Simple Obstacle Avoidance Algorithm for Rehabilitation Robots

Floran H. A. Stuyt, GertWillem R. B. E. Römer, and Harry J. A. Stuyt

Abstract— **The efficiency of a rehabilitation robot is improved by offering *record-and-replay* to operate the robot. While automatically moving to a stored target (replay) collisions of the robot with obstacles in its work space must be avoided. A simple, though effective, generic and deterministic algorithm for obstacle avoidance was developed. The algorithm derives a collision free path of the end-effector of the robot around known obstacles to the target location in $O(n)$ time. In a case study, using the rehabilitation robot ARM, the performance of the algorithm was tested. As was a newly Human-Machine-Interface offering this record-and-replay functionality to the user.**

I. INTRODUCTION

Since users of rehabilitation robots, such as the ARM (Fig. 1), often repeat certain tasks, usability can be improved by allowing users to record the robot's movements and replay them later on. This functionality is usally referred to as *record-and-replay* functionality or macro-functionality. That is, tasks are (partly) automated by specificying a target for the manipulator to reach (i.e. a location only and not a path), and then let the manipulator calculate which path to follow towards the target. In this approach, an obstacle avoidance algorithm is essential to generate obstacle free paths in the workspace of the robot. Examples of obstacles are the wheelchair the robot is attached to, furniture, tables, tableware, etc. The robot itself can also be considered as an obstable. This paper presents an obstacle avoidance algorithm, which can be used in such situations.

Rehabiliation robots like the ARM, are mobile and operate in an unstructured environment. This implies that the location and dimensions of obstacles which are not fixed to the robot or the wheelchair, may vary in time. Avoiding these obstacles automatically requires sensors (e.g. cameras) to dynamically build a model of the environment. This paper assumes that such a world model is available. However, due to the associated complexity and costs, the obstacle avoidance algorithm presented in this paper was tested on static obstacles only.

Existing solutions for obstacle avoidance are discussed in the next paragraph. Subsequently, an obstacle avoidance algorithm is proposed in Section III. The new method is evaluated in a casestudy in Section IV. Finally, a conclusion is given and future work on the algorithm is presented.

Manuscript received February 9th, 2007. Floran H. A. Stuyt (corresponding author), GertWillem R. B. E. Römer and Harry J. A. Stuyt are with Exact Dynamics BV, Kervel 4, NL-6942 SC, Didam, The Netherlands, phone: +31 316 334114, fax: +31 316 331327 (email: f.h.a.stuijt@exactdynamics.nl).

Fig. 1. The Assistive Robotic Manipulator (ARM)

II. LITERATURE REVIEW

Obstacle avoidance involves calculating a path through an environment, based on constraints within that environment (obstacles) as well as mechanical limits of the robot (joint limits and workspace boundaries). In general, solving this problem is referred to as *path planning*.

Methods have been proposed which are able to (partly) solve the path planning problem (e.g. [2] and [3], which will be discussed below). First, there are two methods to represent the environment; either the 3 dimensional workspace of the robot, or a n dimensional space in which n is the amount of degrees of freedom (DOF) of the kinematic chain of the robot.

The first method, makes use of the representation of the location and dimensions of each obstacle in three dimensional space, referred to as the *work space*. Using this representation, path planning involves finding a collision free path through this three dimensional space. When using this representation, the algorithm becomes considerably complex when taking into acount the robot itself (i.e. the robot should avoid self collisions) as well as parts other than the end

effector (obstacles). It is relatively easy to plan a collision free path for the end effector of the robot, while the other links of the robot are still able to collide with the obstacles in the environment or with each other. The second method to represent the environment is by defining a n dimensional space in which n is the number of DOFs in the kinematic chain, referred to as the *configuration space* [2]. Each point in this space represents a configuration of the kinematic chain. This n dimensional configuration space can be treated as a search space for the path planner. In both representations the space in which the robot can move without collisions, is called the *free space*.

One category of path planning methods which make use of these representations are cell decomposition methods [4]. In general, these methods are based on a decomposition of the free space of the representational space into cells. After decomposition, graph search algorithms can be applied to this (discrete) representational space. Cell decomposition methods differ mainly in the way which the representation space is decomposed. For example, the space could be divided into equal sized cells, or into irregulary shaped cells [6]. The latter example reduces the search space, but increases the computational cost for calculating the search graph.

Another approach for solving path planning problems is the use of so called artificial potential fields [5]. Basically, the workspace in this approach is made up of an artificial energy field in which the end effector is being attracted towards the goal position. Obstacles are represented in this field as repulsive areas. In this approach the (obstacle free) path is calculated by minimizing the amount of energy. High energy means repulsion, and low energy attraction. This method lets the end effector try to find a way around these repulsive areas to get to the goal position. The major disadvantage of potential field methods is the fact that it is possible to get 'stuck' in a local minimum.

In case of a kinematic chain with a high number of DOFs, it is hard to build an explicit representation of the configuration space. This problem can be solved by probing the configuration space using random sampling [3]. Since this probing method requires random sampling, it can introduce unpredictable behaviour of the rehabilitation robot. That is, unpredictable behaviour, from the point of view of the user of the robot. In case of rehabilition robotics, the primary concern for solving the obstacle avoidance problem, is safety and predicatbility. Another concern is the likely limited processing power available to the robot.

Since the above methods require heavy processing load due to random sampling of the configuration space, they are not suitable for solving our problem. This limits our problem to finding a path in the work space of the robot for the end effector only. In this case, the search space can be explicitly represented on which it is possible to build a deterministic algorithm which can solve the path finding problem in linear time $O(n)$, in which the complexity depends on the number of obstacles presented in the work space, which is discussed in the next section.

In this paper, a simple obstacle avoidance algorithm is presented which has te following advantages over the methods described above:

- it is deterministic and therefore results in predictable behavior;
- it does not involve time consuming computations;
- it is easy to implement.

This proposed method is both deterministic and simple to use, hence the title of this article.

III. Method

In this section, an obstacle avoidance algorithm is described which is capable of producing a collision free path for the end-effector of the robot. It assumes a start position of the end-effector, denoted by $\mathbf{s}$, and a target or goal position, dentoted by $\mathbf{g}$. In our method, each obstacle is represented as a rectangular box, which is defined by a position vector ${}^{A}P_{ORG}$ relative to the coordinate system of the robot, and three orientation vectors ${}^{A}P_{X}$, ${}^{A}P_{Y}$, and ${}^{A}P_{Z}$ (see Fig. 2). The algorithm can be be easily adapted to include differently shaped objects, for example cylindrical and spherical obstacles. Furthermore, complex shaped objects can be formed by a series of boxes, cylinderical, spheres etc. However, for simplicity, in this paper multiple rectangluar objects are discussed only.

Basically, the algorithm calculates a path over the surfaces or planes of the obstacle. The algorithm consists of the following five steps for each obstacle, each of which are subsequently discussed in detail:

1) Transformation (translation and rotation) of the object, as well as $\mathbf{s}$ and $\mathbf{g}$, relative to the reference frame of the robot;
2) Calculation of the entry point $\mathbf{p}_{in}$ and exit point $\mathbf{p}_{out}$ of the path through an obstacle;
3) Calculation of projections of $\mathbf{p}_{in}$ and $\mathbf{p}_{out}$ on edges of corresponding planes of the obstacle;
4) Determination of the shortest path over the planes, via the edge points calculated in step 3;
5) Retransformation of the shortest path, determined in step 4, to the original location of the object.

A. Step 1: Start and Goal Transformation

The object and the start ($\mathbf{s}$) and target ($\mathbf{g}$) location are expressed in the robot frame $\{A\}$, in an arbitray orientiation and at an arbitray distance the to the origin of the robot frame, see Fig. 2. However, the algorithm assumes that the obstacle is aligned with the robot's frame. In addition, it assumes that the origin is at the object center. Therefore, each obstacle, as well as the start ($\mathbf{s}$) and traget ($\mathbf{g}$) location are transformed (by homogenous transformations) so that the robot $\{A\}$ and object $\{B\}$ frame are aligned, see Fig. 3.

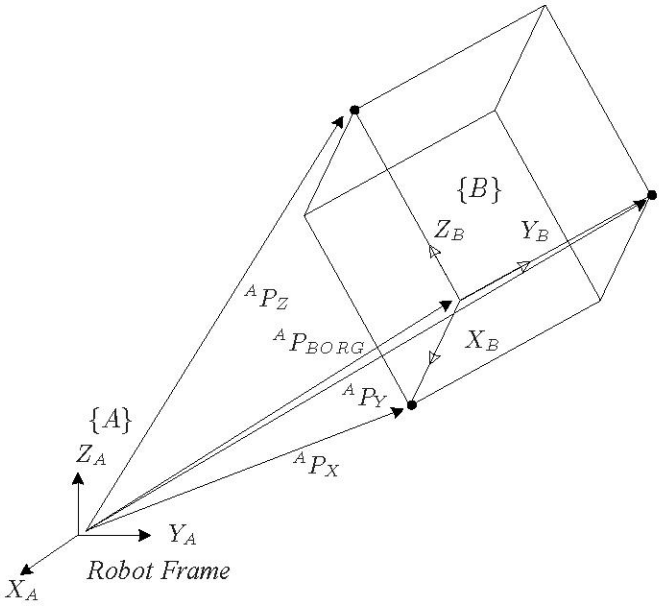

Fig. 2. Step 1: Transformation.

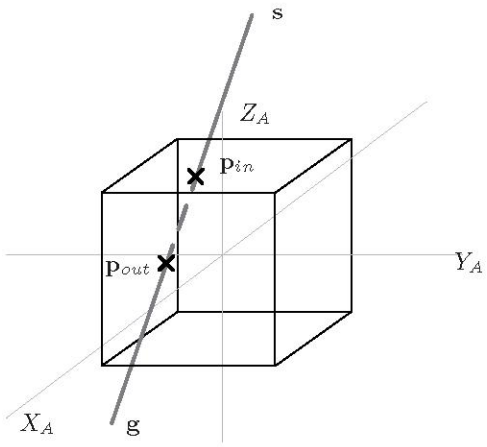

Fig. 3. Step 2: Calculation of $\mathbf{p}_{in}$ and $\mathbf{p}_{out}$.

B. Step 2: Calculation of Entry and Exit Point

This step involves the determination of the intersections of the original path from $\mathbf{s}$ to $\mathbf{g}$ with the planes of the obstacle. These intyersections are referred to as $\mathbf{p}_{in}$ and $\mathbf{p}_{out}$ for respectively the entry point and the exit point of path with the obstacle (See Fig. 3). The path from $\mathbf{s}$ to $\mathbf{g}$ is mathematically described by

$$\mathbf{s} + t(\mathbf{g} - \mathbf{s}) \tag{1}$$

where $0 \leq t \leq 1$. All intersections $p_i = \mathbf{s} + t_i\mathbf{g}$, $i = 1, 2, \ldots, 6$ of the path with planes in which the surfaces of the box lie, are the points on the path with a value of t (1) which equal:

$$t_1 = \frac{r_x - \mathbf{s}_x}{\mathbf{g}_x - \mathbf{s}_x} \tag{2}$$

$$t_2 = \frac{-r_x - \mathbf{s}_x}{\mathbf{g}_x - \mathbf{s}_x} \tag{3}$$

$$t_3 = \frac{r_y - \mathbf{s}_y}{\mathbf{g}_y - \mathbf{s}_y} \tag{4}$$

$$t_4 = \frac{-r_y - \mathbf{s}_y}{\mathbf{g}_y - \mathbf{s}_y} \tag{5}$$

$$t_5 = \frac{r_z - \mathbf{s}_z}{\mathbf{g}_z - \mathbf{s}_z} \tag{6}$$

$$t_6 = \frac{-r_z - \mathbf{s}_z}{\mathbf{g}_z - \mathbf{s}_z} \tag{7}$$

where s_x, s_y, s_z, and g_x, g_y, g_z are the coordinates $\mathbf{s}$ and $\mathbf{g}$, and $2r_j$, $j = x, y, z$ is the length of the edge of the rectangular box is in the j^{th} direction. Only points p_i for which the inequalty

$$-r_j \leq p_i \leq r_j \tag{8}$$

holds, are lying in a surface of the rectangular box. If none of the points p_i meet (8) then the next obstacle can be processed, since the path does not collide with the current obstacle. The path of equation (1) may intersect with an edge or vertice of the box, implying that $\mathbf{p}_{in} = \mathbf{p}_{out}$. For simplicity, the algorithm threats these intersection points as seperate points though.

C. Step 3: Calculation of Points on Edges

For the surface holding $\mathbf{p}_{in}$, adjacent points a_1, a_2, a_3, a_4 on the edges of the plane are calculated by projecting $\mathbf{p}_{in}$ on the edges of the plane (Fig. 4). Similar, the projection of $\mathbf{p}_{out}$ on the edges of its corresponding surface yield b_1, b_2, b_3, b_4.

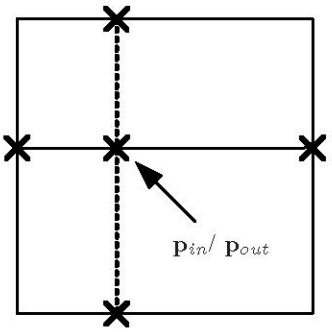

Fig. 4. Step 3: Calculation of points on edges.

D. Step 4: Determination of Shortest Path

Next, all paths from $\mathbf{p}_{in}$, via a_i and b_j to $\mathbf{p}_{out}$ over the surfaces of the box are calculated. The length of each path is calculated. The algorithm verfies whether adjacent points of the candidate path join an surface of the box. If not the candidate path is discarded. The shortest path is chosen, see for example Fig. 5.

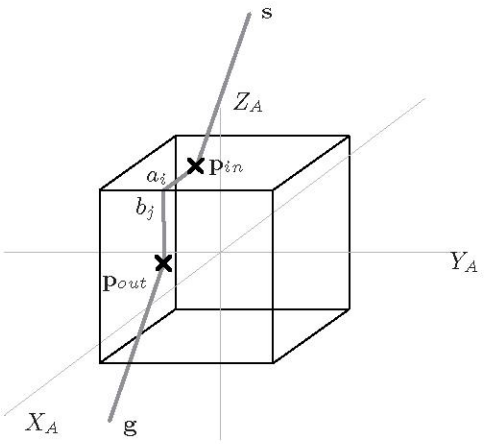

Fig. 5. Resulting path for obstacle

E. Step 5: (Re)Transformation

Finally, the calculated, obstacle free path, is transformed back to the original location of the object (Fig. 2), using the inverse of the transformation of Step 1.

F. Discussion

The presented algorithm allows to calculate a collision free path from the start to target location, for the end effector. However, it does not take the links of the kinematic chain into account. This means that the algorithm cannot take care of planning a collision free path without self collision and without ensuring that the other links in the kinematic chain will not collide with an obstacle or with other links of the robot.

In general, the orientation of the gripper in the start and goal location differ. Hence, in general, for the gripper orientation, no conditions are known during the execution of the calculated path, except that at the goal location the gripper should have the targeted orientation. Therefore, it was chosen to not take into account the orientation of the gripper. The orientation of the end effector remains the same during the execution of the path. However, the algorithm could be extended easily to gradually change the orientation of the end effector to a specified value during the execution of the path.

IV. Case Study

The presented algorithm has been implemented for use and tested on the Assistive Robotic Manipulator (ARM), see Fig. 1. In case of the ARM, users often perform the same task, for example moving the ARM to the ground, next to the waste-paper basket. Such tasks, called repetitive tasks, can be performed more quickly by using *record and replay*. A practical example of an obstacle is the tray of the user's wheelchair. In this case study, the tray has been modelled as as a rectangular box, of which the width and depth equal the surface of the tray. The height of this box reaches from the floor to top of the tray, which has the advantage that the wheelchair, the part below the tray, is also considered as an obstacle.

A. Assistive Robotic Manipulator (ARM)

The ARM [1], assists disabled people having very limited or non-existent hand and/or arm function. Table I lists some characteristics of the ARM. The typical ARM user may suffer from Duchenne, Muscular Dystrophy (MD), Multiple Sclerosis (MS), Cerebral Palsy, Rheumatism, or spinal-cord lesions.

The ARM is mounted on a wheelchair, and allows a variety of Activitities of Daily Living (ADL) tasks to be carried out in the home, at work, and outdoors. These tasks include drinking from a glass, removing an item from a desk, scratching ones head, discarding an item in a trash receptacle, handling a floppy disk, or posting a letter. The ARM can be operated using a wide range of input devices that include, but are not limited to, a keypad (sixteen-buttons in a 4x4 grid), or a joystick (e.g. the joystick of the wheelchair). Additionally, a headband or spectacle mounted laser pointer, or other specially adapted device can be devised and constructed to function by the use of a non-disabled body part, such as the chin. Table I lists some technical characteristics of the ARM. Today, more than 275 ARMs are operational world wide. Since the commercial introduction of the Manus, and later of the ARM, the rehabilitation robots have proven to be safe, efficient, and highly appreciated assistive devices.

TABLE I
CHARACTERISTICS AND PROPERTIES OF THE ARM

Property	Value
Degrees of Freedom	6 + gripper + lift (8 in total)
Reach	80 cm + 25 cm (lift, optional)
Weight	13 to 18 kg depending on options
Max. payload	Up to 2 kg
Gripper	2 fingers with 4-point grasping
Gripper force	20N
Repeatability	+-1.5 mm
Max. velocity	9.9 cm/s (max. joint vel. 30°/s)
Safety features	Slip-couplings, limited velocity, limited acceleration and limited gripper force
Power supply	24VDC@1A cont., 3A peak (wheelchair batt.)
Input devices	Joystick, keypad, switches, UniScanner, EasyRider, etc.
Display	5x7 LED matrix, with buzzer
Control modes:	Cartesian & joint
ROI	1 to 2 years [1]

B. Human Machine Interface

The Human Machine Interface (HMI) of the ARM consists of an input device (e.g. keypad or joystick, etc.), a corresponding menu structure, and the display. The structure of the ARMs control menu was modified to include submenus for

record and replay. An example of the keypad menustructure, modified to include functionality for record and replay, is shown in Fig. 6. The replay functions make use of the above presented obstacle avoidance algorithm.

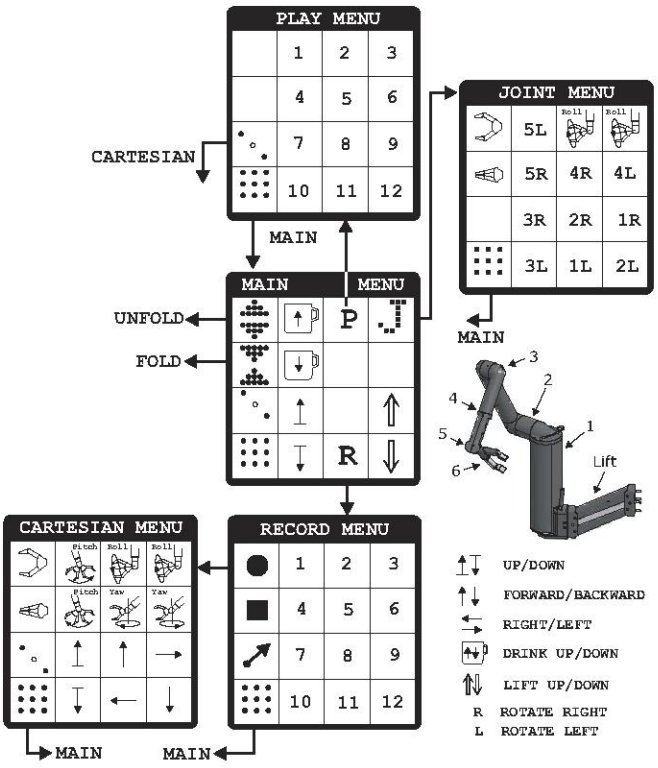

Fig. 6. Keypad menu structure extended with a record and play sub-menu.

The main menu (see MAIN MENU) allows the user to select between different modes of control of the robot. The joint mode (see JOINT MENU) allows the user to rotate each joint if the robot seperately. The Cartesian mode (see CARTESIAN MENU) allows the user to move the gripper along Cartesian axes x, y, and z, while maintaining the orientation of the gripper. The latter is especially useful when manipulating containers of liquids, for example a cup of coffee. In this mode, also the *yaw*, *pitch*, and *roll* of the gripper is controllable.

The fold submenu (not shown) allows the user to conveniently fold (i.e. park) the ARM compactly next to the wheelchair. The unfold submenu commands the ARM to move to its home position in front of the user. Both movements are preprogrammed.

If the user wants to store a target location, he or she will first move the gripper into the desired position using either the Cartesian or joint mode. Then, when pressing the R button in the main menu, the record submenu (see RECORD MENU) is activated. Targets to be recorded are stored in one of the twelve *patches*. Therefore, the user selects a patch by pressing the appropriate numbered button. Next, the lower left arrow button has to be pressed for 2 seconds, to store the target location in the patch. When pressing the main menu button (lower left), the main menu is activated. To reposition the gripper to a stored target, the submenu play (see PLAY MENU) must be activated by pressing the P button in the main menu. Next, one of the numbered buttons must be pressed to select the appropriate patch/target. A shortcut to the Cartesian mode submenu is available in this submenu.

The shown menu structure also allows the user to record a timed sequence of *actions*, instead of one target only. An action is a command available in the HMI, such as lift up for 2 seconds, rotate the fifth joint for 3 seconds, etc. Storing and recalling these timed sequences of actions is similar to storing and recalling targets. The two upper left buttons (circle and square) in the record submenu are used to start and stop recording of the sequence of these actions respectively. Note that, the patches in the play submenu do not discriminate between targets and sequences of actions.

A similar implementation of the HMI was developed for the menu structure of other input devices, such as joystick control.

C. Test Results

First, the obstacle avoidance algorithm was tested and optimized in a computer simulation environment. The simulation environment consisted of an empty room, containing the ARM itself and about 10 random located obstacles having random (limited) dimensions and orientation. For about 100 random environments, each having a random start and goal position, the algorithm was tested by letting it calculate an obstacle free path (see Fig. 7 for an example). The resulting path, consisting of several line segments, was then tested for obstacle collision. It was found that the algorithm resulted in incorrect paths, when a path had to be calculated around overlapping obstacles. The algorithm should be improved to be robust in cases of the presence of overlapping obstacles.

After simulation, a real enviroment was set up for testing the algorithm. The algorithm and HMI were tested for correctness and robustness in the laboratory by using a ARM and a table. The latter represents the tray of a wheelchair as an obstacle. These obstacles were described in a file, in which each obstacle was described by a total of four (three dimensional) vectors. Three of these vectors are used for orientation, of which the norm of the vectors indictated the dimensions of the obstacle. One of these four vectors was used for the position of the obstacle (relative to the base frame). After extensive testing, no obstacle collisions where found for the resulting paths.

In both situations (simulated and real), the algorithm was executed on a Pentium 3 machine, running at 350 MHz, having 64 MB of internal memory. Altough no calculation times where measured, the algorithm performed well on the target platform for the specified amount of obstacles.

The ARM is equipped with 6 axes, of which 5 have no mechanical nor functional rotation limit. In addition, it was chosen to fix the configuration of the elbow to be up (i.e.

pointing upward) in all cases of the (inverse) kinematics calculations. As a result of these characteristics, almost every path calculated by the obstacle avoidance algorithm is a path which can be executed by the ARM in the workspace in front of the user.

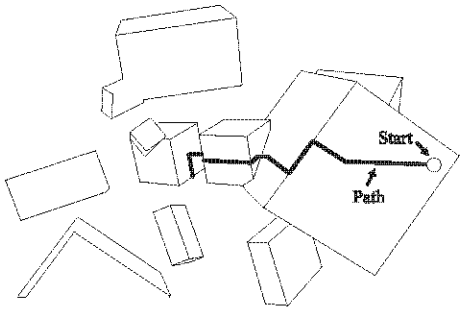

Fig. 7. Example of a calculated path in a random environment

V. CONCLUSION

A obstacle avoidance algorithm for the end-effector of a (rehabiliation) robot has been proposed, implemented and tested. It solves the problem of path planning with linear complexity (O(n)). The algorithm has been implemented on the Assistive Robotic Manipulator (ARM), and a Human-Machine-Interface was developed offering record and replay functionaly to the user. Although it is only possible to avoid static obstacles (relative to the robot), a significant improvement in usablility was achieved by implementing the algorithm on the ARM.

Future work includes, testing of the Human-Machine-Interface and the obstacle avoidance algorithm by users. Moreover, routines for the avoidance of self collission, as well as avoidance of links colliding with obstacles will be added to the algorithm. Future work also includes a procedure in which the users can define the obstacles by pointing the ARM to significant vertices of the obstacle.

REFERENCES

[1] G. R. B. E. Römer, H. J. A. Stuyt, & A. Peters. "Cost-Savings and Economic Benefits due to the Assistive Robotic Manipulator (ARM)". in *Proceedings of the 2005 IEEE 9th International Conference on Rehabilition Robotics*, Chicago, IL, USA, June 28 - July 1, 2005, pp. 201-204.

[2] Lozana-Perez, T. (1983), Spatial Planning: A Configuration Space Approach, *IEEE Transactions on Computers*, Vol. 32 (2), pp. 108-120

[3] Kavraki, L. E., Svestka, P., Latombe, J.-C., & Overmars, M. H. (1996), Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics & Automation*, Vol. 12(4), pp. 566580

[4] Russel, S.J. and Norvig, P., Artificial intelligence: A Modern Approach, 2003, Prentice-Hall

[5] Murphy, R. R., Introduction to AI Robotics, 2000, MIT Press

[6] Latombe, J. C., Robot Motion Planning, 1991, Springer