Robust-IAC Exploration Explorer for Matlab

Release 1.0

Adrien Baranes, Pierre-Yves Oudeyer
INRIA Bordeaux - Sud-Ouest
351 Cours de la Liberation, 33405 Talence, France
{Adrien.Baranes, Pierre-Yves.Oudeyer}@inria.fr

August 7, 2009

Abstract

The *R-IAC Exploration Explorer* allows experiments in different robotic setups of the *Robust-Intelligent Adaptive Curiosity* algorithm introduced in [Baranes and Oudeyer, 2009]. It consists of a tuning interface fixing exploration parameters and allowing to make automatic experiments for statistics studies.

Contents

1 What is R-IAC Exploration Explorer? 3

2 How to use it ? 3

3 Global Mechanism 3

4 Blocks 3
- 4.1 Interactions System-Environment : *Interaction.m* . . . 3
- 4.2 Initialization : *World_Initialize.m* . . . 4
- 4.3 Exploration Data : *ProcessData.m* . . . 4

5 Global Exploration System : Tuning Interface 5
- 5.1 Function Number . . . 6
- 5.2 Test Database . . . 6
- 5.3 Exploration Criterion : Random Exploitation . . . 6
- 5.4 Exploration Criterion : Max Exploitation . . . 6
- 5.5 Exploration Criterion : Max Exploration . . . 6
- 5.6 Exploration Criterion : Random Exploration . . . 6
- 5.7 Total Number of Iterations . . . 6
- 5.8 Criter Regions : Maximal Size of a Region . . . 6
- 5.9 Sample . . . 6
- 5.10 Number of Exploration . . . 6
- 5.11 Run . . . 6

6 Demonstrations 7
- 6.1 1-Dimension Approach : $NumFct = 1$ . . . 7
- 6.2 2-Dimension Arm-Camera Experiment : $NumFct = 2$ . . . 8
- 6.3 Hand-Eye Experiment : $NumFct = 3$ . . . 9

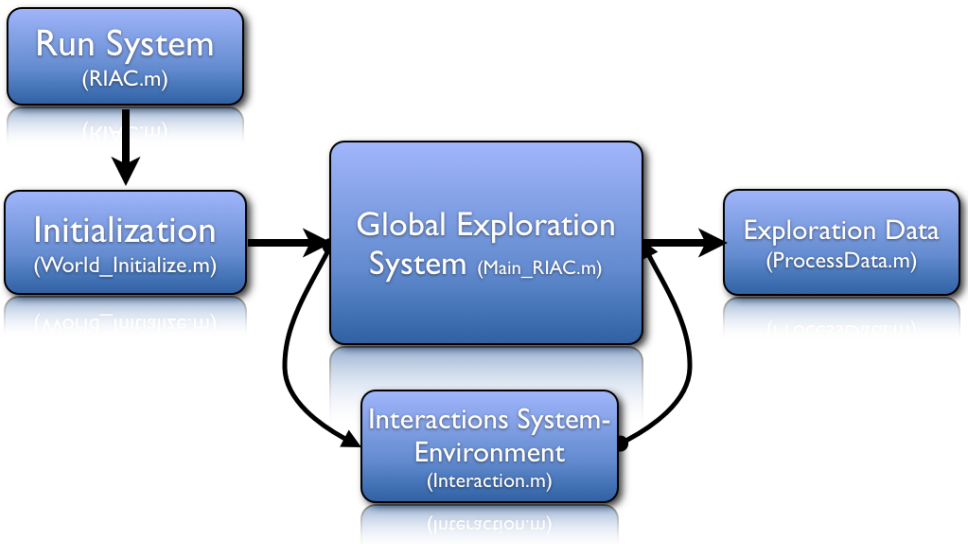
1 What is R-IAC Exploration Explorer?

The Exploration Explorer is a Matlab toolbox designed to allow the evaluation and systematic comparison of various intrinsically motivated exploration algorithms. It includes a graphical interface for tuning exploration parameters, and for choosing the sensorimotor space to be used for evaluation.

2 How to use it ?

This exploration algorithm can be applied in a robotics environment defined by user as an Input-Output function. While running, the program executes actions over the defined world, and increase its knowledge of consequences of it. The list of experimented actions, and the knowledge database are then accessible for the user. Two repertories are accessible in the package, allowing the user to reproduce our experiments, the first, **IAC** (executed using *IAC.m*) contains the first curiosity algorithm proposed in [Oudeyer et al., 2007], and the second, **R-IAC** (executed using *RIAC.m*) contains the new version of Robust-IAC, introduced in [Baranes and Oudeyer, 2009].

3 Global Mechanism



The global mechanism is described using the blocks presented here. The *Global Exploration System* is considered as the heart of the RIAC Exploration Explorer, this one being responsible of action selection and learning. Depending on the tuning of different exploration parameters, introduced using an interface presented in the next part, this block directly implements the different active learning heuristics, presented in R-IAC. To be efficient, this block uses different learning algorithm like GMM/GMR introduced in [Calinon and Billard, 2007] and Approximate Nearest Neighbors, whose Matlab implementation has been provided by Dahua Lin.

The three blocks *Initialization*, *Interactions System-Environment*, and *Exploraton Data* are depending on the mechanical or simulated system where the *Global Exploration System* is introduced. Considering its own physical system, the user will be able to interact with the exploration algorithm using and modifying the three functions visible inside these blocks : *World_Initialize.m*, *Interaction.m*, and *ProcessData.m*

4 Blocks

4.1 Interactions System-Environment : *Interaction.m*

Interactions are defined as the observation of consequences, produced after given candidate actions. In the R-IAC knowledge based framework, an action is defined as a vector $candidateAction \in [0;1]^n$ where each value $candidateAction_i$ corresponds to individual actions of the system, inside the defined environment. These actions could corresponds to high level approaches, like "go forward one meter" motor command, for a mobile robot, or lower, like "move one degree", considering joint positions.

The consequences of actions are defined as a vector $consequence \in R^m$ according to the mapping I representing the forward model of the considered world : $consequence = I(candidateAction)$

The *Interactions System-Environment* block, *Interaction.m* allows the user to configure its own function I, representing a simulated or interfaced robotic setup. The robotic environment is thus implanted using the function:

$$consequence = Interaction(candidateAction, NumFct) \tag{1}$$

Where *candidateAction* and *consequence* are two Matlab vectors and *NumFct* is a parameter defining the number of the considered environment, the function *Interaction.m* being able to represent different forward models.

Here is an example of *Interaction* function:

Algorithm 1 Interaction.m

```
if NumFct == 1
consequence = cos(candidateAction(1)) + sin(candidateAction(2));
end
if NumFct == 2
consequence = [sum(candidateAction(1 : 2)); [candidateAction(3) candidateAction(4) * 2];
end
```

This example shows that a first value which has to be initialized while performing an exploration, is the number of the used function $NumFct$. Once this value chosen, (accessible directly via the interface), the system assign to the *consequence* vector a value depending on the forward model described.

We also have to consider the different number n of inputs (length of *candidateAction*) and outputs m (length of *consequence*) which can be different, according to the value of $NumFct$. Here, if $NumFct = 1$, $n = 2$ and $m = 2$ and if $NumFct = 2$, $n = 4$ and $m = 3$.

To fix the number of inputs/outputs corresponding to each function contained in *Interaction.m*, an initialization stage has to be executed (following part *World_Initialize*), creating a correspondence between each value $NumFct$ and their number of inputs/outputs.

4.2 Initialization : *World_Initialize.m*

The definition of the number of input and output dimensions has to be defined according to the function ($NumFct$) considered. For the example of *Interaction.m* shown previously, we define the file *World_Initialize.m* as containing :

Algorithm 2 Interaction.m

```
if NumFct == 1
dimInput = 2;
dimOutput = 2;
end
if NumFct == 2
dimInput = 4;
dimOutput = 3;
end
```

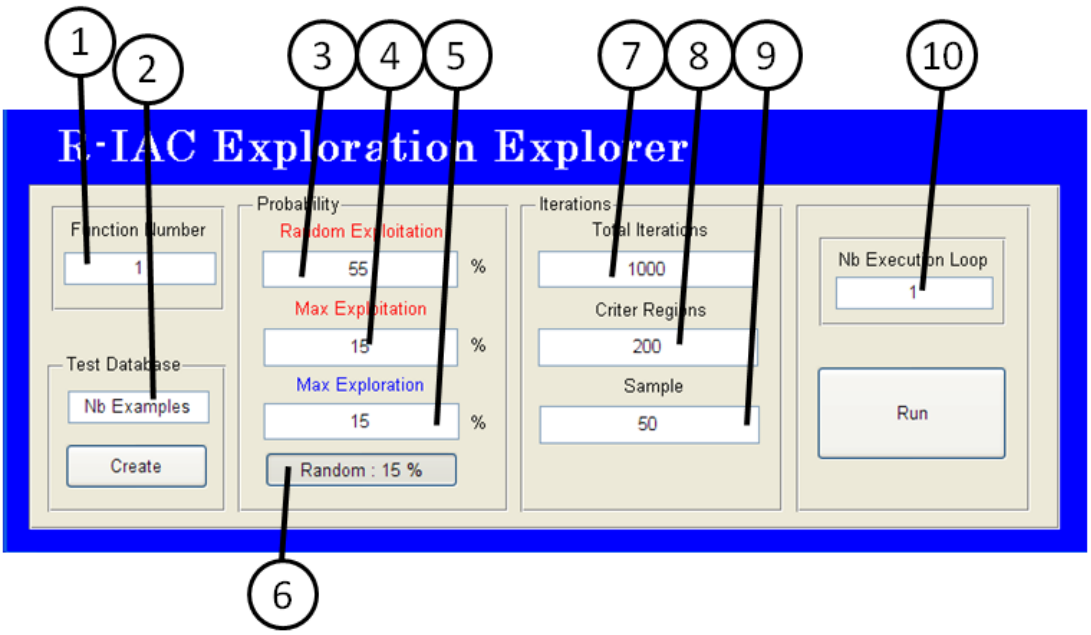
4.3 Exploration Data : *ProcessData.m*

The *Exploration Data* block, containing *ProcessData.m* allows a precise examination of the exploration process, allowing the access to the list of actions performed by the *Global Exploration System*, but also to the global knowledge (forward model) learnt during the exploration. The following data are then extracted in a matlab **.mat** file *ObtainedActions.mat* :

```
Actions         : List of  input actions executed during the exploration
MaxExploitation : Fixed probability of Max Exploitation
MaxExploration  : Fixed probability of Max Exploration
Predictor       : Vals of candidated inputs and outputs obtained
ProbExpl        : Fixed probability of Random Exploitation
ProbRandom      : Fixed probability of Random Exploration
treeRegions     : Tree representing the segmentation of regions
```

5 Global Exploration System : Tuning Interface

Here is the global tuning interface of R-IAC Exploration Explorer. This section details all features avalaible to launch an exploration process.



(1) **Function Number : value of** $NumFct$
(2) **Number of Examples in Test Database**
(3) **Exploration Criterion : Random Exploitation (%)**
(4) **Exploration Criterion : Max Exploitation (%)**
(5) **Exploration Criterion : Max Exploration (%)**
(6) **Exploration Criterion : Random Exploration (%)**
(7) **Total Number of Iterations**
(8) **Criter Regions : Maximal Size of a Region**
(9) **Sample : Minimal Number of Examples inside a Region to Use Learning Progress**
(10) **Number of Execution of the Exploration**
(Run) **Launch the Exploration**

5.1 Function Number

The value of **Functin Number** corresponds to the number of the function $NumFct$, and define the environment which has to be used for the exploration process.

5.2 Test Database

Allow the creation of a test *test database* recorded in **testdatabase.mat**, by performing random actions inside the environment described by the **Function Number**

5.3 Exploration Criterion : Random Exploitation

Probability (%) to choose the ***Learning Progress Maximization Exploration Mode***

5.4 Exploration Criterion : Max Exploitation

Probability (%) to choose the ***Error Maximization Exploration Mode***

5.5 Exploration Criterion : Max Exploration

Probability (%) to choose the ***Error Maximization Exploration Mode*** considering the whole space and not the region chosen by the ***Learning Progress Maximizatin Exploratin Mode***.

5.6 Exploration Criterion : Random Exploration

Probability (%) of Random Exploration, depending of the three previous values (3, 4 and 5).

5.7 Total Number of Iterations

Total number of sensorimotor experiments (actions) to perform.

5.8 Criter Regions : Maximal Size of a Region

Number maximal of learning exemplars which can be contained inside a region. Once this size reached, the region is split.

5.9 Sample

Number of exemplars needed inside a region to begin the computation of the learning progress

5.10 Number of Exploration

Number of execution of the R-IAC algorithm, described with the previously fixed parameters. This can be used for statistical studies.

5.11 Run

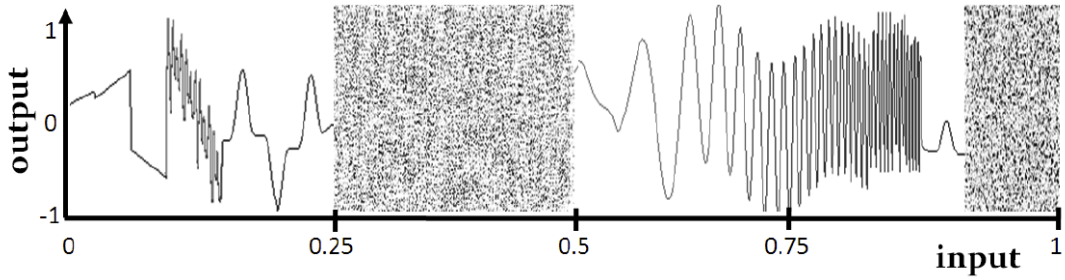
Execute the R-IAC algorithm with fixed parameters. Once the exploration finished, a file name **ObtainedActions.mat** is created by **ProcessData.m**

6 Demonstrations

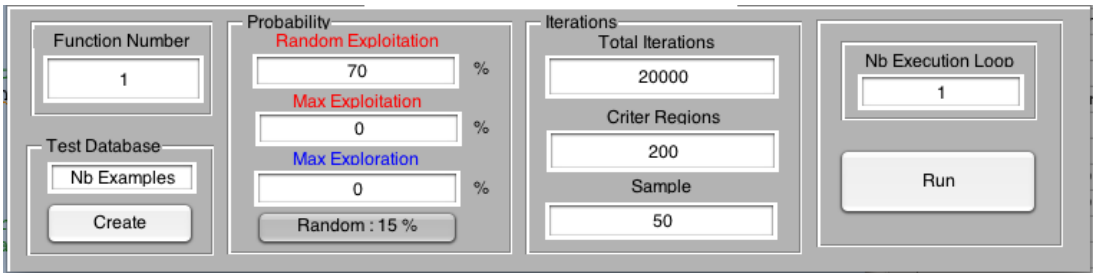
Different demonstration functions are proposed as an introduction to *R-IAC Exploration Explorer*. The demo part are directly introduced in the files *World_Initialize.m*, *Interaction.m*, and *Process-Data.m* as different sub-functions (determined using the **Function Number** $NumFct$).

6.1 1-Dimension Approach : $NumFct = 1$

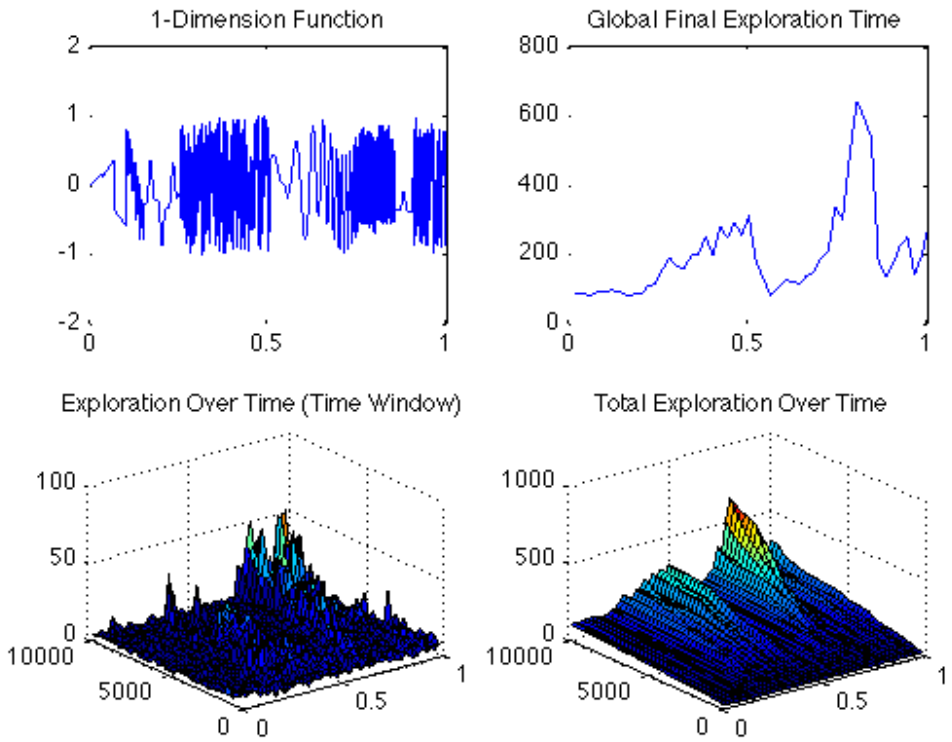
We study here the evolution of the exploration in a simple 1-dimensional environment containing two noisy parts $[0.25; 0.5]$ and $[0.8; 1]$ and an "incresing difficulty part" $[0.5; 0.8]$ showed in the following figure :



This experiment correspond to the function $NumFct = 1$ of *Interaction.m*, using the following parameters :



We can observe the exploration described using histograms of "Exploration over time":



Here, we thus observe that the R-IAC algorithm is able to focus on the increasing complexity part $[0.5; 0.8]$ and is less interested to learn the noisy part.

6.2 2-Dimension Arm-Camera Experiment : $NumFct = 2$

We designed a simulated mechanical system, using the Matlab robotics toolbox [31]. It consists of a robotic arm using two degrees of freedom, represented by the two rotational axes q_1, q_2. The upper part of the arm has been conceived as a bow, which creates a redundancy in the system: for each position and orientation of the tip of the arm, there are two corresponding possible articulatory/joint angle configurations.

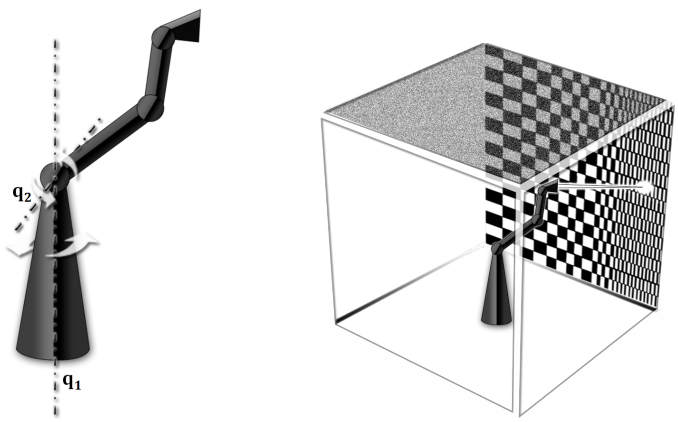
This system's sensory system consists in a one-pixel camera, returning an intensity value p, set on its extremity. The arm is put in a cubic painted environment V, whose wallpapers are visible to the one-pixel camera, according to articulatory configurations.

Intensity values measured by the cameras are consequences of both environment V and rotational axes q_1,q_2. So, we can describe the system input/output mapping with two input dimensions, and one output as:

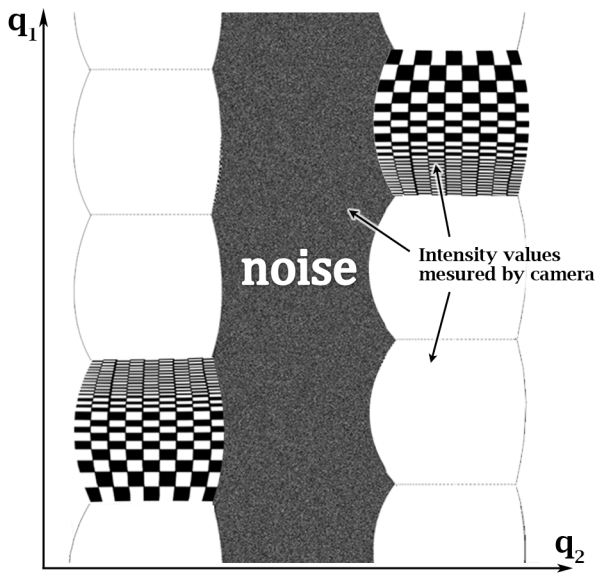
$$p = V(q_1, q_2) \tag{2}$$

Thus, in this system the mapping to be learnt is state independent since here trajectories are not considered (only end positions are measured) and the perceptual result of applying motor joint angle commands does not depend on the starting configuration.

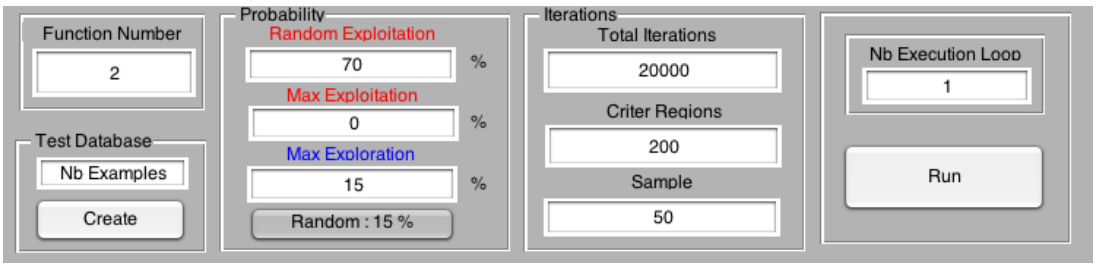
The front wall consists of an increasing precision checker, conceived with a black and white pattern. The designed ceiling contains animated wallpaper with white noise, returning a random value to the camera when this one is watching upward bound. Finally, other walls and ground are just painted in white.



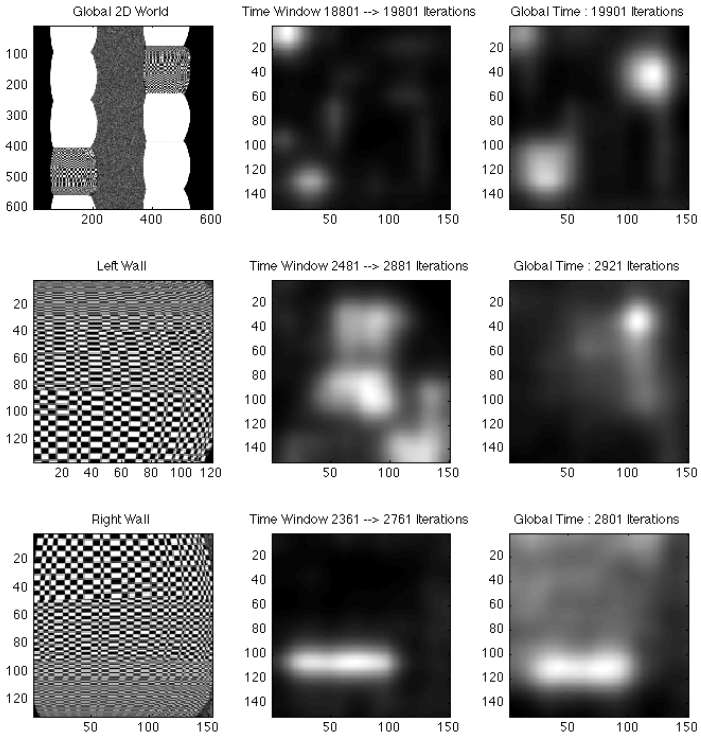
Because the system has just two motor dimensions and one sensory dimension, it can be visualized using a 2D projection on a plane such as in the next figure (In our Matlab implementation, we use a 2-Dimension Image). This projection shows a central vertical zone corresponding to the dynamic noise projected on the ceiling. Then, we can easily distinguish the front wall, represented on both sides of the noisy area, because of the redundancy of the arm. The remaining white parts correspond to other walls and the floor.



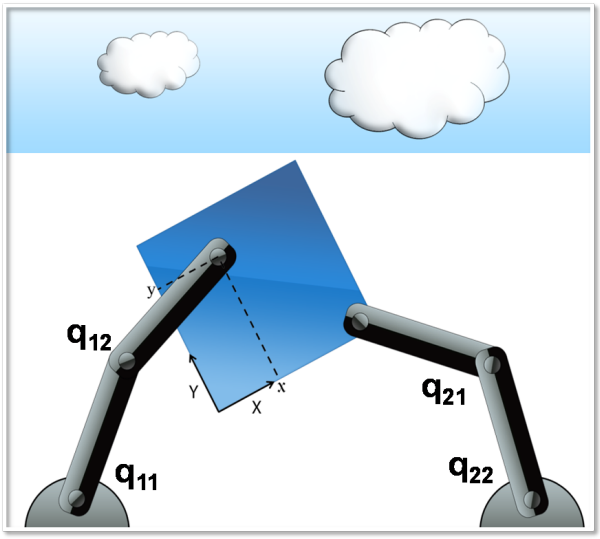
This experiment corresponds to the function $NumFct = 2$ of *Interaction.m*, using the following parameters :



We can observe the following kind of graphs representing using movies, the exploration over time inside the whole world, and also, inside subpart of it (the front wall)



6.3 Hand-Eye Experiment : $NumFct = 3$

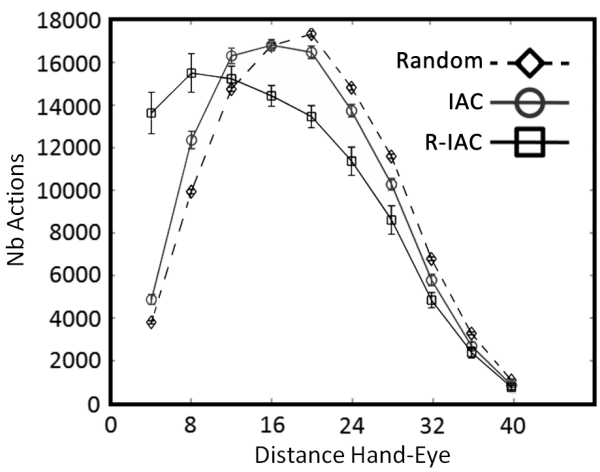


In this experiment, a simulated robot has two 2-D arms, each with two links and two revolute joints whose angles are controlled by motor inputs $q_1 1, q_1 2, q_2 1, q_2 2$ (see figure). On the tip of one of the two arms is attached a square camera capable to detect the sensory position (x,y) of point-blobs relative to the square. These point-blobs can be either the tip of the other arm or clouds in the sky (see figure 12). This means that when the right arm is positioned such that the camera is over the clouds, which move randomly, the relation between motor configurations and perception is quasi-random. If on the contrary the arms are such that the camera is on top of the tip of the other arm, then there is an interesting sensorimotor relationship to learn. Formally, the system has the relation:

$$(x, y) = E(q_1 1, q_(12), q_2 1, q_2 2) \tag{3}$$

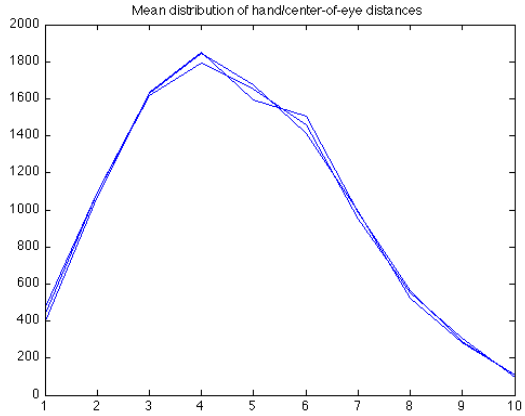
- The camera is placed over the white wall: nothing has been detected: (x,y)= (-10, -10);
- The camera is on top of the left hand: the value (x,y) of the relative position of the hand in the camera referential C is taken. According to the camera size, the x and y values are in the interval [0; 6];
- The camera is looking at the window: Two random values (x,y) playing the role of random clouds displacement are chosen for output. The interval of outputs corresponds to camera size.
- The camera is looking at the window and sees both hand and cloud: the output value (x,y) is random, like if just a cloud had been detected.

A first study of what happens consists in monitoring the distance between the center of the eye (camera), and the hand (tip of the other arm). A small distance means that the eye is looking the hand, and a high, that it is focusing on clouds (noisy part) or on the white wall. The next figure shows histograms of these distances statistically obtained:



The function *ProcessData.m* applied considering this system (with $NumFct = 1$) allows to obtain this king of statistics, considering one system at a time. As an example, using R-IAC 3 times, with

20000 experiments and exploration Random = 30% , Random Exploitation = 70%, we obtain the following kind of statistics, showing a curve, for each launch.



References

[Baranes and Oudeyer, 2009] Baranes, A. and Oudeyer, P.-Y. (2009). R-iac: Robust intrinsically motivated active learning. In *Proc. of the IEEE International Conference on Learning and Development.*

[Calinon and Billard, 2007] Calinon, S., G. F. and Billard, A. (2007). On learning, representing and generalizing a task in a humanoid robot. *IEEE Transactions on Systems, Man and Cybernetics, Part B.*

[Oudeyer et al., 2007] Oudeyer, P.-Y., Kaplan, F., and Hafner, V. (2007). Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):pp. 265–286.