The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Improving Motion Planning in Weakly Connected Configuration Spaces

David Flavigné, Michel Taïx

Abstract— **Even if the probabilistic motion planner methods (PRM or RRT) have been successful for robot path planning, it remains a challenge in a constrained cases with narrow passages. The RRT is a powerful tool for a simple request, but the performances of approaches falls sharply when the search has several narrow passages. The introduction of several trees can reduce this problem, but has the disadvantage of requiring the control of the number and the growth of these trees. However, this can be done using the properties of the Visibility-PRM. Combining the ideas of the Visibility-PRM with a multi-RRT (local trees), a new algorithm is presented and experimental results show the importance and effectiveness of the method.**

I. Introduction

Path planning has been an important problem in robotics for many years [11]. During the past decade the path planning problem was widely extended in areas such as computer animation, virtual prototyping and also in drug-design. With the recent results in random planning algorithms [13] it is possible to automatically solve problems for systems with a large number of degrees of freedom. The algorithm computes a collision free roadmap in configuration space (*CS*) where the object is reduced to a point. This point represents the robot's model in the environment. The algorithm searches then a free path for a point in the roadmap.

One problem of the random planning approach is the computation time in high dimensional space with complex workspace. The dimension of *CS* is equal to the number of degrees of freedom of the mechanical system and the computation time is growing with the dimension of *CS*. The random approach is less time-sensitive to the dimension of *CS* than a classical approach, and is a function of the number of objects in the workspace (collision tests) and the number of narrow passages in *CS* (connexity).

There are mainly two families of methods for building a roadmap, the Probabilistic Roadmap (PRM) [10] and the Rapidly-exploring Random Tree (RRT) [14].

The PRM [10] is a typical multiple queries method. The main idea is to randomly sample configurations in *CS*, keep only the free configurations and try to connect the sampled configurations by a free local path. The result is a global roadmap that captures the connectivity of the free *CS*.

RRT approach [14], originally proposed by Lavalle, is more interesting in our case because it is faster in the simple query case, when the roadmap computation must be limited in time. An interesting variant is the RRT algorithm [14] which uses bidirectional expansion strategy between two trees (one rooted at the start and the other at the goal).

Even if these approaches work well in a big variety of examples ([5], [8], [17],...), they do not allow to resolve all the cases. The typical example is the case of an object that has to pass through a series of narrow passages. The difficulty for the algorithm is to find the passage entrance. To do so it will have to investigate randomly all the *CS* dimensions. On the other hand, when the algorithm will find the narrow passage entrance in the *CS*, it will quickly progress to find a solution with RRT approach.

Many works present variants of PRM or RRT methods to improve the algorithm performance : to find narrow passages [2], [1], [21], [20], [24], to reduce the number of nodes [19], or the number of expensive iterations [23], [6], to guide expansion toward interesting regions [3], [4], [22], [16] or combined methods [15] or sampling methods [18], [9]. Generally, these methods allow to resolve difficult cases for a special class of problem (for example, solution trajectory close to the contact). The main contribution of the paper is to use properties of the Visibility PRM method (VISPRM) [19] to control the Local Trees method (LTRRT) [20]. The VISPRM reduces the number of nodes in the roadmap, but also have information on the cover of free space to stop the algorithm. Using this approach for the management of local trees can help to overcome the narrow passages problem and especially in cases difficult for VISPRM. The goal is to develop an algorithm which works well in the case of large free spaces separated by narrow passages (called weakly connected configuration spaces in this article).

The paper is organized as follows. The related works which are the basis of our work are summarized in section II. Section III gives an overview of our planner. The motion planning algorithm Visibility Local Tree (VISLT) is presented in section IV. Finally, simulation results are presented in section V. Section VI summarizes the main results.

II. Related work

Starting at a given initial configuration, RRT [12] incrementally searches 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. By definition the RRT tends to rapidly grow in the unexplored regions of the *CS*. But with narrow passages the cost of RRT increases and the cost gap between RRT and PRM decreases.

David Flavigné and Michel Taïx are with CNRS; LAAS ; 7, av. du Colonel Roche, 31077 Toulouse, France, Université de Toulouse; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France dflavigne@laas.fr, taix@laas.fr

[20] introduces LTRRT to augment the number of samples in difficult regions. A LTRRT is an RRT with more than two trees and by growing them, it can go out of difficult regions. The idea is illustrated in figure 1. If we can develop a local tree $T(q_2)$(root at q_2) in the middle region (R2), the probability to connect $T(q_1)$ and $T(q_2)$ or $T(q_2)$ and $T(q_3)$ will be bigger than the probability to connect directly $T(q_1)$ to $T(q_3)$.

This approach is a compromise between a pure RRT (only two trees are diffusing), and a PRM (every node introduced into the roadmap can be merged with a connected component, or create a new one, but cannot be extended).

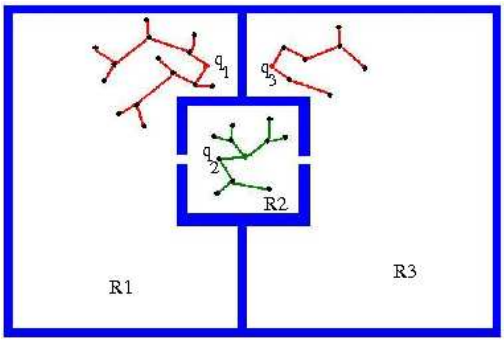

Fig. 1. An example of LTRRT. The method has a high probability of generating nodes in the three regions R_i. The result will be a spread of three trees in the three regions. The probability of connecting two trees will be higher if there is a tree in the region R_2 that if there is only the two initial trees in regions R_1 and R_3.

To increase performances it is necessary to solve several questions about LTRRT : when to create a new tree ? When to allow it to grow ? When try to merge with another tree ? when to stop the random process ? In [20], the author answers the first three questions by using heuristics. Its method is interesting, but when the number of local trees increases, the computation time also increases rapidly because the number of nodes in every tree grows and the number of feasible connections between local trees also grows. An idea to improve the behavior of the algorithm is to limit the number of nodes and if possible the number of trees.

This idea is the basis of an original variant of PRM that uses the notion of visibility between nodes to produce small roadmaps called Visibility Roadmaps [19]. The author defines two types of nodes : "guard" for the coverage of free *CS* and "connector" for creating connection. The algorithm keeps only configurations which either increase the visibility of the free *CS* (guard node, not visible by other guard nodes) or connect two components of the roadmap (connection node). The result is a roadmap with a small number of nodes and the control of the end of the roadmap construction is done by estimation of the *CS* coverage.

But this method has difficulties to find narrow passages, the main problem is illustrated on the figure 2. The probability to sample a node connecting two components of the roadmap is low when generated guards visibility domains have a very small intersection.

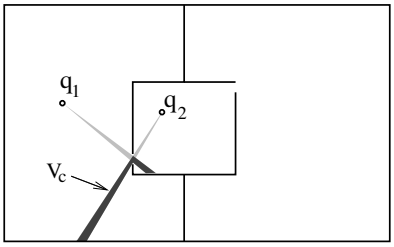

Fig. 2. A difficult example to create a connector node in VISPRM. A connector node can be created only if it is sampled in the visibility region (V_C) common to q_1 and q_2. The probability to sample a configuration in the region V_C (in dark gray) is low.

The main idea of the VISLT algorithm is to combine VISPRM and LTRRT strategies in order to overcome drawbacks on both sides.

III. Overview of the planner

As said earlier, the LTRRT method has three points to resolve in order to be really efficient : where and when to start a local tree, when to grow it and when to stop it. The three of them can be answered at once with the VISPRM strategy. Local trees are made for exploring unreachable areas (neither from the start nor the goal tree), typically "rooms" (local spaces) separated from the remaining CS by narrow passages, and to connect them more easily. So, local trees must start in those unreachable areas. But how to define them? Guard nodes from the VISPRM can be roots for new local trees. For trees growth, we decided to add

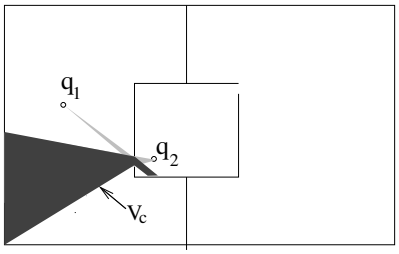

Fig. 3. Compare this figure to figure 2 for V_C sampling domain.

nodes only when useful, to keep their number as small as possible. For this, Visibility Roadmap is the best, but on the other hand narrow passages dramatically reduce the visibility domain common to the trees surrounding the passages. These ones are then difficult to connect together. They are all the more difficult to connect that the tree nodes are far from the passage. On the contrary, the closer nodes are from the passage, the larger is the common visibility domain (see figure 3). From this observation, we decide to add a third type of node : "scout nodes" to get closer to narrow passages. This is done by keeping only nodes that move away from roots. Indeed, if roots are already near a narrow passage, connections are made pretty fast as the common visibility domain is larger, and scout nodes are not added.

The last point is when to stop trees growth. In the VISPRM, roadmap growth is stopped after a certain amount

of time or number of node, but it can be stopped as soon as the number of nodes is not growing although the planner keeps sampling. In this case, we can say that almost the whole free space is covered by the roadmap. In the VISLT method, this is the same because trees will stop growing as soon as they will reach borders (Scout nodes can't go further).

The VISLT method resulting from the integration of these three points is a RRT-based method combining both strategies, each one overcoming the drawbacks of the other.

IV. Visibility Local Tree Algorithm

The VISLT algorithm is shown in Algorithm 1 and is structured in three parts for each step :

First, the sampling part, which is standard in RRT and PRM methods. The sampling method gives us a new random configuration q_{rand}. For implementing the VISLT method we choose a simple uniform sampling method through the CS.

Then, for each tree T_i (or connected component) in the roadmap, we try to find out if we can connect them to the newly sampled configuration q_{rand}, without connecting them actually. We keep the number of connectable trees (*nbPossibleConnection*). Instead of testing each node of each connected component for connection, we choose to test only the nearest node of the sampled configuration.

Finally, depending on *nbPossibleConnection* we can distinguish three cases :

- If there is no possible connection, q_{rand} is added to the roadmap as a guard node and the root of a new local tree.

- If there is only one possible connection, we compute the distance between q_{rand} and the root of the current tree T_c that can be connected. If this distance (DIST_ROOT) is greater than the distance from node in T_n that has to be connected to the root (q_{near_c}), then q_{rand} is added to the roadmap (Scout node) and the connection is made. The computed distance is stored as a label in the node structure, to save time for later iterations. In this case the local tree grows, and then its visibility domain is augmented.

- Otherwise, we add q_{rand} in the roadmap as a connector node and make connection for each connectable tree. These trees are then merged into one and a new root is chosen (for each node that was connected, we recompute the distance to the new root, when their relative root has changed). The NEAREST_NEIGHBOR is computed only once, we memorize the result and use it during this step.

V. Results

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 of ram. All the reported values on the tables are average over 100 runs.

We compare our VISLT with basic RRT, LTRRT and VISPRM. The basic functions (sampling method, collision checking, nearest neighbour) which have a big influence on the results are identical in the various algorithms. If we change one or some of these functions, we can improve the performances of the algorithms, but the objective of this paper is not to find the *best choice* in a precise case of environment but to be able to compare the various algorithms between them.

[1]KineoWorks is the path planning dedicated Software Developed Kit developed by Kineo CAM.

Algorithm 1 The Visibility - Local Trees algorithm

```
F ← T_{q_init}, T_{q_goal}
Guards ← ∅
nbPossibleConnection = 0
for k = 0 to N do
  q_rand ← RANDOM_CONFIG()
  for All Trees T_i in F do
    q_near_i ← NEAREST_NEIGHBOR(q_rand, T_i)
    if CAN_CONNECT(T_i, q_rand,q_near_i) then
      nbPossibleConnection++
    end if
  end for
  if nbPossibleConnection = 0 then
    R ← q_rand
    Guards ← q_rand
    T_new = ADD_NEW_TREE(q_rand)
    F ← F ∪ T_new
  else
    if nbPossibleConnection = 1 then
      if DIST_ROOT(q_rand,T_c)> DIST_ROOT(q_near_c,T_c)
      then
        Add_Vertex(T_c, q_rand)
        Add_Edge(T_c, q_near_c,q_rand)
      end if
    end if
  else
    for All Connectable Trees T_i in F do
      q_near_i ← NEAREST_NEIGHBOR(q_rand, T_i)
      Add_Vertex(T_i, q_rand)
      Add_Edge(T_i, q_near_i,q_rand)
    end for
  end if
end for
```

A. Simple and Double Room

The first tests are made for a 2-dimensional workspace (see figure 4) with a 3-dimensional *CS* (X,Y,θ). The CS dimension is the same in the two figures, but the workspace is more difficult. The main difference between these two examples is the number of narrow passages, increasing the problem difficulty.

To take into account the narrow passage difficulty we take a parameter K which corresponds to :

$K = \frac{\texttt{width of the narrow passage}}{\texttt{width of robot}}$.

The lower is K the higher is the difficulty to find the entrance of the narrow passage.

Table I summarize the mean values for tests made for the four algorithms (complete results in [7]) for the double room case.

The first objective is to verify the interest of the approach. We can see that the number of nodes in the roadmap is lower in VISLT than in RRT and we can see in table I that in all cases the VISLT gives the best results.

In [7] we can notice that computation time as well as number of nodes are bigger in the double room case although the *CS* dimension is only three. We also compute the same case for the robot only in translation (*CS* dimension = 2) for the simple room (we don't show this too simple case). The performance gap between these two cases is smaller than the one between the simple room and the double room cases. This confirms that the main difficulty is the number of narrow passages more than the dimension of *CS*.

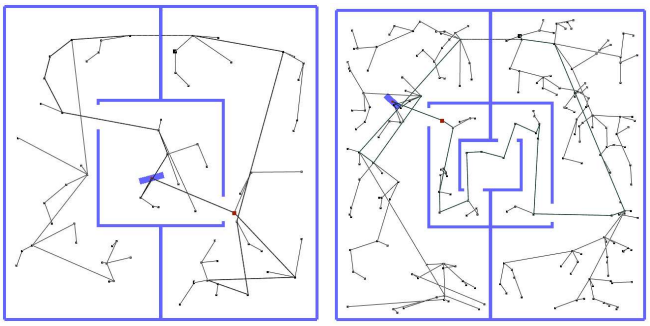

Fig. 4. VISLT for a simple (left) or a for a double room (right).

TABLE I

Case	Algorithm	Iterations	Nodes	Time
(K=3)	RRT	20426	8029	925 s
	LTRRT	3053	1297	292 s
	VISPRM	2539	38	368 s
	VISLT	1396	196	88 s
(K=2.5)	RRT	35246	13631	1916 s
	LTRRT	6105	2405	460 s
	VISPRM	11081	106	1728 s
	VISLT	2543	284	152 s
(K=2)	RRT	77590	29418	5182 s
	LTRRT	16931	6154	1433 s
	VISPRM	46215	316	7380 s
	VISLT	7639	557	410 s
(K=1.5)	RRT	190310	70098	18741 s
	LTRRT	74787	23490	7760 s
	VISPRM	110220	704	18429 s
	VISLT	33149	1363	2027 s

B. The Walls

In this case (figure 5), the *CS* is six-dimensional, as the robot is a free flyer. The problem is to go through the walls by the holes. The dimensions of each wall are 100x100x20, and each hole and 20x20. The dimensions of the robot are 5x5x25. We tested the algorithms on four environments, which differs on their number of narrow passages (walls) : 2, 4, 6 and 8 walls. We can see on table II that the VISLT solved the problem in less time and with less iterations than other algorithms. The number of added nodes in the roadmap is also shorter in the VISLT roadmap than in the LTRRT one.

In this case, the RRT is omitted because the execution is too slow and has no interest in the comparison.

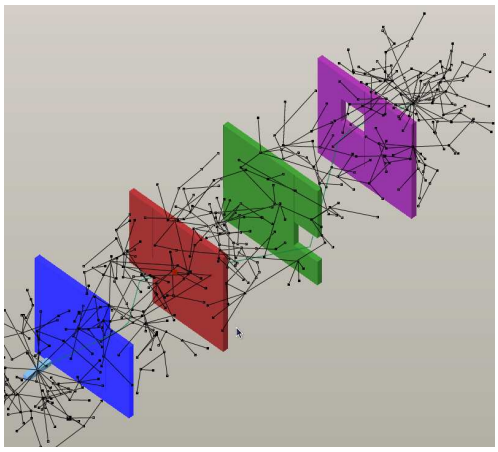

Fig. 5. The environment with four narrow passages.

TABLE II

2 Walls 4p0	Algorithm	Iterations	Nodes	Time
	LTRRT	3880.7	1972.7	328.95s
	VISPRM	3820.2	7.25	231.6s
	VISLT	1294	383.65	81.3s
4 Walls 4p0	LTRRT	18273	7715.9	1755.1s
	VISPRM	11954	18.65	1228.7s
	VISLT	8288.5	1402.8	552.1s
6 Walls 4p0	LTRRT	31972	11920	3329.2s
	VISPRM	36359	116.45	5553.9s
	VISLT	11771	1756.5	894.35s
8 Walls 4p0	LTRRT	51016	15218	5357.3s
	VISPRM	40606	162.17	9513.7s
	VISLT	22084	2634.2	2005.7s

VI. Discussion about Properties of Visibility Local Tree

A. Decreasing number of nodes

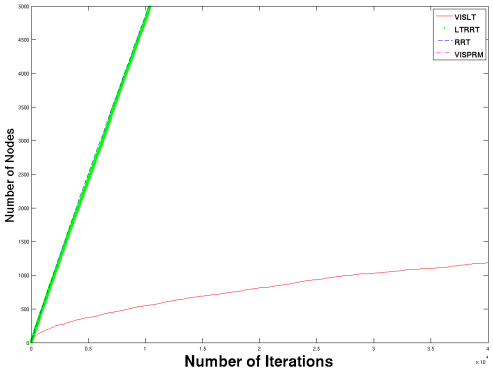

Fig. 6. Expansion nodes.

To show the interest of our method we consider the example of a start and a goal in two unconnected rooms, which has no solution. The algorithms kept running until they had reached the time limit (10000 seconds). Figure 6 shows the number of nodes created for each algorithm against the number of iterations. We can see that the number of nodes increases constantly for the LTRRT and RRT, while

the VISLT reduces the number of added nodes gradually as the number of iterations is growing. This result shows that the node creation in the VISLT is made only when useful. A consequence of this node selection is a reduction of the number of iterations needed to find a solution. The VISPRM algorithm didn't create any node (only the initial and goal nodes).

B. Performance given the difficulty

Figure 7 shows the increasing difficulty, due to narrow passages decreasing size in the double room case with a three dimensional *CS*. On each one, we can see that the VISLT algorithm has the best performances. This mean that it is less time-consuming than basic RRT and LTRRT, and that the number of iterations as well as the number of nodes needed to find a path are lesser for the VISLT.

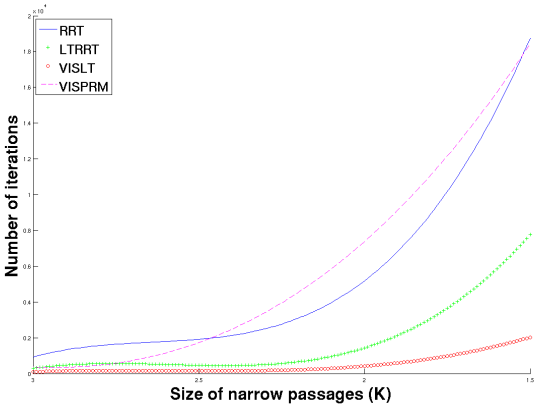

Fig. 7. Influence of the narrow passage size in the double room case.

If we look the slope of curves represented on figure 7, we can see that even if the VISLT gives the best result, its times complexity grows in a similar way with the other algorithms.

We have the same result with the simple room case [7], but here we can notice that the slope is slightly weaker in the case of the VISLT. It catches more easily narrow passages and seems more adapted for motion planning in the case of a configuration space with a succession of narrow passages connecting parts of free space.

C. Performance given the number of weakly connected areas

After a comparison given the size, it can be interesting to compare the influence of the number of narrow passages. This number reflects the connexity of the environment. On figure 8, we can see the performances of the VISLT, compared to VISPRM and LTRRT. The control of the created local trees makes the other algorithms greedier than VISLT (in number of node for LTRRT and in time for VISPRM).

D. Roadmap connexity evolution

VISLT, LTRRT and VISPRM are algorithms that all implies several trees for building their roadmap and solving a problem. On figure 9, we can see the evolution of the trees for each algorithm during a run. The tests were done on

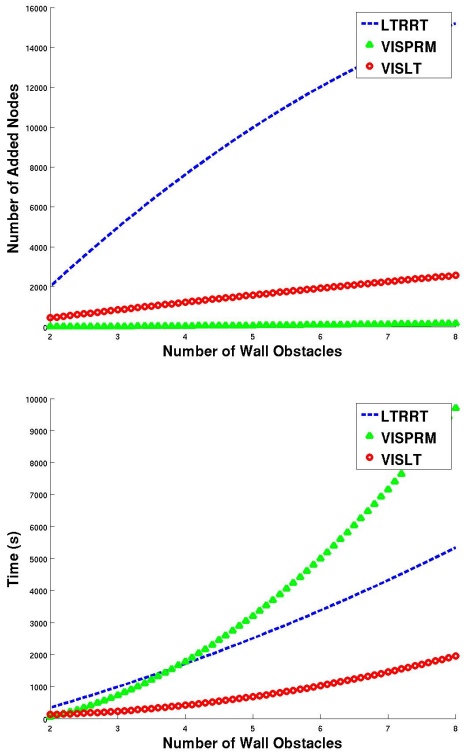

Fig. 8. Performances given the number of narrow passages (number of walls) in the environment.

a wall-type environment (like figure 5) with 8 walls. Each run was stopped after 20000 iterations, and the curves are averaged over 10 runs. This figure shows that the VISLT creates less trees, and is able to connect them together faster than both VISPRM an LTRRT (on the first iterations, the red curve decreases faster than the blue and green ones).

The curves noise means that trees are created and then linked to another one after few iterations. These transient trees are located in small areas that are not really diffcult to access but cannot be directly linked to a component due to obstacle proximity (near the walls for example). In our examples, this is mostly due to rotations and to the linear steering method used to link two configurations.

On figure 9, the VISPRM curve is less noisy than the other curves because it has difficulties to connect newly created trees between them. On the contrary, VISLT and LTRRT, creates new trees and connect them more easily to the existing ones (thus making their curve noisy). But in the LTRRT case, more nodes are added before a connection and before a connected component creation. This explains why the VISLT number of trees is always lower than LTRRT.

We can also see that the VISLT curve is noisier in the beginning than in the end. This proves the exploration capacity of VISLT. Most of the small areas are found and quickly connected to the roadmap, as well as wider areas, reducing the number of created trees. The VISPRM creates components in large easy-to-find areas, but then has more difficulties to find the small areas, explaining why its curve

is less noisy at the end. Finally, the LTRRT is as noisy at the beginning than at the end. This is due to the large number of nodes added to the roadmap, and to its density.

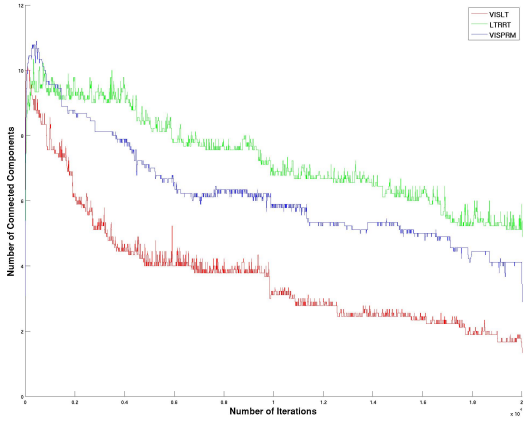

Fig. 9. Number of trees evolution during one run.

E. Local Trees questions answered

In parts II and III, we spoke about several points to solve in the LTRRT algorithm. These concerned the creation, the growth and the end of growth of a tree. We saw on the previous examples that all of them were solved by the VISLT. The creation problem is solved by the PRM part of the algorithm : trees are created only when there's no possible connection with the roadmap, that is to say when a guard node is created. The component is growing until the bounds of the local environment (area that is visible by the connected component) are reached. Then the number of added nodes decreases for this component, thus favouring large and fast exploration to dense exploration (unlike RRT and LTRRT). The growth is almost stopped (below a given threshold of added nodes, the growth can be explicitly stopped) when the local area is filled with the connected component, solving the third LTRRT problem. Another way to "stop" the growth for a tree is to merge it with another one, when a connector node is found. The connected trees are then growing as a single one.

VII. CONCLUSIONS AND FUTURE WORKS

In this article, we described a motion planner combining advantages of a Local Trees RRT and a Visibility PRM. This method is an RRT-based method using Local Trees to accelerate exploration and going through narrow passages more easily and to limit the number of nodes with a constraint of visibility, maximizing thus exploration. This algorithm works well in configuration spaces with succession of large free spaces and narrow passages. The Visibility Local Trees algorithm is a good trade-off between roadmap density, which slows down the resolution of a problem, and a tiny roadmap which needs too much time (for a single query problem) to represent the environment connectivity. It would be interesting to study more in details how are scout nodes added, thus allowing to tune on-line the connected component density for finding really narrow passages faster.

VIII. ACKNOWLEDGMENTS

This work was partially supported by AMSI-ANR project.

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*, 1998.
[2] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *IEEE Int. Conference on Robotics & Automation*, 1999.
[3] B. Burns and O. Brock. Single-query entropy-guided path planning. In *IEEE Int. Conference on Robotics and Automation*, 2005.
[4] B. Burns and O. Brock. Toward optimal configuration space sampling. In *of Robotics: Science and Systems*, 2005.
[5] J. Cortés and T. Siméon. Sampling-based motion planning under kinematic loop-closure constraints. In *Int. Workshop on Algorithmic Foundations of Robotics*, 2004.
[6] S. Dalibard and J. Laumond. Control of probabilistic diffusion in motion planning. In *Workshop on Algorithmic Foundations of Robotics*, 2008.
[7] D. Flavigné and M. Taïx. Visibility local tree for motion planning. Technical Report 09355, LAAS-CNRS, june 2009.
[8] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*. A.K. Peters, Wellesley, MA, 2001.
[9] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *IEEE Int. Conf. on Robotics & Automation*, 2005.
[10] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics & Automation*, 12(4), June 1996.
[11] J.-C. Latombe. *Robot Motion Planning*. Kluwer, Boston, MA, 1991.
[12] 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.
[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.
[14] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Workshop on the Algorithmic Foundations of Robotics (WAFR'00)*, 2000.
[15] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Trans. on Robotics*, 21(4), 2005.
[16] M. Rickert, O. Brock, and A. Knoll. Balancing exploration and exploitation in motion planning. In *of the IEEE International Conference on Robotics and Automation*, 2008.
[17] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *International Journal of Robotics Research*, 17, 1998.
[18] X. T. Shawna Thomas, Marco Morales and N. M. Amato. Biasing samplers to improve motion planning performance. In *IEEE Int. Conference on Robotics & Automation*, Kobe, Japan, may 2007.
[19] T. Siméon, J.-P. Laumond, and C. Nissoux. Visibility based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6), 2000.
[20] M. Strandberg. Augmenting RRT-planners with local trees. In *IEEE International Conference on Robotics & Automation*, 2004.
[21] 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), 2005.
[22] C. Urmson and R. Simmons. Approaches for heuristically biasing RRT growth. In *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, 2003.
[23] A. Yershova, L. Jaillet, T. Siméon, and S. M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *IEEE Int. Conference on Robotics & Automation*, 2005.
[24] L. Zhang and D. Manocha. An efficient retraction-based RRT planner. In *IEEE Int. Conference on Robotics and Automation*, 2008.