OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible.

This is an author's version published in: http://oatao.univ-toulouse.fr/23621

To cite this version:


Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr
H-RRT-C : Haptic Motion Planning with Contact

Nassime Blin¹, Michel Taïx², Philippe Fillatreau³ and Jean-Yves Fourquet³

Abstract— This paper focuses on interactive motion planning processes intended to assist a human operator when simulating industrial tasks in Virtual Reality. Such applications need motion planning on surfaces. We propose an original haptic path planning algorithm with contact, H-RRT-C, based on a RRT planner and a real-time interactive approach involving a haptic device for computer-operator authority sharing. Force feedback allows the human operator to keep contact consistently and provides the user with the feel of the contact, and the force applied by the operator on the haptic device is used to control the roadmap extension. Our approach has been validated through two experimental examples, and brings significant improvement over state of the art methods in both free and contact space to solve path-planning queries and contact operations such as insertion or sliding in highly constrained environments.

I. INTRODUCTION

Our goal is to perform industrial assembly/disassembly tasks in an immersive Virtual Reality (VR) environment by using a haptic device and automatic motion planning methods. In order to do this, we propose a novel approach based on the combination of haptic device, virtual reality and path planning techniques.

Many works show the advantages of combining the use of a virtual reality environment and a haptic device providing force feedback. The authors of [1] assess the advantages of using both techniques simultaneously rather than separately, for education and training to manipulation tasks. [2] explains how interactive simulation using haptic feedback benefits from the cognitive and manual skills of the operator in the case of complex industrial assembly tasks.

In the robotics community, probabilistic motion planning techniques, such as RRT or PRM, have been intensively studied [3], [4] for assembly task. Such methods are generic but can be very slow to solve problems in a highly cluttered environment.

Several papers present interactive motion planning approaches featuring the use of motion planning techniques with a human operator in the loop [5], [6], [7], [8], [9]. In other studies, the user interaction can be made using a haptically controlled object to modify or define critical object configurations, [10], [11], [12], [13], [14].

All these previous works show the great significance of contact for realistic simulation of industrial tasks. We have recently started works [15] tackling the issue of interactive path planning with contact but with a naive approach and without immersion. We proposed a first interactive algorithm, based on a RRT algorithm, called I-RRT-C, and able to explore the whole workspace (free and contact spaces) and to plan directly on surfaces. Our approach brought promising improvements (more relevant paths obtained and drastic reduction of processing times, for better real time interaction) for path planning queries in highly cluttered environments [16].

The main limitation of [15] was due to the interactive device. We used a 6D mouse to move the object and to create contact with the environment. Improving the guidance for user and algorithm is impossible (or difficult) without force/torque contact feedback between them.

In this paper our main contribution is to take into account contact information to assist the user and the algorithm. A new immersive and interactive path planning algorithm with contact called H-RRT-C is developed. A haptic arm and a virtual reality environment allow to exchange contact informations between the algorithm and the user.

The following improvements are brought: a) force feedback allows the human operator to keep contact more consistently b) the force applied by the operator on the haptic device is used to control the roadmap extension and c) the orientation of the manipulated part is now variable.

This paper is organized as follows. In section II we present our previous interactive motion planning algorithm in contact, then our new contact method. Section III presents our multimodal interactive motion planner in free and contact spaces, section IV presents our software and hardware experimental architecture. V presents our experimental results and section VI our conclusions and the next steps of our works.

II. CONTACT HAPTIC PATH PLANNING

A. Interactive Motion Planning in Contact

The present work is based on our previous contribution, I-RRT-C [15] : Interactive Motion Planning in Contact without immersion. The aim of this I-RRT-C algorithm is to let the user cooperate with the computer. Using the speed of the computer and the cognitive capacities of the user, we observe significant improvement over a standalone RRT.

The user is controlling the object to be planned in the workspace using a 6D mouse. The configuration of the object handled by the user is called $q_{device}$. We introduced a parameter (see algorithm 1) allowing to share authority between the computer and the user, called $\alpha$. For each sample, a random number $a$ is picked. If $a \leq \alpha$, a random configuration is shot and added to the roadmap if it is not
in collision. If \( a > \alpha \) the chosen configuration is \( q_{\text{device}} \). By doing so, we can set given percentages of processing times allocated to the capture of human-defined and the computation of machine-defined input configurations.

**Algorithm 1 Interactive Planning**

**Require:** \( W, T, \alpha, q_{\text{device}} \)

1: loop
2: \( a \leftarrow \text{rand}(0, 1) \)
3: if \( a > \alpha \) then
4: \( q_{\text{current}} \leftarrow q_{\text{device}} \)
5: \( T \leftarrow \text{Add_Tree}(q_{\text{current}}) \)
6: else
7: \( q_{\text{current}} \leftarrow \text{Random_Shooter()} \)
8: \( T \leftarrow \text{Add_Tree}(q_{\text{current}}) \)
9: end if
10: end loop

The second contribution of I-RRT-C is ContactSampling. When the user approaches an obstacle with the guided object, the planner switches to contact mode. The algorithm samples configurations on the surface the user approached with the object. This work suffered from many limitations, in particular we didn’t consider the manner in which the contact is created by the user as it is impossible with a 6D mouse.

**B. Haptic Sampling**

Our novel algorithm, called H-RRT-C, described in section III, is capable of interactive planning in contact while using our immersive architecture described in section IV. We describe here the principal contribution: the contact subpart called **HapticSampling**.

First, we improve contact sampling. Previously, random configurations were sampled on the contact plane bounded by the workspace limits. Now, we sample inside an ellipse \( \mathcal{E} \) centered at the contact point. The ellipse parameters depend on the user’s intention. Samples are constrained around a user-defined point; this allows to drastically reduce colliding points or irrelevant samples not consistent with the user’s sampling intention.

Second, we implemented a method changing the sampling behavior depending on the actions of the user. When getting in collision with an obstacle, the user feels force feedback, thanks to a haptic arm and a collision detection algorithm provided by the haptic arm constructor. Thus, we can use the force provided by the operator through the device at any time. We can also measure the position and orientation of the moving object handled with the haptic arm.

This algorithm adapts contact sampling using the intentions of the operator by measuring three parameters provided by the haptic device.

1) **The intensity of the colliding user force** \( \| f_u \| \): we state that the bigger this force is, the more interested the user is with the colliding surface. It traduces the interest of the operator for the current contact surface. For example, when the operator slightly touches a surface then goes away, this may be either by mistake or intentionally. On the opposite, if a user pushes firmly the manipulated object towards a surface, we consider this is done intentionally. The surface of \( \mathcal{E} \) is made proportional to the force applied by the operator. If \( \| f_u \| \) is small, the surface of the ellipse will be small. If \( \| f_u \| \) grows, so will the surface of \( \mathcal{E} \). If a user collides with an edge or a surface instead of a point, the behavior is the same.

2) **The angle to normal \( \varphi \):** it expresses the operator’s intention to move. When the user collides with an obstacle surface, we get the first colliding point, and compute the normal \( n_e \) to the contact surface at this point. If the angle \( \varphi \) to the normal to the contact surface \( n_e \) is equal to zero (i.e. the force applied by the user is perpendicular to the contact surface), we have no information about any intention and sample inside a disk. If the operator lets the manipulated object to slide along the contact surface, the measure of \( \varphi \) allows to determine his intentions and the algorithm adapts by sampling more along one direction by elongating the ellipse, see figures 1 and 2.

We compute \( \varphi \) as followed:

\[
\varphi = \cos^{-1} \frac{f_u \cdot n_e}{\| f_u \| \cdot \| n_e \|}
\]

(1)

If this angle is zero, the minor and major axis of the ellipse are the same length and the ellipse becomes a disk. Whenever this angle grows, the ellipse becomes more and more elongated; the length of the major axis increases and the length of the minor axis decreases. Let \( s_a \) be the length of the minor axis and \( s_b \) the length of the minor axis and major big axis respectively. We compute the axis lengths as follows:

\[
b_a = \exp(2 \cdot \varphi) \quad s_a = 1/b_a
\]

(2)

3) **The \( \Delta_p \) vector:** it indicates towards where the user is actually moving. Its projection on the ellipse is \( \Delta_t \). The big axis of \( \mathcal{E} \) is aligned with this vector’s projection on the contact surface. This permits to sample in the preferred direction, see figure 3.

While being in contact, we compute the \( \Delta_p \) vector formed by the subtraction of two consecutive positions. \( \Delta_p \) corresponds to the motion intended by the user.

We use the tangent vector to the surface \( t_e \) and \( \Delta_p \) to compute the orientation \( \theta \) of the ellipse. First \( \Delta_p \) is projected on the sampling surface: \( \Delta_t = \Delta_p - \Delta_p \cdot n_e \times n_e \).

Then, we compute \( \theta \):
\[ \theta = \cos^{-1} \frac{\Delta_t \cdot t_c}{\|\Delta_t\| \cdot \|t_c\|} \]  

(3)

\[ \text{Algorithm 2 HapticSampling} \]

\begin{algorithm}
\begin{algorithmic}[1]
\Require \text{W}
\State \textbf{if} contact \textbf{then}
\State \textbf{end if}
\end{algorithmic}
\end{algorithm}

In lines 2, 3 and 4, we read the information given by the haptic arm. Using the haptic arm, the user produces a 3D force \( f_u \) when the handled object touches an obstacle in the environment. This force quickly grows when in contact because of collision detection and force feedback. The operator can push firmly against obstacles, thus generating a large force \( f_u \) (for example in the case of an insertion task).

With these different informations we can define the characteristics of the sampling ellipse \( E \), line 5.

- the surface of the ellipse in which configurations will be shot depends on \( \|f_u\| \)
- \( \varphi \) defines the ratio between the two axes
- \( \theta \) defines the orientation of the ellipse

Having a shape for the ellipse, we call the ContactSampling method, line 6 which randomly draws one configuration returned line 7. It is drawn at the surface of the chosen obstacle, inside an ellipsoidal subpart of this surface.

ContactShooter works in the following way: we have six degrees of freedom for the object. The three rotations are given by the operator moving the object. We keep the distance to contact null. Two parameters are left to sample randomly, these are the surface coordinates. They are sampled inside the contact surface subpart bounded by the ellipse. Contact sampling stops when the operator moves away from surface.

\section*{D. Examples}

The following example presents the way the three parameters \( \|f_u\|, \varphi \) and \( \theta \) are used by sampling on a planar surface. Various situations are presented to describe the use of every parameter. For each example, the user is pushing a cube against the green wall.

Figure 4 presents an example where the user pushes gently towards the green surface. Configurations are sampled in a small area around the contact point. As the user pushed the cube orthogonally, the ellipse is round, not elongated.

Figure 5 presents a case where the user also pushed the cube orthogonally to the surface. This time, he pushed stronger and the ellipse is still round but much bigger.

Next is figure 6 where we see a very elongated ellipse, this is because there is a big \( \varphi \) angle between the user force \( f_u \) and the normal to the surface \( n \).

Figure 7 represents the results obtained when the user pushes stronger. The figure shows a first small ellipse surrounded by a much bigger one. Borders are displayed using dashed lines for clarity. The user pushed slightly then pushed more firmly. These two actions gave two ellipse sizes.

\section*{III. Interactive Haptic Path Planning}

Using the HapticSampling algorithm in an interactive approach we can plan trajectories for an object manipulated in a highly constrained environment and achieve tasks such as sliding and insertion by using force feedback for better control on (faster) contact sampling.

Our new algorithm permits authority sharing when sampling in contact which was not allowed in I-RRT-C algorithm
because contact sampling on the contact surface was completely automatic. Now we use the intentions of the operator during contact with three different parameters to speedup the overall process.

While being in contact, the user can change the orientation of the object he handles. This was not possible in our previous work as the operator had to choose a specific orientation before getting in contact. Now we can plan in contact for different orientations that can be changed in real time.

Algorithm 3 H-RRT-C: Haptic RRT in Contact

Require: $W, T, HapticArm, \alpha$

\begin{verbatim}
1: loop
2: if contact then
3: \hspace{1cm} $q_{current} \leftarrow$HapticSampling(HapticArm)
4: \hspace{1cm} $T \leftarrow$ Add_TREE($q_{current}$)
5: else
6: \hspace{1cm} $a \leftarrow$ rand(0, 1)
7: \hspace{1cm} if $a \leq \alpha$ then
8: \hspace{2cm} $q_{current} \leftarrow$ RandomShooter()
9: \hspace{2cm} $T \leftarrow$ Add_TREE($q_{current}$)
10: else
11: \hspace{2cm} $q_{current} \leftarrow q_{device}$
12: \hspace{2cm} $T \leftarrow$ Add_TREE($q_{current}$)
13: end if
14: end if
15: end loop
\end{verbatim}

The planner is either in contact mode or in free space mode. In free space mode, a random number chooses human or machine mode. Contact mode starts when at least one collision is detected. We get out of contact mode as soon as no more collisions are detected. We choose the sampling surface as the plane defined by the normal to the contact surface at the first contact point.

Line 3, we call HapticSampling described in algorithm 2. It adds a valid contact configuration to the tree (line 4) using the operator’s command.

Line 5, the algorithm is not in contact mode. A random number $a$ between 0 and 1 is shot (line 6).

Line 7 we choose human or machine mode as in algorithm 1. If $a \leq \alpha$ we are in machine mode otherwise we enter human mode (line 10). The $\alpha$ parameter is a fixed number describing the percentage of machine input configurations.

Line 8 is the probabilistic motion planning mode where a completely random configuration is shot and added to the tree if in free space.

Line 11 is the human mode. The current configuration $q_{current}$ of the object held by the operator in the free space is added to the tree $T$.

Our examples show that the sampling surface in contact can be changed in real time with three parameters $\theta, \varphi$ and $\Delta_t$ given by the operator. The shape of the ellipse traduces his intention. The instantaneous displacement is the direction in which the object should move. The applied force is the interest the user have in the current surface.

IV. ARCHITECTURE

Our last contribution is an architecture that permits interactive motion planning using the LAAS-Gepetto motion planner: HPP [17], along with a virtual reality (VR) environment provided by LGP-ENIT laboratory.

A. Architecture Overview

The VR environment uses a haptic arm Virtuose 6D 35-45. It is a sensorimotor bidirectional device. We also have a large 3D screen and a motion capture system attached to the 3D glasses.

Two separate simulations are run simultaneously, on two separate computers. Motion planning is run with HPP [17] embedding our H-RRT-C algorithm. It is coded in C++ and Python and runs on a first computer under Ubuntu 14.0.

The second computer runs Virtools® under Windows XP® controlling the VR environment the user will interact with. Some Virtools scripts control the simulation, the motion capture and the haptic arm.

Both computers use the same mesh models of the environment and the robot. We use ZeroMQ [18] to communicate through the network between these two different computers. Our ZeroMQ client is coded in C++.

The user moves the arm whose configuration $q_{device}$ in the workspace (the virtual environment) is sent to Virtools. Visualization of the process and moving around the scene is also done in Virtools using our virtual reality hardware: a man sized 3D screen, 3D glasses and markers attached to it. Last, a 6DoF robotic arm is used as a haptic device. The figure 8 shows the general architecture.

Fig. 8: Overview of the architecture.

B. Virtual Reality Environment

As our goal is an immersive experience, we have implemented virtual reality features to help the user to feel contact with the virtual environment. We believe that immersive motion planning will help the operator to have a better understanding of his environment and thus, perform tasks more naturally and more accurately.

First, our previous 6D mouse was replaced by a user-controlled haptic arm, model Virtuose 6D by Haption company, see figure 9. As before, the operator can move an object in six dimensions. The difference is that we now have a physicalisation process of the environment and object mesh.
models to a voxmap consisting of unbalanced voxels. This is done by using IPSI®, a library provided by the haptic arm constructor and used in our virtual reality software, Virtools®. Their collision algorithm detects how many voxels are in collision and generates a repulsive pseudo-force.

Collisions of the object with the environment are therefore felt by the operator through the haptic arm. Force feedback permits the user to feel obstacles. The normal to contact \( n_c \) is provided by IPSI® simulating contact. Friction forces are generated that permit to measure an angle between the normal \( n_c \) and the user force \( f_u \).

V. EXPERIMENTAL RESULTS

We have tested our H-RRT-C algorithm on various environments. These are the same we used on our previous work [15] [16] to be able to compare algorithm performance and results consistently.

A. Narrow tunnels

We first tested our algorithm on an environment that consists in a long tube inside which the object can barely move except in the direction of a very narrow passage, see figure 10. The goal position is displayed on the left of the figure. The tunnel is followed by two thin planes with a small opening. The first plane is rotated around one axis, the second plane is rotated around two axis. This makes a very cluttered place and a big challenge for an operator as the object needs to be rotated correctly and very accurately. The result is shown figure 11 for \( \alpha = 0.5 \).

The table I shows the results for various \( \alpha \) values. Extreme values considerably degrades the algorithm performances. More than two minutes are needed for a scenario without any computer help (\( \alpha = 0 \)) when the problem is solved in only 27 seconds with \( \alpha = 0.2 \). These results should be compared to our previous best performance of 61 seconds using I-RRT-C best parameters [16]. We solve the problem twice faster.

We can see here that a very low \( \alpha \) value is better. This is because the long tunnel before the two orientated planes is extremely difficult to reach for a computer. It is still needed though, to cross the planes so the performances degrades without any automatic help.

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>Time (s)</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>135</td>
<td>2 110</td>
<td>4 218</td>
</tr>
<tr>
<td>0.05</td>
<td>60</td>
<td>871</td>
<td>1 740</td>
</tr>
<tr>
<td>0.1</td>
<td>67</td>
<td>924</td>
<td>1 846</td>
</tr>
<tr>
<td>0.2</td>
<td>27</td>
<td>536</td>
<td>1 070</td>
</tr>
<tr>
<td>0.5</td>
<td>33</td>
<td>661</td>
<td>1 320</td>
</tr>
<tr>
<td>0.8</td>
<td>28</td>
<td>671</td>
<td>1 340</td>
</tr>
<tr>
<td>0.95</td>
<td>38</td>
<td>669</td>
<td>1 336</td>
</tr>
</tbody>
</table>

TABLE I: Influence of \( \alpha \) with H-RRT-C.

B. Crossing planes

The second environment is composed of two pairs of planes in which the free space is narrow. Figure 12 shows the result with a zero \( \alpha \) value. Figure 13 shows the nodes samples at the surface of one of the first two planes as the user pushed the object against this surface.

This environment is tested two algorithms, results in table II. These tests do not measure H-RRT-C but I-RRT-C, our previous method. In the first line we try to solve the problem with a standard RRT. This method takes more than two hours to find a solution. For the second line, alpha is set to zero. It means that no free space random nodes are shot but we still have contact nodes generated when in contact mode. The problem is solved in 25 seconds.

The H-RRT-C algorithm is then benchmarked with several tests using different \( \alpha \) values, see table III. Apart from very high values, we are almost always faster than the best I-RRT-C parametrization [15]. Optimal \( \alpha \) values should be around...
said that the time needed to solve the problem is 50% lower. This gives hints about the behavior of the algorithm regarding which parameter we want to minimize.

C. Discussion

Our algorithm is capable of solving motion planning queries using the capabilities of both a computer and a human operator.

From the results shown above, we can see that for both environments, we solve the problems 50-200% faster than our previous algorithm depending on the geometry of the problem.

The crossing planes experiment shows that the sole use of a haptic arm instead of a non-actuated 6D mouse speeds up the process. It also gives the possibility to an untrained operator to be efficient quicker when using a 6D mouse needs more training.

Our tests show that our new sampling method also contributes to speed our previous method. It is done by constraining contact sampling around the user’s movements and by generating them by taking into account its intentions.

Regarding the \( \alpha \) parameter, our previous work I-RRT-C was very dependent on the value of \( \alpha \) value. The time needed to solve the problem had big variations depending on its value and the environment. With our new work H-RRT-C, the performances of the algorithm is still environment-dependent but we observe that we can choose any \( \alpha \) apart from extreme values and the problems would still be solved twice quicker using our new method.

VI. CONCLUSION

In this paper, we have proposed a novel haptic path planning algorithm with contact, called H-RRT-C, allowing to explore both the free and contact spaces. When planning in contact, several significant improvements are brought. We can use a haptic arm providing force feedback to feel the environment. Force feedback allows to take into account the user defined motion in contact to efficiently guide the extension of the roadmap: samples are constrained inside a surface defined by the operator.

An fine analysis of the importance and influence in the overall process of each parameter is under work. We will also benchmark our methods with many users to quantify the advantage of H-RRT-C algorithm on a more representative population.

In our future works, we will address the issue of adapting the algorithm parameters such as the ellipse size and interactivity parameter \( \alpha \) in real time depending on the user’s behavior. Regarding contact we will develop a method that automatically switches contact planning on successive surfaces instead of having to choose them manually.

REFERENCES


<table>
<thead>
<tr>
<th>Scenario</th>
<th>( \alpha )</th>
<th>Time (s)</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRT</td>
<td>1</td>
<td>2h13m57s</td>
<td>6919</td>
<td>13 836</td>
</tr>
<tr>
<td>I-RRT-C</td>
<td>0</td>
<td>25</td>
<td>961</td>
<td>1 920</td>
</tr>
</tbody>
</table>

**TABLE III**: Influence of \( \alpha \), second scenario.

0.1 with an experiment duration as low as 10 seconds which is much faster than any other method.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>( \alpha )</th>
<th>Time (s)</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>H-RRT-C</td>
<td>0</td>
<td>15</td>
<td>848</td>
<td>966</td>
</tr>
<tr>
<td>H-RRT-C</td>
<td>0.005</td>
<td>18</td>
<td>778</td>
<td>1 548</td>
</tr>
<tr>
<td>H-RRT-C</td>
<td>0.01</td>
<td>10</td>
<td>669</td>
<td>1 336</td>
</tr>
<tr>
<td>H-RRT-C</td>
<td>0.1</td>
<td>12</td>
<td>862</td>
<td>1 722</td>
</tr>
<tr>
<td>H-RRT-C</td>
<td>0.3</td>
<td>20</td>
<td>926</td>
<td>1 850</td>
</tr>
<tr>
<td>H-RRT-C</td>
<td>0.6</td>
<td>21</td>
<td>954</td>
<td>1 897</td>
</tr>
<tr>
<td>H-RRT-C</td>
<td>0.9</td>
<td>39</td>
<td>1 262</td>
<td>2 522</td>
</tr>
</tbody>
</table>

**TABLE II**: Influence of algorithm, second scenario.