Design of semi-decentralized control laws for distributed-air-jet micromanipulators by reinforcement learning

Laëtitia Matignon, Guillaume J. Laurent and Nadine Le Fort-Piat

Abstract— **Recently, a great deal of interest has been developed in learning in multi-agent systems to achieve decentralized control. Machine learning is a popular approach to find controllers that are tailored exactly to the system without any prior model. In this paper, we propose a semi-decentralized reinforcement learning control approach in order to position and convey an object on a contact-free MEMS-based distributed-manipulation system. The experimental results validate the semi-decentralized reinforcement learning method as a way to design control laws for such distributed systems.**

I. INTRODUCTION

In recent years there has been increased interest in decentralized approaches combined with machine learning techniques to solve complex real world problems [1], [2], [3]. A decentralized point of view offers several potential advantages in scalability and robustness, and can overcome the complexity constraints of conventional central control. Machine learning is a popular approach to find controllers that are tailored exactly to the system [4].

We propose to use reinforcement learning (RL) control techniques in a semi-decentralized perspective as a way to design control laws for distributed manipulation systems. Distributed manipulation systems are based on MEMS[1]-actuator arrays that can be used for positioning, conveying and sorting of small parts, or as bulk-fabricated (cheap), ultra-thin transport mechanisms, e.g. for paper in copy machines or printers. Distributed manipulation systems can be divided in two categories: contact systems and contact-free systems. Contact systems simulate cilia and can mainly perform high load capacity and accurate positioning [5], [6]. Contact-free systems use air-flow levitation and have several advantages including higher velocity and no friction problems but they require a greater level of complexity for the control [7], [8].

In this paper, we aim at controlling a distributed-air-jet MEMS-based micromanipulator designed by Chapuis et al. [7] as part of a research project funded by the NRA (French National Research Agency). This project federates five French research teams and a Japanese one from five laboratories[2]. The device to control is an array of micro-electro-valves able to produce controlled and directed micro-air-jets (cf. figure 1). The joint action of air-jets can achieve some positioning and conveyance tasks of a small part.

L. Matignon, G. J. Laurent and N. Le Fort-Piat are with FEMTO-ST/UFC-ENSMM-UTBM-CNRS, Université de Franche-Comté, Besançon, France, corresponding author: `guillaume.laurent@ens2m.fr`

[1]Micro Electro Mechanical Systems

[2]FEMTO-ST (Besançon, France), InESS (Strasbourg, France), LAAS (Toulouse, France), LIFC (Besançon, France) and LIMMS (Tokyo, Japan)

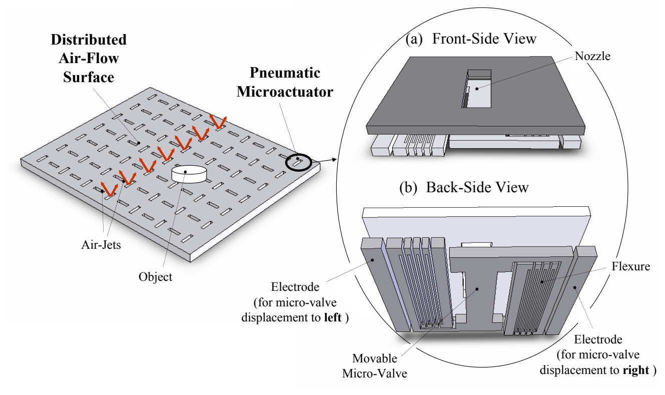

Distributed Air-Flow Surface
Pneumatic Microactuator
Air-Jets
Object
(a) Front-Side View
Nozzle
(b) Back-Side View
Flexure
Electrode (for micro-valve displacement to left)
Movable Micro-Valve
Electrode (for micro-valve displacement to right)

Fig. 1. Distributed-air-jet MEMS-based micromanipulator [7].

The long-term objective is to design and develop a fully integrated distributed micro-manipulation system, that we call *smart surface*, for conveying, fine positioning and sorting of parts at meso-scale (µm to mm). Fully integrated means that the control must be embedded. However, a fully integrated approach still remains rigid and costly in micro-scale fabrication for research works on control. Consequently, in this paper, we experiment our control algorithms with a model of the distributed-air-jet micromanipulator.

There are many challenging issues concerning the motion control and the increase of stability of the manipulated part. Finding coherent control laws for hundreds of independent air-jets is a complex problem. Fluidic effects are also very hard to control and actuators are not perfectly the same due to process dispersion factors. In the second part of this paper, we show that the open-loop control approach, usually proposed for contact distributed systems and called *programmable vector field* [9], [5], is not satisfactory for this purpose. That's why we investigate decentralized RL control techniques.

The proposed control architecture is based on RL in cooperative multi-agent systems, where multiple agents are cooperating to solve a joint task [10]. This framework allows to find local control laws which joint actions are optimal at the global level. Moreover, RL can find control laws without prior knowledge on the system. Thanks to this property, our approach could be applied to different distributed manipulation systems.

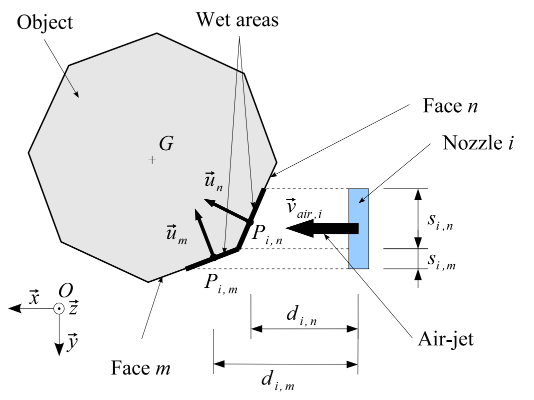


Fig. 2. Geometric notations in case of a nozzle close to the object, normal to $\vec{x}$ and orienting an air-jet with a velocity $\vec{v}_{air,i}$ (top view).

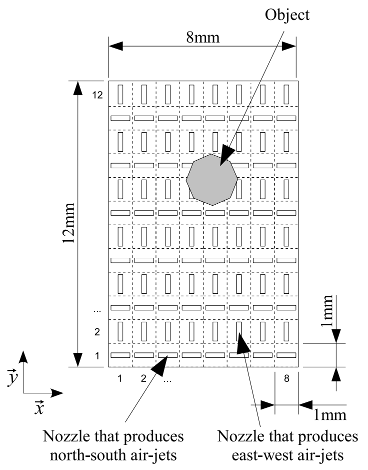

Object
8mm
12mm
12
...
2
1
1mm
1
2
...
8
1mm
$\vec{y}$
$\vec{x}$
Nozzle that produces north-south air-jets
Nozzle that produces east-west air-jets

Fig. 3. Geometry of the distributed-air-jet micromanipulator. The device has a total of 96 actuators (air-jet).

II. Distributed-air-jet micromanipulator

A. Real system

Figure 1 illustrates principles of the distributed-air-jet MEMS-based micromanipulator. It consists of an active surface based on an array of pneumatic micro-nozzles. Airflow comes through electrostatic micro-valves in the backside of the device. Each electrostatic micro-actuator works as a normally closed air-valve but has two opened positions, each generating opposite directed horizontal air-jet. When the valve is closed a slight vertical air-jet is produced to ensure levitation. In the front-side, the active surface is simply represented by holes where air-jets are produced. An object can be carried by actuating air-jets independently at each point. An overhead camera is used to get the position of the object's center. See [7] for more details about MEMS's design and working.

B. Model

The aim of this section is to state a realistic model of the distributed-air-jet micromanipulator. A multi-domain simulation of this distributed-parameter system is a self-challenging task. We make here some assumptions in order to keep only a finite set of differential equations based on fluidic forces exerted by an air-flow on a body.

1) Fluidic forces: Each nozzle is centered on a $1\,mm^2$ square. A nozzle generates :

- either a vertical air-jet $\vec{v}_{air} = v_{air}\vec{z}$ when the micro-valve is closed (the valve state a is 0),
- or an oriented air-jet $\vec{v}_{air} = v_{air}\vec{x}$ (nozzle normal to $\vec{x}$) or $\vec{v}_{air} = v_{air}\vec{y}$ (nozzle normal to $\vec{y}$) when the micro-valve is opened (the valve state a is ± 1).

By the combined action of vertical and oriented air-jets, forces and moments are exerted on the object. Two fluidic forces are defined : the levitation force F_L applied in the vertical direction $\vec{z}$ and the thrust force F_T which conveys the object on the surface $(\vec{x}, \vec{y})$. A torque also applies leading to the object's rotation around $\vec{z}$.

2) Hypotheses: For the conveyance model in two dimensions, we only model the thrust force F_T so the hypotheses are :

- the levitation force F_L is not evaluated; the object is supposed to be constantly in levitation,
- effects of vertical air-jets on the object's motion are neglected,
- we model only interactions between oriented air-jets and the object's edges. Oriented air-jets reaching the back side of the object are thus neglected,
- air-jets are independent, *i.e.* there aren't any interactions between air-jets.

3) Thrust force: To establish a model of the thrust force, we need some notations shown on figure 2. The object is represented by a N-faces polygon and is conveyed in $(\vec{x}, \vec{y})$. Its center of gravity is noted G, G's coordinates are noted (x, y). All the air-nozzles are numbered between 1 to M. When an air-nozzle is less than 5 mm away to the object and when the air-jet is in the good direction, the produced air-flow reaches a small area of the object's edge, called wet area. We assume that the wet area of the face is just the normal projection of the nozzle in $(O, \vec{y}, \vec{z})$. This relation between a face n and a nozzle i is described using the following variables:

- $\vec{u}_n$ is the normal vector of the face n,
- $s_{i,n}$ is the surface of the normal projection of the wet area of the face n,
- $P_{i,n}$ is the central point of the wet area of the face n,
- $d_{i,n}$ is the distance between $P_{i,n}$ and the nozzle i.

The elementary thrust force of the air-jet i on the face n is :

$$\vec{f}_{i,n} = \frac{1}{2}\rho C_x s_{i,n} v_{i,n}^2 \vec{u}_n \tag{1}$$

where C_x is the drag coefficient, ρ the density of the air and $v_{i,n}$ the relative speed of the air at the point $P_{i,n}$. Each elementary thrust force $\vec{f}_{i,n}$ is equal to 0 with a probability of 0.05 to imitate possible micro-actuator failures. The relative speed is given by :

$$\vec{v}_{i,n} = (\vec{v}_{air}(d_{i,n}, a_i) - \vec{v}_{obj}) \cdot \vec{u}_n \tag{2}$$

where $\vec{v}_{obj}$ is the speed of the object and $\vec{v}_{air}$ is the speed of the air-jet. The direction of $\vec{v}_{air}$ is normal to the direction of the nozzle. The magnitude of $\vec{v}_{air}$ results from a finite element simulation [7] that gives air-jet velocity decreasing with the distance d to the nozzle and according to the valve state a :

$$v_{air}(d,a) = \begin{cases} 0 & \text{if } a = 0 \\ 5500\,\mathrm{sgn}(a) e^{-\frac{d^2}{3.38}} & \text{if } a \neq 0 \text{ and } d \geq 0 \\ -1000\,\mathrm{sgn}(a) e^{-\frac{d^2}{0.08}} & \text{if } a \neq 0 \text{ and } d < 0 \end{cases} \tag{3}$$

We can notice a negative air-jet velocity if $d < 0$. Indeed, when a nozzle generates an air-jet oriented along $\vec{x}$, a slight opposite air-jet is produced along $-\vec{x}$. That leads an object moving slowly along $\vec{x}$ to bounce of opposite air-jets.

Then, the model sums all elementary thrust forces generated by the M air-jets on the N object edges according to the surface geometry (cf. figure 3). The resulting thrust force F_T is applied at its center of gravity G :

$$\vec{F}_T = \sum_{i=1}^{M} \sum_{n=1}^{N} \delta_{i,n} \vec{f}_{i,n} \tag{4}$$

where $\delta_{i,n}$ equals 1 if the air-jet i reaches the face n and zero else. $\delta_{i,n}$ is also zero if the air-nozzle is under the object (see hypotheses). The resulting torque is :

$$\vec{\mathcal{M}}_{F_T} = \sum_{i=1}^{M} \sum_{n=1}^{N} \delta_{i,n} (\vec{GP}_{i,n} \wedge \vec{f}_{i,n}). \tag{5}$$

4) Viscous force: Friction forces due to the surrounding air are spread over the surface of the object. The resulting force is the viscous force :

$$\vec{F}_V = -K\eta\vec{v} \tag{6}$$

where K is a geometric coefficient dependent on the shape of the object, η the coefficient of viscosity and $\vec{v}$ the speed of the object.

5) Conveyance model: The object's dynamic follows the equations :

$$\begin{cases} m\ddot{x} &= \vec{F}_T \cdot \vec{x} - K\eta\dot{x} \\ m\ddot{y} &= \vec{F}_T \cdot \vec{y} - K\eta\dot{y} \\ J\ddot{\theta} &= \vec{\mathcal{M}}_{F_T} \cdot \vec{z} \end{cases} \tag{7}$$

where m is the mass of the object, J its inertia moment and $\dot{\theta}$ its angular speed.

III. Open-loop control by programmable vector field

A. Control problem

In this section, we assume that no sensor is available (open-loop control). The problem is to address a direction to each air-jet in order to get a specific motion of the object.

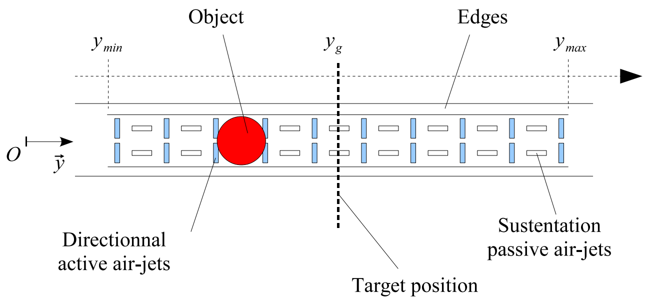


Fig. 4. Task description.

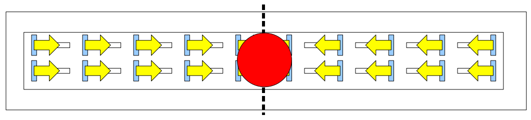


Fig. 5. Vector field.

The magnitude of an air-jet is not adjustable. For instance, we focus here on a stabilization task; a cylinder (2 mm in diameter, 0.25 mm in height) must be maintained at a given place of the surface. The cylinder is modeled by a 21-faces polygon.

We aim to regulate the position in only one direction. For that, we use a restricted area of the active surface. This restricted area contains 19 columns per 2 rows of nozzles (cf. figure 4). All east-west air-jets are assigned to a constant air-jet direction (east, middle or west). North-south air-jets stay in middle position to ensure object's levitation. For this simple task, we try to use the programmable vector field method to determine the direction of each air-jet.

B. Programmable vector field

Programmable vector field was introduced by Böhringer *et al.* [9], [5] to control contact-actuator arrays and transversely vibrating plates. Programmable vector field is sensor-less and may be employed to orient, sort, feed, assemble parts, etc.

The principle of programmable vector field is simple: actuators are assigned to a specific direction with regard to their position, then, when a part is placed on the device, the vector field induces a force and a torque upon it. Over time, the part may come to rest in a dynamic equilibrium state.

For the generation of manipulation plans with programmable vector fields, it is essential to be able to predict the motion of a part in the field and to determine the stable equilibrium poses a part can reach in which all forces and moments are balanced. To create a stable equilibrium point in the middle of the surface, we used the vector field shown by figure 5. The middle of the surface is located at $y_g = 9.5$ mm.

C. Experimental results

Figure 6 shows the position of the cylinder's center versus the time. Three short perturbations at time 0, 5 and 10 s are done to test the robustness of the regulation.

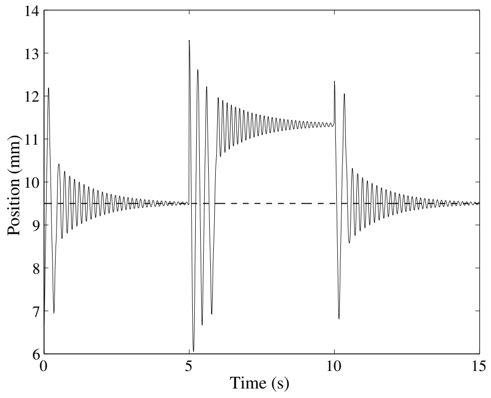


Fig. 6. Position of the cylinder's center according to the time using programmable vector field open-loop control (solid line), the dashed line represents the target position. Short perturbations are done at time steps 0, 5 and 10 s.

As we can see on the curve, the motion is very oscillatory and non-harmonic. Two oscillating phases can be distinguished. Moreover, in some cases due to a short perturbation, the object misses the target position and stays in another stable position (at time step 5 s). The reason is that slight counter-jets are produced in the opposite direction of the air-jets as described by equation 3. This phenomena was observed with the real system.

The first conclusion is that the vector field can create a stable equilibrium point. But the stable area is very small and the motion of the object is very oscillatory. The constant programmable vector field is acting as a simple on-off regulator with a high gain, so turning the velocity down might get rid of some of the oscillations and overshoots at the cost of slower convergence. This could be done by reducing the input pressure. However, as programmable vector field control is open-loop, it will not be able to reject perturbations.

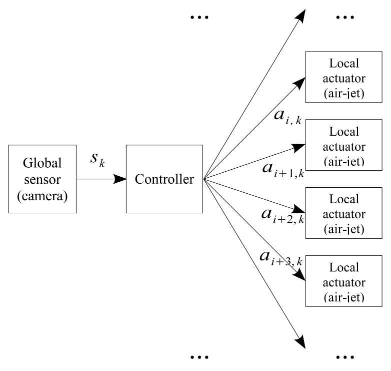
The second conclusion is that the control of the air-jet micromanipulator is not obvious. So, we get a strong benchmark to test various control approaches.

IV. Control by reinforcement learning

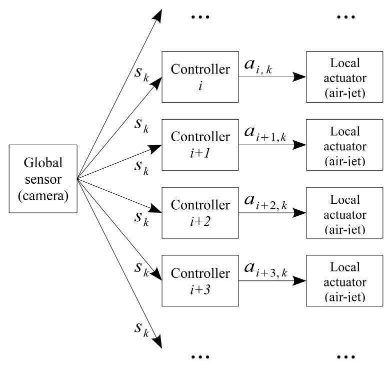
A. Control problem

In this section, we assume that the manipulation area is supervised by a global sensor like a camera. Thanks to this sensor, the object's position is known at intervals of time T (sampling period). The position at time step k is noted (x_k, y_k).

In order to reject perturbations or to control the object trajectory, a suitable combination of air-jets must be calculated at each sampling period in accordance with the position (x_k, y_k) of the part. The control signal (air-jet direction) sent to the ith air-jet is noted $a_{i,k}$ (that can take one of the three possible discrete values). To control the entire surface, there are 96 control signals to process at each step! A fully centralized control architecture is not suitable due to processing complexity and the number of communication channels required (cf. figure 7a).



(a) Centralized control



(b) Semi-decentralized control

Fig. 7. Control architectures.

Another solution is to use one controller per air-jet. The object's position is broadcasted to each independent controller so as to close the loop. Then, each controller sends a command to its associated air-jet. We call this architecture semi-decentralized because acting and decision-making are local but sensing is global (cf. figure 7b).

The control problem is to design local control laws that generate a satisfactory global behavior of the part. Under some assumptions, reinforcement learning techniques are able to find such controllers.

B. Reinforcement learning

RL methods are inspired by dynamic programming concepts and have been studied extensively in centralized framework [4]. A controller, also called agent, learns by interactions with its environment, using a scalar reward signal called reinforcement as performance feedback. The studies about reinforcement learning algorithms in multi-agent systems are based on Markov game framework.

Definition 1: A *cooperative Markov game*[3] is defined as a tuple $< m, S, A_1, ..., A_m, T, R >$ where : m is the number of agents; S is a finite set of states; A_i is the set of actions available to the agent i (and $A = A_1 \times ... \times A_m$ the joint action space); $T : S \times A \times S \rightarrow [0, 1]$ is a transition function

[3]also called *team game*.

that defines transition probabilities between states; $R : S \times A \rightarrow \Re$ is the reward function.

This framework is equivalent to the semi-decentralized architecture we presented because all agents have access to the complete observable state s. Reinforcement function is determined by the task to achieve (see below). The transition function T is unknown from agent's perspective (learning hypothesis).

C. Think globally, act locally

The objective of the group (or the global objective) is to find a joint policy $\boldsymbol{\pi}$ that maximizes for all states s in S and joint actions a in A the expected sum of the discounted rewards in the future,

$$Q^{\boldsymbol{\pi}}(s,a) = E^{\boldsymbol{\pi}}\left\{\sum_{j=0}^{\infty}\gamma^j r_{j+k+1}\big|s,a\right\} \tag{8}$$

$Q^{\boldsymbol{\pi}}(s,a)$ is called the joint or global action-value function.

In the multi-agent system framework, independent learners (ILs) were introduced in [11] as agents which don't know the actions taken by the other agents. The objective of an IL is then to find a local policy π_i that maximizes the expected sum of the discounted rewards in the future for *its own* action a_i in A_i,

$$Q_i^{\pi_i}(s,a_i) = E^{\pi_i}\left\{\sum_{j=0}^{\infty}\gamma^j r_{j+k+1}\big|s,a_i\right\} \tag{9}$$

$Q_i^{\pi_i}(s,a_i)$ is called the local action-value function. The IL approach brings the benefit that the size of the state-action space is independent of the number of agents. This choice is pertinent for the distributed-air-jet micromanipulator in order to avoid exponential growth of action space with the number of agents.

It is important to notice that it is necessary for each independent learner to find its local optimal action-value function, in order that the group achieves the global optimum [12].

D. Decentralized Q-Learning

Q-learning [13] is one of the most used RL algorithm in single-agent framework because of its simplicity and robustness. That's also why it was one of the first to be applied to multi-agent environments [14]. Despite some difficulties as the coordination or the loss of theoretical guarantees [15], it was successfully applied with ILs on some applications [16], [17], [18], [19].

For an IL i, Q-Learning consists in getting a more and more accurate estimation of the optimal local action-value function using a recursive updating equation, that is:

$$Q_i(s_{k-1},a_{i,k-1}) \leftarrow Q_i(s_{k-1},a_{i,k-1}) + \alpha\delta \tag{10}$$

where $\delta = r_k + \gamma \max_{b\in A_i} Q_i(s_k,b) - Q_i(s_{k-1},a_{i,k-1})$, $a_{i,k-1}$ is the individual action chosen by the agent i at time step $k-1$, $\alpha \in]0;1]$ is the learning rate and $\gamma \in [0;1[$ is the discount factor. $Q_i(s,a_i)$ is the current value of the state-action pair (s,a_i) for the agent i. $Q_i(s,a_i)$ values are stored in a $|S| \times |A_i|$ array.

We propose to combine decentralized Q-learning with eligibility traces to obtain a more efficient method. The eligibility trace $e(s,a)$ is a measurement of the age of the last visit of the state-action pair (s,a). Action-value function is then globally updated according to eligibility trace, that is:

$$Q_i(s_{k-1},a_{i,k-1}) \leftarrow Q_i(s_{k-1},a_{i,k-1}) + \alpha\delta e_i(s_{k-1},a_{i,k-1}) \tag{11}$$

Much more Q-values are then updated at each transition. This method is a decentralized version for ILs of the Watkins's $Q(\lambda)$ algorithm[4] [20].

At each time step, a new action $a_{i,k}$ is selected according to $Q_i(s_k,*)$ values and to an exploration/exploitation compromise. We use the ϵ-greedy action selection method[5].

V. Experimental results

A. Part stabilization

As illustrated on the figure 4, we aim to control columns of east-west nozzles to regulate the position of the part in only one direction like in the previous section. All the east-west nozzles of a column are controlled together according to the air-jet direction required. So the system requires as much controllers as east-west nozzles columns, i.e. 10 controllers. Possible actions of each controller are: directing the air-jet on the east or on the west or closing the valve, i.e. 3 actions (east, west, middle). So, the cardinal $|A_i|$ of the action space A_i of the agent i is 3.

The state of the system s_k at time step k is the object's current and previous positions, $s_k = (y_k, y_{k-1})$. To apply RL in the form presented previously, time axis and continuous state must be discretized. The sample time of our simulation is 0.01 seconds (between step the integration method is ODE45). For object's position, a 41×41 spatial tile-coding is used. So, this yields Q_i tables of size $41\times41\times3$ for each 10 controllers, to compare with a Q table of size $41\times41\times3^{10}$ in a centralized view.

According to [21] and in order to stabilize the object at the position y_g with null speed, the chosen reward function is :

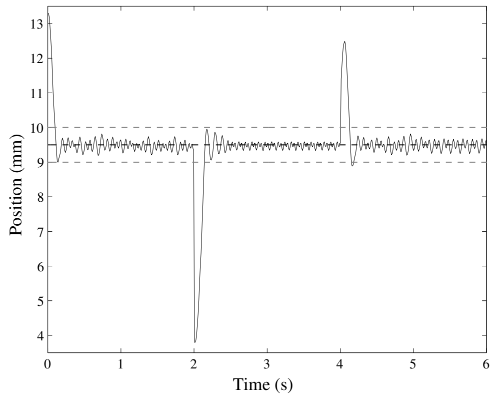
$$R(y_k,y_{k-1}) = \begin{cases} 1 & \text{if } (y_k,y_{k-1}) \in [y_g-\rho, y_g+\rho]^2 \\ -1 & \text{if } y_k < y_{min} \text{ or } y_k > y_{max} \\ 0 & \text{else} \end{cases} \tag{12}$$

where ρ sets a margin, y_{min} the minimal abscissa and y_{max} the maximal abscissa.

Independent controllers learn by decentralized Q-Learning during 300 trials. Each trial starts with the object in a random initial state ($y_0 \in [3,15]$ mm) and runs at the most 10

[4] λ is the decay parameter for eligibility traces.

[5] The probability of taking a random action for an agent i is ϵ and, otherwise, the selected action is the one with the largest Q_i-value in the current state.



Result with the simulated air-jet micromanipulator (short perturbations are done at time steps 0, 2 and 4 s)

Fig. 8. Position of the cylinder's center according to the time (solid line) after learning with the Q-Learning algorithm, the dashed line represents the target position. The control architecture is semi-decentralized.

seconds. A trial ends if the object gets out from the restricted area ($y_k < y_{min}$ or $y_k > y_{max}$). All trials use $\alpha = 0.1$, $\gamma = 0.9$, $\epsilon = 0.01$ and $\rho = 0.5\,\text{mm}$.

Figure 8 shows the position of the object's center according to the time after learning. It takes around 0.2 seconds for the ILs to regulate the object's position with an oscillation range of 0.1 mm. The perturbations at time steps 0, 2 and 4 s are quickly rejected.

Semi-decentralized RL manages to stabilize the object with a high damp factor and with a good robustness to perturbations. This result demonstrates the potential capacity of RL control to regulate position and speed of a levitating part on distributed-air-jet micromanipulator.

B. *Part conveying*

The objective is to convey the object to the middle of the north border of the surface illustrated on the figure 3. The object is fed to the surface at the initial position $(x_0, y_0) = (7.5, 1.5)\,mm$ and with a random speed ($\dot{x} \in [-50, -10]\,mm/s$). Open-loop control by programmable vector field is first applied to this task. The vector field shown by figure 9 is used to convey the object. The figure 10a shows the object's trajectories for various initial speeds of the object ($\dot{x} = \{-10, -20, -30, -40, -50\}\ mm/s$). As illustrated on the figure, this vector field is not able to convey the object to its target position for all initial speeds.

We propose to test if semi-decentralized reinforcement learning control is robust to the different initial speeds of the object. This time, 96 independent controllers learn by decentralized $Q(\lambda)$ during 1000 trials. Each trial starts with the object at the initial position and with a random speed and runs at the most 10 seconds. A trial ends if the object gets out from the surface. All trials use $\alpha = 0.1$, $\gamma = 0.9$, $\lambda = 0.7$, $\epsilon = 0.01$. The state of the system s_k at time step k is the object's current and previous positions, $s_k = (x_k, y_k, x_{k-1}, y_{k-1})$. The sample time of our simulation is 0.01 seconds (between step the integration method is ODE45). For object's position, a $13 \times 13 \times 9 \times 9$ spatial

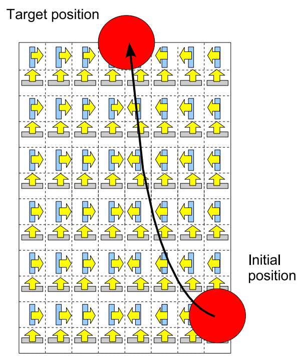
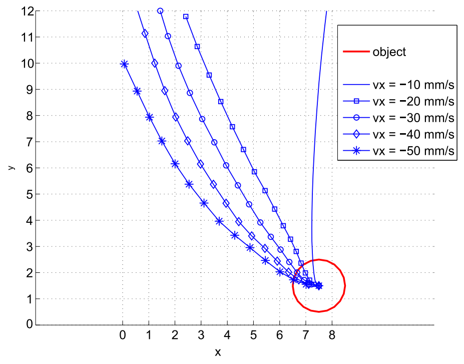
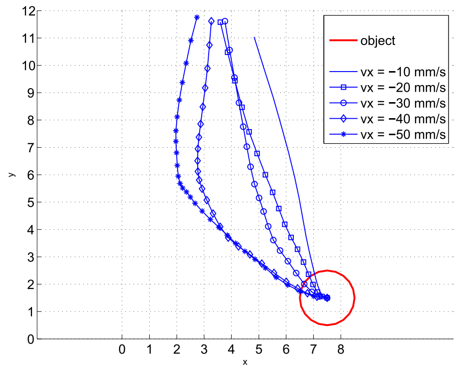


Fig. 9. Vector field for the part conveying.



(a) Results with programmable vector field



(b) Results with semi-decentralized RL control

Fig. 10. Object's trajectories for various initial speeds of the object. The initial speed $\dot{x}$ is here noted vx.

tile-coding is used.

According to [21], we incorporate prior knowledge or bias to speed up RL. We choose to use a transient bias embedded in initial Q values. This bias advises controllers to generate the previous vector field (figure 9) at the beginning of the learning process. In order to realize the part conveying, the chosen reward function is :

$$R(x_k, y_k) = \begin{cases} 10e^{-\frac{(x - \frac{x_{max}}{2})^2}{2}} \text{ if } y \geq y_{max} \\ -3 \text{ if } x < x_{min} \text{ or } x > x_{max} \text{ or } y < y_{min} \\ 0 \text{ else.} \end{cases} \quad (13)$$

where (x, y) are the object's coordinates. The reinforcement function rewards agents when the goal is reached, and punishes them when another border is crossed.

At the end of the learning process, some conveying tasks with various initial speeds of the object ($\dot{x} = \{-10, -20, -30, -40, -50\}$ mm/s) are realized with controllers following their greedy policy[6]. The figure 10b shows the object's trajectories.

Semi-decentralized RL manages to convey the object near to the target position for all tested initial speeds. Controllers adjusted the initial bias so as to fit the speed of the object. This result confirms the potential capacity of semi-decentralized RL methods to control such distributed systems.

VI. Conclusions and future works

We first showed that the usual open-loop control for contact distributed-manipulation system, called *programmable vector field*, fails to damp object's motion and to reject perturbations on the contact-free distributed micromanipulator.

Then, a semi-decentralized reinforcement learning control approach has been investigated. This way of control has been validated in comparison with programmable vector field. Notably, it achieves good positioning and conveying performance and has high control stability. This result is a proof-of-concept of using decentralized RL control for distributed-manipulation systems. It is also a new successful application of decentralized Q-learning variant algorithms for independent agents.

We made here some strong assumptions in the model used for simulation. Although the behavior of the model is realistic, some assumptions as the independence of air-jets may be simplistic. Well, the purpose of this paper is precisely to propose a control approach by learning that does not need to state any model to find a controller. One major interest of this approach is to adapt itself to any systems. So, there is likelihood that decentralized RL control will achieve the control of a real contact-free distributed micromanipulator

Acknowledgment

The authors gratefully acknowledge Joël Agnus and David Guibert from the FEMTO-ST Institute for their technical assistance. This work was supported in part by the Smart Surface NRA (French National Research Agency) project (ANR_06_ROBO_0009_03).

Appendix I

Numerical data of dynamical model

Parameter	Caption	Value	Unit
m	object's mass	$6, 6.10^{-3}$	g
l	object's thickness	$2, 5.10^{-1}$	mm
C_x	drag coefficient	$1, 11$	
ρ	air density	$1, 3$	kg/m^3
J	object's moment of inertia	$0, 05$	g/mm^2
η	air viscosity	$1, 81.10^{-5}$	kg/m.s
K	viscosity coefficient	$2, 75$	

[6]The selected action is the one with the largest Q_i-value in the current state.

References

[1] K. Verbeeck, A. Nowé, J. Parent, and K. Tuyls, "Exploring selfish reinforcement learning in repeated games with stochastic rewards," *Autonomous Agents and Multi-Agent Systems*, vol. 14, no. 3, pp. 239–269, 2007.

[2] K. Tumer and A. Agogino, "Distributed agent-based air traffic flow management," in *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM, 2007, pp. 1–8.

[3] D. Wolpert, J. Sill, and K. Tumer, "Reinforcement learning in distributed domains: Beyond team games," in *Proceedings of the Seventeenth IJCAI*, 2001.

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, 1998.

[5] K.-F. Böhringer, B. R. Donald, R. Mihailovich, and N. C. MacDonald, "Sensorless manipulation using massively parallel microfabricated actuator arrays," in *Proc. of IEEE ICRA*, San Diego, CA, May 1994, pp. 826–833.

[6] M. Ataka, B. Legrand, L. Buchaillot, D. Collard, and H. Fujita, "Design, fabrication and operation of two dimensional conveyance system with ciliary actuator arrays," *IEEE/ASME Transactions on-Mechatronics*, vol. 14, pp. 119–125, 2009.

[7] Y. Fukuta, Y.-A. Chapuis, Y. Mita, and H. Fujita, "Design, fabrication and control of mems-based actuator arrays for air-flow distributed micromanipulation," *Journal of Micro-Electro-Mechanical Systems*, 2006.

[8] S. Konishi and H. Fujita, "A conveyance system using air flow based on the concept of distributed micro motion systems," *Journal of Micro-Electro-Mechanical Systems*, vol. 3, no. 2, pp. 54–58, 1994.

[9] K.-F. Bohringer, B. Randall, D. Noel, and C. Macdonald, "What programmable vector fields can (and cannot) do: Force field algorithms for mems and vibratory parts feeders," in *Proc. of IEEE ICRA*, 1996, pp. 822–829.

[10] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[11] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems." in *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998, pp. 746–752.

[12] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. of the International Conference on Machine Learning*. Morgan Kaufmann, 2000, pp. 535–542. [Online]. Available: citeseer.ist.psu.edu/lauer00algorithm.html

[13] C. Watkins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[14] M. Tan, "Multiagent reinforcement learning: Independent vs. cooperative agents," in *10th International Conference on Machine Learning*, 1993, pp. 330–337.

[15] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "A study of fmq heuristic in cooperative multi-agent games," in *Proceedings of the Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS), Workshop 10: Multi-Agent Sequential Decision Making in Uncertain Multi-Agent Domains,*, 2008.

[16] S. Sen and M. Sekaran, "Individual learning of coordination knowledge," *JETAI*, vol. 10, no. 3, pp. 333–356, 1998.

[17] L. Busoniu, R. Babuska, and B. D. Schutter, "Decentralized reinforcement learning control of a robotic manipulator," in *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV 2006)*, Singapore, Dec. 2006, pp. 1347–1352.

[18] Y. Wang and C. W. de Silva, "Multi-robot box-pushing: Single-agent q-learning vs. team q-learning," in *Proc. of IROS*, 2006, pp. 3694–3699.

[19] H. Guo and Y. Meng, "Dynamic correlation matrix based multi-q learning for a multi-robot system," in *IROS*, 2008, pp. 840–845.

[20] C. J. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Cambridge University, Cambridge, England, 1989.

[21] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Improving reinforcement learning speed for robot control," in *Proc. of the IEEE International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 9–15 2006.