The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Distributed control architecture for smart surfaces

K. Boutoustous, G. J. Laurent, E. Dedu, L. Matignon, J. Bourgeois and N. Le Fort-Piat

Abstract— **This paper presents a distributed control architecture to perform part recognition and closed-loop control of a distributed manipulation device. This architecture is based on decentralized cells able to communicate with their four neighbors thanks to peer-to-peer links. Various original algorithms are proposed to reconstruct, recognize and convey the object levitating on a new contactless distributed manipulation device. Experimental results show that each algorithm does a good job for itself and that all the algorithms together succeed in sorting and conveying the objects to their final destination. In the future, this architecture may be used to control MEMS-arrayed manipulation surfaces in order to develop *Smart Surfaces*, for conveying, fine positioning and sorting of very small parts for micro-systems assembly lines.**

I. Introduction

Distributed manipulation has drawn attention in several recent studies, particularly in the micro scale area with Micro-Electro-Mechanical Systems (MEMS) actuator arrays. In robotics, we use the term of "distributed manipulation" for devices which can induce motion on objects through the application of force at many points at the same time. This approach offers new possibilities with many advantages in macro as well as in micro scale: many small inexpensive mechanisms can move large heavy objects (with regard to their scale) and can handle several objects at the same time. The manipulation device is also flexible and scalable by adding/suppressing elements. Moreover it is highly robust to failures due to redundancy of mechanisms.

Distributed manipulation has been a very active topic since the 1990's. These pioneer researches have developed different types of distributed manipulators, based on servoed roller wheels [1], cilia actuators [2], [3], [4], [5], [6], suction nozzles with air-hockey tables [7], [8], [9] and directed air-jets [10], [11], [12]. All these works require a centralized control and therefore are not scalable.

Some of these preliminary studies use sensorless manipulation schemes based on Goldberg's algorithm [13] for parallel jaw grippers. The jaw grippers are obtained with actuator arrays by creating opposite field forces which then can orient and move the parts. Böhringer *et al.* [14] have proposed a programmable vector field method based on Goldberg's algorithm. This sensorless scheme is well-adapted for contact manipulation arrays but has shown some limitations when applied to contactless manipulators [15]. For instance, the absence of closed-loop control has led to undamped oscillations and unwanted behaviors.

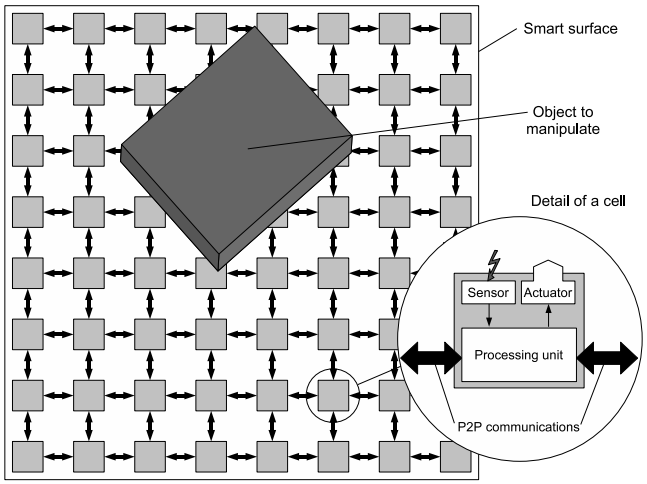


Fig. 1. The Smart Surface concept.

Current researches aim to include sensors and to add intelligence to distributed manipulators, work which has never been done. The long-term objective is to design and develop fully integrated distributed MEMS devices that it is called *Smart Surface*, for conveying, fine positioning and sorting of very small parts. For instance the goal could be to sort rectangular and circular parts, by sending rectangles to north border of the surface, and circles to the south one. Before being conveyed to the right position, the parts should be first recognized. Fully integrated means that the control and the part recognition must be embedded and decentralized.

In this paper, we propose a distributed control architecture able to perform part recognition and closed-loop control of distributed manipulation devices (see Figure 1). This architecture is composed of decentralized cells. Each cell is linked to a local sensor and to an actuator. A communication network allows each cell to exchange messages with its direct neighbors. Using the adjacent network, the cells work autonomously and in a distributed manner by exchanging their local information of the part. The main contributions of this paper lie in the following facts:

- A new distributed control architecture for smart surfaces using peer-to-peer communication is introduced;
- Distributed algorithms to reconstruct, differentiate and

This work was supported by the Smart Surface NRA (French National Research Agency) project (ANR_06_ROBO_0009).

K. Boutoustous, E. Dedu and J. Bourgeois are with Laboratoire d'Informatique de l'Université de Franche-Comté, Montbéliard, France, {Firstname.Lastname}@pu-pm.univ-fcomte.fr

G. J. Laurent, L. Matignon and N. Le Fort-Piat are with the Automatic Control and Micro-Mechatronic Systems Department, FEMTO-ST Institute, UFC-ENSMM-UTBM-CNRS, Université de Franche-Comté, Besançon, France, {Firstname.Lastname}@ens2m.fr

L. Matignon is with the GREYC, University of Caen Basse-Normandie, France, laetitia.matignon@info.unicaen.fr

move the parts are proposed;
- The distributed control architecture is validated on a new contactless distributed manipulation device.

II. Smart Surface Overview

A. The Smart Surface Concept

As referred previously, Smart Surfaces have to take into account functions such as recognition, conveyance and positioning of an object. The implementation of these functions must also meet the requirement of scalability, modularity and robustness of distributed manipulation systems. Moreover, since the objects are small compared to sensors and can be rotated, classical recognition methods such as neural networks are not appropriate.

Following Chapuis [16], we consider that the global system can be composed by a large number of cells, each cell being able:

- to sense locally the state of the object (for example the presence/absence of the object);
- to act locally on the object;
- to decide its action by itself.

To achieve complex tasks as shape recognition or complex manipulations, local sensing is not sufficient. So, we propose to implement a communication network in order to reconstruct higher-level observations of the system.

Figure 1 illustrates the Smart Surface concept. Each cell receives measures from its sensor, acts on the object by its actuator and is able to communicate with its four neighbors. A programmable processing unit is able to perform recognition tasks and calculation of local control laws.

Due to scalability, robustness to failures and even simplicity, we have conceived the communication network as a peer-to-peer network. The network consists of links between each two direct neighbors (4-connectivity). The communication must be faster than the sampling period. On the other hand all the measures must be done at the same time t at each sampling period Te.

B. The Distributed Control Architecture

Figure 2 details the distributed control architecture. The state $b_i(k)$ of the sensors is received by part-reconstruction modules. These blocks communicate with their neighbors in order to reconstruct the image of the object. Then, the differentiation functions use the reconstructed image to recognize the shape of the object and to give the proper information $s(k)$ to the controller blocks. Then, the motion controllers independently allocate a state $u_i(k)$ to each actuator i so as to generate a specific motion of the object. The functions of reconstruction, differentiation and control are implemented following a fully distributed scheme.

III. Part Reconstruction and Differentiation

The part recognition is done inside the "Part reconstruction and differentiation" block, in two stages (see figure 2):

1) The offline stage is executed before using the Smart Surface and can be done on a computer with a camera taking images of the models. A model is the shape of one of the objects, i.e. micro-parts, which can be placed on the smart surface. It allows to construct a database in which each model is characterized by a set of criteria values, such as surface and perimeter, as defined in [17].
2) The online stage allows to differentiate an object placed on the Smart Surface. Differentiation is performed after extraction of the binary representation of the object. It aims to find out the model which corresponds to the object which is on the Smart Surface.

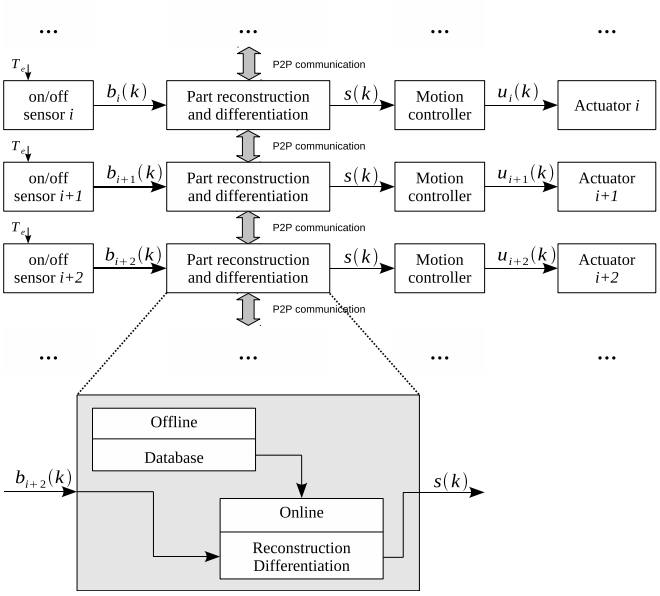


Fig. 2. The distributed control architecture.

A. Offline stage

This stage allows to associate to each model of the Smart Surface a set of criteria values. It consists of five phases, depicted in Figure 3:

1) all rotations of 1° are generated, $MRot_i$, from the image of the model M_i, given as input, with respect to the centre of the image, and all translations of $width(MRot)/10$ pixels are generated from the image $MRot_i$;
2) a grid corresponding to the positions of the sensors (middle point of the cell) is drawn on the images;
3) the images are discretized in a matrix by affecting 1 if the sensor is covered by the object and 0 otherwise (in order to have negligible errors for the rotation, the resolution of the image should be much greater than the resolution of the surface);
4) matrices corresponding to the initial matrix without the rows and columns that contain only zeros are generated; these matrices will be called *masks* in the following, and only unique masks will be taken into account;
5) the values of each criterion are calculated for all masks of the model.

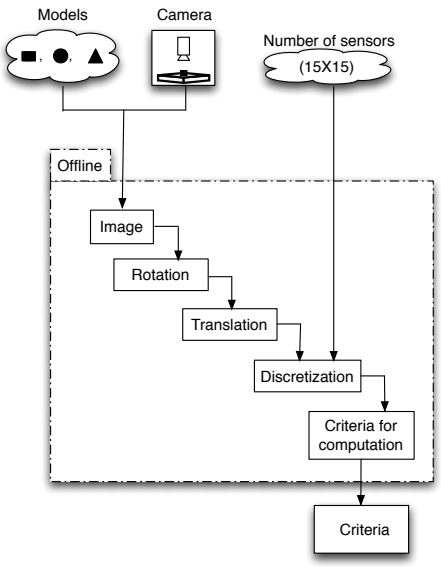


Fig. 3. Description of the process executed in the offline phase.

The following algorithm details these 5 phases, with M_i being the models of the Smart Surface, Im the image of the model, $ImRot$ and $ImTrans$ the rotated image and the translated image respectively, d the rotation step and x, y the translation steps:

for size of sensors grid $(15, 15)$ **do**
  for each $M_i \subset M_1, M_2, M_3, ..., M_{nbr\_models}$ **do**
    Acquire the image Im of the model on the surface
    for $d = 1^\circ$ to 360° **do**
      Generate $ImRot$ by rotating the image Im by d degrees
      for each $y \subset 0, 10, 20, 30, ..., Size(ImRot)$ **do**
        for each $x \subset 0, 10, 20, 30, ..., Size(ImRot)$ **do**
          Generate $ImTrans$ by translating the image $ImRot$ by x steps on Ox and y steps on Oy
          Discretize the image $ImTrans$
          Generate the mask
          Calculate and save the value for each criterion
        end for
      end for
    end for
  end for
end for

In the set of all criteria, only criteria with a good differentiation rate, low cost memory and fast execution time [17] will be used in online stage. In the end, these criteria values form the database to be used as input in the online stage.

B. Online stage

The aim of this stage is to differentiate, in real-time, an object on the Smart Surface using criteria values previously calculated in the offline stage. But to do this, we must first reconstruct the global view of the object, then apply the differentiation algorithm to differentiate the object. The online stage is divided in two phases, reconstruction phase and differentiation phase, detailed in the following.

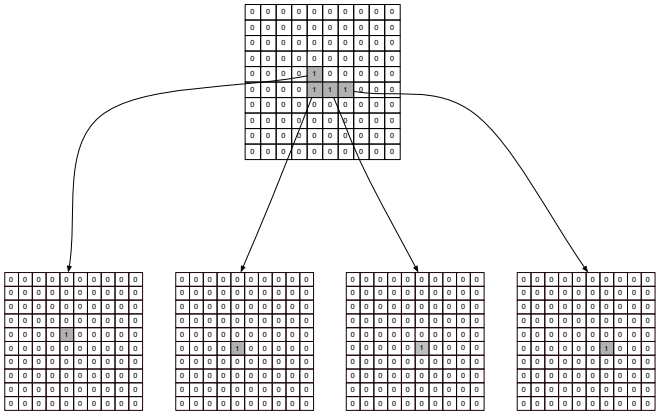


Fig. 4. The binary matrix of some cells during the first communication step.

1) Reconstruction phase: Once the object is on the Smart Surface, each cell knows its status (if it is covered by the object or not), but it does not know the status of its neighbors. The cells covered by the object are called *active cells*. Each active cell does an iterative process and aims to reconstruct the binary representation of the object. This iterative process consists in two steps: communication and computation steps.

a) Communication step: In this step, all active cells are communicating with their neighbors. This communication is done by sending a message. Each active cell will send a message that consists of a matrix of bits, the matrix size is the same as the smart surface size[1] (1 bit per cell). Initially, the matrix of each cell is filled with 0, except for the bit corresponding to itself, which is set to 0 if no object is seen by the sensor, and 1 if the object covers the sensor; in fact, in the first communication all the active cells know only their status and nothing about the state of their neighbors (see Figure 4).

Throughout the re-iteration of the communication phase, all the active cells will extend their state thanks to the received matrices from their neighbors, until they reach a global view of the object.

b) Computation step: The computation step consists in updating the view of each cell according to the views collected from its neighbors in the communication step. This update consists in applying the union operator between all received matrices and itself one. Formally, let M_{Cell} be a matrix corresponding to the view of cell *Cell* and $M_{1,1}$, $M_{2,1}$, $M_{3,1}$, $M_{1,2}$, $M_{3,2}$, $M_{3,1}$, $M_{3,2}$ and $M_{3,3}$ the matrices corresponding to the 8 neighboring views respectively. The updated matrix of the cell *Cell* is obtained by applying the equation 1:

$$M_{Cell} = \bigcup_{i=-1,j=-1}^{i=1,j=1} M_{i,j} \quad (1)$$

Figure 5 shows, as example, the matrix of the four active cells during each computation step.

[1]In order to reduce memory footprint, it is possible to only use a rectangle of size maximum mask, plus a few lines/columns to avoid oversize errors.

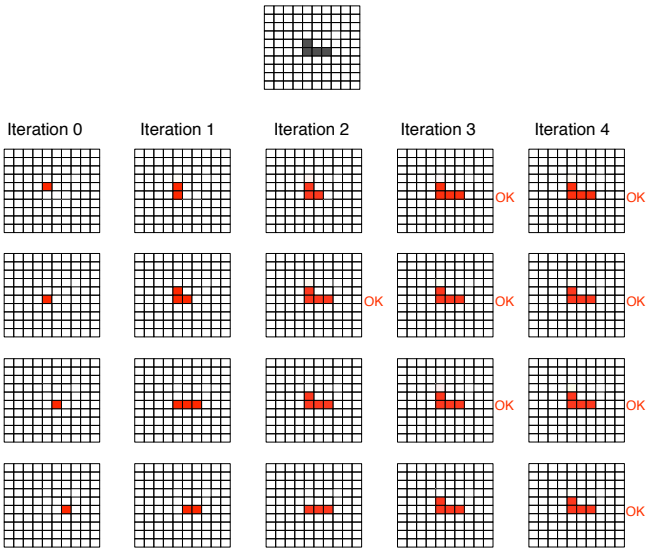


Fig. 5. Example of the matrix of the four active cells during each computation phase.

To sum up, during the reconstruction phase all the cells execute simultaneously (the time is discrete) the following steps:

1: **repeat**
2: Send its matrix to the neighbors
3: Receive the matrices from its four neighbors
4: Do a union bit by bit among its matrix and the four matrices received
5: Store the result in its own matrix
6: **until** NbIterations $>$ MaximumSize

It can be proved that after N steps (N is the height plus the width of the object), this algorithm converges and each cell has the binary representation of the object currently on the smart surface. The number of steps is therefore not dependent on the smart surface size.

2) Differentiation phase: Once each cell has the binary representation of the object, the criteria are computed. These values are compared to the criteria values for each model, computed in the offline stage. If there is a criterion or a combination of criteria which uniquely identifies the object, then the object is differentiated. If not, then the object is moved and the online stage restarts.

The algorithm is repeated for each new image sent by the camera that films the moving object on the Smart Surface.

The differentiation phase can be summarized by the following algorithm:

1: **repeat**
2: Calculating the values of criteria
3: Comparison of criteria values calculated with the models in the database
4: Save the differentiation result
5: **until** differentiation result $>$ threshold
6: **if** object is differentiated **then**
7: Send the differentiation result to the control block
8: **else**
9: Wait for the object to move to another place and run the process again
10: **end if**

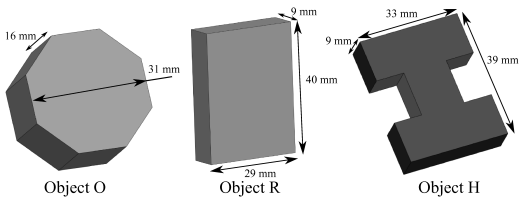


Fig. 6. The three different objects to sort.

This finishes the differentiation stage. The differentiation block sends to the control block the information of the differentiation result and the position of the object (coordinates of the bounding rectangle) on the Smart Surface. The control block takes the responsibility from now on, either to give the object a small movement in order to retry differentiation or to move the object to its destination.

IV. Motion Control

The application of the Smart Surface is to sort various objects which are successively placed on the distributed-air-jet manipulator. Each kind of object must be conveyed to the right position to be collected. For instance, we chose a set of 3 different objects to carry out the experiments (see Figure 6). The O and R objects must be sent to the east edge of the manipulator, the H object to the west edge. This task could fit for example in the feeding of assembly line workstations.

As explained in the previous section, as soon as the object is not discriminated, it is maintained (while moving) in the center of the manipulator. Once it is recognized, the control corresponding to the type of the object is applied. Finally, when the object is collected outside the manipulator, another one is placed on the manipulator and the process restarts.

The differentiation stage returns the type of the object and the coordinates of its bounding rectangle. This information is sent to each independent controller at intervals of time T (sampling period) and is noted $s(k)$ at time step k (see Figure 2). $s(k)$ contains five discrete values:

- $x_{min}(k)$ the abscissa of the western active sensor;
- $x_{max}(k)$ the abscissa of the eastern active sensor;
- $y_{min}(k)$ the ordinate of the the northern active sensor;
- $y_{max}(k)$ the ordinate of the southern active sensor;
- $t(k)$ the type of the object (0 if not discriminated yet, 1 for H object, 2 for R object and 3 for O object).

Then each controller sends a command to its associated valve. The control signal sent to the ith valve is noted $u_i(k)$ and can take two discrete values (1 for opening and 0 for closing).

The goal of the control is to send the object toward a direction according to the type of the object. The direction is coded by 1 for east, -1 for west and 0 if the object must not be conveyed at the moment. The direction is noted $d(k)$

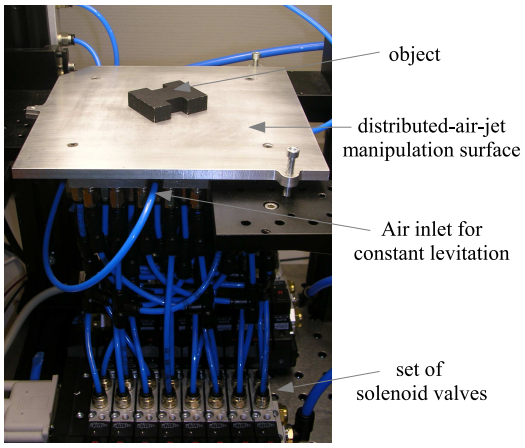


Fig. 7. The distributed-air-jet manipulation surface.

and is defined by:

$$d(k) = \begin{cases} -1 & \text{if } t(k) = 1 \\ 0 & \text{if } t(k) = 0 \\ 1 & \text{if } t(k) = 2 \text{ or } t(k) = 3 \end{cases} \quad (2)$$

More complex tasks could be achieved. Actually, the objective is to conduct the proof-of-concept of our distributed control architecture as the following experiments show it.

V. Experimental Results

A fully integrated approach using MEMS still remains rigid and costly in fabrication. So to test and validate the Smart Surface concept, we designed a prototype of pneumatic surface with conventional technologies. This distributed-air-jet manipulation surface associated to a software emulating the distributed architecture helps to conduct the proof-of-concept of future Smart Surfaces. Our basic idea behind emulation is to develop programs (emulators) that can be run on current software computing platforms to validate the functions and behaviors of "virtual" hardware distributed cells.

A. The Distributed-Air-Jet Manipulation Surface

The distributed-air-jet manipulation surface is a $120 \times 120\ mm^2$ square surface upon which an object is moving in aerodynamic levitation (see Figure 7). The device consists of 3 parts detailed in the exploded view of Figure 8:

- The upper-block is drilled of 15×15 orifices; each orifice is 0.4 mm in diameter;
- The lower-block is drilled of 112 holes in staggered rows; These holes connect one hole out of two of the upper-block to independent air inlets; between holes a network of diagonal channels connects the other holes of the upper-block to a common air inlet located on the side of the lower-block.

The object is maintained in constant levitation thanks to the airflow that comes through the lateral common air inlet. The airflow spreads over the network of diagonal channels and then through one hole out of 2 of the upper-block.

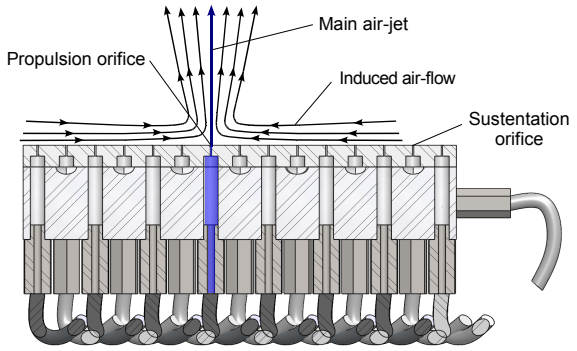


Fig. 8. Cross view of the distributed-air-jet manipulation surface with one opened valve.

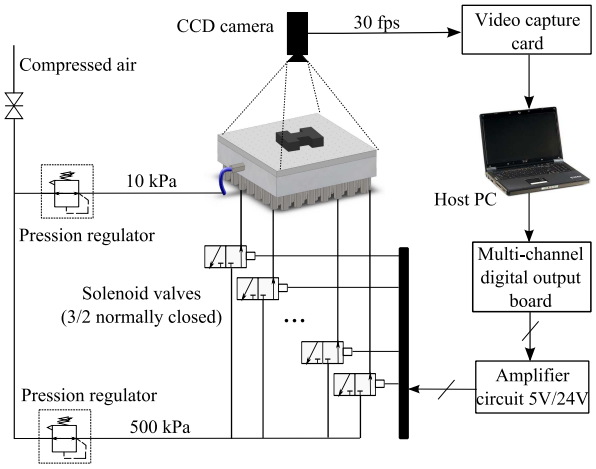
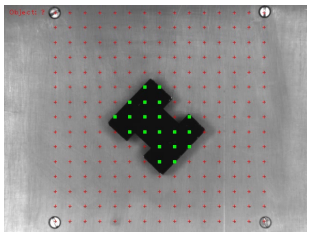


Fig. 9. Complete hardware configuration.

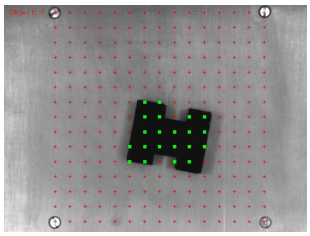
An object can be moved by generating strong vertical air-jets through the others holes of the surface. Each orifice is driven by an independent solenoid valve (3/2 normally closed valve). When a valve is open, air-flow comes through both blocks and generates a vertical air-jet on the front side of the upper-block. The air-jet creates an induced air flow that pulls the object toward the orifice (see Figure 8).

The experimental setup for the distributed-air-jet manipulator is composed of air pressure control systems, the set of solenoid valves and its control system, and computer vision processing. Figure 9 describes the complete hardware configuration. More details on the working of this device can be found in [18].

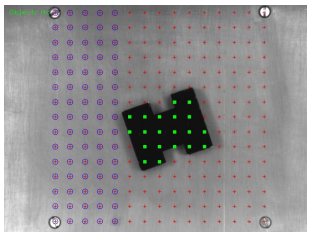
For conducting the proof-of-concept of a future Smart Surface composed of distributed cells integrating sensors, we used a camera to emulate the cells sensors in order to do the offline and online stages of the first phase. The camera is placed above the distributed-air-jet manipulator and provides the top view of the entire surface. The image processing calculates the state $b_i(k)$ of every emulated sensor i for each frame k based on the grey level of pixels. $b_i(k)$ is one if the object is above the sensor i and zero elsewhere. The number of sensors of the grid was determined beforehand



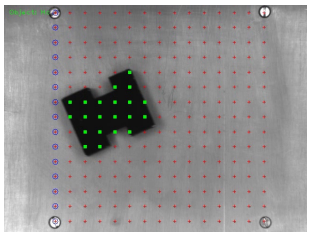
(a) t=0.4s, the object is not identified



(b) t=2.4s, the object is still not identified



(c) t=3.4s, the object is identified and the valves on its left are opening



(d) t=4.1s, the object is moving toward the west edge of the surface

Fig. 10. Image sequence of an experiment with the H object (dark crosses represent inactive sensors, light squares represent active sensors and dark circles represent opened valves).

by a Sensor Network Calibrator and the size of this grid was set to (15×15) sensors [19]. Figure 10 shows the working of the sensor emulation.

B. Motion Control

For this task, the control laws are quite simple. To carry the object to a direction, the valve in this direction must be opened. We note with x_i the abscissa of the air-jet linked to the valve i. The control law of an air-jet is defined by:

$$u_i(k) = \begin{cases} 1 & \text{if } d(k) = 1 \text{ and } x_i > x_{max}(k) \\ 1 & \text{if } d(k) = -1 \text{ and } x_i < x_{min}(k) \\ 0 & \text{elsewhere} \end{cases} \quad (3)$$

C. Experimental Results

Figure 10 shows an image sequence extracted from a video made during the experimental manipulation. First, an object is placed on the surface. The distributed architecture observes several images of the object before identifying it. Finally, when the object is identified (H in this case), the controllers open the correct valves in order to convey the object in the desired direction.

Figure 11 shows the trajectory (abscissa projection) of three different objects placed successively on the surface. These results can be further appreciated in the video clip accompanying this paper[2].

Before conveying the object, it must be recognized (differentiated) using emulated sensor values (which give a binary matrix). The differentiation is obtained by calculating criteria values from a set of values obtained from emulated sensors. These criteria values will be compared to those from the database.

[2]Also available at http://www.femto-st.fr/~guillaume.laurent/

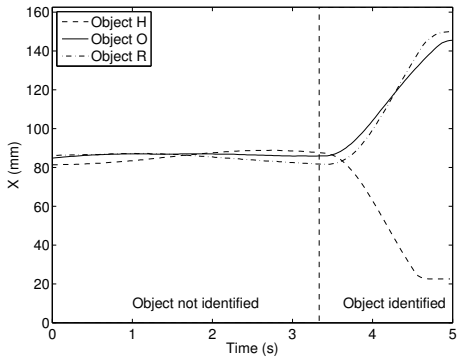


Fig. 11. Abscissa of the center of the object according to time for three experiments with different parts.

We assume that an object is moving on the surface and, as it is moving, the measurements are received from the emulated sensors. While this happens, we try to differentiate the object.

In these experiments we use a set of 17 criteria sorted by their relevance for differentiation. When an object is on the surface, we calculate the values of the first criterion and compare it to the values of the same criterion from our database. If we successfully differentiate the object (as being of type R, O or H), the process stops, otherwise it passes to the following criterion. If no criterion is successful, the object is called not differentiated *NoDiff*.

Sometimes, the object can be wrongly differentiated. To cope with this, we use a predefined threshold and, when a type exceeds this threshold, we consider the object as being of that type.

Figure 12 shows the differentiation result obtained for the distributed architecture by observing several postures of the object before identifying it. The differentiation threshold is arbitrarily set to 60%. In each figure we notice that the differentiation threshold greater that 60% gives the same result as for H, O and R.

During object moving, it is possible that between two sets of values received from the emulated sensors, the binary representation of the object does not change. That's why, we are also interested in having different binary representations of objects (*DiffBR*), representing the values received from the emulated sensors.

In the same figure it can be noticed that after 3.3 seconds, the three objects have different *DiffBR*: the object H has 45 *DiffBR*, while O and R have only 10 and 17 respectively.

Note that in these experiments we do not consider the execution time, because our aim is not to obtain a fast differentiation but only to differentiate an object using a set of values obtained from emulated sensors.

The process is highly reproducible and robust. We performed dozens of experiments for each of the objects. It can be noticed that for each experiment, the object was successfully identified and conveyed.

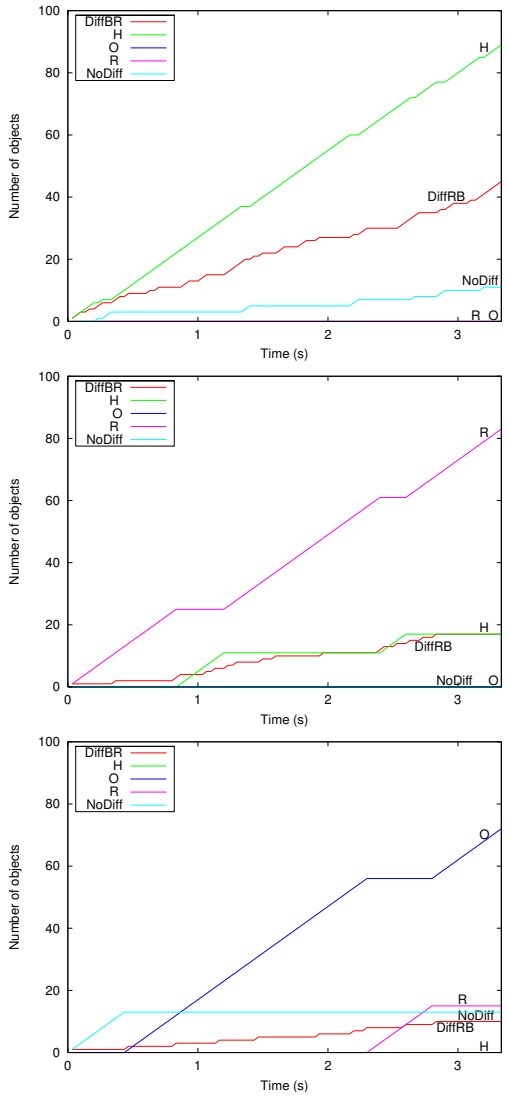


Fig. 12. Differentiation result for experiments when the H, O, and R object respectively is put on the Smart Surface.

VI. Conclusion and Future Works

The paper presents a distributed control architecture for a smart surface used to sort different micro-parts. The surface is a grid of cells, each cell containing a sensor, an actuator and a processing unit. This distributed architecture is validated on a real manipulation surface. Results show that, thanks to the algorithms used, this specific architecture is really able to recognize the micro-parts, and to convey them to the desired border of the Smart Surface.

One of the ideas of our future work is to test the distributed architecture with less different shapes than O, H and R, and to perform more complex tasks such as the sorting of two objects simultaneously. The second objective is to increase recognition speed and to implement this distributed architecture in a decentralized hardware circuit.

Acknowledgements

The authors gratefully acknowledge Joël Agnus and David Guibert from the FEMTO-ST Institute for their technical assistance.

References

[1] J. E. Luntz, W. Messner, and H. Choset. Distributed manipulation using discrete actuator arrays. *The Int. Journal of Robotics Research*, 20(7):553–583, 2001.

[2] 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 Trans. on Mechatronics*, 14:119–125, 2009.

[3] K.-F. Böhringer, B. R. Donald, and N. C. MacDonald. Single-crystal silicon actuator arrays for micro manipulation tasks. In *IEEE Int. Workshop on Micro Electromechanical Systems*, pages 7–12, 1996.

[4] J. W. Suh, R. Bruce Darling, K.-F. Böhringer, B. R. Donald, H. Baltes, and G. T. A. Kovacs. Cmos integrated ciliary actuator array as a general-purpose micromanipulation tool for small objects. *Journal of Microelectromechanical Systems*, 8(4):483–496, 1999.

[5] G. Bourbon and P. Minotti. Toward smart surfaces using high-density arrays of silicon-based mechanical oscillators. *Journal of Intelligent Material Systems and Structures*, 10:534–540, 1999.

[6] C. Liu, T. Tsao, P. Will, Y.C. Tai, and W.H. Liu. A micromachined permalloy magnetic actuator array for micro robotics assembly systems. In *Int. Conf. on Solid-State Sensors and Actuators*, 1995.

[7] P.-J. Ku, K. T. Winther, and H. E. Stephanou. Distributed control system for an active surface device. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 3417–3422, 2001.

[8] J. Luntz and H. Moon. Distributed manipulation with passive air flow. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 195–201, 2001.

[9] K. Varsos and J. Luntz. Superposition methods for distributed manipulation using quadratic potential force fields. *IEEE Trans. on robotics*, 22(6):1202–1215, 2006.

[10] A. Berlin, D. Biegelsen, P. Cheung, M. Fromherz, D. Goldberg, W. Jackson, B. Preas, J. Reich, and L.-E. Swartz. Motion control of planar objects using large-area arrays of mems-like distributed manipulators. *Micromechatronics*, 2000.

[11] S. Konishi and H. Fujita. A conveyance system using air flow based on the concept of distributed micro motion systems. *IEEE Journal of Microelectromechanical Systems*, 3(2):54–58, 1994.

[12] Y. Fukuta, Y.-A. Chapuis, Y. Mita, and H. Fujita. Design, fabrication and control of mems-based actuator arrays for air-flow distributed micromanipulation. *IEEE Journal of Microelectromechanical Systems*, 15(4):912–926, 2006.

[13] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2-4):210–225, 1993.

[14] K.-F. Böhringer, V. Bhatt, B. R. Donald, and K. Y. Goldberg. Algorithms for sensorless manipulation using a vibrating surface. *Algorithmica*, 26(3-4):389–429, 2000.

[15] Laëtitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Designing decentralized controllers for distributed-air-jet mems-based micromanipulators by reinforcement learning. *Journal of Intelligent and Robotic Systems*, 2010.

[16] Y.-A. Chapuis, Y. Fukuta, Y. Mita, and H. Fujita. Autonomous decentralized systems based on distributed controlled mems actuator for micro conveyance application. *Journal of Institute of Industrial Science, University of Tokyo*, 56(1):109–115, 2004.

[17] K. Boutoustous, E. Dedu, and J. Bourgeois. An exhaustive comparison framework for distributed shape differentiation in a MEMS sensor actuator array. In *International Symposium on Parallel and Distributed Computing (ISPDC)*, pages 429–433, Krakow, Poland, July 2008. IEEE computer society press.

[18] A. Delettre, G. J. Laurent, and N. Le Fort-Piat. A new contactless conveyor system for handling clean and delicate products using induced air flows. In *IEEE Int. Conf. on Intelligent Robots and Systems*, 2010.

[19] K. Boutoustous, E. Dedu, and J. Bourgeois. A framework to calibrate a MEMS sensor network. In *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing*, volume 5585 of *LNCS*, pages 136–149, 2009.