Robot motion planning using P systems: A roadmap on the current work

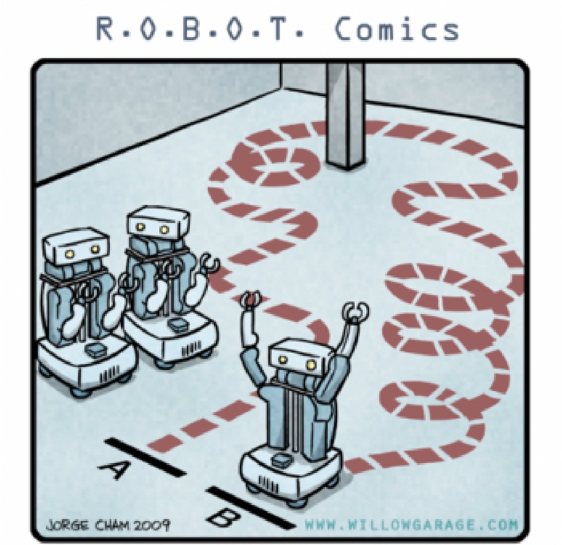
I. Pérez-Hurtado, M.A. Martínez-del-Amor,
D. Orellana-Martín and M.J. Pérez-Jiménez

Research Group on Natural Computing
Dpt. Computer Science and Artificial Intelligence
University of Seville

18th Brainstorming Week on Membrane Computing
BWMC 2020

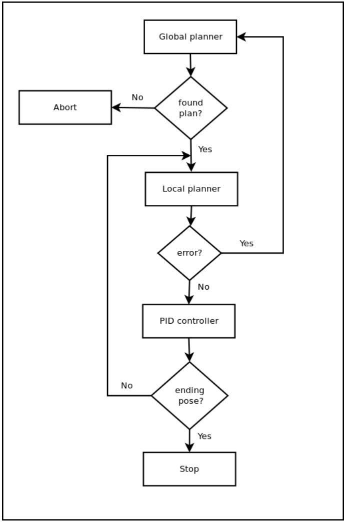
Summary

Robot motion planning



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Global and local planning



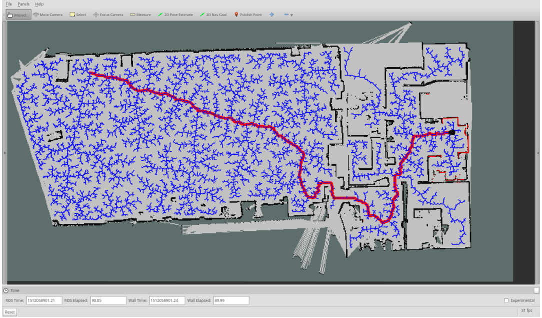
Global planning algorithms

The RRT algorithm

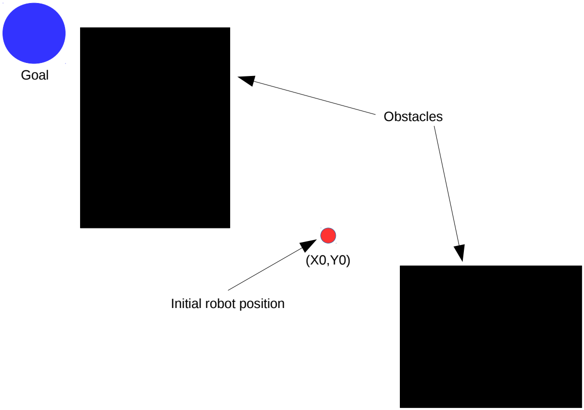
Definition

An RRT [a] is a randomized tree structure for rapidly exploring the obstacle-free configuration space.

[a] LaValle, S.M, *Rapidly-exploring random trees: A new tool for path planning*, technical report, 1998

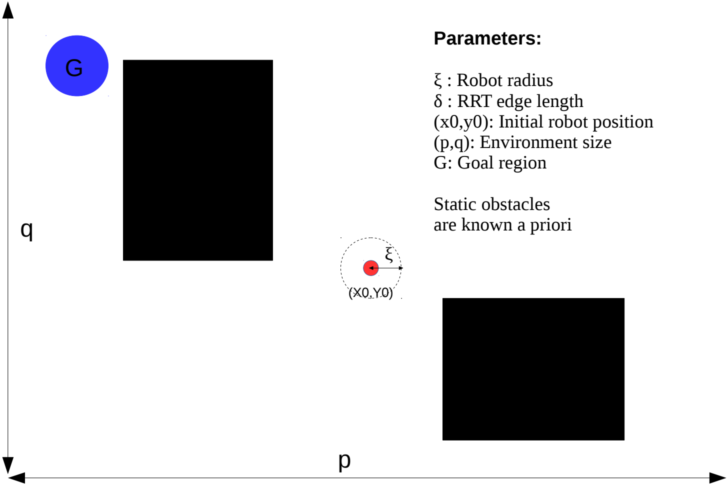


The RRT algorithm
Initial configuration



The RRT algorithm

Input parameters

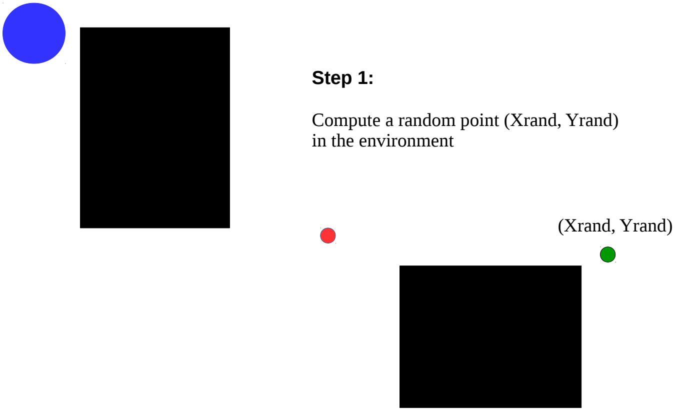


Parameters:

ξ : Robot radius
δ : RRT edge length
(x0,y0): Initial robot position
(p,q): Environment size
G: Goal region

Static obstacles
are known a priori

The RRT algorithm
Step 1



Step 1:

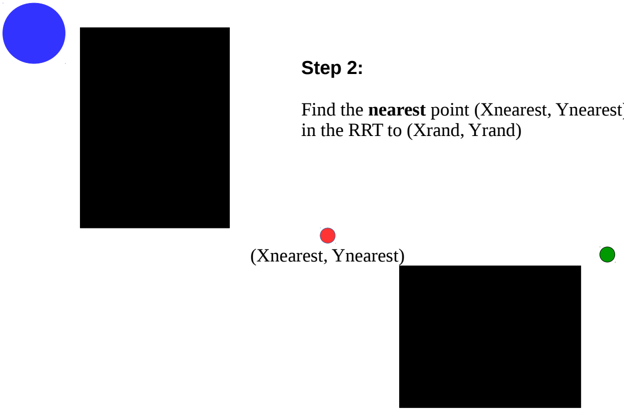
Compute a random point (Xrand, Yrand) in the environment

The RRT algorithm

Step 2

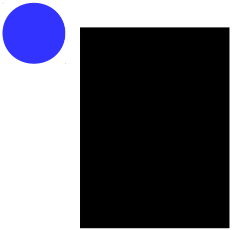
Step 2:

Find the **nearest** point (Xnearest, Ynearest) in the RRT to (Xrand, Yrand)



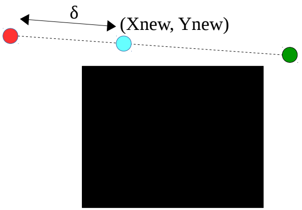
The RRT algorithm

Step 3

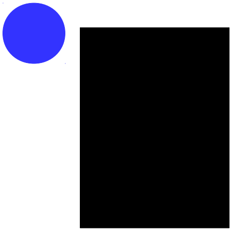


Step 3:

Compute the point (Xnew, Ynew) as the point at δ distance to (Xnearest, Ynearest) in the segment [(Xnearest,Ynearest),(Xrand,Yrand)]

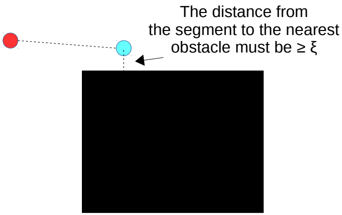


The RRT algorithm
Step 4



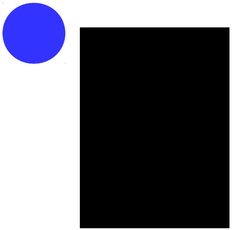
Step 4:

Check if the segment [(Xnearest,Ynearest),(Xnew,Ynew)] is **obstacle free**



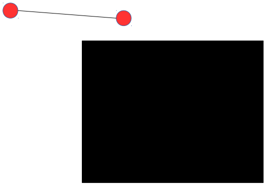
The RRT algorithm

Step 5



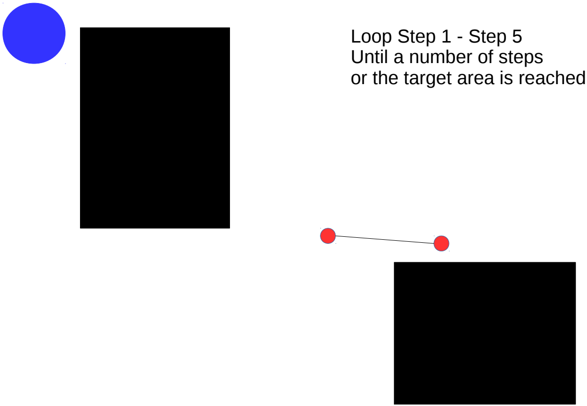
Step 5:

If the segment is obstacle free,
then include a node (Xnew,Ynew) to the RRT and an edge [(Xnearest,Ynearest),(Xnew,Ynew)]



The RRT algorithm

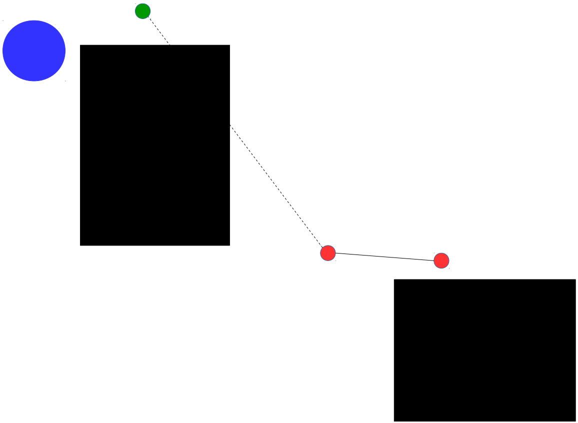
Loop Step 1 - Step 5



Loop Step 1 - Step 5
Until a number of steps
or the target area is reached

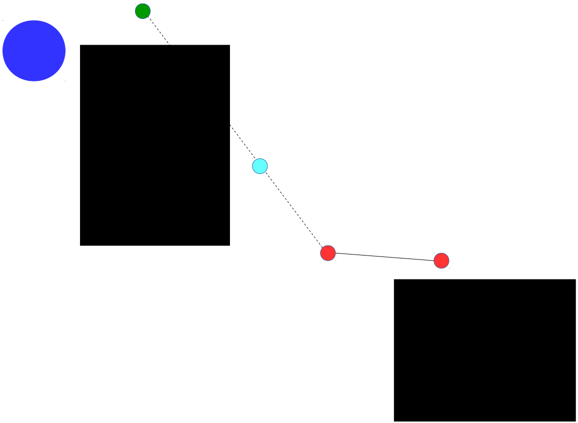
The RRT algorithm

Loop Step 1 - Step 5



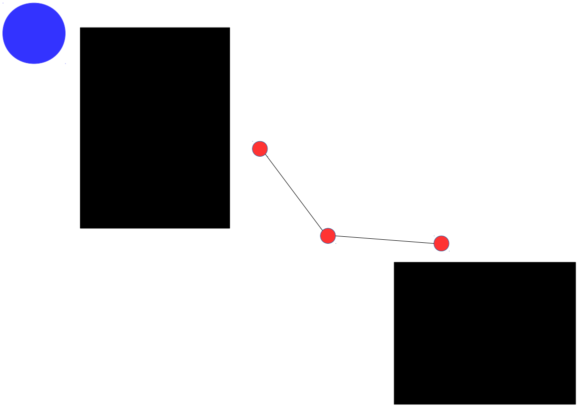
The RRT algorithm

Loop Step 1 - Step 5



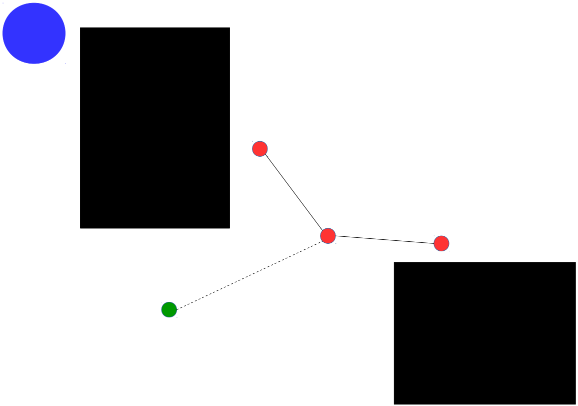
The RRT algorithm

Loop Step 1 - Step 5



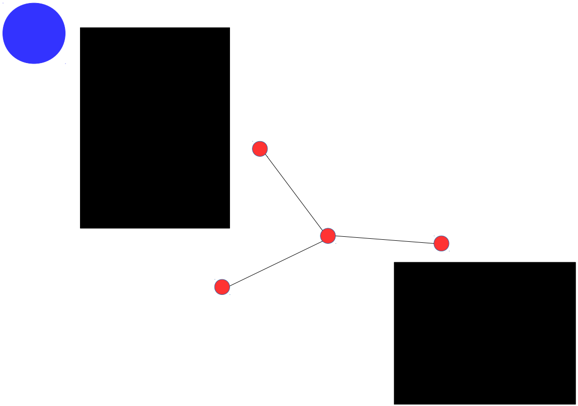
The RRT algorithm

Loop Step 1 - Step 5



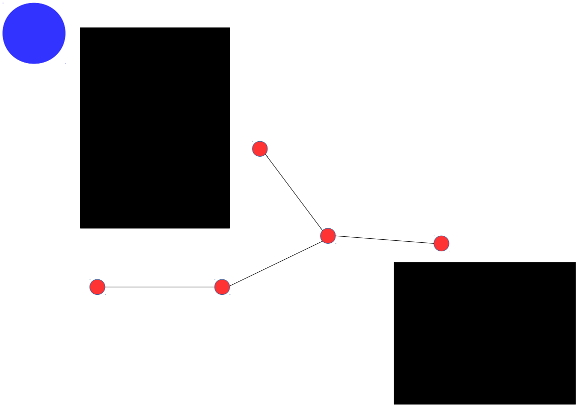
The RRT algorithm

Loop Step 1 - Step 5



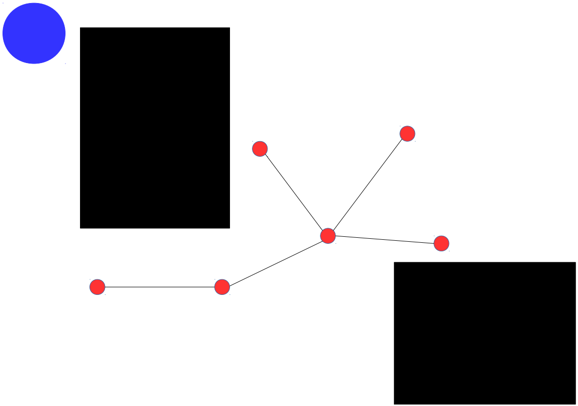
The RRT algorithm

Loop Step 1 - Step 5



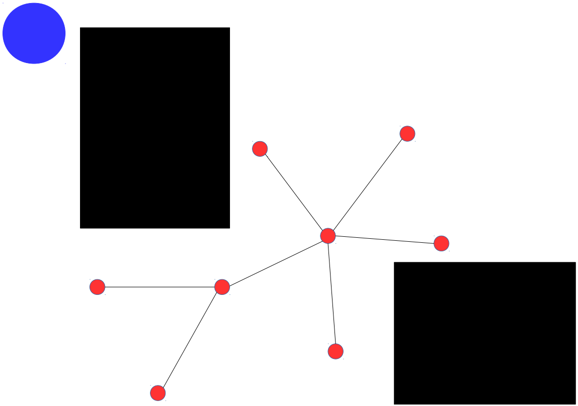
The RRT algorithm

Loop Step 1 - Step 5



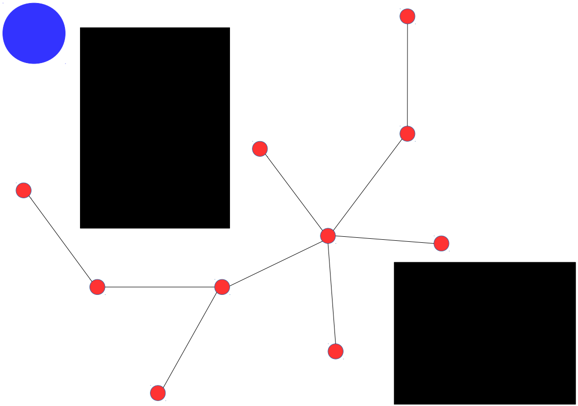
The RRT algorithm

Loop Step 1 - Step 5



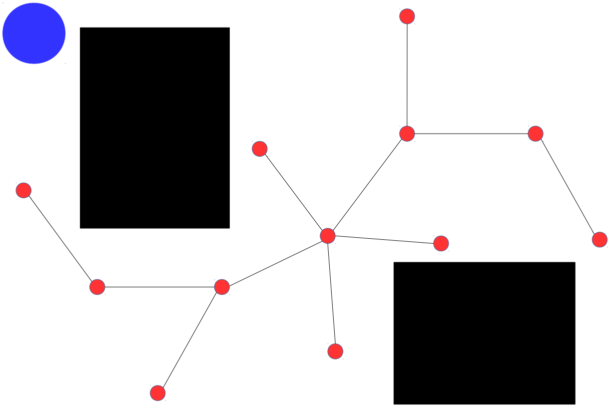
The RRT algorithm

Loop Step 1 - Step 5



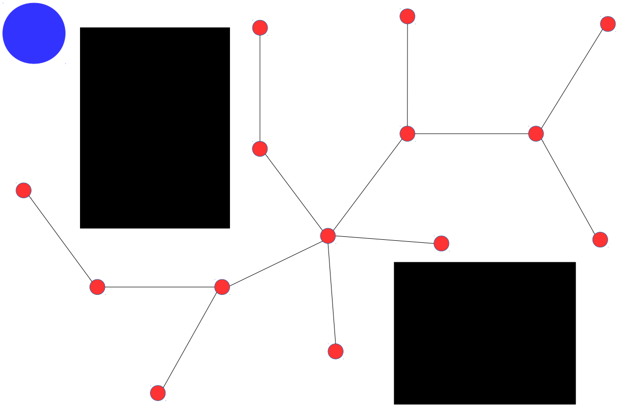
The RRT algorithm

Loop Step 1 - Step 5



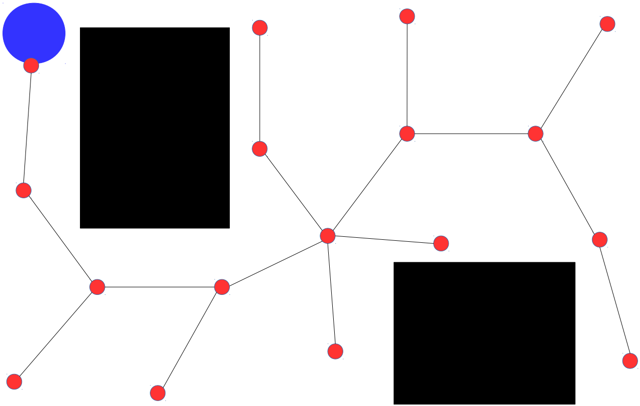
The RRT algorithm

Loop Step 1 - Step 5



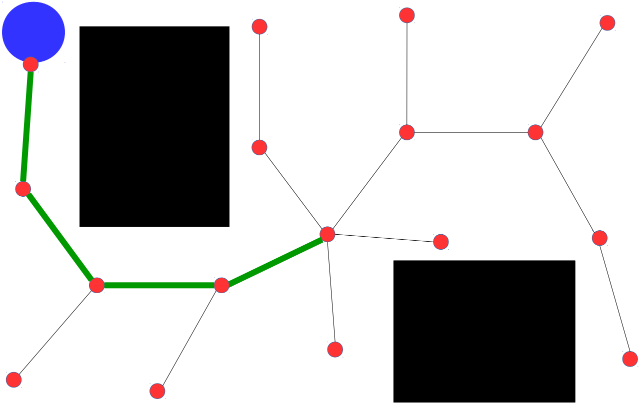
The RRT algorithm

Loop Step 1 - Step 5



The RRT algorithm

Loop Step 1 - Step 5



The RRT* algorithm

- A variant of the RRT algorithm
- It produces *good* solutions considering a cost function
- It refines its solution while adding nodes to the tree
- When the number of nodes is infinity, the solution is optimal

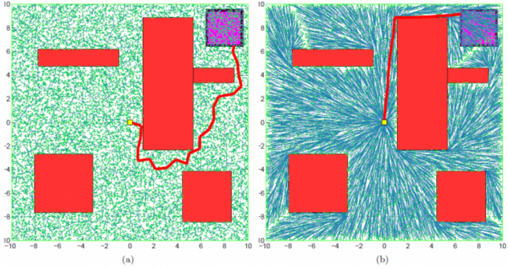


Figure 14: A Comparison of the RRT (shown in (a)) and RRT* (shown in (b)) algorithms on a simulation example with obstacles. Both algorithms were run with the same sample sequence for 20,000 samples. The cost of best path in the RRT and the RRG were 21.02 and 14.51, respectively.

Global planning algorithms

Membrane computing models

- I. Pérez-Hurtado, M.A. Martínez-Del-Amor, G. Zhang, F. Neri, M.J. Pérez-Jiménez A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning. *Integrated Computer-Aided Engineering*, in press, accepted manuscript, 2020
- I. Pérez-Hurtado, M.J. Pérez-Jiménez, G. Zhang, D. Orellana-Martín. *Simulation of Rapidly-Exploring Random Trees in Membrane Computing with P-Lingua and Automatic Programming*. International Journal of Computers, Communications and Control, Volume 13, No 6, December 2018, pages 1007-1031
- I. Pérez-Hurtado, M.J. Pérez-Jiménez Generation of rapidly-exploring random tree by using a new class of membrane system. *Pre-proceedings of Asian Conference on Membrane Computing (ACMC2017)*, Xihua University, Chengdu, China, September 21-25, 2017, Pages 534-546

Enzymatic Numerical P systems

ENPS[a] are an extension of Numerical P Systems (NPS) used for modeling and simulating membrane controllers for autonomous mobile robots.

[a] Pavel et al., *A. et al. Enzymatic Numerical P systems - A new class of membrane computing systems*, BIC-TA 2010

Enzimatic Numerical P systems

$$\Pi = (m, H, \mu, (Var_1, E_1, Pr_1, Var_1(0)), \ldots, (Var_m, E_m, Pr_m, Varalgorithms_m(0)))$$

where

- m is the number of membranes; $m \geq 1$;
- H is an alphabet of labels, containing m symbols;
- μ is the membrane structure;
- Var_i is a set of variables for compartment i, being $Var_i(0)$ their initial values;
- E_i is a set of variables $E_i \subseteq Var_i$ called enzymes.
- Pr_i is a set of programs for compartment i. The syntax of a program is the following:

$$F(x_1, \ldots, x_k)|_{Cond(e_1,\ldots,e_r)} \rightarrow c_1|v_1 + \ldots + c_n|v_n$$

An ENPS model for the RRT algorithm

RRT(n,m,p,q,δ,ξ)

Loop_{RRT}(n,m,p,q,δ)

Nearest_{RRT}(n)

$\text{ObstacleFree}_{RRT}$(m,$\xi$)

Extend_{RRT}(n,m)

Where 2^n is the number of points to add to the RRT and 2^m is the number of obstacles in the environment. algorithms

An ENPS model for the RRT algorithm

The $Loop_{RRT}$ module

Loop$_{RRT}$**(n,m,p,q,**δ**)**

$x_1\,[input], y_1\,[input]$

$a_i[input], b_i[input] : 1 \leq i \leq 2^m$

$x_i[3\cdot p], y_i[3\cdot q] : 2 \leq i \leq 2^n$

$px_i[0], py_i[0] : 2 \leq i \leq 2^n$

$x_{rand}[0], y_{rand}[0], x_{new}[0], y_{new}[0],$

$x_{nearest}[0], y_{nearest}[0], collision[0], \alpha[1], index[2], halt[0]$

$0|_{\alpha=1} \rightarrow collision$

$p\cdot random()|_{\alpha=1} \rightarrow x_{rand}$

$q\cdot random()|_{\alpha=1} \rightarrow y_{rand}$

$x_{nearest} + \delta \cdot \frac{(x_{rand} - x_{nearest})}{\sqrt{d_1}}|_{\alpha=n+3} \rightarrow x_{new}$

$y_{nearest} + \delta \cdot \frac{(y_{rand} - y_{nearest})}{\sqrt{d_1}}|_{\alpha=n+3} \rightarrow y_{new}$

$rm(\alpha, m+n+6)+1|_{collision\leq 0} \rightarrow \alpha$

$1|_{collision>0} \rightarrow \alpha$

$index+1|_{\alpha=m+n+6} \rightarrow index$

$1|_{index=2^n+1} \rightarrow halt$

An ENPS model for the RRT algorithm

The $Nearest_{RRT}$ module

Nearest$_{RRT}$(n)

$d_i[0], x'_i[0], y'_i[0] : 1 \leq i \leq 2^n$

$(x_i - x_{rand})^2 + (y_i - y_{rand})^2|_{\alpha=2} \to d_i : 1 \leq i \leq 2^n$

$x_i|_{\alpha=2} \to x'_i : 1 \leq i \leq 2^n$

$y_i|_{\alpha=2} \to y'_i : 1 \leq i \leq 2^n$

$min(d_i, d_{i+2^{n-j}})|_{\alpha=j+2} \to d_i : 1 \leq i \leq 2^{n-j}, 1 \leq j \leq n$

$min^*(x'_i, x'_{2^{n-j}}, d_i, d_{2^{n-j}})|_{\alpha=j+2} \to x'_i : 1 \leq i \leq 2^{n-j}, 1 \leq j < n$

$min^*(y'_i, y'_{2^{n-j}}, d_i, d_{2^{n-j}})|_{\alpha=j+2} \to y'_i : 1 \leq i \leq 2^{n-j}, 1 \leq j < n$

$min^*(x'_1, x'_2, d_1, d_2)|_{\alpha=n+2} \to x_{nearest}$

$min^*(y'_1, y'_2, d_1, d_2)|_{\alpha=n+2} \to y_{nearest}$

An ENPS model for the RRT algorithm

The $ObstacleFree_{RRT}$ module

ObstacleFree$_{RRT}$**(m,**ξ**)**

$d'_i[0] : 1 \leq i \leq 2^m$

$pDist(a_i, b_i, x_{nearest}, y_{nearest}, x_{new}, y_{new})|_{\alpha=n+4} \rightarrow d'_i : 1 \leq i \leq 2^m$

$min(d'_i, d'_{i+2^{m-j}})|_{\alpha=j+n+4} \rightarrow d'_i : 1 \leq i \leq 2^{m-j}, 1 \leq j < m$

$min(\xi - d'_1, \xi - d'_2)|_{\alpha=m+n+4} \rightarrow collision$

An ENPS model for the RRT algorithm

The $Extend_{RRT}$ module

Extend$_{RRT}$(n,m)

$$x_{new}|_{\alpha=m+n+6} \rightarrow x_{[index]}$$

$$y_{new}|_{\alpha=m+n+6} \rightarrow y_{[index]}$$

$$x_{nearest}|_{\alpha=m+n+6} \rightarrow px_{[index]}$$

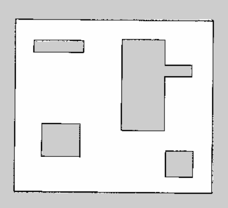
$$y_{nearest}|_{\alpha=m+n+6} \rightarrow py_{[index]}$$

Developed software

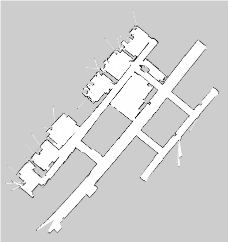
- An ad-hoc simulator in C++
- It manages image files in PGM defining the obstacle maps.
- The resolution is $5cm^2$/pixel (Hokuyo LIDAR resolution)
- δ is 15 cm
- ξ is 20 cm

Developed software

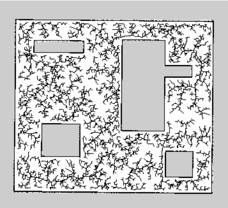
Input and output maps



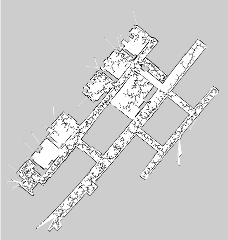
(a) map1



(b) ccia_h



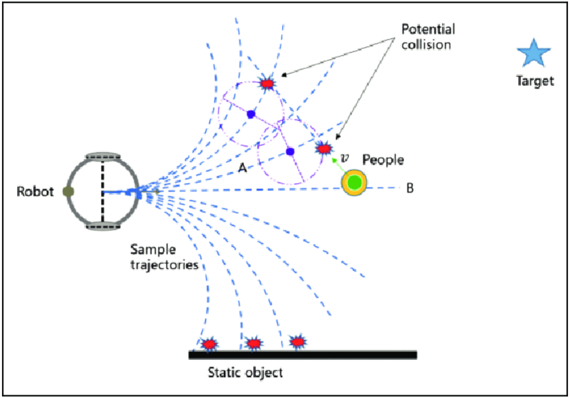
(c) Output map1



(d) Output ccia_h

Local planning algorithms

The dynamic window approach



Conclusions and future work

- ENPS has been used for robot controllers
- We have designed ENPS models for global planning: ENPS-RRT and ENPS-RRT*
- The models are compatible with current robot controllers.
- Simulators have been implemented for validation and testing using CUDA and OpenMP
- As present work, we are designing ENPS models for local planning
- By using embedded CUDA platforms (Jetson TX2), a *membrane computing inside* robot for navigation can be implemented.
- As alternative to CUDA and OpenMP, we are interested in FPGA implementations.

Thanks!