Continuous Collision Detection for a Robotic Arm Mounted on a Cable-Driven Parallel Robot
Diane Bury, Jean-Baptiste Izard, Marc Gouttefarde, Florent Lamiraux

To cite this version:

HAL Id: hal-02294402
https://hal.archives-ouvertes.fr/hal-02294402
Submitted on 23 Sep 2019

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.
Continuous Collision Detection for a Robotic Arm Mounted on a Cable-Driven Parallel Robot

Diane Bury\textsuperscript{1,2}, Jean-Baptiste Izard\textsuperscript{1}, Marc Gouttefarde\textsuperscript{3}, Florent Lamiraux\textsuperscript{2}

Abstract—A continuous collision checking method for a cable-driven parallel robot with an embarked robotic arm is proposed in this paper. The method aims at validating paths by checking for collisions between any pair of robot bodies (mobile platform, cables, and arm links). For a pair of bodies, an upper bound on their relative velocity and a lower bound on the distance between the bodies are computed and used to validate a portion of the path. These computations are done repeatedly until a collision is found or the path is validated. The method is integrated within the Humanoid Path Planner (HPP) software, tested with the cable-driven parallel robot CoGiRo, and compared to a discretized validation method.

I. INTRODUCTION

This paper deals with continuous collision checking for a cable-driven parallel robot (CDPR) with a robotic arm mounted on it. The method developed in this paper is applied to the robot CoGiRo, shown in Fig. 1, a redundantly-actuated cable-suspended CDPR developed by LIRMM and Tecnalia. The problem consists in determining exactly whether an input path between two configurations of the robot (consisting of the CDPR and the arm) is collision-free, or if there exists a configuration along the path for which one of the robot body or cable is in collision with another robot body, cable or static part of the environment. Although for academic instances of path planning problems, discretized collision checking is usually enough, exact or continuous collision checking is very important when dealing with real robots in industrial settings. In some applications like path length optimization for instance [1], it is important that if a path has been validated, any sub-path be computed as valid using the same validation algorithm. This is not the case with discretized collision checking since the samples tested for collision may not be the same.

The developed method is integrated within the existing open-source software Humanoid Path Planner (HPP) [2] which includes sampling-based planning algorithms such as Probabilistic RoadMaps (PRM) and Rapidly-exploring Random Trees (RRT), and different optimization methods.

Several previous works on CDPRs dealt with the issues of cable-cable, cable-platform and cable-object collisions. The determination of the loci of cable collisions within a prescribed workspace is presented in [3], [4] in the case of a constant-orientation workspace and in [5], [6], [7] in the case of a 6D workspace. Fast heuristic approaches are proposed in [7] whereas certified calculations based on interval analysis are introduced in [5], [6]. Besides, an approximate determination of the volume swept by a cable when the mobile platform of a CDPR moves within a prescribed workspace is discussed in [8] and the geometric determination of cable-cylinder collision loci within the workspace of a CDPR is dealt with in [9]. In the latter, the cylinder is a fixed object located inside the CDPR workspace. Moreover, in [10], the collision-free printing workspace is calculated for fully-constrained CDPRs intended to print large-dimension objects in a sequence of horizontal layers. While all these methods determine various types of cable collision loci within a prescribed workspace, other previous works address the issue of checking cable collisions along a prescribed CDPR mobile platform path [5], [11], [12], which is also the purpose of the present paper. In [12], a classic discretized collision checking applied to CDPRs is presented. More advanced methods based on interval analysis, which can account for parameter uncertainties and round-off errors in numerical calculation, are introduced in [5], [11].

The contribution of the present paper is a continuous cable collision checking method integrated within the open-source software HPP which can test collisions along a prescribed path of the mobile platform of a CDPR. Compared to [12], the proposed method allows continuous collision checking instead of discretized checking. The methods presented in [5], [11] to check cable collisions along a path can be classified as continuous checking method. Indeed, a proper use of interval analysis ensures that there does not exist any robot configuration along the path where a cable collision can occur, taking into account model parameter uncertainties and round-off errors. However, to the best of our knowledge,
the paper. HPP are described in Section IV and Section V concludes different types of pair of bodies. Implementation results in velocity upper bound and the distance lower bound for the Section III details the different methods used to calculate the and the overall algorithm used to validate a continuous path.

A. Definitions and notations

We consider a CDPR composed of a mobile platform suspended by cables attached to a fixed frame structure, as depicted in Fig. 2. An articulated robotic arm is mounted on the platform for the purpose of grabbing and moving objects in the environment. Some simplifications are made to modelize the cables:

- The cables are considered to be cylinders.
- Sagging of the cables is either neglected or taken into account in the cylinder radius.
- Each cable exits the fixed structure at a fixed point \( A_i \).
- Each cable is attached to the mobile platform at a point \( B_i \), fixed in the platform frame.

During motion planning, straight paths are computed and must be checked against collisions. We assume that along a straight path, the platform rotates or translates at a constant linear or angular velocity — respectively \( v_p \) and \( \omega_p \) — in the fixed reference frame. To validate a path in regard to collisions, every pair of bodies has to be checked for collisions. Those pairs are called collision elements, and can be of different types: a cable-cable collision, a cable-platform collision, a cable-arm collision, or robot-robot collision. In the software, collision element types are represented as derived classes of an abstract class, CollisionElement. The path to validate is set at the very start of the algorithm for all collision elements. This class has a method ValidateElement, which takes as input the time parameter \( t \) corresponding to the configuration to validate. The output is a Boolean value indicating whether or not the configuration at the time parameter \( t \) is valid, in which case the method also returns a collision-free interval containing \( t \), denoted as interval. The size of interval is computed using an upper bound on the velocity \( V_{max} \) of one of the bodies of the pair relative to the other, and a lower bound on the distance between the two bodies \( D_{min} \). The calculations for those bounds differ depending on the type of the collision element as detailed in section III.
Algorithm 1 Validation of a straight path using the dichotomy method

1: function VALIDATESTRAIGHTPATH(path)
2:     t ← 0
3:     validSubset ← ∅
4:     valid ← True
5:     while valid is True and validSubset ≠ [0,T] do
6:         success, validInterval ← VALIDATEALLELEMENTS(t)
7:         if not success then
8:             valid ← False
9:         else
10:             validSubset ← validInterval ∪ validSubset
11:         end if
12:     t ← middle of first interval of validSubset
13:     end while
14:     if not valid then
15:         return first interval of validSubset
16:     else
17:         return [0,T]
18:     end if
19: end function

B. Continuous validation algorithm

1) Validation of a straight path: The input path to validate is a concatenation of linear interpolation paths that are called straight paths. Thus we need to successively validate each straight path using the function VALIDATESTRAIGHTPATH, until a collision is found or until all straight paths have been validated.

Let us denote by [0,T] the interval of definition of the straight path to validate. If I is a subset of real numbers, we denote by Ī the complement of I in [0,T]. Algorithm 1 validates a path by creating a subset of validated intervals, validSubset. This subset is initialized with the empty set. The algorithm then loops until validSubset is equal to [0,T]. t is initialized to 0. Function VALIDATEALLELEMENTS is called with t and returns an interval containing t valid for all the collision elements. validSubset is augmented with this newly validated interval (the new validSubset is the union of the previous validSubset and the newly validated interval). Then t receives the parameter value of the middle of the first non-tested interval. The algorithm then progresses until either [0,T] is entirely validated, or a collision is found.

2) Finding a valid interval for all collision elements around a given configuration: As detailed in Algorithm 2, the function VALIDATEALLELEMENTS takes as input a valid parameter value in [0,T] and returns a valid interval containing the input parameter value. The function loops over all the collision elements, stored in a list collisionElemList. A local variable validInterval is initialized to ]−∞:+∞[ and represents the interval currently validated. For each collisionElem, an interval valid for this collision element only is calculated with VALIDATEELEMENT and the total valid interval validInterval is trimmed by doing an intersection with this newly calculated interval.

3) Validating an interval for one collision element: The function VALIDATEELEMENT returns an interval around the given time parameter t for a collision element. Each collision element stores the set of intervals currently validated for a path. This variable is reset each time the algorithm starts working on a new path. Before computing a valid interval around t, it is first checked if the interval to validate has already been validated.

If not, an upper bound Vmax of the velocity of one of the bodies relative to the other body of the pair is calculated, as well as a lower bound Dmin of the distance at parameter t between the two bodies of the element (see next section for the calculation methods). If a collision is found during the calculation of Dmin, VALIDATEELEMENT returns false and reports the collision. If no collision is found, the value of Dmin is a strictly positive real number. We define the output valid interval newValidInterval in (1).

\[
\text{newValidInterval} = [t - \frac{D_{\text{min}}}{V_{\text{max}}}, t + \frac{D_{\text{min}}}{V_{\text{max}}}] \quad \text{(1)}
\]

III. CALCULATION OF THE VELOCITY AND DISTANCE BOUNDS

A. Collision between two bodies of the robot

As previously stated, for a pair of non-cable bodies of the robot (mobile platform, fixed structure, arm body), we calculate Vmax using the method of [13]. Dmin is computed using the FCL library [14] (BSD License). If the pair is composed of a link and its child link, then the collision checking is disabled, because the two bodies either touch each other by design, or cannot be in collision due to joint bounds.

B. Collision between a cable and the platform

Because each cable i is by design in contact with the platform at its attachment point, the method of [13] used for two bodies of the robot cannot be used as it stands. We choose to simply consider a shortened version of the cable that stops at a fixed distance d of the cable attachment point Bi on the platform. We calculate Vmax an upper bound on the velocity of any point of the cable relative to the platform, and Dmin, a lower bound on the distance between the shortened cable and the platform.

A previous iteration of our algorithm used a sampling-based model, computed off-line, to reduce the computing time, but at the expense of the accuracy.

Calculation of a velocity upper bound

We consider a straight path, for which the linear and angular velocities of the platform are constant and known as stated in Section II. They are respectively noted vP and ωp, and we note ωp = ∥ωp∥2 and vp = ∥vP∥2. C is the origin of the platform frame. We want to compute an upper bound on the velocities of all the points of cable i in the platform frame.
Algorithm 2 Validation of a configuration by validating each interval element

1: function VALIDATEALLCOLLISIONELEMENTS(t)
2:   validInterval ← −∞, ∞
3: for each collision element collisionElem in collisionElemList do
4:      (valid, newValidInterval) ← VALIDATEELEMENT(collisionElem, t, validInterval)
5: if not valid then  // there is a collision for config(t)
6: return (False, validInterval, report)
7: else  // there is no collision for config(t)
8:   validInterval ← validInterval ∩ newValidInterval
9: end if
10: end for
11: return (True, validInterval)  // validInterval is continuously valid for every collision element

For cable $i$, we consider an orthogonal local frame $F^B_i$ centered on $B_i$ with its x-axis aligned with $B_iA_i$ and directed toward $A_i$. Let $P_i$ be the point on cable $i$ at a fixed distance $r$ of $B_i$. The relation between the coordinate vector $P^B_i$ of $P_i$ in the platform frame and its coordinate vector $P^i$ in the local cable frame $F_i$ is given by (2).

$$
\left( \begin{array}{c}
P_i^p \\
1
\end{array} \right) = M_{i/p} \left( \begin{array}{c}
P_i^i \\
1
\end{array} \right)
$$

$M_{i/p} = \left( \begin{array}{ccc}
R_{i/p} & T_{i/p} \\
0 & 0 & 1
\end{array} \right)$ is the homogeneous matrix representing the position and orientation of the local cable frame in the platform frame. $R_{i/p} \in SO(3)$ is a rotation matrix. $T_{i/p} = \overrightarrow{C_iB_i} = b_i \in \mathbb{R}^3$ is the position of the attachment point $B_i$ in the platform frame. Since $B_i$ is fixed in the platform frame, $T_{i/p}$ is constant. With $P_i^i = (r \ 0 \ 0)^T$ being constant, by differentiating (2) with respect to time, we get

$$
\left( \dot{P}_i^p \right) = \left[ \omega_{i/p} \right] \times \left( \begin{array}{c}
P_i^p \\
1
\end{array} \right)
$$

(3)

where $\dot{P}_i^p$ is the velocity of point $P_i$ in the platform frame, $[\omega_{i/p}] \times$ is the antisymmetric matrix corresponding to the cross product with the cable frame angular velocity vector $\omega_{i/p} \in \mathbb{R}^3$ in the platform frame, of norm $\omega_{i/p}$. This gives:

$$
P_i^p = [\omega_{i/p}] \times R_{i/p} P_i^i
$$

(4)

Therefore, we can bound the norm:

$$
\|\dot{P}_i^p\|_2 \leq \omega_{i/p}\|P_i^i\|_2
$$

(5)

By composition of angular velocities, we have $\omega_{i/0} = \omega_{i/o} + \omega_{o/p}$, and by bounding the norm: $\omega_{i/p} \leq \omega_{i/o} + \omega_p$, with $\omega_{i/o}$ being the angular velocity of the cable joint frame $i$ in the global reference frame.

Since $B_i$ is fixed in the platform frame, which has known fixed linear and angular velocities $v_p$ and $\omega_p$, $B_i$ is moving around $A_i$ with a linear velocity of norm $v_{B_i} \leq v_p + \omega_p b_i$. Since $\|A_iB_i\|_2 \geq L_{i,min}$, we get (6) and then (7).

$$
\omega_{i/0} \leq \frac{v_p + \omega_p b_i}{L_{i,min}}
$$

(6)

$$
\omega_{i/p} \leq \frac{v_p + \omega_p b_i}{L_{i,min}}
$$

(7)

From (5) and (7), and knowing we also have an upper bound on $\|P_i^i\|_2$ which is $L_{i,max}$, we get:

$$
\|\dot{P}_i^p\|_2 \leq L_{i,max} \left( \frac{v_p + \omega_p b_i}{L_{i,min}} + \omega_p \right) = V_{max}
$$

(8)

where $b_i$ is a known constant value depending on the design of the CPDR, and $L_{i,min}$ and $L_{i,max}$ can be considered either as constant values depending on the shape of the workspace, or calculated for each straight path. Eq. (8) gives an upper bound on the velocity of point $P_i$ in the platform reference frame along a straight path.

Calculation of a distance lower bound

The FCL library is used to compute a lower bound $D_{min}$ on the distance between a cable $i$ and the mobile platform. A STL file represents the 3D collision model of the platform, while the cable is represented by a cylinder. Since the cable and the platform are connected at point $B_i$, as stated above, we consider a shortened portion of the cylinder $A_iB_i$ where $\overrightarrow{B_iB_i} = d\overrightarrow{B_iA_i}$, $d$ is a fixed distance chosen so that for any configuration, if a point of $[B_iB_i]$ is in collision with the platform, then at least a point of $[B_iA_i]$ is also in collision with the platform. By design, $D_{min}$ will never be greater than $d$, and it should be noticed that the closest point between the platform and the cable will generally be the (virtual) end of the cable $B_i$, i.e. $D_{min} = d$. Choosing $d$ as large as possible reduces the number of iterations necessary to validate an interval, and thus the computing time.

C. Collision between two cables

If two cables have the same attachment points on the platform or the same exit points on the base structure, collision checking between them is disabled since these two cables cannot collide. For all other cable pairs, a velocity upper bound $V_{max}$ and a distance lower bound $D_{min}$ are calculated and used in (1).

Calculation of a velocity upper bound

We need to find an upper bound on the velocities of all the points of cable $i$ in the local frame of cable $j$. Let $P_i$ be
the point on cable \( i \) at a fixed distance \( r \) of \( B_i \), with \( P_i^j \) its coordinate vector in the local frame of cable \( j \). Similarly to the calculations in Section III-B, we can write:

\[
\begin{pmatrix}
0 \\
1
\end{pmatrix} = M_{i/j} \begin{pmatrix}
P_i^j \\
1
\end{pmatrix}
\tag{9}
\]

where \( M_{i/j} = \begin{pmatrix}
R_{i/j} & T_{i/j} \\
0 & 1
\end{pmatrix} \) denotes the homogeneous matrix defining the position and orientation of the local frame of cable \( i \) with respect to the local frame of cable \( j \). \( \omega_{i/j} \) is the rotation matrix of norm \( \omega_{i/j} \) of the local frame of cable \( i \) relative to the local frame of cable \( j \).

Differentiating with respect to time, and knowing that \( P_i^j = (r \ 0 \ 0)^T \) and \( T_{i/j} = B_iB_j \) are constant:

\[
\begin{pmatrix}
\dot{P}_i^j \\
0
\end{pmatrix} = \begin{pmatrix}
|\omega_{i/j}| \times R_{i/j} \\
0 \\
0
\end{pmatrix} \begin{pmatrix}
P_i^j \\
1
\end{pmatrix}
\tag{10}
\]

We then have \( \dot{P}_i^j = |\omega_{i/j}| \times R_{i/j} P_i^j \). This gives \( \|\dot{P}_i^j\|_2 \leq |\omega_{i/j}| \|P_i^j\|_2 \). We have \( \omega_{i/j} = \omega_{i/o} - \omega_{j/o} \). Using (6) and knowing also we have an upper bound on \( \|P_i^j\|_2 \) which is \( L_{i,max} \), we obtain:

\[
\|\dot{P}_i^j\|_2 \leq L_{i,max} \left( \frac{v_p + \omega_j b_j}{L_{i,min}} + \frac{v_p + \omega_j b_j}{L_{i,min}} \right) = V_{max}
\tag{11}
\]

**Calculation of a distance lower bound**

We use the FCL library to compute a lower bound on the distance between two cables, represented by cylinders. The radius of the cylinder depend on the actual diameter of the cable, and can take into account a safety margin if desired. For two cables, the distance between their attachment points on the platform is a fixed upper bound on the distance between the cables and thus on \( D_{min} \). It means that the proposed method is slower for a CDPR with cables attached close to another on the platform than for a CDPR with cables attached far from one another. Indeed, if the cable attachment points are close, \( D_{min} \) is small and the algorithm can only validate small portions of the path.

**D. Collision between a cable and the robotic arm**

We consider a cable \( i \) and a body of the robotic arm, attached to joint \( J_a \). Let \( J_0 = J_p, J_1, ... , J_{m-1} = J_a \) be the list of joints linking the platform joint \( J_p \) to \( J_a \). \( J_1 \) is the joint attaching the robotic arm to the platform.

**Calculation of a velocity upper bound**

Let \( P_i \) be a point on cable \( i \), and \( P_i^b \) its coordinate vector in the joint frame \( J_{a} \).

\[
\begin{pmatrix}
P_i^a \\
1
\end{pmatrix} = M_{j_{m-2}/j_{m-1}} M_{j_{m-3}/j_{m-2}} ... M_{j_{0}/j_{1}} M_{i/p} \begin{pmatrix}
P_i^b \\
1
\end{pmatrix}
\tag{12}
\]

Where \( M_{j_{k}/j_{k+1}} = \begin{pmatrix}
R_{j_{k}/j_{k+1}} & T_{j_{k}/j_{k+1}} \\
0 & 1
\end{pmatrix} \) is the homogeneous matrix representing the position of joint \( J_{k} \) in the reference frame of \( J_{k+1} \).

By differentiating (12) with respect to time, we obtain

\[
\begin{pmatrix}
\dot{P}_i^a \\
0
\end{pmatrix} = \begin{pmatrix}
\omega_{j_{m-2}/j_{m-1}} R_{j_{m-2}/j_{m-1}} & 0 & ... & 0 \\
0 & 0 & ... & 0 \\
\dot{P}_i^b \\
0
\end{pmatrix} \begin{pmatrix}
P_i^b \\
1
\end{pmatrix}
\tag{13}
\]

Then by bounding the norm using properties of rigid-body transformations:

\[
\|\dot{P}_i^a\|_2 \leq v_{i/p} + \omega_{i/p} \|P_i^b\|_2 \\
+ v_{j_{0}/j_{1}} (\|P_i^b\|_2 + \|T_{i/p}\|_2) \\
+ v_{j_{1}/j_{2}} (\|P_i^b\|_2 + \|T_{i/p}\|_2 + \|T_{j_{0}/j_{1}}\|_2) \\
\vdots \\
+ v_{j_{m-2}/j_{m-1}} (\|P_i^b\|_2 + \|T_{j_{m-3}/j_{m-2}}\|_2 + \|T_{j_{m-2}/j_{m-1}}\|_2)
\tag{14}
\]

We note \( D_k \) the cumulative length of joint \( J_k \):

\[
D_0 = 0 \\
D_k = \sum_{i=0}^{k-1} \|T_{j_{i}/j_{i+1}}\|_2 \text{ for } k \geq 1
\tag{15}
\]

Since \( L_{i,max} \) being an upper bound on \( \|P_i^b\|_2 \), the origin \( B_i \) of the local cable frame is fixed in the platform frame so \( v_{i/p} = 0 \), and using (7), we obtain an upper bound on the velocity of all points of cable \( i \) relative to the frame of joint \( J_a \):

\[
\|\dot{P}_i^a\|_2 \leq \omega_{i/p} + \frac{v_{p} + \omega_{j_{k} b_{j_{k}}}}{L_{i,min}} L_{i,max} + \sum_{k=0}^{m-2} (v_{j_{k}/j_{k+1}} L_{i,max} + \|T_{j_{k}/j_{k+1}}\|_2 + D_k))
\tag{16}
\]

\( \|T_{j_{k}/j_{k+1}}\|_2 \) and all the \( \|T_{j_{k}/j_{k+1}}\|_2 \) are known by design of the robot. The same reasoning can be applied to validate collisions between a cable and an object of the environment.

**Calculation of a distance lower bound**

The FCL library is used once again to compute a lower bound on the distance between a cable, represented as a cylinder, and the body of the robotic arm, represented by a 3D collision model.

**IV. IMPLEMENTATION RESULTS IN HPP**

We have implemented the continuous collision checking method proposed in Sections II and III in the software HPP. The CoGiRo CDPR is simulated in HPP. The continuous method is compared to a discretized collision checking method, which uses a configuration validation method and a time step \( \tau \).

The computer used to run the software HPP is an "Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz" with 4,096 KB of cache memory and 16 GB of RAM.

This benchmark is performed on CoGiRo with the robotic arm. Random *straight paths* are generated by shooting random configurations in the configuration space, and keeping
<table>
<thead>
<tr>
<th>$\tau$ (s)</th>
<th>True pos</th>
<th>True neg</th>
<th>New true pos</th>
<th>False pos</th>
<th>False neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>633</td>
<td>362</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.01</td>
<td>634</td>
<td>362</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.001</td>
<td>634</td>
<td>362</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table I
Results for the continuous validation compared to the discretized validation with a time step $\tau$ for 1000 random straight paths

the first two valid configurations. Each straight path is validated using the continuous method and the discretized method with different time steps. Results are put into five categories:

- True positive: a collision was both detected by the discretized and the continuous methods.
- True negative: no collision were detected either by the discretized nor the continuous method.
- New true positive: a real collision was found by the continuous method, but was not detected by the discretized method.
- False positive: a collision was detected by the continuous method but is not an actual collision as determined by the configuration validation method.
- False negative: the continuous method failed to detect a collision that was found by the discretized method.

Results are resumed in Tables I and II with different time steps tested for the discretized method. The smaller the time step is, the bigger the average computation time is for the discretized method.

As shown in Table I, there is no false positive, thanks to the fact that the FCL library only returns zero as a lower bound of the distance between two objects if they are actually in collision. The continuous method is guaranteed to find every collision, thus there is no false negative. The discretized method misses collisions that the continuous method is able to find, even when the time step is reduced. With a time step of 0.001s, the discretized method has a longer computing time and fails to detect collisions. These results show the efficiency of the continuous method, and its usefulness in situations where no collision is tolerated.

V. CONCLUSIONS

This paper introduced a method which extends a continuous collision checking method for a multi-arm robot to a CDPR with a robotic arm mounted on its platform, to take into account collisions including the cables. The method has been integrated within the software HPP and tested in simulations on the robot CoGiRo. Results show that the method is effective and is able to find collisions which are undetected by a discretized method even with a small time step. Although our method is exact and can validate a path regarding collisions, it does not ensure the path is adapted to be performed on the physical robot. The algorithm could be improved using a cost-base planning method to maximize distance to obstacles. Future work will include applications on the physical robot CoGiRo to plan movements in a cluttered environment.