Trajectory planning for micromanipulation with a non-redundant digital microrobot: shortest path algorithm optimization with hypercube graph representation

Vincent Chalvet, Yassine Haddab, Philippe Lutz

▶ **To cite this version:**

Vincent Chalvet, Yassine Haddab, Philippe Lutz. Trajectory planning for micromanipulation with a non-redundant digital microrobot: shortest path algorithm optimization with hypercube graph representation. Journal of Mechanisms and Robotics, 2016, 8 (2), pp.021013 (9). hal-02868185

HAL Id: hal-02868185
https://hal.science/hal-02868185

Submitted on 15 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Trajectory planning for micromanipulation with a non-redundant digital microrobot: shortest path algorithm optimization with a hypercube graph representation

Vincent Chalvet
FEMTO-ST Institute, AS2M dpt.

Yassine Haddab
FEMTO-ST Institute, AS2M dpt.

Philippe Lutz
FEMTO-ST Institute, AS2M dpt.
Univ. Bourgogne Franche-Comté, Univ. de Franche-Comté / CNRS / ENSMM
24 rue Savary, F-25000 Besançon, France.

Microrobotics is an ongoing study all over the world for which design is often inspired from macroscale robots. We have proposed the design of a new kind of microfabricated microrobot based on the use of binary actuators in order to generate a highly accurate and repeatable tool for positioning tasks at microscale without any sensor (with open-loop control). Our previous work consisted in the design, modeling, fabrication and characterization of the first planar digital microrobot. In this paper we focus on the motion planning of this robot for micromanipulation tasks. The complex motion pattern of this robot requires the use of algorithms. Graph theory is well suited for the discrete workspace generated by this robot. The comparison between several well-known trajectory planning algorithms is done. A new graphical representation, named the hypercubic graph, is used for improving the computation speed of the algorithm. This is particularly useful for large workspace robots.

1 Introduction

Accurate micromanipulation tools were developed during the last decade in order to manipulate micrometric sized objects, either biological ones as in [1] or artificial ones as in [2] or [3]. Many studies such as [4] or [5] focused on the development of microtweezers able to handle and even position micro-objects . They now include several sensors (position sensors and force sensors such as in [6] and [7]) in order to manipulate these micro-objects with high accuracy and high security (preventing micro-objects deterioration).

The design of holder microrobotic structures were however not studied as much as the microtweezers were. Their purpose it to position the microtweezer inside the micromanipulation scene, and are mainly inspired from traditional macroscale robotic structures. The use of traditional joints in those robotic architectures generates the same drawbacks as in macroscale robotics, such as friction, wear, backlash, etc. Accuracy loss resulting from those drawbacks becomes troublesome for manipulation tasks at microscale. For that reason, flexible joints were studied in [8] and in [9] for the design of complex positioning structures. Active materials are well suited as actuators in this kind of structure for micromanipulation tasks ([10], [11]) thanks to their sub-micrometric resolution. Their highly non-linear behavior is however difficult to handle, making the design of efficient controllers a hard task as shown in [12]. Sensors are also needed during micromanipulation tasks. They are expensive for the required measurement resolution, and bulky for micromanipulation tasks and thus are not well suited for applications in confined environment.

To overcome several of these problems a new paradigm in the design of microrobots was used. It is inspired from digital robotics, developed in [13] and deeply studied in [14] and [15], which can generate highly accurate positioning without any sensor. These particular robots make use of binary actuators that generate a displacement between two positions (named state 0 and state 1), with very high accuracy, high repeatability and high robustness characteristics. In addition to the force generated by this kind of robot, its open-loop control is well adapted for micromanipulation tasks in confined environment.

Our previous papers have presented the paradigm of digital robotics applied to microscale usage. [16, 17] were dedicated to the design and fabrication of the binary actuator (the bistable module), which was used in the first digital microrobot presented in [18, 19], named the DiMiBot.

This robot generates a discrete planar workspace in which all the reachable positions are homogeneously distributed. Despite its numerous advantages, such as high accuracy and passive position maintaining, the bistable property of the modules can make the navigation inside that non-

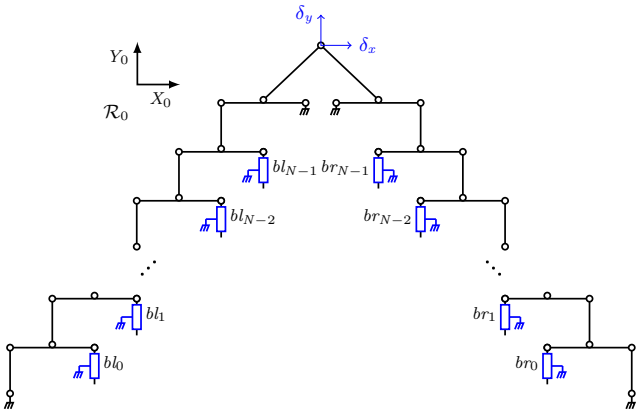


Fig. 1: Kinematic structure of the DiMiBot, with $2N$ bistable modules.

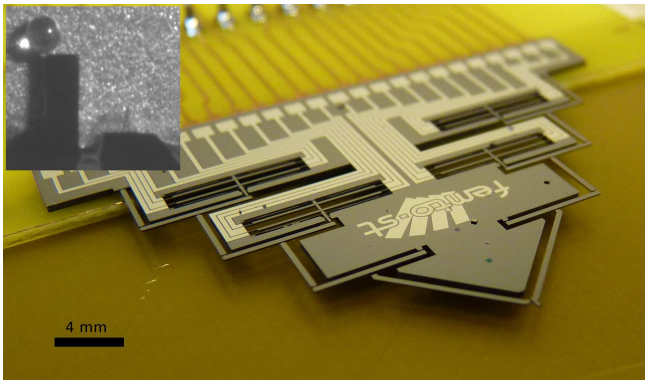


Fig. 2: Digital microrobot containing 4 bistable modules ($N = 2$), and a zoom view of the end-effector while manipulating a 150 μm diameter glass ball.

redundant workspace troublesome, even in teleoperation as it is usually done nowadays. In order to precisely control the trajectory of the robot's end-effector, only one bistable module can be switched at a time. As each bistable module has its own effect on the end-effector's displacement inside the workspace, finding the shortest trajectory between two points of the workspace is not a simple task. For micromanipulation purpose, that task is even more difficult when facing the presence of multiple micro-objects in the manipulation scene, which can be seen as obstacles.

This paper is dedicated to the study of trajectory planning optimization for that particular microrobot. We first present the kinematic structure of this robot, and its forward and inverse kinematics modeling. In order to navigate inside the discrete generated workspace, the control strategy based on optimal path finding is presented in section 3, while section 4 shows the first trajectory planning using combinatorial optimization based on graph theory analysis. The use of Dijkstra's algorithm from [20] and A* algorithm [21] are detailed in this section, explaining their use and showing some results. An improved algorithm is then presented in section 5 based on a new graphical representation allowing faster computation, the hypercube graph representation. Finally, extensions to more complex robots is discussed in section 6.

2 The DiMiBot

2.1 Robot Kinematics

The DiMiBot's kinematics was designed to fit several specifications in order to generate a desired workspace. Fig. 1 is the kinematic representation of the designed robotic structure. It is composed of $2N$ bistable modules, N on the left side (named bl_i, $0 \leq i \leq N-1$, from bottom to top) and N on the right side (named br_j, $0 \leq j \leq N-1$), all fixed to the same basis and generating the same displacement Δ between their two stable states. The displacement of each bistable module is transmitted to the end-effector thanks to an articulated structure. For fabrication purpose and performance improvements, the revolute joints are designed as monolithic flexure joints, in the real structure. This design leads to the generation of a discrete workspace containing 2^{2N} distinct reachable positions (for a robot with $2N$ bistable modules), corresponding to all the possible configurations of the binary actuators' states.

The motion of this robot's end-effector ($\begin{bmatrix}\delta_x & \delta_y\end{bmatrix}^T$ in the $\mathcal{R}_0$ referential of Fig. 1) is driven by Equation (1) (obtained with the forward kinematic model detailed in [19]).

$$\begin{bmatrix}\delta_x \\ \delta_y\end{bmatrix} = K \cdot \Delta \cdot \begin{bmatrix}\mathbf{A}(N) & -\mathbf{A}(N) \\ \mathbf{A}(N) & \mathbf{A}(N)\end{bmatrix} \cdot \begin{bmatrix} bl_{N-1} \\ \vdots \\ bl_0 \\ br_{N-1} \\ \vdots \\ br_0 \end{bmatrix} \quad (1)$$

where K is a constant depending on the geometric dimensions of the robotic architecture ($= 0.138$ in this case), and $\mathbf{A}(n) = \begin{bmatrix}1 & \frac{1}{2} & \dots & \left(\frac{1}{2}\right)^{n-1}\end{bmatrix}$ is a matrix of size $1 \times n$. The bl_i and the br_j are booleans representing the state (0 or 1) of the corresponding module.

Fig. 2 shows the picture of the first microfabricated digital microrobot whose design, fabrication and characterization are detailed in [19]. It is composed of 4 bistable modules ($N = 2$), each generating a displacement of $\Delta = 35$ μm between their two states, resulting in the generation of a workspace in which the 16 reachable positions are evenly spread in a square of 10.5 μm length, with a resolution of 3.5 μm and a measured repeatability better than 90 nm, in open loop (without any feedback control). The dimensions of this DiMiBot are 36 $mm \times 24$ $mm \times 400$ μm.

2.2 Non-redundant Workspace

With respect to the specifications for this microrobot, the displacements of the end-effector generate a square workspace in which all the 2^{2N} discrete reachable positions are homogeneously distributed. Fig. 3 represents the gener-

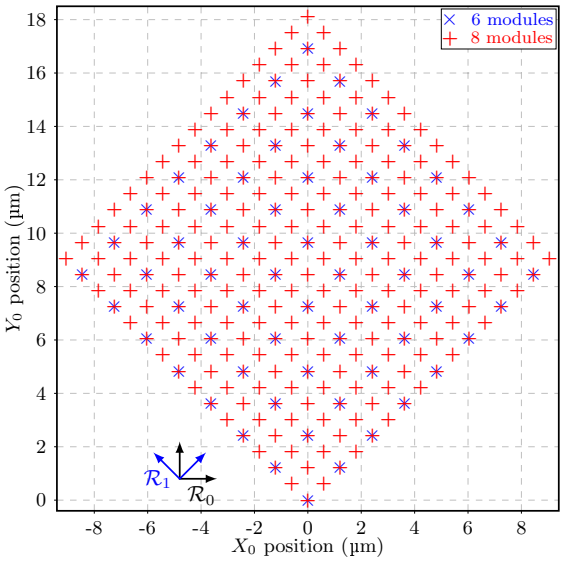


Fig. 3: Workspace of two different digital microrobots, one containing 6 bistable modules ($N = 3$), and one containing 8 ($N = 4$).

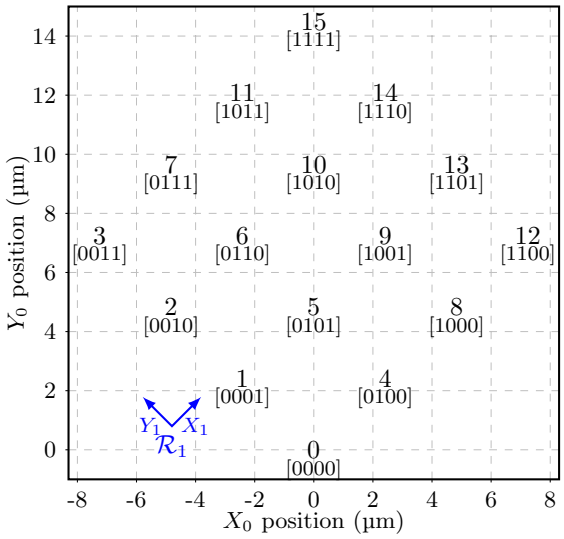


Fig. 4: Workspace numbering for $N = 2$, and the corresponding state of the bistable modules $[bl_1 bl_0 br_1 br_0]$.

ated non-redundant Cartesian workspaces for two different DiMiBots, one containing 6 bistable modules ($N = 3$) and one containing 8 bistable modules ($N = 4$). Each bistable module generates a displacement $\Delta = 35\,\mu m$ between its two stable positions.

In the example of a DiMiBot containing 6 bistable modules, the 64 reachable positions are spread in a square of $11.95\,\mu m$ length, with a resolution of $1.71\,\mu m$. From this figure, the influence of the number of bistable modules used in the structure can be seen. By adding two more bistable modules at the bottom of the robot, it induces a resolution improvement of the reachable workspace, which becomes twice as good.

Equation (2) gives the resolution of this workspace on any axis of the $\mathcal{R}_1$ referential (see Fig. 3). It depends on the geometric characteristics of the flexible structure, on the displacement generated by the bistable modules (Δ) and on the number of modules used on each side (N).

$$r = K \cdot \Delta \cdot \frac{\sqrt{2}}{2^{N-1}} \tag{2}$$

It is then theoretically possible to constantly improve the resolution of a digital microrobot by adding bistable modules to the structure. A nanometric resolution could theoretically be obtained with a robot containing $N = 13$ bistable modules on each side of the structure.

2.3 Inverse Kinematics Model

Each of the 2^{2N} reachable positions of the workspace are addressed by a binary word representing the state (0 or 1) of each bistable module. The binary word is expressed as $\boxed{bl_{N-1}|\cdots|bl_1|bl_0|br_{N-1}|\cdots|br_1|br_0}$. To each reachable position corresponds a number which is the decimal representation of the binary word. This numbering and the corresponding binary word are represented in Fig. 4 for a DiMiBot containing 4 bistable modules ($N = 2$).

By understanding the workspace distribution, we managed to establish an appropriate inverse kinematic model. It enables the calculation of the state of every bistable module (bl_i and br_j) in order to reach a desired position ($\left[{}^1x_d \;\; {}^1y_d\right]^T$ in $\mathcal{R}_1$ referential) as expressed by the boolean equations in (3).

$$\begin{cases} bl_i = \left(\left(\left\lfloor \dfrac{{}^1x_d}{r} \right\rceil \& 2^i \right) \neg = 0 \right) \\ br_j = \left(\left(\left\lfloor \dfrac{{}^1y_d}{r} \right\rceil \& 2^j \right) \neg = 0 \right) \end{cases} \tag{3}$$

$\lfloor \rceil$ is the closest integer function;
${}^1x_d$ and ${}^1y_d$ are the coordinates of the desired position (in $\mathcal{R}_1$ referential);
r is the workspace resolution, as calculated in (2);
& is the bitwise AND function;
$\neg =$ is the boolean difference test;
bl_i is the state of module bl_i ($0 \leq i \leq N-1$);
br_j is the state of module br_j ($0 \leq j \leq N-1$).

3 Path Planning for the DiMiBot

3.1 Context

In that study the supposed application of the DiMiBot is to push micro-objects in the workspace. For that purpose a very thin tip, as represented in the picture of Fig. 5, is assumed to be going from the robot's end-effector down to

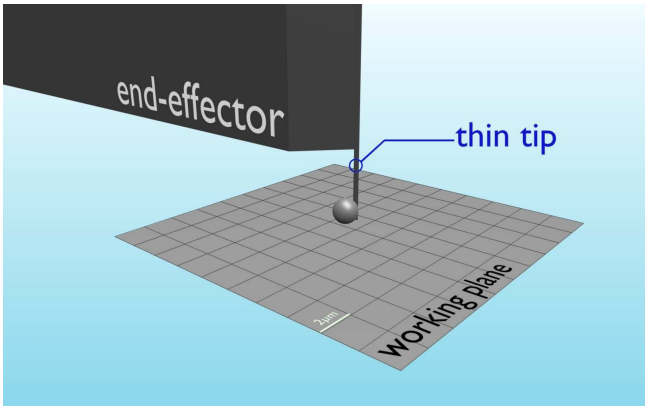


Fig. 5: Illustration of the thin tip going from the robot's end-effector down to the working plane.

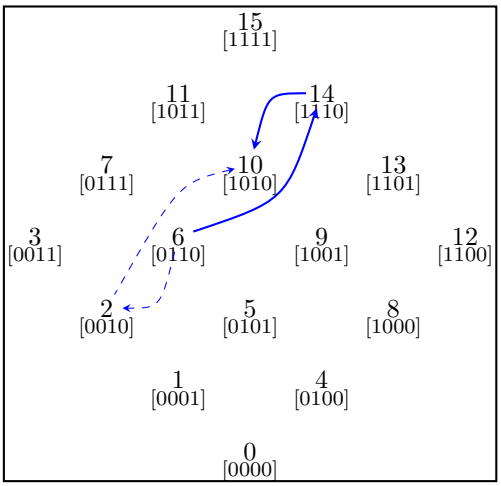


Fig. 6: The two shortest paths when going from point 6 to point 10 (in the case $N = 2$).

the working plane (reaching the objects to be manipulated). Thus the robotic structure would stand over the manipulation scene and the micro-objects would be pushed in any direction by that tip. With this hypothesis the robot does not represent an obstacle to its own displacements for manipulation. In the context of microscale object manipulation, the presence of obstacles is however often encountered. Micromanipulation scene is generally quite small compared to the size of the manipulation tool, and is often populated with multiple micro-objects. The goal of the microrobotic structure is either to organize them, or to assemble them.

The needed actuation for generating a displacement of the end-effector between two points can be calculated thanks to the inverse kinematic model. This calculation firstly involves the boolean function bitwise XOR applied between the binary word from starting point and binary word from ending point. The result of that calculation is a binary word in which all set bits (value 1) describe a module to switch. For instance for switching from point 6 ([0110]) to point 10 ([1010]) in Fig. 4, the XOR function returns [1100], thus we need to switch both bl_1 and bl_0 modules. By switching all these calculated modules simultaneously, the end-effector instantly moves to the desired ending point. However, this method does not allow the control of the end-effector's trajectory because of some dynamic properties of this particular microrobot.

3.2 Dynamic Properties

The DiMiBot uses a particular kind of binary actuators, the bistable modules. As expressed by their name they make use of a mechanically bistable structure which can be switched between its two stable positions (0 and 1). Electrothermal actuators are used to push back and forth the bistable structure. Thanks to the bistable nature, no energy is needed to maintain the module in any of its two stable positions; energy consumption is only needed when switching state. The actuation of the electrothermal actuators is done for approximatively $15\,ms$ with a $20\,V$ and $100\,mA$ input ($30\,mJ$). High speed camera observations showed that the actual switching action (displacement of the shuttle from one state to the other one) only occurs for a fraction of a millisecond. Knowing the exact moment when the switching will happen within the $15\,ms$ actuation interval of time is impossible. Consequently when actuating several bistable modules at the same time, the switching moment will differ for every bistable module. The order in which the modules are actually switched cannot be known, resulting in unpredictable end-effector motion. Because of that particular behavior, precise control of the end-effector's trajectory can only be obtained when switching only one bistable module at a time. This motion strategy however introduces new difficulties.

3.3 Motion Pattern

In the previous example, the robot had to move from point 6 to point 10 (in the case $N = 2$). It requires two steps, corresponding to the switching of the bistable modules bl_0 and bl_1. For instance by first switching bl_1, the robot moves to point 14, and finally moves to point 6 by switching bl_0 (see thick arrows in Fig. 6). It seems like the robot goes too far before coming back to the desired position, but this is actually one of the shortest paths achievable by this robot. The other shortest path consists in choosing to switch module bl_0 first, resulting in a similar motion where the intermediate reached position is point 2 (dashed arrows in Fig. 6).

This motion pattern is not intuitive for the person operating this robot, and can be source of problems.

Indeed imagine the scenario where an obstacle is present at point 14. The operator will not consider this obstacle for a motion between points 6 and 10, which can result in collision. Evaluating this kind of problem requires the use of algorithms for bigger robots.

3.4 Path Finding

Microscale manipulation requires sub-micrometric precision and accuracy. Accuracy is defined by the mechanical repeatability of the bistable modules, while the precision is defined by the reachable workspace resolution. For micro-

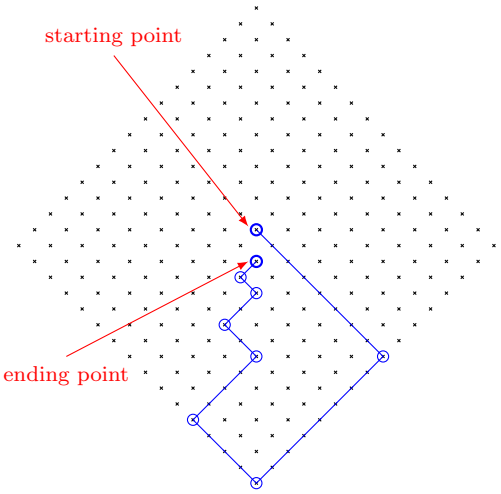


Fig. 7: Path for reaching two points inside a workspace containing 256 reachable positions (DiMiBot for $N = 4$)

manipulation tasks, $200\,nm$ (or better) precision is needed, which is translated in our case by the use of robots containing $N \geq 6$ bistable modules. Because of these high precision requirements, the path finding algorithm should reach the exact end-position desired.

Furthermore, the particular workspace prevents the use of goal region search (defining the goal as a region instead of a single point). The main reason behind this assessment is the difference between configuration space (space of articulated coordinates) and the working space. Two close points in the working space (in term of Euclidean distance) can be far apart in the configuration space (require a long sequence of actuation). This difference is explained in the example of Fig. 7. It shows one of the paths the end-effector has to run through in order to reach two close points (points that could be part of a same goal region) inside the workspace of a robot for $N = 4$.

The motion between these two points is not intuitive. It requires eight steps detailed in TABLE 1. We notice that the bistable modules generate displacements of different length. Module br_3 (first step) generates a displacement of $8 \cdot r$, while module bl_0 (last step) generates a displacement of $1 \cdot r$. The eight bistable modules can be switched in any chosen order, resulting in a path of same length. The length is $30 \cdot r$ (there is no shorter path), while the Euclidean distance between the two points is $\sqrt{2} \cdot r$.

With this example, we can easily understand that motion to different points of the same region requires totally different actuation. This results in an impossibility to specify such region for path planning.

Finding an optimal path between two points of a workspace is the challenge we describe in this paper. Considering the discrete nature of the generated workspace, we propose to use combinatorial optimization based on graph theory.

Table 1: Path corresponding to Fig. 7

point	binary word	traveled distance
starting point	10001000	0
	10000000	$8 \cdot r$
	00000000	$16 \cdot r$
	00000100	$20 \cdot r$
	01000100	$24 \cdot r$
	01000110	$26 \cdot r$
	01100110	$28 \cdot r$
	01100111	$29 \cdot r$
ending point	01110111	$30 \cdot r$

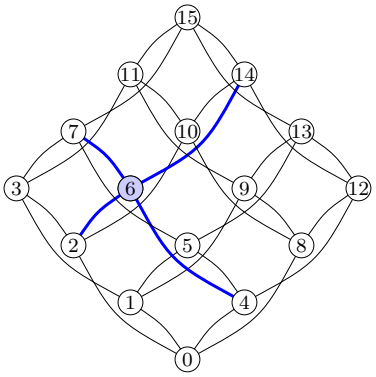


Fig. 8: Graph representation of the workspace of a 4 bistable modules DiMiBot ($N = 2$). Possible displacements from node 6 are highlighted.

4 Combinatorial Optimization

4.1 Graph Theory

Graph theory allows the representation of all the possible displacements between all the reachable positions of the workspace. This representation is usually used to solve path finding problems. A graph is composed of two lists:

vertex list (list of all reachable positions);
edge list (list of all the possible displacements between two adjacent vertices. Vertices are defined as adjacent if their binary word only differs by 1 bit).

The graph associated with the workspace of a DiMiBot with 4 bistable modules ($N = 2$) is presented in Fig. 8. It represents all 16 reachable positions (the same ones as in Fig. 4) by vertices which are connected together with respect to the *one module at a time* rule. The edges of the graph represent the possible displacements inside the workspace when switching each bistable module.

Each of the 16 vertices are connected to 4 other vertices, corresponding to the switching action of one of the 4 bistable modules constituting the DiMiBot. For instance in the case of vertex 6, which corresponds to the binary word [0110],

Table 2: Weighted adjacency matrix $\mathrm{Adj}_{2,2}$ corresponding to the graph of Fig. 8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	2	0	1	0	0	0	2	0	0	0	0	0	0	0
1	1	0	0	2	0	1	0	0	0	2	0	0	0	0	0	0
2	2	0	0	1	0	0	1	0	0	0	2	0	0	0	0	0
3	0	2	1	0	0	0	0	1	0	0	0	2	0	0	0	0
4	1	0	0	0	0	1	2	0	0	0	0	0	2	0	0	0
5	0	1	0	0	1	0	0	2	0	0	0	0	0	2	0	0
6	0	0	1	0	2	0	0	1	0	0	0	0	0	0	2	0
7	0	0	0	1	0	2	1	0	0	0	0	0	0	0	0	2
8	2	0	0	0	0	0	0	0	0	1	2	0	1	0	0	0
9	0	2	0	0	0	0	0	0	1	0	0	2	0	1	0	0
10	0	0	2	0	0	0	0	0	2	0	0	1	0	0	1	0
11	0	0	0	2	0	0	0	0	0	2	1	0	0	0	0	1
12	0	0	0	0	2	0	0	0	1	0	0	0	0	1	2	0
13	0	0	0	0	0	2	0	0	0	1	0	0	1	0	0	2
14	0	0	0	0	0	0	2	0	0	0	1	0	2	0	0	1
15	0	0	0	0	0	0	0	2	0	0	0	1	0	2	1	0

the switching of the bistable modules bl_1, bl_0, br_1 and br_0 respectively generate a displacement to points 14 ([**1**110]), 2 ([0**0**10]), 4 ([01**0**0]) and 7 ([0111]). These 4 links are highlighted in Fig. 8.

In order to find the shortest path between two points, a weight function has to be defined. The minimization of the path's weight will determine the shortest path between two points. Two choices can be made with the DiMiBot for an appropriate weight function:

counting the number of switching (module actuation) needed;
measuring the distance the end-effector runs through.

Those two cases involve the same kind of algorithm implementation. However in the first case, the weight associated to one edge is either 0 or 1, while for the distance this weight becomes more difficult to calculate. In this study we focus on the distance weight function, as the use of the switching weight function can be easily adapted from it. As seen previously the distance traveled by the end-effector depends on the bistable module involved. The bistable module bl_i generates a $2^{i-1} \cdot r$ displacement of the end-effector along X_1 direction. The bistable module br_j generates a $2^{j-1} \cdot r$ displacement of the end-effector along Y_1 direction.

The weighted adjacency matrix is used as the mathematical representation of this graph. It represents the adjacency and weight between all vertices of the graph. It is a square symmetric matrix, in the case of non-oriented graphs, of size 2^{2N} for a DiMiBot with $2N$ bistable modules. The example of the adjacency matrix for a DiMiBot containing 4 ($N = 2$) bistable modules is represented in TABLE 2, in which we can identify the adjacency vertices of vertex 6, and their respective weight value (2 or 1 times the resolution r).

This adjacency matrix is calculated recursively (first over parameter i and then over parameter j) with the system of Equations (4).

$$
\begin{aligned}
\mathrm{Adj}_{1,0} &= \begin{bmatrix} 0 & 2^0 \\ 2^0 & 0 \end{bmatrix} \\
\mathrm{Adj}_{i,0} &= \begin{bmatrix} \mathrm{Adj}_{i-1,0} & 2^{i-1} I(2^{i-1}) \\ 2^{i-1} I(2^{i-1}) & \mathrm{Adj}_{i-1,0} \end{bmatrix} \qquad 1 < i < N \\
\mathrm{Adj}_{N,j} &= \begin{bmatrix} \mathrm{Adj}_{N,j-1} & 2^{j-1} I(2^{N+j-1}) \\ 2^{j-1} I(2^{N+j-1}) & \mathrm{Adj}_{N,j-1} \end{bmatrix} \quad 1 < j < N
\end{aligned}
\tag{4}
$$

where $I(k)$ is the identity matrix of size k. In the case the other weight function is used (counting the number of switching needed) a similar adjacency matrix is obtained, in which only 0 and 1 can be found. The Equation (4) is then modified, and all 2^k factors in front of identity matrices are replaced by ones.

4.2 Path Planning With Obstacles

We first use Dijkstra's algorithm from [20] for point to point shortest path search, which ensures to find one of the shortest paths. Using such an algorithm is mainly useful in the case of obstacle presence inside the workspace. The obstacles are taken into account by modifying the adjacency matrix. These modifications will be explained with an example from the graph of Fig. 8. Assuming an obstacle is present at vertex 6, all four displacements toward vertex 6 have to be deleted, this is done by resetting all value of column 6 and row 6 in the adjacency matrix. But other displacements going through vertex 6 have to be deleted. This is the case of the displacements between vertices 5 and 7, and between vertices 2 and 10. The four elements of the adjacency matrix corresponding to these two displacements have to be reset to 0. Finding those elements of the adjacency matrix to be erased can be troublesome for huge robots (containing 10 or more bistable modules) and is then done with an appropriate script detailed in **Algorithm 1**.

Fig. 9 represents the evolution of the Dijkstra's algorithm calculation between two points when facing different obstacle configurations (up to 5 obstacles), for a DiMiBot containing 10 bistable modules ($N = 5$, generating a workspace with 1024 reachable positions). This shows how slight changes in the obstacles configuration results in completely different paths. The first five cases generate a trajectory with the same weight (same distance traveled by the end-effector $= 44 \cdot r$), while for the last case the shortest path generates a traveled distance of $46 \cdot r$ (no shorter path can be found).

This algorithm is however time consuming. This can be improved using Dijkstra's bidirectional algorithm which is almost twice as fast.

4.3 A* Algorithm

Another well known alternative is the A* algorithm from [21]. It is particularly well suited for geographic exploration in which it can be seen as a function that directs

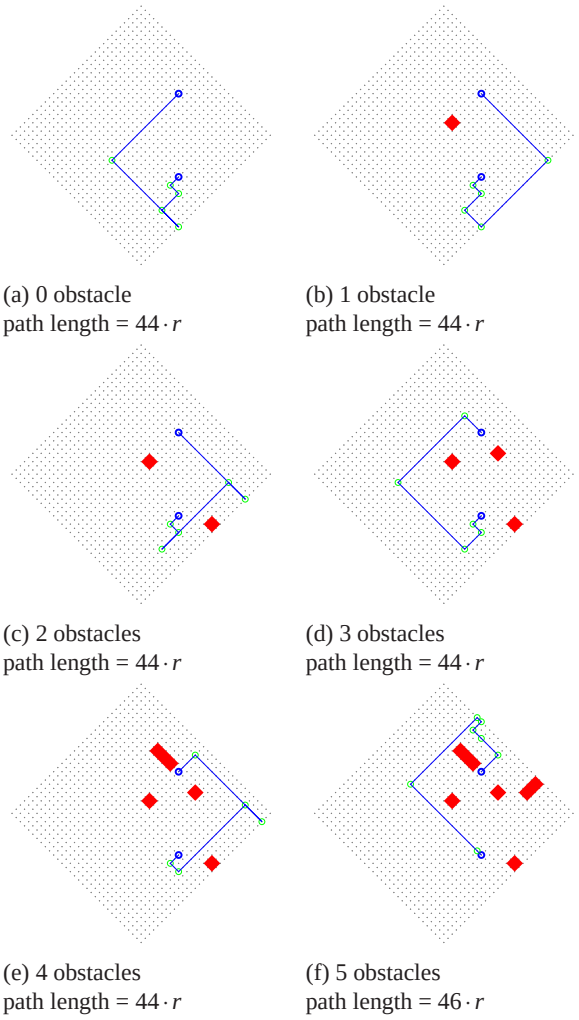
Algorithm 1 Script to adapt the adjacency matrix with the presence of obstacles.

```
for all obstacles Obs do
    O_x ← line of Obs on X_1 direction
    O_y ← line of Obs on Y_1 direction
    if MostSignificantBit(O_x) = 1 then
        domain = [O_x, 2^{N_L} − 1]
    else
        domain = [0, O_x]
    end if
    for all point P in domain do
        for all bit B of P do
            P2 ← switchBit(B, P)
            if O_x is between P and P2 then
                Ob_1 ← binaryConcatenate(P, O_y)
                Ob_2 ← binaryConcatenate(P2, O_y)
                MatAdj[Ob_1, Ob_2] ← 0
                MatAdj[Ob_2, Ob_1] ← 0
            end if
        end for
    end for
    Do the same for O_y
end for
```

the search in one direction. It is based on the calculation of a cost function that estimates the remaining path to travel.

In the case of the workspace generated by the digital microrobot the geographic analogy cannot be used because of the particular distribution of points in the workspace. As it was explained in §3.4, Euclidean distance cannot be used to describe proximity of points for finding the shortest path.

Unlike Dijkstra's algorithm, the calculated path is only assured to be optimum under a certain condition on the chosen estimation cost function. The condition is that the cost function does not overestimate the remaining distance to the goal. In our case the use of the $\mathcal{R}_1$ distance satisfies that requirement and ensures to find an optimal path.



(a) 0 obstacle
path length = $44 \cdot r$

(b) 1 obstacle
path length = $44 \cdot r$

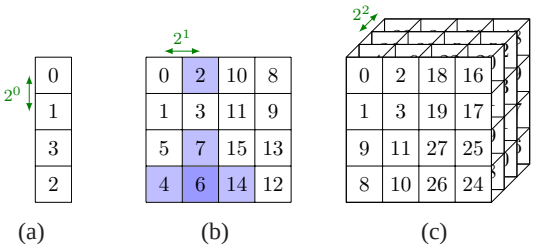
(c) 2 obstacles
path length = $44 \cdot r$

(d) 3 obstacles
path length = $44 \cdot r$

(e) 4 obstacles
path length = $44 \cdot r$

(f) 5 obstacles
path length = $46 \cdot r$

Fig. 9: Shortest path found with the Dijkstra's algorithm in the case of a DiMiBot with 10 bistable modules ($N = 5$) and with the presence of 0 to 5 obstacles.



(a) (b) (c)

Fig. 10: Cubic graph of DiMiBots containing 2, 4 and 6 bistable modules, respectively hypercubes of dimension 1, 2 and 3. In the hypercube of dimension 2, the node 6 and its neighbours are highlighted.

5 Improving the Algorithm

5.1 Hypercube Graph

In order to reduce even more calculation time, a new graphical representation is used. It is a hypercube graph representation, adapted to the DiMiBot's generated workspace. It allows a geographical visualization of the adjacent vertices. That visualization for workspaces generated by DiMiBots containing 2, 4 and 6 bistable modules (N respectively equal to 1,2 and 3) is shown in Fig. 10.

That graph takes the form of a hypercube (of dimension N) in which the 2^{2N} reachable positions are all represented. The particularity of this representation is that all adjacent vertices are also geometric neighbors, in a N-dimensional space (with periodic boundary conditions). The geometric analogy can then be applied with this representation. In the example of the DiMiBot with 4 bistable modules (represented in Fig. 10b), it takes the form of a square containing the 16 reachable positions in which are highlighted the vertices adjacent to vertex 6, which are its four geographic neighbors in this representation. We immediately see that for going from vertex 6 to vertex 10, choosing to go through point 2 or point 14 is the same, and those are the only two shortest paths.

This hypercube graph can also be calculated for any

DiMiBot. Equation (5) for this calculation uses the MATLAB function for matrix concatenation on dimension i.

$$\begin{aligned} G_0 &= 0 \\ G_i &= concat(i, G_{i-1}, G_{i-1} + 2^{i-1}, \\ &\qquad G_{i-1} + 2^{i-1} + 2^{N+i-1}, G_{i-1} + 2^{N+i-1}) \end{aligned} \tag{5}$$

where G_i is the MATLAB representation of the hypercube graph (i.e. a matrix of dimension i).

5.2 Use of the Hypercube

To use this new tool for the calculation of the shortest path, several new functions need to be defined:

find the position of a vertex inside the cubic graph;
find all the neighbors of this vertex;
calculate the weight corresponding to each neighbor pair;
calculate the remaining path to go from this vertex to the ending vertex (estimation function).

The first point, finding the position of a desired vertex in the hypercube graph, can be done with **Algorithm 2**:

Algorithm 2 invCubic : Algorithm to find the position of one vertex in the hypercube graph.

```
for all stage i of the DiMiBot do
    if bl_i br_i=00 then
        res← 1
    else if bl_i br_i=01 then
        res← 2
    else if bl_i br_i=11 then
        res← 3
    else if bl_i br_i=10 then
        res← 4
    end if
end for
pos[i] ← res
```

This algorithm gives the row in which the desired vertex can be found in each dimension of the graph. Then the desired vertex n_d can be found at position $G_N(pos[1], \ldots, pos[N])$.

Finding the neighbors is intuitive with this representation and is less calculation consuming than the use of the adjacency matrix (which has an $O(2^{2N})$ complexity calculation). Calculating the weight of a displacement is also easily deduced from the hypercube representation. The weight corresponds to the dimension in which the displacement is done. A displacement along the i^{th} dimension corresponds to a weight of 2^{i-1} (see Fig. 8). This weight is shown for the three first dimensions in Fig. 10. With that in mind the remaining path can also be estimated easily with **Algorithm 3**.

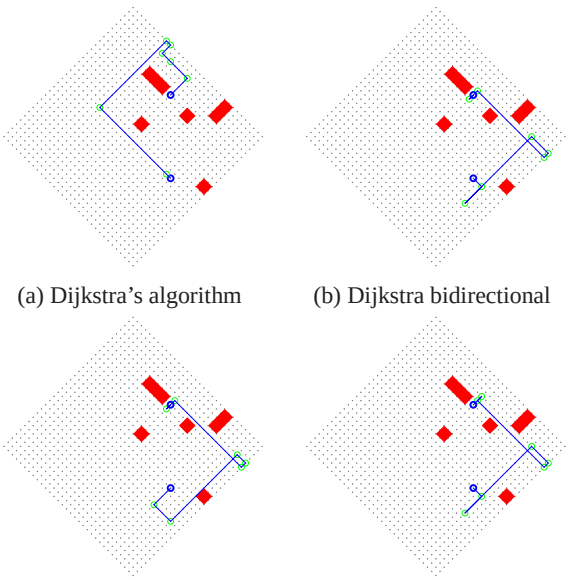
Algorithm 3 Remaining path estimation

```
starting vertex : n_s
ending vertex : n_e
posS ← invCubic(n_s)
posE ← invCubic(n_e)
dist ← 0
for all dimension i of the hypercube do
    dist ← dist + |mod(posS[i]-posE[i],2)| × 2^(i-1)
end for
remain_path ← dist
```

5.3 Comparison Between Algorithms

The newly designed algorithm has now to be compared to the other ones. The four compared algorithms are:

Dijkstra's algorithm;
Dijkstra's bidirectional algorithm;
A* algorithm in association with the adjacency matrix;
A* algorithm in association with the hypercube graph representation.



(a) Dijkstra's algorithm (b) Dijkstra bidirectional

(c) A* - adjacency matrix (d) A* - cubic graph

Fig. 11: Shortest path found by different algorithms

A first element of comparison is presented in Fig. 11. It shows the shortest path found for the four algorithms in the case of a robot with 10 bistable modules ($N = 5$) and in the presence of 5 obstacles in the workspace. All the four algorithms find a different path, but it is always one of the shortest paths (distance $= 46 \cdot r$). The main difference between these algorithms is the time spent[1] to find a path. To compare the computation time of the algorithms, numerous simulations were done in the presence of a random number of obstacles

[1] simulations done on MATLAB under Linux with an Intel Core 2 Duo processor at 3.00 GHz

and the computing time was recorded. The results of these simulations are reported in Fig. 12 for robots with different number of bistable modules used (from 4 to 20). This figure shows the average computation time over 300 simulations[2], for different algorithms and different robot size.

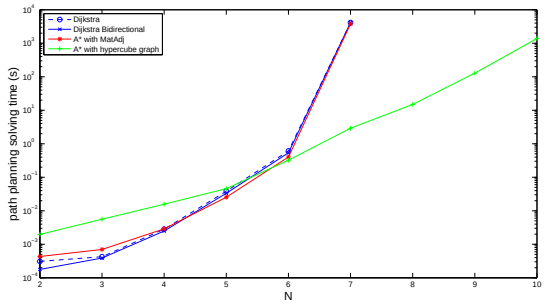


Fig. 12: Comparison in computation time for the different algorithms in the cases for robots up to 20 bistable modules ($N = 10$).

A first thing worth noticing is the inability for algorithms using the adjacency matrix to compute the shortest path for a robot containing 16 bistable modules ($N = 8$) or more. The reason lies in the MATLAB memory limitations which prevent the computation of $2^{16} \times 2^{16}$ size matrices. Very few differences can be noticed for the use of the three first algorithms (using the adjacency matrix). The use of the hypercube graph representation shows a large difference in computation time, and is shown to be much more efficient for big DiMiBots ($N > 5$). However for smaller robots, using the adjacency matrix seems a better solution.

6 Extension to Non-Symmetric Robots

Non-symmetric robotic structures can also be considered. A different number of modules on the right side of the structure and on the left side is used. The N_L bistable modules on the left side generate a displacement of the end-effector in the X_1 direction, while the N_R modules on the right side generate its displacement in the Y_1 direction. The workspace contains $2^{N_L+N_R}$ reachable positions, and is no longer homogeneous. In this case two different resolution values have to be calculated, the resolution in X_1 direction (r_x) and that in the Y_1 direction (r_y). They are calculated with the following equations:

$$r_x = K \cdot \Delta \cdot \frac{\sqrt{2}}{2^{N_L-1}} \quad ; \quad r_y = K \cdot \Delta \cdot \frac{\sqrt{2}}{2^{N_R-1}} \tag{6}$$

The forward and inverse kinematic models are also modified accordingly, resulting in the following equations.

$$\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = K \cdot \Delta \cdot \begin{bmatrix} \mathbf{A}(N_L) & -\mathbf{A}(N_R) \\ \mathbf{A}(N_L) & \mathbf{A}(N_R) \end{bmatrix} \cdot \begin{bmatrix} bl_{N_L-1} \\ \vdots \\ bl_0 \\ br_{N_R-1} \\ \vdots \\ br_0 \end{bmatrix} \tag{7}$$

$$\begin{cases} bl_i = \left(\left(\left\lfloor \dfrac{^1x_d}{r_x} \right\rfloor \& 2^i \right) \neg = 0 \right) \\ br_j = \left(\left(\left\lfloor \dfrac{^1y_d}{r_y} \right\rfloor \& 2^j \right) \neg = 0 \right) \end{cases} \tag{8}$$

The adjacency matrix will also be easily calculated with a small modification to the recursive calculation.

$$\begin{aligned} \mathrm{Adj}_{1,0} &= \begin{bmatrix} 0 & 2^0 \\ 2^0 & 0 \end{bmatrix} \\ \mathrm{Adj}_{i,0} &= \begin{bmatrix} \mathrm{Adj}_{i-1,0} & 2^{i-1} I(2^{i-1}) \\ 2^{i-1} I(2^{i-1}) & \mathrm{Adj}_{i-1,0} \end{bmatrix} \qquad 1 < i < N_L \\ \mathrm{Adj}_{N_L,j} &= \begin{bmatrix} \mathrm{Adj}_{N_L,j-1} & 2^{j-1} I(2^{N_L+j-1}) \\ 2^{j-1} I(2^{N_L+j-1}) & \mathrm{Adj}_{N_L,j-1} \end{bmatrix} \quad 1 < j < N_R \end{aligned} \tag{9}$$

Finally, the hypercube graph representation will become a hyper-parallelepiped. Assuming $N_L < N_R$, this hyper-parallelepiped contains 2 elements on the first $N_R - N_L$ dimensions and 4 elements on the remaining N_L dimensions. Each dimension of the hyper-parallelepiped graph corresponds to one stage of the DiMiBot. The last dimension corresponds to the displacements generated by the bistable modules bl_{N_L-1} and br_{N_R-1} (4 different configurations), while the first dimension corresponds to the displacements only generated by the bistable module br_0 (2 different configurations) as there is no symmetric module on the left side of the structure (if $N_L < N_R$). A similar calculation as (5) could be used for non-symmetric robots.

7 Conclusion

This paper presents the control strategy used for trajectory planning of the DiMiBot. This Digital MicroroBot is a new kind of robot which only needs open-loop control for accurate and precise micropositioning tasks. The control of this kind of robot is focused on the trajectory planning based on combinatorial optimization. Algorithms such as Dijkstra's algorithm or A* algorithm are well adapted to trajectory planning for the workspace of the DiMiBot. A new graphical representation, particularly well adapted to the workspace generated by the DiMiBot is developed, and presents very

[2]because of large computation time, only 20 simulations were done for a 14 module robot ($N = 7$)

good results in trajectory planning for DiMiBots of any size. The designed algorithm could perform shortest path search for DiMiBots with 16 bistable modules or more in a fairly low amount of time. The use of the hypercube graph representation developed here facilitates the search of the adjacent vertices and the calculation of the remaining path. This trajectory planning algorithm is well suited for the DiMiBot in the presence of obstacles.

Acknowledgment

The authors would like to thank the Direction Générale de l'Armement (DGA) for their financial support. This work has been supported by the Labex ACTION project (contract "ANR-11-LABX-0001-01"). This work has been supported by the Equipex ROBOTEX project (contract "ANR-10-EQPX-44-01"). This work has been supported by the French RENATECH network and its FEMTO-ST technological facility.

References

[1] Yamahata, C., Collard, D., Legrand, B., Takekawa, T., Kumemura, M., Hashiguchi, G., and Fujita, H., 2008. "Silicon nanotweezers with subnanometer resolution for the micromanipulation of biomolecules". *Microelectromechanical Systems, Journal of*, **17**(3), pp. 623–631.

[2] Das, A., J., S., Popa, D., and Stephanou, H., 2008. "On the precision alignment and hybrid assembly aspects in manufacturing of a microspectrometer". In IEEE Conference on Automation Science and Engineering, pp. 959–966.

[3] Rhee, M., and Burns, M. A., 2008. "Microfluidic assembly blocks". *Lab on a Chip*, **8**(8), pp. 1365–1373.

[4] Chronis, N., and Lee, N., 2005. "Electrothermally activated su-8 microgripper for single cell manipulation in solution". In Journal of Microelectromechanical Systems, pp. 857–863.

[5] Schmoeckel, F., and Wörn, H., 2001. "Remotely controllable mobile microrobots acting as nano positioners and intelligent tweezers in scanning electron microscopes (sems)". In IEEE International Conference on Robotics and Automation, pp. 3909–3913.

[6] Beyeler, F., Neild, A., Oberti, S., Bell, D., Sun, Y., Dual, J., and Nelson, B., 2007. "Monolithically fabricated microgripper with integrated force sensor for manipulating microobjects and biological cells aligned in an ultrasonic field". *Journal of Microelectromechanical Systems*, **16**(1), pp. 7–15.

[7] Kim, K., Liu, S., Zhang, Y., and Sun, Y., 2008. "Nanonewton force-controlled manipulation of biological cells using a monolithic mems microgripper with two-axis force feedback". *Journal of Micromechanics Microengineering*, **18**(5).

[8] Yong, Y., Aphale, S., and Moheimani, S., 2009. "Design, identification, and control of a flexure-based xy stage for fast nanoscale positioning". *IEEE Transactions on Nanotechnology*, **8**(1), p. 46.

[9] Koseki, Y., Tanikawa, T., Koyachi, N., and Arai, T., 2000. "Kinematic analysis of translational 3-dof micro parallel mechanism using matrix method". *Advanced Robotics*, **16**, pp. 251–264.

[10] Dong, J., Mukhopadhyay, D., and Ferreira, P. M., 2007. "Design, fabrication and testing of a silicon-on-insulator (soi) mems parallel kinematics xy stage". *Journal of Micromechanics and Microengineering*, **17**(6), pp. 1154–1161.

[11] Tian, Y., Shirinzadeh, B., Zhang, D., Liu, X., and Chetwynd, D., 2009. "Design and forward kinematics of the compliant micro-manipulator with lever mechanisms". *Precision Engineering*, **33**(4), pp. 466 – 475.

[12] Rakotondrabe, M., Clévy, C., and Lutz, P., 2008. "Hysteresis and vibration compensation in a nonlinear unimorph piezocantilever". In Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, IEEE, pp. 558–563.

[13] Chirikjian, G., 1994. "A binary paradigm for robotic manipulators". In IEEE International Conference on Robotics and Automation, pp. 3063–3069.

[14] Ebert-Uphoff, I., and Chirikjian, G., 1995. "Efficient workspace generation for binary manipulators with many actuators". *Journal of Robotics Systems*, **12**, pp. 383–400.

[15] Lees, D., and Chirikjian, G., 1996. "A combinatorial approach to trajectory planning for binary manipulators". In IEEE International Conference on Robotics and Automation, Vol. 3, pp. 2749–2754.

[16] Chen, Q., Haddab, Y., and Lutz, P., 2008. "Digital microrobotics based on bistable modules: Design of compliant bistable structures". In IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, pp. 36–41.

[17] Chen, Q., Haddab, Y., and Lutz, P., 2011. "Microfabricated bistable module for digital microrobotics". *Journal of Micro - Nano Mechatronics*, **6**, pp. 1–12. 10.1007/s12213-010-0025-2.

[18] Chalvet, V., Zarzycki, A., Haddab, Y., and Lutz, P., 2011. "Digital microrobotics based on bistable modules: Design of a non-redundant digital micropositioning robot". In IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 3628–3633.

[19] Chalvet, V., Haddab, Y., and Lutz, P., 2013. "A microfabricated planar digital microrobot for precise positioning based on bistable modules". *IEEE Transactions on Robotics*, **29**(3), June, pp. 641–649.

[20] Dijkstra, E. W., 1959. "A note on two problems in connexion with graphs". *Numerische Mathematik*, **1**, pp. 269–271. 10.1007/BF01386390.

[21] Hart, P., Nilsson, N., and Raphael, B., 1968. "A formal basis for the heuristic determination of minimum cost paths". *Systems Science and Cybernetics, IEEE Transactions on*, **4**(2), july, pp. 100 –107.

The author has requested enhancement of the downloaded file. All in-text references underlined in blue are linked to publications on ResearchGate.