Motion Planning with interactive devices

Michel Taïx, David Flavigné

Motion Planning with interactive devices

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

***Abstract*—This paper presents interaction between a user and a robot to guide motion through motion planning algorithm. 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 a Interactive-RRT. This method is based on exchanging 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 and analysis are done in function on the user's dexterity to manipulate devices.**

I. Introduction

This work concerns the conception and the realization of a new approach to solve problems in mechanical assembly of numeric models (3D C.A.D) based on interaction between a user and a motion planning algorithm. The assembly problem inside the digital mockup can be solved generally by two approaches : virtual reality or automatic path planning.

The goal of using Virtual Reality is to improve the quality of the designed parts and to reduce the cost and time needed for their final design. The classical approach uses a Virtual Reality environment and an interaction device (3D mouse or a haptic arm) in order to interact with CAD software. Haptics is a recent enhancement that provides an additional perceptual modality in virtual environments [3]. and improve the use of Virtual Reality [21], [4]. The use of a 3D mouse for interaction, is less expensive than a haptic device and can give very good results in many design processes [5]. But in complex scenes with a real non-convex object, the user may need help to find a solution.

In robotics, a lot of work in motion planning has been done to compute free paths in digital models for mechanical systems. With the recent results in random planning algorithm [14] 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. Now, companies uses automatic path planning to solve PLM applications [7], [12].

There are mainly two families of methods for building a roadmap, the Probabilistic Roadmap (PRM) [10] and the Rapidly-exploring Random Tree (RRT) [15]. The roadmap in the PRM's case is a graph and in RRT's case a tree. 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 [6], 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 [2], [20], [19], [18]). 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 passage 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 complementary fashion. The complementarity between user and algorithm is the essential idea of this work.

We choose to integrate the human into the planning loop via an interaction device, which can be a 3D 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 main contribution of the method is to allow simultaneously cooperation between the loop of roadmap search and the loop of interaction for the user.

The paper is organized as follow. We begin by describing previous work in interactive motion planning. The general interaction loop between user and motion planning algorithm which is at the basis of our work and the problem statement are summarized in section III. 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 functions are presented in section V. Finally, simulation results with an industrial application device are presented and analysed in section VI and VII.

II. Previous works

The idea of taking into account user input has been studied by various authors specialized in motion planning. In [1] the authors use user-input to work with 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 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 [17]. 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 Algorithm including heuristics based on a study of the workspace are presented in [11], [16]. 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 [9] the user selects critical robot configurations with an haptic interface to facilitate the automatic path planning research. The advantages of human's 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.

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. In our method we do not separate the two stages.

Following result in [8], this work presents an industrial application with haptic device and an analysis of the behavior of the algorithm based on user interaction.

III. Approach Overview

We consider that the human use an interactive devices, I-Device, (3D-mouse or a haptic device) to guide the robot motion in a virtual environment. The proposed approach aims at improving the operator guidance with the help of an automatic path planning algorithm.

We consider that the user only wants to guide the trajectory of the object without controlling all the dof. The user move an objet for assembly/dissasembly robotic stack (free-flying robot) or a robot.

The approach has to take into account two constraints:

- The user's direction of movement has to lead the planner development.
- The planner has to return some useful information to the user (haptic and/or visual information) concerning obstacles proximity and if this area has already been explored.

The general plan is constituted by two main loops which will work in parallel (see figure 1). In one loop, the path planner runs to explore the free configuration space and searches an automatic solution. The RRT method is modified to take into account the user interaction and to search a solution in the six dimensional configuration space

In the other loop, the user moves the objet with the I-Device as he wishes. The user input, F_u, moves the object in the virtual environment using the physics laws appropriate for the feedback perception. The returned algorithm pseudo-force, F_a, must be seen as disturbance. 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. If the I-Device is a real haptic device, these vectors are transformed into physical forces. We use the name *pseudo-force* also to represent the real force with haptic device.

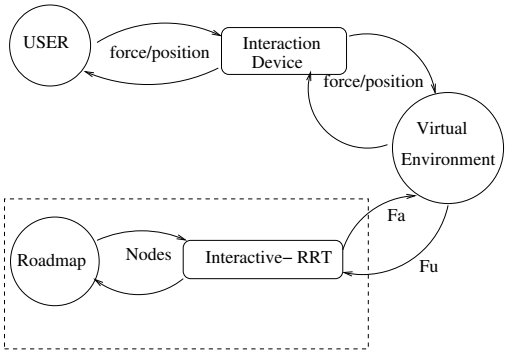


Fig. 1. General loop

IV. RRT Path Planning

[13] introduced the Rapidly-exploring Random Tree (RRT) approach which is the base of our method. The RRT algorithm is shown in Algorithm 1.

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 attemped. If the extension succeeds (*CONNECT*) a new node, q_{new}, and an edge in the roadmap are created.

Algorithm 1 The RRT-CONNECT algorithm

```
T(q_init)
for i = 0 to N do
  q_rand ← RANDOM_CONFIG()
  q_near ← 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
```

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 Euclidian distance but also the most interesting nodes with regard to the task of the user.
- At each iteration, the new I-Device position must be taken into account to compute these differents 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.

V. Interactive Motion Planning System

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 is sampled and the extension from the nearest node in the tree toward this sample is attempted. If the extension succeeds a new node in the roadmap is created at a distance ε of the nearest node. This extension process will be called classical extension in this article.

The idea of the I-RRT is to take advantage of both human and computer capabilities for planning a motion into a virtual scene. The algorithm is shown in Algorithm 2 and the principal steps are:

- Compute an attractive pseudo-force F_a from the current tree using the roadmap data (nodes) to influence the user movement via an I-Device and haptic and/or visual hints.
- Compute an interaction pseudo-force F_u from the user's input (position and/or movement) to take into account the user's intention.
- Compute sampling from a pseudo-force F_r: choose an efficient way to combine user's intentions and algorithm automatic search (from F_a and F_u).
- Choose nearest neighbor in the tree.
- Extend the roadmap and label the nodes.

A. Pseudo-force computation

We first retrieve the user's position in the virtual scene and then take some roadmap nodes (the p nearest nodes) near this position (*NEAREST_NEIGHBORS*). After getting their respective labels, we compute the center of mass. The attractive pseudo-force F_a (*Compute_FAlgo*) is given by the vector that goes from the user's position (q_{user}) to the center of mass (see figure 2).

This force is aimed to avoid the user getting to far from the roadmap, which can make him too difficult to follow for the algorithm.

Algorithm 2 The Interactive RRT Algorithm

$T(q_{init})$
for $i = 0$ to N **do**
  $q_{rand} \leftarrow$ SAMPLED_CONFIG(F_a,F_u)
  $q_{near} \leftarrow$ NEAREST_NODE(q_{rand}, T)
  $L_{near} \leftarrow$ NEAREST_NEIGHBORS(q_{user}, T)
  if CONNECT(T, q_{rand}, q_{near}, q_{new}) **then**
    Add_Vertex(T, q_{new})
    Add_Edge(T, q_{near}, q_{new})
    Update_Labels(T, q_{near}, q_{new})
    $COM \leftarrow$ Compute_CenterOfMass(L_{near})
    $F_a \leftarrow$ Compute_FAlgo(COM, q_{user});
    $F_u \leftarrow$ Compute_FUser(q_{user});
  end if
end for

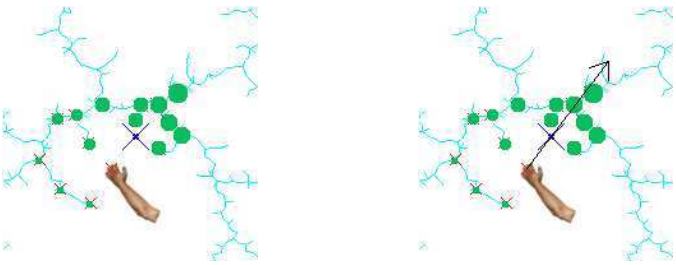


Fig. 2. Get weights of the nearest neighbors and compute the Center of Mass and the Force F_a.

The user interaction pseudo-force F_u is return by the I-Device or computed by user's movement (*Compute_FUser*).

With the two previous computed pseudo-forces, we can compute a new one by :

$F_r = \alpha.F_u + (1-\alpha).F_a$ (see figure 3).

The parameter α allows to tune the part of user's intention compared with the algorithm search. If α is set to zero, the algorithm is equivalent to a RRT, without user input. In our implementation, α is fixed and constant. F_r is used for the sampling.

B. Sampling method

The sampling is the basis of our method for interactivity between the algorithm and the user. In many probabilistic motion planning methods, the sampling is crucial as it determines the way the roadmap extends itself. Allowing the user to control the sampling is allowing him to lead the search. The simplest solution is to sample around the user's position, but that means the algorithm is no help to the user and simply follows him, when possible.

We use F_r to define a Gaussian (elliptic) shooting area centered on the user's position and oriented along the force direction. This allows the algorithm being guided toward the user's position while still being close to the space that has already been explored i.e., the roadmap.

Meanwhile, F_a is sent to the I-Device, thus providing a guiding pseudo-force that will slightly influence the user's movements, and give him some information towards a good

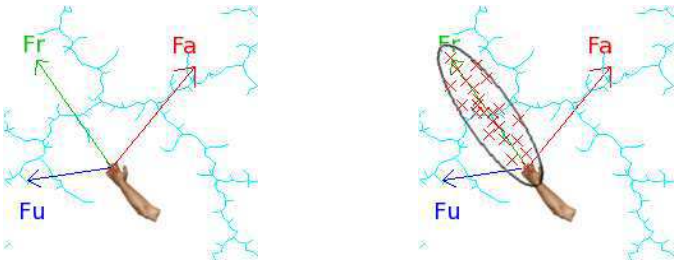


Fig. 3. Sampling area taking into account F_a and F_u.

direction to follow according to nodes labels. The aim is to have a sampling area deformed along the pseudo-force direction. If the user moves the I-Device quickly, the sampling area is close to the F_u direction (in dimension 6)

The sampling is realized by shooting N different random variables (N corresponding to the dimension of the *CS*) with each a Gaussian probability center along the F_r direction.

To take advantage of the algorithm capabilities we also allow random samples (uniform distribution) once every x shoots (x is dynamically set from the user's force). This allows the algorithm to explore globally the virtual scene and to gather data from it.

C. Nearest neighbor

The nearest neighbor search is done by comparing the Euclidean distance between the sampled configuration and each node of the roadmap.

Then we can compute the center of mass using the previously computed labels (distance to goal configuration) as weights for each node. We take a fixed number p of neighbors (the nearest ones) for this.

D. Extension

First we compute a classical extension. Then, we analyze the result and classify it in one of these three categories:

- Collision : extension has failed
- Reached : extension succeeds and reaches the last shot sample q_{rand}
- Obstacle : extension reaches the node q_{new} but did not reach the shot sample q_{rand}.

E. Node Labelling

According to this, we compute collision labels for the new node and the extended node.

In the RRT algorithm, we shoot randomly configurations and then tests if they are colliding. We do not know "a priori" the scene topology. The only piece of information we can obtain is from these collisions tests. In the interactive algorithm, we used them to label roadmap nodes. There are two types of node labels that are taken into account in the algorithm: the distance from the node to the goal configuration and the number of collisions that occur from trying to extend the node (towards a shot configuration, when this node was chosen as the nearest neighbor).

The distance label is a function to guide towards the goal.

The collision label is more interesting. It is a function of the covered distance during the extension, the extended node's number of extensions and his number of collisions.

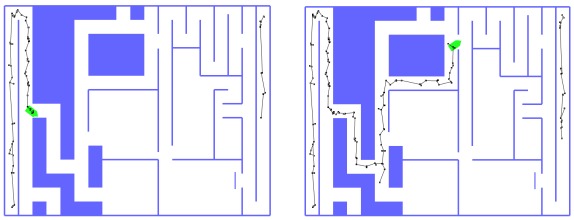
These labels are used to give visual hints the user and influence his movement. Information delivered by these labels are displayed visually by colorizing nodes.

VI. Results

We use a haptic device ($haption - virtuose$) to move the object in CAD model. We did not forbid the penetration of the virtual object (moved by the user) into obstacles, because without a real 3D display the user can be stuck in a narrow passage in a complex environment without seeing why he cannot move. During all the tests the α parameter is equal to 0.5 for balancing user and algorithm part. The number of nearest neighbors p is fixed to ten.

The algorithm was implemented in C++ in the software platform HPP developed at LAAS based on KineoWorks[1] The experiments were performed on a PC Dual Core, 2.1 Ghz and 2 GB ram.

A. Simple 2D Labyrinth



(a) (b)

Fig. 4. The I-RRT follows the user. (a) At start, nodes are sampled independently. (b) Algorithm tracks the user's position.

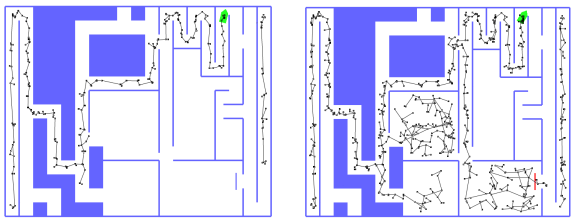
The aim of this example is just to show the principle of our algorithms when running. We have to move an arrow-shaped block (Green blocks in figure) in a 2D labyrinth from the left-bottom corner to the right-top corner (see figure 6. The block has three degrees of freedom (x, y, θ, CS is a 3-dimensional space).

At the beginning of the labyrinth, either user and algorithm can easily find the path to follow. Then, the block must go through a narrow passage, which entrance is hard to find for the algorithm. So the user goes forward. The algorithm has to track the user's position (figure 4).

After the narrow passage, there are two big rooms, with no possible exit. The user can see that and avoid going into them. A classical RRT algorithm will go inside (see figure 6) and try to explore them, which is really time consuming.

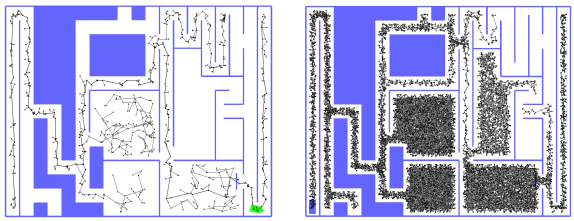
In figure 4 (middle), the user sees a possible solution and tries it. Unfortunately, the block cannot turn and the user is stuck. During the time the user tries to go through this passage, the algorithm explores other ways and finds another passage (figure 4, right). The user cannot see this solution, it is often

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



(a) (b)

Fig. 5. Breaking the deadlock.(a) The user fall in a dead end.(b) Extension of the roadmap.



(a) (b)

Fig. 6. Roadmap comparison between Interactive solution (a) and RRT solution (b).

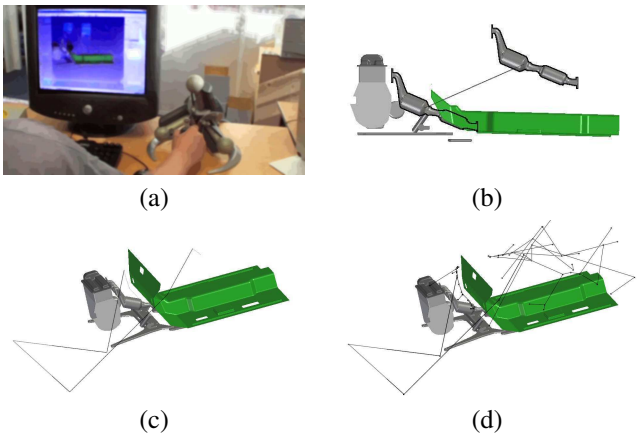
the case when the user has to move an object in a complex 3D CAD model. It is the relevance of the I-RRT to guide the user and explore the 6D configuration space.

Finally, the user follows the path found by the algorithm and reaches the final configuration (figure 6).

B. Industrial case in PLM application

Next example is real use-cases from the automotive industry. The problem is to find a collision free path dismounting a part of a car (a silencer) to check the manufacturing process or the maintainability of the assembly.

We show only a part of the real environment for visualization problem and the environment is limited to represent the reality.



(a) (b)

(c) (d)

Fig. 7. The automotive industrial case.

The input of the algorithm is a CAD model with initial and final configuration for the silencer (figure 7). The goal of the user is to take out the silencer by the top.

The example is solved in a few minutes (solution in figure 7 The computation time with the basic RRT takes much more time.

The computation time to find the solution is strongly dependant on the user dexterity with the 3D-mouse. If the user do not move the 3D-mouse (or move in the deadlock direction) the computation time is the RRT time computation.

The roadmap build by the I-RRT with the user's motion is shown in figure 7.

VII. ANALYSIS

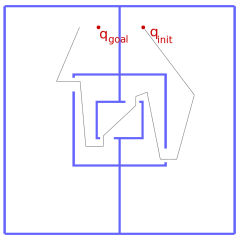


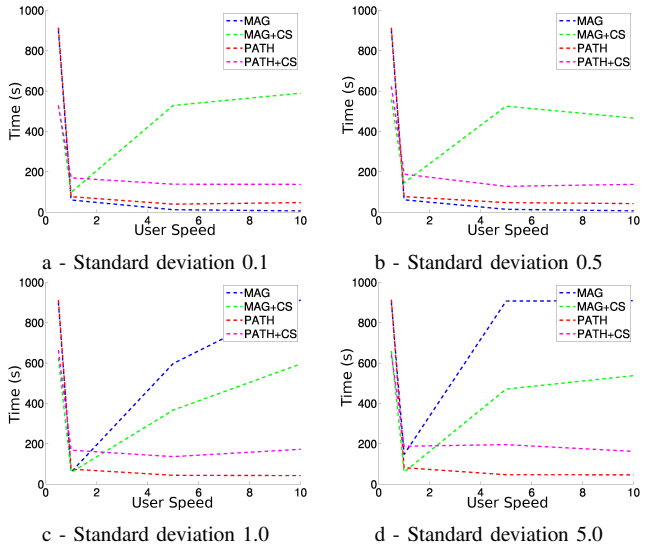
Fig. 8. Environment used for the tests. The user's simulated path is represented in black.

Due to the base principle of this algorithm, it is really dependant on the user's dexterity to manipulate devices and find solutions. Several parameters can greatly modify its performances. We choose the three parameters that seemed to influence the most the computing time. The first one is the sampling method. From the method described in the article, we derived three other sampling method for comparison. Thus, we have four sampling methods : a method that samples around the user's position (MAG), a method that samples around the user's path ($PATH$), and the same methods but with a uniform sampling added ($MAG+CS$ and $PATH+CS$). The second parameter is the standard deviation used by the sampling methods. It modifies how precise is the tracking of the position/path of the user. Four different values were used. The last parameter represents the user's dexterity and is the user's speed. During the tests, the user's speed was simulated by incrementing the user's position by a step ε along a predefined path. Four different values were used.

On figure 8, we can see the environment used for the tests. All the results on the curves are means over 50 runs. The computing time was limited to 900 seconds.

On figure 9, we can observe differences between sampling methods concerning computing times given the user's speed. Each of these four curves shows the case for one standard deviation value. On parts (a) and (b), we can see that the sampling method MAG is more efficient than others in terms of computing time. It takes the best out of the user's speed. The performances with the $PATH$ method are negligible. When a uniform sampling component is added, we observe that the computing time is substantially increasing. (magenta curve $PATH+CS$ and green curve $MAG+CS$).

On parts (c) and (d), the methods based on user's position tracking are penalized by the increase of the standard deviation



a - Standard deviation 0.1 b - Standard deviation 0.5

c - Standard deviation 1.0 d - Standard deviation 5.0

Fig. 9. Comparing effects of the sampling method on the computing time given the user's speed for different standard deviation.

value. On each part, the first values of the curves are equivalent to the time limit because the user was not fast enough to reach the goal configuration before.

We can conclude that the user's speed, the standard deviation and the sampling method are three important parameters that modifies performances. They should be adapted for each case. Methods based on simple user tracking (without uniform sampling of the configuration space) are adapted to experienced user who can move fast and with precision. Methods with a uniform component can improve performances for unexperienced users. They allow a global exploration of the configuration space but are more time consuming. Methods based on path tracking ($PATH$, $PATH+CS$) are almost not affected by the user's speed and the standard deviation.

VIII. CONCLUSIONS AND FUTURE WORKS

In this work, we described a simple interactive method designed for assembly/disassembly planning. This method is based on pseudo-forces exchange between the algorithm and the user, and on data gathering (labels) from the virtual scene. We saw that it works well in simple environments and real industrial case: the algorithm and the user can help each other, and the trajectory search benefits from each part.

We should integrate a real haptic device in order to use real forces and contact perception from the virtual scene instead of using only movements speeds. We have to define a normalization forces, in order to let the user decide all along the process. Visual hints have to be inserted from the labels to have a visual feedback. We also plan to make more tests on real industrial cases. An interesting point can be to decouple partially the 3D user motion and the 6D configuration space in which the graph is developed, to accelerate the algorithm. Finally, in this work the user moves an object in a six dimensional space and the roadmap is built in the same dimensional CS, it can be interesting to extend the same method for moving object with an interactive device with a higher dimensional roadmap. For example the user move an object (6 D) which is held by a virtual mannequin arm (dimension 10 or more).

IX. ACKNOWLEDGMENTS

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

REFERENCES

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

[2] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 1018–1023, 1999.

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

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

[5] L. Chen and C. G. Brown. A 3d mouse for interacting with virtual objects. In *IEEE International Symposium on Circuits and Systems*, 2005.

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

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

[8] 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.

[9] X. He and Y. Chen. Haptic-aided robot path planning based on virtual tele-operation. *Robot. Comput.-Integr. Manuf.*, 25(4-5):792–803, 2009.

[10] 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.

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

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

[13] 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.

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

[15] 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.

[16] 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.

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

[18] M. Strandberg. Augmenting RRT-planners with local trees. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3258–3262, 2004.

[19] 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.

[20] 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 *Proceeding of IEEE International Conference on Robotics & Automation*, 1999.

[21] 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.