Social robotics and membrane computing

Ignacio Pérez-Hurtado

Research Group on Natural Computing
Dpt. Computer Science and Artificial Intelligence
University of Seville, Spain
perezh@us.es

January 30, 2018, 16th BWMC, Seville

Outline

- Social robotics
  - FROG project
  - TERESA project
- Robot navigation
  - The RRT algorithm
- The RENPSM framework
- Conclusions and future work

Social robotics

- A social robot is an autonomous robot that interacts with humans by following social behaviors.
- From 2014 to 2017 I was working at the Service Robotics Lab at UPO
- I was involved in two FP7 European Projects:
  - FROG: Fun Robotic Outdoor Guide
  - TERESA: Telepresence Reinforcement-learning Social Agent

FROG

Fun Robotic Outdoor Guide (https://www.frogrobot.eu)

FROG | Fun Robotic Outdoor Guide

Summary: FROG proposes to develop a guide robot with a winning personality and behaviours that will engage tourists in a fun exploration of outdoor attractions.

Collaborative project under the FP7-ICT-2011.2.1 Cognitive Systems and Robotics (a), (d) area of activity

Consortium:

- University of Amsterdam (UVA)
- YDreams - Informatica S.A. (YD)
- IDMind - Engenharia de Sistemas lda (IDM)
- Universidad Pablo de Olavide (UPO)
- Imperial College of Science, Technology and Medicine (ICL)
- University of Twente (UT)

FROG
FUN ROBOTIC OUTDOOR GUIDE

www.frogrobot.eu

FROG FUN ROBOTIC OUTDOOR GUIDE
SEVENTH FRAMEWORK PROGRAMME

University of Amsterdam — *People detection, tracking and body pose recognition*

Universidad Pablo de Olavide Sevilla — *Outdoor navigation in crowded scenarios*

YDreams — *Multimedia contents, augmented reality and integration*

IDmind — *Robot platform development and system integration*

Imperial College London — *Vision-based human tracking and affective behaviour understanding*

University of Twente. — *Human robot interaction strategies and usability evaluation*

14174 M&C Hannover flyer A5.indd 1 01/04/14 11:03

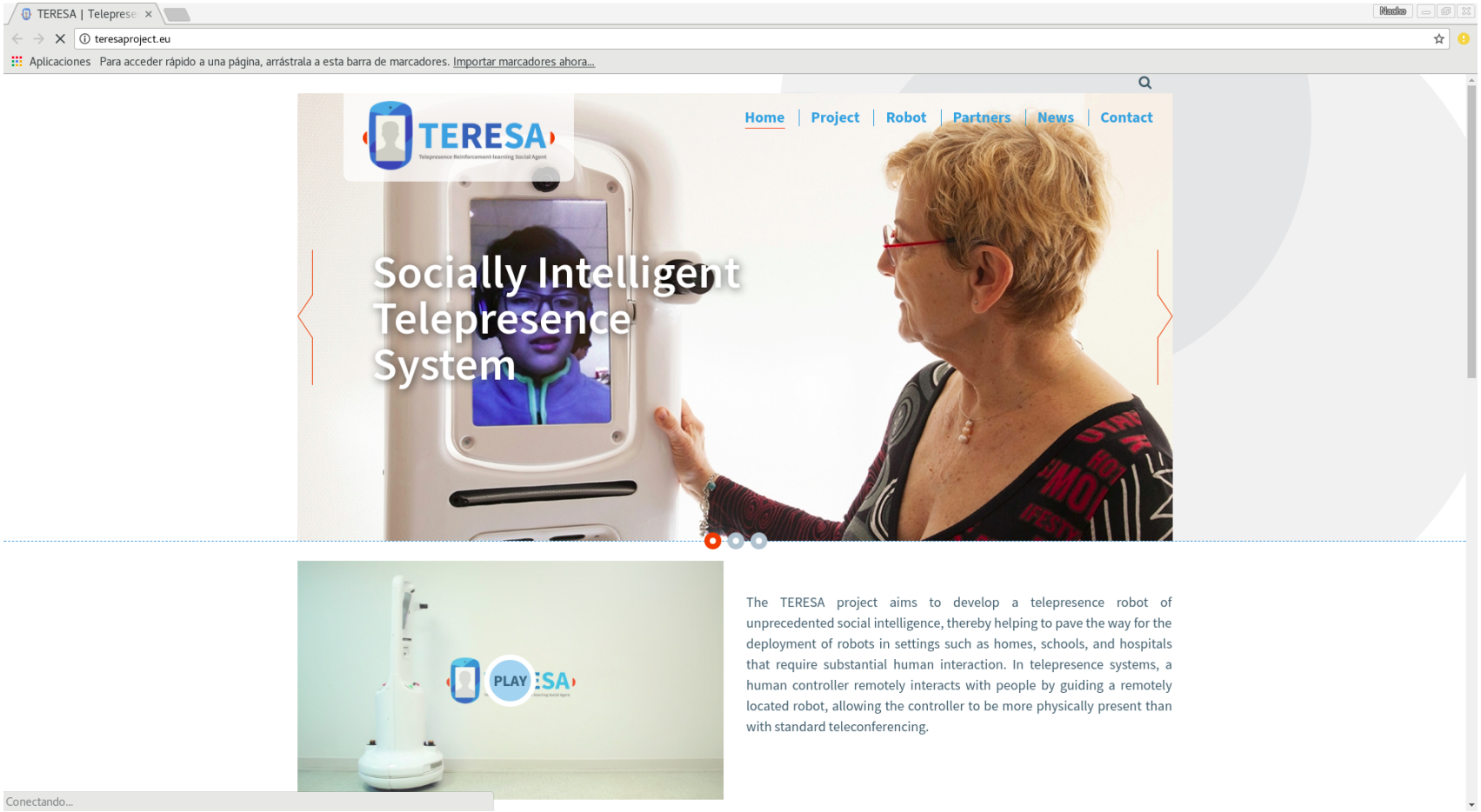
FROG

Fun Robotic Outdoor Guide (https://www.frogrobot.eu)

In the *Alcazar of Seville* (2014)

TERESA

TElepresence REinforcement-learning Social Agent (http://teresaproject.eu/)



TERESA

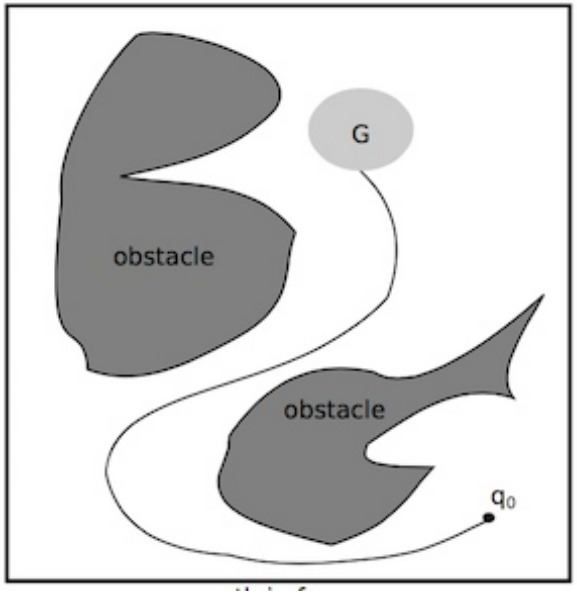
TElepresence REinforcement-learning Social Agent (http://teresaproject.eu/)

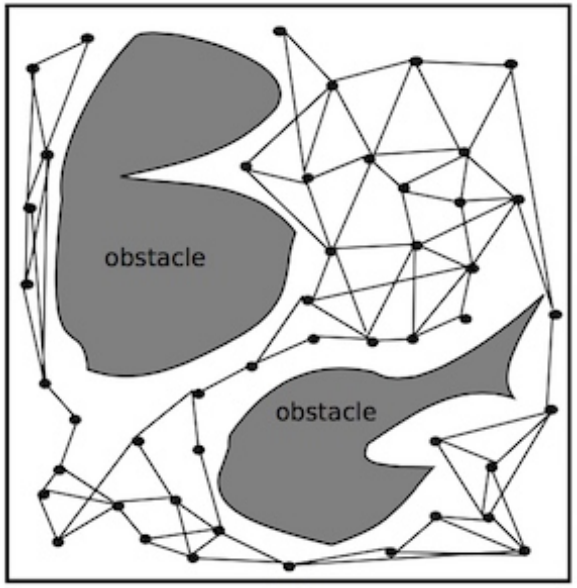
TERESA

TElepresence REinforcement-learning Social Agent (http://teresaproject.eu/)

Robot navigation

Particular case of the motion planning problem



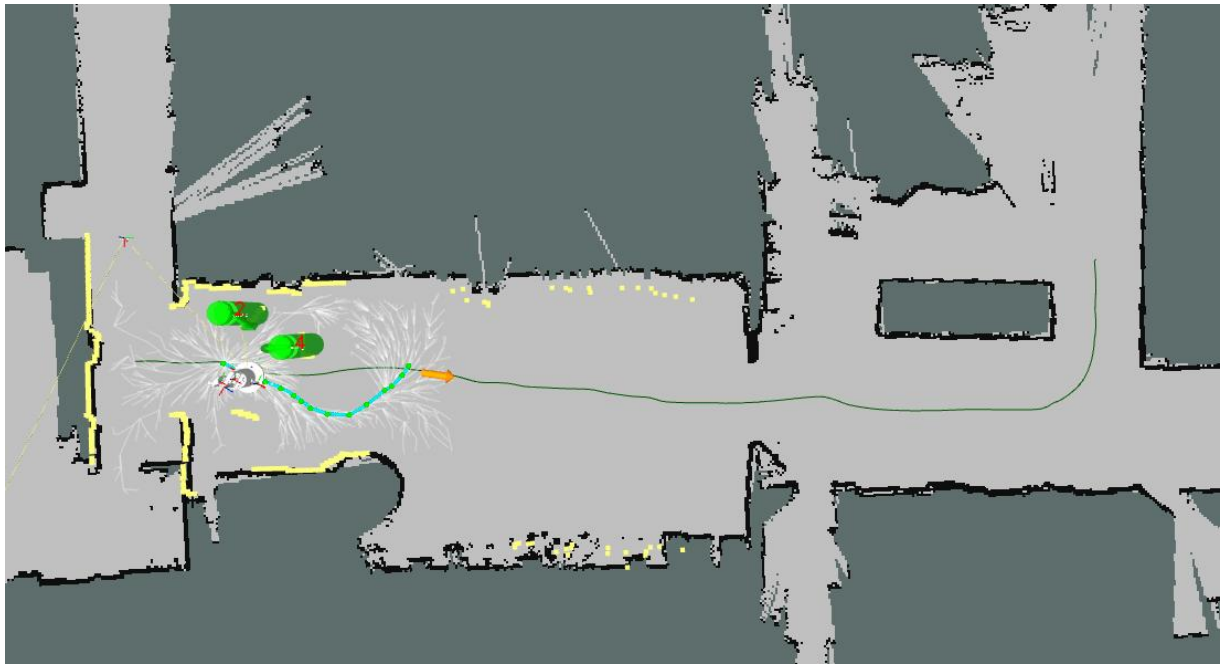


Given (1) a start configuration state, (2) a goal configuration state, (3) a geometric description of the robot, and (4) a geometric description of the environment: **find a path that moves the robot gradually from start to goal.**

TERESA

TElepresence Einforcement-learning Social Agent (http://teresaproject.eu/)

TERESA uses a variant of the RRT algorithm[1] for navigation

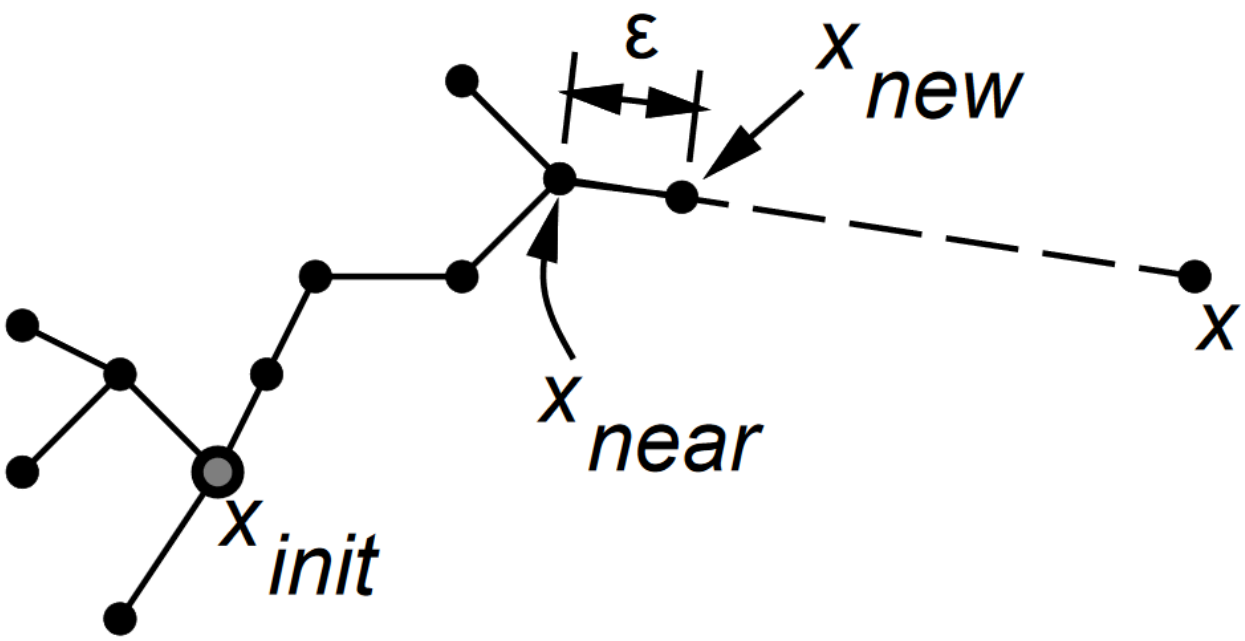


[1] S.M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning, *Computer Science Dept.*, Iowa State University, October 1998

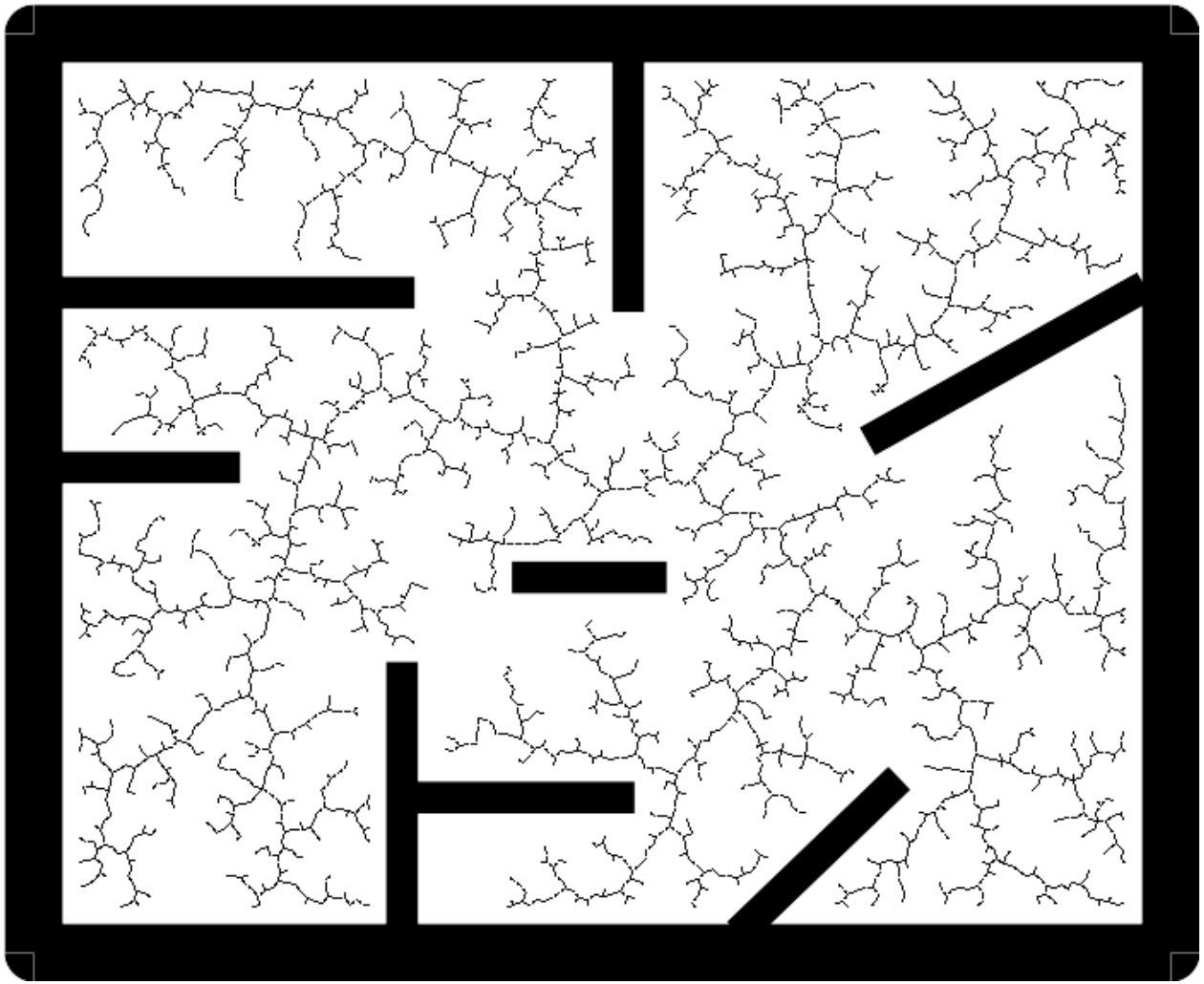
The RRT algorithm

$V_\tau \leftarrow \{x_{init}\}$
$E_\tau \leftarrow \emptyset$
for $k = 1$ **to** K **do**
  $x_{rand} \leftarrow$ RANDOM_STATE(X);
  $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, τ);
  if DISTANCE(x_{rand}, x_{near}) $\geq d_{min}$ **then**
    $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
    if $\neg$ COLLISION($x_{near}, u, \Delta t, X_{obs}$) **then**
      $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
      $V_\tau \leftarrow V_\tau \cup \{x_{new}\}$
      $E_\tau \leftarrow E_\tau \cup \{(x_{near}, x_{new})\}$
    end if
  end if
end for
return $\tau = (V_\tau, E_\tau)$

The RRT algorithm



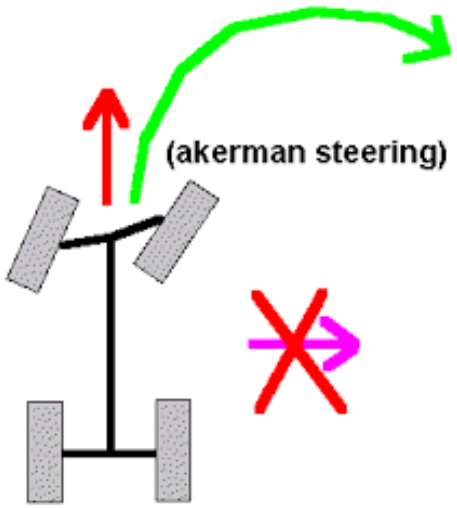
The RRT algorithm



The RRT algorithm

Properties

- It manages nonholonomic, kinodynamic and environment restrictions
- The valid space is explored in an 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*[2]



[2] S.M. LaValle, and J.J. Kuffner. Randomized kinodynamic planning. *Proceedings IEEE International Conference on Robotics and Automation*, 1999 pages 473–479

The RENPSM framework[3]

A preliminary first approach for the global planner

- We propose an extension of ENPS to solve the global planning by using the RRT algorithm
- New ingredients have been added to ENPS framework to fit the RRT algorithm requirements

[3] I.Pérez-Hurtado, M.J. Pérez-Jiménez. Generation of rapidly-exploring random trees by using a new class of membrane systems. *ACMC17*

The RENPSM framework

RENPSM: Random Enzymatic Numerical P systems with Proteins and Shared Memory

- Membranes in a cell-like structure represent RRT nodes
- A special membrane called *mem* is used as a shared memory
- Random numbers can be generated in the shared memory
- Proteins are used as regular objects for synchronization

The RENPSM framework

$$\Pi = (H, \mu, h_0, P, E_{mem}, E_{mem}(0), \{(P_h(0), Var_h, Var_h(0), Pr_h) \mid h \in H\}, \mathcal{R})$$

1. $H = \{1, \ldots, p \cdot q\} \cup \{mem\}$, $mem \notin \{1, \ldots, p \cdot q\}$, is the set of labels;
2. μ is a dynamical membrane structure;
3. h_0 is the label of an initial membrane in μ;
4. mem is the label of the shared memory;
5. P is a set proteins;
6. E_{mem} is a set of enzymes;
7. $Var_h, h \in H$, is a finite set of variables associated with region h;
8. $Pr_h, h \in H$, is a set of programs associated with region h:

$$F(x_{1,h}, \ldots, x_{k_F,h}) \xrightarrow{e(F);\alpha(F)} c_1 | v_1, \ldots, c_{n_F} | v_{n_F}$$

The RENPSM framework

9. $\mathcal{R}$ is a finite set of rules of the following form:

   - *Protein evolution rules*:
   $$[\,\alpha \rightarrow \alpha'\,]_h$$
   - *Writing-only communication rules*
   $$(h\,,\,X_h\,/\,Y_{h,mem}\,,\,mem)^W_\alpha$$
   - *Reading-only communication rules:*
   $$(h\,,\,X_h\,/\,Y_{mem}\,,\,mem)^R_\alpha$$
   - *Membrane creation rules*:
   $$[\;[\;X_{1,h}\,,\,X_{2,h},\,,\,\ldots\,,\,X_{n,h}\;]_h\;]_{h'}\;;\,\alpha$$

One iteration of the RRT by means of a RENPSM

For holonomic robots without obstacles

This algorithm will be simulated by a random enzymatic numerical P system with proteins and shared memory of degree (p, q)

$$\Pi = (H, \mu, P, E_{mem}, E_{mem}(0), \{(P_h(0), Var_h, Var_h(0), Pr_h) \mid h \in H\}, \mathcal{R}, h_0)$$

defined as follows:

- $H = \{1, \ldots, p \cdot q\} \cup \{mem\}$, $mem \notin \{1, \ldots, p \cdot q\}$.
- $\mu = \{h_0\}$ with $h_0 \in \{1, \ldots, p \cdot q\}$.
- $P = \{\alpha_i \mid 1 \leq i \leq 12\}$, and $P_h(0) = \{\alpha_1\}$, for each $h \in H$.
- $E_{mem} = \{Flag_{mem}, p \cdot q + 1\}$ and $E_{mem}(0) = \{p \cdot q + 1\}$.
- The set of variables is:
  - $Var_h = \{X_{1,h}, X_{2,h}, Y_{1,h}, Y_{2,h}, D_h\}$, for each $h, 1 \leq h \leq p \cdot q$.
  - $Var_{mem} = \{X_{1,mem}, X_{2,mem}, Y_{1,mem}, Y_{2,mem}, Z_{1,mem}, Z_{2,mem}\} \cup \{U_{1,mem}, U_{2,mem}\} \cup \{D_{h,mem}, Y_{h,mem} \mid 1 \leq h \leq p \cdot q\}$.

One iteration of the RRT by means of a RENPSM

Two random numbers i, j $(1 \leq i \leq p, 1 \leq j \leq q)$ are generated in the shared memory.

$$\begin{cases} \textit{Production function} : F(X_{1,mem}) = Random(i, 1 \leq i \leq p) \\ \textit{Repartition protocol} : 1|X_{1,mem} \\ \textit{Protein} : \alpha_1 \end{cases}$$

$$\begin{cases} \textit{Production function} : F(X_{2,mem}) = Random(i, 1 \leq i \leq q) \\ \textit{Repartition protocol} : 1|X_{2,mem} \\ \textit{Protein} : \alpha_1 \end{cases}$$

One iteration of the RRT by means of a RENPSM

Each membrane $h \in \{1, \ldots, p \cdot q\}$ will read the random numbers previously generated, sharing them with the variables $X_{1,h}, X_{2,h}$.

$$\begin{cases} (h\,,\, X_{1,h} \,/\, X_{1,mem}\,,\, mem)^R_{\alpha_2} \\ \\ (h\,,\, X_{2,h} \,/\, X_{2,mem}\,,\, mem)^R_{\alpha_2} \end{cases}$$

One iteration of the RRT by means of a RENPSM

For each membrane $h \in \mu$, the distance D_h between its position $(Y_{1,h}, Y_{2,h})$ and the position given by $(X_{1,mem}, X_{2,mem})$ is computed.

$$\begin{cases} \textit{Production function} : F(X_{1,h}, X_{2,h}, Y_{1,h}, Y_{2,h}) = \begin{cases} \sqrt{\sum_{j=1}^{2}(X_{j,h} - Y_{j,h})^2} & \text{if} \quad h \in \mu \\ p \cdot q + 1 & \text{if} \quad h \notin \mu \end{cases} \\ \textit{Repartition protocol} : 1|D_h \\ \textit{Protein} : \alpha_3 \end{cases}$$

One iteration of the RRT by means of a RENPSM

Each membrane h writes its value D_h to the shared memory.

$$(h\,,\ D_h\,/\,D_{h,mem}\,,\ mem)^{W}_{\alpha_4}$$

One iteration of the RRT by means of a RENPSM

The minimum of all distances D_h is computed in the shared memory.

- *Production function*: $F(D_{1,mem}, \ldots, D_{p \cdot q,mem}) = \min\{D_{1,mem}, \ldots, D_{p \cdot q,mem}\}$
- *Repartition protocol*: $1|D_{min,mem}$
- *Protein*: α_5

One iteration of the RRT by means of a RENPSM

Variable (enzyme) $Flag_{mem}$ is set to zero if $D_{min,mem} \leq Threshold$.

- *Production function*: $F(D_{1,mem}, \ldots, D_{p \cdot q,mem}) = \begin{cases} 0 & \text{if } D_{min,mem} \leq Threshold \\ p \cdot q + 1 & \text{otherwise} \end{cases}$
- *Repartition protocol*: $1|Flag_{mem}$
- *Protein*: α_6

One iteration of the RRT by means of a RENPSM

The label *near*, corresponding to the closer membrane to the randomly generated position, is obtained.

- *Production function*: $F(D_{1,mem}, \ldots, D_{p \cdot q,mem}) = \text{arg-min}\{D_{1,mem}, \ldots, D_{p \cdot q,mem}\}$
- *Repartition protocol*: $1|Y_{near,mem}$
- *Protein*: α_7
- *Enzyme*: $Flag_{mem}$

One iteration of the RRT by means of a RENPSM

The position of membrane *near* is computed.

$$\begin{cases} \textit{Production function} : F(Y_{near,mem}) = 1 + qt(Y_{near,mem}, q) \\ \textit{Repartition protocol} : 1|Y_{1,mem} \\ \textit{Protein} : \alpha_8 \\ \textit{Enzyme} : Flag_{mem} \end{cases}$$

$$\begin{cases} \textit{Production function} : F(Y_{near,mem}) = rm(Y_{near,mem}, q) \\ \textit{Repartition protocol} : 1|Y_{2,mem} \\ \textit{Protein} : \alpha_8 \\ \textit{Enzyme} : Flag_{mem} \end{cases}$$

One iteration of the RRT by means of a RENPSM

The unitary vector is created in the shared memory.

$$\begin{cases} \textit{Production function} : F(X_{1,mem}, X_{2,mem}, Y_{1,mem}, Y_{2,mem}) = \frac{X_{1,mem} - Y_{1,mem}}{\sqrt{\sum_{j=1}^{2}(X_{j,mem} - Y_{j,mem})^2}} \\ \textit{Repartition protocol} : 1|U_{1,mem} \\ \textit{Protein} : \alpha_9 \\ \textit{Enzyme} : Flag_{mem} \end{cases}$$

$$\begin{cases} \textit{Production function} : F(X_{1,mem}, X_{2,mem}, Y_{1,mem}, Y_{2,mem}) = \frac{X_{2,mem} - Y_{2,mem}}{\sqrt{\sum_{j=1}^{2}(X_{j,mem} - Y_{j,mem})^2}} \\ \textit{Repartition protocol} : 1|U_{2,mem} \\ \textit{Protein} : \alpha_9 \\ \textit{Enzyme} : Flag_{mem} \end{cases}$$

One iteration of the RRT by means of a RENPSM

The position of the new membrane is computed in the shared memory.

$$\begin{cases} \textit{Production function} : F(Y_{1,mem}, U_{1,mem}) = Y_{1,mem} + U_{1,mem} \cdot \Delta t \\ \textit{Repartition protocol} : 1|Z_{1,mem} \\ \textit{Protein} : \alpha_{10} \\ \textit{Enzyme} : Flag_{mem} \end{cases}$$

$$\begin{cases} \textit{Production function} : F(Y_{2,mem}, U_{2,mem}) = Y_{2,mem} + U_{2,mem} \cdot \Delta t \\ \textit{Repartition protocol} : 1|Z_{2,mem} \\ \textit{Protein} : \alpha_{10} \\ \textit{Enzyme} : Flag_{mem} \end{cases}$$

One iteration of the RRT by means of a RENPSM

The membrane labelled by $Y_{near,mem}$ will read the position (Z_1, Z_2) corresponding to the new membrane from the shared memory.

$$\begin{cases} (Y_{near,mem}\ ,\ Z_{1,Y_{near,mem}}\ /\ Z_{1,mem}\ ,\ mem)^R_{\alpha_{11}} \\ \\ (Y_{near,mem}\ ,\ Z_{2,Y_{near,mem}}\ /\ Z_{2,mem}\ ,\ mem)^R_{\alpha_{11}} \end{cases}$$

One iteration of the RRT by means of a RENPSM

A child membrane with position (Z_1, Z_2) is created in Y_{near}, that is, a new state is added to the tree.

$$\left[\left[\begin{array}{cc} X_{1,h} & X_{2,h} \\ Y_{1,h} & Y_{2,h} \\ Z_{1,h} & Z_{2,h} \\ D_h & \end{array} \right]_h \right]_{Y_{near,mem}}$$

Being $h = (Z_{1,Y_{near,mem}} - 1) \cdot q + Z_{2,Y_{near,mem}}$. This rule is mediated by protein α_{12}.

Conclusions and Future work

- There are robots controllers based on MC
- We could use fast hardware in order to obtain advantage of the current state of the art of simulators and implementations in MC
- We are working in a variant of ENPS in order to simulate RRT algorithms
- There is a preliminary first approach presented in the last ACMC.
- We are currently adapting the RRT* algorithm: optimal path planning.
- We would like to mix our solution with current robotic controllers based on MC
- We would like to improve our simulator

Thanks for your attention!