Robot Path Planning using Rapidly-exploring Random Trees: A Membrane Computing Approach

Ignacio Pérez-Hurtado[1], Mario J. Pérez-Jiménez[1], Gexiang Zhang[2], **David Orellana-Martín**[1]

[1]Research Group on Natural Computing
Dept. of Computer Science and Artificial Intelligence
Universidad de Sevilla, Seville, Spain
[2]School of Electrical Engineering
Southwest Jiaotong University
Chengdu, Sichuan, China

Seville, Spain, Jan 7, 2019

Introduction

- Where are we?

Introduction

- Where are we?
- Global planning
- Local planning
- Control

Introduction

- Where are we?
- Global planning
- Local planning
- Control

Path planning

- Motion planning problem

Path planning

- Motion planning problem
- Holonomic vs Nonholonomic

Path planning

- Motion planning problem
- Holonomic vs Nonholonomic (x, y, θ)

Path planning

- Motion planning problem
- Holonomic vs Nonholonomic (x, y, θ)
- Dimensions are important here: **degrees of freedom**

Motion planning problem

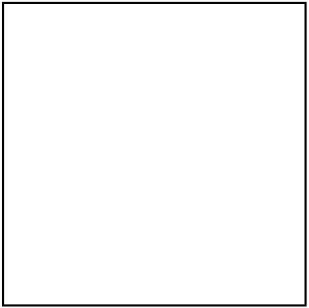
Typical scenario in a motion planning problem:

[1] Or its complementary, a *free region* X_{free}

Motion planning problem

Typical scenario in a motion planning problem:

- A *n*-dimensional space X

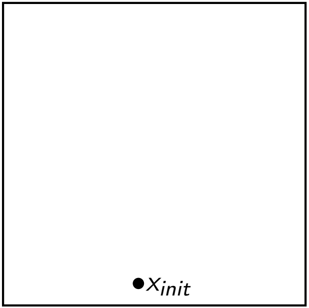


[1]Or its complementary, a *free region* X_{free}

Motion planning problem

Typical scenario in a motion planning problem:

- A *n*-dimensional space X
- An initial state x_{init}

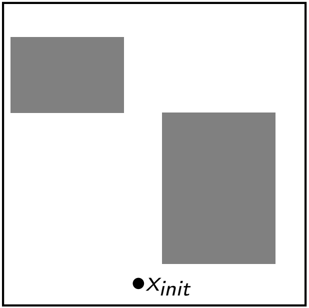


[1] Or its complementary, a *free region* X_{free}

Motion planning problem

Typical scenario in a motion planning problem:

- A n-dimensional space X
- An initial state x_{init}
- An *obstacle region* X_{obs} [1]

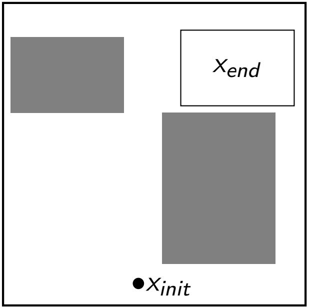


[1] Or its complementary, a *free region* X_{free}

Motion planning problem

Typical scenario in a motion planning problem:

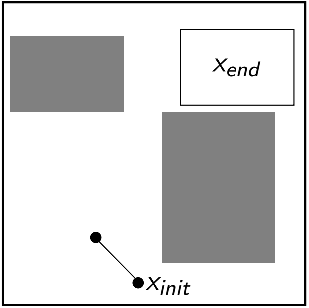
- A n-dimensional space X
- An initial state x_{init}
- An *obstacle region* X_{obs} [1]
- A *target region* X_{end}



[1] Or its complementary, a *free region* X_{free}

RRT algorithm: extension

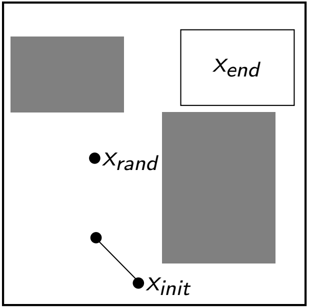
An extension of the generated tree step:



RRT algorithm: extension

An extension of the generated tree step:

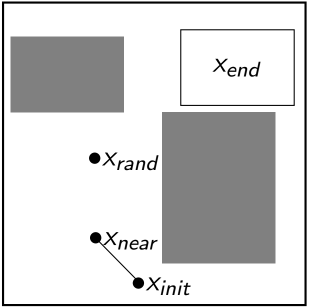
- We generate a random point x_{rand}



RRT algorithm: extension

An extension of the generated tree step:

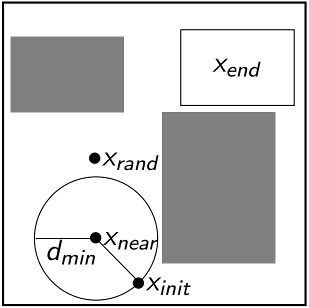
- We generate a random point x_{rand}
- x_{near} = nearest neighbor of the tree of x_{rand}



RRT algorithm: extension

An extension of the generated tree step:

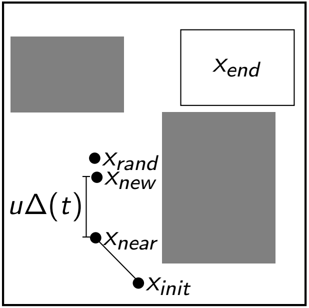
- We generate a random point x_{rand}
- x_{near} = nearest neighbor of the tree of x_{rand}
- If $distance(x_{rand}, x_{near}) \geq d_{min}$



RRT algorithm: extension

An extension of the generated tree step:

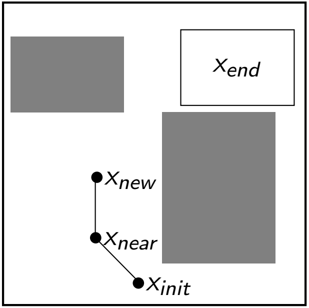
- We generate a random point x_{rand}
- x_{near} = nearest neighbor of the tree of x_{rand}
- If $distance(x_{rand}, x_{near}) \geq d_{min}$
- Create x_{new} depending on x_{near}



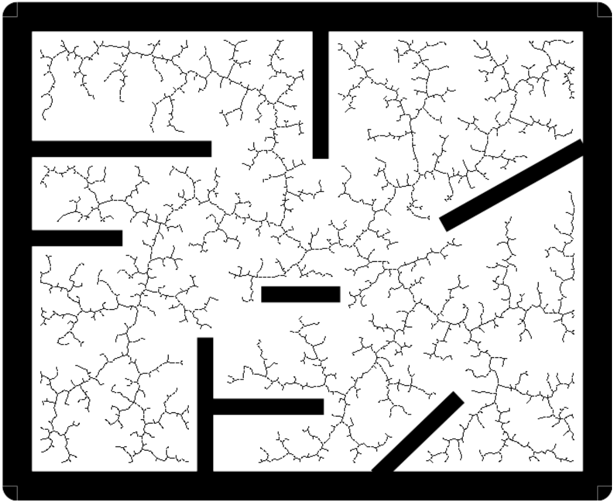
RRT algorithm: extension

An extension of the generated tree step:

- We generate a random point x_{rand}
- x_{near} = nearest neighbor of the tree of x_{rand}
- If $distance(x_{rand}, x_{near}) \geq d_{min}$
- Create x_{new} depending on x_{near}
- Add x_{new} to the current tree



RRT algorithm



RRT algorithm

- It manages nonholonomic, kinodynamic and environment restrictions

RRT algorithm

- It manages nonholonomic, kinodynamic and environment restrictions
- The *free* space is explored in a uniform way

RRT algorithm

- It manages nonholonomic, kinodynamic and environment restrictions
- The *free* space is explored in a uniform way
- It is computationally tractable

RRT algorithm

- It manages nonholonomic, kinodynamic and environment restrictions
- The *free* space is explored in a uniform way
- It is computationally tractable
- A path can be generated by connecting two RRT: one from the starting state, another from the goal state: **Bidirectional** RRT algorithm

Bidirectional RRT algorithm: a hint

- It is easy by its name:

Bidirectional RRT algorithm: a hint

- It is easy by its name:
- An initial node at x_{init}
- An initial node at x_{end}

Bidirectional RRT algorithm: a hint

- It is easy by its name:
- An initial node at x_{init}
- An initial node at x_{end}
- When they "find" each other, we return the path

Bidirectional RRT algorithm: a hint

- It is easy by its name:
- An initial node at x_{init}
- An initial node at x_{end}
- When they "find" each other, we return the path [2]

[2] Remember d_{min}

Random Enzymatic Numerical P systems with Proteins and Shared Memory

- **Variables** with **numerical** values

Random Enzymatic Numerical P systems with Proteins and Shared Memory

- **Variables** with **numerical** values
- Specific kind of variables, called **enzymes**

Random Enzymatic Numerical P systems with Proteins and Shared Memory

- **Variables** with **numerical** values
- Specific kind of variables, called **enzymes**
- Special alphabet of **proteins**

Random Enzymatic Numerical P systems with Proteins and Shared Memory

- **Variables** with **numerical** values
- Specific kind of variables, called **enzymes**
- Special alphabet of **proteins**
- Finite set of programs $F(x_{1,h}, \ldots, x_{k_F,h}) \xrightarrow{e(F);\alpha(F)} c_1 \mid v_1, \ldots c_{n_F} \mid v_{n_F}$

Random Enzymatic Numerical P systems with Proteins and Shared Memory

- **Variables** with **numerical** values
- Specific kind of variables, called **enzymes**
- Special alphabet of **proteins**
- Finite set of programs $F(x_{1,h}, \ldots, x_{k_F,h}) \xrightarrow{e(F);\alpha(F)} c_1 \mid v_1, \ldots c_{n_F} \mid v_{n_F}$
- A membrane representing a shared memory

A few slides ago...

- Where are we?
- Global planning
- Local planning
- Control

A few slides ago...

- Where are we?
- Global planning
- Local planning
- Control

A few slides ago...

- Where are we?
- **Global planning**
- Local planning
- Control

Simulation of the Bidirectional RRT algorithm by RENPSM

Algorithm 3 GENERATE_PATH

Require: $x_{init}, x_{end}, K, \rho, \Delta t, X, X_{obs}, d_{min}$

```
V_{τa} ← {x_init}; E_{τa} ← ∅;
V_{τb} ← {x_end}; E_{τb} ← ∅;
for k = 1 to K do
    x_rand ← RANDOM_STATE(X);
    if EXTEND(τa, x_rand, ρ, Δt, X_obs, d_min) ≠ Trapped then
        if EXTEND(τb, x_new, ρ, Δt, X_obs, d_min) = Reached then
            return PATH(τa, τb);
        end if
    end if
    SWAP(τa, τb);
end for
return Failure
```

Simulation of the Bidirectional RRT algorithm by RENPSM

Algorithm 4 GENERATE_PATH_PARALLEL

Require: $(x_{init}, x_{end}, K, \rho, \Delta t, X, X_{obs}, d_{min}$

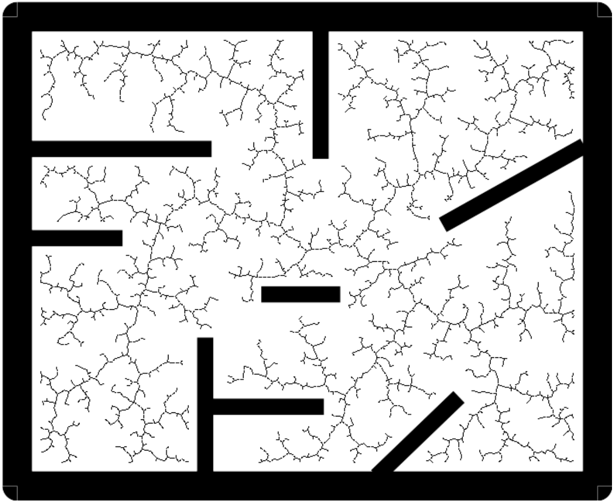
$V_{\tau_a} \leftarrow \{x_{init}\}$; $E_{\tau_a} \leftarrow \emptyset$;
$V_{\tau_b} \leftarrow \{x_{end}\}$; $E_{\tau_b} \leftarrow \emptyset$;
for $k = 1$ **to** K **do**
  $x_{rand,a} \leftarrow$ RANDOM_STATE(X);
  $x_{rand,b} \leftarrow$ RANDOM_STATE(X);
  begin parallel block
    $result_a$ = EXTEND($\tau_a, x_{rand,a}, \rho, \Delta t, X_{obs}, d_{min}$);
    $result_b$ = EXTEND($\tau_b, x_{rand,b}, \rho, \Delta t, X_{obs}, d_{min}$);
  end parallel block
  if $result_a \neq$ *Trapped* **then**
    if EXTEND($\tau_b, x_{new}, \rho, \Delta t, X_{obs}, d_{min}$) = *Reached* **then**
      return PATH(τ_a, τ_b);
    end if
  end if
  if $result_b \neq$ *Trapped* **then**
    if EXTEND($\tau_a, x_{new}, \rho, \Delta t, X_{obs}, d_{min}$) = *Reached* **then**
      return PATH(τ_a, τ_b);
    end if
  end if
end for
return *Failure*

Keys in the simulation

- Proteins as commanders of the computation
- Each 18 steps, a new iteration is started
- $Halt_{mem}$ is the halting variable

Min. cost	11.77 m
Max. cost	17.96 m
Avg. cost	13.42 m
Std. dev.	0.795 m
N. experiments	1435

Simulation of the Bidirectional RRT algorithm by RENPSM



Future work

- Formal verification of RENPSM
- Change to **non**honolomic robots
- Use differents kinds of P systems
- ...

DANKE

Gràcies Merci

MULȚUMESC

谢谢 THANKYOU

Obrigado Gracias Dankon