Adaptability Features in a Nonlinear System-based Swarm Robotic Framework

Kevin DENAMGANAI - 2160104105
Electrical and Information Systems
Osaka Prefecture University

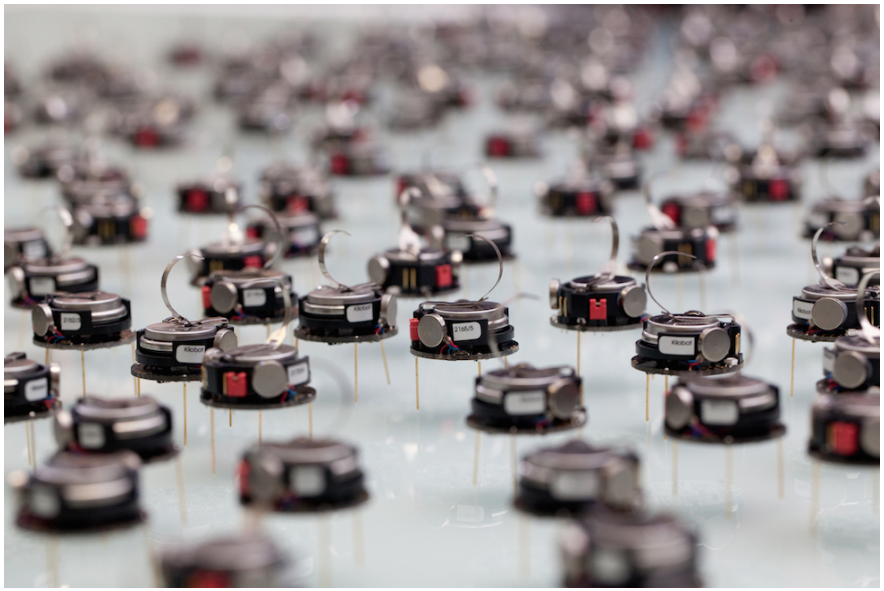
August 2017

Outline

Outline

Swarm Robotics

- Inspired by nature (bees, ants, birds...)
- Parallelized : efficient workload management.
- Modular : can alleviate easily when issues happen.



Nonlinear Systems VS. Algorithms

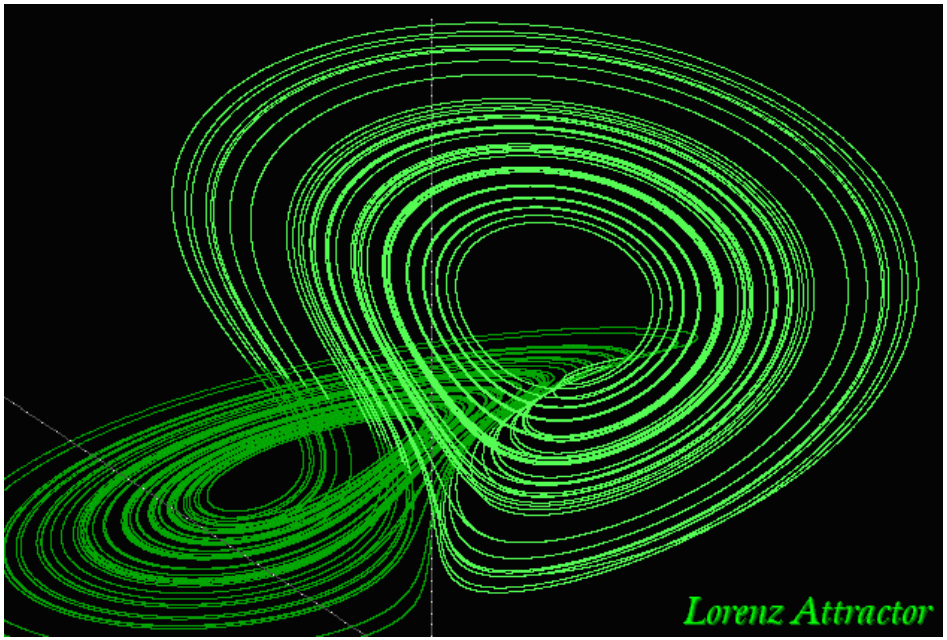


Figure – Behavior created by Lorenz's attractor nonlinear system.

$$\begin{cases} \dot{\vec{X}} &= f(\vec{X}, \vec{U}) \\ \vec{Y} &= g(\vec{X}) \end{cases}$$

VS.

Data: this text

Result: how to write algorithm with LaTeX2e

initialization;

while *not at end of this document* **do**

  read current;

  if *understand* **then**

    go to next section;

    current section becomes this one;

  else

    go back to the beginning of current section;

  end

end

Algorithm 1: How to write algorithms

Outline

Early Problem Formulation

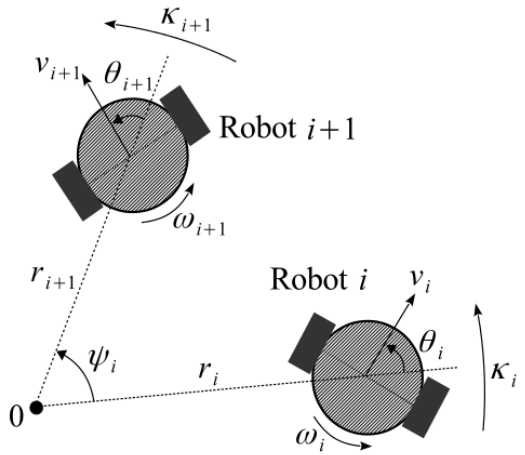
Apply Nonlinear Systems, more specifically Coupled-Oscillators, to the control of Swarms of Robots.

$\Rightarrow$ Transfer all the advantages of Nonlinear Systems to the realm of Swarm Robotics.

"Stability analysis of mobile robot formations based on synchronization of coupled oscillators"

(T.Nakamura et al., IFAC NecSys 2016)

- Target encircling behavior using a Kuramoto Phase's Coupled Oscillators-based nonlinear system.
- Apply Stability Analysis' tools in the Swarm Robotics field : stable control law.



	$\frac{l}{N} \in [0, \frac{1}{4})$	$\frac{l}{N} \in (\frac{1}{4}, \frac{1}{2}]$	$\frac{l}{N} \in [\frac{1}{2}, \frac{3}{4})$	$\frac{l}{N} \in (\frac{3}{4}, 1]$
$\Omega \geq 0$	$k_\omega > 0$			
	$\varepsilon > 0$		$\varepsilon < 0$	
	$k_v > 0$, $a > 0$	$k_v < 0$, $a < 0$	$k_v > 0$, $a > 0$	$k_v < 0$, $a < 0$
$\Omega \leq 0$	$k_\omega < 0$			
	$\varepsilon < 0$		$\varepsilon > 0$	
	$k_v < 0$, $a < 0$	$k_v > 0$, $a > 0$	$k_v < 0$, $a < 0$	$k_v > 0$, $a > 0$

"Stability analysis of mobile robot formations based on synchronization of coupled oscillators"

(T.Nakamura et al., IFAC 2016)

Non-linear System for robot i :

$$\begin{bmatrix} \dot{r}_i \\ r_i\kappa_i \\ \dot{\theta}_i \end{bmatrix} = \begin{bmatrix} \cos\theta_i & 0 \\ \sin\theta_i & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \nu_i \\ \omega_i \end{bmatrix}$$

Control Law :

$$\nu_i = k_\nu \left\{ f(r_i, \hat{r}_i)\cos\theta_i + r_i g_{\Omega,\epsilon}(\psi_i)\sin\theta_i \right\}$$

$$\omega_i = k_\omega \left\{ r_i g_{\Omega,\epsilon}(\psi_i)\cos\theta_i - f(r_i, \hat{r}_i)\sin\theta_i \right\}$$

with :

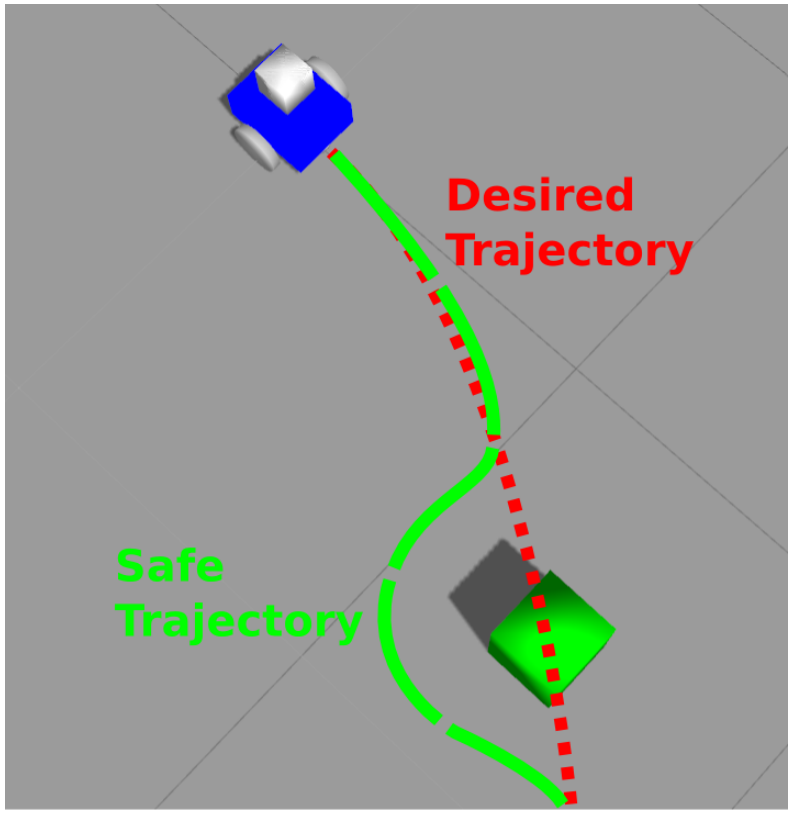
$$\dot{r}_i = f(r_i, \hat{r}_i) = a r_i \left(1 - \frac{{r_i}^2}{{\hat{r}_i}^2}\right)$$

$$\kappa_i = g_{\Omega,\epsilon}(\psi_i) = \Omega + \epsilon \sin\psi_i$$

Outline

- Adaptation to the environment : the current system cannot adapt to the objects that are external to the swarm of robots : environment's obstacles.

⇒ Design and integration to the system of an obstacle avoidance control law.



Outline

Outline

How to create an Obstacle Avoidance Behaviour ?

Previously :

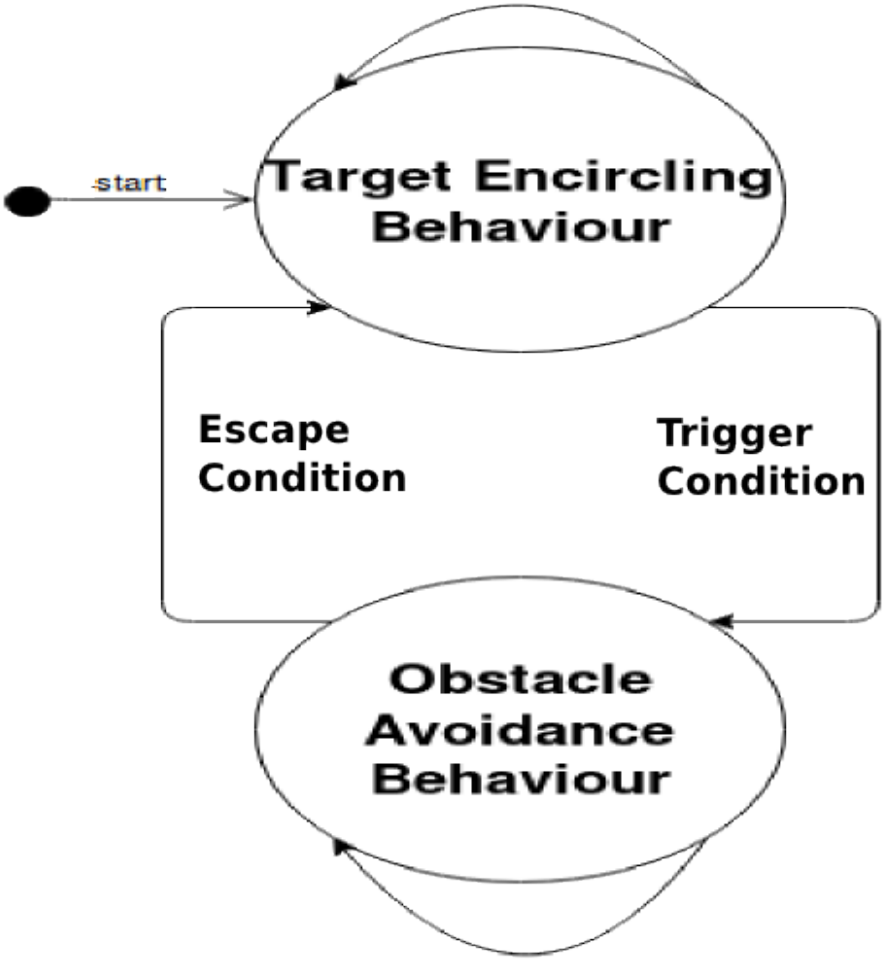
- Target Encircling behaviour

New behaviour :

- Obstacle Avoidance behaviour by partially encircling around the obstacle in order to avoid it.

Hybrid System

...with a State-machine :



Discussion

- Design of an effective static obstacle avoidance behaviour.
- Integration of the obstacle avoidance behaviour to the Nonlinear System-based Swarm Robotic Framework.

- Problem : dependence to a state machine prevents us from using stability analysis tools to investigate the system...

Outline

Approach : Hierarchically Stacked Nonlinear Controllers

Previously :

- Hybrid System with level-based controller :
  - Level 0 : State Machine that controls which behaviour to use.
  - Level -1 : Nonlinear controllers that controls what kind of robot dynamic to apply.

$\Rightarrow$ Rely solely on Nonlinear Controllers organized in a level-based / hierarchical way.

Approach : Hierarchically Stacked Nonlinear Controllers

- Level 1 : user-fixed parameter $\hat{\hat{r}}_i$: desired radius of the target-encircling behavior.
- Level 0 : Nonlinear Controller for the radius parameter $\hat{r}_i$ in order to avoid obstacles.

$$\hat{r}_i \longrightarrow \hat{\hat{r}}_i$$

- Level -1 : Nonlinear Controller for the Target-Encircling Behaviour, that uses the radius parameter $\hat{r}_i$.

$$r_i \longrightarrow \hat{r}_i$$

Obstacle Avoidance Control Law

Level 0 : Nonlinear Controller for the radius parameter $\hat{r}_i$ of the target-encircling behaviour :

$$\dot{\hat{r}}_i(t) = \{1 - \cos([\theta^i_{obs}(t)])\} h_1(\hat{r}_i(t)) - \cos([\theta^i_{obs}(t)]) h_2\left(i, [\theta^i_{obs}(t)]\right)$$

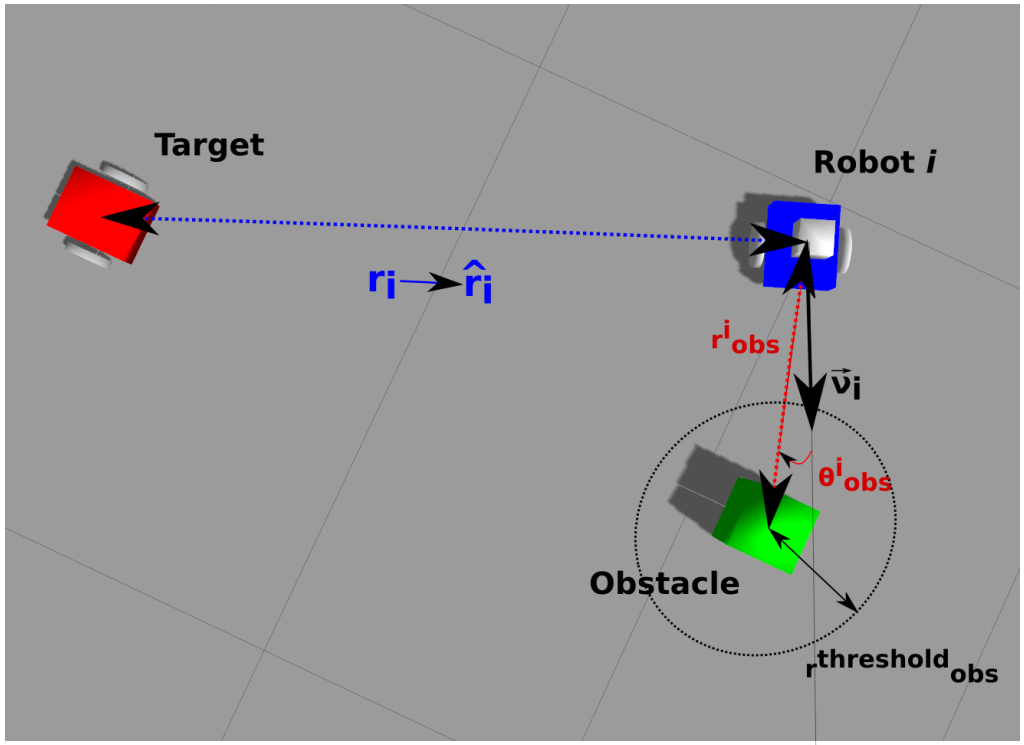
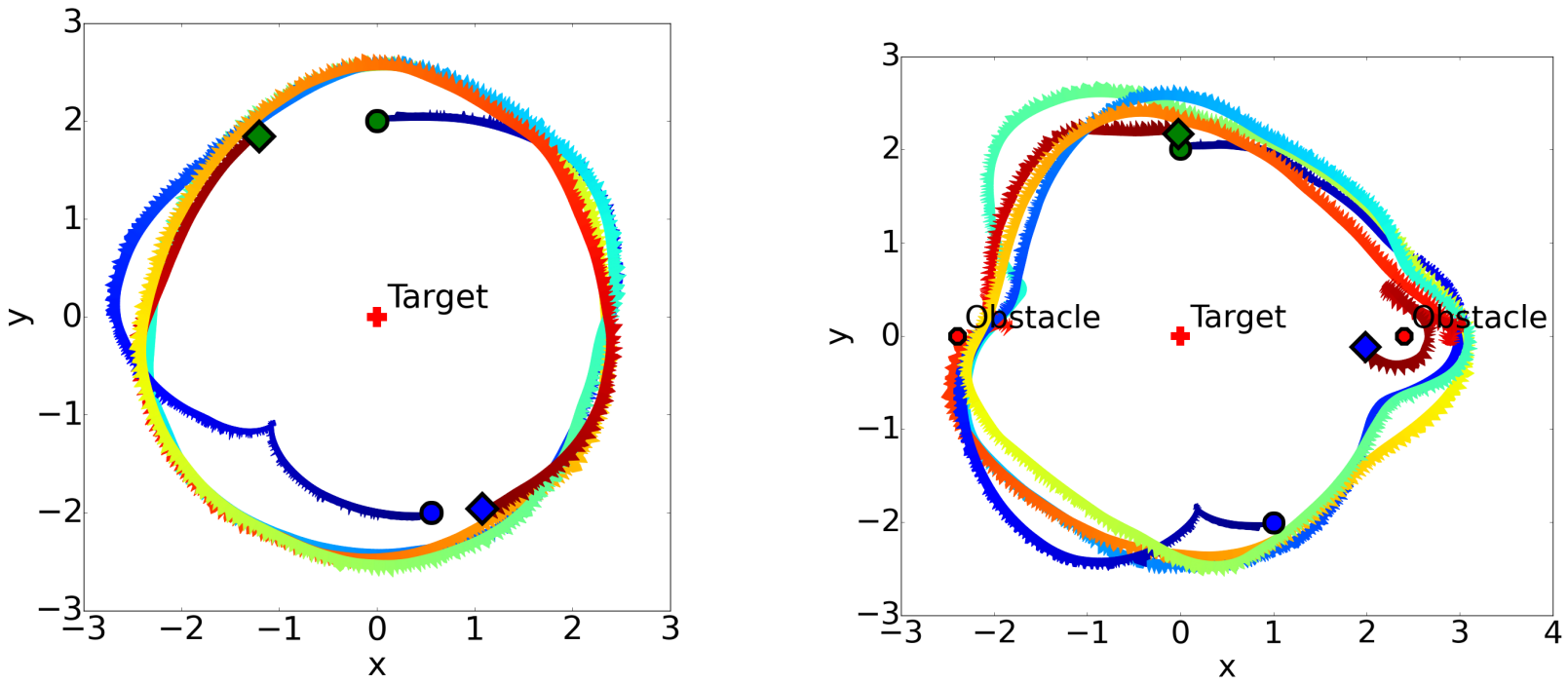


Figure – Situation near an obstacle with the deviation-based approach.

Validation of the Obstacle Avoidance Control Laws



(a) (b)

Figure – Trajectories of two robots in a swarm (circles : the trajectories at $t = 0$; squares : the trajectories at the end of the simulations) : (a) without obstacle and (b) with two obstacles (control law relying on the Trajectory Deviations).

Discussion

- Design of an obstacle avoidance (deviation-based) control law, with an approach based entirely on Nonlinear Systems.
- Hierarchically Stacked Nonlinear Controllers (HSNC) approach is still in need of some improvements...

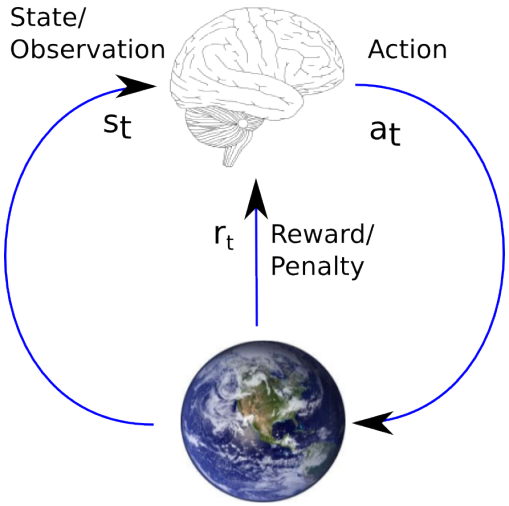
- Problem : How to improve the HSNC approach ?

$\Rightarrow$ Investigate the **coupling** of Deep Reinforcement Learning with Nonlinear Systems.

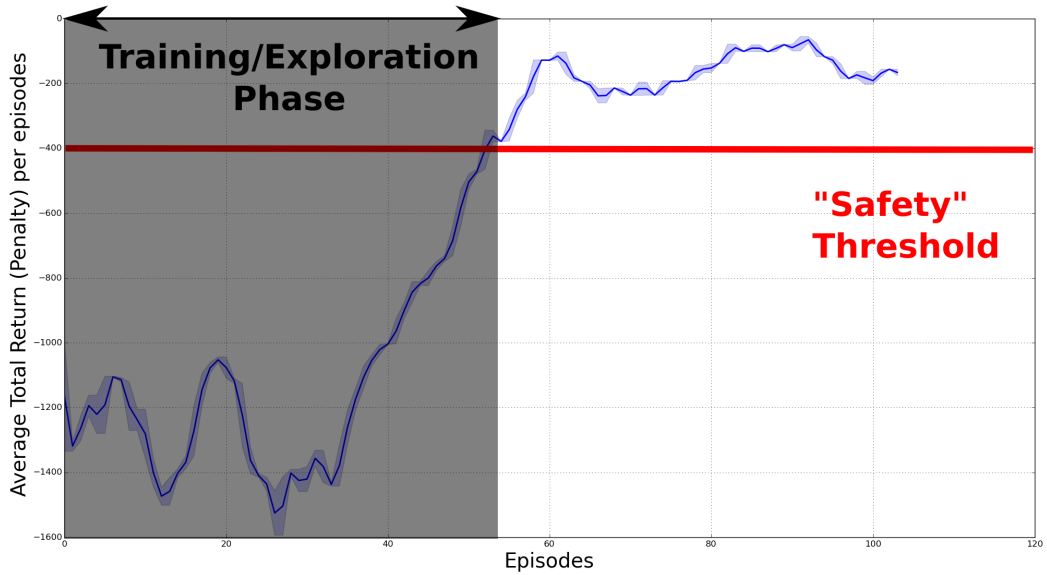
Outline

Deep Reinforcement Learning

Agent-Environment Loop :



Evolution of an Agent's Total Return through time :



Total Return :

$R_T = \sum_{t=0}^{T} r_t$

The higher R_T, the better the Agent is able to fulfill its goal.

Training/Exploration Phase :

Trials and Errors $\Rightarrow$ HARMFUL for the controller's real system.

Our Framework

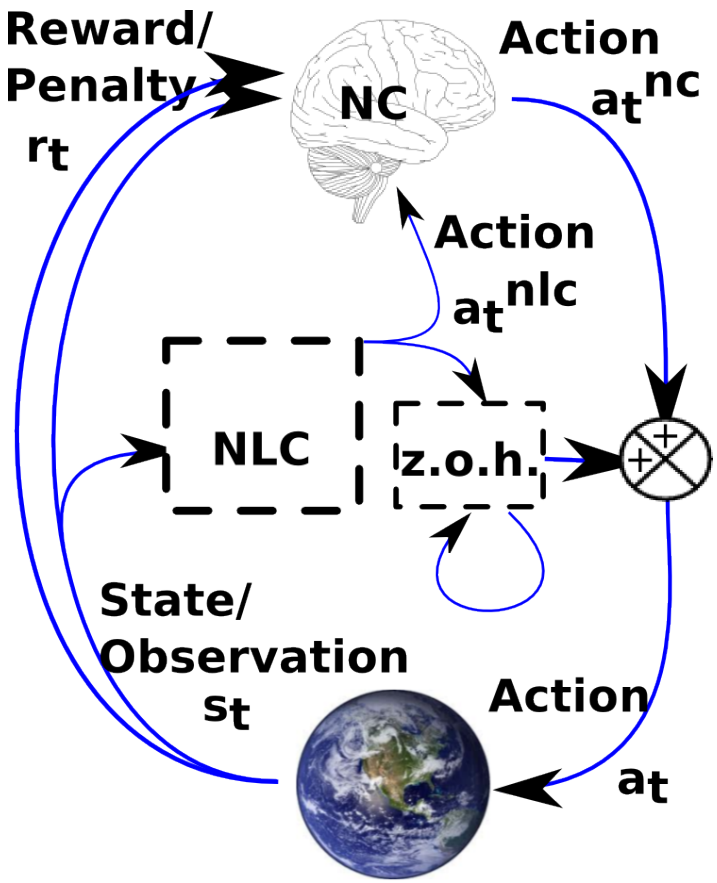


Figure – Reinforced NonLinear-based Neural Controller paradigm.

Reward Shaping,
based on Nonlinear Systems' Energy :

Nonlinear System :

$$\dot{x}(t) = f(x(t))$$
$$f(x_0) = 0$$

Reward Function :

$$r_t = \frac{1}{E_t + \alpha\|x_0 - x(t)\|}$$

Energy :

$$E_t = \frac{1}{2}\dot{x}(t)^T\dot{x}(t)$$

Numerical Results

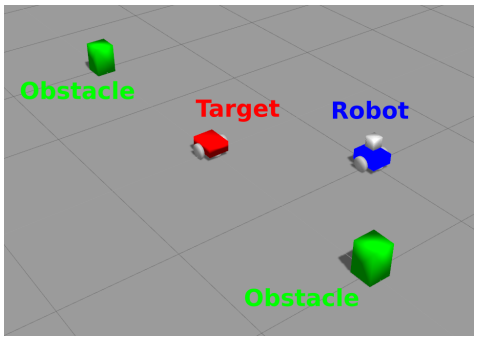


Figure – Experimental setup of the environment.

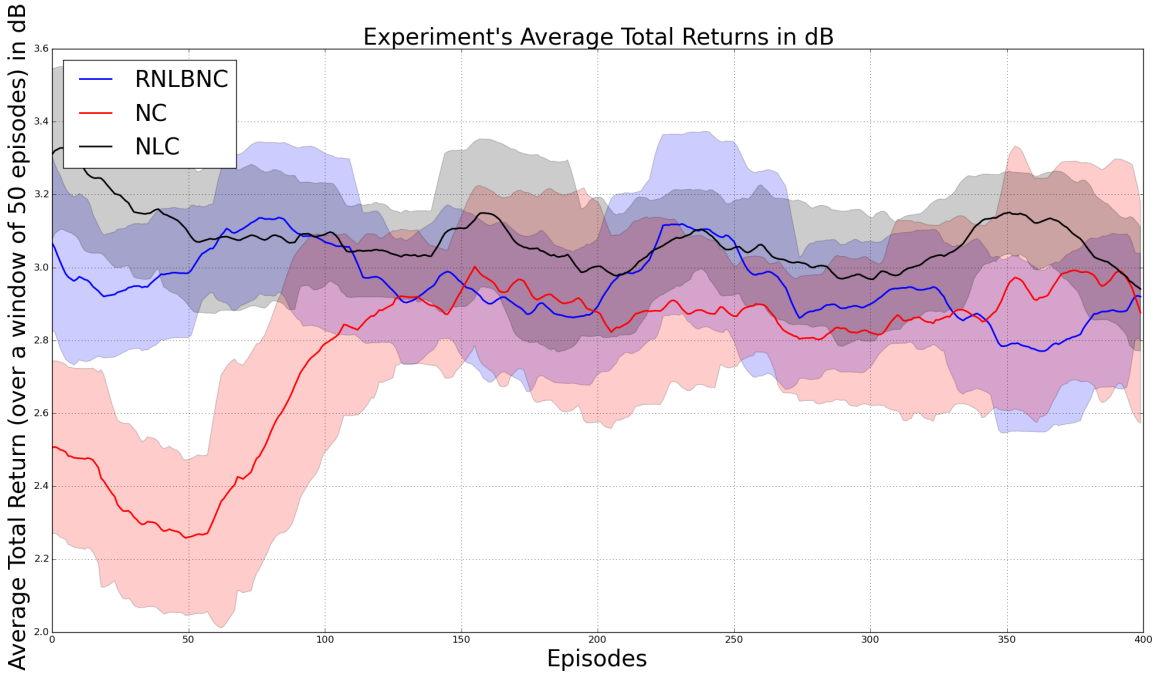


Figure – Total reward (in decibels) per episode for the Neural Controller only (NC), the NonLinear controller only (NL), and the Reinforced NonLinear-Based Neural Controller (RNLBNC).

Outline

Conclusion

- Design of a Nonlinear System-based Swarm Robotic Framework : exhibits simple adaptation features to the environment, through obstacle avoidance behavior relying solely on Nonlinear Systems.
- Design of the RNLBNC paradigm : paves the way for the combination of Nonlinear Systems with Deep Reinforcement Learning, in order to exhibit more complex adaptation features.

Conclusion

Thank you for your attention.

Vector Field Analysis

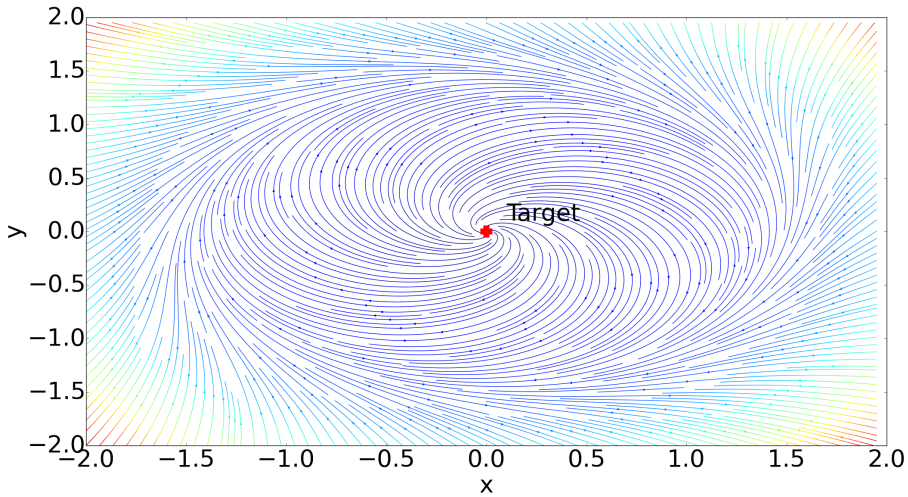


Figure – Vector field of the encircling behavior control law only.

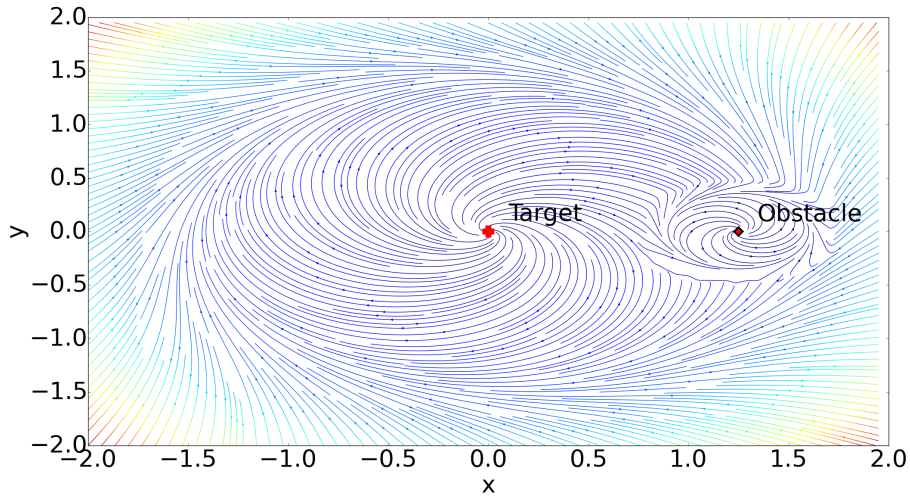


Figure – Vector field of the encircling behavior control law integrating the obstacle avoidance control law.

Switch between behaviours ?

Condition to :

- Triggers the obstacle avoidance behaviour :

$$r^i_{obs} < r^{trigger}_{obs}$$
$$|\theta^i_{obs}| < \theta^{trigger}_{obs}$$

- Escape the obstacle avoidance behaviour :

$$|\omega_i| \leq \alpha |a k_\omega \hat{r}_i|$$

where $\alpha > 0$ is a normalizing constant.

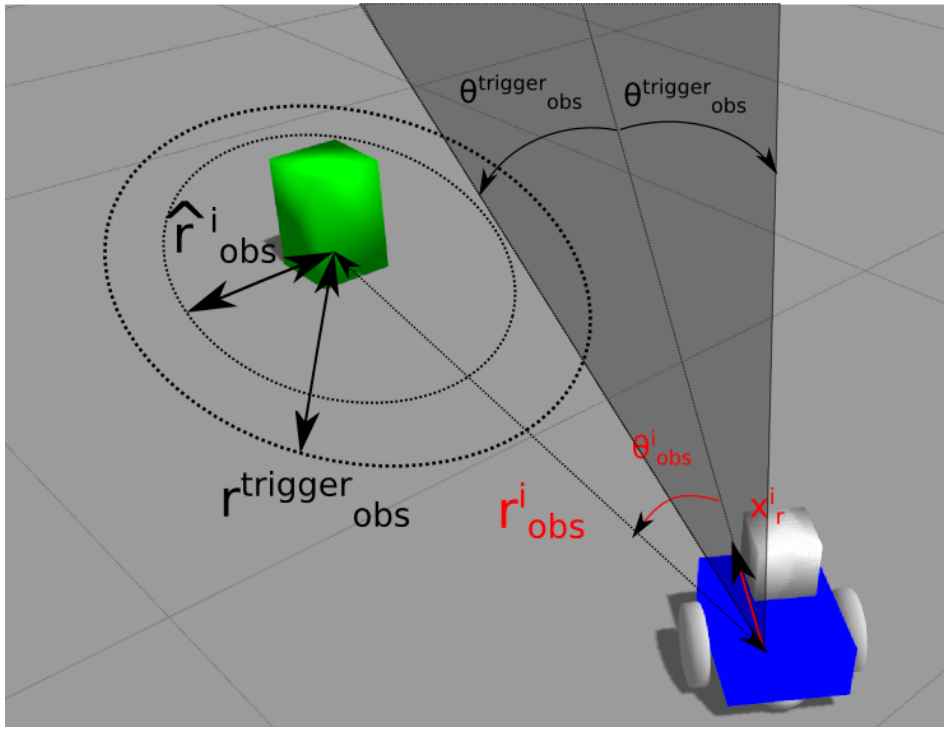


Figure – Situation near an obstacle.

Obstacle Avoidance Control Law

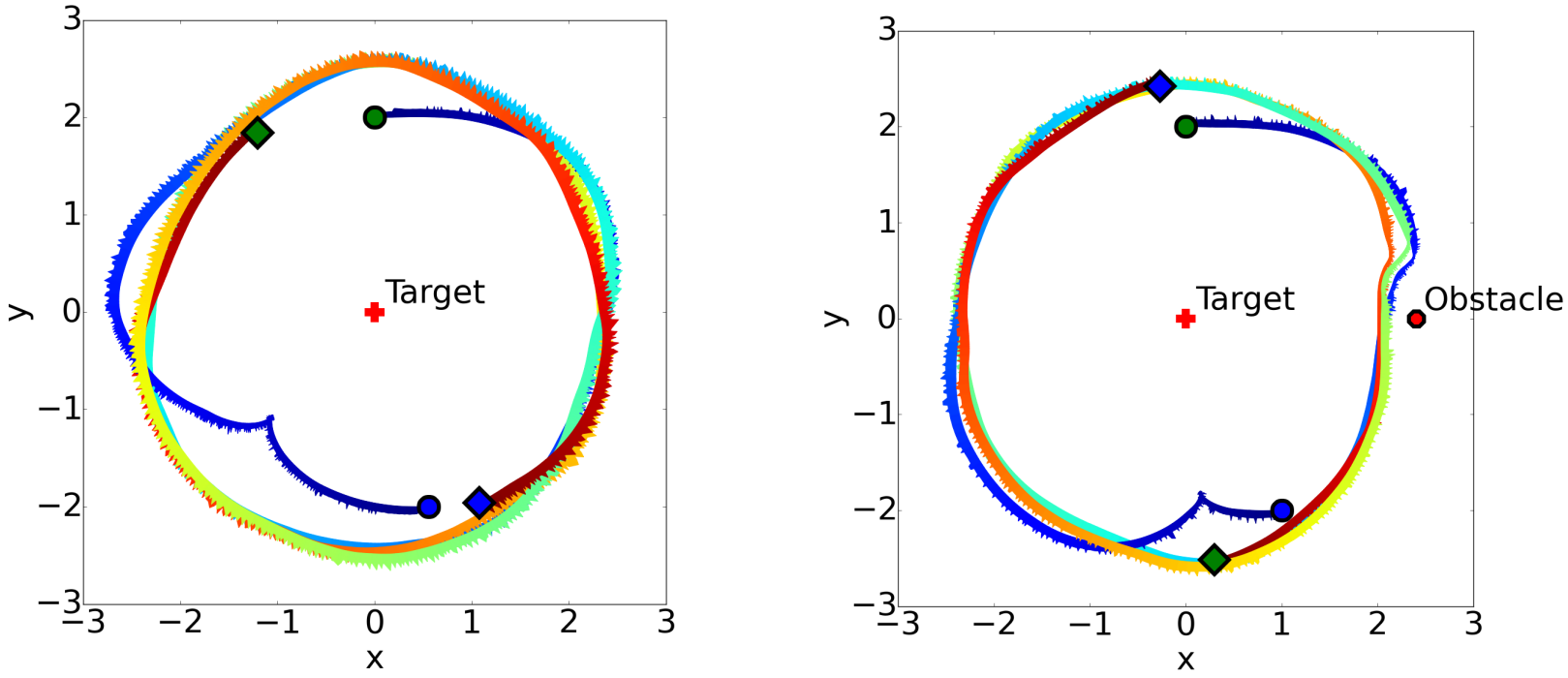
Idea : Let us partially encircle around the obstacle in order to avoid it.

$$\nu_i^{obs} = k_\nu \left\{ f(r_{obs}^i, \hat{r}_{obs}^i) \cos\theta_i^{obs} + r_i g_{\Omega_{obs},0}(\psi_i) \sin\theta_i^{obs} \right\}$$

$$\omega_i^{obs} = k_\omega \left\{ r_{obs}^i g_{\Omega_{obs},0}(\psi_i) \cos\theta_i^{obs} - f(r_{obs}^i, \hat{r}_{obs}^i) \sin\theta_i^{obs} \right\}$$

with $\theta_i^{obs} = \pi - \theta_{obs}^i$ and Ω_{obs} : desired velocity of the rotation around the obstacle

Validation of the Obstacle Avoidance Control Laws



(a) (b)

Figure – Trajectories of two robots in a swarm (circles : the trajectories at $t = 0$; squares : the trajectories at the end of the simulations) : (a) without obstacle and (b) with one obstacle (control law relying on the Hybrid System).

Obstacle Avoidance Control Law

Level 0 : Nonlinear Controller for the radius parameter $\hat{r}_i$ of the target-encircling behaviour :

$$h_1(r) := k_{\hat{r}}\left(\hat{\hat{r}}_i - r\right)$$

$$h_2(i,[\theta]) := \frac{\text{sign}([\theta])}{\log\left\{1 + k_{\hat{r}}|c_i(r_{obs}^{threshold})|\right\}}$$

where

$$c_i(r_{obs}^{threshold}) := r_{obs}^{i} - r_{obs}^{threshold} \geq 0$$

$$\text{sign} \ : \ x \mapsto \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases},$$

$$[\,] \ : \ \theta \mapsto \begin{cases} \frac{\pi}{2} & \text{if } |\theta| \geq \frac{\pi}{2} \\ \theta & \text{otherwise} \end{cases}.$$

Level −1 : Nonlinear Controller for the Target-Encircling behaviour robot dynamic (following previous works) :

$$\nu_i = k_\nu \left\{ f(r_i, \hat{r}_i) \cos\theta_i + r_i g_{\Omega,\epsilon}(\psi_i) \sin\theta_i \right\}$$

$$\omega_i = k_\omega \left\{ r_i g_{\Omega,\epsilon}(\psi_i) \cos\theta_i - f(r_i, \hat{r}_i) \sin\theta_i \right\}$$

Nonlinear Systems & Deep Reinforcement Learning

	Accountable	Robust	Online Adaptability	Training Phase	...
Nonlinear Systems	✓	✓			...
Deep Reinforcement Learning		✓ (after the training phase only...)	✓	✓	...

$\Rightarrow$ great synthesis potential

Issues tackled by the current report :

1. Adaptation to the environment : the current system cannot adapt to the objects that are external to the swarm of robots : environment's obstacles.
2. Swarm Architecture : The current formulation is not viable in a distributed swarm architecture : it relies on a global reference frame that is currently not accessible from the viewpoint of a each robot.

Centralized vs Distributed Architecture

Problem :

- Global sensor - not scalable
- Strong computational requirements - not lightweight

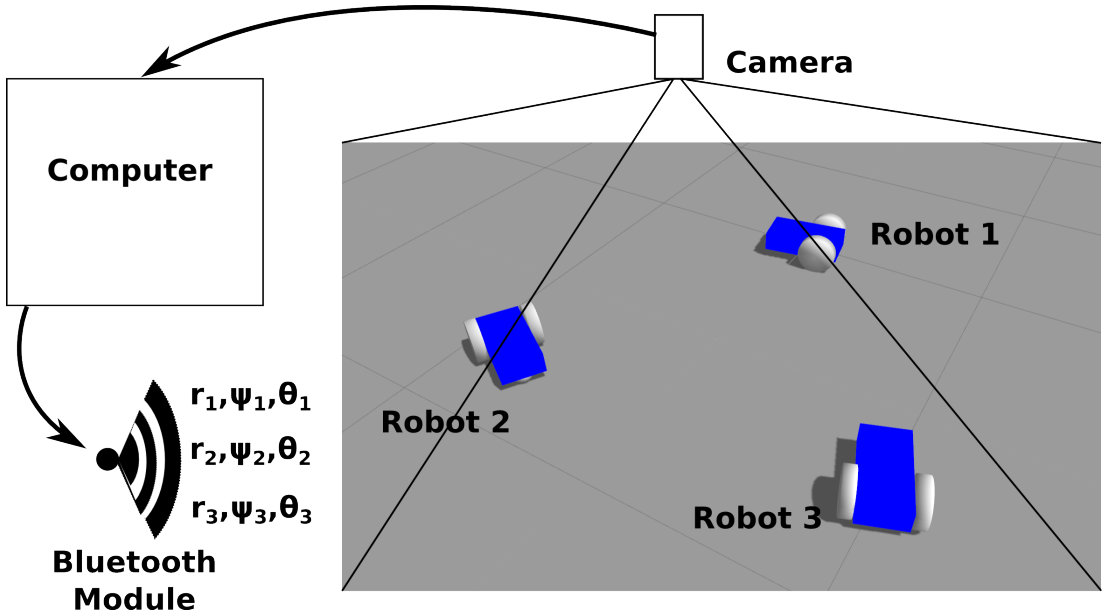


Figure – Architecture of the pipeline in previous experimentation (Tsukiji et al.).

Centralized vs Distributed Architecture

Independently follow the paradigm :

- SENSE
- THINK
- ACT

in an efficient, modular, lightweight and scalable way.

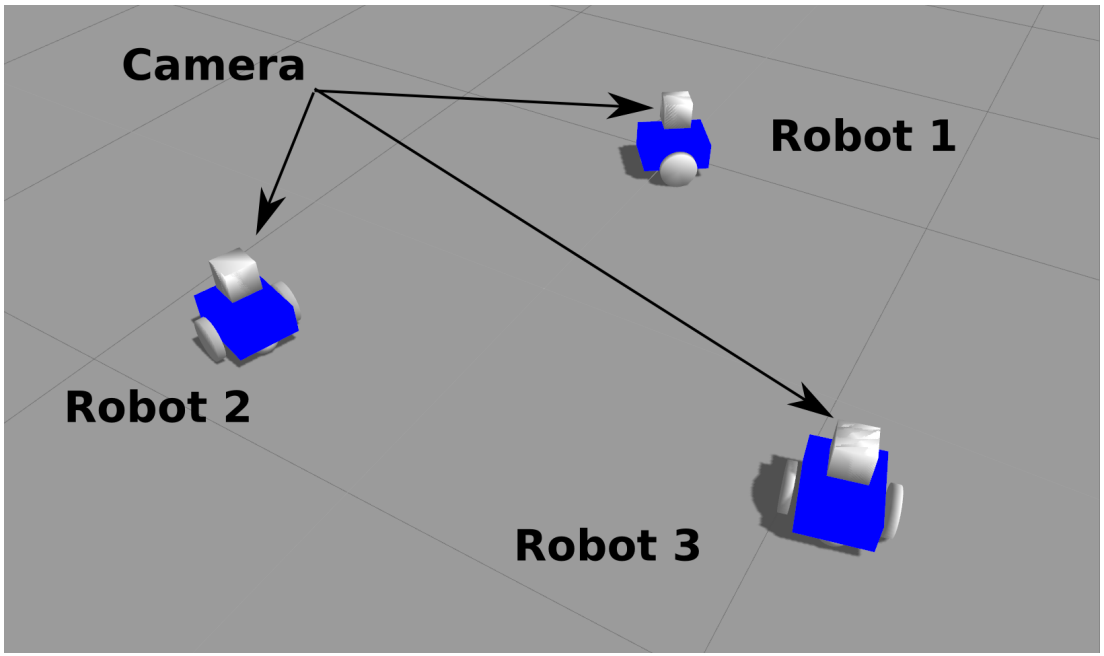


Figure – Distributed architecture of the pipeline in this report's implementation.

Global Reference Frame vs Multiple Robot-focused Local Reference Frames

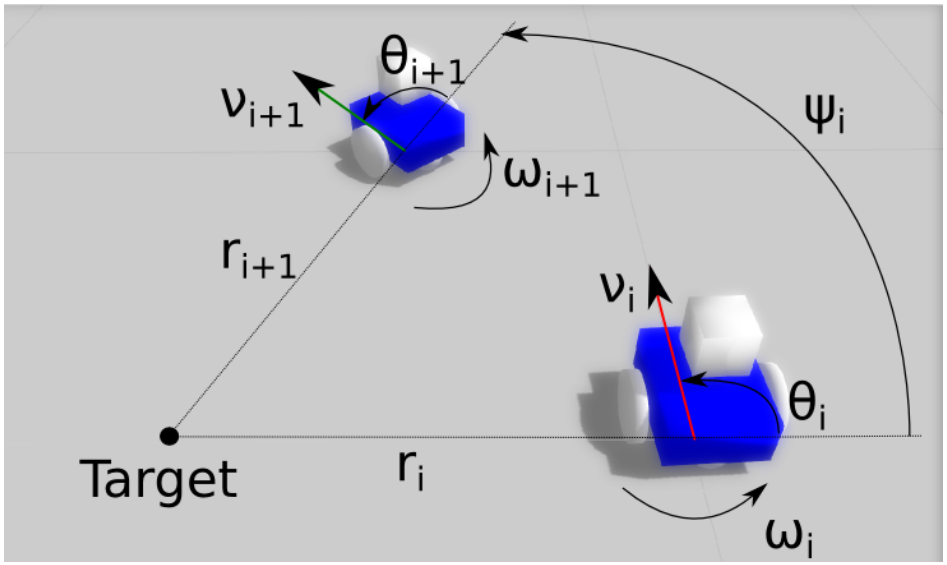


Figure – Two-wheeled robots in their Global Reference Frame.

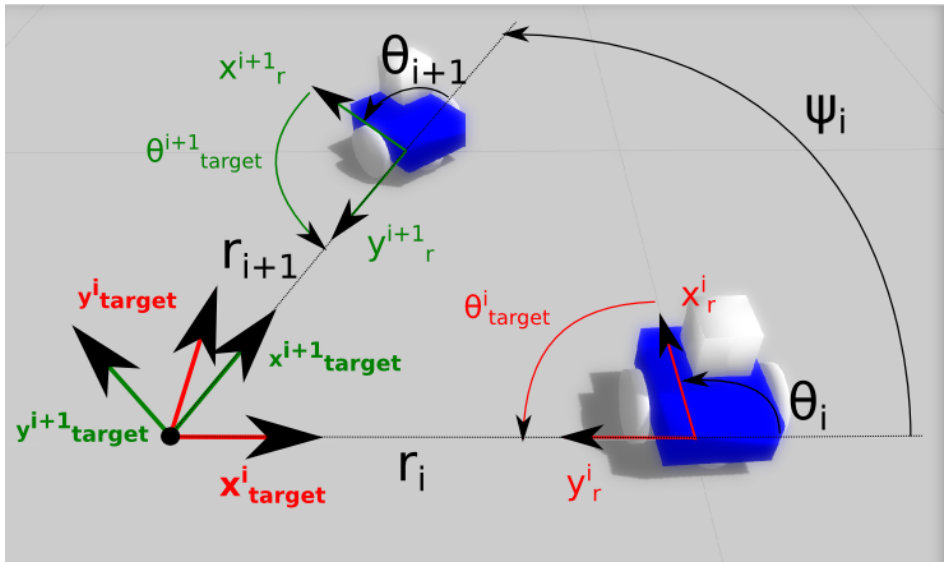


Figure – Robot-focused Local Reference Frames $\mathcal{R}_r^i$ and $\mathcal{R}_r^{i+1}$, and Robot-focused Target-centered Local Reference Frames $\mathcal{R}_{target}^i$ and $\mathcal{R}_{target}^{i+1}$.

From the i-th Robot-centered Local Reference Frame to the i-th Target-centered Local Reference Frame

Transformation of a coordinate p from the Robot-focused Robot-centered Local Reference Frame $\mathcal{R}_r^i$ to the Robot-focused Target-centered Local Reference Frame $\mathcal{R}_{target}^i$, where i is the index of the robot within the swarm :

$$p_{\mathcal{R}_{target}^i} = \begin{bmatrix} R(\theta_{target}^i - \pi) & \begin{matrix} -r_i \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{bmatrix} p_{\mathcal{R}_r^i}$$

$$R : \phi \mapsto \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

Validation of the Distributed Architecture

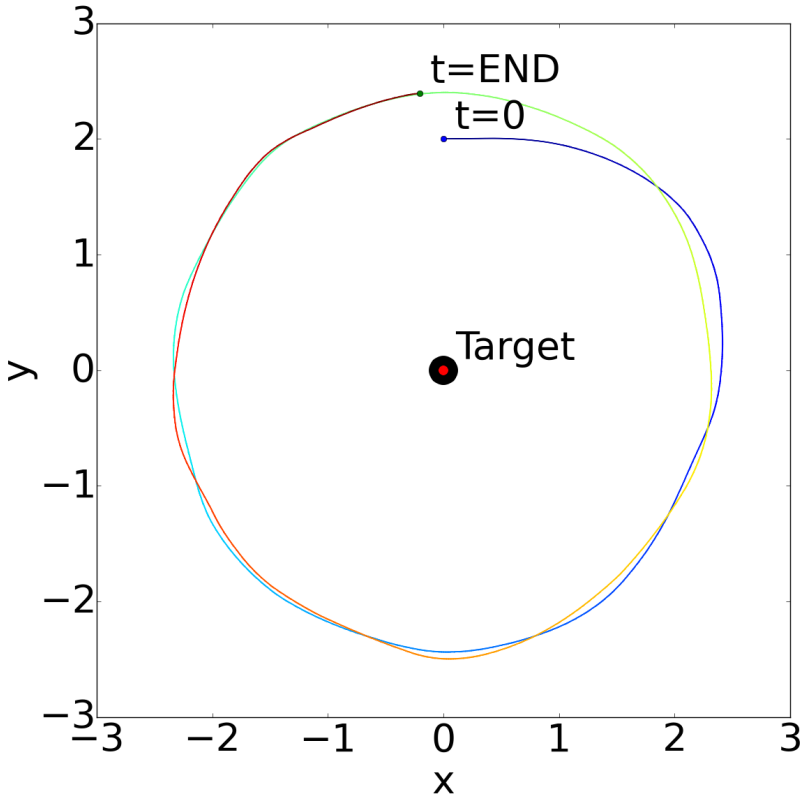


Figure – Trajectory of one robot encircling the target with the distributed pipeline.

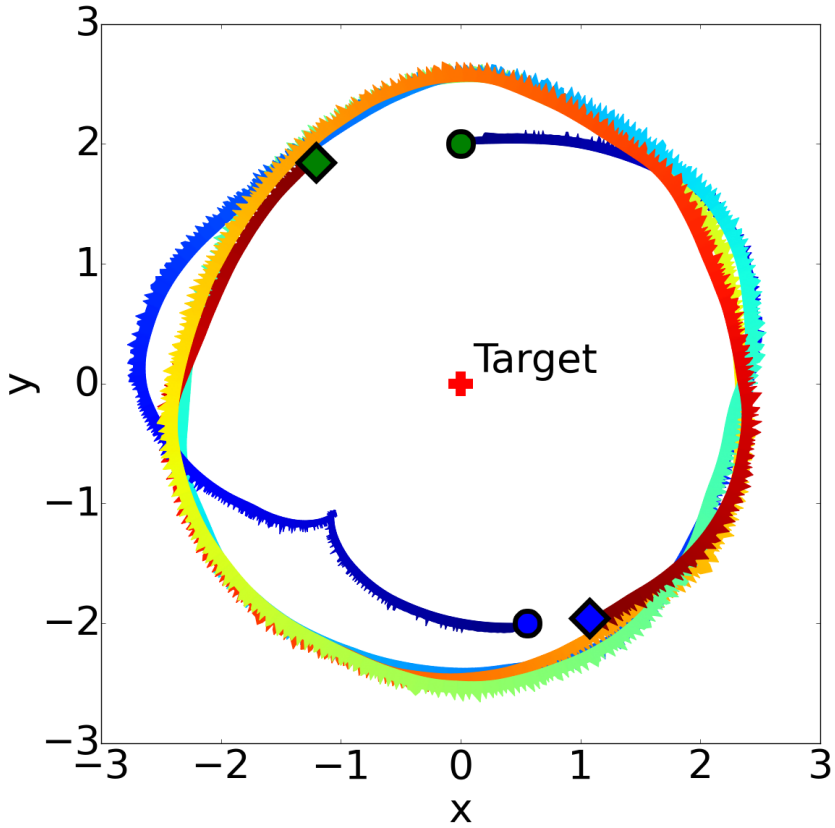
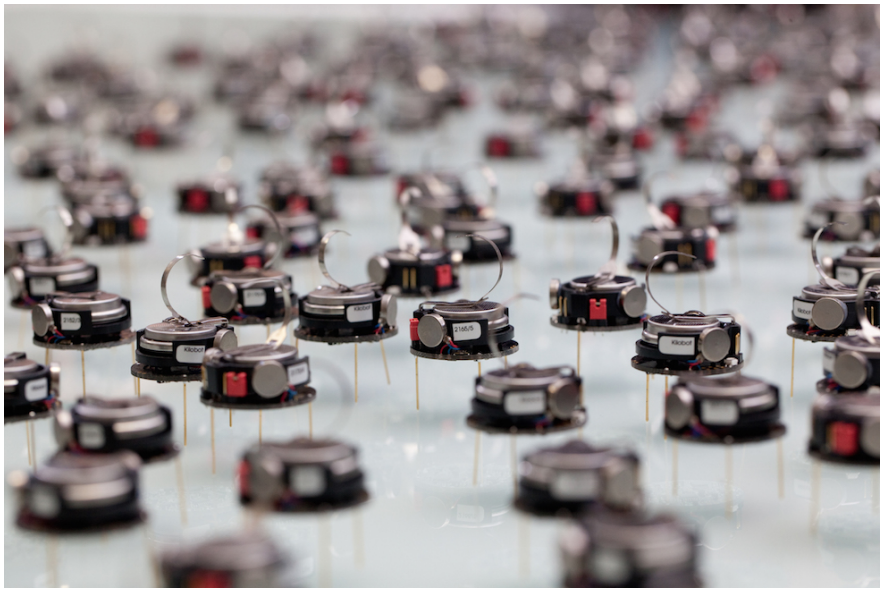


Figure – Trajectory of two robots encircling the target with the distributed pipeline.

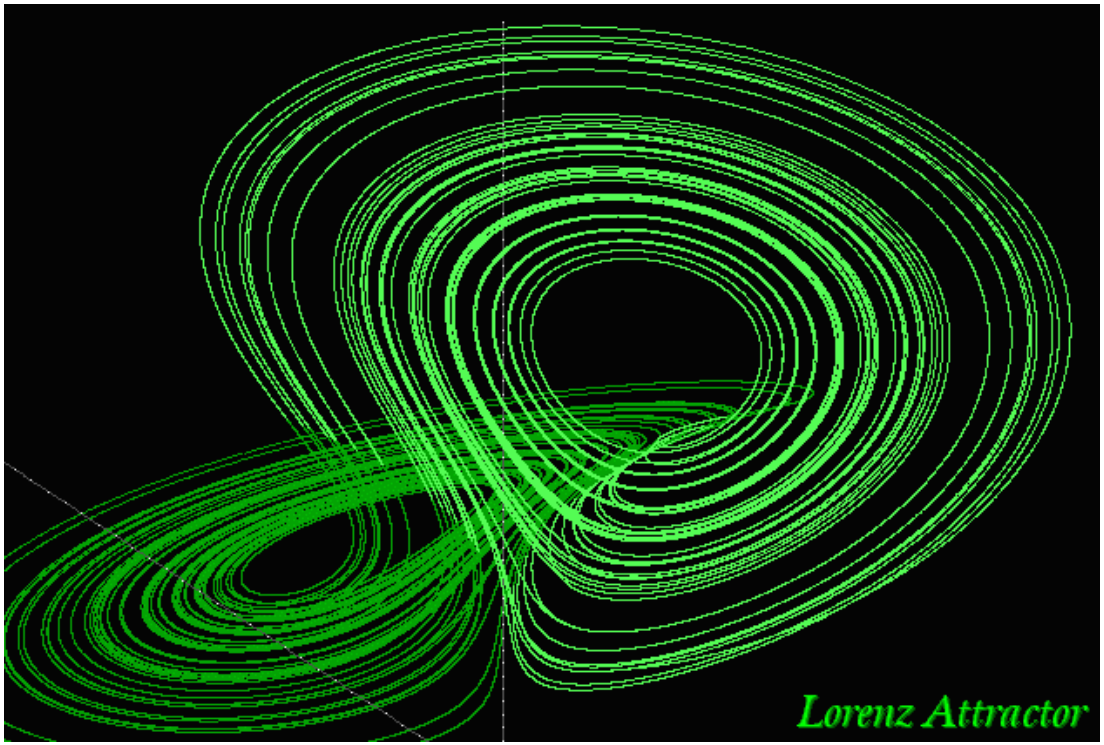
Swarm Robotics

- Inspired by nature (bees, ants...)
- Efficient/parallel workload handling
- Efficient contingency alleviation process



Nonlinear Systems

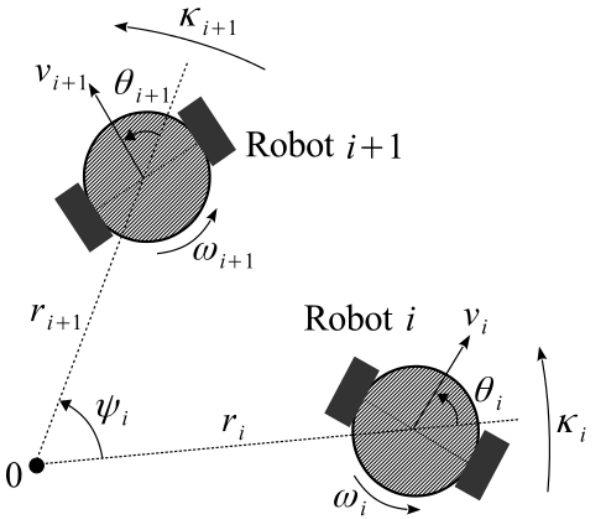
- Complex systems that enables a miscellany of concise behaviours
- Academic interest of its application to robotics.



Formulation

Apply Nonlinear Systems, more specifically Coupled-Oscillators, to the control of a Swarm of Two-Wheeled Robots.

Phase Coupled Oscillators applied to Control a Group of Robots and Stability Analysis (T.Nakamura et al., IFAC 2016)



$$f(r_i, \hat{r}_i) := a r_i \left(1 - \frac{r_i^2}{\hat{r}_i^2}\right)$$

$$g(\psi_i) := \Omega + \varepsilon \sin \psi_i.$$

Non-linear System :

$$\dot{r}_i = \bar{v}_i . \cos\theta_i$$

$$\dot{\theta}_i = \bar{\omega}_i$$

$$\dot{\psi}_i = \frac{\bar{v}_{i+1}}{r_{i+1}} . \cos\theta_{i+1} - \frac{\bar{v}_i}{r_i} . \cos\theta_i$$

Control Law :

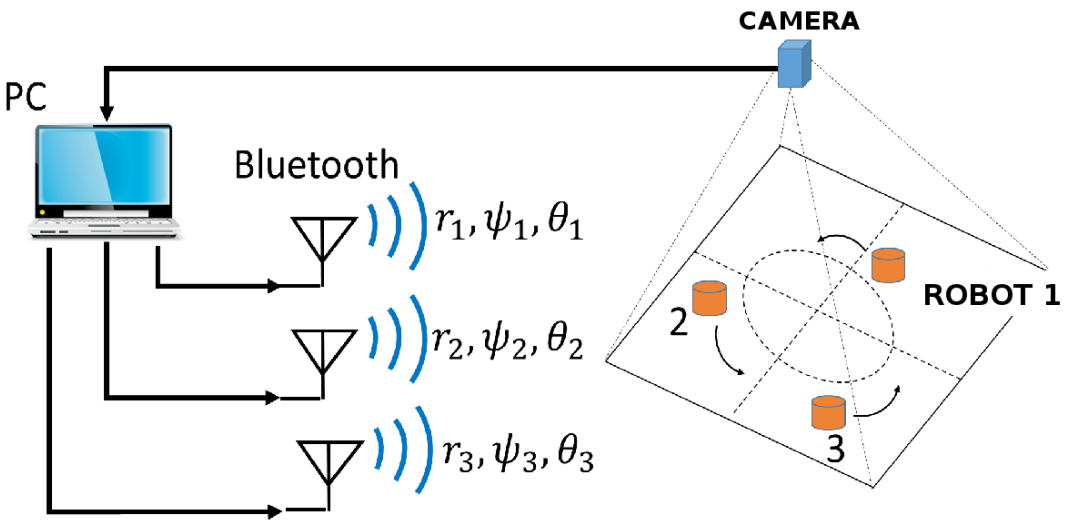
$$v_i = \overline{v}_i := k_v \left\{ f(r_i, \hat{r}_i) \cos\theta_i + r_i g(\psi_i) \sin\theta_i \right\}$$

$$\omega_i = \overline{\omega}_i := k_\omega \left\{ r_i g(\psi_i) \cos\theta_i - f(r_i, \hat{r}_i) \sin\theta_i \right\}$$

Centralized vs Distributed Architecture

Problem :

- Global sensor - not scalable
- Strong computational requirements - not lightweight

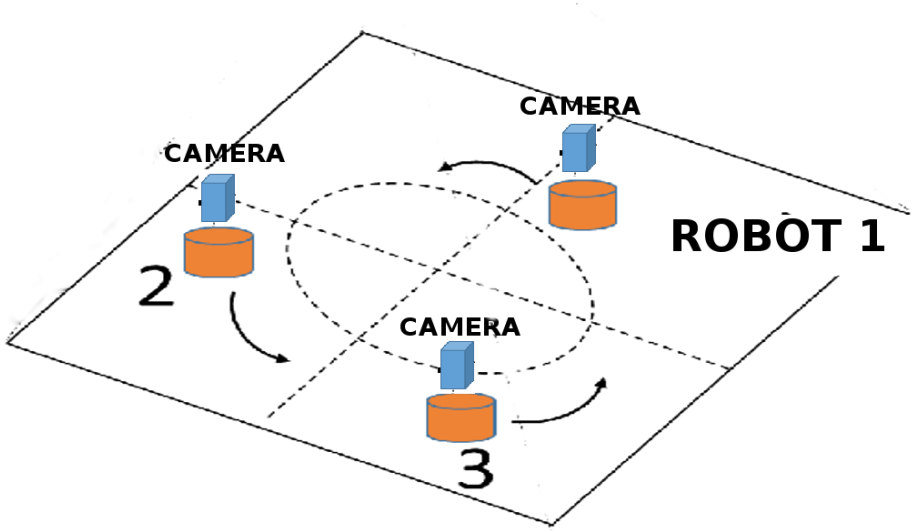


Centralized vs Distributed Architecture

Independently follow the paradigm :

- SENSE
- THINK
- ACT

in an efficient, lightweight and scalable way.



Expand the Possible Behaviours

Previously :

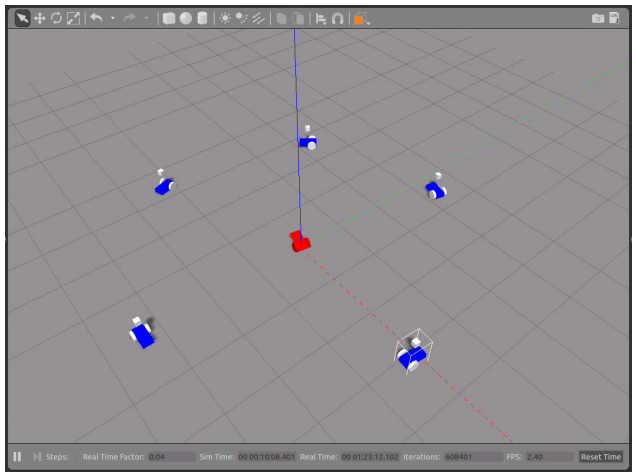
- Target Encircling

New behaviours in order to make that swarm more general-purpose :

- Target Following
- Obstacle Avoidance

Results, so far :

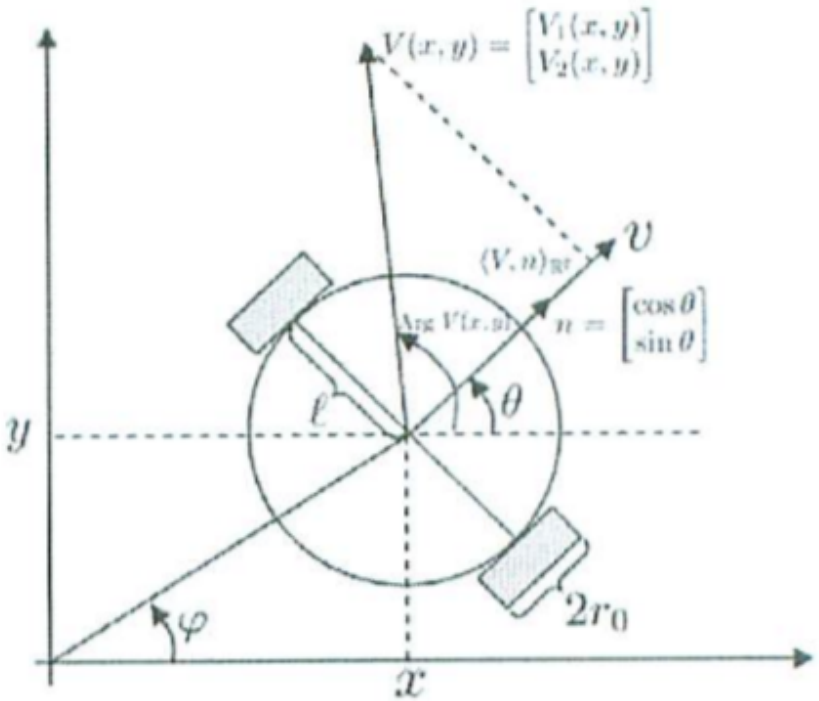
- Distribued Architecure : Computer Vision + Machine Learning : neighbour robots Recognition development.
- Distributed Architecture : proof-checked in simulation (using Gazebo and ROS)
- Multiple Behaviours : development of the control laws

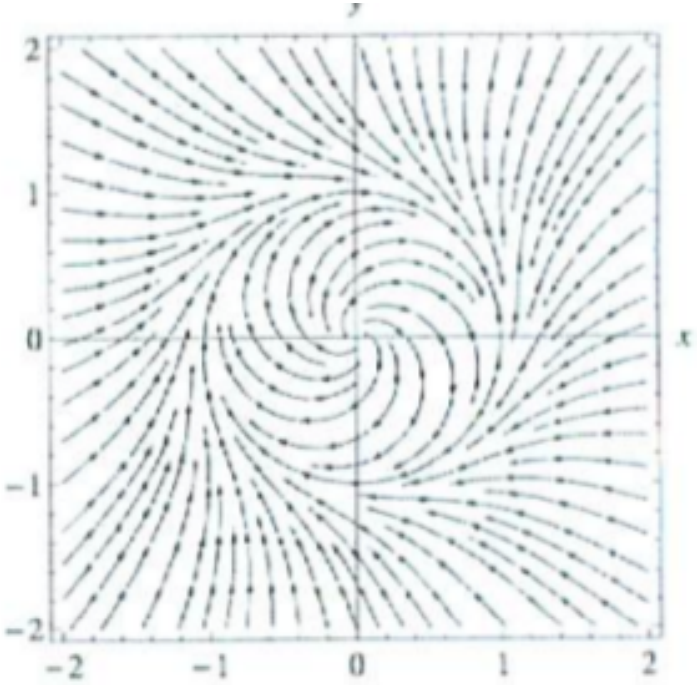


To Do :

- Distributed Architecture : implements the required hardware on the real swarm.
- Multiple Behaviours : proof-check in simulation and implementation on the real swarm of robots.

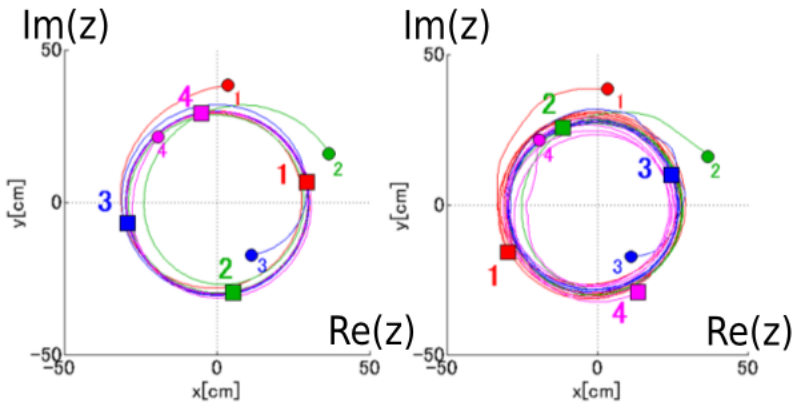
Trajectory Tracking of a single two-wheeled robot along a Circular Limit Cycle Vector Field (N. Hara et al., American Control Conference 2010)





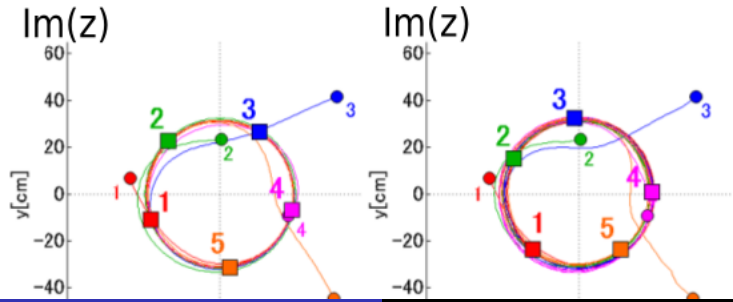
Control a Group of Robots (M.Tsukiji et al., SICE Annual Conference 2014)

$$\dot{z}_j = \left(a + i\omega - \left|z_j\right|^2\right) z_j + \varepsilon \sum_{m=1}^{d} \left(z_{j+m} - z_j + \delta(z_{j-m} - z_j)\right)$$

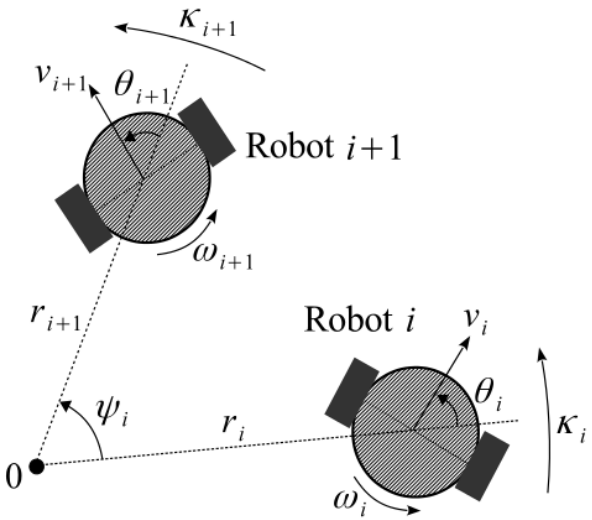


(a) Simulation result. (b) Experimental result.

Fig. 3 Trajectories of robot group ($N = 4, k = 1$).



Phase Coupled Oscillators applied to Control a Group of Robots and Stability Analysis (T.Nakamura et al., IFAC 2016)



$$f(r_i, \hat{r}_i) := a r_i \left(1 - \frac{r_i^2}{\hat{r}_i^2}\right)$$

$$g(\psi_i) := \Omega + \varepsilon \sin \psi_i.$$

Non-linear System :

$$\dot{r}_i = \bar{v}_i . \cos\theta_i$$

$$\dot{\theta}_i = \bar{\omega}_i$$

$$\dot{\psi}_i = \frac{\bar{v}_{i+1}}{r_{i+1}} . \cos\theta_{i+1} - \frac{\bar{v}_i}{r_i} . \cos\theta_i$$

Control Law :

$$v_i = \overline{v}_i := k_v \left\{ f(r_i, \hat{r}_i) \cos\theta_i + r_i g(\psi_i) \sin\theta_i \right\}$$

$$\omega_i = \overline{\omega}_i := k_\omega \left\{ r_i g(\psi_i) \cos\theta_i - f(r_i, \hat{r}_i) \sin\theta_i \right\}$$

Phase Coupled Oscillators applied to Control a Group of Robots and Stability Analysis (T.Nakamura et al., IFAC 2016)

Stability Analysis of the Swarm of Robots :

	$\frac{l}{N} \in \left[0, \frac{1}{4}\right)$	$\frac{l}{N} \in \left(\frac{1}{4}, \frac{1}{2}\right]$	$\frac{l}{N} \in \left[\frac{1}{2}, \frac{3}{4}\right)$	$\frac{l}{N} \in \left(\frac{3}{4}, 1\right]$
$\Omega \geq 0$	$k_\omega > 0$			
	$\varepsilon > 0$		$\varepsilon < 0$	
	$k_v > 0$ $a > 0$	$k_v < 0$ $a < 0$	$k_v > 0$ $a > 0$	$k_v < 0$ $a < 0$
$\Omega \leq 0$	$k_\omega < 0$			
	$\varepsilon < 0$		$\varepsilon > 0$	
	$k_v < 0$ $a < 0$	$k_v > 0$ $a > 0$	$k_v < 0$ $a < 0$	$k_v > 0$ $a > 0$

DATMO Problem

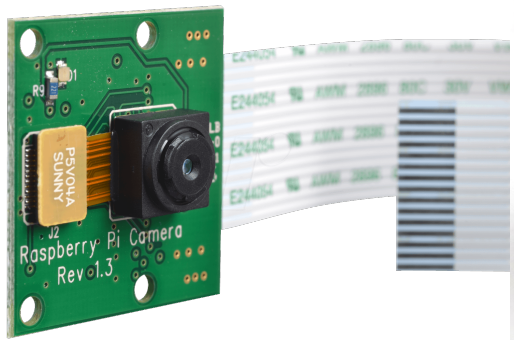
Detection **A**nd **T**racking of **M**oving **O**bjects

Decentralized Sensing + Homogeneous Robots :
Each robots within the swarm will have to detect and track its neighbours.

Approach : Computer Vision + Kalman Filter.

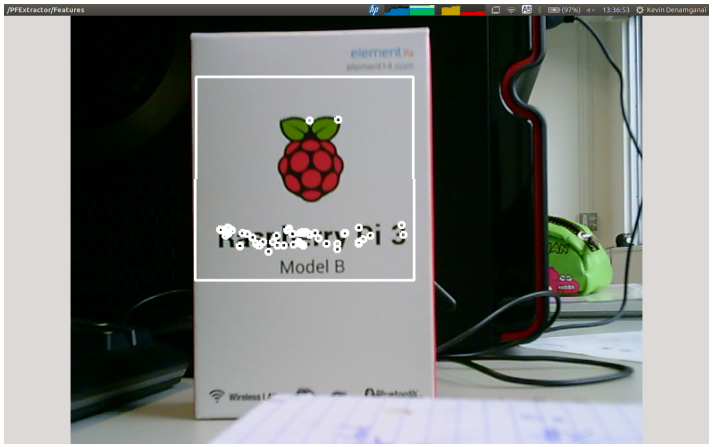
DATMO Problem - Hardware

Raspberry Pi 3 + Raspberry Pi Camera Module
+ very cheap and low quality Omnidirectional/Panoramic lens
Kogeto Dot :



Classical Computer Vision : Planar-based Solution

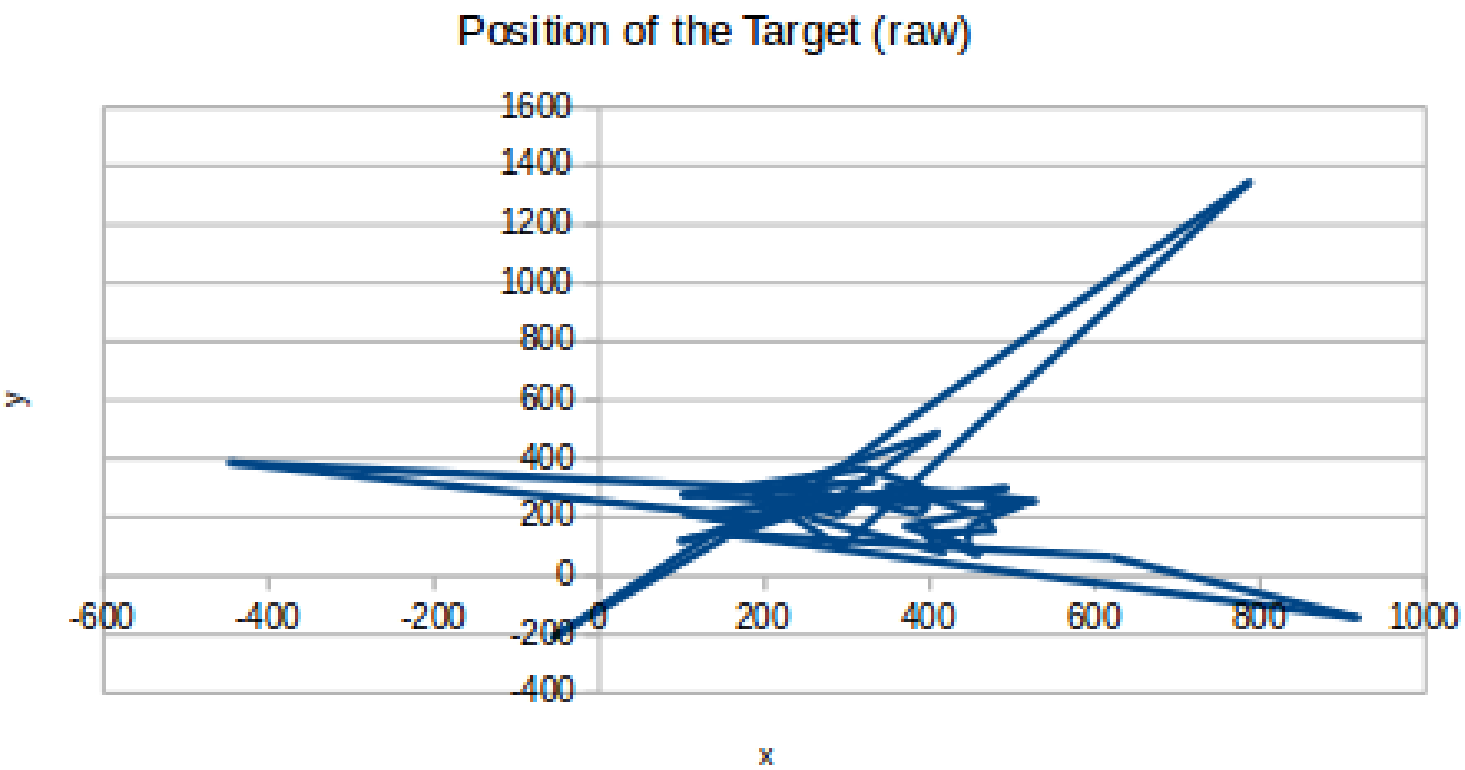
Goal : Recognition of feature points of given objects, e.g. neighbour robots, and the planar shape that they describe.



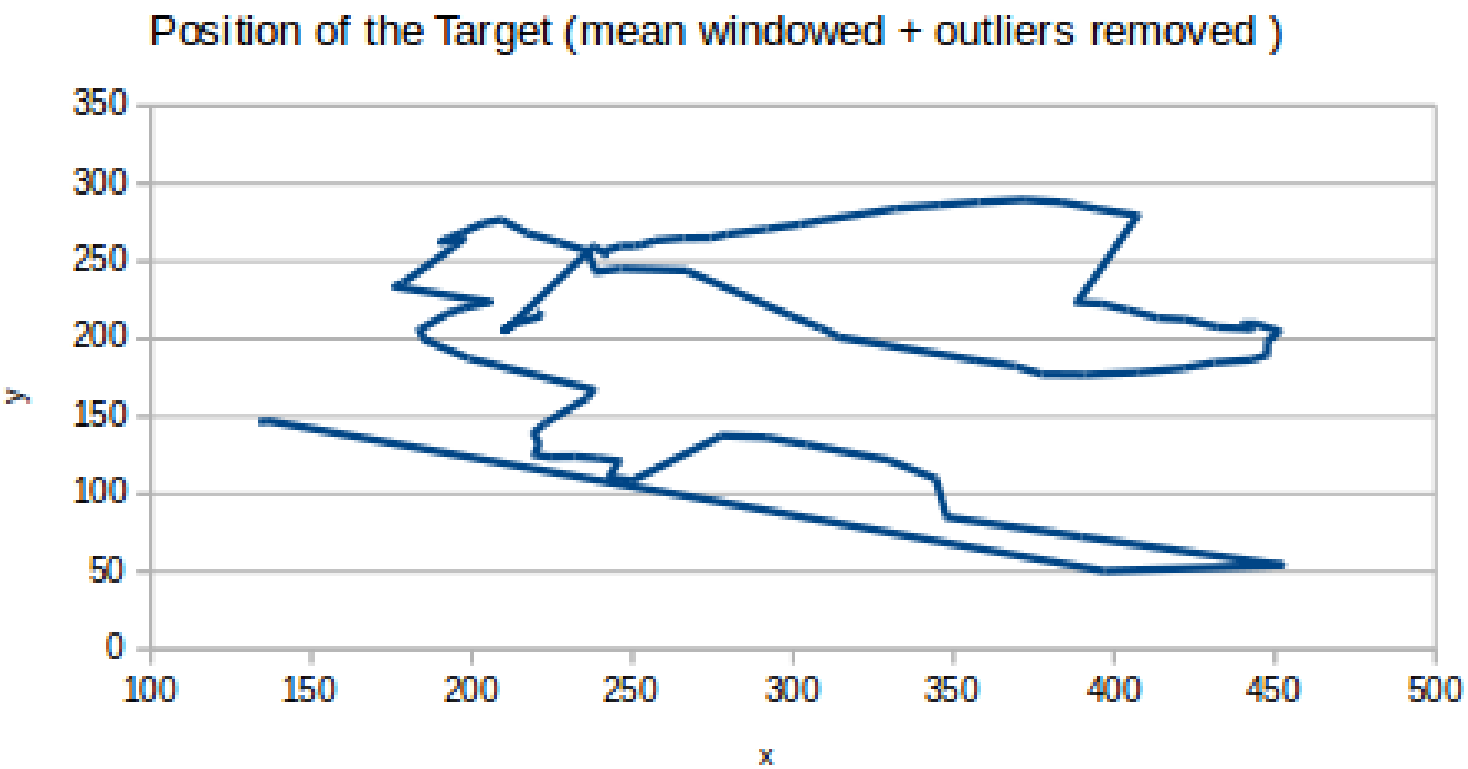
Issue(s) :

- Lightning-based changes of the features remains an unsolvable issue.
- Recognition rate too low due to the poor quality of the picture when used with the omnidirectional lens.
- Execution time too long when used with more powerfull feature point descriptors...

Planar-based Solution



Planar-based Solution



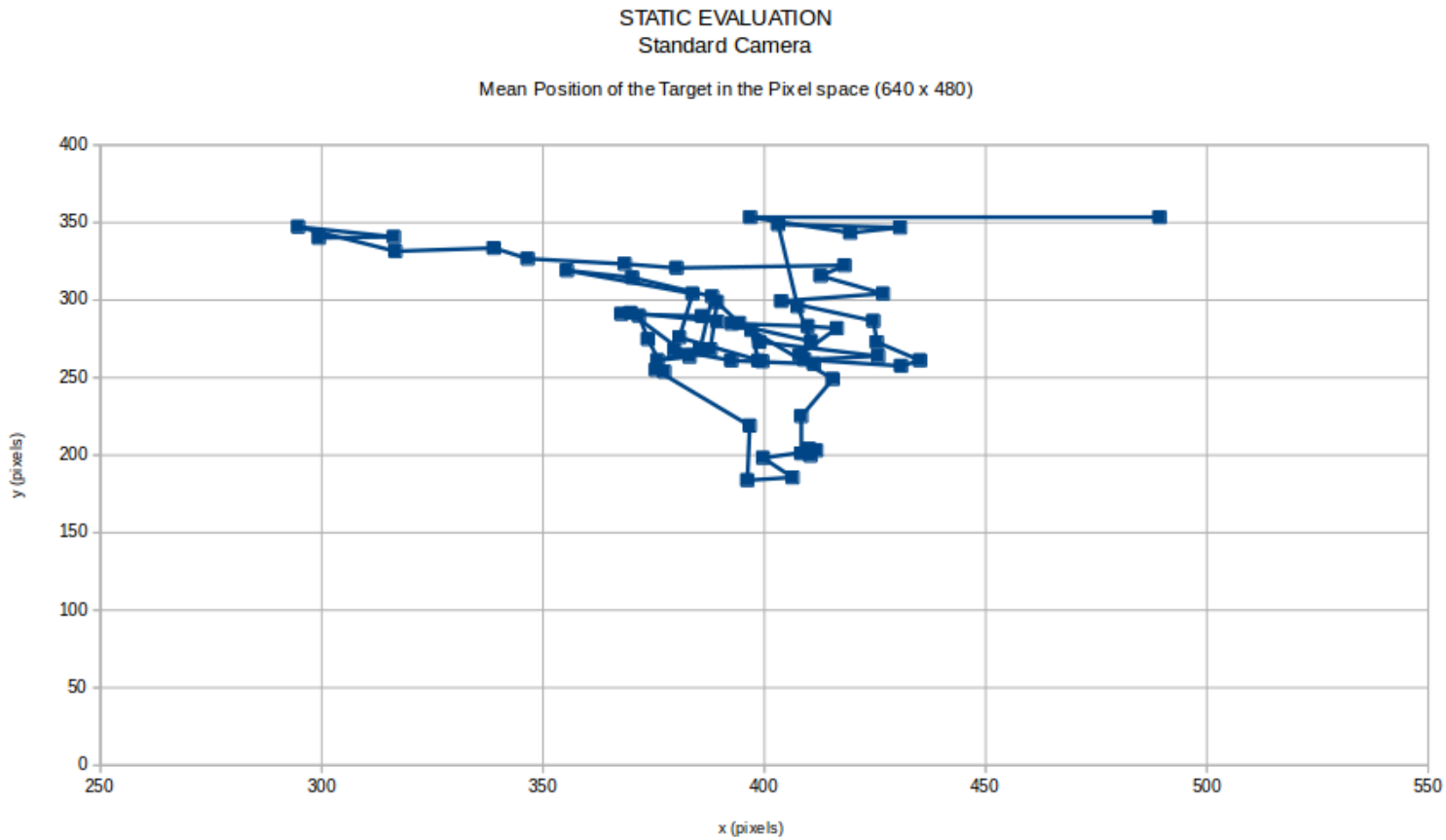
Planar-based Solution

To Do :

- quantitative test with the Omnidirectional/Panoramic Lens
- solve the poor resolution issue that might appear.

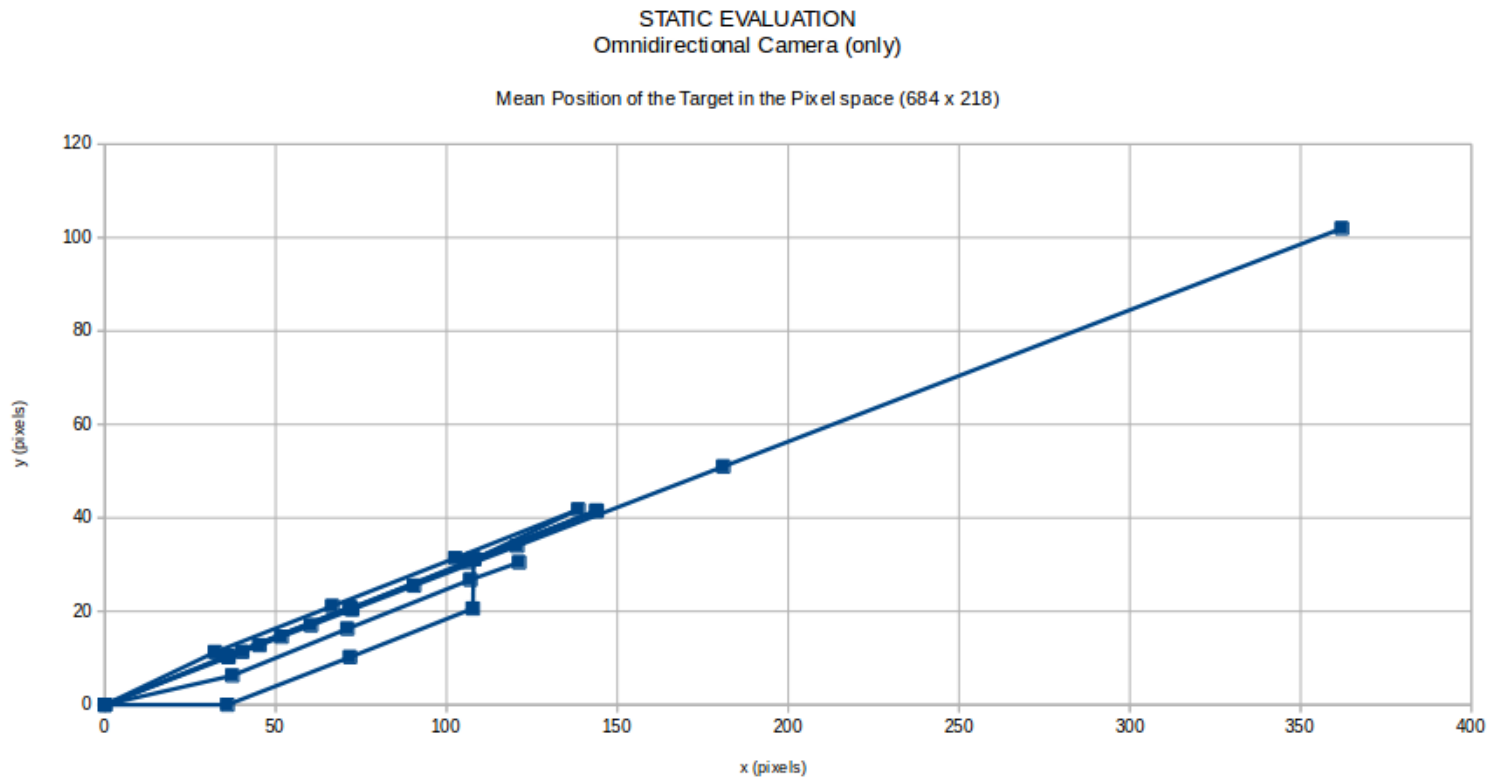
Quantitative Static Evaluation of the Planar-based Solution

Target-to-Camera distance : approx. 30 centimeters.
Maximal Standard Deviations in X and Y : 20.3% & 15.2%



Quantitative Static Evaluation of the Planar-based Solution

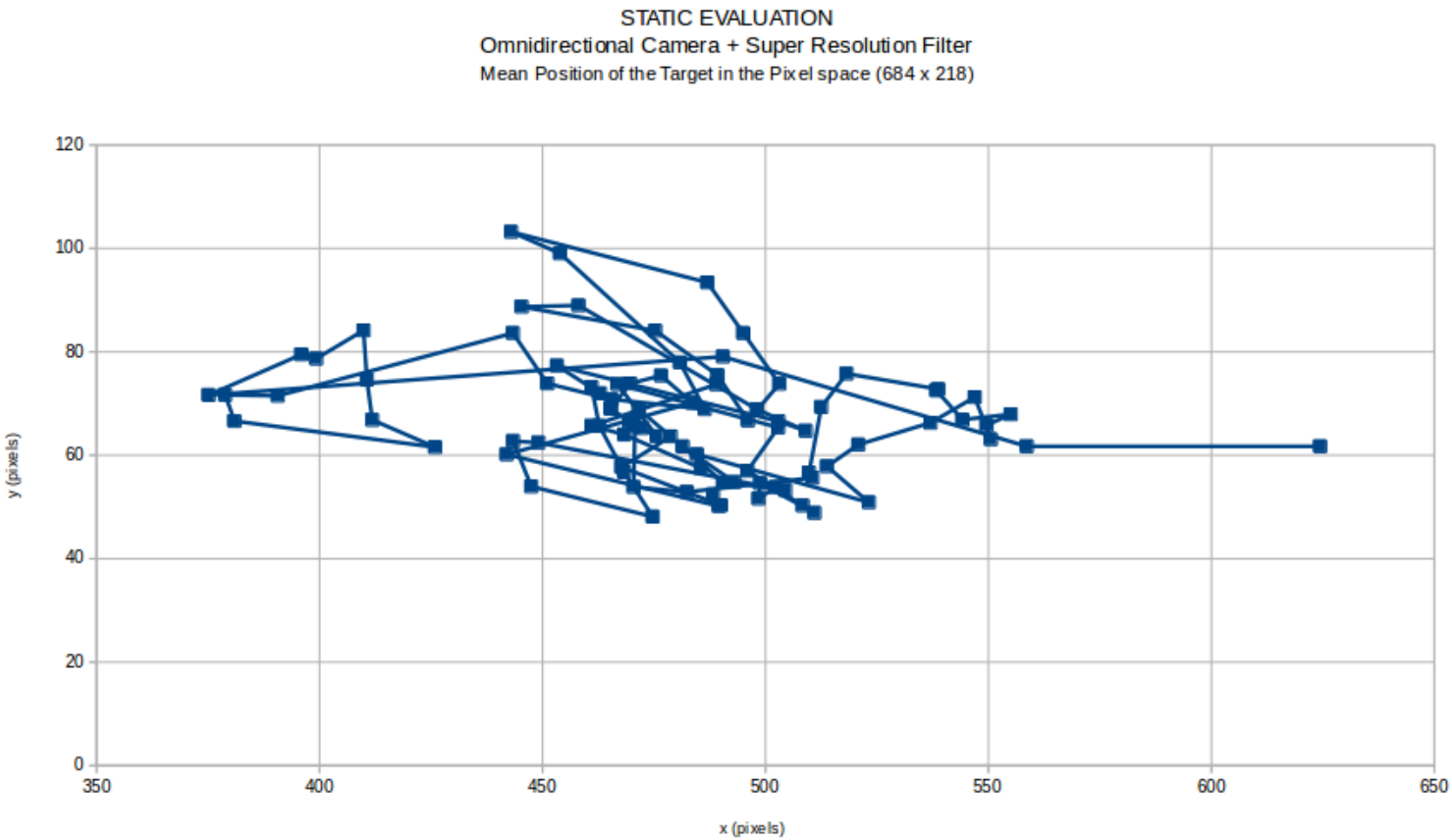
Target-to-Camera distance : approx. 30 centimeters.
Maximal Standard Deviations in X and Y : 44.1% & 39.4%



EXTREMELY INEFFICIENT...

Quantitative Static Evaluation of the Planar-based Solution

Target-to-Camera distance : approx. 30 centimeters.
Maximal Standard Deviations in X and Y : 21.0% & 16.5%



Almost the same as the Standard Camera, that is the best efficiency possible...

Quantitative Static Evaluation of the Planar-based Solution

Target-to-Camera distance : approx. 30 centimeters.
Maximal Standard Deviations in X and Y : 21.0% & 16.5%

Conclusion : roughly 18% of error on the recognition of the robots for every robot will make those measures not trustworthy enough to build a whole system over it.

Other solution : Neural Network-based solution
Advantages : **can cope with very poor resolution pictures.**

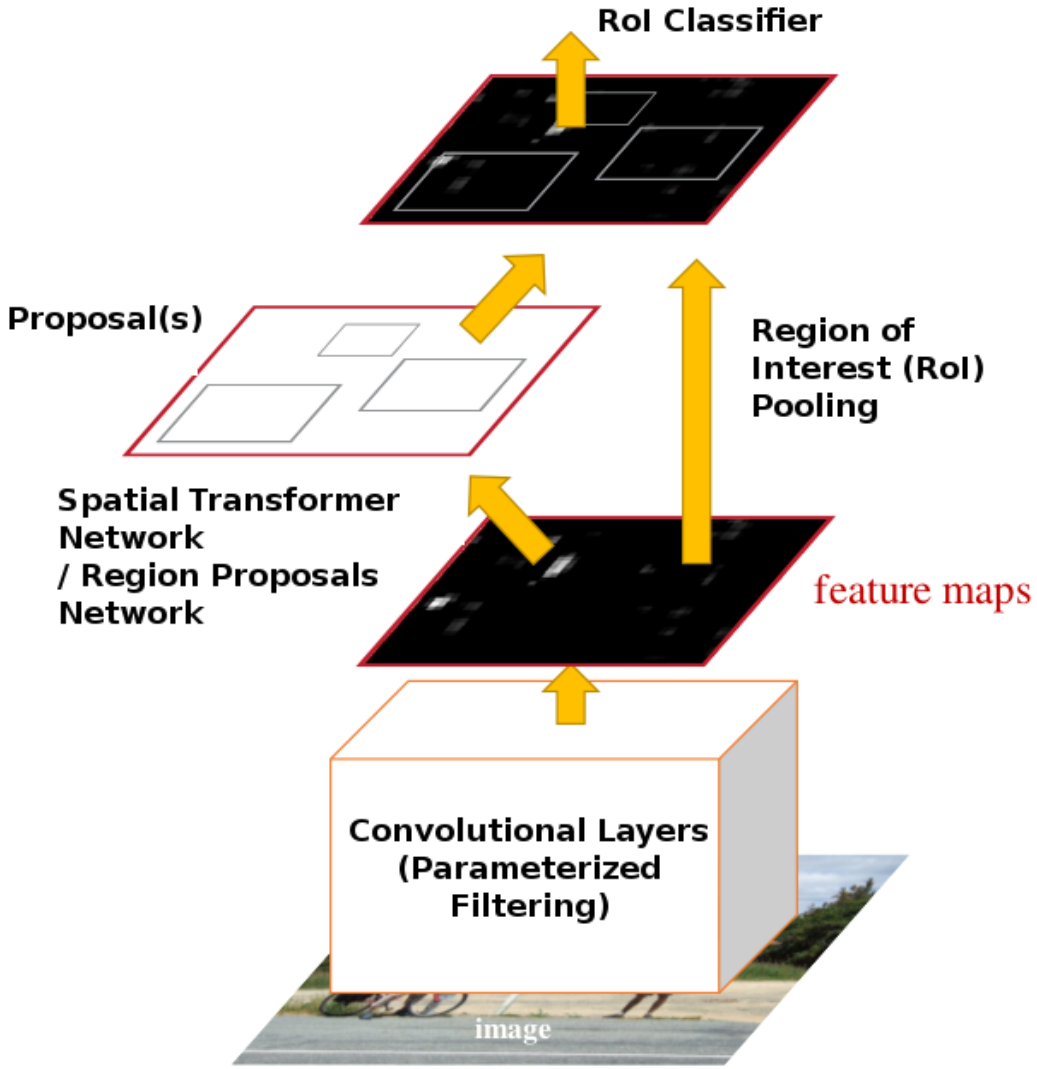
Neural Network-based Solution

Two tasks :

- Recognition/Classification Task ("Is it a robot or a chair ?")
- Bounding Box Regression Task ("Where is it located in the picture ?")

Two main architectures :

- Faster R-CNN
- Spatial Transformer Network.



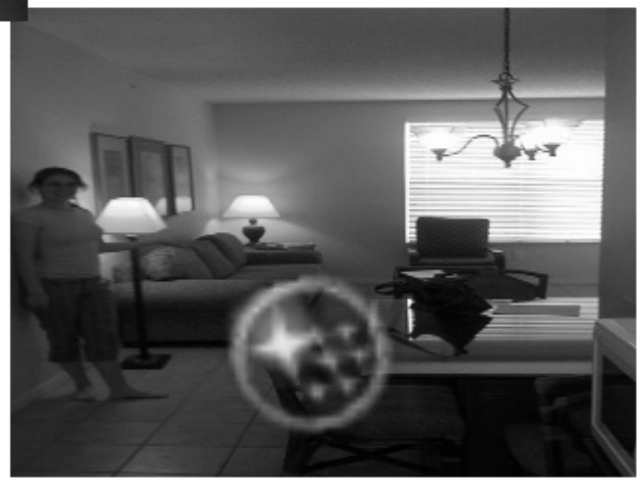
Neural Network-based Solution

Supervised Learning :
NEED A DATASET
(images and
labels/bounding box of the
object)

Creation of an artificial
dataset :
Hyundai VS Subaru +
Bounding Box Regression
Artificial Dataset

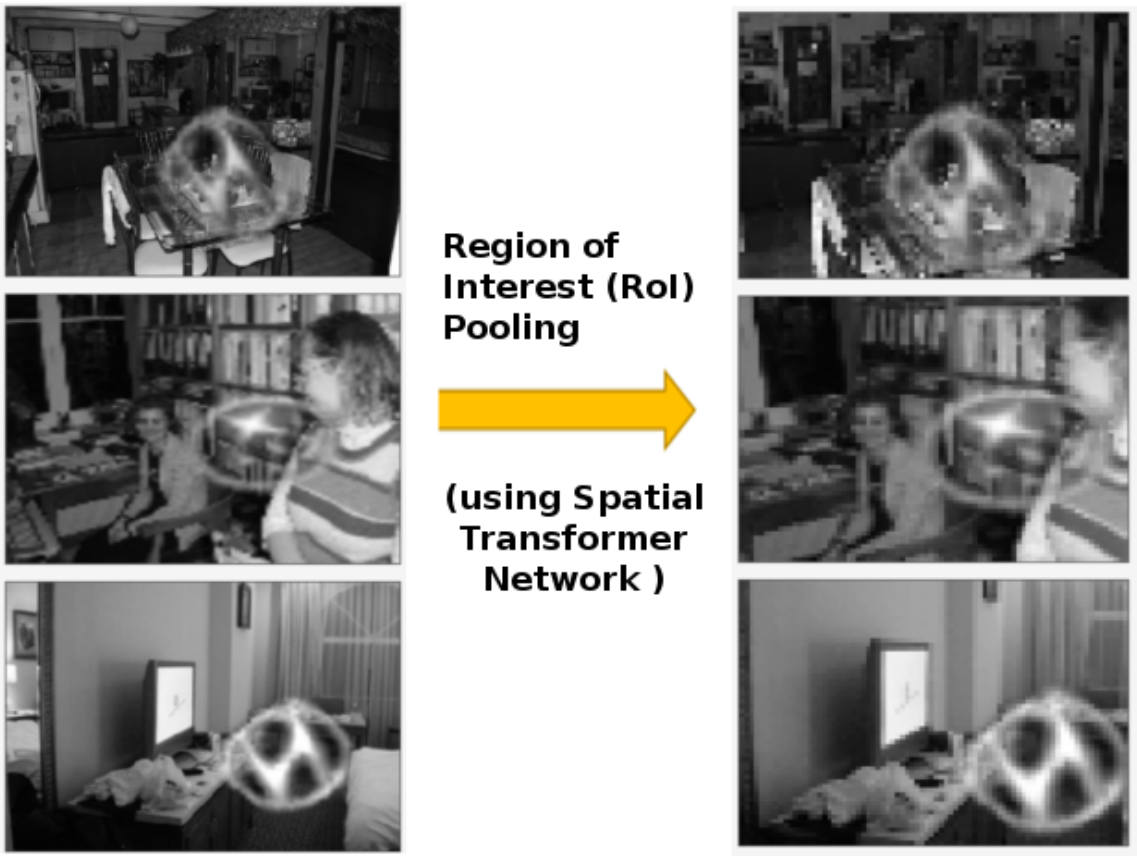
Hyundai

Subaru



Spatial Transformer Network-based Solution

Example of Region of Interest (extracted during learning phase) :
Bounding Box Regression Task



Training Spatial Transformer Network - Recognition Task

First try : OVERFITTING

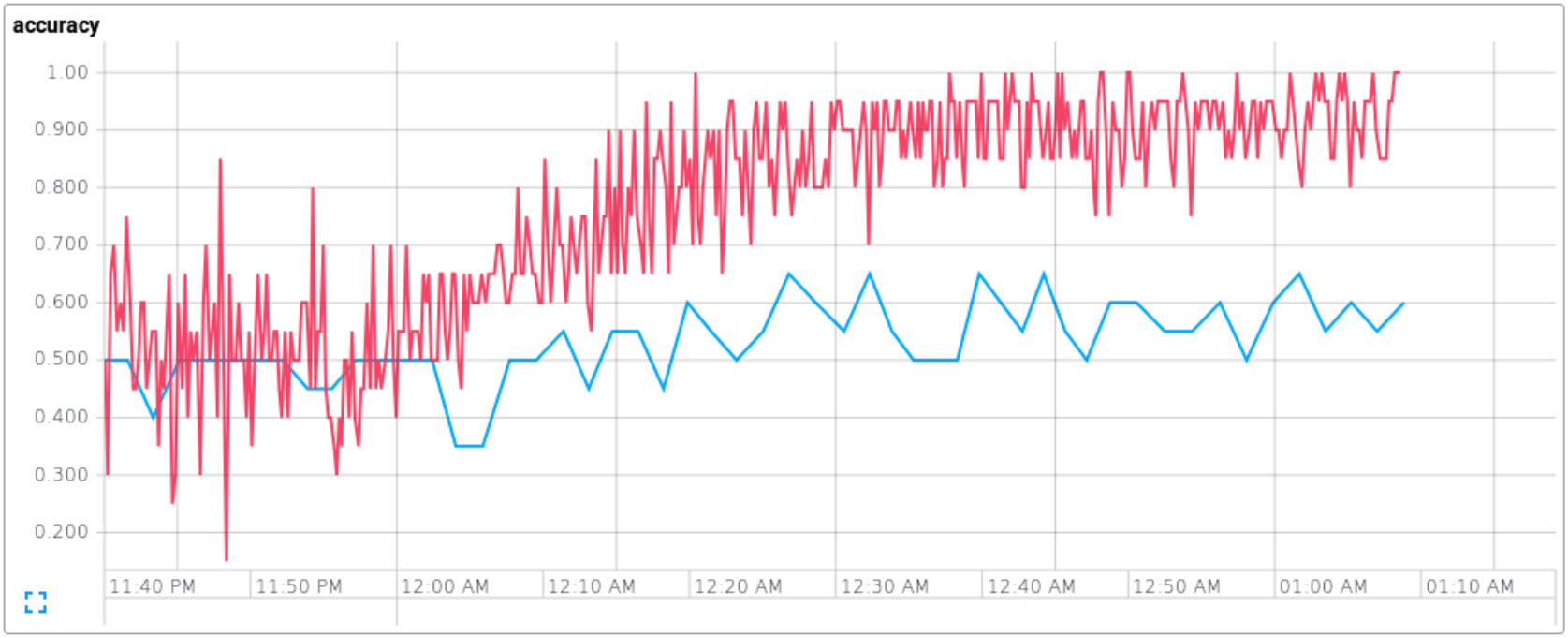


Figure – Test & train accuracies of a {convx2+STN+convx1+1024xFC} neural network during training on the 200 first pictures of the dataset SubaruVSHyundai.

Training Spatial Transformer Network - Recognition Task

Finally, after tweaking with the architecture parameters :
90 ± 10% of accuracy on test dataset

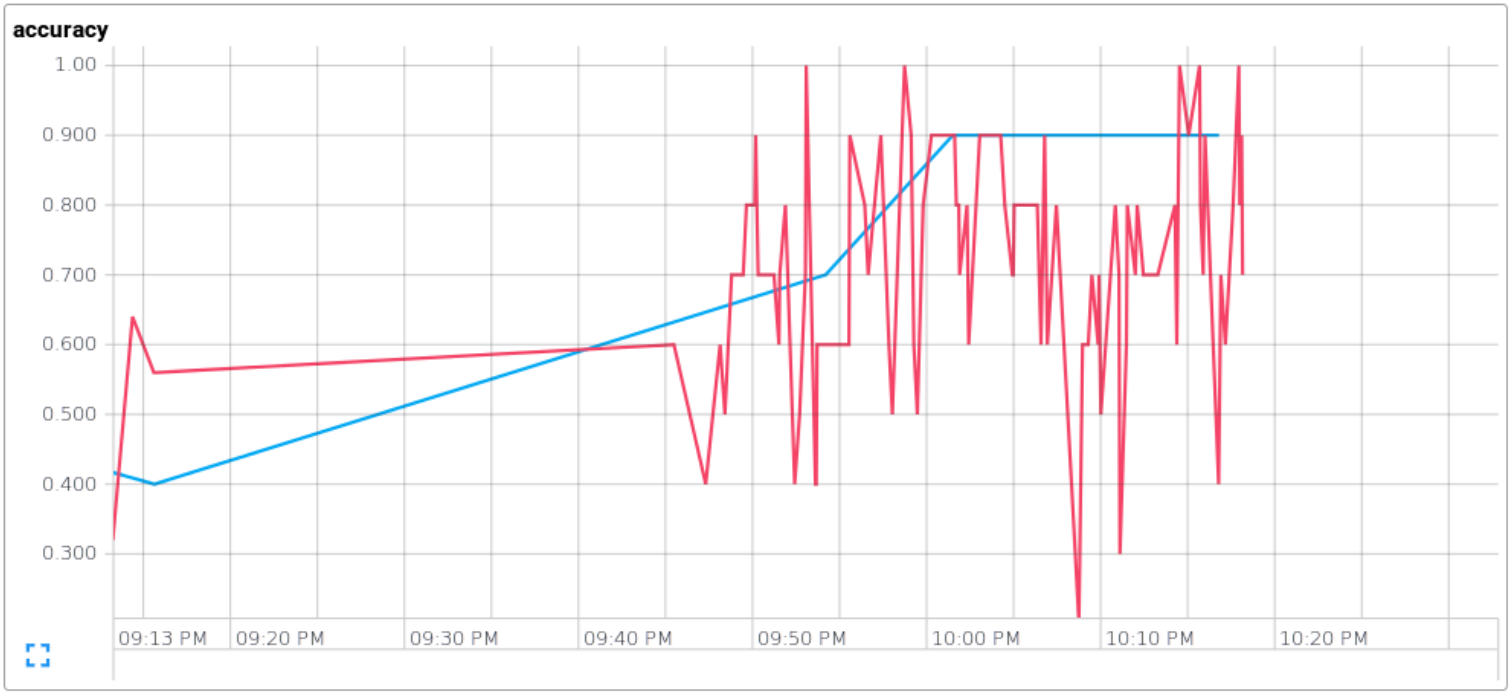


Figure – Test & train accuracies of a {reluconvx2-16-32+STN+relumaxpoolconvx1-64+128xFC} neural network.

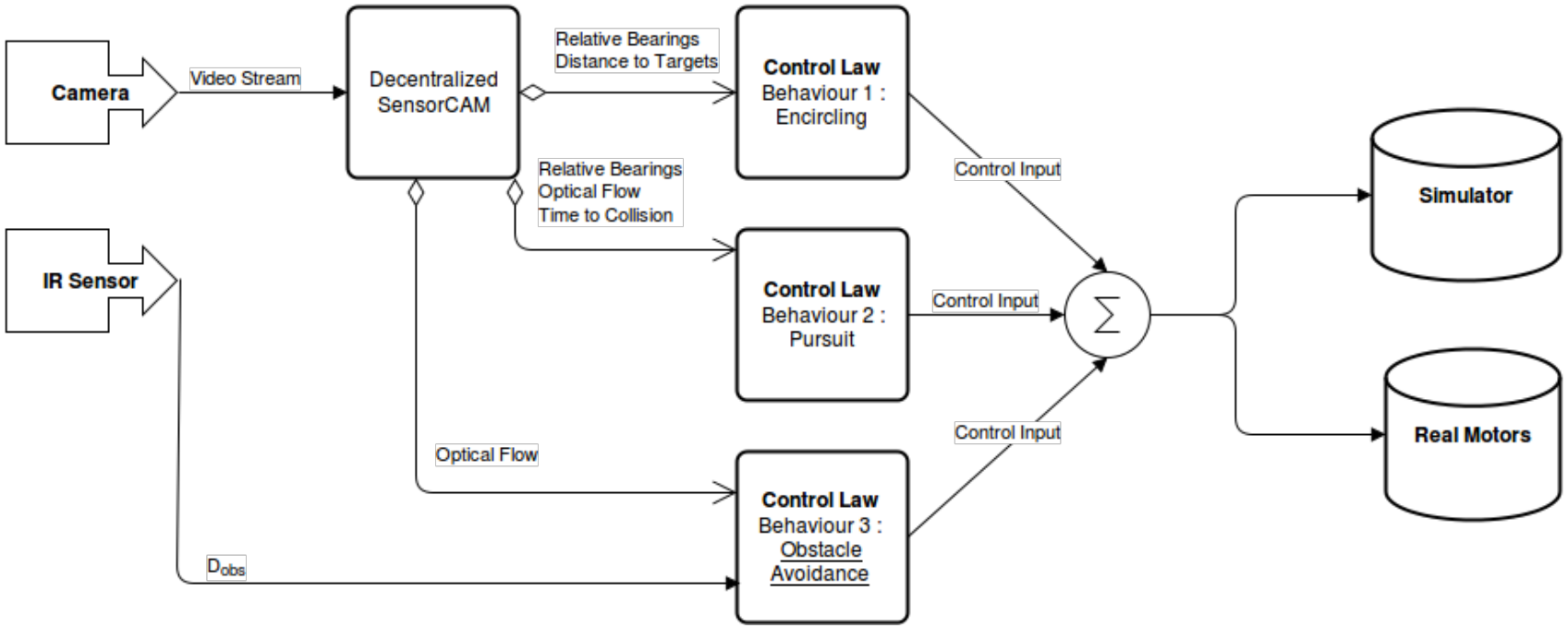
Conclusion :

$90 \pm 10\%$ of accuracy on the test dataset for the Recognition Task. It should be efficient enough to go on with it.

To Do :

- quantitative test with regards to the Bounding Box Regression Task.
- quantitative test with the Omnidirectional/Panoramic Lens.

Block Diagram for a Robot within the Swarm



Simulated Environment

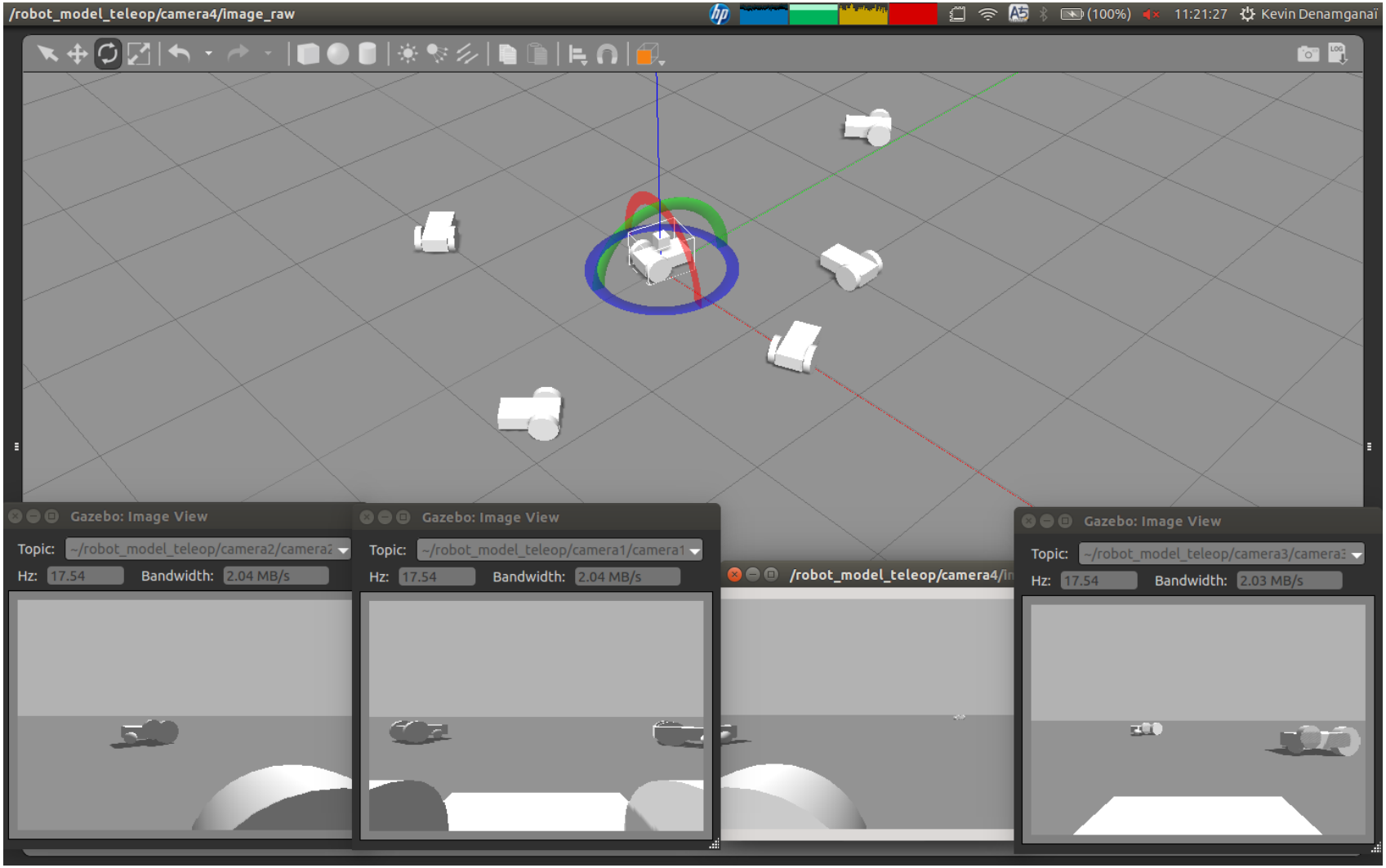
Description of the simulated Swarm :

- 5 to 15 robots.
- Omnidirectional vision for each robots.
- 2 or 3 proximity/distance sensors for each robots.

Description of the other simulated objects :

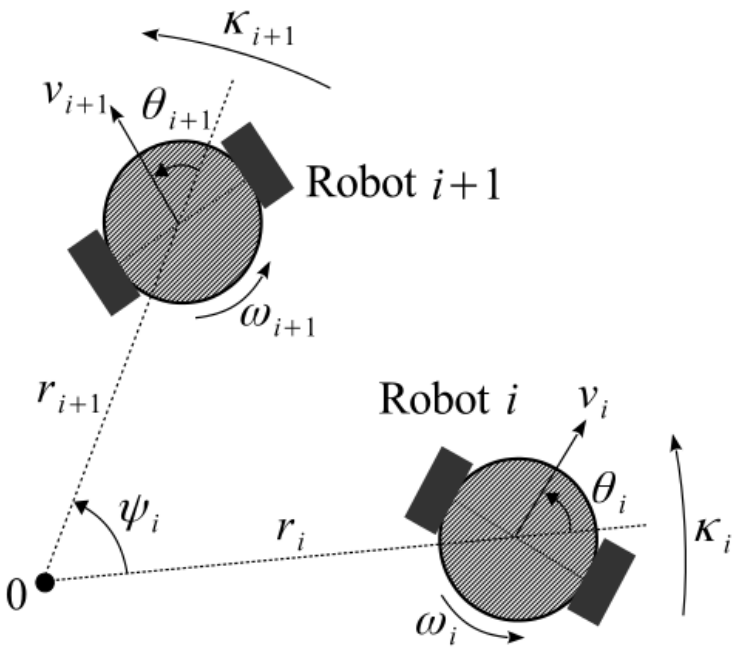
- Obstacles (walls, chairs, tables, moving objects...)
- Target that the swarm must follow.

Simulated Environment

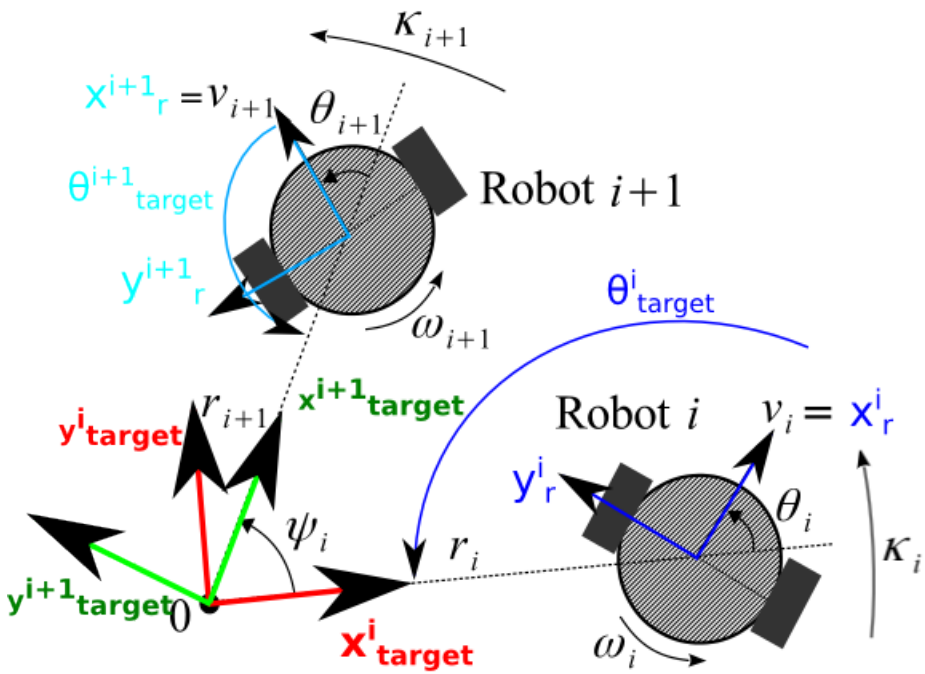


Dealing with the lack of an Universal Reference Frame

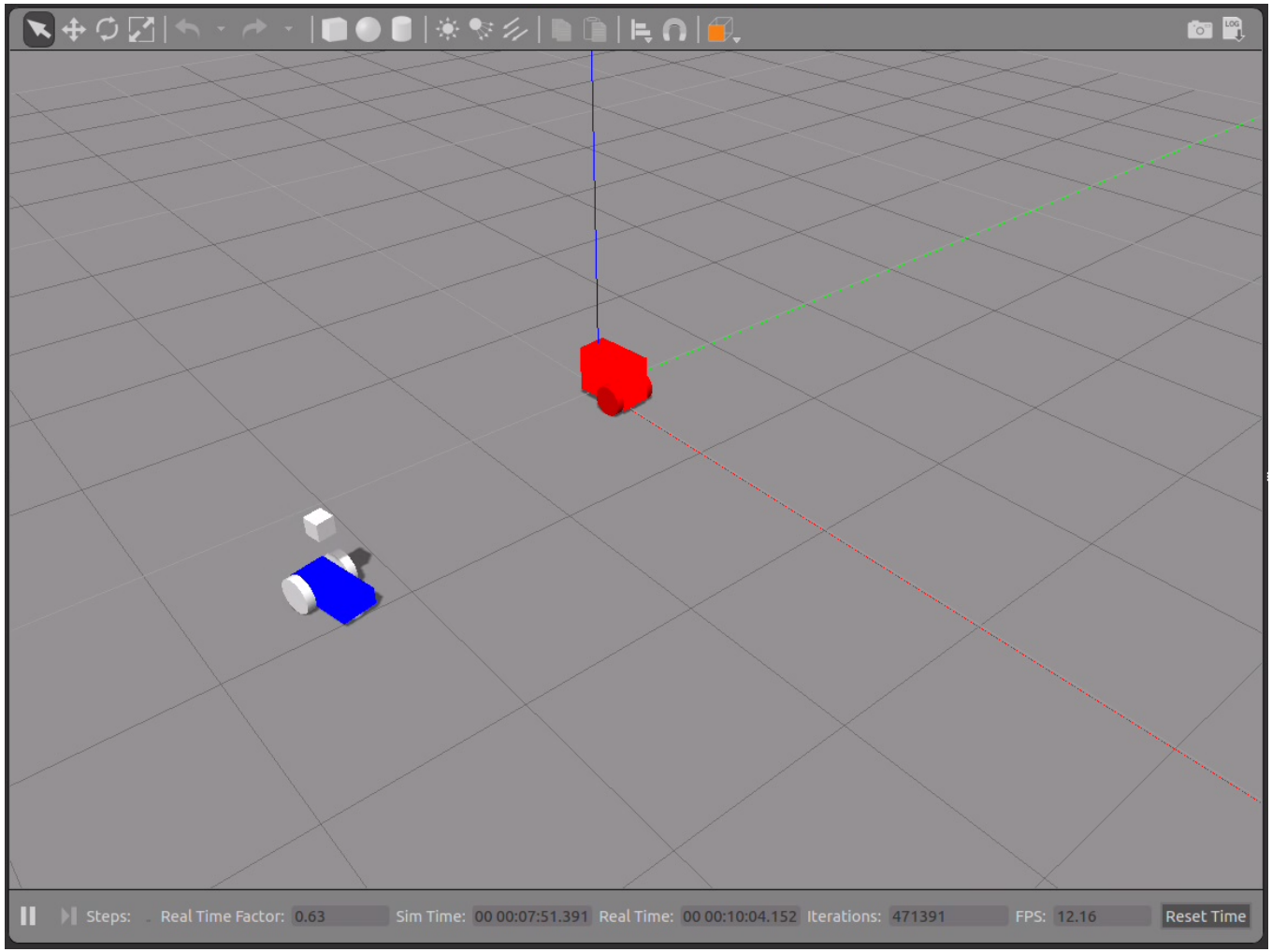
Universal Target-centered Reference Frame :



Multiple Local Robot-related Reference Frames :



Control Law : Encircling Behaviour in a Decentralized Way



Control Law Simulation

Conclusion :
Following the implementation of the sensing pipeline, an efficient decentralization of the centralized encircling control law has been devised.

To Do :

- Implementation of the control law for the two others behaviours (target pursuit and obstacle avoidance).
- Create simulation scenarios with obstacles and a moving target to further validate the combined control laws.

Conclusion

ご清聴ありがとうございました。

Different Architectures - 諸建築について

Current Architecture and Desired Architecture :

Heterogeneous	vs	Homogeneous/Anonymous
No communication	vs	Communication
Centralized Sensors	vs	Decentralized Sensors
Decentralized Computation	vs	Centralized Computation
Leader-free	vs	Relying on a Leader/Target

DATMO Problem

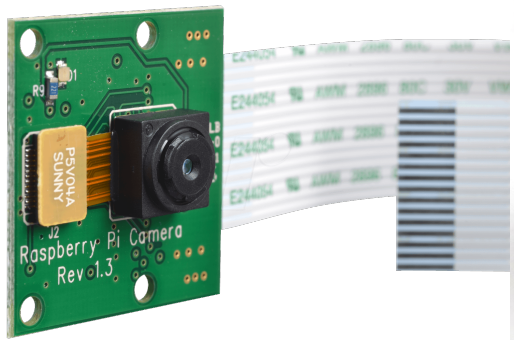
Detection **A**nd **T**racking of **M**oving **O**bjects

Decentralized Sensing + Homogeneous Robots :
Each robots within the swarm will have to detect and track its neighbours.

Approach : Computer Vision + Kalman Filter.

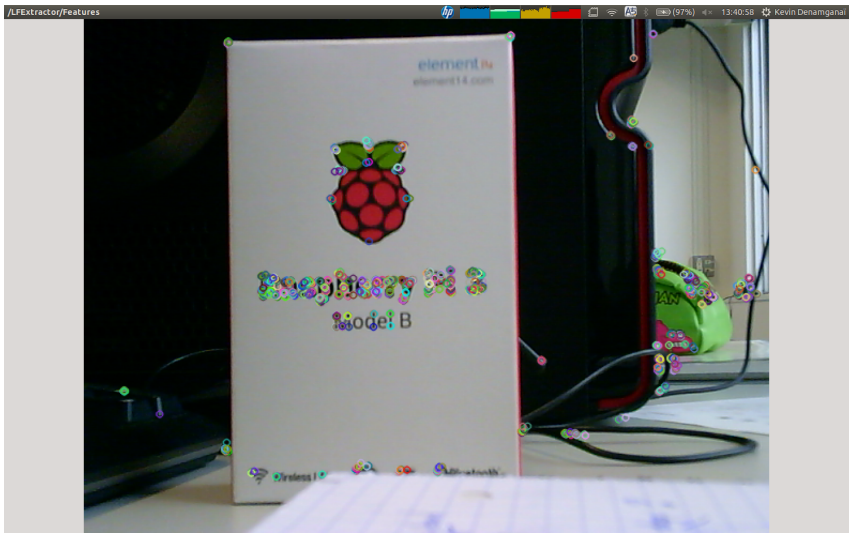
DATMO Problem - Hardware

Raspberry Pi 3 + Raspberry Pi Camera Module
+ very cheap and low quality Omnidirectional/Panoramic lens
Kogeto Dot :



Feature-based Solution

Goal : Recognition of feature points of given objects, e.g. neighbour robots.

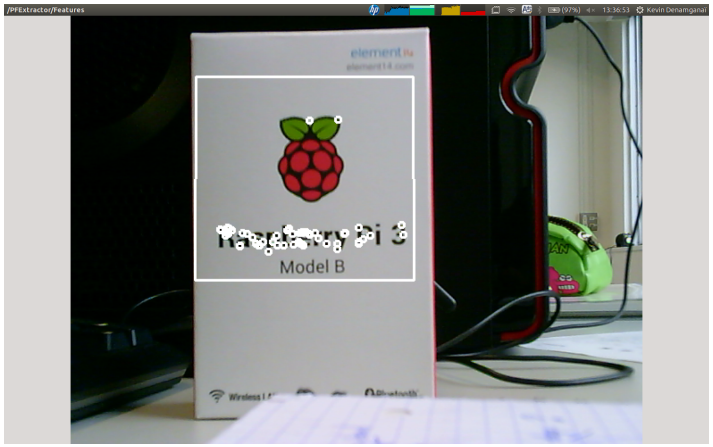


Issue(s) :

- Lightning-based changes of the features remains an unsolvable issue.
- Recognition rate too low due to the poor quality of the picture when used with the omnidirectional lens.

Planar-based Solution

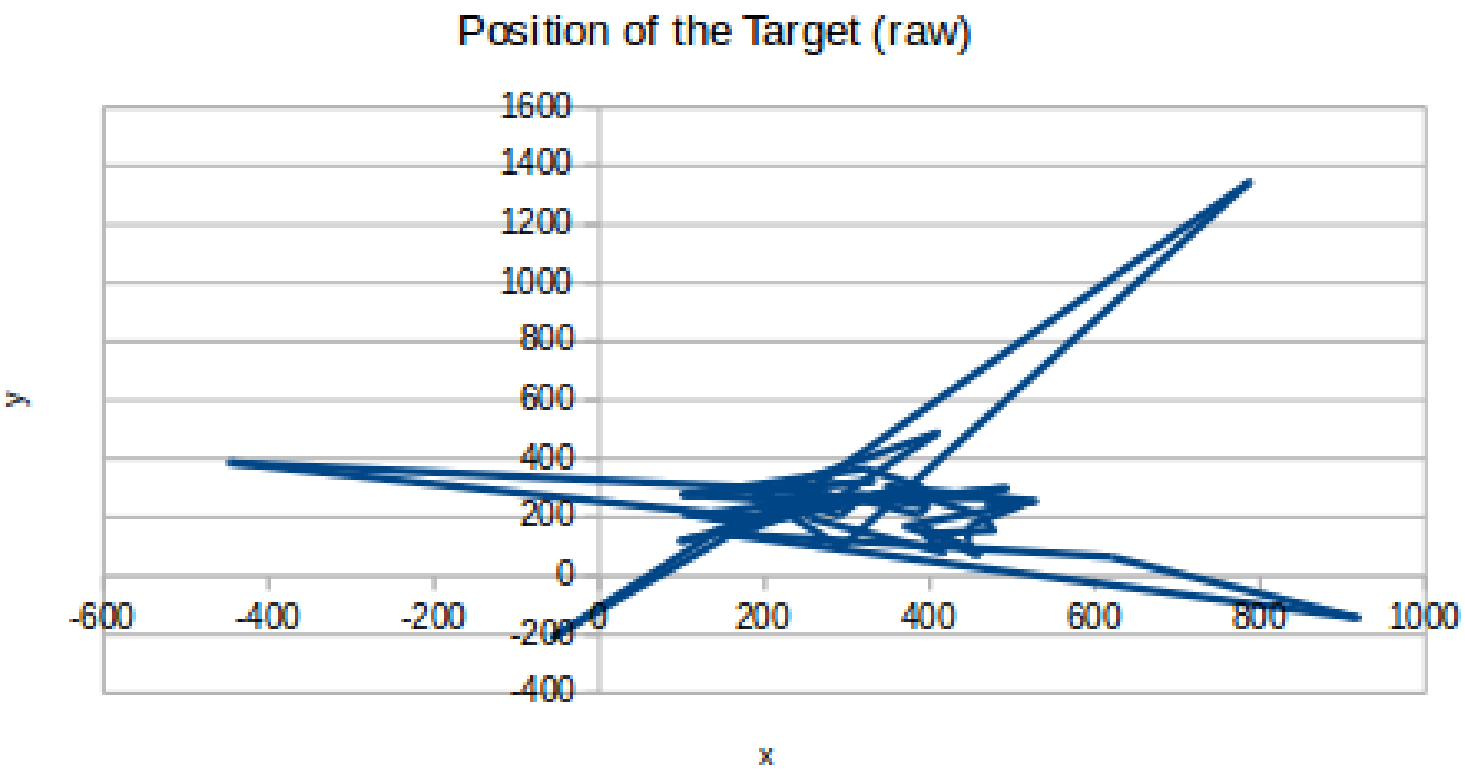
Goal : Recognition of feature points of given objects, e.g. neighbour robots, and the planar shape that they describe.



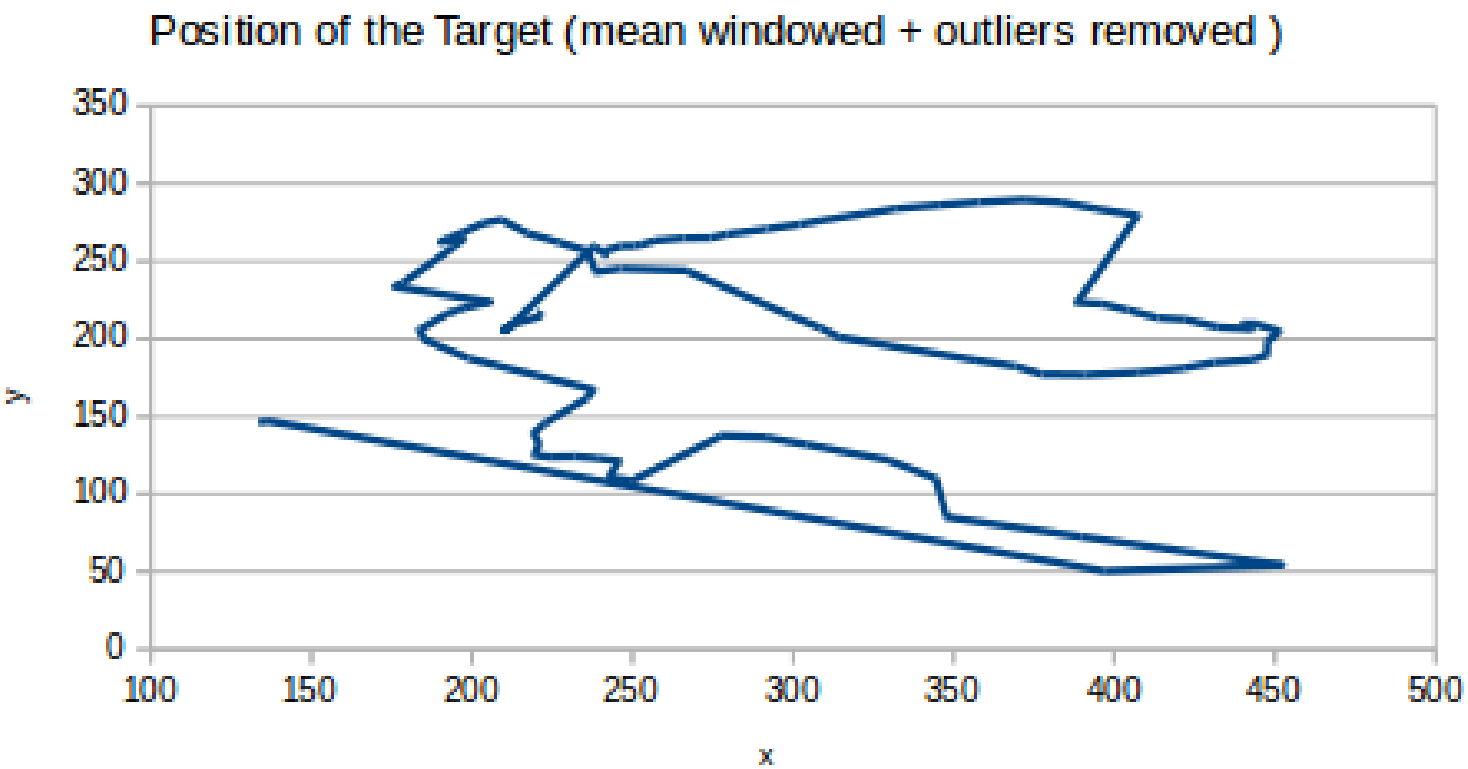
Issue(s) :

- Lightning-based changes of the features remains an unsolvable issue.
- Recognition rate too low due to the poor quality of the picture when used with the omnidirectional lens.
- Execution time too long when used with more powerfull feature point descriptors...

Planar-based Solution



Planar-based Solution



Planar-based Solution

To Do :

- quantitative test with the Omnidirectional/Panoramic Lens
- solve the poor resolution issue that might appear.

Other solution : Neural Network-based
Advantages : **can cope with very poor resolution pictures.**

SLAM Problem

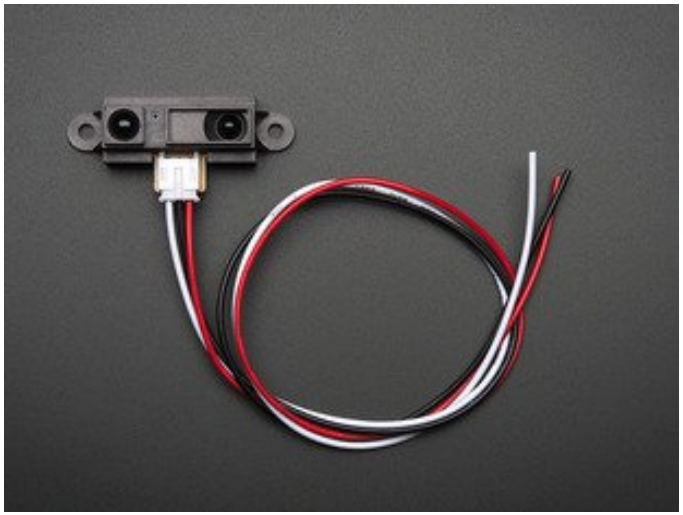
Simultaneous **L**ocalization **A**nd **M**apping

Relying on a Leader/Target (in movement) + Autonomous Behaviour in a Dynamical Environment :
Each robots within the swarm will have to localize itself in the frame of the Target and identify **obstacles** to avoid them.

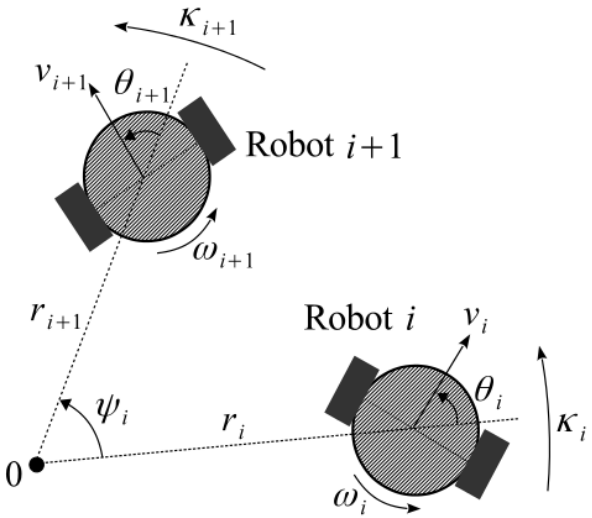
Approach : Computer Vision + Infra Red or UltraSonic Sensors or Laser Range Finder.

SLAM Problem - Hardware

Raspberry Pi 3 + Raspberry Pi Camera Module + very cheap and low quality Omnidirectional Lens Kogeto Dot
+ **IR Sensor (Analogic or Digital)**



Current Behaviour : Encircling (Nakamura et al.)



$$f(r_i, \hat{r}_i) := a r_i \left(1 - \frac{r_i^2}{\hat{r}_i^2}\right)$$
$$g(\psi_i) := \Omega + \varepsilon \sin \psi_i.$$

Non-linear System :

$$\boxed{\dot{r}_i = \bar{v}_i . \cos\theta_i}$$

$$\boxed{\dot{\theta}_i = \bar{\omega}_i}$$

$$\dot{\psi}_i = \frac{\bar{v}_{i+1}}{r_{i+1}} . \cos\theta_{i+1} - \frac{\bar{v}_i}{r_i} . \cos\theta_i$$

Control Law :

$$v_i = \overline{v}_i := k_v \{ f(r_i, \hat{r}_i) \cos\theta_i + r_i g(\psi_i) \sin\theta_i \}$$
$$\omega_i = \overline{\omega}_i := k_\omega \{ r_i g(\psi_i) \cos\theta_i - f(r_i, \hat{r}_i) \sin\theta_i \}$$

Addition 1 : Pursuit (Moshtagh et al.)

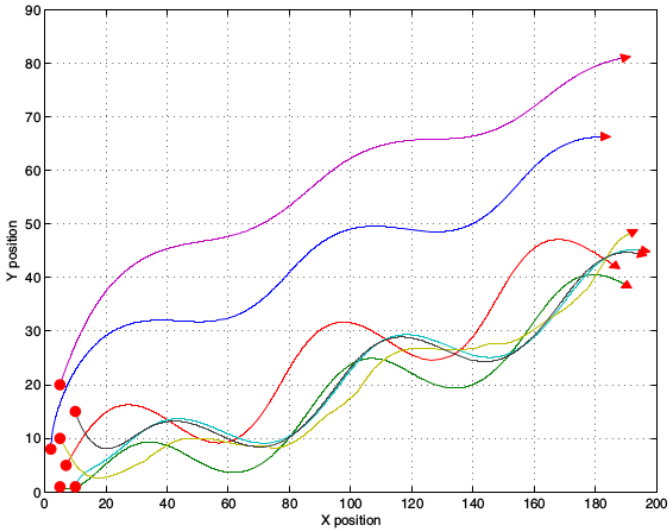


Figure – Simulation with a Leader/Target

Vision-based Control input :

$$\dot{\theta}_i = \frac{\sum_{j \in \mathcal{N}_i} (\dot{\beta}_{ij} \sin \beta_{ij} - \frac{1}{\tau_{ij}} \cos \beta_{ij})}{1 - \sum_{j \in \mathcal{N}_i} \sin \beta_{ij}}$$

Can be computed via Computer Vision techniques :

- β_{ij} : **Relative Bearing** with Agents j in the body-frame of Agents i.
- $\dot{\beta}_{ij}$: Rate of change of the relative bearing : **Optical Flow**.
- τ_{ij} : **Time to collision** with Agents j.

Addition 2 : Obstacle Avoidance (Bio-inspired)

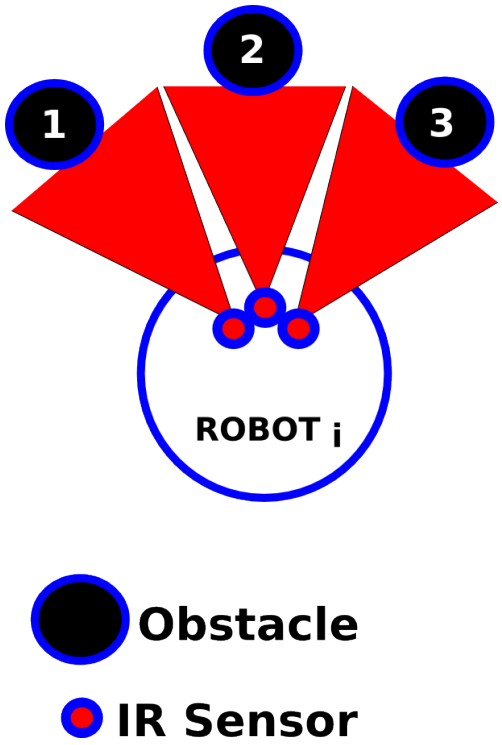


Figure – Situations that can occur : 1 : : obstacle on the left ; 2 : : obstacle in front ; 3 : : obstacle on the right.

IR Sensor-based Control input :

- **Situation 1 :**

$$\dot{\theta}_i = -K_{obstacle} * \Delta\theta$$

- **Situation 2 :**

$$\dot{\theta}_i = -K_{obstacle} * PID(\theta_i == \theta_{swarm})$$

- **Situation 3 :**

$$\dot{\theta}_i = K_{obstacle} * \Delta\theta$$

Control Law

Linear Composition of each behaviour :

$$\begin{bmatrix} \dot{r}_i \\ \dot{\theta}_i \end{bmatrix} = (1-\lambda)(1-\mu) \begin{bmatrix} \dot{r}_i^{Encircle} \\ \dot{\theta}_i^{Encircle} \end{bmatrix} + \lambda \begin{bmatrix} \dot{r}_i^{Obs} \\ \dot{\theta}_i^{Obs} \end{bmatrix} + \mu \begin{bmatrix} \dot{r}_i^{Pursuit} \\ \dot{\theta}_i^{Pursuit} \end{bmatrix}$$

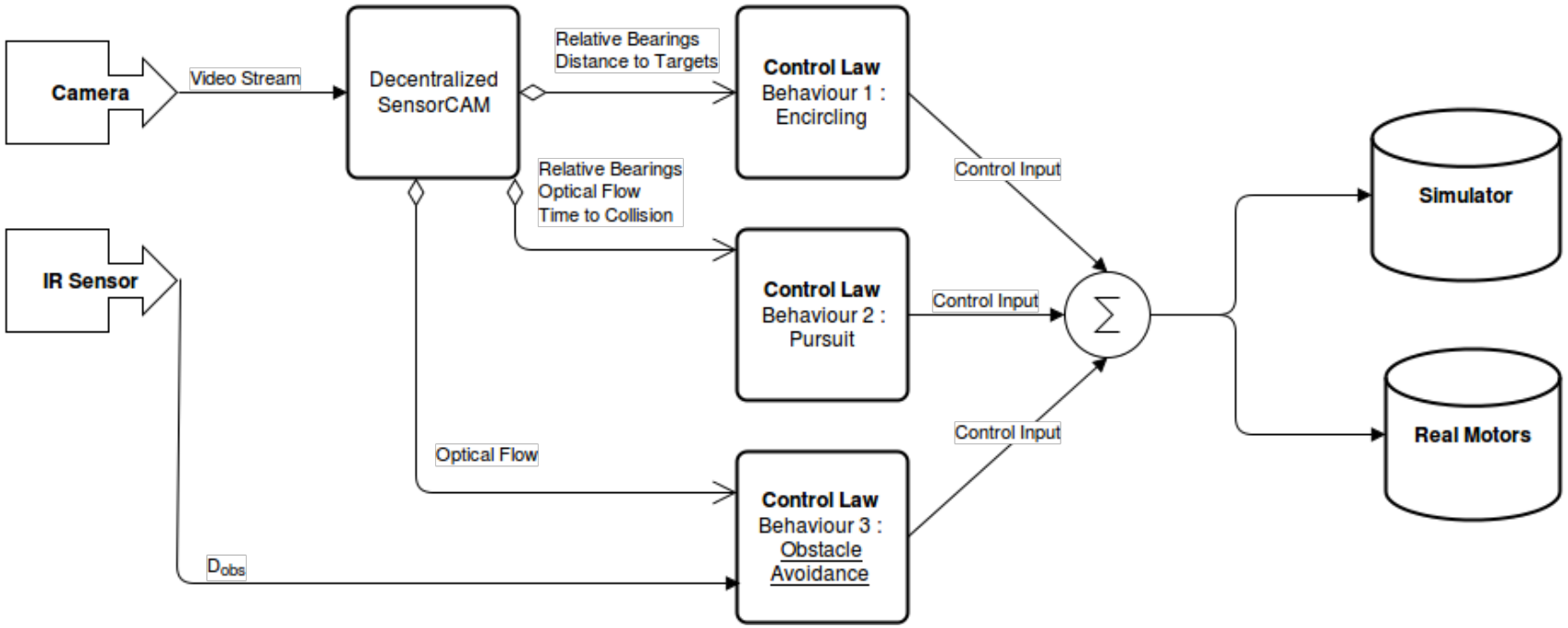
with :

$$\lambda = \frac{treshold_{obs}}{max(D_{obs}, treshold_{obs})} \parallel \mu = \frac{min(treshold_{pursuit}, \|\beta_{iTarget}\|)}{treshold_{pursuit}}$$

where :

- D_{obs} : distance to the obstacle, given by the IR Sensor.
- $\beta_{iTarget}$: optical flow of the Target, given by SensorCAM.

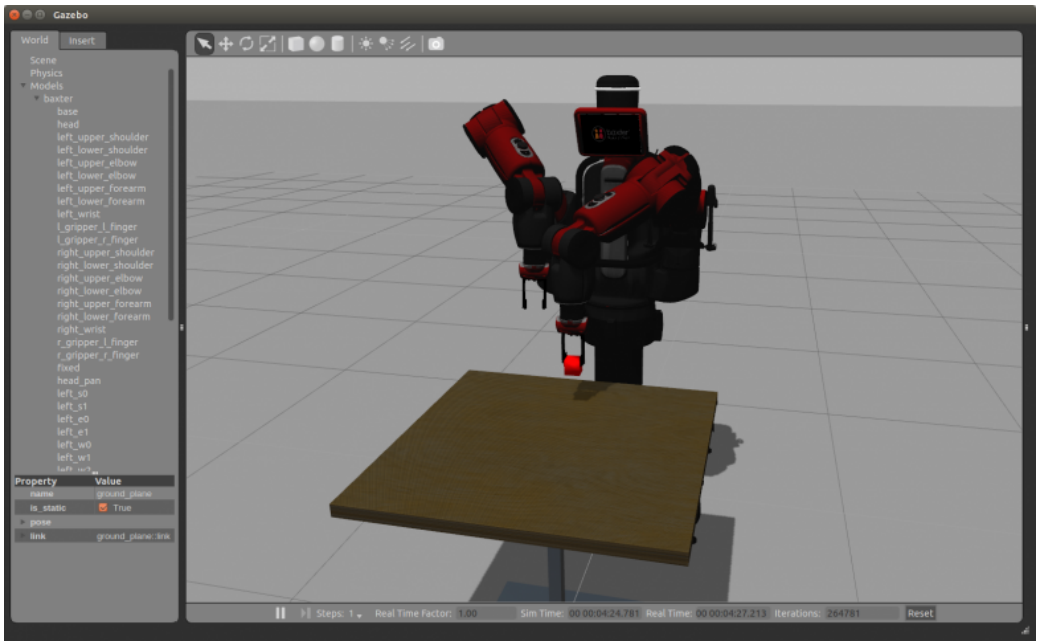
Block Diagram



Presentation

- Dynamics/Physic-based simulation using the best physics engine ODE, Bullets and Simbody.
- Easy to use and modify to suit your needs.
- Account for sensors simulation and noise.

GAZEBO



Simulated Environment

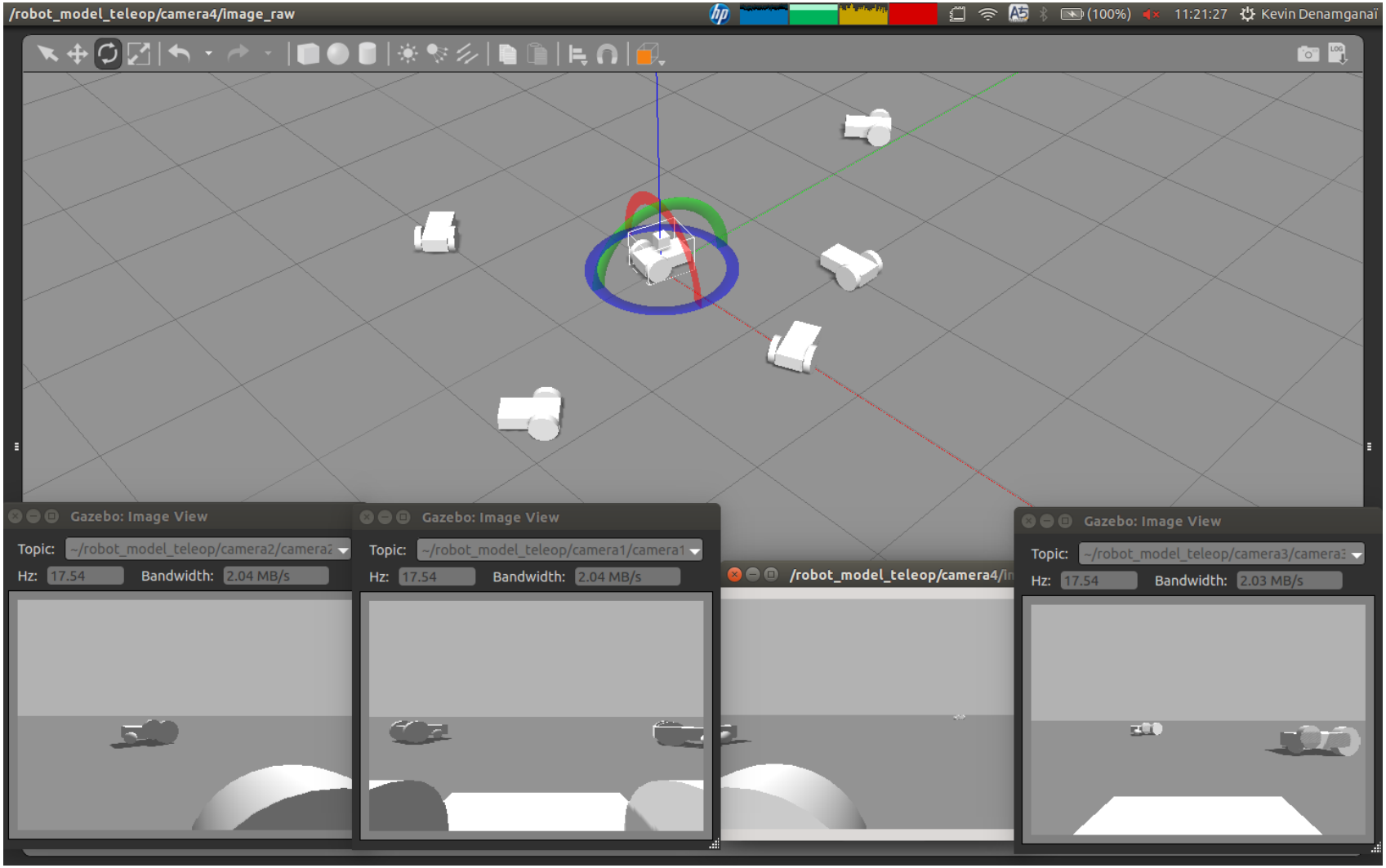
Description of the simulated Swarm :

- 5 to 15 robots.
- Omnidirectional vision for each robots.
- 2 or 3 proximity/distance sensors for each robots.

Description of the other simulated objects :

- Obstacles (walls, chairs, tables, moving objects...)
- Target that the swarm must follow.

Simulated Environment



Simulation

To Do :

- Implementation of the controller as a ROS node.
- Create simulation scenarios with obstacles and a moving target.

Once the control law is validated by the simulation, we will be able to purchase safely everything for the upgrade of the swarm robots.

Conclusion

Thank you for your atttention. ご清聴ありがとうございました。

Goal :
Being able to **add** and **substract** robots to or from the swarm without disrupting the swarm behaviour.

Current swarm behaviour :
Encircling a target in an evenly-spaced manner.

Problem :
How to recognize the robots that are currently interacting in the swarm ?

Train a Neural Network to **recognize digits/labels** associated to each robot.

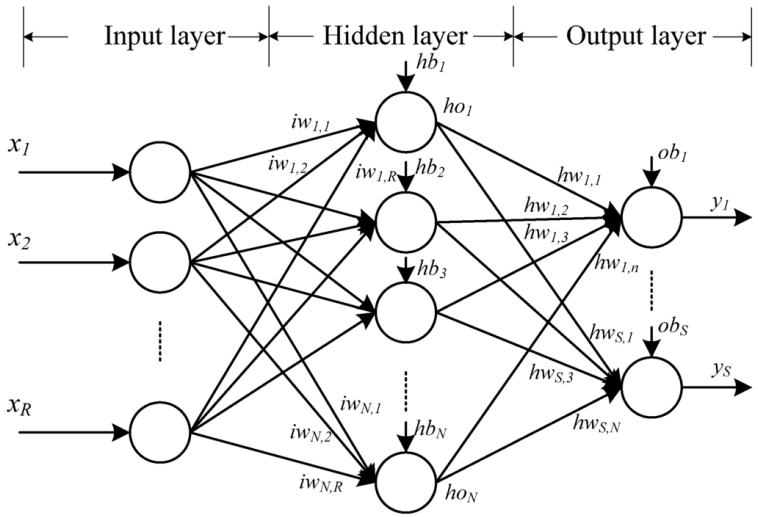


Figure – Neural Network architecture used on Digit Recognition.

Supervised Learning with a dataset of handwritten digits :

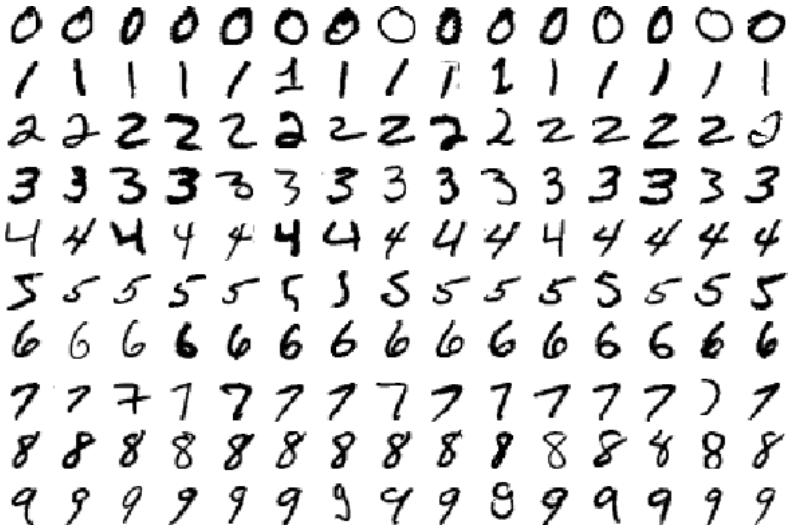


Figure – MNIST Dataset used to train the Neural Network on the recognition task.

Current test :

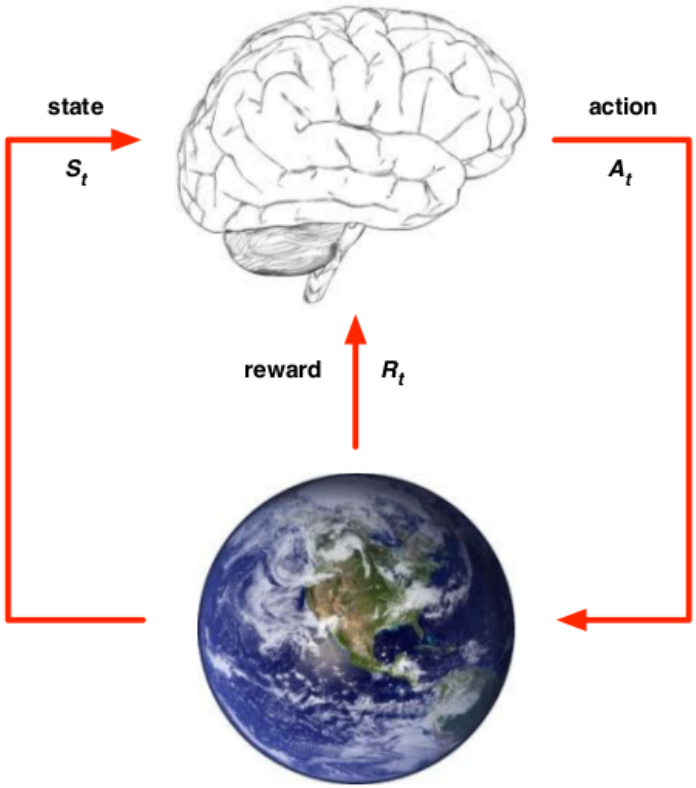
Some false recognitions remain but it is working effeciently enough to be used :

Figure – Current results

Learning & Reinforcement

Agent & Environment

Idea : *Solve* intelligence and use it to solve everything else.



Return : Total Reward from time step t given the discount factor $\gamma \in [0, 1]$:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}$$

Goal :
Choose the actions A_t that maximize the Return G_t.

Learning & Reinforcement

What is the Agent doing ?

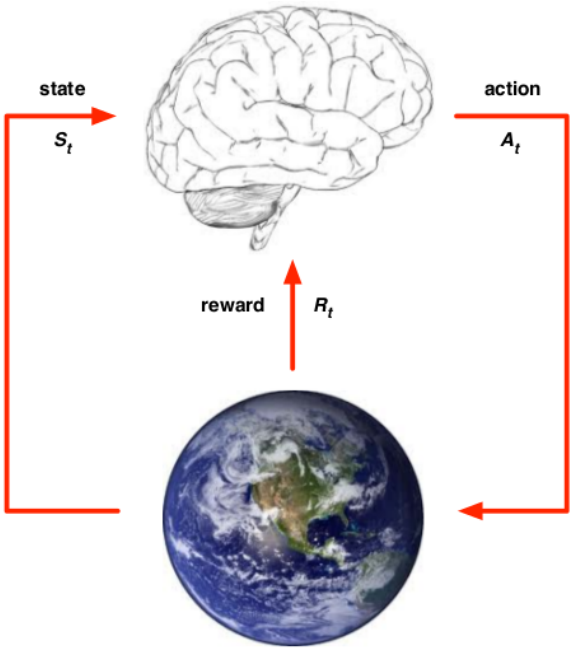


Figure – Reinforcement Learning typical loop of interactions.

- Learn a **probability distribution** of Actions a to apply given the States s : a **Policy** :

$$\pi(a, s) = \mathbb{P}[A_t = a | S_t = s]$$

$$\boxed{A_t = argmax_a \, \pi(a, s)}$$

- Learn a *sort of* **look-up table** of the **Expected Return** for each State s, or State-Action pair (s,a) :

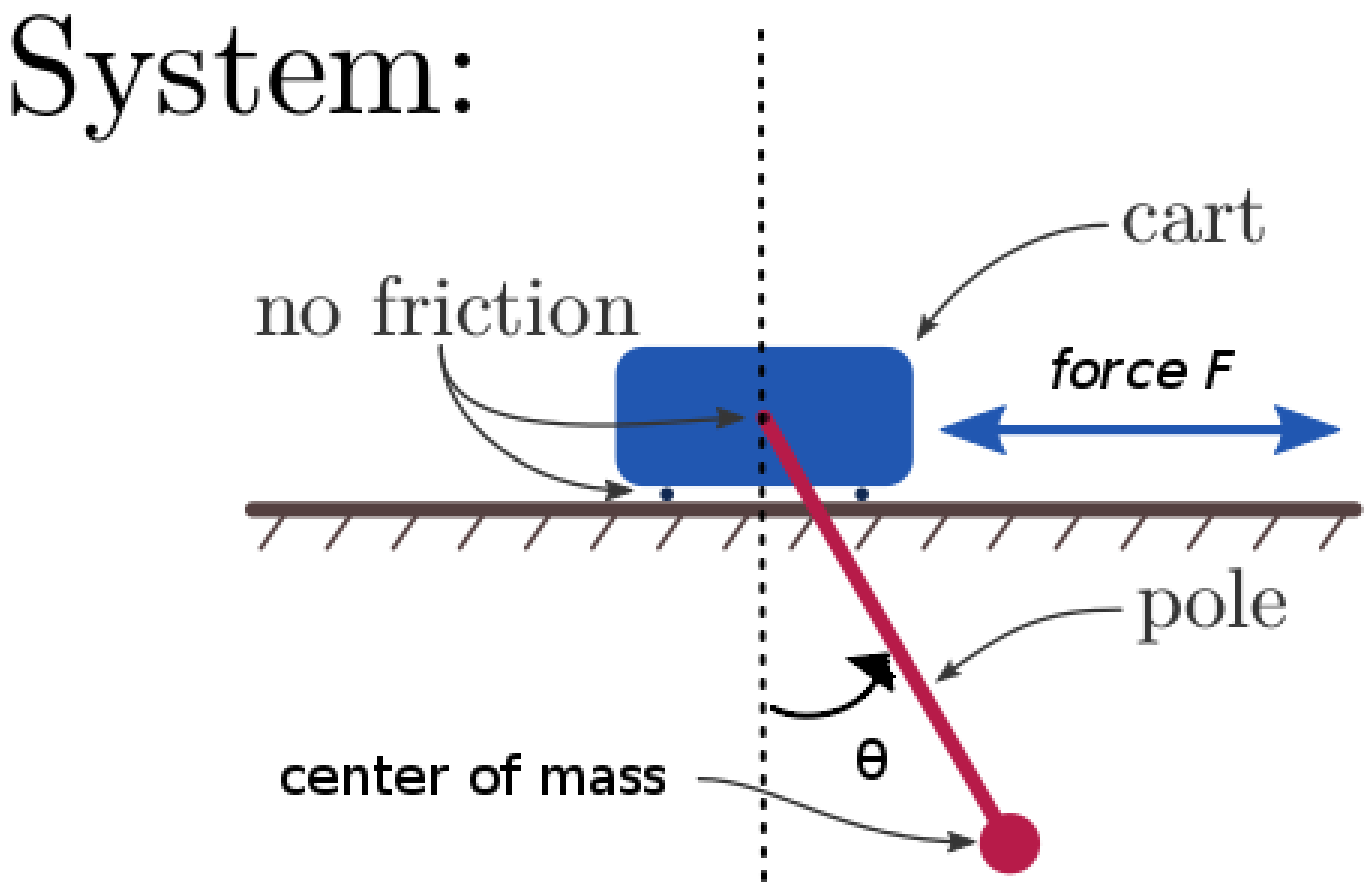
$$V(s) = \mathbb{E}[G_t | S_t = s]$$

$$Q(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

$$\boxed{A_t = argmax_a \, Q(s, a)}$$

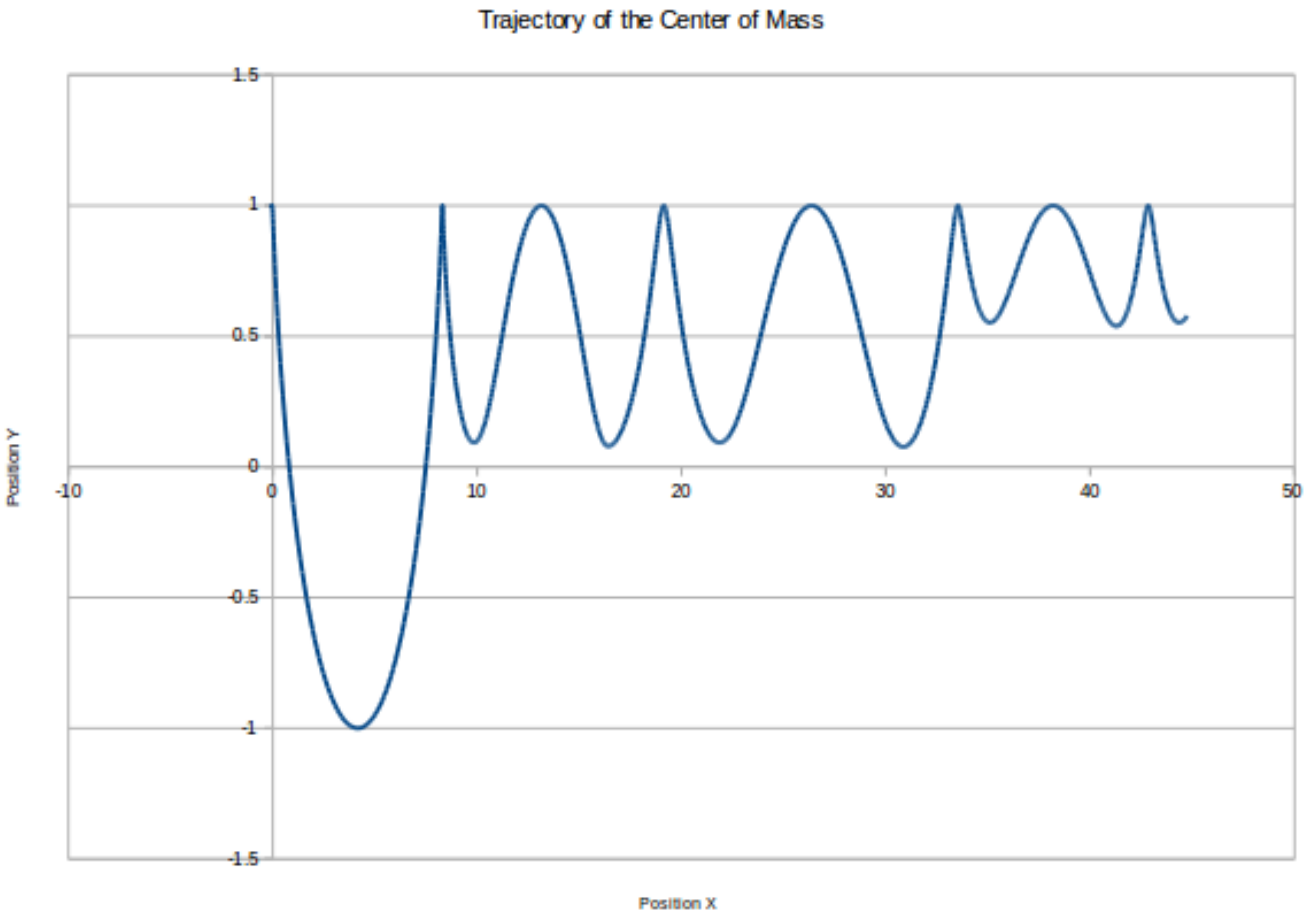
Validation Problem

A well-known problem : **Cart Pole Problem**



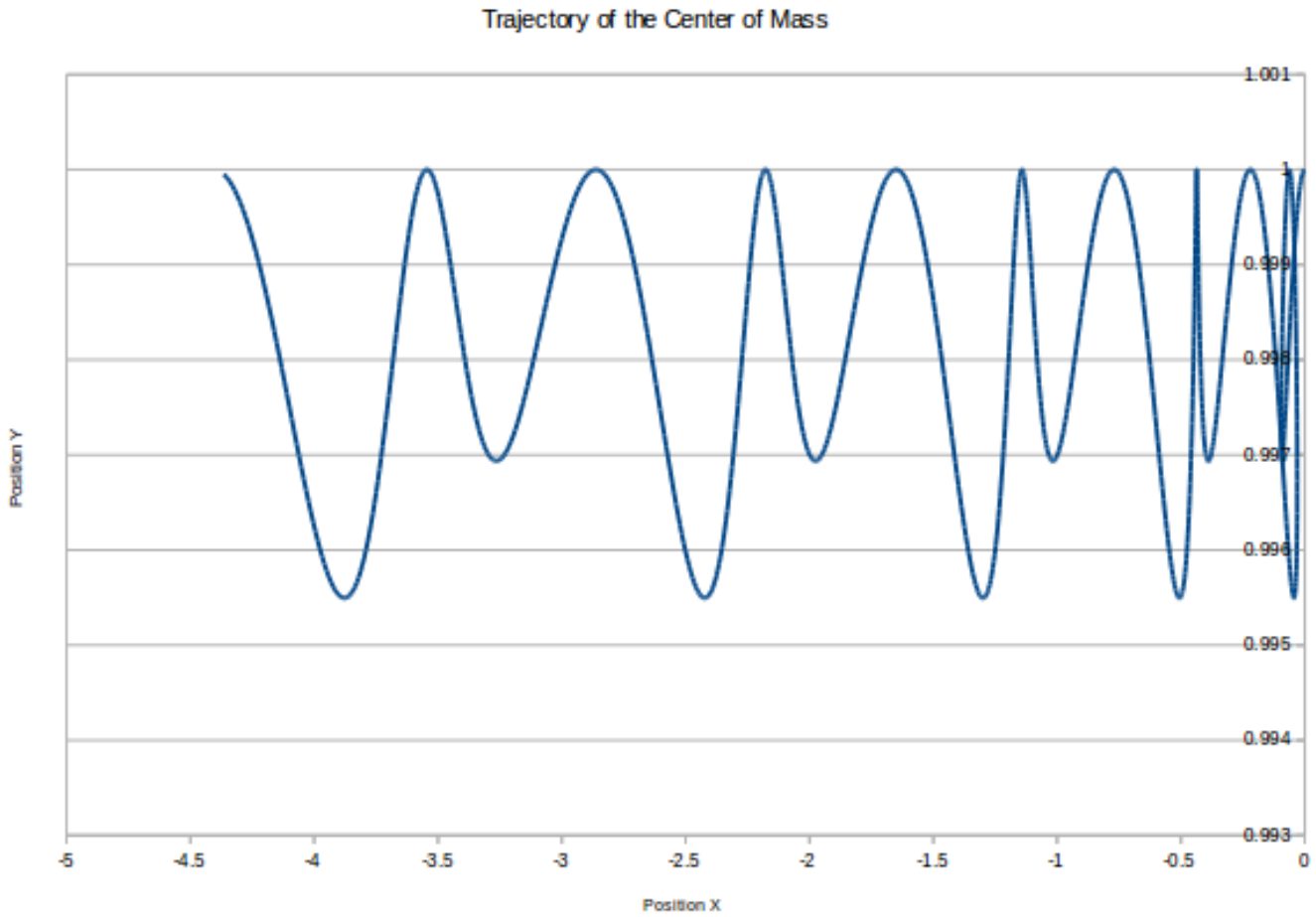
Evolution of the trajectories during the learning process :

Episode 600 :



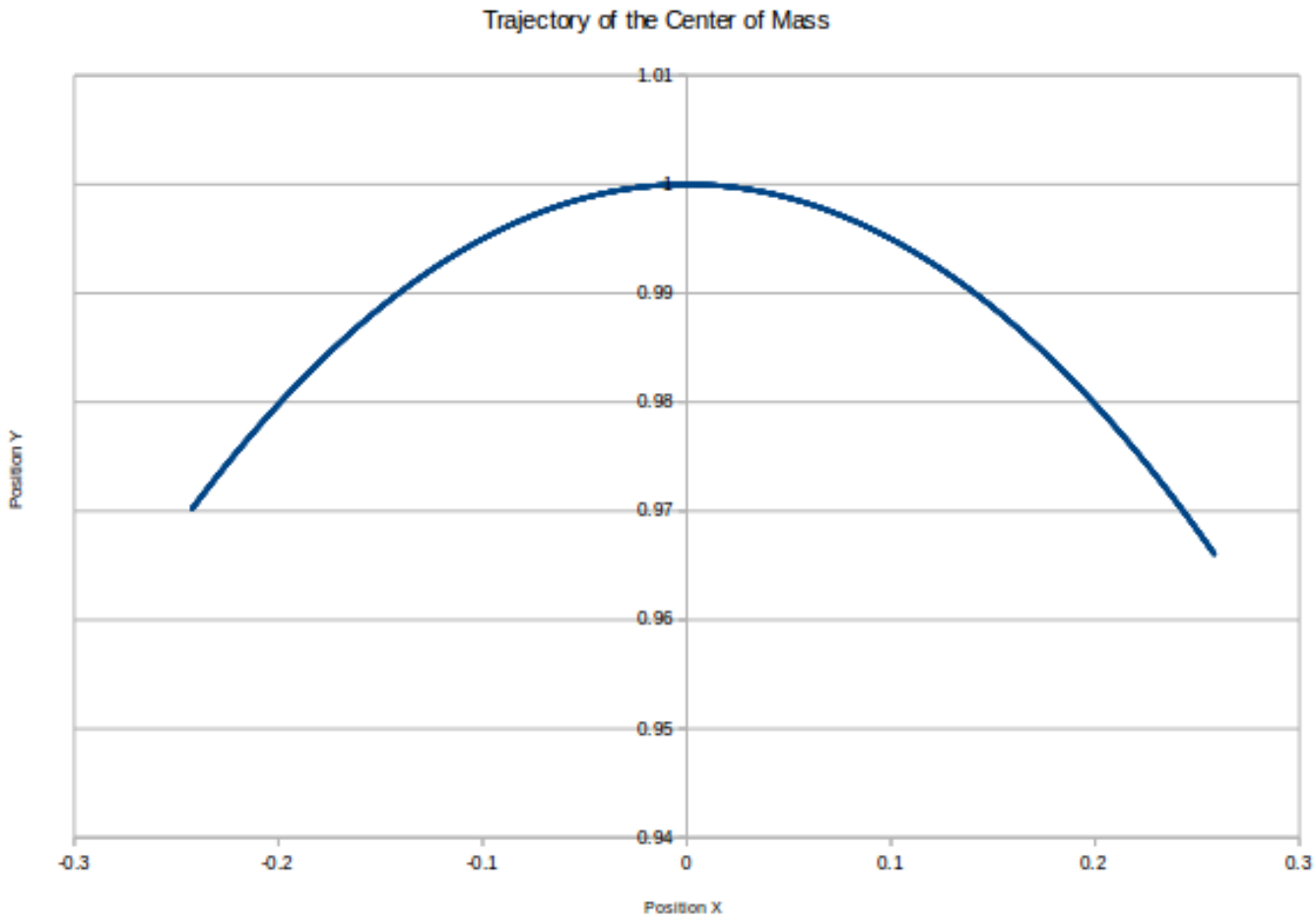
Evolution of the trajectories during the learning process :

Episode 1080 :



Evolution of the trajectories during the learning process :

Episode 2000 :



Effect of Reward Shaping

Previous Reward Function :

$$R_t = -(\theta - \pi)^2$$

New Reward Function :

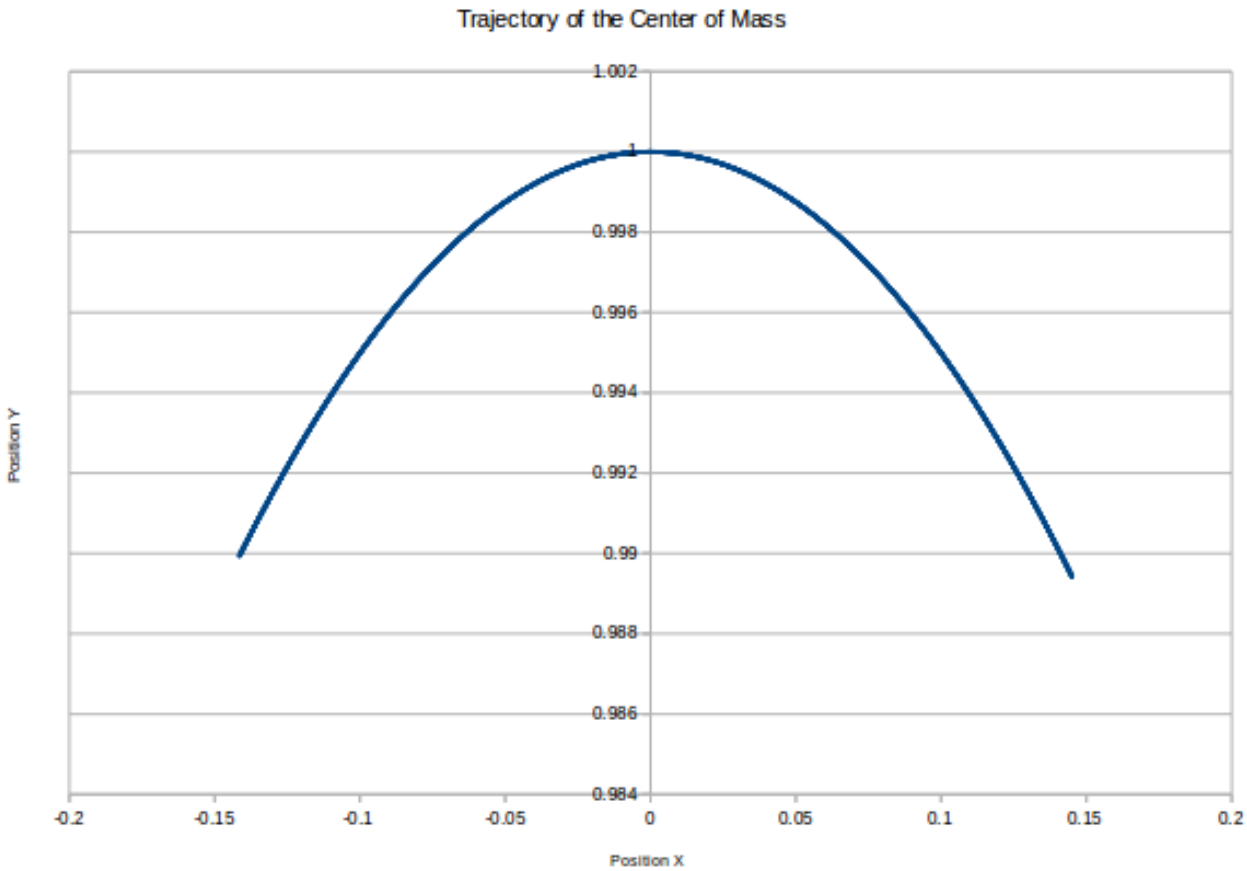
$$\boxed{R_t = -(\frac{\theta - \pi}{\pi})^2 - \beta x^2 - \gamma |F|}$$

$\beta = 5$, $\gamma = 5$

Effect of Reward Shaping

Optimal Policy

Learning with the new Reward Function : Episode 40



Agent & Environment

Formulation of our problem

Current Problem : Instability in the case of addition and/or substraction of robot to the swarm.

Formulation :

$$State : S_t \quad \begin{bmatrix} \Omega \\ \epsilon \\ a \\ k_v \\ k_w \\ r_1 \\ \vdots \\ r_n \end{bmatrix}_t \rightarrow \left\{ \quad \textbf{AGENT} \quad \right\} \rightarrow \quad Action : A_t \quad \begin{bmatrix} \Omega \\ \epsilon \\ a \\ k_v \\ k_w \\ \dot{r}_1 \\ \vdots \\ \dot{r}_n \end{bmatrix}_t$$

Different Architectures -

Current Architecture and Desired Architecture :

Heterogeneous	vs	Homogeneous/Anonymous
No communication	vs	Communication
Centralized Sensors	vs	Decentralized Sensors
Decentralized Computation	vs	Centralized Computation
Leader-free	vs	Relying on a Leader/Target

DATMO Problem

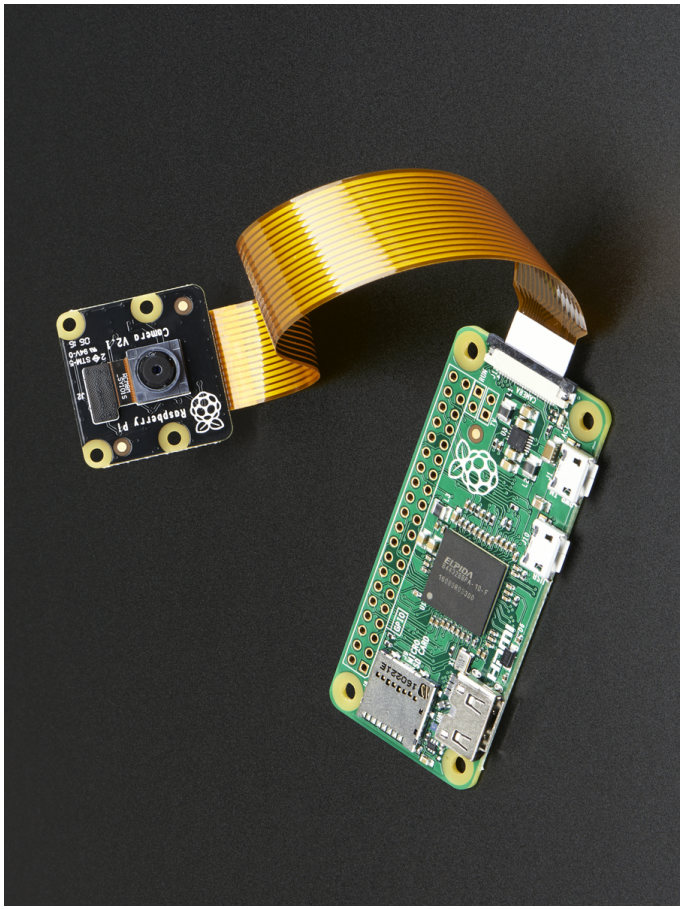
Detection **A**nd **T**racking of **M**oving **O**bjects

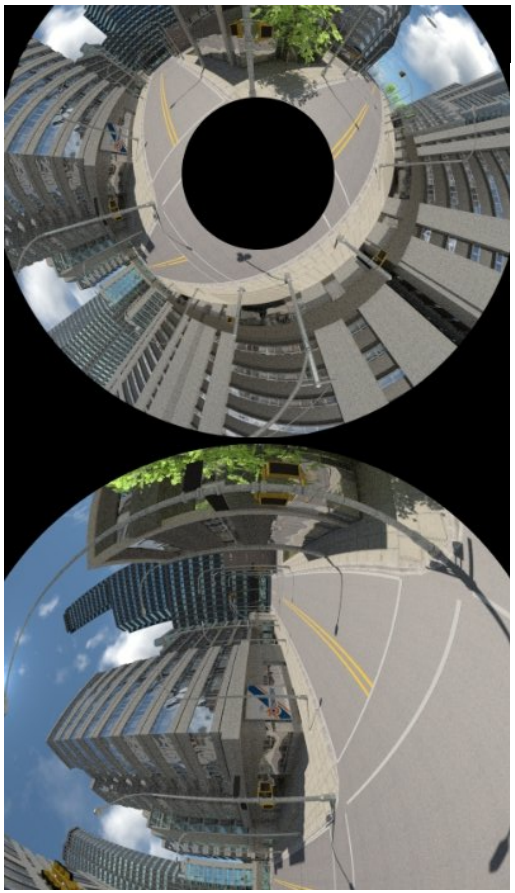
Decentralized Sensing + Homogeneous Robots :
Each robots within the swarm will have to detect and track its neighbours.

Approach : Computer Vision + Kalman Filter.

DATMO Problem - Potential Hardware

Raspberry Pi + Fish-Eye/Omnidirectional Camera :





SLAM Problem

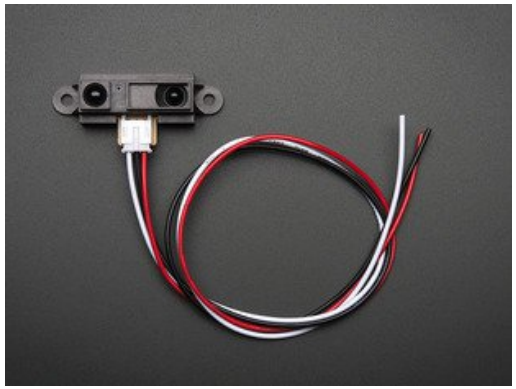
Simultaneous **L**ocalization **A**nd **M**apping

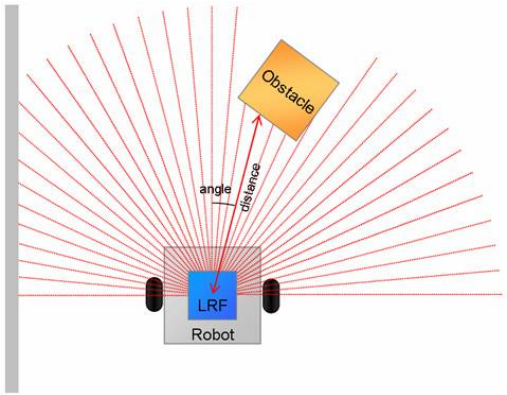
Relying on a Leader/Target (in movement) + Autonomous Behaviour in a Dynamical Environment :
Each robots within the swarm will have to localize itself in the frame of the Target and identify **obstacles** to avoid them.

Approach : Computer Vision + Infra Red or UltraSonic Sensors or Laser Range Finder.

SLAM Problem - Potential Hardware

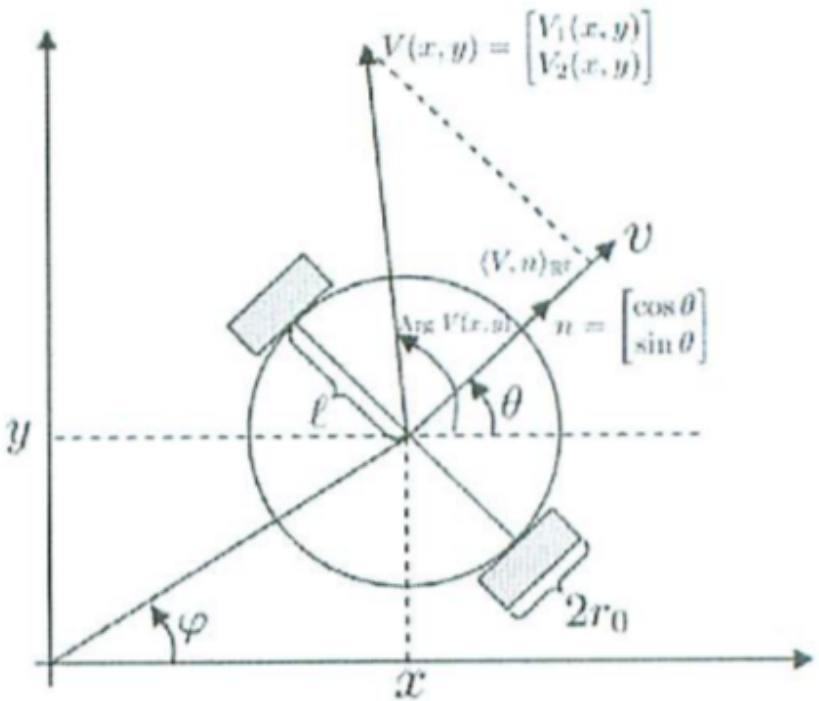
Raspberry Pi + *Omnidirectional Camera* + **IR/UltraSonic Sensor** or **Laser Range Finder** :

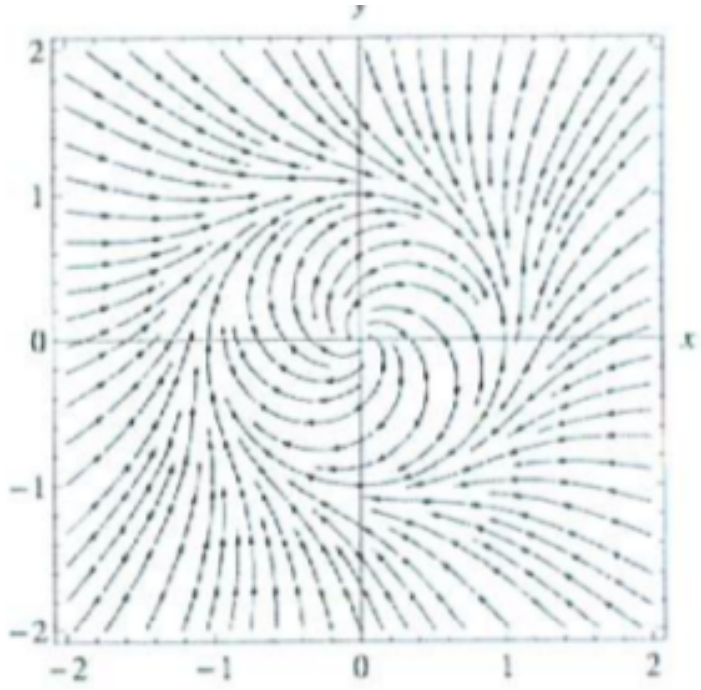




Trajectory Tracking of a single two-wheeled robot along a Circular Limit Cycle Vector Field - 単一の二輪ロボットの円状のリミットサイクルベクトル場への軌道追従

(N. Hara et al., American Control Conference 2010)

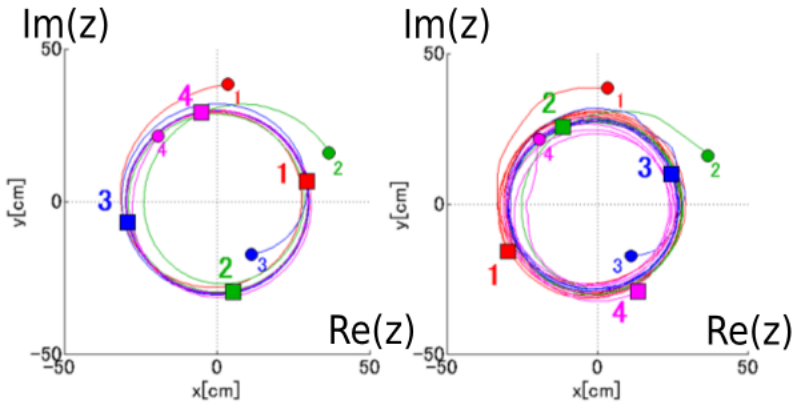




Landau-Stuart Coupled Oscillators applied to Control a Group of Robots - ロボットの群れを結合させのために,Landau-Stuartの結合発振器の特性をロボット群に応用

(M.Tsukiji et al., SICE Annual Conference 2014)

$$\dot{z}_j = \left(a + i\omega - \left|z_j\right|^2\right) z_j + \varepsilon \sum_{m=1}^{d} \left(z_{j+m} - z_j + \delta(z_{j-m} - z_j)\right)$$



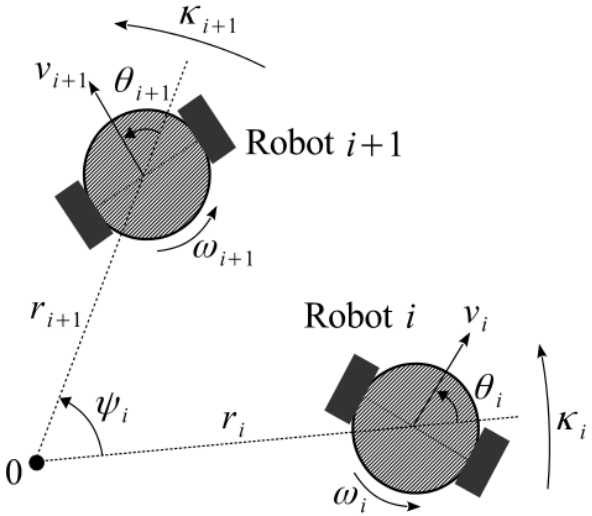
(a) Simulation result. (b) Experimental result.

Fig. 3 Trajectories of robot group ($N = 4, k = 1$).

Im(z) Im(z)

Phase Coupled Oscillators applied to Control a Group of Robots - 位相結合振動子をロボット群に応用

(T.Nakamura et al., IFAC 2016, to be accepted)



$$f(r_i, \hat{r}_i) := a r_i \left(1 - \frac{r_i^2}{\hat{r}_i^2}\right)$$

$$g(\psi_i) := \Omega + \varepsilon \sin \psi_i.$$

Non-linear System :

$$\dot{r}_i = \bar{v}_i . \cos\theta_i$$

$$\dot{\theta}_i = \bar{\omega}_i$$

$$\dot{\psi}_i = \frac{\bar{v}_{i+1}}{r_{i+1}} . \cos\theta_{i+1} - \frac{\bar{v}_i}{r_i} . \cos\theta_i$$

Control Law :

$$v_i = \overline{v}_i := k_v \left\{ f(r_i, \hat{r}_i) \cos\theta_i + r_i g(\psi_i) \sin\theta_i \right\}$$

$$\omega_i = \overline{\omega}_i := k_\omega \left\{ r_i g(\psi_i) \cos\theta_i - f(r_i, \hat{r}_i) \sin\theta_i \right\}$$

Phase Coupled Oscillators applied to Control a Group of Robots - 位相結合振動子をロボット群に応用

(T.Nakamura et al., IFAC 2016, to be accepted)

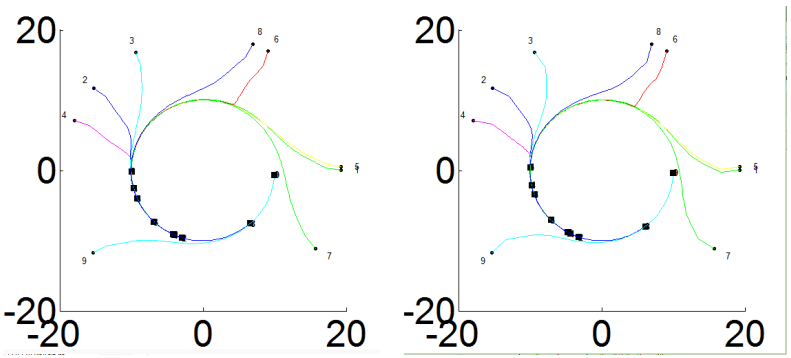
Stability Analysis of the Swarm of Robots :

	$\frac{l}{N} \in \left[0, \frac{1}{4}\right)$	$\frac{l}{N} \in \left(\frac{1}{4}, \frac{1}{2}\right]$	$\frac{l}{N} \in \left[\frac{1}{2}, \frac{3}{4}\right)$	$\frac{l}{N} \in \left(\frac{3}{4}, 1\right]$
$\Omega \geq 0$	$k_\omega > 0$			
	$\varepsilon > 0$		$\varepsilon < 0$	
	$k_v > 0$ $a > 0$	$k_v < 0$ $a < 0$	$k_v > 0$ $a > 0$	$k_v < 0$ $a < 0$
$\Omega \leq 0$	$k_\omega < 0$			
	$\varepsilon < 0$		$\varepsilon > 0$	
	$k_v < 0$ $a < 0$	$k_v > 0$ $a > 0$	$k_v < 0$ $a < 0$	$k_v > 0$ $a > 0$

Familiarization with the system : Simulation-related Issues

Runge-Kutta vs Euler

Simulation Time : 10 seconds.
Time Step : 0.1 second



Summed Differences : 255.0313
Average Error on each trajectory : 27.7813

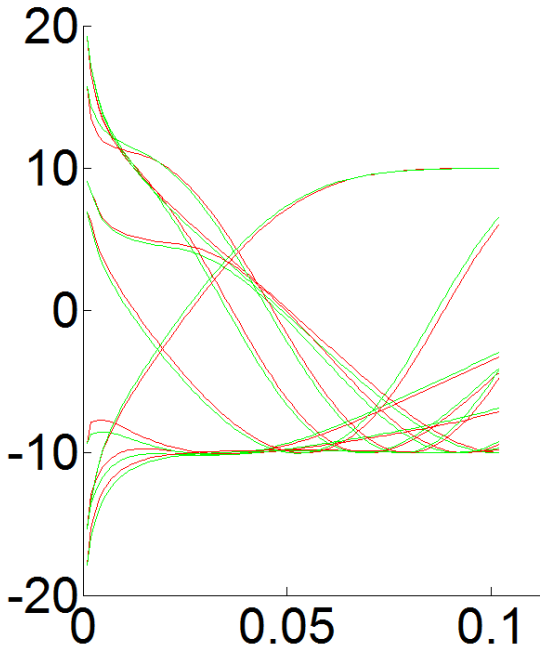


Figure – X-axis : iteration/100 ; Y-Axis : X space coordinate of the robots.

Current Model / Problem

$$\begin{bmatrix} \Omega \\ \epsilon \\ a \\ k_v \\ k_w \end{bmatrix}_t \rightarrow \left\{ \quad \text{システム} \quad \right\} \rightarrow \textit{Robust Behaviour}$$

Robustness to disturbances such as :

- the addition of a robot to the system, or
- the substraction of a robot to the system, due to its failure.

Learning & Reinforcement

Exploitation vs Exploration

Exploration : The more state S_t we visit **the better is our estimation** of the Q value-state function.

$$\epsilon - greedy(s) = \begin{cases} \text{random action} a & \text{with probability} \epsilon \\ greedy(s) & \text{with probability} 1 - \epsilon \end{cases}$$

Exploitation : Our goal is to maximize the return G_t.

$$greedy(s) = argmax_a \, Q(s, a)$$

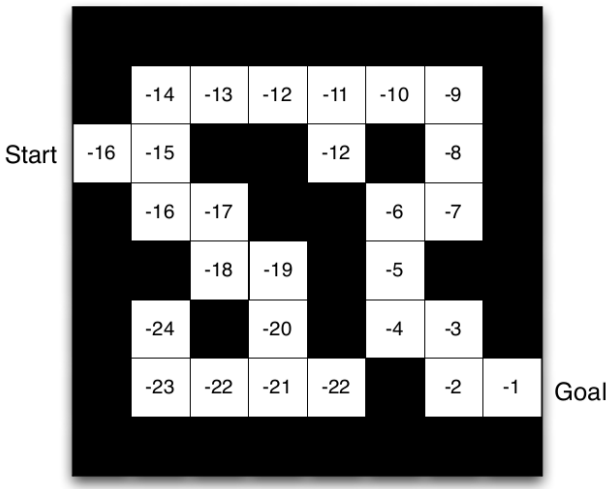
Q-Learning Algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$;
  until S is terminal

Function Approximator & Deep Neural Networks

Basics

From Discrete Finite Action & State Spaces : **Table Lookup**



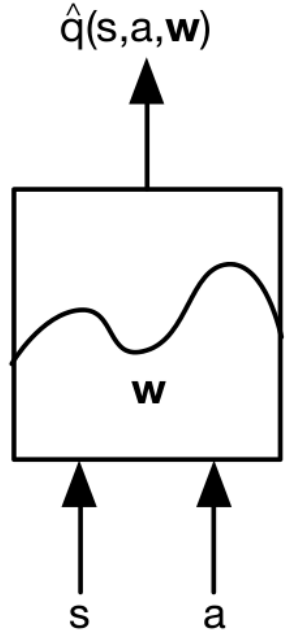
To Continuous Action & State Spaces : **Function Approximator**

For instances :

- **neural network** with weights as parameters
- linear combination of features with parameter w :

$$\phi(s, a) = \mu(s) * w(a)^T$$

- decision tree
- ...



Function Approximator & Deep Neural Networks

Atari Game Player Example

Q state-value function approximated by a Neural Network :

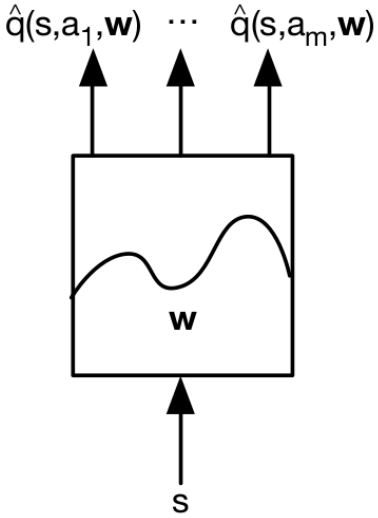


Figure – Draft architecture of the Neural Networks.

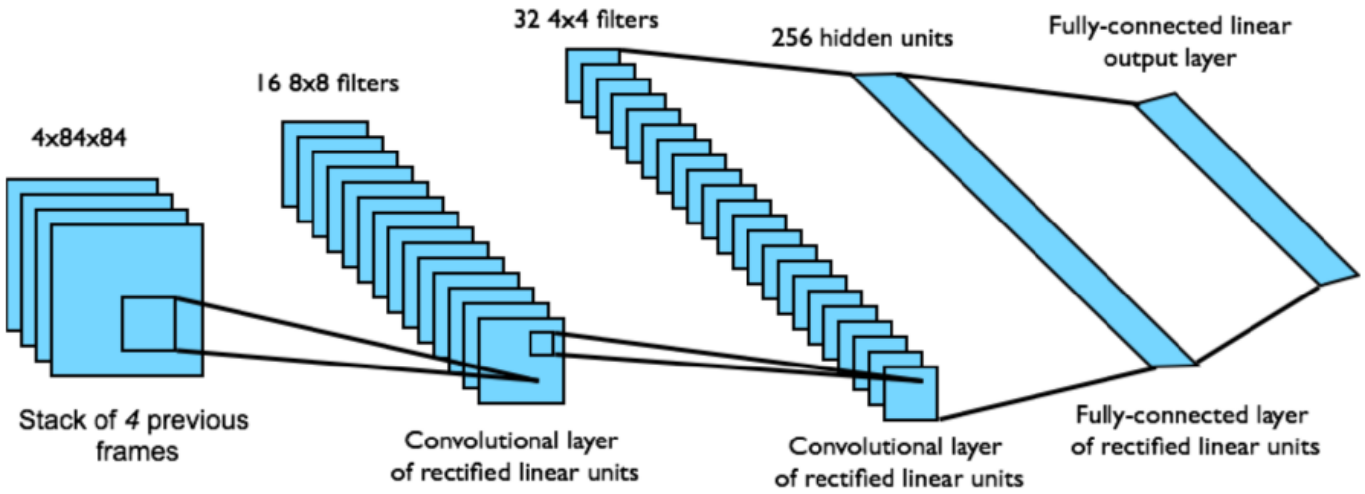


Figure – Deep Neural Networks architecture used to play Atari games.

Agent & Environment

Into the Code

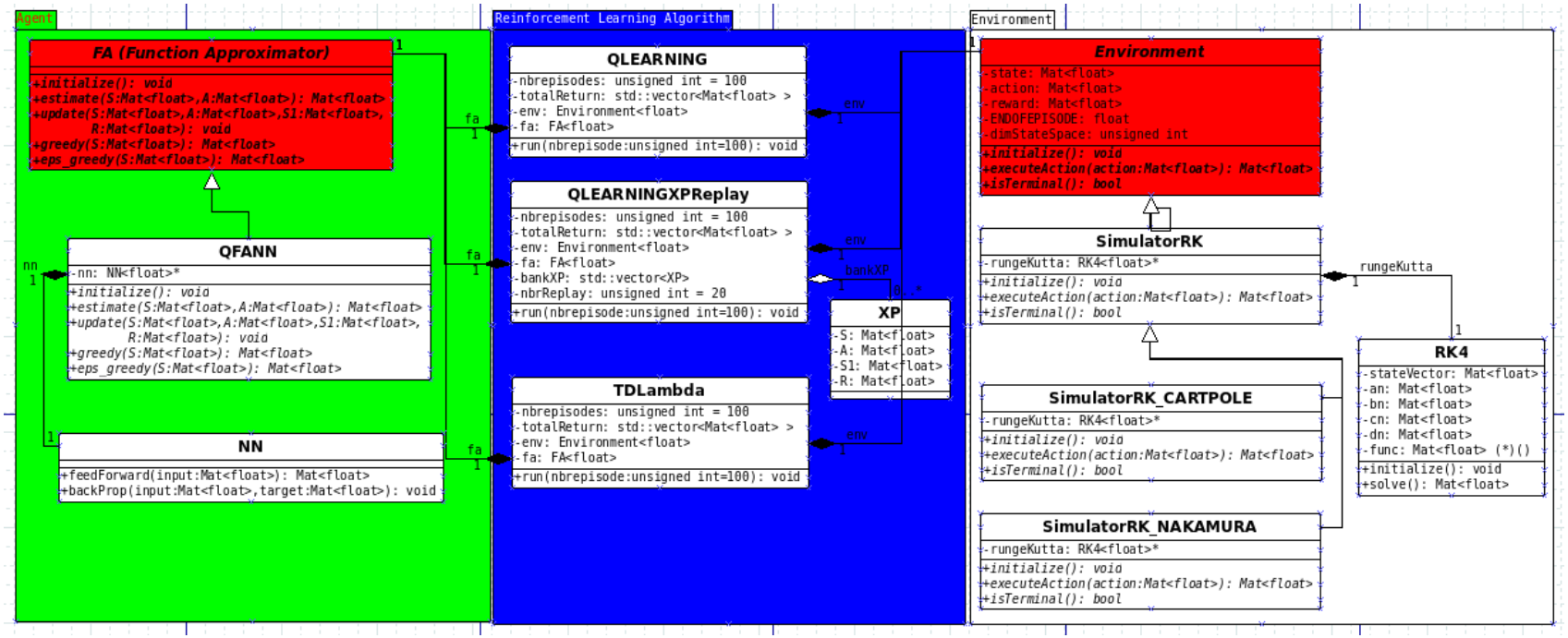


Figure – Reinforcement Learning Framework in C/C++

Agent & Environment

Best Results

Still far from convergence...