Chapter 11

On-Line Learning for Micro-Object Manipulation

11.1. Introduction

This chapter presents the application of a reinforcement learning algorithm to learning on-line how to manipulate micro-objects. What makes this application original is that the action policy has been learned not thanks to a simulator but by controlling the real process. This work related to reinforcement learning for micro-robotics has been conducted at the *Automatic control and Micro-Mechatronic Systems* department of the FEMTO-ST Institute (Besançon, France). It was first described in [ADD 05].

Micro-robotics has the general objective of designing, realizing and controlling compact robotic systems used to manipulate objects whose dimensions typically range between one micrometer and one millimeter for various applications (instrumentation, micro-assembly, biomedical applications).

Considering the dimensions and the required precision, micro-robotics faces practical difficulties which are quite different from in classical robotics:

– at the actuators level: micro-robotics employs new, more compact, actuation principles, for example based on the use of active materials; such actuators are often strongly non-linear;

– at the sensors level: the reduced volume of the applications makes it difficult to set up a sufficient number of sensors; employing a vision system (*via* a microscope) is often the main mean to observe and measure;

Chapter written by Guillaume LAURENT.

– at the level of the interactions between the robot and the surrounding objects: at this scale, surface forces become preponderant over volume forces; the inertia of objects is very weak; the friction between objects generates dry friction forces which are important and difficult to quantify; under 100 μm adhesion (capillarity) and Van der Waals forces make the objects "stick" to each other; these phenomena make manipulations extremely delicate and hazardous.

The actuators' non-linearity, the sensors' imprecision, the complexity of surface forces, make the processes hard to model. Because of the lack of any precise model, the synthesis of controllers through traditional approaches from the control field is difficult. In the contrary, reinforcement learning frees us from using any model of the controlled process and allows us to take into account the uncertainty in this process *via* a stochastic approach. These control methods are therefore adapted to micro-robotics.

This chapter is structured in three sections. The first section presents the context of micro-manipulation by pushing as well as the device to control. The second section describes the reinforcement learning algorithm employed for controlling the manipulator. Experimental results are presented in the last section.

11.2. Manipulation Device

11.2.1. *Micro-Positioning by Pushing*

In the industry, positioning is an essential function for machining or assembling parts. In the field of micro-robotics, classical solutions as "pick-and-place" are not directly transposable. In these conditions, it is often easier to push a micro-object than to hold it in a gripper. Thus, numerous works use this approach for micro-manipulation [CHE 07, GOR 06, GAU 06, SIT 04, MOL 02, ZES 98] and also for nano-manipulation [KOR 09, BAU 98, HAN 98, RES 00].

If these manipulators are simpler to design, things are different for their control. Indeed, the problem of predicting the movement of an object pushed at a given point is already complex at the macroscopic scale. Under a millimeter, classical friction equations based on the objects' weights (Coulomb's law) do not apply, due to the important preponderance of surface forces. The movement of an object being pushed depends on various parameters such as the surface roughness of the stand and of the object, air humidity, the distribution of electrostatic charges, etc. In these conditions, the movement of the object is difficult to predict, what led to consider a reinforcement learning approach for control. The objective therefore consists in synthesizing — by learning — high-performance control policies for a manipulator performing micro-positioning tasks by pushing.

11.2.2. *Manipulation Device*

The controlled manipulation device is inspired from existing manipulation devices that allow for pushing and pulling micro-parts one-by-one thanks to an atomic

force microscope's cantilever [GAU 06, SIT 04, ZES 98, HAN 98]. In the contrary, the scale of the device we describe is very different since this is about positioning millimeter-sized objects like watch gears. This device is a testbed whose objective is to demonstrate the feasibility of control by reinforcement learning for micro-positioning by pushing.

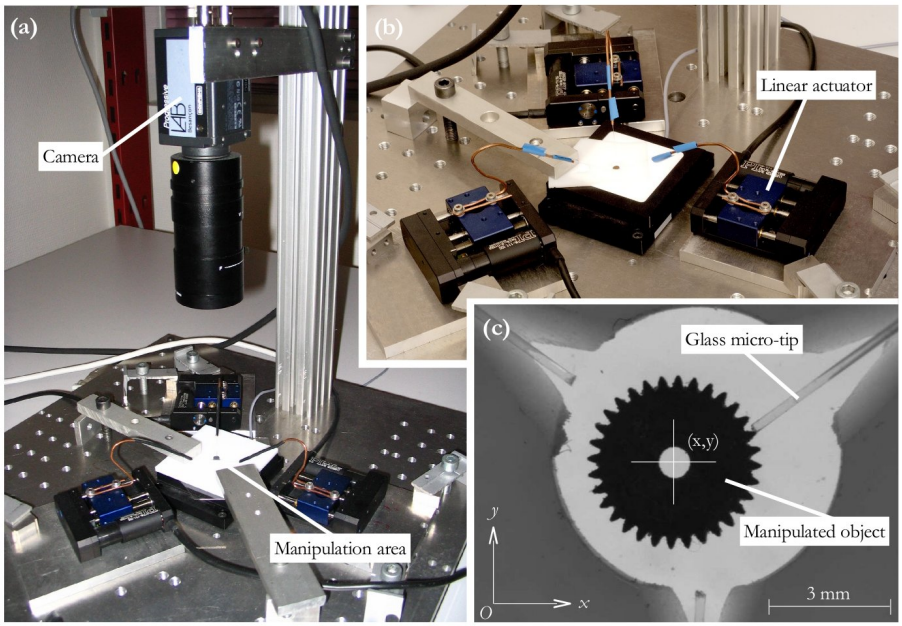


Figure 11.1. *Manipulation device: (a) overview, (b) zoom on the manipulation area surrounded by three linear actuators, (c) detail of the manipulation area (camera image). The object is localized with the vision system.*

The manipulator is equipped with three glass micro-tips mounted on linear actuators whose position is regulated with a micro-metric precision (see Figure 11.1b). The object to manipulate is laid on a glass slide between three micro-tips (see Figure 11.1c). Each tip can come in contact with the object then exert a thrust over a short distance. The objective is to move this object towards a given position through an adequate sequence of pushes.

The micro-tips' displacement axes are concurrent. The manipulation area is surrounded by a circular wall preventing the manipulated object from escaping. This disposition allows for moving a (circular) cylindrical object to any position.

The object position is measured *via* a vision system (see Figure 11.1a). The object is located through its x and y axes in the video image. The camera's resolution being limited, the precision of the localization is of about 23 µm.

11.2.3. *Control Loop*

To control the manipulator, two control levels are used: a low level and a high level (see Figure 11.2). The low level handles the position regulation of the micro-tips (with traditional control methods). This allows for bringing a micro-tip in contact with the object, then pushing it over a distance specified by the high level controller. The high level is the decision stage that plans pushes over the long term to bring the object to a given position.

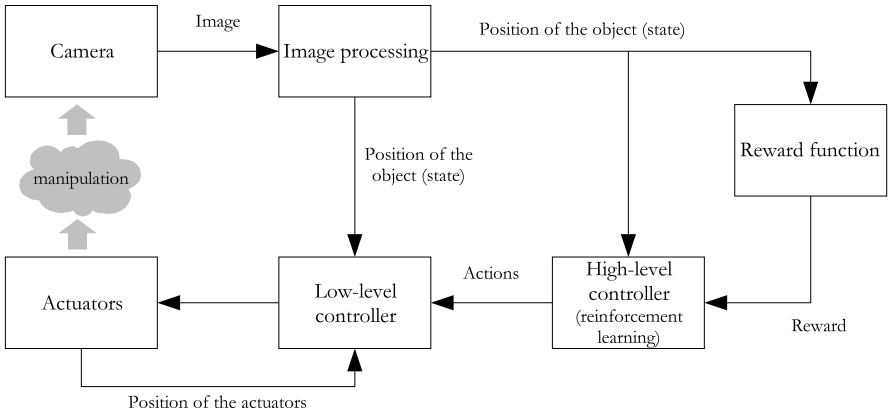


Figure 11.2. *Sensory-Motor Loop*

A high level action's duration (i.e. a push) is variable: from one to three seconds. At the end of a push, the object immediately stops because of its very weak inertia. Thus, only the static behavior of the object is taken into account. Its dynamic behavior is neglected.

11.2.4. *Representation of the Manipulation Task as an MDP*

11.2.4.1. *Definition of the State Space*

The system being considered as static, its state s is simply defined by the position (x, y) of the object's center in the video image.

The manipulation area having a diameter of 7 mm and the object a diameter of 4 mm, the position of the object's center can evolve within a disc with a diameter of 3 mm. Given the low resolution of the video sensor, the camera allows for localizing the center of the object in a circular area of diameter 131 pixels, i.e. about 13 500 different positions ($\mathrm{card}\,\mathcal{S} \approx 13\,500$).

11.2.4.2. *Definition of the Action Space*

In the conducted experiments, only 6 distinct actions have been used. Each of the three micro-tips can push the object over two determined distances: a "long" push of

1 mm — used for large displacements — or a "short" push of 100 µm — essential for fine positioning. We therefore have: $\mathrm{card}\,\mathcal{A} = 6$.

This set of 6 actions is the result of a compromise between the quality of the trajectories obtained and the learning time. More variety in these actions would probably allow for a faster manipulation but would considerably increase the size of the search space and therefore the overall learning time.

11.2.4.3. *Definition of the Reward Function*

The objective of the manipulation is to bring an object to a given position with a given precision. In the conducted experiments, the goal was to position the object at the center of the manipulation area with a 140 µm precision (6 camera pixels). The set of states located in this goal area is noted $\mathcal{S}_{goal}$.

Thus, the goal is to lead the process from any state to one of the states in $\mathcal{S}_{goal}$. The reward function is therefore defined as follows:

$$r(s,a,s') = \begin{cases} 1 & \text{if } s' \in \mathcal{S}_{goal}, \\ 0 & \text{otherwise.} \end{cases} \tag{11.1}$$

11.2.4.4. *Definition of an Episode*

Before each manipulation, the object is put in a random position in the manipulation area (i.e. the initial state is random). The manipulator is then controlled by the high level control algorithm, in this case a reinforcement learning algorithm, and an episodic task starts. The episode ends when the object has reached a goal state.

11.3. Choice of the Reinforcement Learning Algorithm

11.3.1. *Characteristics of the MDP*

The manipulation system being particularly slow (one push every one to three seconds), it is imperative to reduce to a minimum the number of episodes necessary to obtain a good policy. Furthermore, it is possible to perform many offline computations between two pushes.

The vision-based localization system provides a discrete observation of the system state. Considering the low resolution of the video sensor, there is a notable imprecision on the measure of the process state: it is not fully observable. Yet, the observation is sufficient to consider that the process is discrete, fully observable and slightly stochastic (quantization noise).

For these reasons, a discrete and indirect method (see Chapter 2) has been used so as to exploit as well as possible — during the duration of pushes — all the past interactions between the algorithm and the process. The most classical indirect algorithm is *Dyna-Q* [SUT 90] (see Chapter 2, Section 2.6.1). In its original version, *Dyna-Q* handles deterministic processes, which means that the same action in the same state always leads to the same next state. As a consequence, in a deterministic case, the

transition model can be represented as a matrix with $|\mathcal{S}|$ lines and $|\mathcal{A}|$ columns, each entry containing the next state. An extension of *Dyna-Q* to stochastic processes is immediate, but requires for each (s, a) pair a probability distribution over all next states, so that a $|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$ matrix is needed to store the probabilities. In our case, with 13 500 states and 6 actions, such a representation is not possible. Similarly, *Prioritized Sweeping* allows for learning a policy with a stochastic process but requires memorizing antecedence relations in the form of a matrix of $|\mathcal{S}| \times |\mathcal{A}|$ lines by $|\mathcal{S}|$ columns. Considering the number of states in the process to control, this method, although efficient, is not applicable either. For these reasons, a less memory consuming algorithm is used to control the manipulator.

11.3.2. *A Suitable Algorithm:* STM-Q

Algorithm 11.1: STM-Q (Short-Term Model-based Q-Learning).

initialization
  foreach $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    $Q(s, a) \leftarrow Q_0$
    $M(s, a) \leftarrow \emptyset$
    `/* M(s,a) is the queue of state-reward pairs observed during previous visits of the (s,a) pair */`

foreach *episode* **do**
  $s \leftarrow$`StateChoice`
  while s *is not a terminal state* **do**
    $a \leftarrow$`ActionChoice(`Q, s`)`
    Perform action a, observe new state s' and new reward r

    if $\operatorname{card} M(s, a) < n_{max}$ **then**
      Add observation (s', r) to queue $M(s, a)$
    else
      Replace oldest observation in queue $M(s, a)$ with observation (s', r)

    $s \leftarrow s'$

    repeat *N* **times**
      `/* this part can be performed offline */`
      Sample an already visited (s, a) pair at random
      $$Q(s, a) \leftarrow \sum_{\forall (y,r) \in M(s,a)} \frac{1}{\operatorname{card} M(s,a)} \left[r + \gamma \max_{v \in \mathcal{A}} Q(y, v) \right]$$

The algorithm being used, called *STM-Q* (*Short-Term Model-based Q-learning*), is an extension of *Dyna-Q* to weakly stochastic systems (see Algorithm 46) [ADD 05]. This is an indirect algorithm that looks for an optimal policy vis-à-vis the γ-weighted criterion. The STM-Q algorithm uses an intermediate representation between a single-record model as Dyna-Q does, and an exhaustive-record model.

As an indirect method, *STM-Q* builds a model of the transitions and the rewards during the interactions with the process. This model is made of a table of queues (FIFO) that store, for each visited state-action pair, the different outcomes observed in the past: each (s, a) pair is associated with a queue $M(s, a)$ whose maximum size cannot exceed a number defined *a priori*, noted n_{max}. Each element in $M(s, a)$ contains a pair made of the next state and the reward received while performing action a in state s. For example, if $n_{max} = 4$ and the pair (s, a) has been visited at least 4 times, one can have a queue like:

$$M(s,a) = \left\{(s'_{t_1}, r_{t_1}), (s'_{t_2}, r_{t_2}), (s'_{t_3}, r_{t_3}), (s'_{t_4}, r_{t_4})\right\} \tag{11.2}$$

with $t_1 < t_2 < t_3 < t_4$. These pairs obviously have different values if the process is not deterministic.

The resulting model allows for computing an estimate of the transition probabilities of the process:

$$\widehat{p}(s'|s,a) = \sum_{\forall (y,r) \in M(s,a) | y = s'} \frac{1}{\operatorname{card} M(s,a)}. \tag{11.3}$$

This estimate is used to optimize the value function $Q(s, a)$ with a process similar to a value iteration algorithm (see Chapter 1). The update equation is then the one appearing at the last line of Algorithm 46.

The size n_{max} of the waiting queue allows for adapting the algorithm to the "degree of determinism" of the process. For the sake of efficiency, n_{max} has to remain small (from about 10 to 50), so that *STM-Q* is rather suited to the control of weakly stochastic processes such as physical systems seen through digital measures and having a low-dimensional state space (dimension 2 to 3).

11.4. Experimental Results

11.4.1. *Experimental Setup*

As explained in Section 11.2, the objective is to select the pushes that will bring the object to a given position as fast as possible. The *STM-Q* algorithm has therefore been implemented in the high level controller (see Figure 11.2). It receives the process state *via* the vision system and sends instructions to the low level controller (choice of the tip and length of the push).

The experiment consists in learning to position an object at the center of a manipulation area from any position. At the beginning of each episode, the object is placed at random in the area and the episode terminates when the object is correctly positioned.

The exploration method retained for `ActionChoice` (Q, s) in the algorithm is the ϵ-greedy method (see Chapter 2).

Before the first episode, the value function is initialized with $Q_0 = 0$. During the experiment, the values of the algorithm's parameters are

– $\epsilon = 0.1$,

– $\gamma = 0.9$,

– $n_{max} = 10$,

– N equal to the number of previously visited distinct state-action pairs (over all episodes), i.e.:

$$N = card\left(\{(s,a) \in \mathcal{S} \times \mathcal{A} | M(s,a) \neq \emptyset\}\right). \tag{11.4}$$

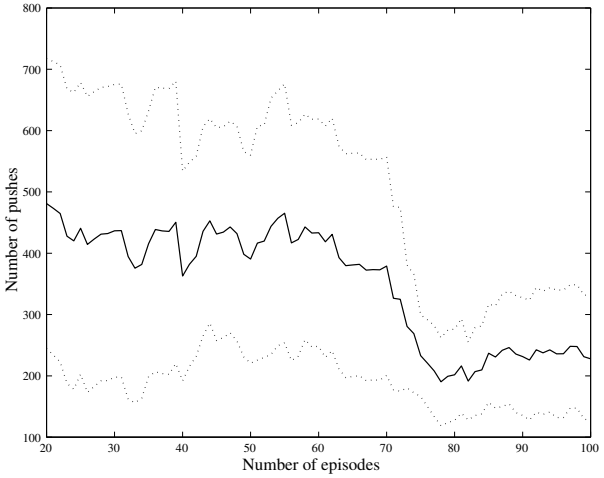


Figure 11.3. *Result of the on-line learning: the solid line shows the mean number of pushes per episode (sliding average over the last 20 episodes). The dotted line shows the standard deviation centered around the mean.*

11.4.2. *Results*

The experiment lasted for a little more than 24 hours for a total of 34 134 pushes spread of 100 episodes. The curve of Figure 11.3 represents the number of pushes (actions) performed per episode.

At the beginning of the learning, an average of 400 to 500 pushes are necessary to position the object. The standard deviation is very large (about 500). The policy consists mainly in exploring the state space.

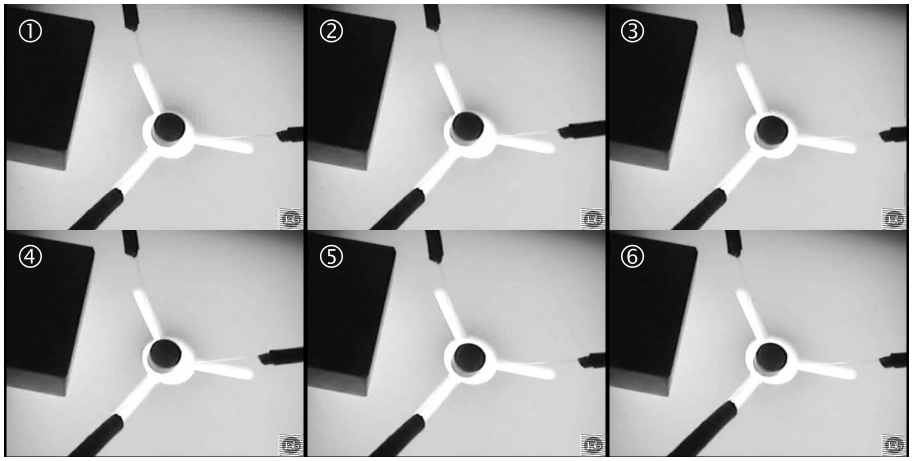
After 80 episodes, we observe a significant decrease of the episode length. An average 250 pushes are needed to position the object, but the standard deviation remains high (about 200). One can nevertheless obtain very short episodes as shown on Figure 11.4b. So, the STM-Q algorithm achieves to improve the initial random policy, but the performances are not uniform through the state space. When starting from some states, the learned policy is very efficient, and from other ones the policy is not.

Indeed, the observation of longer sequences shows that the control is particularly delicate in some states for two major reasons. On one side, when the object is between two micro-tips against the wall, it is difficult to dislodge it from this position. On the other side, to reach the objective it is not sufficient to move the object closer to the center: first, the object must be placed in the axis of a micro-tip, then, the micro-tips pushes the object straight to the goal; if the objective is missed from a short distance, it is often necessary to restart the manipulation "from scratch" by pushing the object back to the wall. These two observations appear in the value function. Figure 11.4b shows that the value function has a higher value on the perimeter than in the middle. It means that the border states are more visited than middle ones. The three peaks located on the micro-tips' axes at about one push from the target show that these area are near the goal in term of action. Finally, there is a ring of low Q-values close to the objective. As we observed, the closest states to the goal in term of distance are not close in term of action.

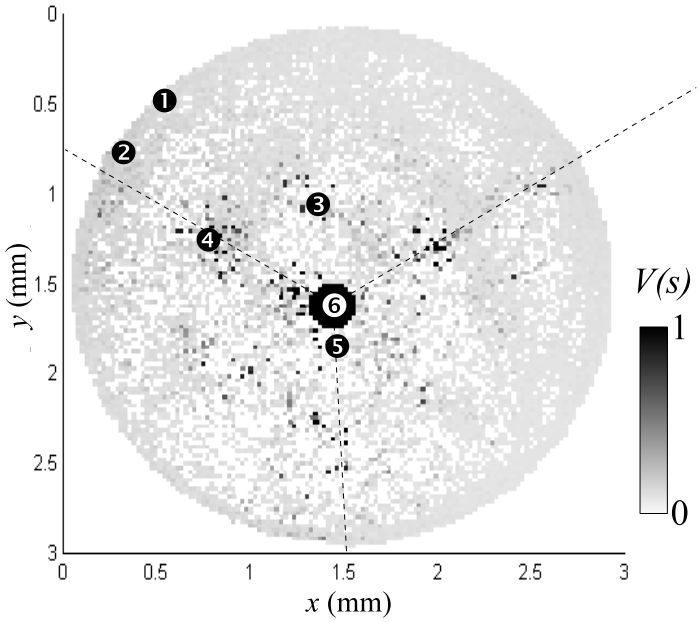
11.5. Conclusion

This experiment shows that it is possible to use an indirect reinforcement learning algorithm to learn on-line how to control a real-world process. Nevertheless, regarding the manipulator being studied, the results obtained are still far from a reliable and efficient automation. Admittedly, the manipulation task is hard even for a trained human operator. But these mixed results also confirm a classical need when learning: generalization.

The employed algorithm, *STM-Q*, builds a model of the process so as to update the value function as fast as possible, but does not generalize the value from one state to another close state, as illustrated by the very sparse aspect of the obtained Q-values (see Figure 11.4b). Moreover, visit statistics of the states (see Table 11.1) indicate that just over one out of eight state-action pairs have been visited during the experiment. As it is not reasonable to extend the already important duration of a learning session, a generalization mechanism would be necessary to reuse acquired experience for non-visited pairs. Thus, beyond this micro-manipulation application, learning on-line to control a real process requires facing a double challenge: use past experience at its



(a) Manipulation sequence: ① initial state, ② long push of the right micro-tip, ③ long push of the top micro-tip, ④ long push of the right micro-tip, ⑤ long push of the top micro-tip, ⑥ short push of the bottom micro-tip. The micro-tips are not very visible because they are fine and transparent (optical fibers).



(b) Representation of the value function v and of the successively visited stated during the manipulation sequence. The white points represent the non visited states.

Figure 11.4. *Example of manipulation episode obtained after learning*

best *via* indirect approaches and generalize acquired experience to adapt as well as possible to an unknown state. Ideas going in this direction are presented in Chapter 4.

	Total	Visited
Number of states	13 500	8 748
Number of state-action pairs	81 000	10 700
Number of actions per state	6	1 action in 7 138 states 2 actions in 1 138 states more than 3 actions in 375 states

Table 11.1. *State and action visit statistics after the on-line learning (which totals 34 134 actions)*

11.6. Bibliography

[ADD 05] ADDA C., LAURENT G. J., LE FORT-PIAT N., "Learning to control a real micropositioning system in the STM-Q framework", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'05)*, p. 4580–4585, April 2005.

[BAU 98] BAUR C., BUGACOV A., KOEL B., MADHUKAR A., MONTOYA N., RAMACHANDRAN T., REQUICHA A., RESCH R., WILL P., "Nanoparticle Manipulation by Mechanical Pushing: Underlying Phenomena and Real-Time Monitoring", *Nanotechnology*, vol. 9, p. 360–364, 1998.

[CHE 07] CHENG P., CAPPELLERI D. J., GAVREA B., KUMAR V., "Planning and Control of Meso-scale Manipulation Tasks with Uncertainties", *Proceedings of the 3rd Robotics: Science and Systems Conference*, 2007.

[GAU 06] GAUTHIER M., RÉGNIER S., ROUGEOT P., CHAILLET N., "Forces analysis for micromanipulation in dry and liquid environments", *International Journal of Micromechatronics, Special issue on Micro-handling*, vol. 3, num. 3, p. 389–413, 2006.

[GOR 06] GORMAN J. J., DAGALAKIS N. G., "Probe-Based Micro-Scale Manipulation and Assembly using Force Feedback", *Proceedings of the International Conference on Robotics and Remote Systems for Hazardous Environments*, p. 621–628, 2006.

[HAN 98] HANSEN T., KÜHLE A., SORENSEN A., BOHR J., LINDELOF P., "A Technique for Positioning Nanoparticles using an Atomic Force Microscope", *Nanotechnology*, vol. 9, p. 337–342, 1998.

[KOR 09] KORAYEM M. H., ZAKERI M., "Sensitivity analysis of nanoparticles pushing critical conditions in 2-D controlled nanomanipulation based on AFM", *International Journal of Advanced Manufacturing Technology*, vol. 41, p. 741–726, 2009.

[MOL 02] MOLL M., GOLDBERG K., ERDMANN M. A., FEARING R., "Orienting micro-scale parts with squeeze and roll primitives", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'02)*, p. 11–15, 2002.

[RES 00] RESCH R., LEWIS D., MELTZER S., MONTOYA N., KOEL B., MADHUKAR A., REQUICHA A., WILL P., "Manipulation of Gold Nanoparticles in Liquid Environnements using Scanning Force Microscopy", *Ultramicroscopy*, vol. 82, p. 135–139, 2000.

[SIT 04] SITTI M., "Atomic force microcope probe based controlled pushing for nanotribological characterization", *IEEE/ASME Transactions on Mechatronics*, vol. 9, num. 2, p. 343–349, 2004.

[SUT 90] SUTTON R. S., "Integrated Architectures for Learning, Planning and Reacting Based on Approximating Dynamic Programming", *Proceedings of the 7th International Conference on Machine Learning (ICML'90)*, San Mateo, CA, p. 216–224, 1990.

[ZES 98] ZESCH W., S.FEARING R., "Alignment of Microparts Using Force Controlled Pushing", *Proceedings of the SPIE Conference on Microrobotics and Micromanipulation*, vol. 3519, Boston, Massachusetts, p. 148–156, November 1998.