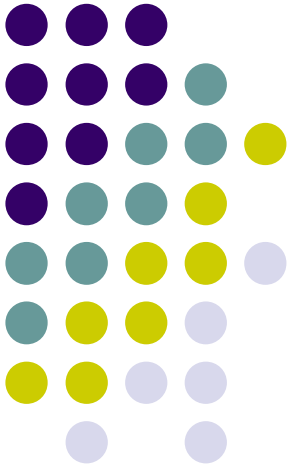
Distributed multi-agent architecture for redundant robots applied to humanoïd

Philippe Lucidarme

philippe.lucidarme@univ-angers.fr

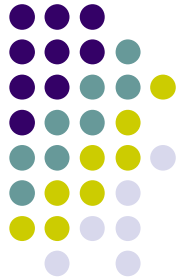
LISA - Université d'Angers



université angers

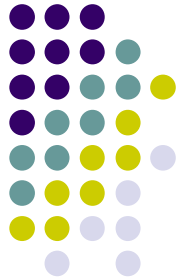
lisa

Contexte



- Architecture distribuée
- Robots sériels redondants
  - Tolérant aux pannes
  - Extraction des minimums locaux
- Application aux robots humanoïdes
- Multi objectifs
  - Ralliement de cible
  - Stabilité statique

Etat de l’art (SMA)

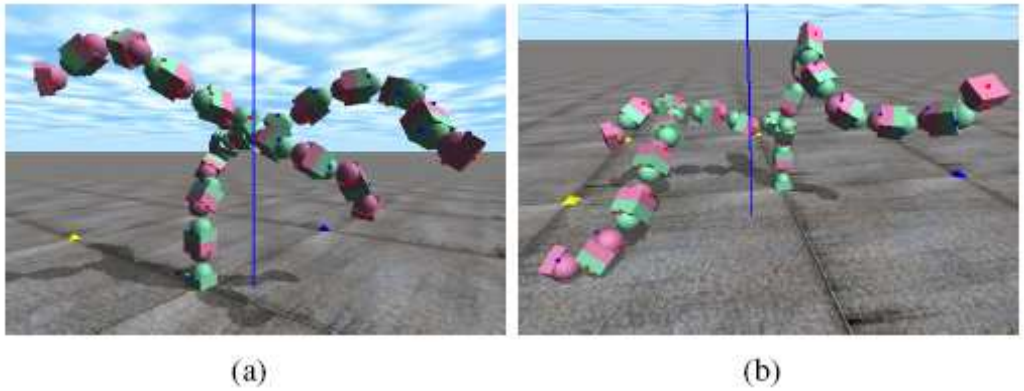


- 80’s : Intelligence artificielle distribuée (Ethologie) [R. Brooks]
- 90’s : Travail coopératifs et le contrôle de formations de robots mobiles [R. Arkin, L.E. Parker, A. Drogoul et J. Ferber]
- 00’s : Applications aux robots redondants / humanoïdes [K. Ning, T. Xu-yan et T. Tao]
- 00’s : Applications aux robots reconfigurables [M. Moll, P. Will, M. Krivokon et W-M. Shen]

Robotique reconfigurable

Video from the National Institute of Advanced Industrial Science and Technology (AIST, Japan)
[A. Kamimura, S. Murata, E. Yoshida, H. Kurokawa, K. Tomita and S. Kokaji]

Control of the CoM of a modular robot [Mark Moll and al. 2005]



(a) (b)

(c) (d)

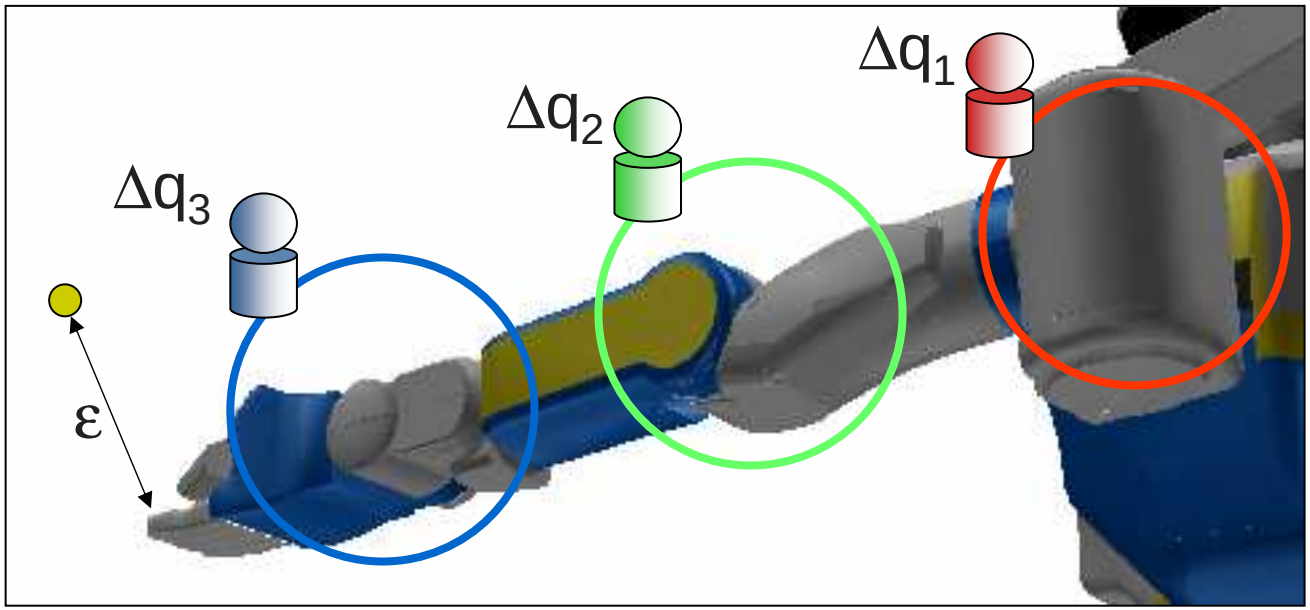
Algorithm 1 UpdateMass

```
1:  while true do
2:      clear update flags for all connectors
3:      while ¬inbox.empty() do        ▷ process all incoming
4:          msg = inbox.pop()          ▷ messages
5:          connectorMass[msg.destination] = msg.mass
6:          mark other connectors for update
7:      end while
8:
9:      RecomputeMass()
10:
11:     for i = 1 . . . n do                    ▷ update neighbors
12:         if connectorMass[i] is marked for update then
13:             send connector i (mass − connectorMass[i])
14:         end if
15:     end for
16: end while
```

Algorithm 2 RecomputeMass()

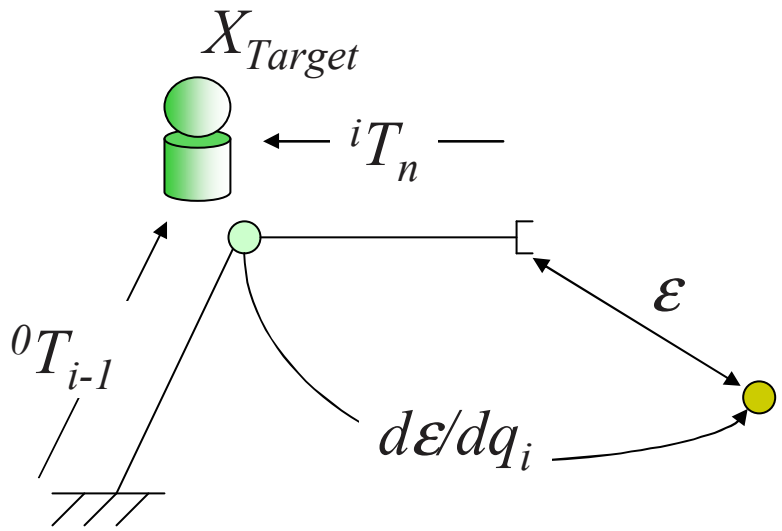
```
1: recompute moduleMass ▷ the mass properties of just this
                        ▷ module
2: if moduleMass has changed then
3:     mark all connectors for update
4: end if
5: mass = moduleMass
6: for i = 1 . . . n do
7:     mass = mass + connectorMass[i]
8: end for
```

Architecture distribuée

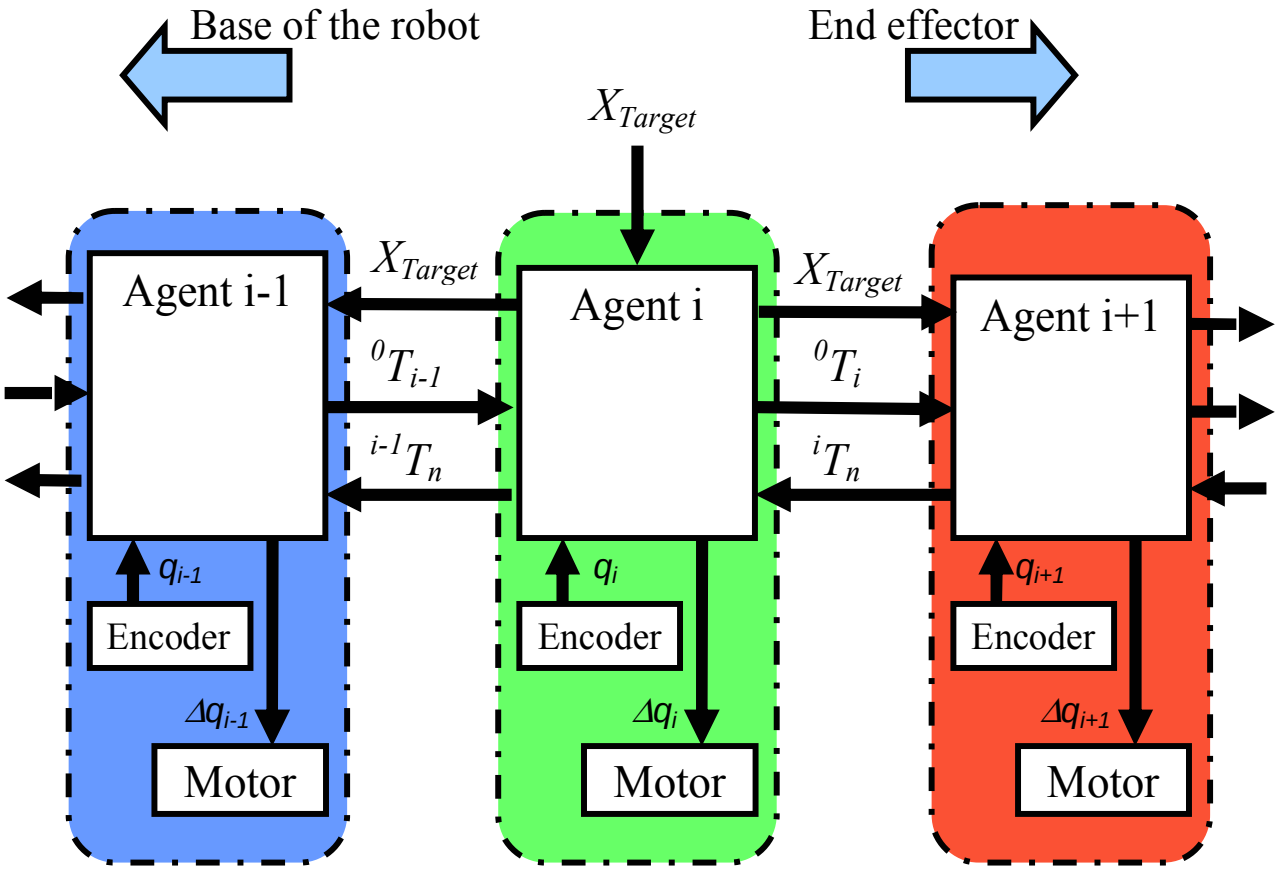


- La position de la cible est connue
- Chaque agent agit indépendamment
- Les communications sont locales

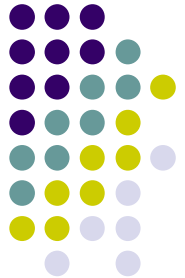
Système équivalent



Vue générale



Comportement

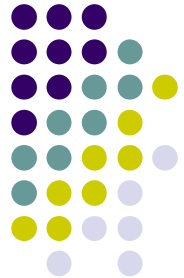


- Descente du gradient

$$\Delta q_i(t) = -\alpha_i . \frac{d\varepsilon}{dq_i}$$

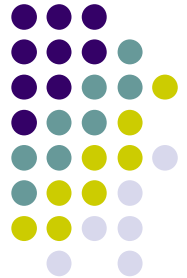
$$\frac{d\varepsilon}{dq_i} = \frac{(x_{Target} - x_n).\frac{dx_n}{dq_i} + (y_{Target} - y_n).\frac{dy_n}{dq_i} + (z_{Target} - z_n).\frac{dz_n}{dq_i}}{\sqrt{(x_n - x_{Target})^2 + (y_n - y_{Target})^2 + (z_n - z_{Target})^2}}$$

Algorithme



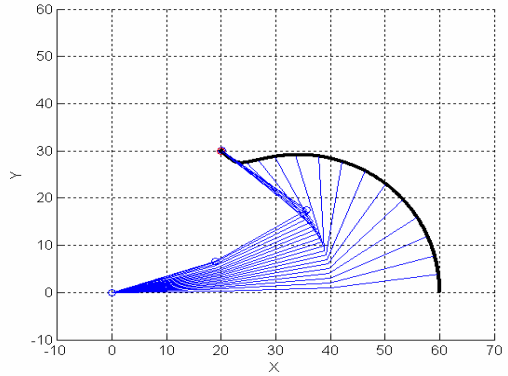
Algorithm **AgentBehavior**
input : ${}^{O}T_{i-1}$, ${}^{i}T_{n}$, P_{Target}, q_i output : Δq_i, ${}^{i-1}T_{n}$, ${}^{O}T_{i}$
At each time step : Compute : ${}^{i-1}T_{i}$ and $\frac{d^{i-1}T_i}{dq_i}$ Compute : $\frac{dP_n}{dq_i}$ and $\frac{d\varepsilon}{dq_i}$ Update : ${}^{O}T_{i} = {}^{0}T_{i-1} . {}^{i-1}T_{i}$ Update : ${}^{i-1}T_{n} = {}^{i-1}T_{i} . {}^{i}T_{n}$ Update : $\Delta qi = -\alpha_i . \frac{d\alpha\varepsilon}{dq_i}$

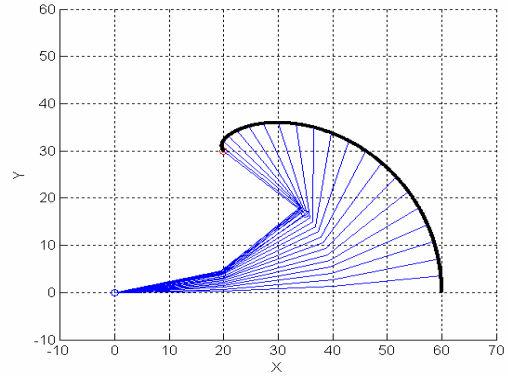
Expression générale

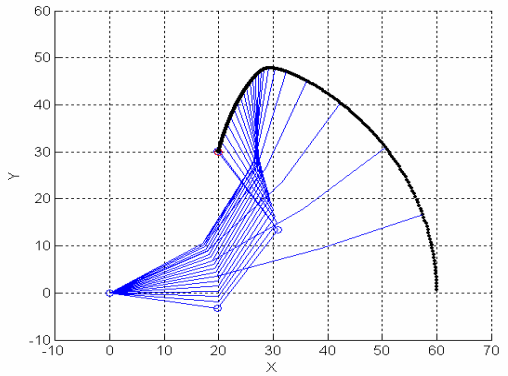


$$\begin{bmatrix} q_1 \\ q_2 \\ q_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_n \end{bmatrix} + \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_n \end{bmatrix} . \begin{bmatrix} \frac{\Delta\varepsilon}{\Delta q_1} \\ \frac{\Delta\varepsilon}{\Delta q_2} \\ \frac{\Delta\varepsilon}{\Delta q_3} \end{bmatrix}$$

Influence des coefficients



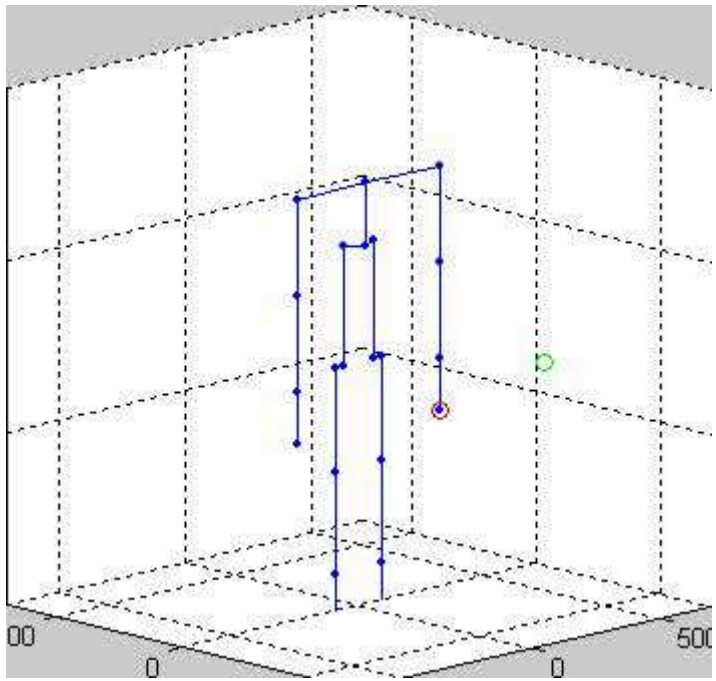




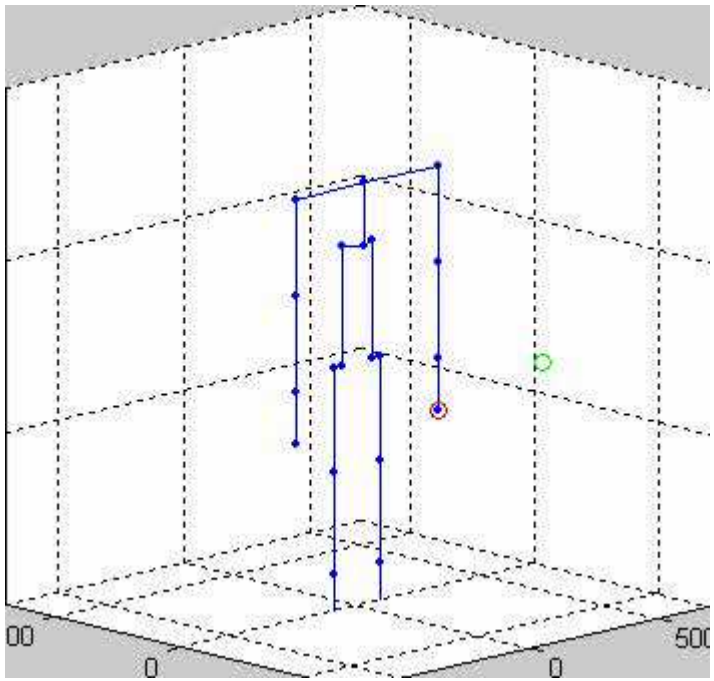
- Algorithme évolutionniste [P. Lucidarme, 2008]

Tolérance aux pannes

Fonctionnement normal

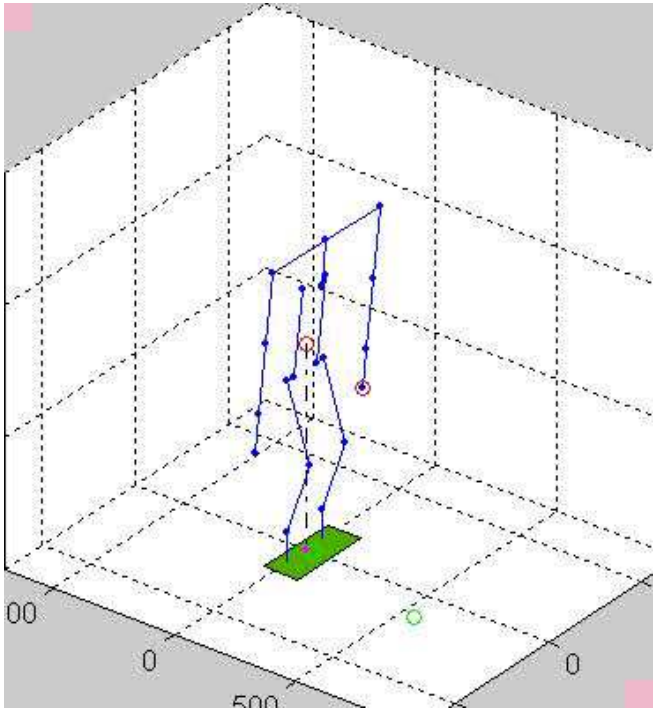


Epaule bloquée



Stabilité

- Garantir que la projection du CdG reste dans le polygone
- Architecture identique
- Minimiser la distance
  - Le centre du polygone
  - La projection du CdG

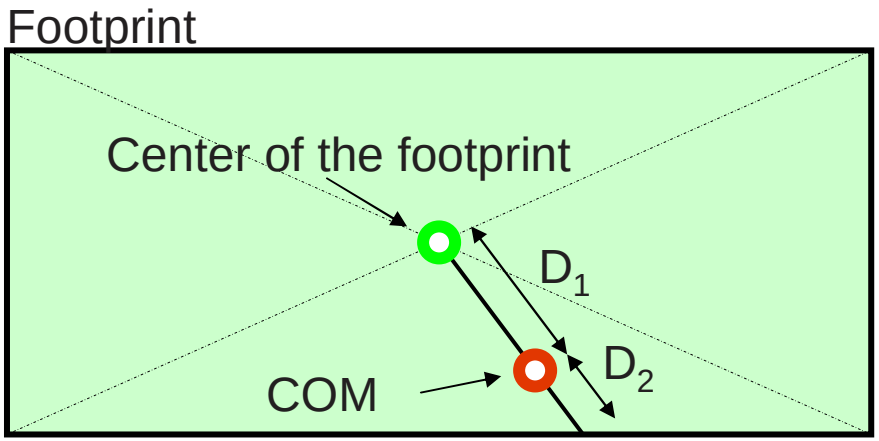


Multi-objectif

$$\Delta q_i = -\gamma.\alpha_i^{Target}.\frac{d\varepsilon_{Target}}{dq_i} - (1-\gamma).\alpha_i^{COM}.\frac{d\varepsilon_{COM}}{dq_i}$$

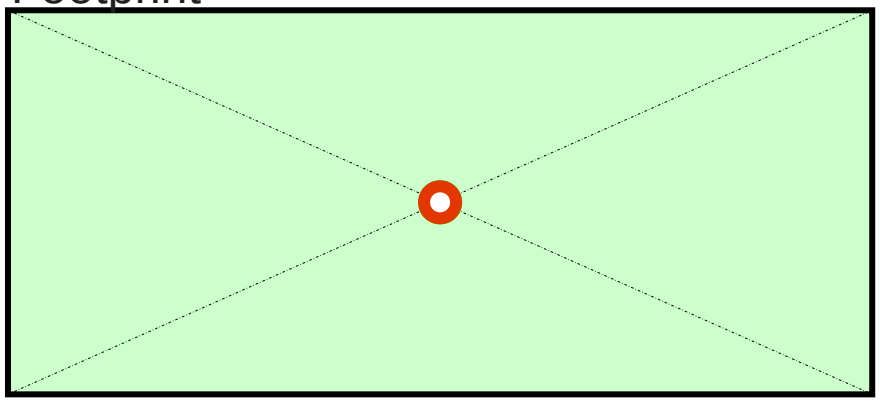
avec $\gamma = \frac{D_2}{D_1 + D_2}$

D_1 : distance entre la projection du CdG *et le centre le l'empreinte*
D_2 : distance entre la projection du CdG et l'arrète la plus proche du polygone



Stabilité

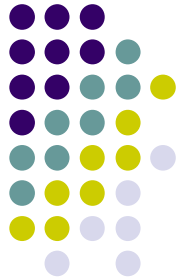
Footprint

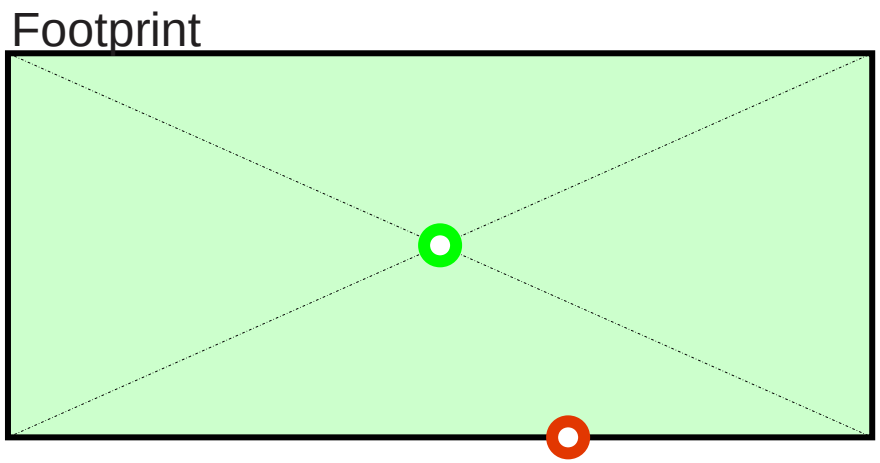


$$\gamma = \frac{D_2}{D_1 + D_2} = 1$$

$$\Delta q_i = -\gamma.\alpha_i^{Target}.\frac{d\varepsilon_{Target}}{dq_i} - (1-\gamma).\alpha_i^{COM}.\frac{d\varepsilon_{COM}}{dq_i} = \alpha_i^{Target}.\frac{d\varepsilon_{Target}}{dq_i}$$

Stabilité

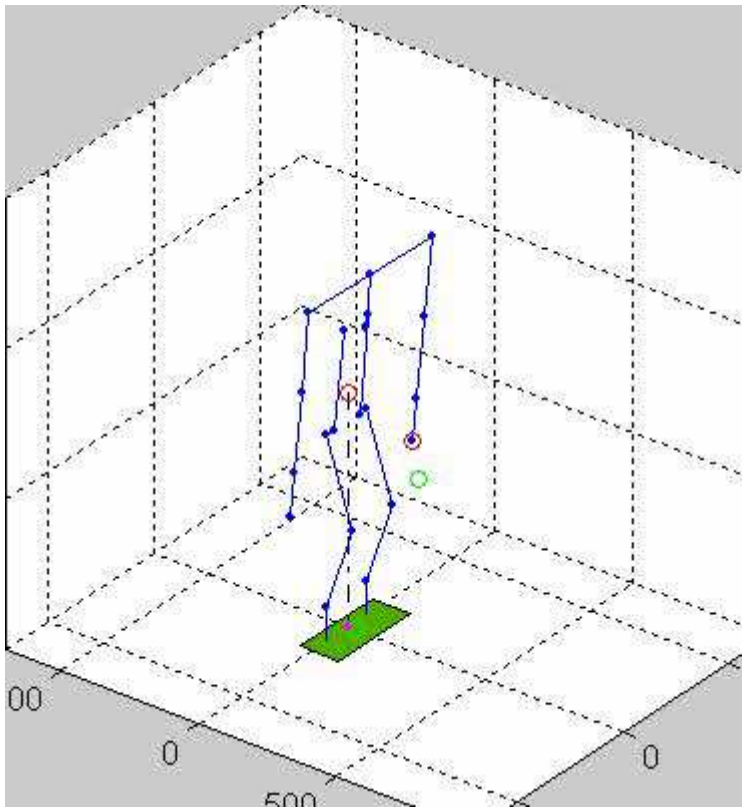


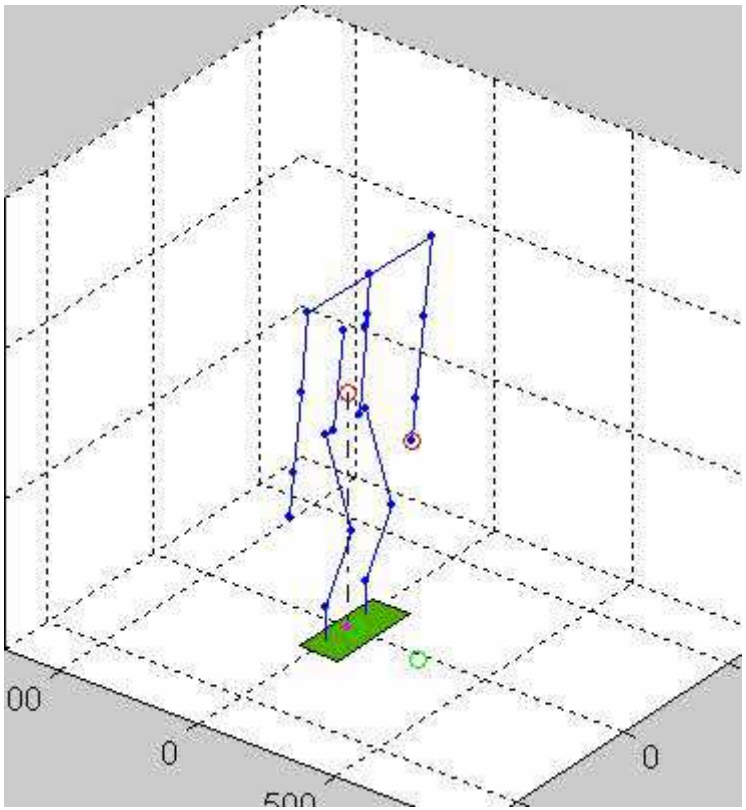


$$\gamma = \frac{D_2}{D_1 + D_2} = 0$$

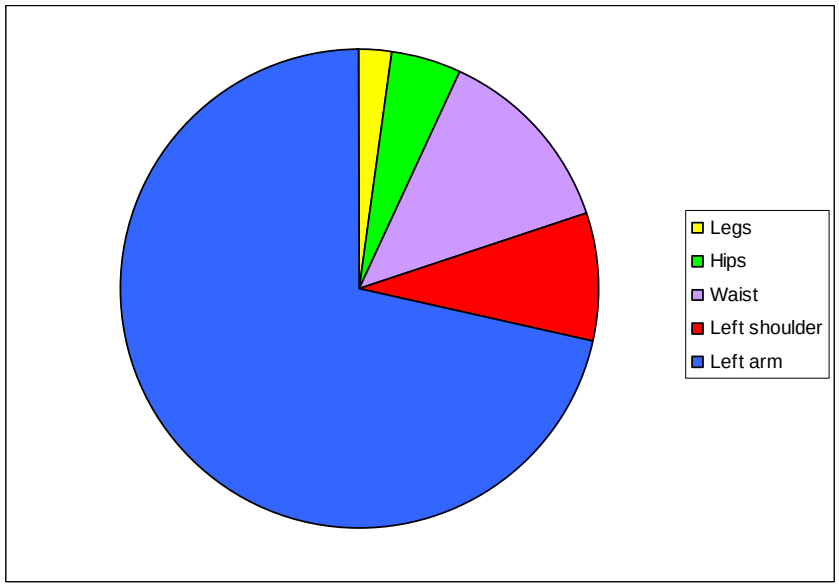
$$\Delta q_i = -\gamma.\alpha_i^{Target}.\frac{d\varepsilon_{Target}}{dq_i} - (1-\gamma).\alpha_i^{COM}.\frac{d\varepsilon_{COM}}{dq_i} = \alpha_i^{COM}.\frac{d\varepsilon_{COM}}{dq_i}$$

Stabilité

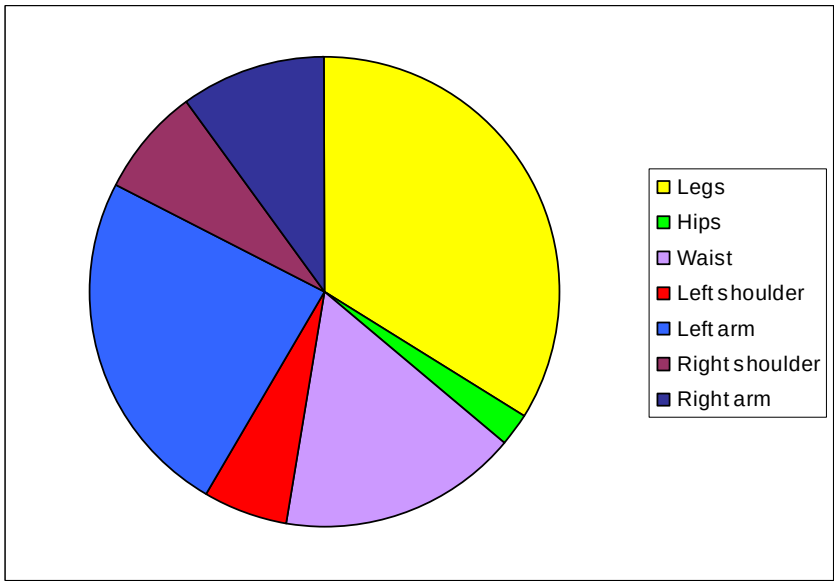




Résultats

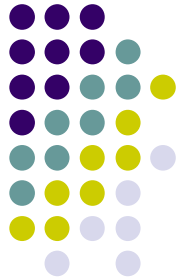


Ralliement de cible



Stabilité

Conclusion



- Avantages
  - Architecture distribuée
  - Robots redondants
  - Tolérante aux pannes
  - Temps de calcul
- Inconvénients
  - Existence de minima locaux
  - Réglage des coefficients

Perspectives

- Gestion de la vitesse de déplacement
- Suivi de trajectoire
- Implémentation
  - Robot humanoïde
  - Gestion des collisions
  - Robotique reconfigurable