Designing decentralized controllers for distributed-air-jet MEMS-based micromanipulators by reinforcement learning

Laëtitia Matignon · Guillaume J. Laurent · Nadine Le Fort-Piat · Yves-André Chapuis

Abstract Distributed-air-jet MEMS-based systems have been proposed to manipulate small parts with high velocities and without any friction problems. The control of such distributed systems is very challenging and usual approaches for contact arrayed system don't produce satisfactory results. In this paper, we investigate reinforcement learning control approaches in order to position and convey an object. Reinforcement learning is a popular approach to find controllers that are tailored exactly to the system without any prior model. We show how to apply reinforcement learning in a decentralized perspective and in order to address the global-local trade-off. The simulation results demonstrate that the reinforcement learning method is a promising way to design control laws for such distributed systems.

Keywords MEMS-based actuator array · smart surface · decentralized control · distributed control · reinforcement learning

1 Introduction

One of potential applications of Micro-Electro-Mechanical Systems (MEMS) concerns the moving and positioning of small parts as the actuators themselves are on a similar scale. Toward the realization of this goal, researchers have been designing and building actuator arrays that can be used for positioning, conveying and sorting of small parts,

Laëtitia Matignon
FEMTO-ST/UFC-ENSMM-UTBM-CNRS, Université de Franche-Comté, Besançon, France
E-mail: laetitia.matignon@ens2m.fr

Guillaume J. Laurent
FEMTO-ST/UFC-ENSMM-UTBM-CNRS, Université de Franche-Comté, Besançon, France
E-mail: guillaume.laurent@ens2m.fr

Nadine Le Fort-Piat
FEMTO-ST/UFC-ENSMM-UTBM-CNRS, Université de Franche-Comté, Besançon, France
E-mail: nadine.piat@ens2m.fr

Yves-André Chapuis
InESS/ULP-CNRS, Université Louis Pasteur, Strasbourg, France
E-mail: yves-andre.chapuis@iness.c-strasbourg.fr

or as bulk-fabricated (cheap), ultra-thin transport mechanisms, e.g. for paper in copy machines or printers.

A wide variety of actuation principles for arrayed systems has been proposed in recent years including electromagnetic actuators [23], electrostatic actuators [25, 4, 16, 12], thermal-bimorph (bimaterial) actuators [2, 1]. Arrayed manipulation systems can be divided into two categories: contact systems and contact-free systems. Contact systems simulate cilia and can mainly perform high load capacity and accurate positioning [4, 2, 1]. Contact-free systems use air-flow levitation and have several advantages including high velocities and the removal of friction problems [25, 16, 12]. In return, they require a greater level of complexity for the control.

In the case of contact systems, the friction forces are very important compared with the inertia of the carried object. The control strategy proposed by Böringher *et al.* and called *programmable vector field* [6, 3, 5] produces satisfactory results for positioning and conveying a part with ciliary actuator arrays.

On the contrary, there are many challenging issues concerning the control of contact-free systems. First, the damping of the motion is very weak. Constant air-flow generates strong instability due to fluid turbulences. Fluid effects are also very non linear. It is very hard to model precisely interactions between tens different air-jets. The shape of the object and the roughness of its surface have a major impact on the exerted fluid forces. Moreover, actuators and then air-jets are not perfectly the same due to process dispersion factors. Lastly, finding coherent control laws for hundreds of independent air-jets is a complex problem.

In this paper, we propose to use reinforcement learning control techniques in a decentralized perspective as a way to design control laws for contact-less distributed manipulation systems. Reinforcement learning is a popular approach to find controllers that are tailored exactly to the system without any prior knowledge. In recent years, reinforcement learning has been applied with reported success to control complex real world problems in a decentralized perspective, like multi-robot cooperative transportation tasks [33], urban traffic control [17] or a flocking system [13]. The challenge to design decentralized control laws is to find a trade-off between global and local perspectives: the global problem is too complex to be solved but solving merely the local problems can lead to poor global performances. The proposed control architecture is based on the game theory perspective, where a team of agents are cooperating to solve a joint task [24, 7]. This framework allows to find local control laws whose joint actions are optimal at the global level.

We focused our study on a distributed-air-jet MEMS-based micromanipulator designed by Fukuta et al. [12] 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[1]. The device to control is an array of micro-electrovalves 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. The long-term objective is to design and develop a fully integrated distributed micromanipulation 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 and decentralized. However, a fully integrated approach still remains rigid and costly in micro-scale fabrication for research works on control. Consequently,

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

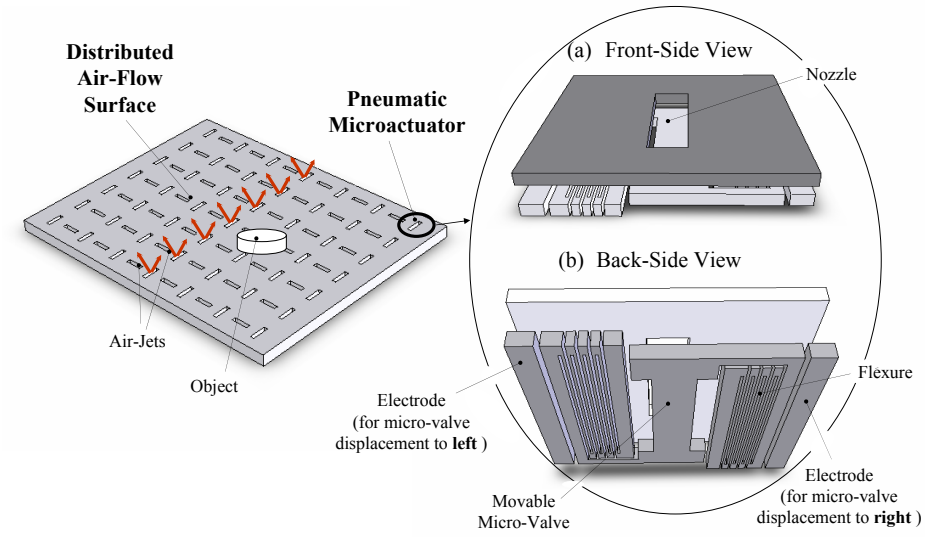


Fig. 1 Distributed-air-jet MEMS-based micromanipulator [12].

in this paper, we experiment our control algorithms on a realistic simulation of the distributed-air-jet micromanipulator.

This paper is organized as follow. The first part presents the distributed-air-jet micromanipulator and a model of it. The second part aims to place our work with regard to previous works and to the usual programmable-vector-field approach for arrayed systems. The next parts present two control architectures and experiments done in simulation on two different tasks.

2 Distributed-air-jet micromanipulator

2.1 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 micro-nozzles. Air-flow comes through electrostatic micro-valves in the back-side 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 [12] for more details about MEMS's design and working.

2.2 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.

2.2.1 Fluid forces

Each nozzle is centered on a $1\,\mathrm{mm}^2$ square. A nozzle generates :

- either a vertical air-jet $\vec{v}_{jet} = v_{lev}\vec{z}$ when the micro-valve is closed (the valve state a is 0); vertical air-jets are useful for levitation,
- or an oriented air-jet $\vec{v}_{jet} = v_{air}\vec{x}+v_{lev}\vec{z}$ (nozzle normal to $\vec{x}$) or $\vec{v}_{air} = v_{air}\vec{y}+v_{lev}\vec{z}$ (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 fluid 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 $\vec{z}$.

2.2.2 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 remain 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.

2.2.3 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 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.

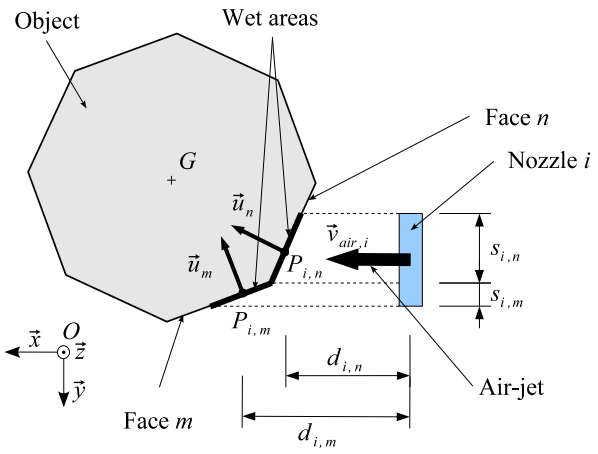


Fig. 2 Geometric notations in case of a nozzle close to the object, normal to $\vec{x}$ and orienting an air-jet with a velocity $\vec{v}_{air,i}$ (top view).

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 $\vec{v}_{obj}$ 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 [12] 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 :

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

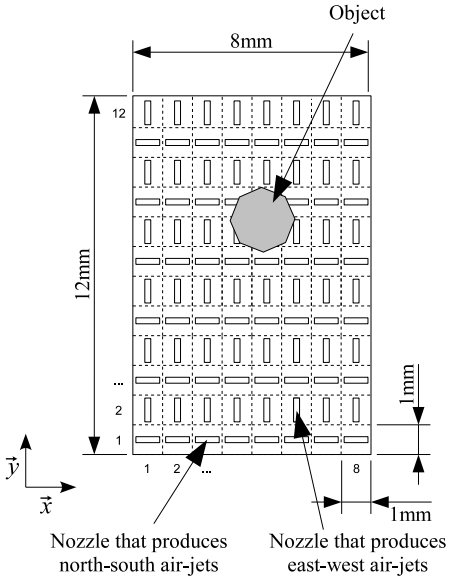


Fig. 3 Geometry of the distributed-air-jet micromanipulator. The device has a total of 96 actuators (air-jet).

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 :

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

2.2.4 Viscous force

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

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

where K is a geometric coefficient dependent on the shape of the object and η the coefficient of viscosity.

2.2.5 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.

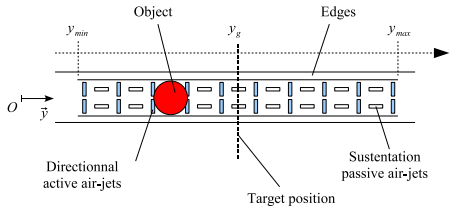


Fig. 4 Part-stabilization-task description.

3 A challenging control problem

3.1 Previous work

Very first control experiments with the real system were done by some of us [12,9]. A close-loop control based on decentralized reactive units was developed. This control system was able to move a $5.6 \times 5.6\,\mathrm{mm}^2$ plastic planar object to one direction but neither the deviation of the object in the other direction nor the speed were controlled during the motion. Periodical on-off pulses of air-flow were also used in order to advance the part by leaps and bounds.

This experiment was a proof-of-concept of the ability of the micromanipulator to convey a small part with decentralized control. However, there are still many challenging issues concerning the motion control and the increase of stability in full levitation.

3.2 Programmable vector field

Programmable vector field was introduced by Böhringer *et al.* [6,3,5] to control actuator arrays and transversely vibrating plates. Programmable vector field is sensor-less and may be employed to orient, sort, feed, assemble parts, etc. It was applied with success to control ciliary actuator arrays [4,1].

The basic principle of programmable vector field is that actuators are assigned to a specific direction and magnitude 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.

3.3 Part stabilization with programmable vector field

First, we focus 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.

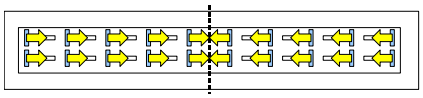


Fig. 5 Vector field for the part-stabilization task.

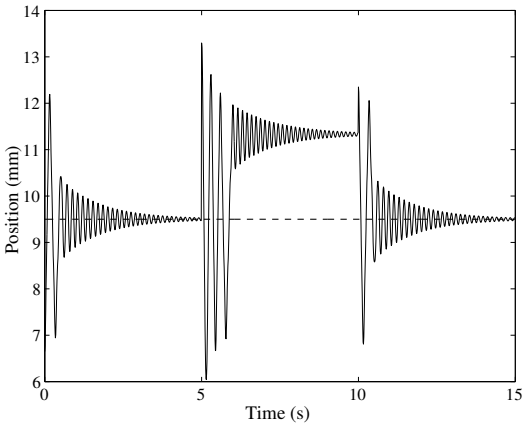


Fig. 6 Position of the cylinder's center according to the time using programmable vector field (solid line), the dashed line represents the target position. Short perturbations are done at time steps 0, 5 and 10 s.

The problem is only to address a direction to each air-jet in order to get a specific motion of the object. The magnitude of an air-jet is not adjustable. 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 $y_g = 9.5\,\text{mm}$.

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

3.4 Part conveying with programmable vector field

The second task is to convey the part to the middle of the north border of the surface illustrated on the figure 3. For this task, the entire surface is used. The part is fed to the surface at the initial position $x_0 = 7.5\,\text{mm}$, $y_0 = 1.5\,\text{mm}$ and with a random

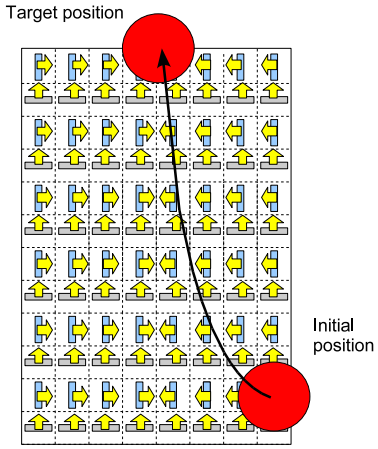


Fig. 7 Vector field for the part-conveying task.

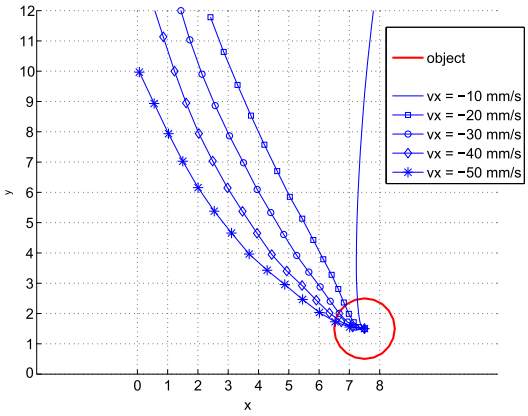


Fig. 8 Object's trajectories for various initial speeds of the object using programmable vector field. The initial speed $\dot{x}$ is here noted vx.

speed $\dot{x} \in [-50, -10]\,\mathrm{mm/s}$. The vector field shown by figure 7 is used to convey the object. The figure 8 shows the part's trajectories for various initial speeds ($\dot{x} = \{-10, -20, -30, -40, -50\}\,\mathrm{mm/s}$). As illustrated on the figure, this vector field is not able to convey the object to its target position for all initial speeds.

3.5 Discussion

These two examples show that programmable vector field is not satisfactory to control the distributed-air-jet system. After all, it is not surprising: programmable vector field is a sensor-less method. It can't adapt to different speeds of the object. In this case, it may be interesting to investigate close-loop control using dynamic vector field.

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

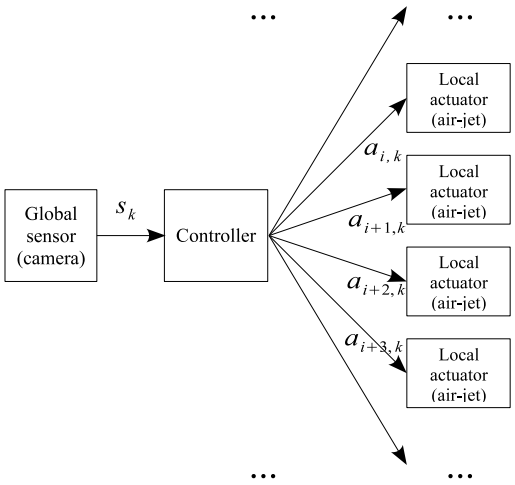


Fig. 9 Centralized control architecture.

4 Semi-decentralized reinforcement learning control

4.1 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 9).

Another solution is to use one controller per air-jet. The object's position is broadcast 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 10).

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.

4.2 Reinforcement learning

Reinforcement learning methods are inspired by dynamic programming concepts. They have been studied extensively and successfully applied in centralized framework [29, 15]. A controller, also called agent, learns by interactions with its environment, using a

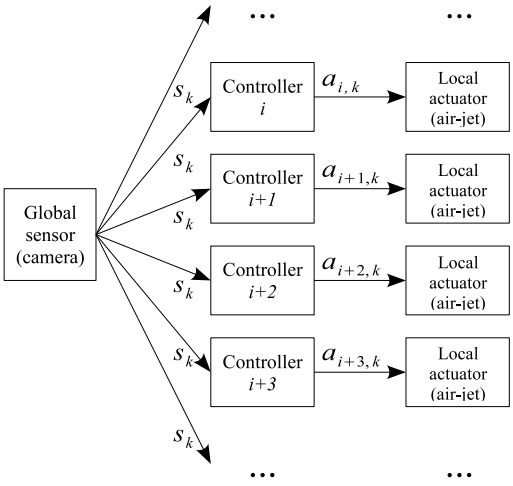


Fig. 10 Semi-decentralized control architecture.

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*[2] 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 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).

4.3 Think globally, act locally

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

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

$Q^{\boldsymbol{\pi}}(s, a)$ is called the joint or global action-value function. $\gamma \in [0; 1[$ is a discount factor.

In the multi-agent system framework, independent learners were introduced in [10] as agents which don't know the actions taken by the other agents. The objective of an

[2] also called *team game*.

independent learner 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 independent learner 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 actuators.

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 [18].

4.4 Decentralized Q-Learning

Q-learning [34] is one of the mostly used reinforcement learning algorithms 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 [31]. Despite some difficulties as the coordination or the loss of theoretical guarantees [19], it was successfully applied with independent learners on some applications [27, 8, 32, 14].

For an independent learner 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 $k-1$ and $\alpha \in]0; 1]$ is the learning rate. $Q_i(s, a_i)$ 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 use 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 visited 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 independent learners of the Watkins's $Q(\lambda)$ algorithm[3] [35].

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[4].

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

[4] 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.

4.5 Part stabilization

As illustrated on the figure 4, we first 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 reinforcement learning 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 $41 \times 41 \times 3$ for each 10 controllers, to compare with a Q table of size $41 \times 41 \times 3^{10}$ in a centralized perspective.

According to [20] 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 Q-Learning during 300 trials ($\lambda = 0$). Each trial starts with the object in a random initial state ($y_0 \in [3, 15]$ mm) and runs at the most 10 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 $\rho = 0.5$ mm.

Figure 11 shows the position of the object's center according to the time after learning. It takes around 0.2 seconds for the independent learners 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 reinforcement learning manages to stabilize the object with a high damp factor and with a good robustness to perturbations. This result demonstrates the potential capacity of reinforcement learning control to regulate position and speed of a levitating part on distributed-air-jet micromanipulator.

4.6 Part conveying

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 $Q(\lambda)$ 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, $s_k = (x_k, y_k, x_{k-1}, y_{k-1})$. For object's position, a $13 \times 13 \times 9 \times 9$ spatial tile-coding is used.

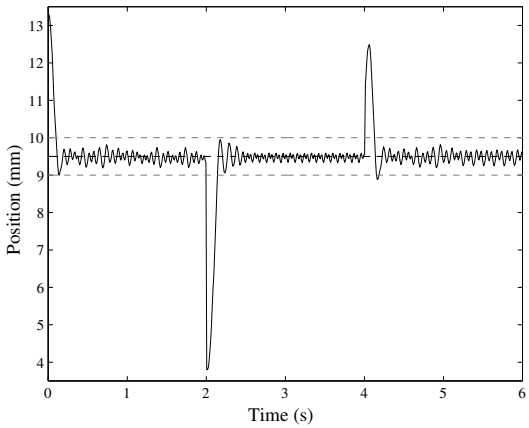


Fig. 11 Position of the cylinder's center according to the time (solid line) after learning with the Q-Learning algorithm, the dashed line represents the target position. The control architecture is semi-decentralized. Short perturbations are done at time steps 0, 2 and 4 s.

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.

Because the slow learning speed is an issue when applying reinforcement learning to real-world problems, alternative strategies have been proposed in the literature as the incorporation of prior knowledge (or bias) [20] or the combination of a conventional controller with the reinforcement learning scheme [28]. According to [20], we choose to use a transient bias embedded in initial Q values. This bias advises controllers to generate the previous vector field (figure 7) at the beginning of the learning process.

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. The figure 12 shows the object's trajectories.

Semi-decentralized reinforcement learning 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 capacity of semi-decentralized reinforcement learning methods to control such distributed systems.

5 Fully-decentralized reinforcement learning control

5.1 Control architecture

In this section, for conducting the proof-of-concept of a future smart surface, integrating sensors, actuators and processing units, we investigate a complete decentralized control architecture using integrated local sensors. Fully-decentralized architecture avoids the

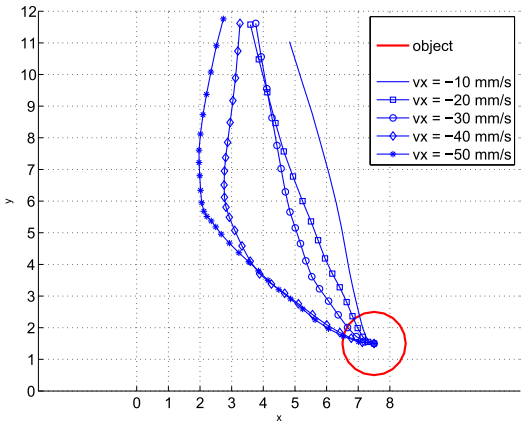


Fig. 12 Object's trajectories for various initial speeds of the object using semi-decentralized reinforcement learning control. The initial speed $\dot{x}$ is here noted vx.

broadcast of the position of the object. Moreover, it provides better scalability and robustness properties to the system.

We assume that sensors are located between nozzles (cf. figure 13). Each sensor detects if the object is above it or not. Controllers must receive sufficient amount of information to control the object's motion. It is obvious that if the system is not observable, the controller will have poor performances. So we propose that sensor's information is locally shared. The control architecture is described by figure 14. Each decentralized controller drives an actuator and receives informations about close sensors.

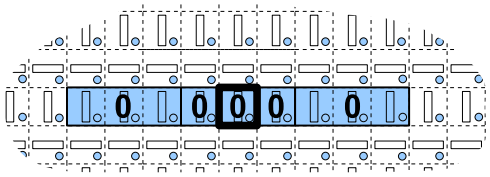
5.2 Controlled and non-controlled observations

To reduce at most sensors sharing, the idea is that controllers have to observe the object only when they can act on it. In other words, when the object is reachable by the controlled air-jet, the controller must receive enough information to be efficient. But, when the object is not reachable, it is not necessary for the controller to receive any information.

Given that the speed of an air-jet is null at 4 mm, we use local perceptions as defined in figure 13. We noted ω_i the observation vector of a controller i. In our case, ω_i is a five-length binary vector describing the state of close detectors in the direction of the air-jet. We can notice that this observation vector is a very coarse observation of the object position.

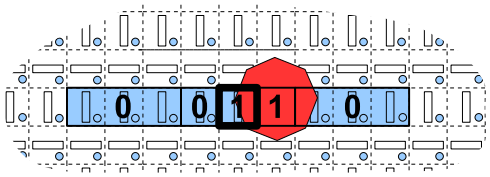
As a controller i receives only ω_i, it does not observe the exact position and the exact speed of the object. Then, the state of the system is said partially observable. To address this partial observability problem, the idea is to split the observation space into two subsets:

– Ω_i is the set of controlled observations, i.e. when the controller can act on the system,



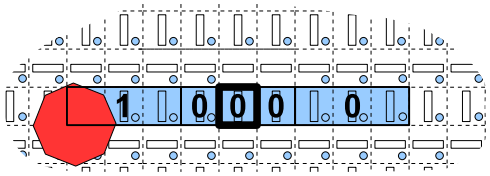
$\omega_i = (0\ 0\ 0\ 0\ 0)$

(a) Agent's local view when all detectors are inactive (sensors are represented by small circles). The object is not reachable by agent's air-jet. This observation is said non-controlled. The agent waits and follows a fixed policy.



$\omega_i = (0\ 0\ 1\ 1\ 0)$

(b) Agent's local view when some detectors are active. The object is reachable by agent's air-jet. This observation is said controlled. The agent learns to control the object's position and follows its current policy.



$\omega_i = (1\ 0\ 0\ 0\ 0)$

(b) Agent's local view when one of three very left detectors are active. The object is reachable by agent's air-jet. This observation is said controlled. The agent learns to control the object's position and follows its current policy.

Fig. 13 Local view and controlled observations principle.

– $\bar{\Omega}_i$ is the set of non-controlled observations, i.e. when the controller can't act on the system.

In our case, $\bar{\Omega}_i$ contains only the observation $(0\ 0\ 0\ 0\ 0)$ and Ω_i all the others.

In order to use reinforcement learning algorithm in this framework, we use the "options" paradigm. Options enable multi-step actions (or macro-actions) to be included in the reinforcement learning framework in a natural and general way [30,26,11].

Non-controlled observations can be seen as an option in which the controller follows a given constant policy without learning (for instance, if $\omega_{i,k} \in \bar{\Omega}_i$ the agent keeps air-jet on the middle). If the observations are controlled, the controller learns as usual and follows a policy based on its action value function.

To be more precise, four cases must be exposed:

– If $\omega_{i,k-1} \in \Omega_i$ and $\omega_{i,k} \in \Omega_i$, the agent stays in controlled observation space and updates its action-value function with the transition $(\omega_{i,k-1}, a_{i,k-1}, \omega_{i,k}, r_k)$ and

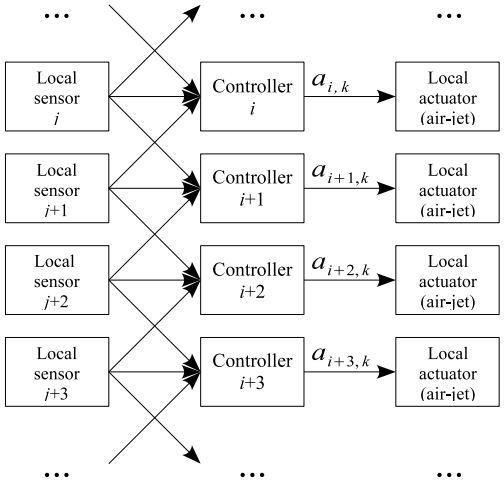


Fig. 14 Fully-decentralized control architecture with sensors sharing.

the discount factor γ using the Q-Learning equation (or another algorithm):

$$Q_i(\omega_{i,k-1}, a_{i,k-1}) \leftarrow (1-\alpha)Q_i(\omega_{i,k-1}, a_{i,k-1}) + \alpha(r_k + \gamma \max_{b \in A_i} Q_i(\omega_{i,k}, b)) \tag{14}$$

– If $\omega_{i,k-1} \in \Omega_i$ and $\omega_{i,k} \in \bar{\Omega}_i$, the agent enters non-controlled observation space and initializes temporary variables:

$$\begin{cases} \bar{k} \leftarrow k-1 \\ \bar{\gamma} \leftarrow \gamma \\ \bar{r} \leftarrow r_k \end{cases} \tag{15}$$

– If $\omega_{i,k-1} \in \bar{\Omega}_i$ and $\omega_{i,k} \in \bar{\Omega}_i$, the agent stays in non-controlled observation space, only temporary variables are updated:

$$\begin{cases} \bar{r} \leftarrow \bar{r} + \bar{\gamma} r_k \\ \bar{\gamma} \leftarrow \bar{\gamma} * \gamma \end{cases} \tag{16}$$

– If $w_{i,k-1} \in \bar{\Omega}_i$ and $w_{i,k} \in \Omega_i$, the agent gets out non-controlled observation space and updates its action-value function with the transition $(\omega_{i,\bar{k}}, a_{i,\bar{k}}, \omega_{i,k}, \bar{r})$ and the discount factor $\bar{\gamma}$ using the Q-Learning equation (or another algorithm):

$$Q_i(\omega_{i,\bar{k}}, a_{i,\bar{k}}) \leftarrow (1-\alpha)Q_i(\omega_{i,\bar{k}}, a_{i,\bar{k}}) + \alpha(\bar{r} + \bar{\gamma} \max_{b \in A_i} Q_i(\omega_{i,k}, b)) \tag{17}$$

5.3 SOaN algorithm

In two dimensions, a lot of path can convey a part to a target point. Some of them could be equivalent in terms of value function. It means there are several joint policies that are optimal. The consequence is that, to achieve cooperation, independent learners must select the same optimal joint policies without any communication. Independent learners based on Q-Learning may fail to achieve policy selection [19].

A second constraint is agents perceive local information with very low sensor resolution. So, agents' observations of their world are very noisy. To conclude, we need a reinforcement learning algorithm for independent learners that would be robust to noise and able to do the policy selection.

The SOaN algorithm that some of us developed in previous works [22, 19] satisfies these requirements. Thanks to the computation of a farsighted frequency and to a heuristic evaluation of the action values, this algorithm sways from optimistic to neutral evaluation according to a detection of the noise in the environment. This algorithm overcomes all mis-coordination factors, notably policy selection, even in weakly noisy Markov games. The robustness of the SOaN algorithm have been demonstrated in practice in many cooperative Markov games with numerous agents.

5.4 Part conveying

We apply both controlled observation framework and SOaN algorithm to control the simulated air-jet micromanipulator.

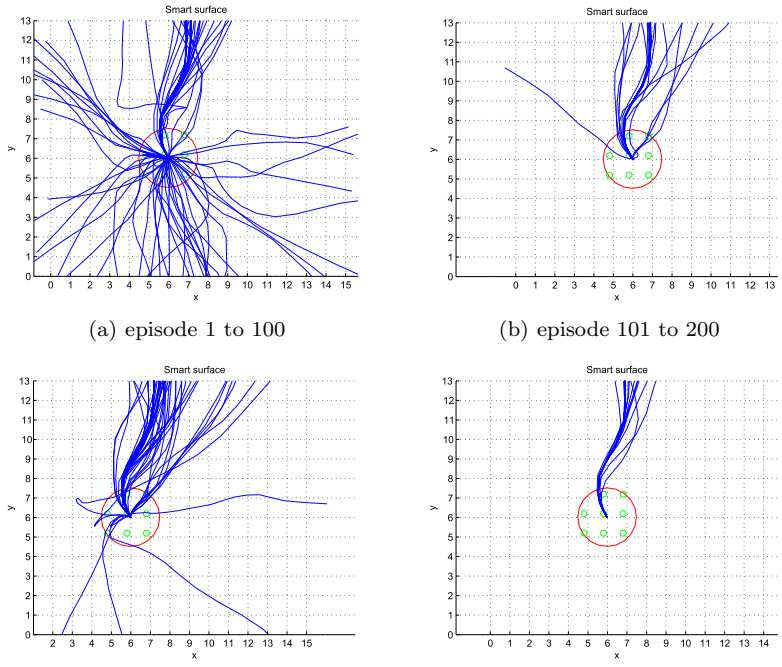
An episode starts with the object at the center of the surface with a slight speed to west (-1 mm/s). An episode stops if the object's center crosses a border. A task is successful if the object's center crosses the north border in the middle plus or minus 2.5 mm. The reinforcement function is the same than equation 13.

In table 1, we compare several approaches on the same tasks. It is clear that SOaN algorithm overcomes Q-Learning and hysteretic Q-Learning, another algorithm for independent learners [21]. The standard deviation is very low and the small part of unsuccessful tasks is simply due to learning as we can see on figure 15. This figure shows the trajectories of the object in hundreds of learning episodes. For the first 100 episodes (cf. figure 15a), trajectories are disappointing most of the time. Indeed, it is the beginning of the learning and no bias is induced in the initialization, so agents have to learn from scratch. That's why the trajectories seem hazardous. For the next 200 episodes (cf. figures 15b and c), the agents automatically adjust their behavior to the environment. Most of the trajectories turn towards north but they are not precise. A few trajectories fail because of exploration actions taken by a few agents. Some risked and corrected trajectories can be observed on figure 15c. Finally, for the last 100 episodes (cf. figure 15d), all trajectories are a successful. Agents manage to coordinate themselves and risked trajectories are quickly corrected. The "S" shape we can see at the end of learning process is induced by the initial speed of object.

The final trajectories are less precise than in the semi-decentralized framework. Indeed, the task is harder since the local observations received by controller are very coarse. However, it shows a real interest to use reinforcement learning to design controllers in a fully decentralized perspective.

Algorithm	Mean of number of successful episodes	Standard deviation of number of successful episodes
Q-learning	63%	24%
Hysteretic Q-learning	19%	31%
SOaN	81%	6%

Table 1 Percentage of episodes where the conveyance task was successful for 400 episodes (means on five independent runs).



(a) episode 1 to 100

(b) episode 101 to 200

(c) episode 201 to 300

(d) episode 301 to 400

Fig. 15 Object's trajectories during learning from scratch with the simulated air-jet micro-manipulator and with the SOaN algorithm. The control architecture is fully-decentralized.

6 Conclusions and future works

We first showed that sensor-less programmable vector field does not provide satisfactory results for the simulated distributed-air-jet system. However, it may be interesting to investigate close-loop control using dynamic vector field.

Then, two decentralized reinforcement learning control approaches were investigated. These approaches were compared with programmable vector field on the same stabilization and conveying tasks. The results demonstrate the capacity of reinforcement learning to control the position or the trajectory of a levitating part on the

simulated distributed-air-jet micromanipulator. Reinforcement learning is a promising way to design control laws for such distributed systems.

If these results are a proof-of-concept of using decentralized reinforcement learning for this kind of system, they also show new successful applications of Q-learning variant algorithms for independent learners.

We made here some strong assumptions in the model used for simulation. Although the behavior of the model is realistic compared to real experiments, 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 reinforcement learning control will achieve the control of a real contact-free distributed micromanipulator. It remains to confirm these results on a real system.

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

A Numerical data of dynamical model

Parameter	Caption	Value	Unit
m	object's mass	6.6e-3	g
l	object's thickness	0.25	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	1.81e-5	kg/m.s
K	viscosity coefficient	2.75	

References

1. Ataka, M., Legrand, B., Buchaillot, L., Collard, D., Fujita, H.: Design, fabrication and operation of two dimensional conveyance system with ciliary actuator arrays. IEEE/ASME Transactions on Mechatronics **14**, 119–125 (2009)
2. Ataka, M., Omodaka, A., Takeshima, N., Fujita, H.: Fabrication and operation of polyimide bimorph actuators for a ciliar motion system. IEEE/ASME Journal of Micro-Electro-Mechanical Systems **2**(4), 146–150 (1994)
3. Böhringer, K.F., Donald, B.R., MacDonald, N.C.: Upper and lower bounds for programmable vector fields with applications to memd and vibratory plate parts feeders. Algorithms for Robotic Motion and Manipulation J.-P. Laumond and M. Overmars (1997)
4. Böhringer, K.F., Donald, B.R., Mihailovich, R., MacDonald, N.C.: Sensorless manipulation using massively parallel microfabricated actuator arrays. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 826–833. San Diego, CA (1994)
5. Bohringer, K.F., Donald, B.R., Kavraki, L., Lamiraux, F.: Part orientation with one or two stable equilibria using programmable vector fields. IEEE Transactions on Robotics and Automation **16**(2), 157–170 (2000)
6. Bohringer, K.F., Randall, B., Noel, D., Macdonald, C.: What programmable vector fields can (and cannot) do: Force field algorithms for mems and vibratory parts feeders. In: Proc. of the IEEE Int. Conf. on Robotics and Automation, pp. 822–829 (1996)
7. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews **38**(2), 156–172 (2008)

8. Busoniu, L., Babuska, R., Schutter, B.D.: Decentralized reinforcement learning control of a robotic manipulator. In: Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV 2006), pp. 1347–1352. Singapore (2006)
9. Chapuis, Y.A., Zhou, L., Fukuta, Y., Mita, Y., Fujita, H.: Fpga-based decentralized control of arrayed mems for microrobotic application. IEEE Transactions on Industrial Electronics **54**(4), 1926–1936 (2007)
10. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence, pp. 746–752 (1998)
11. Elfwing, S., Uchibe, E., Doya, K., Christensen, H.I.: Multi-agent reinforcement learning: Using macro actions to learn a mating task. In: Proceedings of ICRA (2004)
12. Fukuta, Y., Chapuis, Y.A., Mita, Y., Fujita, H.: Design, fabrication and control of mems-based actuator arrays for air-flow distributed micromanipulation. Journal of Micro-Electro-Mechanical Systems (2006)
13. Gu, D., Yang, E.: Fuzzy policy reinforcement learning in cooperative multi-robot systems. Journal of Intelligent and Robotic Systems **48**(1), 7–22 (2007)
14. Guo, H., Meng, Y.: Dynamic correlation matrix based multi-q learning for a multi-robot system. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems IROS 2008, pp. 840–845 (2008)
15. Katić, D.M., Rodić, A.D., Vukobratović, M.K.: Hybrid dynamic control algorithm for humanoid robots based on reinforcement learning. Journal of Intelligent and Robotic Systems **51**(1), 3–30 (2008)
16. Konishi, S., Fujita, H.: A conveyance system using air flow based on the concept of distributed micro motion systems. Journal of Micro-Electro-Mechanical Systems **3**(2), 54–58 (1994)
17. Kuyer, L., Whiteson, S., Bakker, B., Vlassis, N.: Multiagent reinforcement learning for urban traffic control using coordination graphs. In: ECML PKDD '08: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I, *Lecture Notes in Computer Science*, vol. 5211, pp. 656–671. Springer (2008)
18. Lauer, M., Riedmiller, M.: An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: Proc. of the International Conf. on Machine Learning, pp. 535–542. Morgan Kaufmann (2000). URL citeseer.ist.psu.edu/lauer00algorithm.html
19. Matignon, L., Laurent, G.J., Fort-Piat, N.L.: A study of fmq heuristic in cooperative multi-agent games. In: Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems, Workshop 10: Multi-Agent Sequential Decision Making in Uncertain Multi-Agent Domains (2008)
20. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Improving reinforcement learning speed for robot control. In: Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems. Beijing, China (2006)
21. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Hysteretic q-learning : an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In: Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems, pp. 64–69. San Diego, CA, USA (2007)
22. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Coordination of independent learners in cooperative markov games. Tech. rep., Institut FEMTO-ST/UFC-ENSMM-UTBM-CNRS, Université de Franche-Comté, Besançon, France (2009). Http://hal.archives-ouvertes.fr/hal-00370889/fr/
23. Nakazawa, H., Watamasa, Y., Morita, O.: Electromagnetic micro-parts conveyer with coil-diode modules. In: Proc. of the IEEE International Conference of Solid-State Sensors and Actuators (Transducers '99), pp. 1192–1195 (1999)
24. Panait, L., Luke, S.: Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems **11**(3), 387–434 (2005)
25. Pister, K.S.J., Fearing, R., Howe, R.: A planar air levitated electrostatic actuator system. In: Proc. of the IEEE Workshop on Micro Electro Mechanical Systems (MEMS), pp. 67–71. Napa Valley, California (1990)
26. Precup, D.: Temporal abstraction in reinforcement learning. Ph.D. thesis, University of Massachusetts (2000). Director-Richard S. Sutton
27. Sen, S., Sekaran, M.: Individual learning of coordination knowledge. JETAI **10**(3), 333–356 (1998)
28. Song, K.T., Sun, W.Y.: Robot control optimization using reinforcement learning. Journal of Intelligent and Robotic Systems **21**(3), 221–238 (1998)

29. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)
30. Sutton, R.S., Precup, D., Singh, S.: Between mdps and semi-mdps: Learning, planning, and representing knowledge at multiple temporal scales. Artificial Intelligence **112**, 181–211 (1999)
31. Tan, M.: Multiagent reinforcement learning: Independent vs. cooperative agents. In: 10th International Conference on Machine Learning, pp. 330–337 (1993)
32. Wang, Y., de Silva, C.W.: Multi-robot box-pushing: Single-agent q-learning vs. team q-learning. In: Proc. of IROS, pp. 3694–3699 (2006)
33. Wang, Y., de Silva, C.W.: A machine-learning approach to multi-robot coordination. Eng. Appl. Artif. Intell. **21**(3), 470–484 (2008). DOI http://dx.doi.org/10.1016/j.engappai.2007.05.006
34. Watkins, C., Dayan, P.: Technical note: Q-learning. Machine Learning **8**, 279–292 (1992)
35. Watkins, C.J.: Learning from delayed rewards. Ph.D. thesis, Cambridge University, Cambridge, England (1989)