Commande de micromanipulateurs plans par apprentissage par renforcement

Guillaume Laurent, Emmanuel Piat
Laboratoire d'Automatique de Besançon - UMR CNRS 6596
25, rue Alain Savary, 25000 Besançon, France
E-mail : (glaurent,epiat)@ens2m.fr, website : http://www.lab.ens2m.fr

Résumé - *Cet article propose une application de l'apprentissage par renforcement à un problème de micro-manipulation par poussées. Nous disposons d'une plate-forme de manipulation capable de pousser des objets de taille millimétrique sur une plaque de verre sous une caméra CCD. L'objectif est d'automatiser les manipulations en réalisant des tâches de haut niveau. Notre approche est basée sur l'apprentissage par renforcement (Q-Learning) car les modèles de comportement du manipulateur et de la dynamique des objets ne sont pas connus à priori. Le système à commander ayant une complexité trop importante pour un algorithme classique, nous proposons une architecture originale qui réalise plusieurs apprentissages en parallèle. Cette méthode permet de générer une stratégie de commande quasi optimale quel que soit le nombre d'objets manipulés. Des simulations nous ont permis d'optimiser chaque paramètre de l'apprentissage. En particulier, elles ont montré que plus il y a d'objets, plus le contrôleur apprend vite. Les tests expérimentaux montrent qu'après apprentissage, le contrôleur remplit parfaitement sa fonction.*

Mots clé - apprentissage par renforcement, Q-Learning, micro-manipulation plane.

1 Introduction

Actuellement, l'assemblage de micro et de nano systèmes est en développement. Pour les positionnements plans, pousser les objets avec un manipulateur est aussi flexible mais mécaniquement moins complexe que le "pick-and-place" [1]. Un outil de préhension spécifique n'est pas nécessaire. Un micro-manipulateur deux axes comme ceux de Wolfgang [2] et Arai [3], ou un microscope à force atomique [4, 5] pour les nano-manipulations suffisent.

Notre objectif est d'automatiser le pilotage d'une plateforme de micro-manipulation deux axes actuellement en développement au laboratoire (cf. figure 2). Ce micro-manipulateur sera capable de pousser des micro-objets comme des cellules biologiques (environ 10 μm) sur une lame de verre placée sous une microscope. Le principe d'actionnement utilise des champs magnétiques pour déplacer indirectement un micro-outil dans la solution biologique [6].

Le manipulateur utilisé pour les expérimentations de cet article est capable de pousser des objets de tailles millimétriques. Ce manipulateur est seulement un banc d'essais pour notre contrôleur. Ensuite, les résultats seront transposés au micro-manipulateur. Dans cet article, notre objectif est de créer un contrôleur capable de déplacer les objets d'un point à un autre en les poussant avec l'outil magnétique.

La dynamique du block-pushing n'est pas triviale et sa modélisation demeure difficile. De plus, les comportements du micro-manipulateur magnétique et des micro-objets manipulés sont complexes : forces d'adhésion, friction, ... La modélisation de ces phénomènes particuliers liés à l'échelle microscopique est encore plus difficile. Il serait donc intéressant d'utiliser un contrôleur qui ne nécessite pas de modèle. D'autre part, le manipulateur doit pouvoir gérer les déplacements successifs d'objets en optimisant une fonction globale comme le temps total de manipulation de N objets.

Les algorithmes d'apprentissage par renforcement développés dans les années 80 par Sutton [7] et Watkins [8] offrent des caractéristiques intéressantes pour la commande de notre manipulateur. Tout d'abord, ils permettent d'approcher la commande optimale sans connaître le modèle du système. Les modèles du block-pushing, du micro-manipulateur, et des micro-objets manipulés n'ont pas à être connus à priori. L'apprentissage par renforcement fournit un moyen de programmation par récompenses et punitions qui ne nécessite pas de spécifier comment une tâche doit être effectuée. De plus, dans certaines conditions, ils garantissent la convergence vers la commande optimale globale au sens de Bellman. Ce point est particulièrement intéressant pour minimiser le temps total de manipulation.

Bien que l'apprentissage par renforcement s'adapte bien à tous les systèmes, c'est un processus d'apprentissage assez lent. La cause principale est la taille de l'espace d'état à visiter. Si la dimension du système d'état est grande, le temps d'apprentissage est très important et un apprentissage en ligne est impossible.

Pour commander la manipulation d'un objet, un algorithme classique comme le Q-Learning pourrait convenir. Cependant, le contrôleur doit être capable de gérer la présence de plusieurs objets en même temps. L'espace d'état de cette application est beaucoup plus grand et un algorithme traditionnel ne convient plus. Nous proposons donc une architecture originale afin de réduire cette complexité.

Cet article est composé de quatre parties. Les deux premières sont consacrées à la description du manipulateur et de l'architecture de notre contrôleur. La troisième présente les simulations effectuées pour valider ce principe. Enfin, la dernière partie expose les résultats expérimentaux obtenus avec la véritable plateforme de manipulation.

2 La plateforme de manipulation

La plate-forme de manipulation est constituée d'un outil ferro-magnétique déplacé par l'intermédiaire d'un aimant permanent placé sous une plaque de verre (cf. figure 1). L'outil est un cylindre d'acier de 5×5 mm. L'aimant, et par conséquent l'outil, ont deux degrés de liberté : haut/bas et droite/gauche. L'aimant est mû par deux moteurs électriques à courant continu. Il n'y a aucun système d'asservissement de la position de l'aimant. Le système est très difficile à commander : hystérésis entre l'aimant et l'outil, non-linéarités, ...

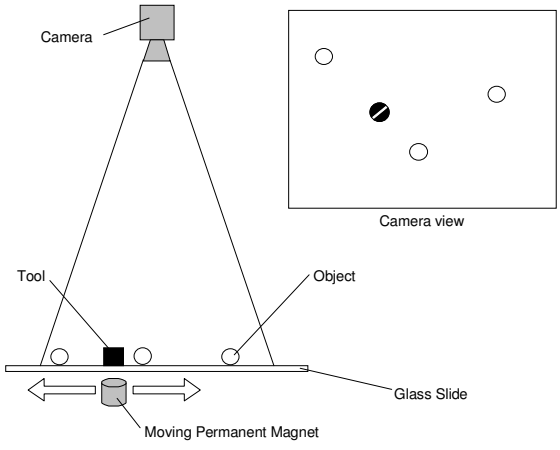

FIG. 1 – *Fonctionnement du manipulateur.*

Les objets à manipuler sont disposés sur la plaque de verre. Il s'agit pour cette expérience de billes de plastique de 3 mm. Au-dessus de la zone de manipulation, une caméra vidéo permet de visualiser entièrement la scène. La figure 2 montre une photographie de la plateforme de manipulation.

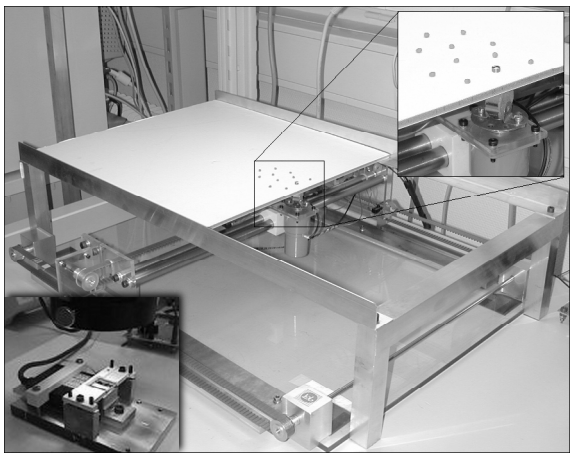

FIG. 2 – *Micro et milli-manipulateurs.*

3 Le contrôleur

3.1 Q-Learning

Le Q-Learning est un algorithme d'apprentissage par renforcement introduit en 1989 par Watkins [8]. Il est aussi décrit dans un livre de Sutton [9] et un article de Kaebling [10].

Le principe de l'apprentissage par renforcement est basé sur l'apprentissage par essais et erreurs. A chaque période d'échantillonnage, le contrôleur reçoit une information sur l'état s actuel du système. Il choisit alors une action a. Cette action change l'état du système et le contrôleur reçoit une récompense $r_{ss'}$ en fonction du nouvel état obtenu s'. Le but du contrôleur est de trouver une stratégie π, associant les actions aux états, qui maximise la somme des récompenses futures. La convergence de cet algorithme vers la stratégie optimale à été prouvée en 1992 par Watkins [11].

Le choix de l'action est basé sur l'expérience passée du contrôleur. Une fonction $Q(s,a)$ est utilisée pour mémoriser l'espérance de gain de l'action a dans l'état s. Cette fonction est appelée fonction d'évaluation. Dans notre cas, la fonction d'évaluation est un tableau à double entrée.

A chaque période d'échantillonnage, la fonction d'évaluation est remise à jour à l'aide de l'équation (1) :

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r_{ss'} + \gamma \max_b Q(s',b) - Q(s,a) \right] \quad (1)$$

avec : s l'ancien état du système, s' le nouvel état du système, a l'action choisie dans l'état s, α et γ des coefficients réels.

Pour choisir une action dans un état s, le contrôleur utilise la fonction d'évaluation Q et une méthode d'exploration/exploitation notée σ. L'action sélectionnée est déterminée par :

$$a = \sigma(s,Q)$$

Dans notre cas, σ est la méthode dite ϵ-gourmand : la plupart du temps, l'action gourmande est sélectionnée

(l'action pour laquelle $Q(s,a)$ est maximum) et avec une faible probabilité ϵ, l'action est choisie au hasard indépendamment des valeurs de $Q(s,a)$. Cette méthode permet de contrôler précisément le taux d'exploration. Afin de garantir une exploration constante, nous avons choisi $\epsilon = 0{,}1$. Ce paramètre peut être abaissé quand l'apprentissage est terminé.

La stratégie π, associant les actions aux états, est complètement définie par la fonction d'évaluation Q associée à la méthode d'exploration/exploitation σ (i.e. $\pi(s) = \sigma(s,Q)$).

L'algorithme du Q-Learning convient très bien pour tous les espaces d'état de petites dimensions. Si le contrôleur devait manipuler un seul objet, l'espace d'état aurait seulement 2 dimensions et un apprentissage serait possible. Mais, le contrôleur doit gérer plusieurs objets. Chaque objet se déplace dans un espace à deux dimensions. Si N objets sont pris en compte, l'espace d'état du système a en tout $2N$ dimensions. Un apprentissage est difficilement réalisable sur un espace de si grande dimension.

3.2 Le contrôleur

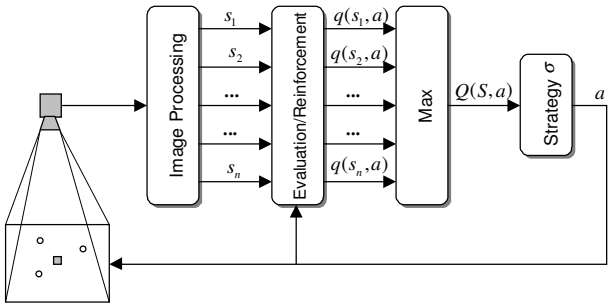

FIG. 3 – *Architecture du contrôleur*

L'hypothèse de base de notre approche est de supposer que tous les objets sont identiques. Chaque objet se déplace de la même manière sur le plan défini par la plaque de verre. Donc, pris indépendamment, chaque objet conduirait à la même fonction d'évaluation q. Pour cette raison, nous utilisons une seule fonction d'évaluation pour tous les objets. Plusieurs apprentissages sont exécutés en parallèle en utilisant le même tableau q.

Cette méthode permet de réduire la complexité du système. De plus, l'apprentissage pourrait être accéléré car un plus grand nombre de mises à jour est effectué. En règle générale, notre approche permet de réduire la complexité des espaces d'état où plusieurs variables évoluent dans le même espace vectoriel. Par exemple, dans notre application, chaque objet évolue dans le même espace vectoriel défini par le plan de la plaque de verre.

Chaque état s_i d'un objet i est défini par sa position dans le plan. A chaque période d'échantillonnage, la fonction d'évaluation est remise à jour pour chaque objet à l'aide de l'équation du Q-Learning :

$$q(s_i,a) \leftarrow q(s_i,a) + \alpha \left[r_{s_i s'_i} + \gamma \max_b q(s'_i,b) - q(s_i,a) \right]$$

Nous définissons ensuite la fonction d'évaluation globale pour l'état global $S = \{s_i, \forall i\}$ du système par :

$$Q(S,a) = \max_i q(s_i,a)$$

Ce point est discuté dans la section suivante. La stratégie σ utilise la fonction d'évaluation globale Q pour choisir l'action à effectuer.

L'architecture de l'algorithme est résumée par le schéma de la figure 3. Chaque objet est traité en parallèle en utilisant la même fonction q. A la fin, les informations sont regroupées pour évaluer la fonction d'évaluation globale. La figure 4 présente une synthèse de l'algorithme de notre contrôleur.

Initialiser l'état de départ $S = \{s_i\}$
Répéter
  $a \leftarrow \sigma(S,Q)$
  Effectuer l'action a
  Observer le nouvel état $S' = \{s'_i\}$
  Pour tous les objets i faire
    $a^* \leftarrow \arg\max_b q(s'_i,b)$
    $q(s_i,a) \leftarrow q(s_i,a) + \alpha[r_{s_i s'_i} + \gamma q(s'_i,a^*) - q(s_i,a)]$
  $S \leftarrow S'$

FIG. 4 – *Algorithme du contrôleur.*

3.3 Approche théorique

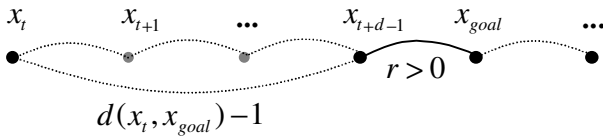

FIG. 5 – *Exemple de trajectoire dans l'espace d'état.*

Premier cas, l'espace d'état est $\mathcal{S}$ et toutes les récompenses sont nulles sauf la récompense $r > 0$ associée à la transition menant à l'état x_{goal} (cf. figure 5).

Si x_t est l'état du système à l'instant t, la fonction d'évaluation optimale est définie par :

$$q^*(x_t,a) = \max_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{x_{t+k} x_{t+k+1}} \right]$$

où $\max_\pi$ signifie que la stratégie qui maximise la somme est utilisée pour le calcul.

Le système atteindra forcément l'état x_{goal}, donc la fonction d'évaluation optimale peut s'écrire :

$$q^*(x_t,a) = \max_\pi \Big[\gamma^0 r_{x_t x_{t+1}} + \gamma^1 r_{x_{t+1} x_{t+2}} + ... \\ ... + \gamma^{d(x_t,x_{goal})-1} r_{x_{t+d-1} x_{goal}} \\ + \sum_{k=d(x_t,x_{goal})}^{\infty} \gamma^k r_{x_{t+k} x_{t+k+1}}\Big]$$

où $d(x_t,x_{goal})$ est le nombre de transitions pour aller de l'état x_t à l'état x_{goal}. Pour tous les états sauf x_{goal}, la récompense est nulle, donc :

$$q^*(x_t,a) = \max_\pi \Big[\gamma^{d(x_t,x_{goal})-1} r_{x_{t+d-1} x_{goal}} \\ + \gamma^{d(x_t,x_{goal})} \sum_{k=0}^{\infty} \gamma^k r_{x_{t+d+k} x_{t+d+k+1}}\Big]$$

En utilisant le principe de Bellman, nous obtenons :

$$q^*(x_t,a) = \gamma^{d^*(x_t,x_{goal})-1}\Big(r_{x_{t+d^*-1} x_{goal}} \\ + \gamma \max_\pi \sum_{k=0}^{\infty} \gamma^k r_{x_{t+d^*+k} x_{t+d^*+k+1}}\Big)$$

i.e. :

$$q^*(x_t,a) = \gamma^{d^*(x_t,x_{goal})-1}\big(r \\ + \gamma q^*(x_{goal},\pi^*(x_{goal}))\big) \quad (2)$$

où $d^*(x_t,x_{goal})$ est la valeur minimum de $d(x_t,x_{goal})$.

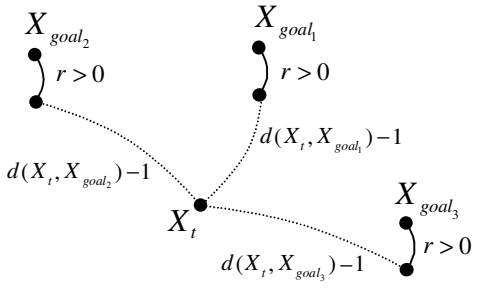

FIG. 6 – *Exemple avec trois objets.*

Second cas, l'espace d'état est $\mathcal{S}^N$ et plusieurs états buts donnent une récompense positive r (cf. figure 6). La nouvelle fonction d'évaluation est notée Q et un état dans $\mathcal{S}^N$ s'écrit :

$$X = (x_0,x_1,...,x_n) \mid \forall i, x_i \in \mathcal{S}$$

Les états buts sont définis par :

$$X_{goal_i} = (x_0,x_1,...,x_n) \mid \exists i, x_i = x_{goal}$$

Dans tous les cas, le contrôleur finira par atteindre un premier état X_{goal_i}, donc la fonction d'évaluation optimale peut s'écrire :

$$Q^*(X_t,a) = \max_i \Big[\gamma^{d^*(X_t,X_{goal_i})-1}\big(r \\ + \gamma Q^*(X_{goal_i},\pi^*(X_{goal_i}))\big)\Big]$$

$Q^*(X_{goal_i},\pi^*(X_{goal_i}))$ est difficile à évaluer car elle dépend de chaque état x_i. Pour continuer, nous supposons que :

$$Q^*(X_{goal_i},\pi^*(X_{goal_i})) = q^*(x_{goal},\pi^*(x_{goal}))$$

i.e. :

$$Q^*(X_t,a) = \max_i \Big[\gamma^{d^*(X_t,X_{goal_i})-1}\big(r \\ + \gamma q^*(x_{goal},\pi^*(x_{goal}))\big)\Big]$$

En utilisant l'équation (2), nous obtenons :

$$Q^*(X_t,a) = \max_i q^*(x_{i_t},a)$$

Cette hypothèse est forte. La stratégie d'un apprentissage utilisant cette méthode ne convergera pas vers la commande optimale globale. L'optimisation sera locale et calculée comme pour un état x_i isolé. Cependant, si les états x_i sont éloignés les uns des autres, l'hypothèse est pratiquement vraie.

4 Simulations

4.1 Modélisation

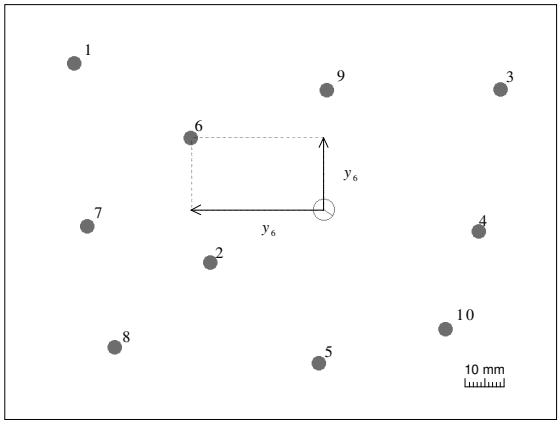

FIG. 7 – *Vue de la simulation.*

L'objectif de la simulation est de tester les performances de l'algorithme d'apprentissage. Les objets et l'outil sont modélisés par des cercles (cf. figure 7). Huit actions générant un mouvement de l'outil sont possibles : haut, bas, droite, gauche ainsi que les quatre diagonales. Toutes les dimensions et les vitesses sont réalistes. L'hystérésis entre l'aimant et l'outil est modélisée par une zone morte. Les interactions mécaniques entre les objets et l'outil sont aussi modélisées : si l'outil rencontre un objet, il le pousse.

Quand un objet touche un bord du plan, il est remis à une position au hasard. Chaque état s_i d'un objet est défini par ses coordonnées (x_i,y_i) calculées relativement au robot. Cette définition des états est importante pour réduire les données au strict nécessaire. Chaque objet peut se trouver dans 110 495 positions différentes.

Nous voulons que le manipulateur déplace tous les objets vers le bord droit du plan. Plus précisément, l'outil doit pousser successivement chaque objet vers la droite. Dans ce but, le contrôleur est récompensé quand un objet est déplacé vers la droite.

4.2 Résultats

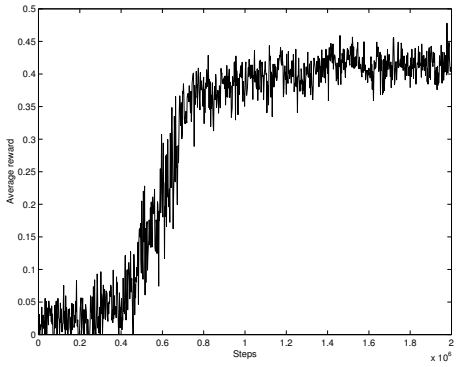

FIG. 8 – *Récompense moyenne en fonction du nombre de pas d'algorithme avec* $\alpha = 0{,}1$ *et* $\gamma = 0{,}5$.

Le contrôleur et l'algorithme se comportent bien. Au cours des simulations, nous calculons la récompense moyenne obtenue pour étudier le comportement de l'algorithme. Cette récompense moyenne est le rapport entre la somme des récompenses collectées et le nombre de boucles d'algorithme. Nous obtenons la courbe caractéristique décrite par la figure 8. Cette courbe montre l'évolution de la récompense moyenne au cours du temps. La récompense moyenne se stabilise vers une valeur limite qui détermine une mesure de la performance de l'apprentissage. Dans le cas de la figure 8, l'algorithme atteint une performance d'environ 0,4. Donc, plus d'une action sur trois est une bonne action de poussée.

Nous avons étudié l'influence des coefficients α et γ sur la performance de l'apprentissage. Les courbes 9 et 10 montrent les résultats des simulations avec dix objets. Elles représentent la récompense moyenne obtenue après convergence en fonction de α et γ. Ces deux courbes présentent chacune un optimum. La performance de l'algorithme est optimale avec $\alpha \simeq 0{,}1$ et $\gamma \simeq 0{,}5$. Nous choisirons ces valeurs pour les tests expérimentaux suivants.

La figure 11 montre que la vitesse de convergence dépend du nombre d'objets. Plus, il y a d'objets, plus la convergence est rapide. En effet, quand il y a plus d'objets, le contrôleur peut effectuer plusieurs remises à jour à chaque période d'échantillonnage, ce qui accélère l'apprentissage. Ce résultat est très intéressant

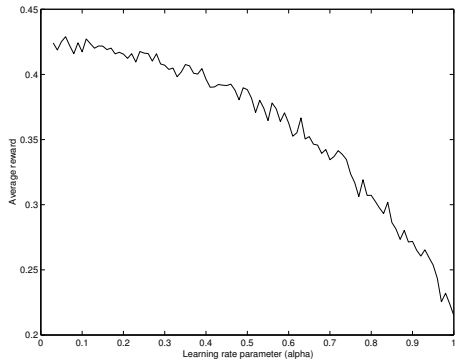

FIG. 9 – *Performance de l'apprentissage en fonction de* α *avec* $\gamma = 0{,}5$.

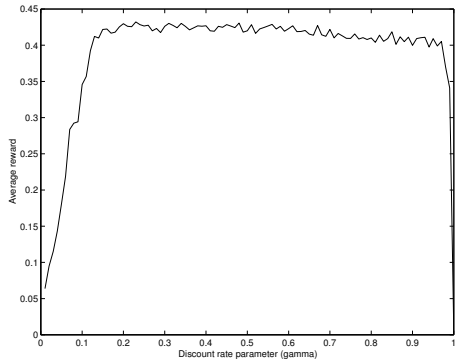

FIG. 10 – *Performance de l'apprentissage en fonction de* γ *avec* $\alpha = 0{,}1$.

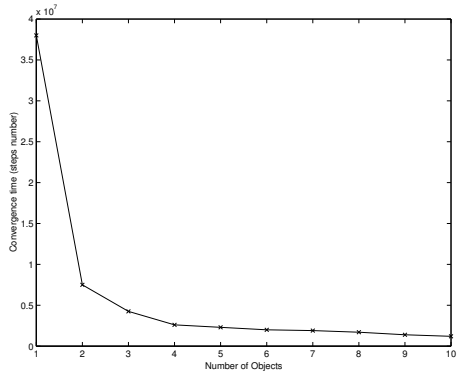

FIG. 11 – *Temps de convergence en fonction du nombre d'objets avec* $\alpha = 0{,}1$ *et* $\gamma = 0{,}5$.

pour réduire le temps d'apprentissage. Avec cette méthode, le nombre important d'objets devient un avantage pour le contrôleur.

5 Résultats Expérimentaux

Après apprentissage sur la simulation, le contrôleur est connecté à la véritable plate-forme de manipulation. Le contrôleur s'adapte à la réalité en peu de temps, car la stratégie générale reste la même. Il y a seulement quelques états à mettre à jour autour des objets.

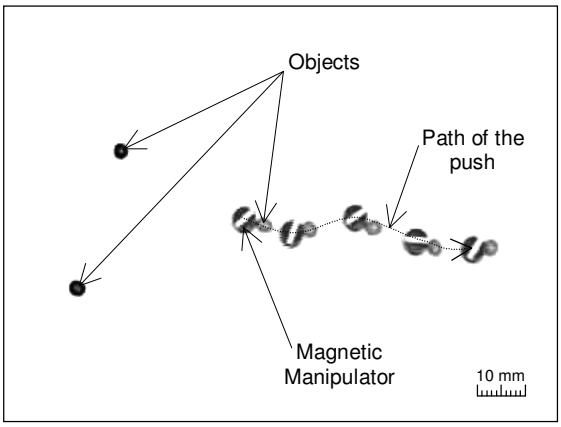

FIG. 12 – *Séquence vidéo de la poussée d'un premier objet*

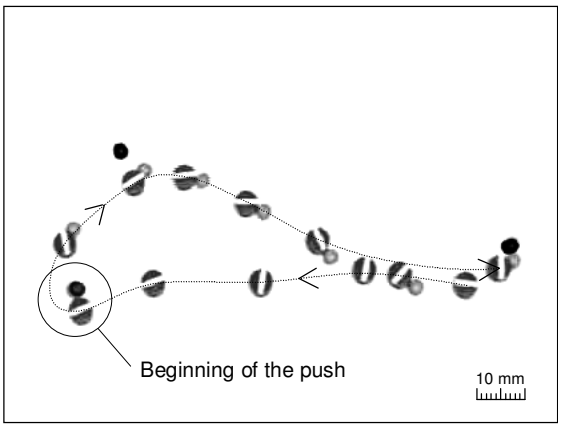

FIG. 13 – *Séquence vidéo de la recherche puis de la poussée d'un second objet*

Une séquence de manipulation est toujours la même, l'outil se dirige vers l'objet le plus proche, le contourne et le pousse en direction du bord droit du plan. La figure 12 montre une séquence où le manipulateur pousse un premier objet. Une fois cette tâche terminée, il entreprend la recherche d'un autre objet et le pousse, etc. (cf. figure 13).

Quand le manipulateur pousse un objet, il essaie de le faire sur une ligne droite car c'est la trajectoire de poussée optimale. De même, lors de la recherche d'un objet, le manipulateur contourne l'objet au plus près pour ne pas perdre de temps. Il emprunte un chemin quasi optimal. Donc, pour un seul objet, le contrôleur optimise les trajectoires de poussée.

Au point de vue global, le contrôleur commence par l'objet le plus proche. Ce choix n'est pas nécessairement optimal. Malgré tout, comme pour le problème du voyageur de commerce, ce choix semble être un bon heuristique dans la plupart des configurations.

6 Conclusion

Le contrôleur est basé sur une approche parallèle de l'algorithme du Q-Learning. Il doit remplir des tâches de déplacement par poussées à l'aide d'un manipulateur imparfait (hystérésis, non-linéarité, ...).

Des simulations nous ont permis de vérifier le comportement du contrôleur. D'un point de vue local, le contrôleur optimise parfaitement ses trajectoires. D'un point de vue global, le contrôleur choisit l'objet le plus proche. Ce comportement est presque optimal.

Cette architecture fait preuve de bonnes performances. L'intérêt majeur est que le contrôleur tourne à son avantage le nombre important d'objets à manipuler : grâce à cette multitude d'objets, il apprend plus vite. Cependant, la vitesse de convergence pourrait être augmentée en utilisant des algorithmes plus rapides comme le dyna-Q ou le priority sweeping.

Références

[1] Mason M.T. Mechanics and planning of manipulator pushing operations. *International Journal of Robotics Research*, 5(3):53–71, 1986.

[2] Wolfgang and Fearing Ronald S. Alignment of microparts using force controlled pushing. *SPIE Conf. on Microrobotics and Micromanipulation*, 3519:148–156, november 1998.

[3] Arai Fumihito, Ogawa Masanobu, and Fukuda Toshio. Indirect manipulation and bilateral control of the microbe by laser manipulated microtools. In *Proc. of the 2000 IEEE/RSJ Int. Conf. On Intelligent Robots and Systems*, 2000.

[4] Hansen Theil L., Kühle A., Sørensen A.H., Bohr J., and Lindelof P.E. A technique for positioning nanoparticles using an atomic force microscope. *Nanotechnology*, 9:337–342, 1998.

[5] Resch R., Lewis D., Meltzer S., Montoya N., Koel B.E., Madhukar A., Requicha A.A.G., and Will P. Manipulation of gold nanoparticles in liquid environnements using scanning force microscopy. *Ultramicroscopy*, 82:135–139, 2000.

[6] Michaël Gauthier, Emmanuel Piat, and Alain Boujault. Force study for micro-objects manipulation in an aqueous medium with a magnetic micromanipulator. *Submitted to Mecatronics 3rd European-Asian Congress*, October 2001.

[7] Sutton Richard S. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, MA, 1984.

[8] Watkins Christopher J.C.H. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

[9] Sutton Richard S. and Barto Andrew G. *Reinforcement Learning : An Introduction*. The MIT Press, 1998.

[10] Kaelbling Leslie Pack and Moore Andrew W. Reinforcement learning : A survey. *Artificial Intelligence Research*, 4:237–285, 1996.

[11] Watkins Christopher J.C.H. and Dayan Peter. Technical note : Q-learning. *Machine Learning*, 8:279–292, 1992.