IPSA
Institut Polytechnique des Sciences Avancées

PROJET

TMo422d

Année Universitaire 2025-2026

Sujet :

Navigation de flotte de robot via Consensus

Date de rendu

Réalisé par

CHERNYSHEV Tikhon
LIGNAC Simon
MBOCK Steeve
PERRET-BARDOU Guilhem

Professeur

Mme. TABTI Nahla

Table des matières

Introduction

Le contrôle de flottes de robots autonomes est un domaine de la robotique en plein essor, avec des applications allant de la logistique entrepôt à l'exploration spatiale, en passant par la surveillance environnementale. L'un des défis centraux de ce domaine est la coordination décentralisée : comment permettre à un ensemble d'agents d'atteindre un objectif commun en n'échangeant que des informations locales avec leurs voisins immédiats ?
Ce rapport présente les travaux réalisés dans le cadre du module *Autonomous Navigation of a Robot Fleet via Consensus*. L'objectif principal est d'implémenter et d'analyser différentes stratégies de communication et de commande pour une flotte de robots à dynamique de simple intégrateur en deux dimensions.
Nous abordons dans un premier temps l'algorithme de **consensus en position**, qui permet à la flotte de converger vers un point commun sans qu'aucune consigne globale ne soit imposée, puis la notion de robot **leader** guidant la flotte vers une destination prescrite. Nous étudions ensuite le paradigme **leader-suiveur** avec maintien de formation, avant d'intégrer l'**évitement d'obstacles** dans une mission complète validée sur Robotarium.

1 Consensus

L'algorithme de consensus repose sur une règle locale : chaque robot calcule sa commande en fonction de l'écart entre sa propre position et celles de ses voisins. La loi de commande du robot i s'écrit :

$$\mathbf{u}_i = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \left(\mathbf{x}_j - \mathbf{x}_i \right), \qquad \alpha_{ij} = k_p \, a_{ij}, \tag{1.1}$$

où $\mathcal{N}_i$ est l'ensemble des voisins du robot i, $\mathbf{x}_i \in \mathbb{R}^2$ sa position, et $k_p > 0$ un gain proportionnel. Si le graphe est connexe, tous les robots convergent asymptotiquement vers le barycentre pondéré des positions initiales.

I – Graphe de liaison consensus

Le graphe de communication est un **graphe complet** sur six nœuds. La matrice d'adjacence vérifie $a_{ij} = 1$ si $i \neq j$, 0 sinon. La Laplacienne $L = D - A$ admet une valeur propre nulle simple associée au vecteur $\mathbf{1}$, garantissant la convergence vers le barycentre.

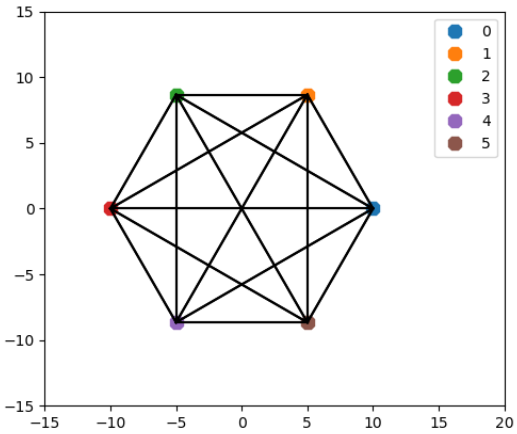


FIGURE 1.1 – Graphe de communication complet pour la flotte de 6 robots.

L'avantage de cette topologie est sa **redondance maximale**. En contrepartie, le nombre de canaux croît comme $\binom{n}{2}$, ce qui le rend peu scalable.

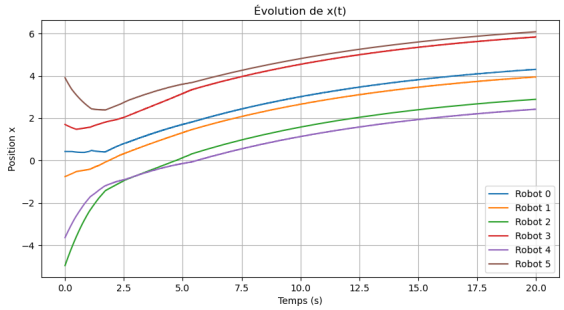


FIGURE 1.2 – Évolution de $x_i(t)$ de chaque robot.

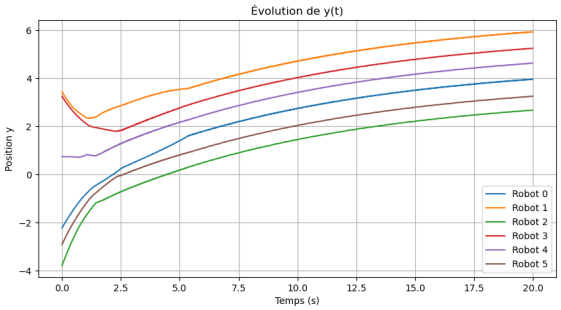


FIGURE 1.3 – Évolution de $y_i(t)$ de chaque robot.

Toutes les trajectoires convergent vers le barycentre des conditions initiales. Le croisement apparent des robots 2 et 4 selon x ne constitue pas une collision : leurs positions selon y restent distinctes à cet instant.

III – Influence de la topologie du graphe

III.a – Graphe en anneau

Dans le **graphe en anneau**, chaque robot i est connecté uniquement à $i-1$ et $i+1$ (modulo 6). Le nombre de liaisons passe de 15 à 6 :

$$a_{ij} = 1 \iff j = (i \pm 1) \bmod 6. \tag{1.2}$$

La convergence est **plus lente** : λ_2 de la Laplacienne est plus petit. En revanche, la complexité de communication est linéaire en n, ce qui rend cette topologie bien plus adaptée aux grandes flottes.

III.b – Suppression de liaisons et risque de collision

La suppression d'une liaison prive deux robots de leur canal direct ; celle d'une deuxième peut rompre la connexité et rendre le consensus global impossible, avec des collisions probables.
Ce résultat illustre un compromis fondamental : réduire les liaisons diminue le coût de communication mais dégrade la robustesse. Pour la suite, on conserve l'**anneau complet**.

IV – Ajout d'un robot Leader

Pour guider la flotte vers une destination (x^r, y^r), le robot 0 (leader) utilise :

$$\mathbf{u}_0 = \sum_{j \in \mathcal{N}_0} k_p \, a_{0j} (\mathbf{x}_j - \mathbf{x}_0) + k_t (\mathbf{x}^r - \mathbf{x}_0) \, . \tag{1.3}$$

Les suiveurs conservent (1.1). On prend $(x^r, y^r) = (2, 2)$.
Deux types de consigne ont été comparés :

- **Position** : terme proportionnel à l'erreur ; simple mais génère une vitesse variable qui perturbe les suiveurs.
- **Vitesse** : vitesse constante imposée ; formation plus stable mais nécessite une compatibilité avec la dynamique des suiveurs.

2 Leader-Follower

I - Modélisation et structure de données

Chaque robot est modélisé par $\dot{x} = u$ avec $x, u \in \mathbb{R}^2$. La classe `Robot` encapsule l'état, le contrôle et la méthode d'intégration `update()`. La classe `Fleet` gère les $N = 6$ robots et leur intégration via `integrateMotion(dt)`. Initialisation aléatoire : $x_i(0) \sim \mathcal{U}([-5, 5]^2)$.

II - Lois de commande

La commande totale combine trois contributions :

$$u_r = u_r^{\text{consensus}} + u_r^{\text{ref}} + u_r^{\text{avoid}},$$

avec :

$$u_r^{\text{consensus}} = k_p \sum_{n \neq r} (x_n - x_r) \ (d_{rn} > d_{\text{seuil}}), \quad u_r^{\text{ref}} = k_r (x_{\text{ref}} - x_r), \quad u_r^{\text{avoid}} = k_{\text{avoid}} \frac{x_r - x_n}{d_{rn}} \ (d_{rn} < d_{\text{safe}}).$$

Cette décomposition s'est révélée difficile à équilibrer simultanément, motivant l'évolution vers une architecture leader-suiveur pure.

III - Trajectoire circulaire sous Robotarium

Le code a été adapté pour Robotarium. Formation en croix, $d = 0{,}3$ m, trajectoire circulaire $R = 0{,}3$ m, $\omega = 0{,}25$ rad/s :

$$x_{\text{ref}}(t) = x_0 + R\cos(\omega t) - R, \qquad y_{\text{ref}}(t) = y_0 + R\sin(\omega t).$$

Saturation $v_{\max} = 0{,}15$ m/s et *barrier certificate* pour l'évitement :

```
dxi[:, 0] = kL * (x_ref - xi[:, 0])
dxi[:, i] = kF * (xi[:, 0] + rRef[i] - xi[:, i])
dxi = si_barrier_cert(dxi, xi)
dxu = si_to_uni_dyn(dxi, poses)
```

Résultats (15 000 itérations) — La flotte converge et maintient sa formation tout au long de la trajectoire :

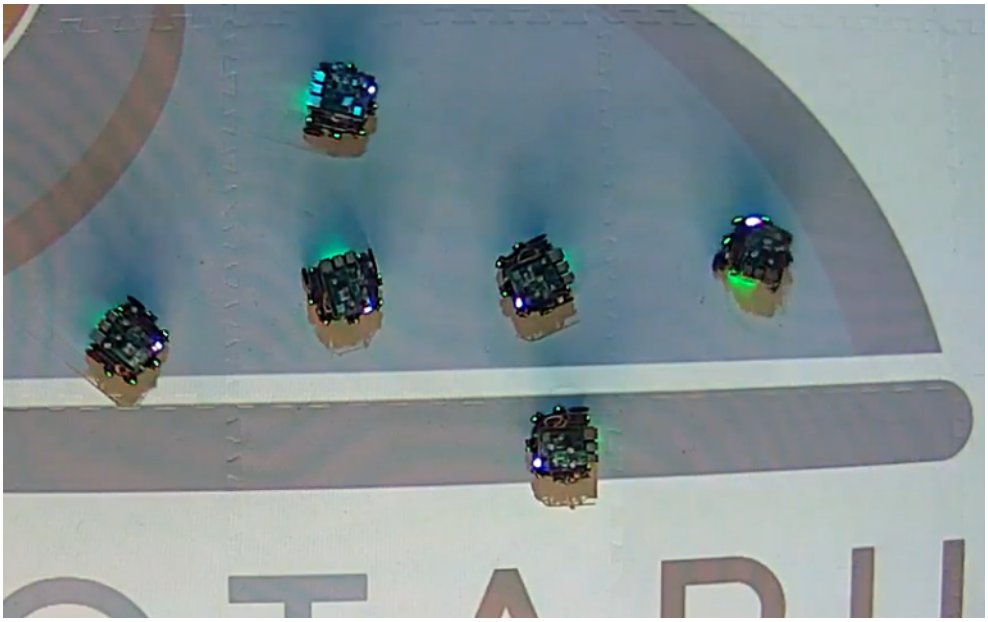


FIGURE 2.1 – Positions initiales de la flotte avant la mise en formation.

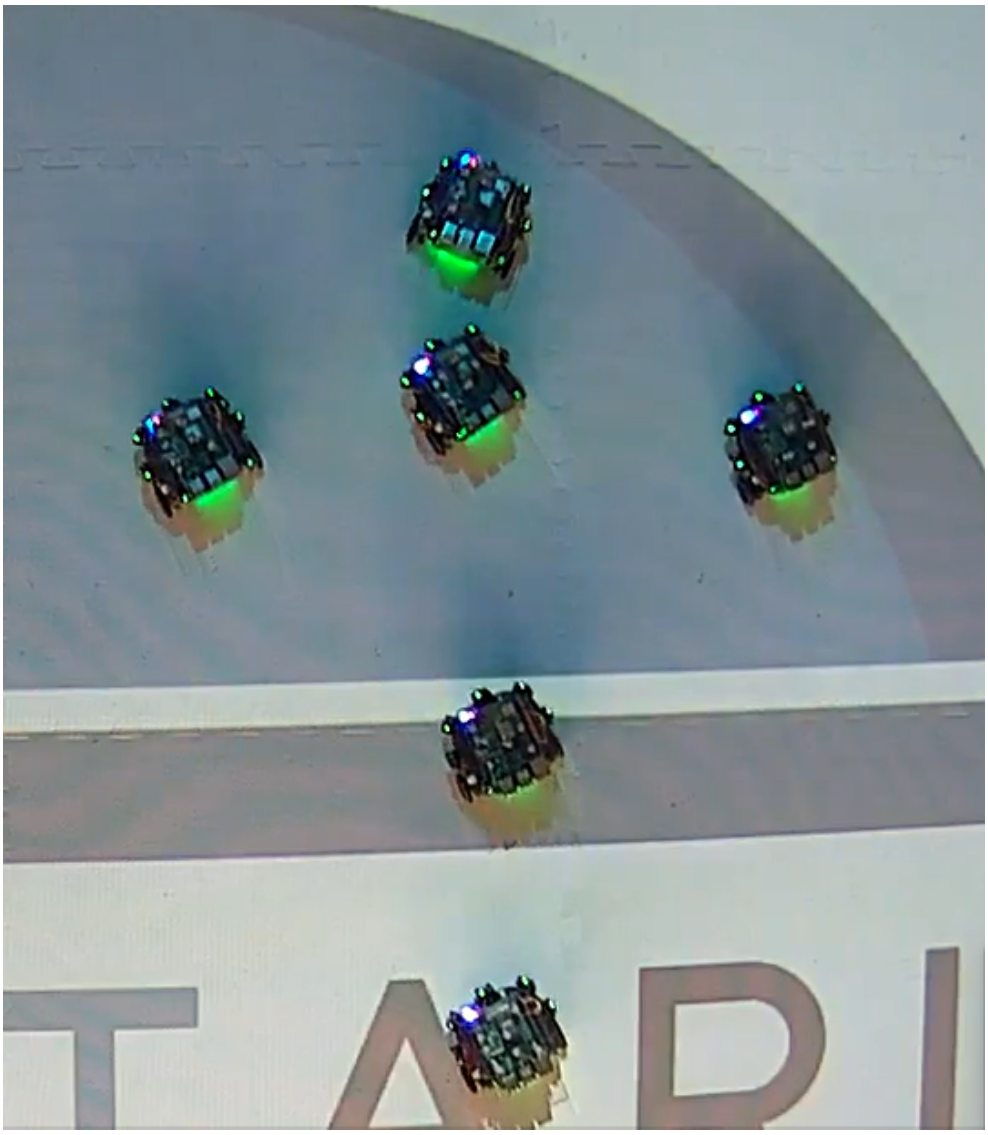


FIGURE 2.2 – Formation en croix atteinte par la flotte.

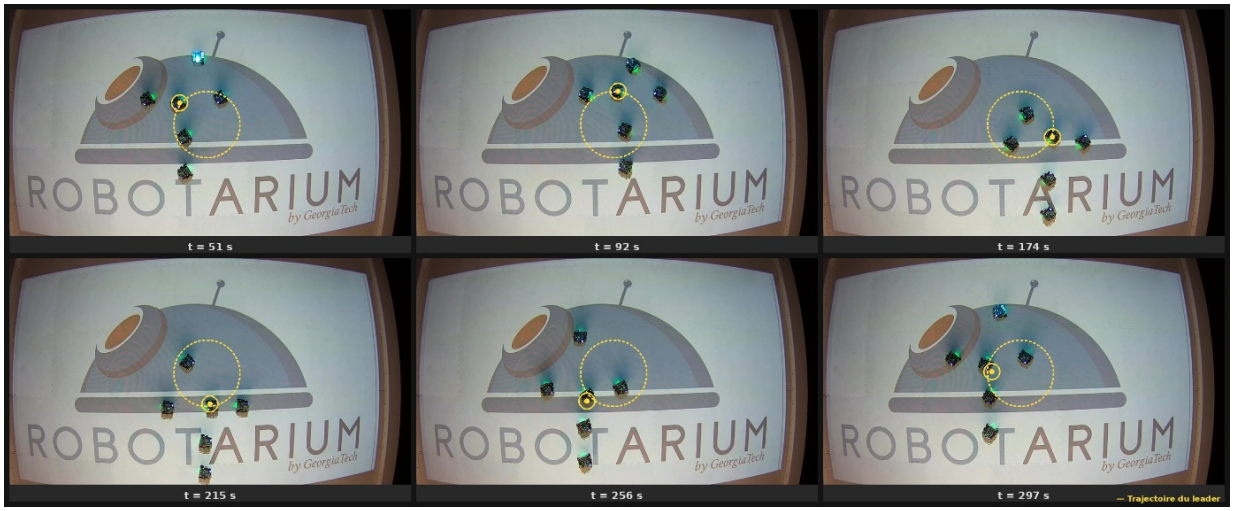


FIGURE 2.3 – Trajectoire circulaire du leader et suivi de formation des autres robots.

Le robot 1 (au-dessus du leader) présente un léger décrochage lors des phases de forte accélération, dû à la saturation de vitesse à 0,15 m/s.

3 Intégration : mission complète avec obstacles

I - Évitement d'obstacles rectangulaires

La fonction `compute_avoidance_ctrl` calcule une force d'évitement en trois étapes :
Point le plus proche.

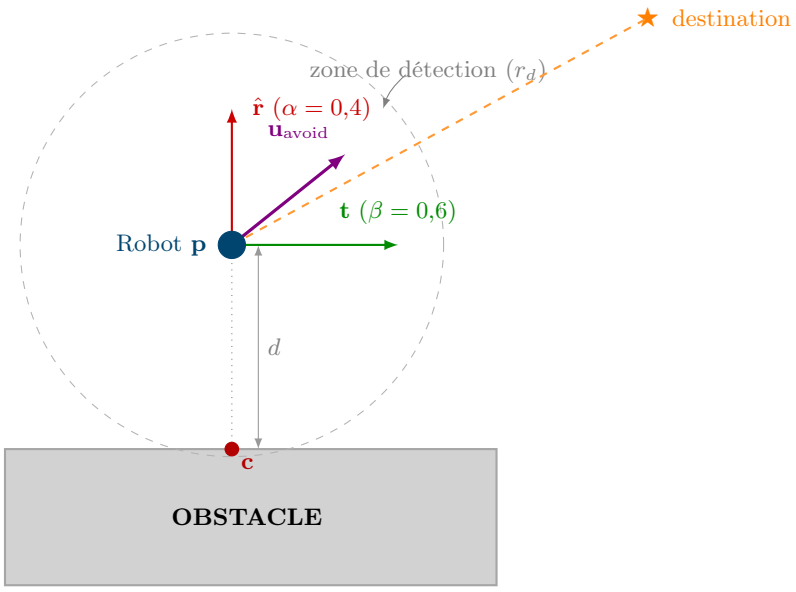
$$\mathbf{c} = \big(\mathrm{clip}(p_x, r_x, r_x{+}w),\ \mathrm{clip}(p_y, r_y, r_y{+}h)\big)^\top. \tag{3.1}$$

Zone de détection. La force est activée si $d = \|\mathbf{p} - \mathbf{c}\| < r_d = 2{,}0$ m.
Force combinée.

$$\hat{\mathbf{r}} = \frac{\mathbf{p} - \mathbf{c}}{d + \varepsilon}, \qquad \lambda = k_{\text{avoid}} \left(\frac{1}{d} - \frac{1}{r_d} \right), \qquad \boxed{\mathbf{u}_{\text{avoid}} = \lambda(\alpha\, \hat{\mathbf{r}} + \beta\, \mathbf{t})}, \tag{3.2}$$

avec $\alpha = 0{,}4$, $\beta = 0{,}6$, et $\mathbf{t}$ la tangente orientée vers la destination.

Paramètre	Valeur	Rôle
r_d	2,0	Rayon de détection
k_{avoid}	1,5	Gain global
α	0,4	Poids répulsion
β	0,6	Poids tangente




```python
def compute_avoidance_ctrl(robot_state, rects, detection_radius, k_avoid=1.5):
    ctrl = np.zeros((2, 1))
    pos = robot_state.flatten()
    for (rx, ry, w, h) in rects:
        obs = closest_point_on_rect(pos[0], pos[1], rx, ry, w, h)
        d = float(np.linalg.norm(pos - obs))
        if d < detection_radius:
            repulse = (pos - obs) / (d + 1e-6)
            tangent_cw  = np.array([ repulse[1], -repulse[0]])
            tangent_ccw = np.array([-repulse[1],  repulse[0]])
            intensity = k_avoid * (1.0 / d - 1.0 / detection_radius)
            intensity = max(intensity, 0.0)
            goal_dir = np.array([8.0, 8.0]) - pos
            tangent = (tangent_ccw
```

```
                        if np.dot(tangent_ccw, goal_dir) >= np.dot(tangent_cw, goal_dir)
                        else tangent_cw)
            alpha, beta = 0.4, 0.6
            ctrl += intensity * (alpha * repulse + beta * tangent).reshape(2, 1)
    return ctrl
```

Conclusion

Ce projet nous a permis d'explorer de façon progressive les problématiques fondamentales de la navigation décentralisée de flottes de robots autonomes.
L'étude du **consensus en position** a mis en évidence le rôle central de la topologie du graphe. Un graphe complet garantit une convergence rapide et robuste, tandis qu'un anneau est plus économe mais plus fragile face aux suppressions de liens. La valeur propre λ_2 de la Laplacienne quantifie cette robustesse : plus elle est faible, plus la convergence est lente et le système fragile.
L'introduction d'un robot **leader** a permis de guider la flotte vers une destination prescrite, avec un choix à faire entre consigne en position (simple mais perturbatrice) et consigne en vitesse (plus stable mais plus contraignante).
Le paradigme **leader-suiveur** a été validé sur Robotarium pour une formation en croix suivant une trajectoire circulaire. Les résultats confirment l'efficacité de l'approche, tout en révélant la limite de la saturation de vitesse lors des phases d'accélération du leader.
Enfin, l'**évitement d'obstacles** par champ potentiel répulsif mixte — répulsion radiale et contournement tangentiel ($\alpha = 0{,}4$, $\beta = 0{,}6$) — a permis la navigation en environnement contraint. La mission finale intègre l'ensemble de ces briques : consensus initial, transition vers le mode leader-suiveur, navigation multi-waypoints et maintien de formation, le tout en présence d'obstacles.
Ce projet constitue une introduction concrète et complète aux méthodes de contrôle décentralisé pour systèmes multi-agents, depuis la théorie des graphes jusqu'à la validation sur simulateur.