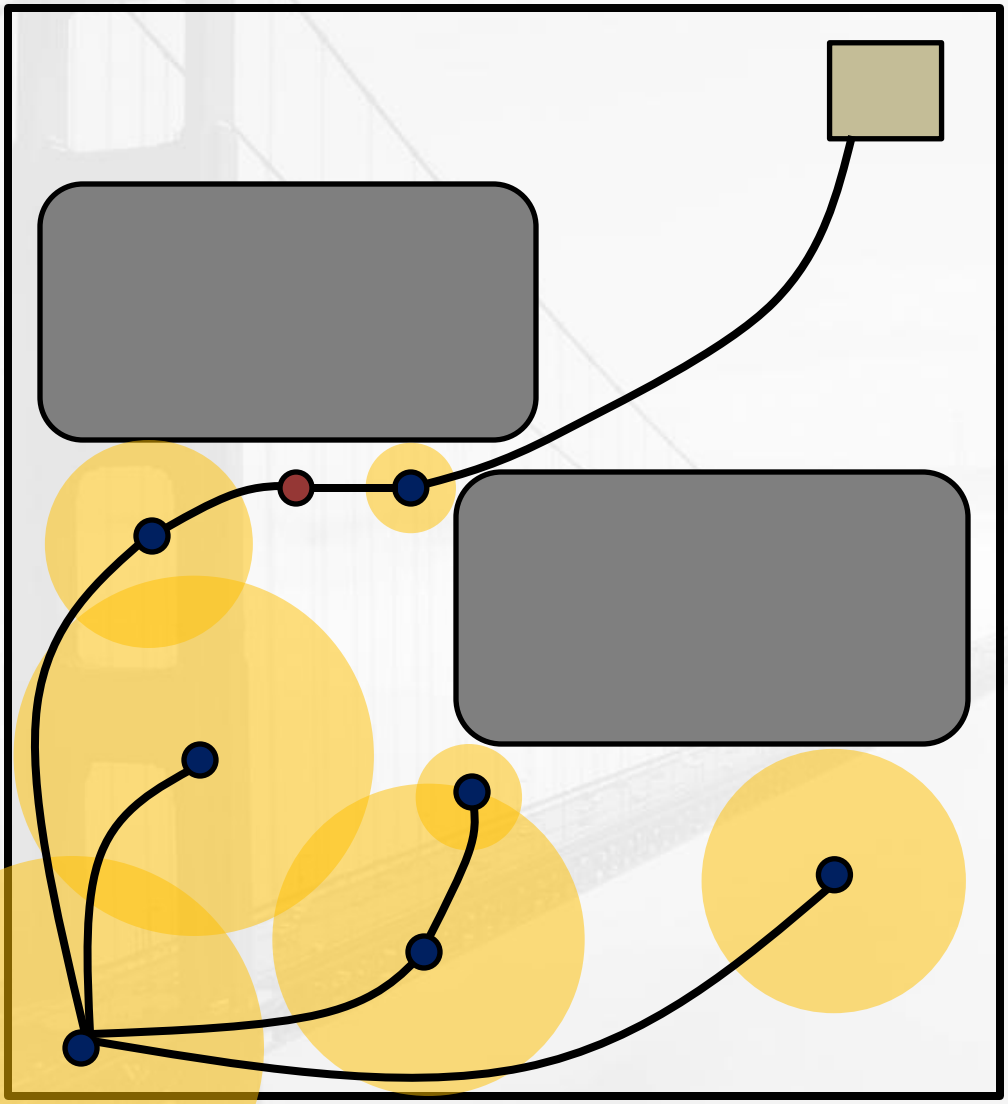
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

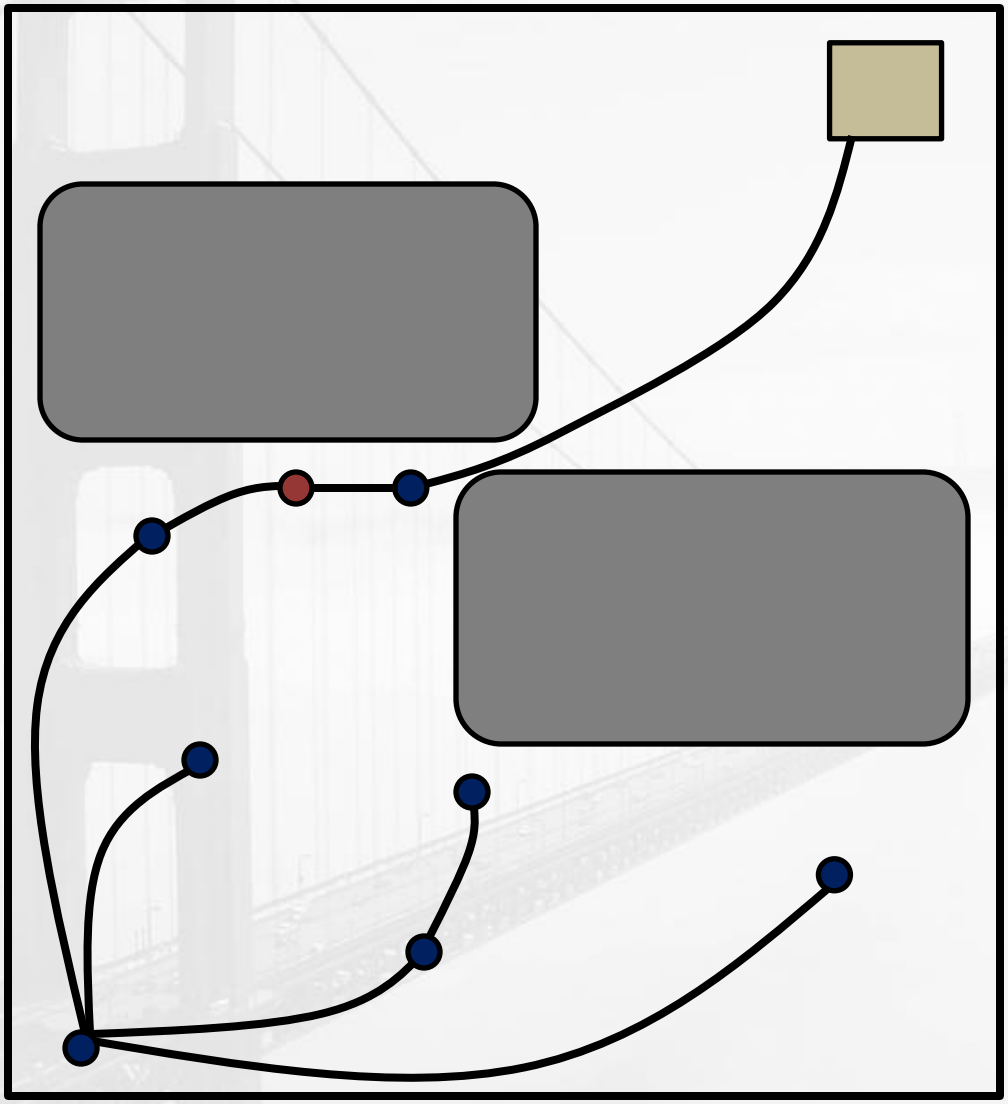
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

RRT* with Ball Trees



The RRT* Algorithm

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
  $X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{new}})$;
  if $X_{\text{near}} = \emptyset$ **then**
    $X_{\text{near}} \leftarrow \texttt{Nearest}(V, x_{\text{new}})$;
  $L_{\text{near}} \leftarrow \texttt{PopulateSortedList}(X_{\text{near}}, x_{\text{near}})$;
  $x_{\text{parent}} \leftarrow \texttt{FindBestParent}(L_{\text{near}}, x_{\text{new}})$;
  if $x_{\text{parent}} \neq$ NULL **then**
    V.add(x_{new});
    E.add$(\,(x_{\text{parent}}, x_{\text{new}})\,)$;
    $E \leftarrow \texttt{RewireVertices}(E, X_{\text{near}}, x_{\text{new}})$;
return $T = (V, E)$.

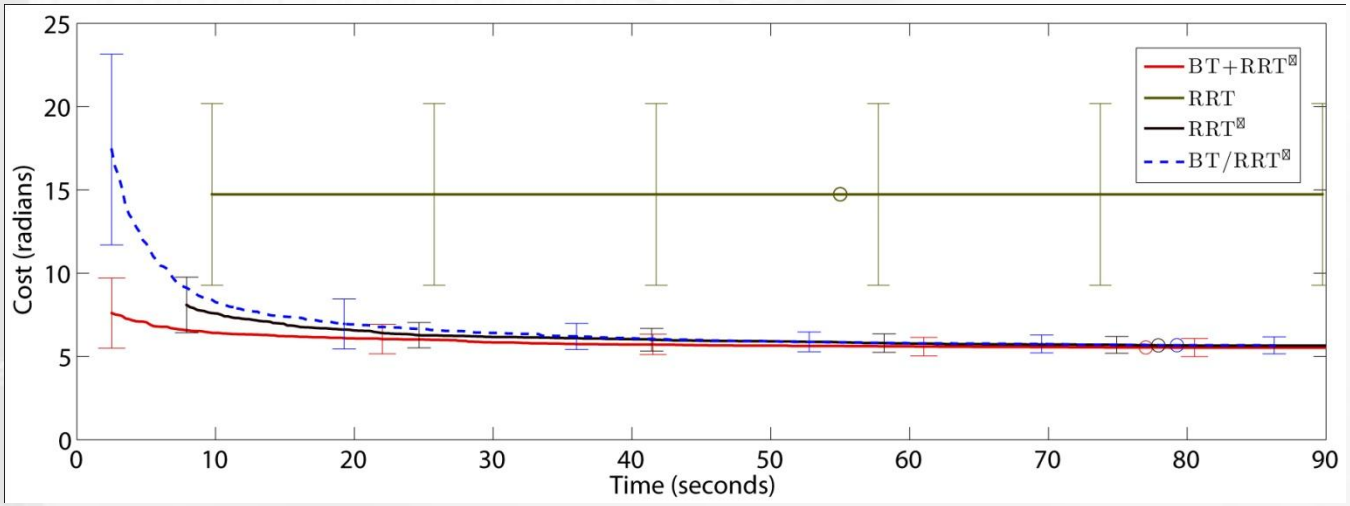
The Ball Tree Algorithm

$V \leftarrow \{x_{\text{init}}, r = 0\}$; $E \leftarrow \emptyset$; $T \leftarrow (V, E)$;
for $i = 1$ *to* N **do**
  while *true* **do**
    $x_{\text{new}} \leftarrow \texttt{Sample}(i)$;
    if InsideBall(x_{new}, T) **then**
      if !CollisionFree(x_{new}) **then**
        $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
        TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
    else
      break;
  $x_{\text{nearest}} \leftarrow \texttt{NearestBall}(V, x_{\text{new}})$;
  $\sigma \leftarrow \texttt{Steer}(x_{\text{new}}, x_{\text{nearest}})$;
  if CollisionFree(σ) **then**
    V.add(x_{new}, r_0);
    E.add$(\,(x_{\text{nearest}}, x_{\text{new}})\,)$;
  else
    TrimRadius$\,(x_{\text{nearest}}, \|x_{\text{new}} - x_{\text{nearest}}\|)$;
return $T = (V, E)$.

Results (7DOF)

RRT
First solution time: 9.75 s
First solution cost: 14.73
Final solution time: 54.96 s
Final solution cost: 14.73

BT+RRT*
First solution time: 2.52 s
First solution cost: 7.61
Final solution time: 77.14 s
Final solution cost: 5.52



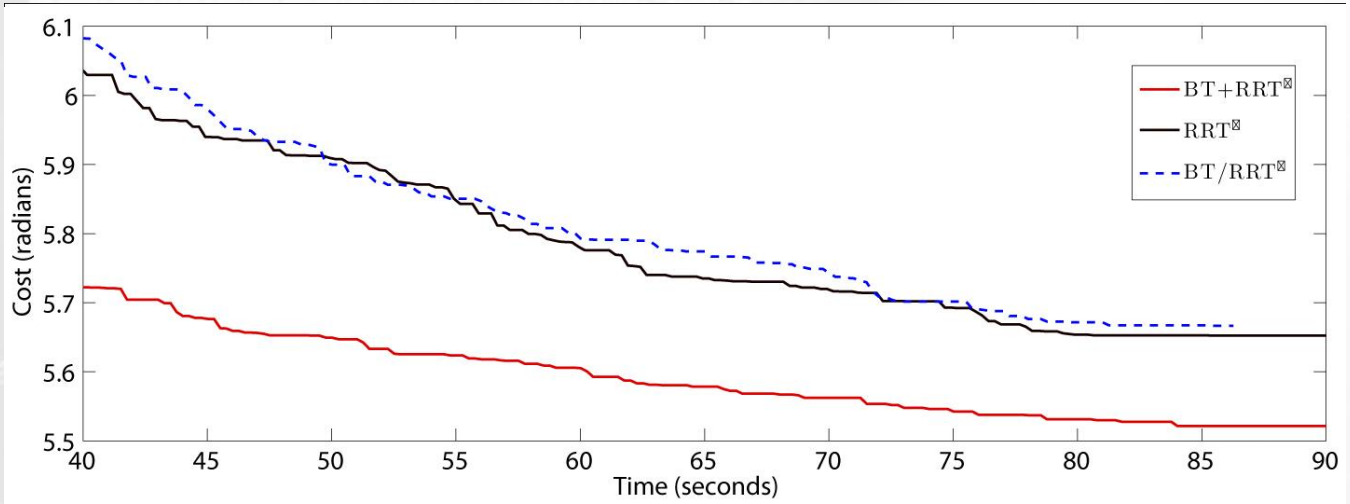
IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

Results (7DOF)

RRT
First solution time: 9.75 s
First solution cost: 14.73
Final solution time: 54.96 s
Final solution cost: 14.73

BT+RRT*
First solution time: 2.52 s
First solution cost: 7.61
Final solution time: 77.14 s
Final solution cost: 5.52



IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

CSAIL

Results (7DOF)

RRT
First solution time: 9.75 s
First solution cost: 14.73
Final solution time: 54.96 s
Final solution cost: 14.73

BT+RRT*
First solution time: 2.52 s
First solution cost: 7.61
Final solution time: 77.14 s
Final solution cost: 5.52

TABLE I

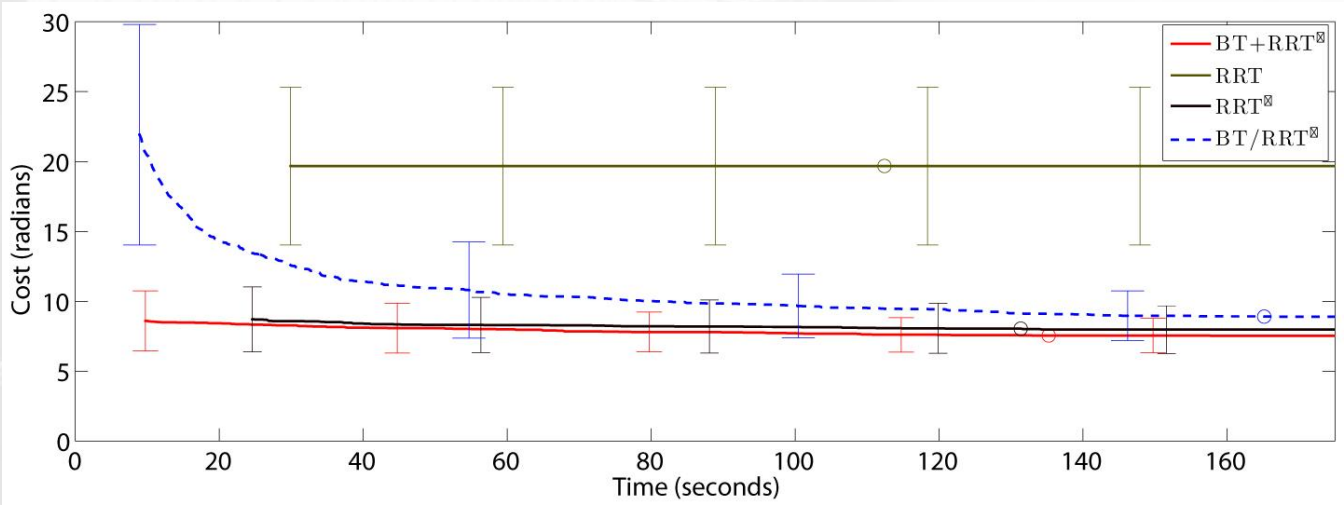
SEVEN DEGREE OF FREEDOM MONTE CARLO RESULTS

		BT+RRT*	RRT	RRT*	BT/RRT*
Success Rate (100 runs)		100.00%	87.00%	99.00%	100.00%
First Solution	Time (s)	2.52 (3.07)	9.75 (12.52)	7.92 (10.97)	2.51 (2.48)
	Cost	7.61 (2.11)	14.73 (5.49)	8.11 (1.67)	17.99 (5.63)
Final Solution	Time (s)	77.14 (4.49)	54.96 (4.75)	77.85 (3.95)	79.21 (4.47)
	Cost	5.52 (0.53)	14.73 (5.49)	5.65 (0.50)	5.67 (0.51)
Time per Iteration (ms)		19.33 (1.13)	13.78 (1.19)	19.51 (0.99)	19.85 (1.12)

Results (12DOF)

RRT
First solution time: 29.92 s
First solution cost: 19.76
Final solution time: 112.41 s
Final solution cost: 19.76

BT+RRT*
First solution time: 9.74 s
First solution cost: 8.59
Final solution time: 135.28 s
Final solution cost: 7.53



IEEE/RSJ International Conference on Intelligent Robots and Systems 2011

Results (12DOF)

RRT
First solution time: 29.92 s
First solution cost: 19.76
Final solution time: 112.41 s
Final solution cost: 19.76

BT+RRT*
First solution time: 9.74 s
First solution cost: 8.59
Final solution time: 135.28 s
Final solution cost: 7.53

TABLE II

TWELVE DEGREE OF FREEDOM MONTE CARLO RESULTS

		BT+RRT*	RRT	RRT*	BT/RRT*
Success Rate (100 runs)		100.00%	58.00%	85.00%	100.00%
First Solution	Time (s)	9.74 (12.84)	29.92 (34.05)	24.61 (32.09)	8.94 (11.06)
	Cost (rad)	8.59 (2.16)	19.76 (5.69)	8.71 (2.34)	22.13 (7.72)
Final Solution	Time (s)	135.28 (15.08)	112.41 (19.46)	131.38 (14.49)	165.28 (28.16)
	Cost (rad)	7.53 (1.21)	19.76 (5.69)	7.97 (1.71)	8.83 (1.73)
Time per Iteration (ms)		22.58 (2.52)	18.77 (3.25)	21.93 (2.42)	27.59 (4.70)

IROS 2011 Video

For more information:

Perez, A., Karaman, S., Shkolnik, A., Frazzoli E., Teller, S. and Walter, M., "Asymptotically-optimal Manipulation Planning using Incremental Sampling-based Algorithms," in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011.

http://ares.lids.mit.edu/manipulation_planning

Interactive Session (Golden Gate Room)

The PR2 Workshop

Also featured in:
MIT News, Slashdot, New Scientist, New England Post, Gizmag, io9, AzoRobotics and Indian Express