Human interaction with Motion Planning Algorithm

Michel Taïx, David Flavigné, Etienne Ferré

Human interaction with Motion Planning Algorithm

Michel Taïx
CNRS; LAAS ; 7, av. du Colonel Roche,
31077 Toulouse, France, Université de Toulouse;
UPS, INSA, INP, ISAE ; LAAS ;
F-31077 Toulouse, France
Email: taix@laas.fr

David Flavigné
ISIR
UPMC Univ Paris 06,
UMR 7222, ISIR, F-75005, Paris, France
Email: flavigne@isir.upmc.fr

Etienne Ferré
Kineo CAM
Rue de la dcouverte, 31677 Labege, France
Email: ef@kineocam.com

Abstract

This paper presents an interactive motion planning system to compute free collision motion in a numerical model. The system is based on interaction between a user and a motion planning algorithm. On one hand the user moves the object with an interactive device and on the other hand a motion planning algorithm searches a solution in the configuration space. The interaction aims at improving the guidance of an operator during a robot motion task in a virtual environment with the help of an automatic path planning algorithm. Existing works use a two-step decomposition which limits the interaction between the user and the ongoing process. We propose a modification of a classic motion planning method, the Rapidly-exploring Random Tree to build an Interactive-RRT. This method is based on exchanging pseudo-forces between the algorithm and the user, and on data gathering (labels) from the virtual scene. Examples are shown to illustrate the Interactive motion planning system with different interactive devices (space mouse and haptic arm). We analyze the influence of the user's dexterity to find a solution depending on various parameters of the algorithm and we show how we can adapt these parameters to a user.

1 Introduction

Manufacturing industries (automotive, aerospace, shipbuilding) take benefits from numerical simulations for several decades. The computer is used almost in all the stages of the product life cycle: design, manufacturing, maintenance. One of the most recent applications of the numerical simulation is the digital mockup for assembly path definition. Thanks to the numerical model of the product, design engineers are able to build the manufacturing processes at the early stages of the product definition. This simulation avoids the building of a physical mockup, which is costly, and time consuming. Moreover, the time needed to build a physical mockup makes the prototype always outdated as soon as it is available. Thanks to the numerical model, the engineers always have access to an up-to-date representation of the current product. Thus, by checking the assembly motion of the product while the design phase, the project team can minimize the risk induced by a non-feasible product. Indeed, an error in the feasibility of assembling a product leads to several months of delay. The engineers use various types of tools to define assembly paths. The goal of the study is the creation of a collision free motion of one part from a position outside the assembly to its mounted position. The assembly problem inside the digital mockup can be solved generally by two approaches : manually or by algorithms. The user can manually create such a motion thanks to highly efficient collision detector warning of a clash while he/she moves the target component. Another solution is the automatic path planner. These algorithms use the numerical model to automatically calculate the collision free trajectory.

When the assembly is done manually, the user can be immersed in a virtual reality environment. Virtual reality allows him to have a better world perception and can use a haptic arm in order to interact with CAD software [2]. Haptics are a recent enhancement that provides an additional perceptual modality in virtual environments [5] and can be used for learning . A force feedback device allows the user to test collision-free paths in assembly tasks during the process of industrial design or benchmark maintenance [13]. The contribution of haptics can improve the use of virtual reality [38, 26, 6]. The use of a space mouse for interaction, is less expensive than a haptic device and can give very good results in many design processes with digital mockup [8, 34]. Manufactured goods using such technology are already available in the market and are used in car and aeronautical industries. The main applications concern the operations of disassembly for maintenance checks as well as during final phase assembly [25].

But in complex scenes the user may get lost and need help in finding the solution easily. It is very easy to move a ball in a 3D environment with a haptic device because in that case the orientation of the object has no importance. But this is not the case with a real non-convex object in a complex scene.

Elsewhere, in robotics, a lot of work in motion planning [20] has been done to compute free paths in digital models for mechanical systems. With the recent results in random planning algorithm [23] it is possible to solve automatically problems for systems with many degrees of freedom. The algorithm computes a collision-free roadmap in a configuration space (*CS*) where the object is reduced to a point. This point represents the robot's model in the environment and the dimension of *CS* is equal to the number of degrees of freedom of the mechanical system. The algorithm searches a free path for a point in the roadmap. One example of applying motion planning techniques

in real studies is in the maintenance and operation of nuclear power stations [32, 31]. Now, companies use automatic path planning to solve PLM applications [10, 21].

There are mainly two families of methods for building a roadmap, the Probabilistic Roadmap (PRM) [18] and the Rapidly-exploring Random Tree (RRT) [24]. The RRT approach is more interesting to our study because it is faster in the single query case, when the roadmap computation must be limited in time. For really constrained environments [9], there are very effective methods to solve problem. But this is not always the case, especially when the object must pass through a narrow passage. It is difficult for the algorithm to find the passage entrance (see for example [4, 37, 1, 35, 33, 27, 11, 29]). To do so it will have to investigate randomly all the *CS* dimensions. On the other hand when the algorithm finds the narrow passage entrance in the *CS*, it will quickly progress to find a solution. Random algorithms progress easily in narrow passages but have no global vision of the solution.

On the contrary, in the case of a free or low-constrained environment, a user will easily find a solution (for example, we quickly steer the key towards the keyhole). On the other hand, a user will have much more difficulties to progress in narrow passages. Thus, the user has a good global vision of the problem, but finds it difficult to make fine movements.

It is noticeable that the algorithm and a user with an interaction device, can both work in a complementary fashion. The complementarity between user and algorithm is the essential idea of this work. The main contribution of the method is to allow simultaneous cooperation between the loop of roadmap search and the loop of interaction for the user. As the roadmap tree iteratively enlarges, it uses the directions of motion of the user device to favor the directions of search extensions.

In this paper, we choose to integrate the human into the planning loop via an interaction device, which can be a space mouse or a haptic arm. The user can thus act on the search, showing areas to explore or to go through. On the other side, the algorithm can gather information while exploring the scene (independently from the user's input) and display them visually and/or haptically. The purpose is not to provide an optimal solution ([15, 17]) but to simultaneously accelerate the search of a solution, and to improve the operator guidance during a motion task in a virtual environment with the help of an automatic path planning algorithm. These results are an extension of [12].

This algorithm can also propose other possible solutions to the user. For example, during the insertion of window glass in a car's door, it may be interesting to test the various solutions to insert the window. For this purpose, human and computer must cooperate in an efficient way along the process.

The main asset of our method is the deep integration of Human into the planning algorithm. This allows a narrow collaboration between human and algorithm (where other human interaction methods have a two-step decomposition that separates the human and the algorithm, our method proposes to make them work in the same loop). This collaboration facilitates better consideration of human constraints that cannot be easily converted to mathematical or logical expressions (preferred solutions because of easier manipulation for the human, or because of flexible obstacles that are not modeled for example). Also, a better perception of the search and solution by the operator : it helps him to decide if the computed solution is applicable (regarding the constraints) in practice and to learn the solution for in real life application.

The paper is organized as follow. We begin by describing previous work in interactive motion planning. The following section describes the basic RRT algorithm and extensions that have been made to answer our problem. The interactive motion planning algorithm, I-RRT, and the main steps are presented in section 4. Simulation results with an industrial application are presented in section 5. Finally, we analyse our results and concludes the paper.

2 Related works

We can find works where the main problem was to move a robot arm in a virtual environment for off-line programming industrial robot by means of interactive path planning method. In [14] the user selects critical robot configurations with a haptic interface to facilitate the automatic path planning research with PRM approach. The advantages of human intuition are exploited to facilitate the robot path planning process. A virtual tele-operation system provides a convenient tool for manipulating a virtual robot arm. They defined a semi-automatic path planning without a real feedback loop between user's motion and motion planning method. [7] used potential field method to guide user's control on the virtual robot through a haptic device. This approach provides little information to the user because of the many local minima induced by the force fields in complex environment. In these two works the motion planning issue is not the important point of the approach.

On the other hand, idea of taking into account user input has been studied by various authors specialized in motion planning. In [3] the authors use user input to work with an automatic motion planner. They show that randomized techniques are quite useful to transform an approximate user input path with collisions into a collision-free path. The idea is to push an approximate user path to free space in CS. The use of a path planning technique based on harmonic functions to generate guiding forces that aid a user in a virtual environment is introduced in [36], [30]. The idea is to compute a solution channel by cell decomposition that connects the initial and the final configuration and then two harmonic functions are computed over the CS to find a guiding path. RRT algorithms including heuristics based on a study of the workspace are presented in [19], [28]. The idea is to discretize the workspace using an unbalanced octree decomposition and generate a free continuous volume between initial and final configuration using A^*. In a second time a free path for the object is computed by using RRT approach.

In all these studies, it is necessary to note that the interaction between the automatic search by motion planning algorithm and the user is simplified by a decomposition in two stages. The stage that includes the user can also include new kinds of peripheral devices that allow users to interact in a more natural way with machines. These various devices such as space mice or haptic arms (figure 1) can be used to improve motion planning algorithm with a natural user input, thus speeding-up path computation. A space mouse is a kind of mouse that can move in three dimensions, and even with orientation, the drawback is that the user has no feedback from the machine. This is done by haptic arms, which can send back forces and torques to the user. The Virtuose 6D Desktop includes three articulated branches whose end is fixed on the sphere of seizure. This haptic interface has a kinematics with 6 degrees of freedom (dof), and the

return of effort is exerted on the 6 degrees.

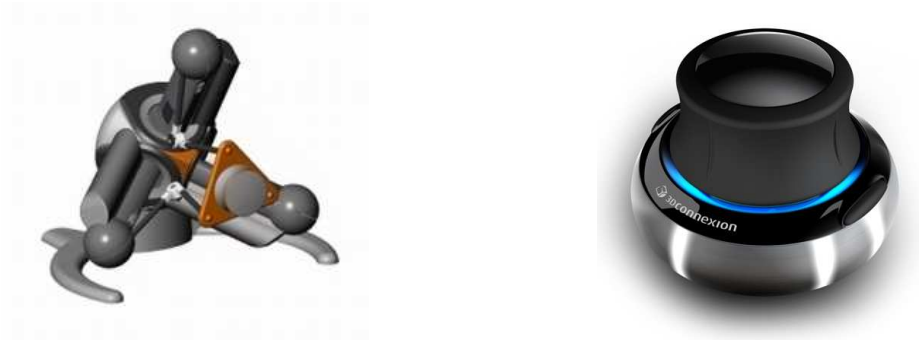


Figure 1: The Virtuose6d and the space mouse.

3 RRT Path Planning

We present the principles of the RRT that we modify thereafter to take into account the interaction with the user. The Rapidly-exploring Random Tree (RRT) was introduced in [22] as an efficient sampling method to quickly search in high dimensional spaces for single query motion planning.

The RRT algorithm is shown in Algorithm 1 and the development is illustrated in Fig. 2. Starting at a given initial configuration, RRTs incrementally search the configuration space for a path connecting the initial and the goal configuration. At each iteration a new configuration, q_{rand}, is sampled ($RANDOM\_CONFIG$) and the extension from the nearest node, q_{near}, in the tree ($NEAREST\_NODE$) toward this sample is attempted. If the extension succeeds ($CONNECT$) a new node, q_{new}, and an edge are created in the roadmap.

Algorithm 1 The RRT-CONNECT algorithm

$T(q_{init})$
for $i = 0$ to N **do**
  $q_{rand} \leftarrow$ RANDOM_CONFIG()
  $q_{near} \leftarrow$ NEAREST_NODE(q_{rand}, T)
  if CONNECT(T, q_{rand},q_{near},q_{new}) **then**
    Add_Vertex(T, q_{new})
    Add_Edge(T, q_{near},q_{new})
  end if
end for

By definition the RRT tends to grow rapidly in the unexplored regions of the CS and the probability to find a path (if it exist) approaches one as the number of RRT

nodes tends to infinity. Many different planners exploit the exploration properties of the basic RRT and try to increase the rapidly exploring property.

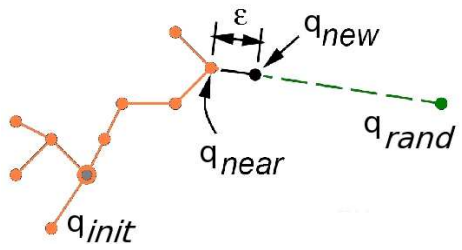


Figure 2: RRT Algorithm.

RRT planner performs well in many applications but gives unsatisfactory results when narrow passages are far from both initial and goal configurations. To improve performances it is necessary to guide exploration by leveraging available information. In our case it is necessary to take into account the interaction with the user.

The main modifications of the basic RRT are to be made on the following stages:

- By definition, the state of the node is binary. If the node is free and the connection path is also collision free, the node/edge is created in the roadmap, otherwise, any information is memorized. It is interesting to memorise information for the user when the CONNECT function return false.
- The sampling area in RANDOM_CONFIG is a crucial point to extend efficiently the roadmap. User's interaction must be taken into account in the definition of the sampling area.
- The NEAREST_NODE function has not only to take into account the Euclidean distance but also the most interesting nodes with regard to the task of the user.
- At each iteration, the new peripheral device position must be taken into account to compute these different steps

To take into account these various points we propose a new algorithm called Interactive Rapidly-exploring Random Tree, I-RRT, which is presented in the next section.

4 Interactive Motion Planning System

In this section, the first part gives an overview of the method and the second part details each step of the I-RRT algorithm.

We note that interactive device can be a space mouse or a haptic device. If the interactive device is a haptic device the method uses force through the haptic feedback, in the case of a space mouse, the motion of the mouse is transformed to be used.

4.1 Approach Overview

The idea behind the I-RRT algorithm is to take advantage of both a human operator and the RRT algorithm capacities to solve a motion planning problem. Motion planning methods that include a human in the loop are focusing on two steps approaches that separate user input from algorithm execution. Our main contribution is to gather these two steps in a single general loop, shown on Fig. 3.

The goal is to improve the user guidance using motion planning algorithms while these algorithms will benefit from the user's experience. One of the key problem in motion planning is the narrow passage problem. It illustrates the purpose of our method. Traditionnal sampling methods have difficulties to find and go through narrow passages in the configuration space, but a human can easily find the entrance of narrow passages and guide the search. For example, steering a key in a keyhole might be a difficult problem for an automatic algorithm but is easy for a human. This was one of the motivation of integrating a human into the planning loop. The operator will deal with narrow passages issues while the algorithm can help in cases that are difficult for him.

The principle is to let the user move an object (the avatar) in the virtual scene to guide the RRT exploration by introducing a bias toward the avatar during the planning phase. The operator uses an interactive device to move the virtual object. Such algorithm has to take two constraints into account :

- The operator movement has to lead the roadmap extension.
- The planning algorithm has to provide useful information to the user concerning environment, obstacles and exploration status.

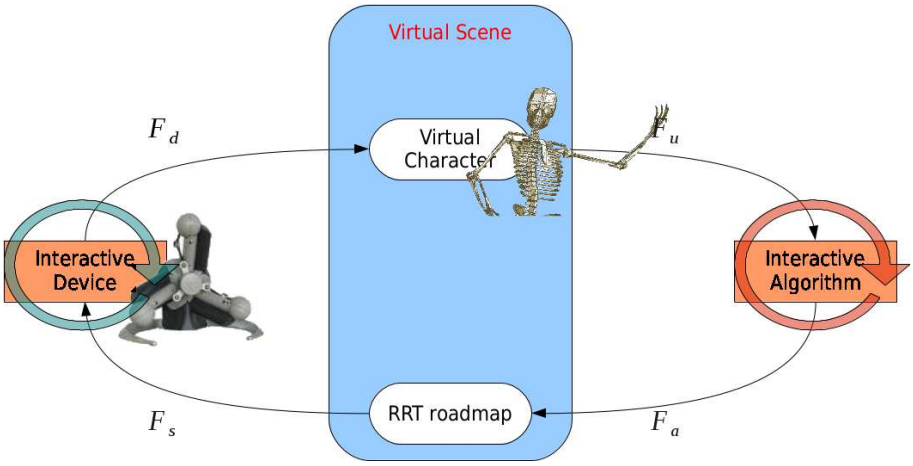


Figure 3: General interaction loop between the algorithm and the user (represented by the interactive device) through the virtual scene.

The proposed planning method is based on a RRT algorithm and can be divided into two distinct parts, represented on Fig. 3 by two loops running in parallel :

1. In the first loop ("interactive algorithm"), the planner explores the configuration space in an automatic way, looking for a collision-free path between start and goal configurations. The sampling and the extension steps are different from the original RRT algorithm, in order to take the user input into account. The user interaction is represented by the pseudo-force F_u on Fig. 3. We use pseudo-force instead of force because data are not a force in the physical sense, it is a vector proportional to a force. In this article, we use the same name to represent both a real force and this kind of vector. Using this pseudo-force, the planner extends its roadmap and gathers data on the virtual scene. This data is sent back to the user through another pseudo-force called F_a. This pseudo-force is also used in the next iteration of the general loop, along with F_u, to extend the roadmap.

2. In the second loop ("interactive device"), the operator moves the avatar in the virtual scene. A pseudo-force F_s is applied to the interactive device if it is possible (haptic device) or directly on the avatar otherwise. This pseudo-force may be interpreted as a disturbance by the operator but is intended to act as a guide towards a better area to explore in the configuration space. The device interaction with the virtual scene is interpreted as a pseudo-force F_d, which can include F_s in the case of a non-haptic device.

These two loops are interacting through the virtual scene. First, the user uses the interactive device to move the avatar (F_d). The avatar movement is used to generate F_u (which includes F_s either directly or applied to the interactive device if possible). Then, the algorithm builds a RRT roadmap using data gathered during previous iterations (F_a) and F_u. This roadmap is used to gather data for next iterations and give feedback to the user (F_s).

The different nature of the two parts of the algorithm allows us to distinguish three different modes of working for the I-RRT algorithm :

- The user is moving the interactive device faster than the algorithm builds its roadmap because he knows where to go to get closer to the goal configuration. In this case, the algorithms tracks the avatar position in the virtual scene. The user's intention is considered predominant. If the solution is obvious to the user, the algorithm follows his movements.

- The user does not move the device. The algorithm keeps exploring the configuration space to gather data on the environment and to give visual and/or (depending on the device) haptic hints to the user. This happens if the user is stuck and does not know how to move the avatar. This mode is close to the original RRT algorithm.

- The user slowly moves the device. In this case, the algorithm extends the roadmap around the user's position while still exploring other part of the configuration space, without the user's influence, to gather more data on the environment.

These three different modes of working are not three separate states of the algorithm, but rather two extremities and the middle of a behaviour line (Fig. 4).

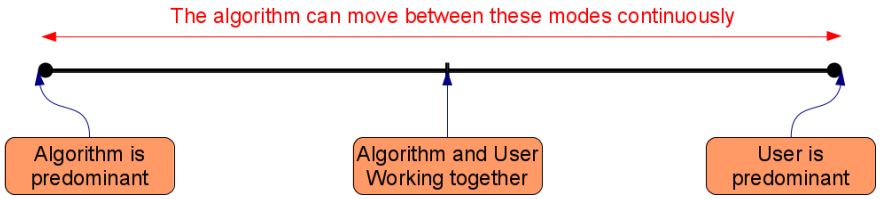


Figure 4: Behaviour line of the algorithm. The algorithm state can be anywhere between fully automatic behaviour (left) and fully manual behaviour (right).

Each loop is running at its own speed. The interactive device loop is faster than the algorithm loop (which can vary due to the increasing number of nodes, thus increasing the number of required collision tests). This is not a problem since they are non-blocking. Only pseudo-forces are shared between each loop. In the next section we detail each step of the algorithm that is different from the original RRT.

4.2 Description

This section is divided into two main parts, each of them describing one of the two loops involved in the interactive motion planning system.

4.2.1 The interactive device loop

This loop is simple as its only purpose is to deal with the interactive device data. It computes necessary pseudo-forces for feedback from and to the user.

Algorithm 2 Interactive device algorithm

```
loop
  F_s ← ALGORITHM_FEEDBACK()
  if USING_HAPTIC() then
    APPLY_FEEDBACK(F_s)
    Q_device ← RETRIEVE_DEVICE_STATE()
    F_d ← COMPUTE_DEVICE_PSEUDO_FORCE(Q_device)
  else
    Q_device ← RETRIEVE_DEVICE_CONFIG()
    F_d ← COMPUTE_DEVICE_PSEUDO_FORCE(Q_device,F_s)
  end if
end loop
```

The algorithm 2 shows the principle of the interactive device loop. First, it starts by retrieving information from the algorithm to guide the user in the virtual scene (ALGORITHM_FEEDBACK). This information is represented by the pseudo-force F_s, which points to areas to explore according to the algorithm. Then, it handles two different cases:

1. The interactive device is a haptic device. F_s is sent to the device to provide the user with a feedback from the algorithm (APPLY_FEEDBACK). The state of the device resulting from this feedback and the user's action is retrieved (RETRIEVE_DEVICE_STATE) and used to compute a pseudo-force F_d (COMPUTE_DEVICE_PSEUDO_FORCE) that is used for moving the avatar in the virtual scene, movement which is transformed into F_u.

2. The interactive device does not have haptic features. In this case the state of the device Q_{device} is mixed with F_s to compute F_d (COMPUTE_DEVICE_PSEUDO_FORCE).

4.2.2 The interactive algorithm loop

The algorithm loop is based on the RRT motion planning algorithm. It is shown on algorithm 3. The main steps of interactive RRT (I-RRT) are :

- Configuration sampling using computed pseudo-forces (F_u, F_a). We are looking for an efficient way of combining user movements with the algorithm automatic search.
- Choosing the nearest neighbour in the roadmap.
- Extending the roadmap.
- Updating node data given the result of the roadmap extension.
- Computing a pseudo-force F_a aimed at attracting the user to relevant areas to explore. It is computed using data contained in the roadmap nodes.
- Computing a pseudo-force F_u from the motion of the avatar in the virtual scene. This motion is generated by the user interaction with the device (F_d) and the virtual scene (obstacles may stop the movement).
- Sending back some useful information to the user by using haptic feedback or visual hints.

Algorithm 3 I-RRT algorithm

$T \leftarrow q_{init}$,q_{goal}
while !PATH(q_{init},q_{goal}) **do**
  $q_{rand} \leftarrow$ SAMPLING(F_a, F_u)
  $q_{near} \leftarrow$ NEAREST_NODE(q_{rand}, T)
  if CONNECT(T, q_{rand}, q_{near}, q_{new}) **then**
    $T \leftarrow q_{new}$,EDGE(q_{near}, q_{new})
  end if
  GATHER_DATA(T, q_{near}, q_{new})
  F_a,$F_u \leftarrow$ COMPUTE_PSEUDO_FORCES(T, q_{user})
end while

Algorithm 4 COMPUTE_PSEUDO_FORCES(T, q_{user})

$L_{near} \leftarrow$ NEAREST_NEIGHBORS(q_{user}, T)
$B \leftarrow$ Compute_Barycenter(L_{near})
$F_a \leftarrow$ Compute_FAlgo(B, q_{user});
$F_u \leftarrow$ Compute_FUser(q_{user});
$F_r \leftarrow$ Compute_FResult(F_a,F_u);

In a first time, the pseudo-forces computation is described because they are the basis of interaction in the method and are used in other parts of the algorithm. Then, the parts of the original RRT algorithm that were modified for our algorithm is explained. Finally, the data gathering and storage and the haptic feedback are described.

Attractive pseudo-force computation

Interaction being the basis of the I-RRT, the contribution of each interacting part (the user and the algorithm) has to be defined. These contributions have to be flexible and have to reflect the planning constraints (obstacles, goal, ...). This means that they vary continuously during the planning, according to the virtual scene. This is achieved through computation of pseudo-forces which are aimed at representing each interacting part influence over the path searching process.

The contribution of the algorithm part is defined by the computation of the attractive pseudo-force F_a (Compute_FAlgo in algorithm 4). This computation is illustrated on Fig. 5. First, the p nearest neighbours of the user position q_{user} are retrieved from the RRT roadmap (NEAREST_NEIGHBORS) and Fig. 5-a). The barycenter B of these nodes is then computed (Compute_Barycenter). The data contained in the nodes are used as weights during this computation (Fig. 5-b and 5-c). The computation of weights is detailed below in the paragraph "Node Labelling". F_a is the vector starting from q_{user} and directed towards the barycenter (Fig. 5-d).

User pseudo-force computation

The other interacting part of the system is the user. Its contribution is defined by the pseudo-force F_u (Compute_FUser). F_u results from the avatar movements in the virtual scene and its interaction with obstacles. As the avatar movements are controlled by the user and the pseudo-force F_d, F_u can be deduced from F_d most of the time. However, this is different if the avatar is not allowed to collide with obstacles during the path search, when there is a contact with an obstacle. The user will apply a force on the device ($F_d \neq 0$), but the avatar will not move if it implies a collision with the obstacle ($F_u = 0$).

Resulting pseudo-force computation

As the path search has to reflect both part of the interaction, the two previously computed pseudo-forces are mixed into a resulting pseudo-force called F_r (Equ. 1 and Fig. 6-a). F_r takes into account the user's intention and the algorithm suggestions for the path search. We can retrieve the three modes of working illustrated on Fig. 4 by studying the variations of F_r :

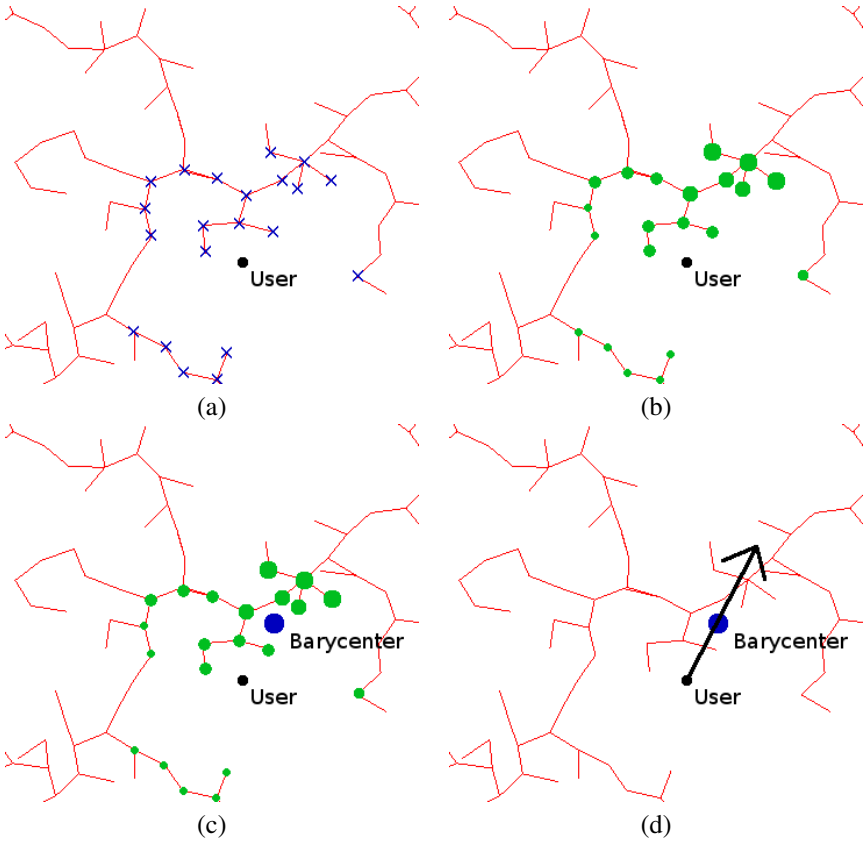


Figure 5: Computing F_a pseudo-force.

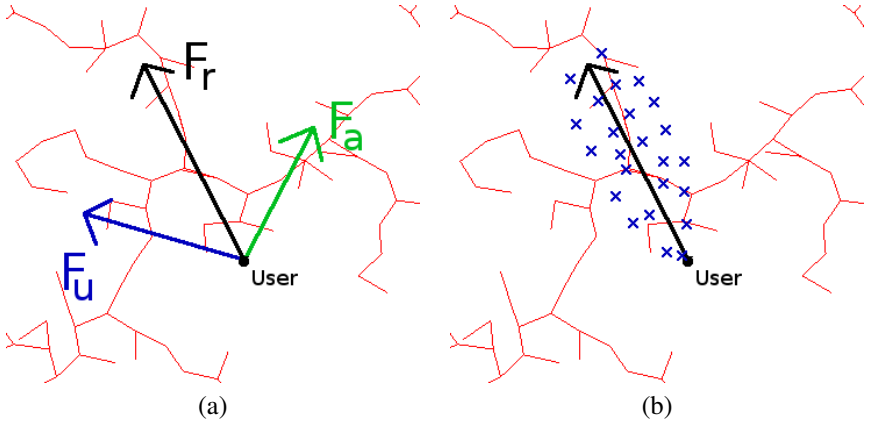
- The user moves fast : $F_u > F_a$. In this case, F_a is negligible compared to F_u and then F_r is almost equivalent to F_u. The user is predominant and leads the path search.
- The user doesn't move : $F_u = 0$ and $F_a > 0$. In this case, $F_r = F_a$. The algorithm is predominant.
- The user moves slowly : The user and the algorithm are working together.

$$F_r = \alpha \cdot F_u + (1 - \alpha) \cdot F_a \tag{1}$$

In equ. 1, α can be used to modify manually the influence of each part. If $\alpha = 0$, the algorithm is equivalent to an RRT algorithm. If $\alpha = 1$, the algorithm is only tracking the user position.

Sampling method

The resulting pseudo-force F_r is computed in order to guide the sampling of the RRT algorithm. The sampling is the heart of probabilistic methods. If the sampling can be controlled, the path planning is controlled as well.



(a) (b)

Figure 6: Sampling a new configuration using user and algorithm interaction.

The F_r pseudo-force computed in the previous step is used by the sampling method to guide the path search (Fig. 6). The sampling method is a gaussian sampling along F_r direction. If the user does not move or move slowly, the sampling is equivalent to a gaussian sampling around the user position but if he is moving fast the sampling anticipates the user's movements. This sampling method can be seen as a position tracking method. The standard deviation σ of the gaussian sampling is used to tune the influence of the algorithm on the search.

But this sampling method has some drawbacks. If the user moves too fast, the RRT algorithm may not be able to link newly sampled nodes to the existing roadmap due to obstacles. To deal with this problem, three other sampling methods were investigated.

- The first method allows creation of several local trees rooted at nodes that cannot be linked directly to the main tree. It is useful in the case of I-RRT method to keep track of user's movements even if the connection with the main tree was lost (this is the drawback of the original sampling method). The missing connection is made by the algorithm part of the interactive motion planner.

- The second method stores the path traveled by the user and samples around this path (gaussian sampling). The advantage is that the connection to the main tree is never lost, but it slows down the resolution. In this method, the influence of the algorithm can also be tuned using σ.

- The third method adds a uniform sampling in the configuration space. This method allows a global exploration of the virtual scene, independently of the user's position, and then the data gathering is also more global compared to the original sampling method. To preserve the interactivity, the uniform sampling should be used with one of the other methods. A parameter β indicates the ratio between the usage of the two used methods.

All sampling-based algorithm are based on random sampling which is dense with probability one to ensure probabilistic completeness. In the worst case, without human intervention, the algorithm will proceed as an RRT and inherits the property of their completeness. With only a sampling around F_r it is not possible to guarantee this property. It is necessary to add a uniform sampling to the main search direction to ensure probabilistic completeness in case the user does not move the interface. This is why the uniform sampling method is always used in combination with one of the other methods described above.

Nearest neighbour

The choice of the nearest neighbour q_{near} for extension is made using the distance d (Equ. 2) between the newly sampled node and each node of the tree (NEAREST_NODE). This distance is also used for the choice of the nearest neighbours of the user's position (NEAREST_NEIGHBORS) during barycenter computation.

$$d = \sqrt{\sum_{i=1}^{n} \omega_i \cdot (dof_{i_{sample}} - dof_{i_{node}})^2} \tag{2}$$

Extend method

The extension step of the I-RRT is the same as a RRT-Connect algorithm : the nearest neighbour in the tree is extended towards the new sample until it reaches this sample or collide with an obstacle. Given the result of this extension, the extend method can return three different states :

- Collision : the extension failed because q_{near} was near an obstacle. No new node is added to the tree.

- Obstacle : the extension was partially done because an obstacle was between q_{rand} and q_{near}. A new node q_{new} is added to the tree.

- Success : q_{near} and q_{rand} were successfully connected. q_{rand} is added to the tree ($q_{new} = q_{rand}$).

For the original RRT algorithm, this is the last step of an iteration. For the I-RRT algorithm, we need to define the feedback for the user. This feedback is computed as hints that can be either haptic hints or visual hints (or both). With this feedback, the interaction loop is closed. The steps needed for this are detailed in the following paragraphs.

Node Labelling

The first type of feedback is visual. The algorithm uses data stored in nodes to display visual hints for the user. These visual hints are modeled as a coloring of nodes and edges according to labels computed from return states of the extend step. These labels are the data stored in the nodes. In the original RRT algorithm, configuration are sampled randomly and the only piece of information returned is their collision state with the scene obstacles. In the I-RRT, these collision tests are used to label roadmap nodes. Two types of labels were defined for visual feedback :

- The distance label is a function to guide towards the goal. It is the distance from the current node to the goal node.
- The collision label is a function of the covered distance during the extension, the number of extension of the current node and the number of collisions resulting from extension attempts. It is computed using the return states of the extend method. It reflects the constraint of workspace to extend this node.

However, as the roadmap grows the screen can be overloaded and the user will not benefit from this feedback. In practice, the graph display can be switched off to ease the search and overcome this drawback.

Haptic feedback computation

The second type of feedback is the haptic feedback. A pseudo-force is inserted in the interaction loop to give a more intuitive guidance for the user and to avoid cluttering the screen with visual artifacts. To compute this pseudo-force F_s, the previously computed attractive force is used :

$$F_s = k \times F_a \tag{3}$$

In equ. 3, k is a scale factor set by the user before algorithm execution.

In the case of the operator using a haptic device, F_s is sent directly to the device. The user constantly feels an effort from the device which guides him towards areas to explore, according to the algorithm. In other cases, the haptic feedback is applied to the avatar movements in the virtual scene. The user does not feel the feedback directly but will be able to see the perturbations due to F_s on the movements of the avatar.

F_s also allows the user to stay within range of the algorithm roadmap and thus limit the problems mentioned in paragraph *Sampling method*.

5 Results

5.1 System implementation

The algorithm was developed in C++ using the HPP (Humanoid Path Planner) platform developed at LAAS and based on the KineoWorks software[1]. Experiments were made on a dual-core PC, 2,1GHz and 2GB RAM.

In practice, using a local trees sampling method mixed with a uniform sampling method minimizes the user's influence on the path search. The main idea behind the I-RRT method is to make user and algorithm interact equally, for this reason these two methods will not be used together in most of the cases. In the following experiments, the uniform sampling method will be used with the sampling method centered on the user's position (along F_r) and with the sampling method along the user's path. In each case, $\beta = 3$. The pseudo-forces F_u and F_a have the same influence on the search ($\alpha = \frac{1}{2}$). The nearest neighbours search is made by comparing each node of the tree. The number of nodes p used to compute F_a is five. For this computation, all the neighbours should remain in the same connected component.

The required time to solve the path planning is dependent on the user's ability to manipulate the interactive device. If he does not move or is stuck in a dead-end, the planning time can increase and the algorithm will not benefit from the user's guidance. In the following case, we decided to let the user go through obstacles (contact forces are not considered) so that user's movements will not be limited, and he will still be able to guide the algorithm. The F_s pseudo-force replaces the contact forces to inform the user of obstacles limiting the exploration capacity of the algorithm in the current area. With this, even if the user path is in collision with the environment, the algorithm can still find a collision-free solution.

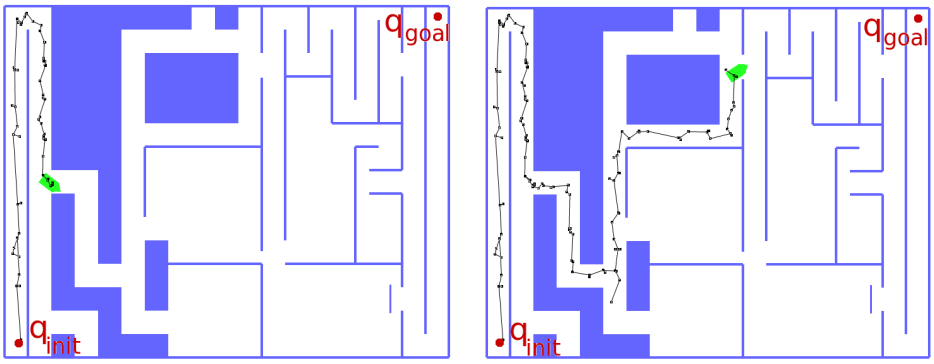
The RRT generated path is usually a broken line. In the context of a physical application, this path is not feasible. To make it physically feasible, this solution has to be deformed to a smooth path. One of the most common solution is to use the final path nodes to build a bezier curve. The drawback is that the deformed path might not be collision free, and making it so is not trivial. Here it is not a problem because the movement of the object is achieved by the user who performs an implicit path smoothing.

5.2 Simple 2D case

The main goal of this first case is to illustrate the principle of the interactive algorithm, and the different modes that can be observed during a planning query. In this case, the user moves the avatar with a space mouse in the virtual scene shown in Fig. 7, 8 and 9. The coloring of nodes was not activated for this example.

The user has to reach q_{goal} from q_{init}. The avatar (the green block) has three degrees-of-freedom : x, y, θ. The first part of the environment is simple and user and algorithm can both easily find the path. The user being faster, he leads the search during this part. Then, the environment becomes tighter and there is a crossroad (Fig 7-a). The way is obvious for the user, but in a non-interactive case the algorithm would

[1] KineoWorks is the path planning dedicated Software Development Kit developed by KineoCAM

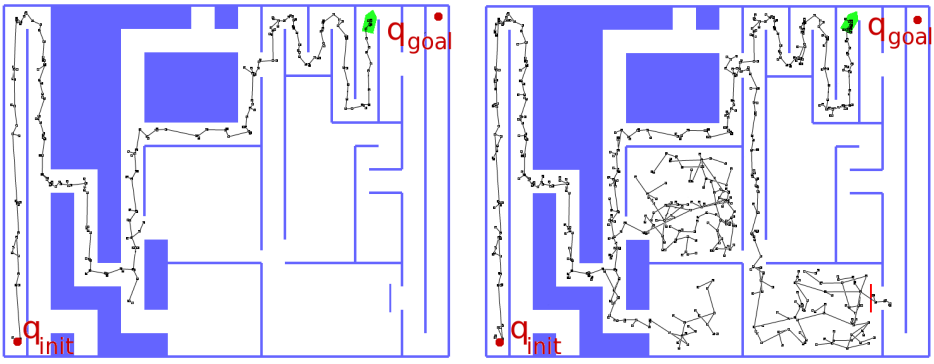


(a) Few nodes are created (b) The user has a dominant influence on the search

Figure 7: The user is leading the search and the algorithm is tracking his position.

have difficulties to find the narrow passage. The user is predominant and leads the search into the narrow passage.

After the narrow passage, the environment has two rooms with no exit. These rooms would have required a large amount of time to be explored in a non-interactive case. Instead, the user avoids them and leads the search in the right direction (Fig. 7-b). The algorithm is not predominant but can initiate a slow exploration towards regions left out by the user (small branches are developed towards the rooms).

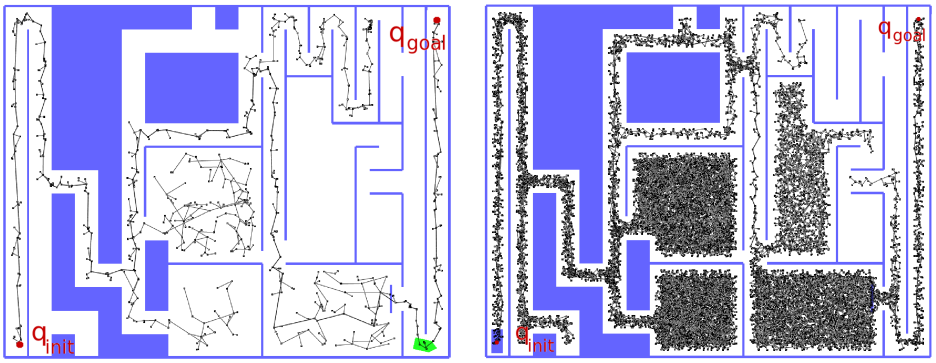


(a) The user leads the search to a dead-end (b) The algorithm explores other possibilities

Figure 8: The user is stuck in a dead-end, the algorithm is searching other possibilities to reach the goal.

On Fig. 8-a, the exploration made by the algorithm part is more visible. Meanwhile, the user saw a possible passage and tries it. The passage is too narrow for the user to find a solution here. While the user is trying to develop nodes in this passage, the algorithm continues exploration in other parts of the environment (uniform sampling), as shown on Fig. 8-b. The algorithm finds a solution left out by the user because it seemed to be too narrow for the robot (red obstacle on Fig. 8-b). The user mistake is

due to his point of view of the virtual scene. The red obstacle is placed below the plan in which the robot can move. This trap may seem artificial but it illustrates perfectly problems that can be met in complex 3D environments.



(a) The I-RRT roadmap after the solution was found (b) A Bi-RRT roadmap

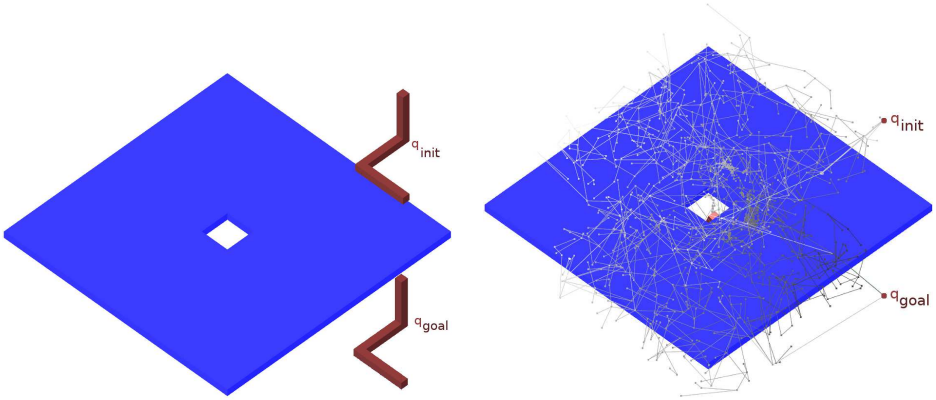
Figure 9: Comparison between the roadmap obtained after using the I-RRT algorithm and after using a Bi-directional RRT algorithm.

On Fig. 9, we can compare the roadmap built by the I-RRT algorithm with the roadmap of a bi-directional RRT algorithm. The interaction between the user and the algorithm allows a lower number of nodes and a faster exploration (few minutes) than a RRT algorithm. The RRT algorithm made a dense exploration and covered the maximum of the space. This kind of exploration requires a lot of time and resources (> 1 hour).

5.3 Simple 6D case

In this example, shown on Fig. 10-a, the user has a haptic arm (Virtuose 6D Desktop - Haption) to manipulate a S-shaped object in order to cross a wall through a hole in its center. The wall dimensions are $700 \times 700 \times 20$ and the hole's are 80×80. The object is composed of three sticks assembled along three edges of a virtual cube. The nodes of the roadmap represent the position of the center of this cube (and thus are not a point on the object itself). Each stick is $20 \times 20 \times 100$. The environment has only one difficult passage which requires the help of the user. The sampling method used are user tracking (using F_r) and local trees are allowed. The node coloring is activated. The configuration space is bounded to force the user to go through the hole, and the object has six degrees-of-freedom.

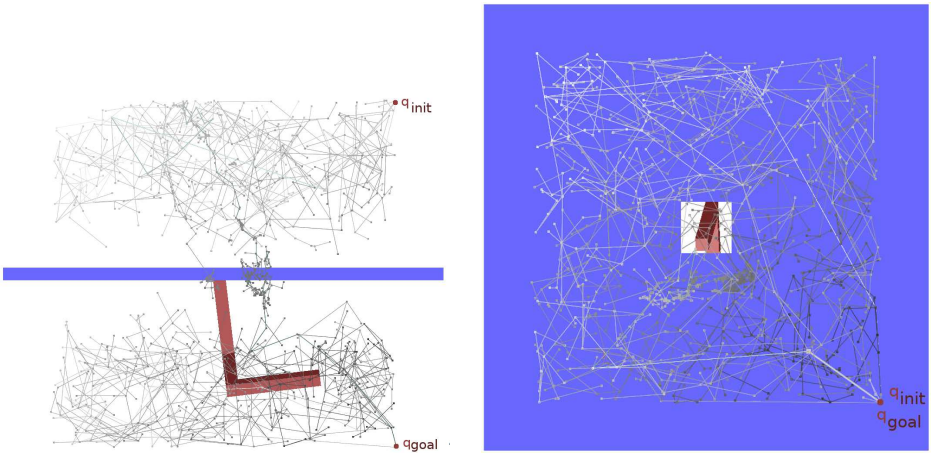
Solving this planning query requires that the user has a good ability to manipulate the interactive device, the passage being really narrow. The user has to slowly guide the search through the hole in order to improve the search. On Fig. 10-b and Fig. 11, the developed roadmap is displayed with nodes colored according to the labelling. Nodes that are closer to the goal are displayed in a darker color than others. The algorithm part covered both part of the wall (due to the local trees) so that the user has only to guide the part in the narrow passage. The roadmap density can testify to the dexterity



(a) (b)

Figure 10: A case with a 6-dimensional configuration space.

of the user. Here, the roadmap is not dense nor sparse, which shows that the user has an average dexterity.



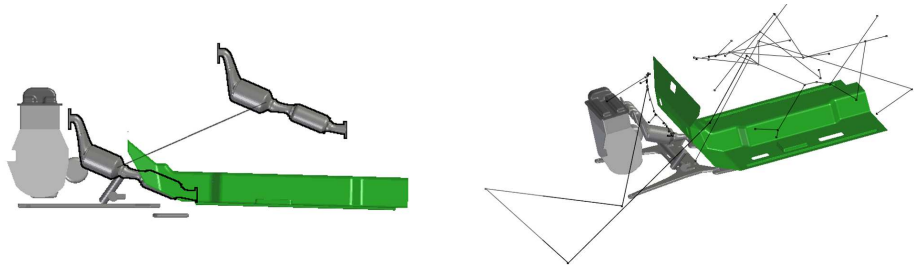
(a) (b)

Figure 11: The roadmap is developed on both sides of the wall.

5.4 Industrial case

The following example is based on a real automotive case. The goal is to find a disassembly path for a car silencer. In this example, the configuration space is bounded. The user is using the same haptic arm as in previous example (Fig. 13) to move the silencer and extract it from its location. The extension of a tree from the goal configu-

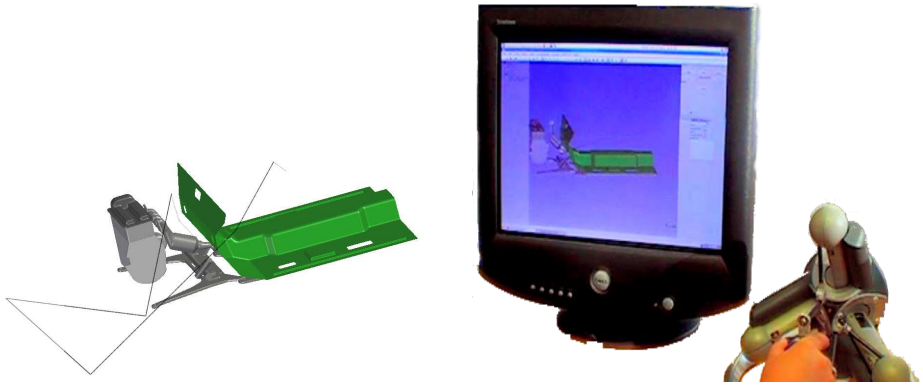
ration is allowed, because it does not modify the interaction user-algorithm : the goal configuration is in a large free space, easily accessible after the silencer is removed.



(a) The planning query (b) The I-RRT roadmap

Figure 12: The silencer has to be remove from bottom to be placed on the top.

The query is solved in few minutes by the user (Fig. 12). The main difficulty is to extract the silencer from the start area, under the green part. This disassembled query can be solved more quickly by specific automatic algorithms [9]. This problem does not favor interaction between user and algorithm but, in the context of guiding the user, the I-RRT method is still valid because the interactivity allows to control the solution and to take into account constraints imposed by the user. On Fig. 12-b, we can see that the roadmap around the goal configuration is developed mainly by the algorithm part and the roadmap around the start configuration by the user part.



(a) The solution path (b) Manipulating the haptic device

Figure 13: Extracting the silencer using an interactive device and the I-RRT algorithm.

6 Analysis

Due to its dual nature, the interactive RRT performances are strongly dependent on several parameters. The first parameters, on the user side, are the user capacities to

visualize the virtual scene and the potential solutions and his skills at manipulating the interactive device. Then, on the algorithm side, the choice of the sampling method to use is determinant. Last, the parameters needed for exchanges between each part of the I-RRT are also to take into account (k for the haptic feedback and α for computing F_r). The standard deviation used for gaussian sampling methods is also important for exchanges.

The settings of the algorithm parameters will influence its performances and the aspect of the resulting path. They can be used to adapt the algorithm to the user skills.

In this section, we choose the three parameters that seemed to be the most representative of each type of parameters. We analyze the influence of these parameters on the algorithm:

- The user dexterity for the user part.
- The sampling method for the algorithm part.
- The standard deviation for the exchange part.

For testing purposes, we choose a simple environment with square block robot in a three-dimensional configuration space (X, Y, θ). For the analysis $\alpha = 1/2$ and $k = 100$.This environment is shown on Fig. 14.

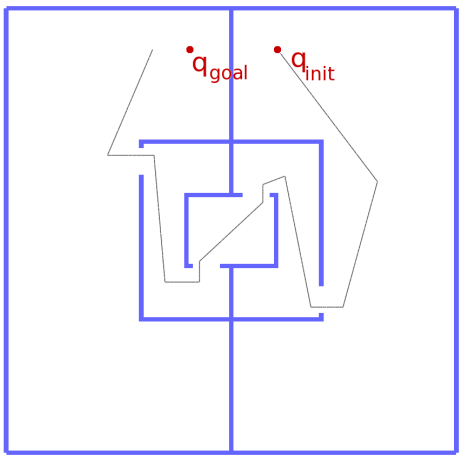


Figure 14: Testing environment. The user's path is displayed in black.

The user dexterity is a parameter difficult to model. In the tests, it is represented by the user speed along a predetermined path $path_p$(Fig. 14). For each iteration of the algorithm part (each time a new configuration is sampled), the user's position is moved by a step ε_v along this path. ε_v is used as equivalent of the user speed. The tests were run for four different values of ε_v.

Four different sampling methods were chosen for the tests:

- *POS*: Gaussian sampling around the user's position.

- $POS + CS$: Gaussian sampling around the user's position and uniform sampling in the configuration space.
- $PATH$: Gaussian sampling around the user's current path (path part along $traj_p$ between start position and user's position).
- $PATH + CS$: Gaussian sampling around the user's current path and uniform sampling in the configuration space.

The sampling method defines how to use the user input to gain information, explore the environment and find the solution. All the methods are based on gaussian sampling as it allows randomness to be guided (by user input in our case). The standard deviation σ is the parameter that defines the influence of the guide on the gaussian sampling. Four different values of standard deviation were chosen for the tests.

Visual and haptic feedback are not considered during the tests. The results shown are average over 50 runs. The time limit for each test was fixed to 900 seconds.

6.1 Sampling

Fig. 15 shows the computation time given the user speed for each values of the standard deviation. Curves (a) to (d) shows the results for each particular sampling method.

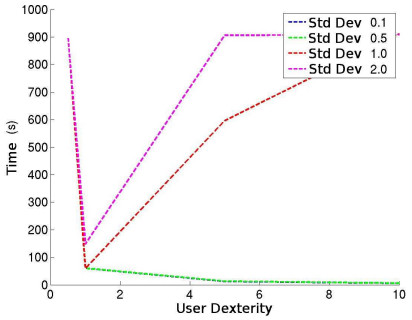
On each curve, we can observe that the computation time is higher (almost reaching the time limit) for lowest user speed values. This is due to the fact that in this case, the user is not fast enough to find a solution before the time limit. When a uniform sampling method is used, however, the problem can be solved within time limit.

The curve (a) shows the results for the POS sampling method. In this case, high values of σ (red and magenta on Fig. 15-a) shows the worst performances because it makes the user position tracking less efficient, and the narrow passages of the environment reduce the probability of linking a new sample to the existing roadmap for this method. On the contrary, if the sampling is precise enough when following the user (low values of σ, blue and green), the increase of the user speed will benefit to the algorithm.

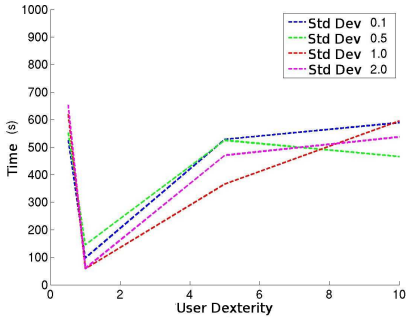
The curve (b) shows the results for the $POS + CS$ sampling method. In this case, a uniform sampling in the configuration space is added to the gaussian sampling around the user position. (one sampling of three is chosen with the uniform method). We observe on the curve that it introduces a latency in computation time for low σ values, but speeds up the resolution for high values. The addition of the uniform method reduces significantly the dependence of the algorithm on the standard deviation.

The curve (c) shows the results for the $PATH$ sampling method. In this case, σ has no influence over the path planning process. The resolution time always benefits from the increasing user speed. The memorization of the user path allows to deal with narrow passages and problems of link between the samples and the existing tree.

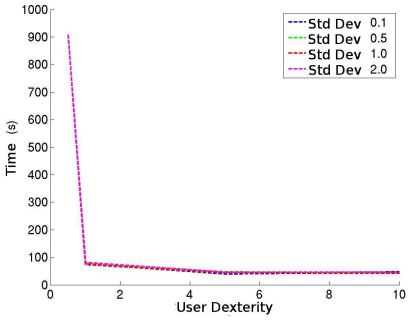
The curve (d) shows the results for the $PATH + CS$ sampling method. The computation time increases a little bit compared to the $PATH$ sampling method, and the differences due to σ are visible, but the addition of the uniform sampling has no significant influence on the planning.



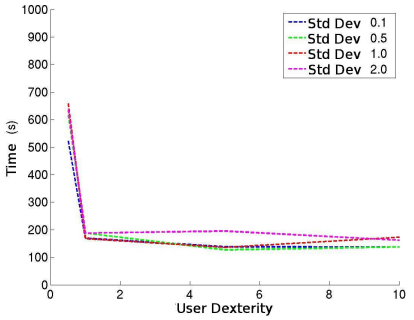
a - Around user position (POS)



b - Around user position and uniform ($POS+CS$)



c - Around user path ($PATH$)

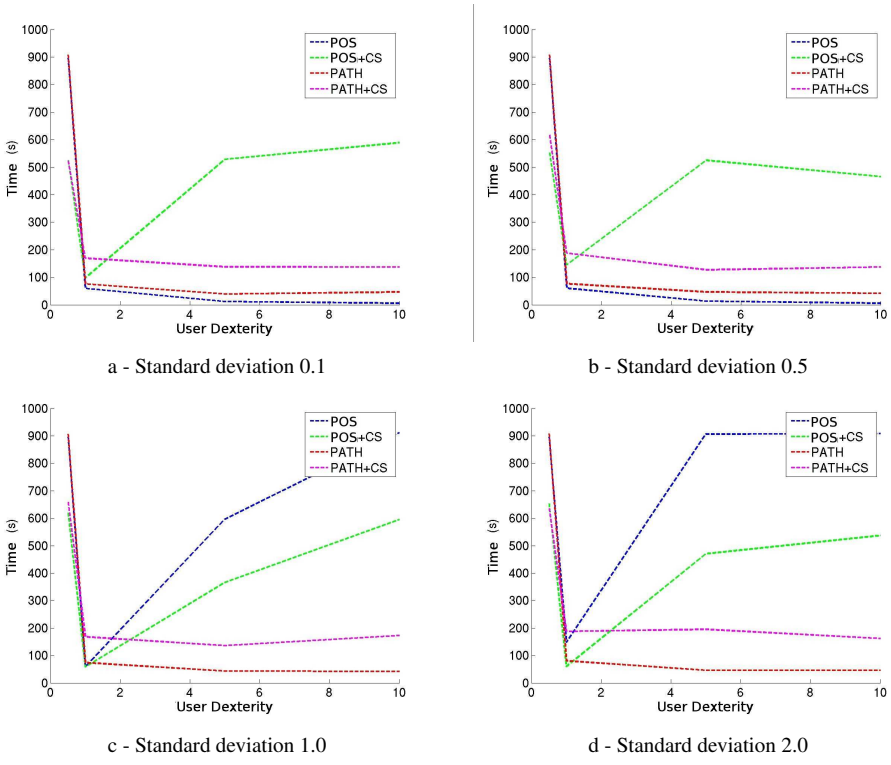


d - Around user path and uniform ($PATH+CS$)

Figure 15: Comparing the standard deviation effect on resolution time given the user speed for different sampling methods.

6.2 Standard deviation

Fig. 16 shows the computation time given the user speed for each sampling method. Curves (a) to (d) shows the results for each value of σ.



a - Standard deviation 0.1

b - Standard deviation 0.5

c - Standard deviation 1.0

d - Standard deviation 2.0

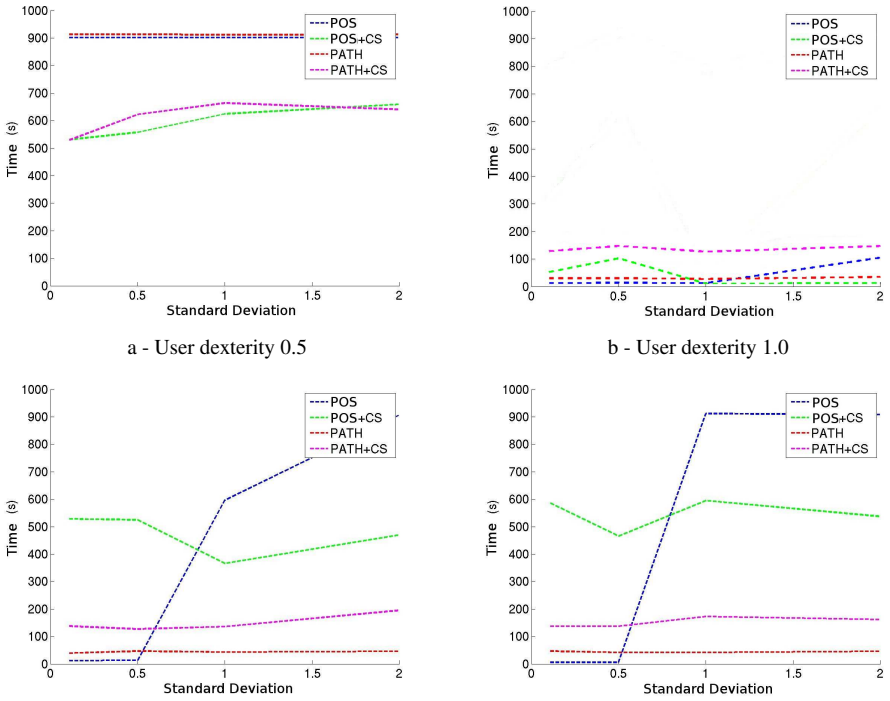
Figure 16: Comparing the sampling method effect on resolution time given the user dexterity for different standard deviation values.

This figure allows us to compare each sampling method directly given a single value of standard deviation. On curves (a) and (b) (low standard deviation values), we can observe that the addition of a uniform component to the sampling method increases the computation time (green and magenta). But while the $PATH+CS$ method follow the same kind of path as $PATH$ and POS (due to the memorization of the user path), the method $POS+CS$ has worse results as the user speed increases. This is due to the slow down caused by the uniform sampling and the linking problem when approaching narrow passages. The POS method gives the best performances: this method benefits from the user speed more than others.

On curves (c) and (d), we can see that the two POS based methods are giving worse results because of the increase of the standard deviation. We can observe on curve (d) that, without the supply of a uniform component, the POS method cannot resolve the problem within the time limit (blue).

6.3 User dexterity

Fig. 17 shows the computation time given the standard deviation for each sampling method. Curves (a) to (d) shows the results for each value of ε_v.



a - User dexterity 0.5

b - User dexterity 1.0

c - User dexterity 5.0

d - User dexterity 10.0

Figure 17: Comparing the sampling method effect on resolution time given the standard deviation for different user dexterity values.

On curve (a), we can observe that computation time are high for every sampling method. As said earlier, this is due to the user speed being to low to reach the goal before the time limit. Sampling methods without a uniform component cannot resolve the path planning problem in this case, but we can see that other methods can do it before the time limit.

On curve (b), we can observe that no matter the σ value and the sampling method, the performances are similar when $\varepsilon_v = 1.0$.

On curves (c) and (d), we see that $PATH$ based sampling methods are not affected either by the user speed or the standard deviation. The $POS + CS$ method is affected by the user speed but not by the standard deviation (green). This is due principally to the uniform sampling because if we look at the POS method, the computation time increases with the standard deviation (narrow passages and linking problem). However, for low values of σ and high user speed, this method is the most efficient.

In this section, three parameters of the I-RRT algorithm were analyzed. The results showed that each of these parameters are important and modifies the performances of the algorithm. They must be adapted to the user capacities and the problem environment.

Sampling methods without a uniform component (POS,$PATH$) are more adapted for an experienced user that can move the avatar in the virtual scene fast and with precision. They can take full benefit from the user guidance. Sampling methods with a uniform component ($POS+CS$,$PATH+CS$) are more efficient when the user is inexperienced (slow user speed) or cannot find the solution. They ease the problem resolution, but require more time. Path tracking methods ($PATH$,$PATH+CS$) are very little affected by user speed or standard deviation. They constitute a good compromise for unknown user capacities or environment.

7 Conclusion

In this paper, we presented a new method allowing a human operator and a path planning algorithm to cooperate for finding a path in an efficient way. The main idea is to gather the algorithm and the user input into the same planning loop, and make them exchange information about the search using visual hints and by exchanging pseudo-forces. The method is made of two separate loops one including the user interaction and the other the planning algorithm. These two loops are communicating through the virtual scene using the pseudo-forces. These pseudo-forces are generated by the user with the interactive device and by the algorithm which uses collected data to compute them. The principal advantage is that user and algorithm are not decoupled in a two step algorithm but work together at the same time, with a varying influence over the path search. Our approach based on RRT algorithm generates non-optimal paths. [16] shows that RRT does not converges to an optimal solution almost surely as the number of samples increases. In our problem, the path optimality is less a priority than finding a path while taking into account the operator constraints (constraints that are not mathematically described easily but are known by the user : relevant work experience, business processes). Different cases are presented to show the validity of the method : a simple two-dimensional case which illustrates the principle, a difficult six-dimensional case, and an industrial case from the automotive industry. The studied cases show that the search benefits from each part : the user speeds up the search by going through difficult passages and the algorithm gives global information to help the user and improve approximate solutions. The usage of both a haptic arm and space mouse in these cases shows that the method can be easily adapted to different devices. Finally, the analysis shows that different parameters can be tuned to adapt to different situations.

In this paper, the contact forces between the object and the obstacles are not considered because they can cause difficulties to the user. They can be useful in the path search if slip is considered and if the user has the possibility to easily deactivate these contact forces when he needs it. In several manipulation tasks slip can guide the user to the solution (ex: steering a key in its keyhole). But if the user get stuck in a dead-end or in a very narrow passage, he must be able to deactivate these contacts to get out of this situation easily and resume the path search in another part of the virtual scene.

It could be interesting to analyze the influence of the different parameters on the algorithm performances for finding paths (number of nodes and number of iterations) and on the shape of the solution path (is it a broken line or a smooth curve). The dynamic tuning of the parameters during the path search phase could be done using this analysis. This allows the algorithm to automatically adapt to the topology of the different parts of the virtual environment or to changing environments. The comparison between solutions obtained by a search leaded by the user (the user is more influential) with solutions obtained by a search leaded by the algorithm can show the respect of human-defined different criteria (is the path natural? intuitive? human feasible?). As the interactive device is an important part of the method, it could be interesting to study its influence on the search and the difference of performances between haptic and non-haptic devices. The visual feedback can also be improved by changing colors (according to labels) in a continuous gradation in a background image instead of displaying each node and edge. This would lighten the display. The cases studied in this paper focus on free-flying objects and assembly problems, but the method can be applied to different situations such as moving animated characters or multi-user path search for very complex problems (using several interactive devices). The algorithm can theoretically be ran on embedded systems but, as is, it will not be suitable for real-time applications as it needs computing power.

8 acknowledgements

Thanks to E Ferré to make available the Haption haptic Device in KineoCAM Company.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Workshop on Algorithmic Foundations of Robotics*, pages 155–168, 1998.

[2] B. Bayart and A. Kheddar. Evaluation of an haptic step-guidance algorithm through a teaching 2d/3d path scenario. In *Haptic, Audio and Visual Environments and Games, 2007. HAVE 2007. IEEE International Workshop on*, pages 124 –129, 12-14 2007.

[3] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. In *IEEE Int. Conf. on Robotics and Automation (ICRA'2000)*, pages 529–536, april 2000.

[4] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *IEEE Inter Conf. on Robotics & Automation (ICRA'99)*, pages 1018–1023, 1999.

[5] G. Burdea. Haptics issues in virtual environments. In *Computer Graphics International*, pages 295–302, 2000.

[6] C. Chabal, C. Megard, and L. Sibile. Emm-3d : a virtual environment for evaluating maintenability from cad models. In *Laval Virtual*, 2005.

[7] C. Chen, S. Ong, A. Nee, and Y. Zhou. Haptic-based interactive path planning for a virtual robot arm. *International Journal on Interactive Design and Manufacturing*, 4(2):113–123, 2010.

[8] L. Chen and C. G. Brown. A 3d mouse for interacting with virtual objects. In *IEEE Inter. Symposium on Circuits and Systems*, pages 5011–5014, 2005.

[9] E. Ferré and J. Laumond. An iterative diffusion algorithm for part disassembly. In *IEEE Int. Conf. on Robotics and Automation (ICRA'2004)*, pages 3149–3154, april 2004.

[10] E. Ferré, J. Laumond, G. Arechavaleta, and C. Estevés. Progresses in assembly path planning. In *Int. Conf. on Product Lifecycle Management (PLM'05)*, pages 373–382, july 2005.

[11] D. Flavigné and M. Taïx. Improving motion planning in weakly connected configuration spaces. In *IEEE Int. Conf. on Intelligents Robots and Systems*, pages 5900–5905, 2010.

[12] D. Flavigné, M. Taïx, and E.Ferré. Interactive motion planning for assembly tasks. In *IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN 09)*, pages 430–435, 2009.

[13] D. Galeano and S. Payandeh. Artificial and natural force constraints in haptic-aided path planning. In *Int. Workshoop on Haptic Audio Visual Environnments and their Applications*, pages 45–50, October 2005.

[14] X. Hea and Y. Chen. Haptic-aided robot path planning based on virtual tele-operation. *Robotics and Computer-Integrated Manufacturing*, 25(4-5):792–803, 2009.

[15] L. Jaillet, J. Cortes, and T. Simon. Sampling-based path planning on configuration-space costmaps. *Robotics, IEEE Transactions on*, 26(4):635 –646, aug. 2010.

[16] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.

[17] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846,894, June 2011.

[18] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics & Automation*, 12(4):566–580, June 1996.

[19] N. Ladezeve, J. Y. Fourquet, and M. Taïx. Interactive motion planning in virtual reality environments. In *Virtual Reality International Conference (VRIC'08)*, april 2008.

[20] J.-C. Latombe. *Robot Motion Planning*. Kluwer, Boston, MA, 1991.

[21] J. Laumond. Kineo cam: a success story of motion planning algorithms. *IEEE Robotics & Automation Magazine*, 13(2):90–93, june 2006.

[22] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Computer Science Dept., Iowa State University, oct 1998.

[23] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[24] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Proceedings Workshop on the Algorithmic Foundations of Robotics (WAFR'00)*, 2000.

[25] A. Lecuyer, A. Kheddar, S. Coquillart, L. Graux, and P. Coiffet. A haptic prototype for the simulations of aeronautics mounting/unmounting operations. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 182 –187, 2001.

[26] A. Lecuyer, M. Vidal, O. Joly, C. Megard, and A. Berthoz. Can haptic feedback improve the perception of self-motion in virtual reality? In *Int. Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 208–215, 2004.

[27] L.Zhang and D. Manocha. An efficient retraction-based rrt planner. In *IEEE Inter. Conf. on Robotics and Automation*, pages 3743 –3750, may 2008.

[28] N.Ladeveze, J.Y.Fourquet, B.Puel, and M.Taïx. Haptic assembly and disassembly task assistance using interactive path planning. In *IEEE Virtual Reality (IEEE VR 09)*, 2009.

[29] M. Rantanen. A connectivity-based method for enhancing sampling in probabilistic roadmap planners. *Journal of Intelligent & Robotic Systems*, pages 1–18, January 2011.

[30] J. Rosell, C. Vazquez, A. Perez, and P.Iniguez. Motion planning for haptic guidance. *Journal of Intelligents. Robotics Systems*, 53(3):223–245, 2008.

[31] T. Siméon, R. Chatila, and J. Laumond. Computer aided motion for logistics in nuclear plants. In *Int. Symposium on Artificial Intelligence, Robotics and Human Centered Technology for Nuclear Applications (AIR'02)*, pages 46–53, january 2002.

[32] T. Siméon, J.-P. Laumond, C. V. Geem, and J. Cortés. Computer aided motion: Move3d within molog. In *Int. Conf. on Robotics and Automation (ICRA'2001)*, pages 1494–1499, may 2001.

[33] M. Strandberg. Augmenting RRT-planners with local trees. In *IEEE Inter. Conf. on Robotics & Automation (ICRA'04)*, pages 3258–3262, 2004.

[34] Y. Su, W. Zhu, and T. Yu. Virtual assembly platform based on pc. In *Inter. Conf. on Audio, Language and Image*, 2008.

[35] Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J. H. Reif. Narrow passage sampling for probabilistic roadmap planning. *IEEE Transactions on Robotics*, 21(6):1105–1115, 2005.

[36] C. Vazquez and J. Rosell. Haptic guidance based on harmonic functions for the execution of teleoperated assembly tasks. In *IFAC Workshop on Intelligent Assembly and Disassembly*, May 2007.

[37] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Maprm: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *IEEE Inter. Conf. on Robotics & Automation (ICRA'99)*, pages 1024–1031, 1999.

[38] N. Ye, P. Banerjee, A. Banerjee, and F. Dech. A comparative study of assembly planning in traditional and virtual environments. *IEEE Transactions on Systems, Man, and Cybernetics*, 29(4):546–555, 1999.