I-RRT-C : Interactive Motion Planning with Contact

Nassime Michel Blin, Michel Taïx, Philippe Fillatreau, Jean-Yves Fourquet

Interactive Motion Planning with Contact

Nassime Blin[1] , Michel Taïx[2] , Philippe Fillatreau[3] and Jean-Yves Fourquet[3]

***Abstract*— This work deals with planning processes for the assistance to manipulation in order to simulate industrial tasks such as assembly, maintenance or disassembly in Virtual Reality. This paper presents a novel interactive path planning algorithm with contact based on an RRT-Connect approach.**

First, we propose a real-time interactive planner where both a computer and a human operator can simultaneously search the workspace therefore largely speeding up the process. An authority sharing parameter can control the autonomy of the computer.

Then we present a novel contact space algorithm able of sampling on the surface of obstacles. This method helps finding paths in cluttered environments or solving specific contact tasks such as insertion or sliding operations. We finish by presenting the results of our interactive path planner with contact through two examples showing significant improvement over usual methods in both free and contact space.

I. Introduction

Our goal is to plan a motion for an industrial part which will be called object in the rest of this paper. Using widely known motion planners we can find a solution to theses problems.

Probabilistic planners such as RRT can be very slow to solve problems in difficult spaces such as cluttered or narrow passages. Therefore, we can use the help of a human operator to solve the planning problem faster. Often humans can find a path very fast or see that a passage is impossible almost instantaneously. On the opposite a human alone may look a long time for a path in an impossible passage because the navigation on the six dimensional space (position and orientation) is difficult. This is why we believe that combining both the computational power of an automatic planner and the capacity of a human operator can be rewarding for planning the motion of an object.

Some industrial need surfaces of obstacles to be fully used and for planning processes in opposition to standard motion planning where avoiding obstacles is the objective. Assembly itself means getting objects to touch each other. Industrial examples may be sliding operations or insertion scenarios [1], [2], [3]. We believe that in theses cases, contact planning would require less nodes and time to find a solution path.

This paper is organized as follows : in section I a survey of motion planners related to our work is presented. Section II introduces our interactive algorithm. A novel contact algorithm is then presented in Section III. Section IV is dedicated to results presentation and analysis. At last, section V presents future work and conclusion.

[1]CNRS, LAAS, 7, avenue du colonel Roche, F31400 Toulouse, France Univ. Féd. de Toulouse, INP-ENIT Université Fédérale de Toulouse, France

[2]CNRS, LAAS, 7, avenue du colonel Roche, F31400 Toulouse, France Univ. Féd. de Toulouse, UPS, LAAS, F 31400 Toulouse, France

[3]INP-ENIT Université Fédérale de Toulouse, France

A. Path Planning

Among all different types of motion planners, we will here discuss only sampling based algorithms. These algorithms rely on the fact that they have probabilistic completeness. The two most used methods are the Probabilistic Roadmap Method (PRM) [4] and the Rapidly-exploring Random Tree algorithm (RRT) [5]. They have been extensively studied [6] and an excellent survey is available [7]. There exist real-time algorithms such as [8] but they are not interactive because tree expansion is not controlled by an operator. We focused our work on RRT because it is a faster solution than PRM, growing faster in constrained environments.

B. Interactive Planning

Here is presented a survey of interactive motion planning algorithms that share path finding between an algorithm and a human operator.

In [9] the authors present a method for cooperation between human and automatic motion planner. Forces of haptic device can guide and improve the interaction between them [10]. In other works, the user interaction can be made by a haptically controlled object to modify or define critical object configurations [11].

Ladevèze introduced an interactive planner [12]. A linear interpolation between the current configuration and the goal configuration generates an attraction force guiding a user through a haptic device.

Flavigné [13] introduced the Interactive RRT (IRRT). Its goal is to move an object in free space. This solution lets an operator control the sampling process using a haptic arm.

More recently, Cailhol [14] implemented an original multi-layered interactive solution. In two steps, he finds a topologic path in the environment and then finds a precise path using geometric information to control an object. Both of theses phases are controlled according to semantic information.

C. Contact Planning

Classically, the goal of motion planning is to move an object in the free space C_{free} of a workspace trying to avoid obstacles. Our goal is to plan motion at the surface of obstacles.

Redon [15] published a solution to locally plan on contact. When an in-contact configuration is found, the next generated node is projected on a set of valid variations. This set satisfies all global non collision constraints. The main drawback of this method is that it is not global and its goal is not to plan in contact but rather to get out of it after the first iteration.

Rodriguez [16] uses obstacle based information to generate configurations parallel to obstacles. This helps expanding PRM graphs in cluttered environments in a much better way than a standard PRM. Though this solution plans parallel to obstacles, it still cannot plan in contact.

Yu Yan [17] proposed a very efficient contact planning solution using a retraction technique. That is, an operator with a device draws a path allowed to have colliding parts. This path is then locally retracted to the surface in a post treatment to generate contact paths.

D. Contribution

To the best of our knowledge, there exist no interactive algorithm capable of searching the whole workspace and able to plan directly on surfaces. Therefore we have implemented a RRT interactive in contact algorithm. Our contribution is twofold: an in-contact solution without any post treatment coupled with an interactive algorithm greatly speeding up motion planning processes.

II. INTERACTIVE MOTION PLANNING

This section describes our interactive algorithm and it implementation details. We will show how can a usual RRT be used with the help of an operator.

A. The computer loop

The computer loop is a classic RRT-connect algorithm implemented in Hpp framework [18]. At each iteration, the algorithm tries to extend the roadmap in direction of a new random configuration. If the edge between the new configuration and the closest node of the roadmap lies in the free space, the new node is then added to the roadmap.

B. The human loop

Our solution uses an interactive device : a six degrees of freedom mouse.

The human loop consists in a moving object controlled by the mouse. The configuration q_{device} is the geometric center of the object's root body. This value is defined by the operator. Both the edges and the nodes of the roadmap can be displayed in a viewer, see figure 1. In this example, only operator configurations are displayed.

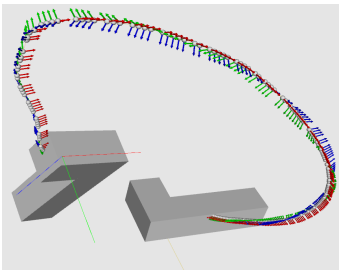

Fig. 1: Moving object and roadmap visualization

C. Interactive real-time motion planning

The presented solution lets an RRT-connect algorithm work with an operator using an interactive device through the visualization of the roadmap.

The interactivity factor α lets the user define the authority sharing between the operator and the computer. It represents the probability to have a computer authority. If $\alpha = 1$ this means the planner is fully automatic whereas if $\alpha = 0$ it is fully manual. Thanks to this factor, we can avoid the following problematic cases : an algorithm spending all its time searching in uninteresting places or an operator trying to pass through an impossible passage.

Algorithm 1 Interactive Planning

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

The algorithm 1 presents our interactive path planner. It uses as an input the workspace W, the tree T, the variable α and the configuration q_{device}.

Line 2 a random number a between 0 and 1 is picked. Depending on the value of a, this means the tree is extended sometimes in a random direction and sometimes in the direction pointed by the human operator in the workspace.

If $a > \alpha$ line 4, the extension is chosen in direction of the user defined configuration q_{device} given by the interactive device. We set the variable $q_{current}$ to equal q_{device}.

Line 5 the tree is extended in the direction of $q_{current}$. This is done by getting the nearest node q_{near} to $q_{current}$ in the tree T. The q_{near} to $q_{current}$ path is discretely validated as collision-free. Whenever an obstacle is met along this path, we name q_{new} the last valid configuration, add it to the roadmap and the q_{near} to q_{new} path is also added as a valid edge to the roadmap.

Line 7 in case $a \leq \alpha$, we set $q_{current}$ to equal a configuration obtained with a random shooter and add it to the tree T

III. CONTACT PLANNING

We will present in this section our novel in contact motion planner capable of sampling at the surface of obstacles. The overall behavior is first explained before presenting our sampling method. Then we give an example of contact sampling. Last we introduce our interactive motion planner in contact which is our main contribution.

A. Nearest Obstacle

We have models of both the environment and the object describing their geometry. A user moves the object in the workspace using an interactive device and whenever he approaches an obstacle sufficiently, the planner switches in contact mode at the surface of this obstacle. Moving the object away from the obstacle ends contact mode.

Using a collision detection library, we can measure the obstacle-object distance for every obstacle. The closest obstacle is then defined as the contact obstacle.

B. Stay in Contact

Starting contact mode, the planner samples on a local tangent plane to the nearest obstacle.

For each of these samples, the actual orientation of the object is chosen by the operator and kept constant during contact allowing only translations. New configurations are randomly chosen along the tangent plane. The object is then able to slide on the obstacles to enhance sampling in specific environments.

When the object switch to contact mode, a predefined number of contact samples are added to the tree. We chose this behavior because we want to multiply contact configurations to search more interesting space. Also, as the space may be cluttered, the probability to sample a node in collision is high leading to more rejected in-collision nodes. We are then tempted to sample a high amount of nodes on the surface.

The algorithm quits contact mode after sampling the predefined number of nodes; then the position of the operator is checked. If he hasn't moved or if he stayed close to an obstacle, the algorithm enters back to contact mode.

C. Contact Algorithm Overview

The algorithm 2 presents our novel in-contact algorithm.

It uses as an input the contact point on the obstacle $\boldsymbol{P_o}$, the nearest point of the object $\boldsymbol{P_n}$, the configuration q_{device} which is the geometric center of the object's root body and the variable N describing the number of contact samples each time entering contact mode. This configuration is driven by a human operator who is able to see all the scene.

Algorithm 2 Contact Sampling

Require: $\boldsymbol{P_o}, \boldsymbol{P_n}, N, q_{device}$
1: $(\boldsymbol{\delta}, \boldsymbol{R}) \leftarrow$ Find_Local_Frame($\boldsymbol{P_o}, \boldsymbol{P_n}, q_{device}$)
2: **for** $i < N$ **do**
3: $\quad q_{current} \leftarrow$ Contact_Shooter($\boldsymbol{R}, \boldsymbol{\delta}$)
4: $\quad T \leftarrow$ Add_tree($q_{current}$)
5: **end for**

Line 1, a call to Find_Local_Frame function returns $\boldsymbol{\delta}$ the distance to contact and $\boldsymbol{R}$ the rotation matrix that is the transformation from the local to the global frame.

The function first finds the normal vector to the obstacle

$$\boldsymbol{n} = \frac{\boldsymbol{P_n} - \boldsymbol{P_o}}{||\boldsymbol{P_n} - \boldsymbol{P_o}||}.$$

The configuration given by the operator is the geometric center of the root body with $\boldsymbol{P_c}$ the 3D point attached to it. Therefore, $\boldsymbol{P_c} = q_{device}$. We project the vector $\boldsymbol{P_n}$ to $\boldsymbol{P_c}$ on $\boldsymbol{n}$ and compute $\boldsymbol{\delta}$ the projected distance to contact:

$$\boldsymbol{\delta} = \boldsymbol{n}^t.(\boldsymbol{P_n} - \boldsymbol{P_c})\boldsymbol{n}$$

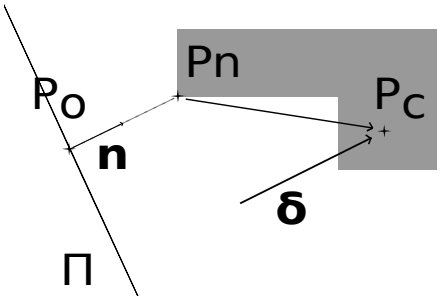

Fig. 2: 2D Projection example

We then find a rotation matrix between the world frame and the frame attached to $\boldsymbol{P_c}$. Using $\boldsymbol{n}$ and Gram-Schmidt process [19] we can generate a local frame of three orthonormalized vectors $(\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{n})$. These vectors give $\boldsymbol{R}$ the rotation matrix from the local to world frame.

$$\boldsymbol{R} = GramSchmidt(\boldsymbol{n})$$

Line 2 starts an iteration of N contact samples. They are computed line 3 calling the function Contact_Shooter(). Instead of randomly shooting every six dimensions like a random shooter would do, we position the rotation to follow the operator's order. Finally we randomly shoot the two last free dimensions and return the values in $q_{current}$.

The position coordinates $\boldsymbol{t}$ of $q_{current}$ are rotated using $\boldsymbol{R}$ and translated using $\boldsymbol{\delta}$ to stay on the contact plan. This transformation keeps shooting configurations on the plane Π passing through $\boldsymbol{P_c}$ and perpendicular to $\boldsymbol{n}$ inside the bounds of the workspace.

$$q_{current} = \boldsymbol{R}\boldsymbol{t} + \boldsymbol{\delta}$$

Line 4, the result is added to the tree for expansion.

D. Interactive Motion Planning with Contact

The real interest of our work and our main contribution is when both previous methods are used simultaneously. We can benefit both from an automated planner searching the whole space and the operator seeking to guide or slide the object.

We will show how contact samples can be generated on obstacle surfaces when an operator approaches them. We decided to let the operator define with his interactive device which surfaces should be sampled because he has the industrial knowledge. On the opposite, we chose to let the computer sample on surfaces because it is very difficult for an operator to be precise. This behavior allows the operator to decide manually when to start and end contact mode and on which surface. Previous contact algorithms (see section I) had costly contact solutions, opposite to our method who can generate many contact samples quickly. Algorithm 3 presents the two methods combined.

For each sample, a random number a defines who holds authority. If $a \leq \alpha$, authority is given to the computer and a random configuration is shot. If $a > \alpha$, authority is given to the operator. In this case, a distance test defines if contact mode should be enabled. If the test fails the new configuration is the one given by the interactive device ; if the test succeeds it means that the operator is very close to an obstacle, N contact samples are generated on its surface.

Line 2, we cycle through all elements in the workspace using Find_Nearest_Obstacle() function to measure the distance between them and the object. This function returns the pair of nearest points : $\boldsymbol{P_o}$ the nearest point on an obstacle to $\boldsymbol{P_n}$ the nearest point on the object.

Line 3, a random number between 0 and 1 is shot. If $a > \alpha$ line 4, control is given to the operator. Otherwise, line 11, the computer works alone.

Let d be a fixed threshold, line 5. If the distance between the object and the obstacle is higher than this parameter, the node $q_{current}$ added to the tree is set to $qdevice$, line 6.

If the distance is lower or equal to d, the planner switches to contact mode, line 9. The function ContactSampling() which is algorithm 2 is called.

Algorithm 3 Interactive RRT with Contact

Require: $W, T, N, q_{device}, \alpha, d$
1: **loop**
2: $(\boldsymbol{P_o}, \boldsymbol{P_n})$ = Find_Nearest_Obstacle(q_{device})
3: $a \leftarrow$ rand$(0, 1)$
4: **if** $a > \alpha$ **then**
5: **if** $|\boldsymbol{P_c} - \boldsymbol{P_n}| \geq d$ **then**
6: $q_{current} \leftarrow q_{device}$
7: $T \leftarrow$ Add_Tree($q_{current}$)
8: **else**
9: ContactSampling($\boldsymbol{P_o}, \boldsymbol{P_n}, N, q_{device}$)
10: **end if**
11: **else**
12: $q_{current} \leftarrow$ Random_Shooter()
13: $T \leftarrow$ Add_Tree($q_{current}$)
14: **end if**
15: **end loop**

E. Example

Here is described an example of our interactive motion planner on a surface. We have a white L shaped object that has to slide on a plane. Both nodes and edges are displayed.

The α parameter describing the authority sharing between the random shooter and the operator shooter is set to 1. This means that the only samples kept in the roadmap will be non colliding user defined configurations and valid contact configurations when in contact mode. This helps observing the behavior of our contact sampler on its own without the help of an standard RRT. When entering contact mode, the number N of configurations to be shot before switching back to normal mode is fixed to 10 for the rest of the paper.

We can see figure 3 two steps of motion planning with different orientations. Starting in position (1), we move the object to the contact and stay a while in this position to sample a lot of nodes all aligned (2). Please note that the nodes display the position of $\boldsymbol{P_c}$. Step (3) gets out of the contact, rotate the object (4) and get back to contact. A new set of points with the different orientation are generated (5).

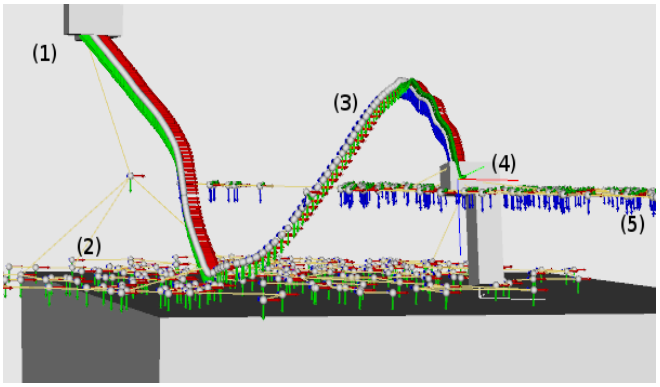

Fig. 3: Contact example

We have implemented a novel motion planner capable of generating configurations at the surface of obstacles with the help of an operator. Our choice was to let the operator impose the orientation because this can be important in some industrial cases such as insertion but there may be other strategies. We illustrate in the following section the usefulness of our contact algorithm with two examples.

IV. IMPLEMENTATION AND RESULTS

The following section describes the performances of our algorithms and their implementation details. In the following examples only edges of the roadmap will be displayed for clearer visualization.

Part A givess important elements regarding implementation. Part B shows the usefulness of our interactive planner with an illustrating example. Last, in part C, we test both our interactive and contact planner in a cluttered environment.

A. Implementation details

The interactive device is a 6D mouse from Immersion, model 3DConnexion. All experiments are performed with Hpp [18] framework developed primarily by the Gepetto team LAAS-CNRS, the collision detection library is Fcl [20], the geometric models are all described using URDF (Unified Robot Description Format) and the 6D mouse driver is our own. Our computer has an Intel®Xeon®CPU E3-1240 v3 @ 3.40GHz processor with 16 GB RAM and runs under Ubuntu 14.04.

A particular attention was given to separate work on two different threads. The goal is to let the standard motion planning process alone on its thread and therefore on its processor. All treatments regarding the operator are treated separately on a different thread. As no parallelization of motion planning is implemented we can easily compare our solution with a standard, single processor motion planning implementation.

Treatments regarding the operator are done in the operator thread. They are: moving the object, reading the interactive device data and integrating positions. The same goes for cycling through obstacles to find the nearest one to the object. Again, computing $\boldsymbol{R}$ the rotation matrix and $\boldsymbol{\delta}$ the object to contact vector is done on the operator thread. Visualization of the scene is a separate process.

The overhead slowing the planner thread is during contact mode because each new configuration is rotated by $\boldsymbol{R}$ and translated by $\boldsymbol{\delta}$. This means that for every contact configuration the computational overhead is one 3x3 matrix multiplication with a vector and three additions as well as changing the value of $q_{current}$ and some test instructions.

B. Maze Example : interactive algorithm

This example shows the behavior of algorithm 1. A maze has to be crossed by a non convex 3D object that is small compared to the dimensions of the walls, see figure 4. No samples can be shot beyond the walls height to forbid shortcuts.

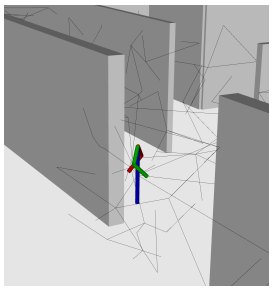

Fig. 4: Maze Object.

In figure 5 a standalone RRT-connect solves the problem in 3 minutes with 3690 nodes and 7378 edges with $\alpha = 0.5$. Figure 6 shows the result of the same problem solved with the help of a user which has much better results, solving the problem in one minute with 1453 nodes and 2904 edges.

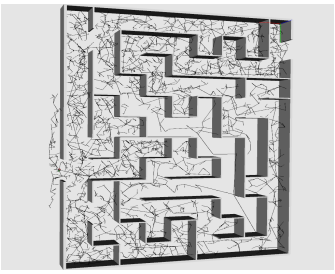

Fig. 5: RRT-connect

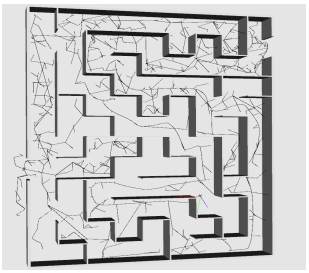

Fig. 6: Interactive

C. Cluttered Environment : contact algorithm

This section presents a cluttered environment that is hard to solve using simple RRT method. We will show that our interactive planner outperforms it radically. Using our contact method (algorithm 3), we are able to improve the efficiency planning processes. We then discuss the influence of authority sharing.

1) Environment: In this experiment, the object is a non-convex L-shape. The environment figure 7 consists of two blue planes forming a cluttered space as the object can rotate only in a few directions. Two red planes form an even more cluttered space where the object can freely rotate only around one axis. Small rotations around other axis, though, are still possible before getting in collision.

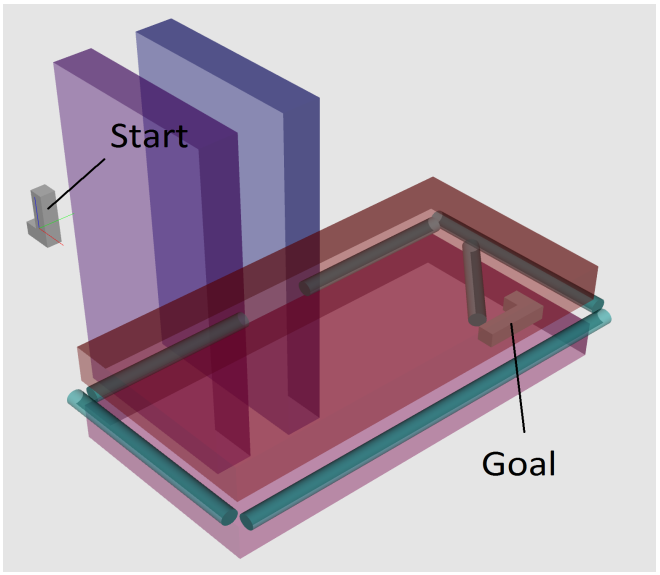

Fig. 7: Cluttered Space

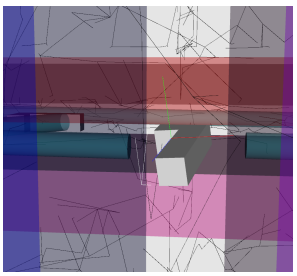

Fig. 8: Narrow passage, side

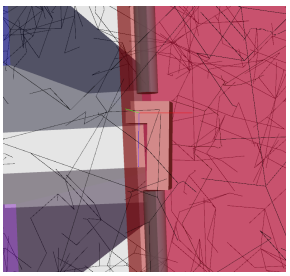

Fig. 9: Narrow passage, up

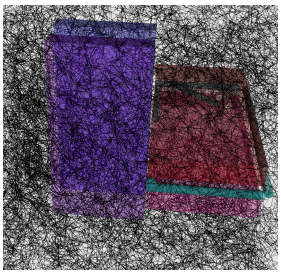

Fig. 10: Simple RRT

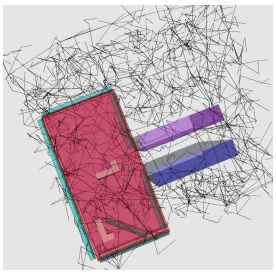

Fig. 11: Interactive

All passages to get inside the red area are blocked with turquoise bars except in a narrow passage shown figures 8 and 9. The distance between the two first walls is 1.2 meter while the distance between the two next walls is 0.75 meter. The length of the object is 1.6 meter, its height 0.8 meter and its width 0.4 meter. This means that the object can pass through the narrow passage with only small variations around roll and pitch axis.

The goal configuration is at a corner of the red area behind an oblique bar therefore the object will have to slide in order to reach the goal.

2) Free space tests: The first experiment shown figure 10 is a simple RRT. It lasted for 2h45 minutes before finding a solution. Were added 27 600 nodes and 55 198 edges to the

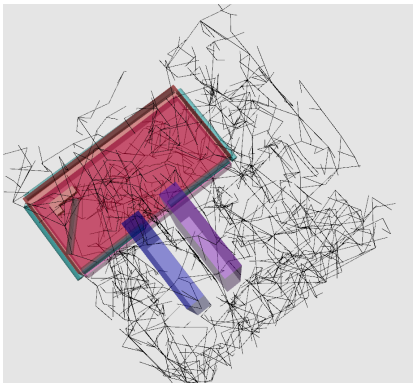

Fig. 12: Contact $\alpha = 0.8$

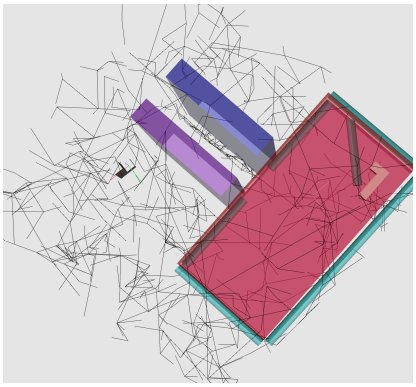

Fig. 13: Contact $\alpha = 0.2$

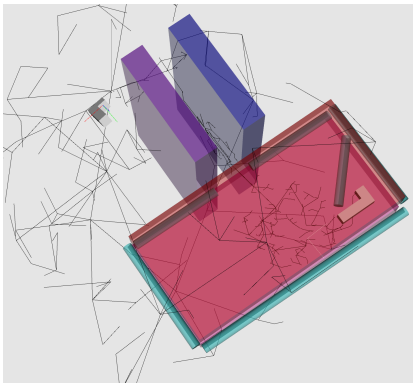

Fig. 14: Contact $\alpha = 0.05$

roadmap . In industrial cases, this time length is not realistic and unacceptable.

The second experiment figure 11 is held with the help of an experienced user 6D mouse with our interactive algorithm 1. Cooperation factor α is set to 0.5. This experiments lasts for 3 minutes with a roadmap holding 3424 nodes and 6846 edges. As expected, the help of an operator radically changed the speed of the process. Rapidly, when samples lay in an interesting space, random samples also tend to be generated in this space, we can see that the amount of nodes sampled is very small in comparison to the RRT method.

3) Contact tests: The performance of our contact algorithm 2 is tested with a series of experiments by changing the value of α. Figures 12, 13 and 14 show three results with different values of α.

Whenever the operator approaches the object to an obstacle, samples are generated along a tangent plane to it. The tree grows fast on surfaces. To get inside the red area near the goal configuration, the object should slide along one plane but this requires freezing some degrees of freedom. While this can be a challenge for a random shooter, our in-contact shooter solves the problem much quicker when we want to stay in contact with the obstacle.

D. Results

The following table presents the results of experiments in the cluttered environment. RRT standalone planner is clearly much slower than any other solution. Comparing interactive mode without contact (algorithm 1) to interactive and contact sampling (algorithm 2), with the same value of α (=0.5), the time and number of nodes decreases significantly.

It is necessary to analyze the contact algorithm according to alpha parameter. With a control sharing factor α getting smaller, the algorithm keeps growing in performance until a minimal point is reached with α = 0.08. This means that 92% of the time is given to the operator. The fact is that this time is not completely given to the operator because when the user move the objet close to contact, the contact mode (Contact Sampling) automatically expands nodes on the surface. We have three operating modes:

- one mode by user when objet is far from obstacle,
- one by algorithm to explore region randomly,
- one by user to guide exploration of contact region

The parameter α must be very small because otherwise too much nodes and edges are added by the computer in uninteresting space. Values of α smaller than 0.08 loose the benefit from random sampling and the obtained results get less competitive. Either way, our in contact algorithm is always much faster, requiring much less nodes and edges than the interactive planner without any contact sampling.

Scenario	Time	Nodes	Edges
RRT, $\alpha = 1$	2h45m	27 600	55 198
Interactive, $\alpha = 0.5$	3m	3 424	6 846
Contact, $\alpha = 0.8$	40s	1 729	3 454
Contact, $\alpha = 0.5$	31s	1 314	2 626
Contact, $\alpha = 0.2$	21s	715	1 428
Contact, $\alpha = 0.1$	22s	679	1 356
Contact, $\alpha = 0.08$	15s	538	1 074
Contact, $\alpha = 0.05$	24s	547	1 092
Contact, $\alpha = 0.04$	21s	602	1 202
Contact, $\alpha = 0.02$	31s	658	1 314
Contact, $\alpha = 0.01$	25s	768	1 534
Contact, $\alpha = 0$	47s	894	1 786

TABLE I: Cluttered environment results

V. CONCLUSION

Our interactive contact algorithm make a step forward in assembly path planning. It is a twofold contribution that helps solving cluttered situations where objects need to slide on each other by keeping contact. Some cluttered environments or insertion cases could benefit from this novel algorithm. We have shown the influence of authority sharing on the results of path planning.

The main drawback of our method is that contact sampling cannot follow multiple contact but stays only on one plane at a time. So far we imposed the contact sampling orientation to be chosen by the user but another strategy could be implemented for different test cases.

Future work would be to implement an algorithm capable of following iteratively many different planes and enabling

change of orientation. We would also like to integrate the user in a Virtual Reality platform with a haptic arm to get force feedback to simplify the work of an operator in complex 3D environments.

REFERENCES

[1] R. Iacob, D. Popescu, and P. Mitrouchev, "Assembly/disassembly analysis and modeling techniques: A review," *Strojniški vestnik-Journal of Mechanical Engineering*, vol. 58, no. 11, pp. 653–664, 2012.

[2] M. Bordegoni, U. Cugini, P. Belluco, and M. Aliverti, "Evaluation of a haptic-based interaction system for virtual manual assembly," in *Virtual and Mixed Reality*. Springer, 2009, pp. 303–312.

[3] L. Tching, G. Dumont, and J. Perret, "Interactive simulation of cad models assemblies using virtual constraint guidance," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 4, no. 2, pp. 95–102, 2010.

[4] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, 1996.

[5] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[6] S. M. LaValle, *Planning algorithms*. Cambridge Univ. press, 2006.

[7] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[8] K. Naderi, J. Rajamäki, and P. Hämäläinen, "Rt-rrt: a real-time path planning algorithm based on rrt," in *Proceedings of the 8th ACM SIGGRAPH Conf. on Motion in Games*. ACM, 2015, pp. 113–118.

[9] O. B. Bayazit, G. Song, and N. M. Amato, "Enhancing randomized motion planners: Exploring with haptic hints," *Autonomous Robots*, vol. 10, no. 2, pp. 163–174, 2001.

[10] J. Rosell, C. Vázquez, A. Pérez, and P. Iñiguez, "Motion planning for haptic guidance," *Journal of Intelligent and Robotic Systems*, vol. 53, no. 3, pp. 223–245, 2008.

[11] X. He and Y. Chen, "Haptic-aided robot path planning based on virtual tele-operation," *Robotics and computer-integrated manufacturing*, vol. 25, no. 4, pp. 792–803, 2009.

[12] N. Ladeveze, J. Y. Fourquet, B. Puel, and M. Taix, "Haptic assembly and disassembly task assistance using interactive path planning," in *Virtual Reality Conf., 2009. VR 2009. IEEE*. IEEE, 2009, pp. 19–25.

[13] D. Flavigné, M. Taï, and E. Ferré, "Interactive motion planning for assembly tasks," in *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*. IEEE, 2009, pp. 430–435.

[14] S. Cailhol, P. Fillatreau, J.-Y. Fourquet, and Y. Zhao, "A hierarchic approach for path planning in virtual reality," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 9, no. 4, pp. 291–302, 2015.

[15] S. Redon and M. C. Lin, "Practical local planning in the contact space," in *Robotics and Automation, 2005. ICRA 2005. Proc. of the 2005 IEEE Int. Conf. on*. IEEE, 2005, pp. 4200–4205.

[16] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 895–900.

[17] Y. Yan, E. Poirson, and F. Bennis, "Integrating user to minimize assembly path planning time in plm," in *Product Lifecycle Management for Society*. Springer, 2013, pp. 471–480.

[18] F. Lamiraux and J. Mirabel, "Hpp: a new software framework for manipulation planning," 2015.

[19] E. Schmidt, "Über die auflösung linearer gleichungen mit unendlich vielen unbekannten," *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, vol. 25, no. 1, pp. 53–77, 1908.

[20] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3859–3866.