

Rapidly-exploring Sorted Random Tree, a self adaptive random motion planning algorithm

Nicolas Jouandeau

Université Paris 8, LIASD,
2, rue de la Liberté,
93526 Saint-Denis Cedex - France,
n@ai.univ-paris8.fr,

WWW home page: <http://www.ai.univ-paris8.fr/~n/>

Abstract. We present a novel single shot random algorithm, named *RSRT*, for *Rapidly-exploring Sorted Random Tree* and based on inherent relations analysis between *RRT* components. Experimental results are realized with a wide set of path planning problems involving a free flying object in a static environment. The results show that our *RSRT* algorithm is faster than existing ones. These results can also stand as a starting point of a massive motion planning benchmark.

1 Motion planning with Rapidly Exploring Random Trees

The problem of motion planning turns out to be solved only by high computational systems due to its inherent complexity [1]. As the main goal of the discipline is to develop practical and efficient solvers that produce automatically motions, random sampling searches successfully reduce the deterministic-polynomial complexity of the resolution [2]. In compensation, the resolution that consists in exploring the space, produce non-determinist solution [3]. Principal alternatives of this search are realized in configuration space C [4], in state space X [5] and in state-time space ST [6]. C is intended to motion planning in static environments. X adds differential constraints. ST adds the possibility of a dynamic environment. The concept of high-dimensional configuration spaces is initiated by J. Barraquand *et al.* [7] to use a manipulator with 31 degrees of freedom. P. Cheng [8] uses these methods with a 12 dimensional state space involving rotating rigid objects in 3D space. S. M. LaValle [9] presents such a space with a hundred dimensions for a robot manipulator or a couple of mobiles. S.M. LaValle [10] is based on the construction of a tree T in the considered space \mathcal{S} . Starting from the initial position q_{init} , the construction of the tree is carried out by integrating control commands iteratively. Each iteration aims at bringing closer the mobile \mathcal{M} to an element e randomly selected in \mathcal{S} . To avoid cycles, two elements e of T cannot be identical. In practice, *RRT* is used to solve various problems such as negotiating narrow passages made of obstacles [11], finding motions that satisfy obstacle-avoidance and dynamic balances constraints [12],

making Mars exploration vehicles strategies [13], searching hidden objects [14], rallying a set of points or playing hide-and-seek with another mobile [15] and many others mentioned in [9]. Thus the *RRT* method can be considered as the most general one by their efficiency to solve a large set of problems,

In its initial formulation, *RRT* algorithms are defined without goal. The exploration tree covers the surrounding space and progress blindly towards free space. A geometrical path planning problem aims generally at joining a final configuration q_{obj} . To solve the path planning problem, the *RRT* method searches a solution by building a tree (ALG. 1) rooted at the initial configuration q_{init} . Each node of the tree results from the mobile constraints integration. Its edges are commands that are applied to move the mobile from a configuration to another.

```

rrt( $q_{init}, k, \Delta t, C$ )
1  init( $q_{init}, T$ );
2  for  $i \leftarrow 1$  to  $k$ 
3       $q_{rand} \leftarrow \text{randomState}(C)$ ;
4       $q_{prox} \leftarrow \text{nearbyState}(q_{rand}, T)$ ;
5       $q_{new} \leftarrow \text{newState}(q_{prox}, q_{rand}, \Delta t)$ ;
6      addState( $q_{new}, T$ );
7      addLink( $q_{prox}, q_{new}, T$ );
8  return  $T$ ;

```

ALG. 1: Basic *RRT* building algorithm.

The *RRT* method is a random incremental search which could be casting in the same framework of Las Vegas Algorithms (*LVA*). It repeats successively a loop made of three phases: generating a random configuration q_{rand} , selecting the nearest configuration q_{prox} , generating a new configuration q_{new} obtained by numerical integration over a fixed time step Δt . The mobile \mathcal{M} and its constraints are not explicitly specified. Therefore, modifications for additional constraints (such as non-holonomic) are considered minor in the algorithm formulation.

In this first version, C is presented without obstacle in an arbitrary space dimension. At each iteration, a local planner is used to connect each couples (q_{new}, q_{prox}) in C . The distance between two configurations in T is defined by the time-step Δt . The local planner is composed by temporal and geometrical integration constraints. The resulting solution accuracy is mainly due to the chosen local planner. k defines the maximum depth of the search. If no solution is found after k iterations, the search can be restarted with the previous T without re-executing the init function (ALG. 1 line 1).

The *RRT* method, inspired by traditional Artificial Intelligent techniques for finding sequences between an initial and a final element (*i.e.* q_{init} and q_{obj}) in a well-known environment, can become a bidirectional search (shortened *Bi-RRT* [16]). Its principle is based on the simultaneous construction of two trees

(called T_{init} and T_{obj}) in which the first grows from q_{init} and the second from q_{obj} . The two trees are developed towards each other while no connection is established between them. This bidirectional search is justified because the meeting configuration of the two trees is nearly the half-course of the configuration space separating q_{init} and q_{obj} . Therefore, the resolution time complexity is reduced [17].

RRT-Connect [18] is a variation of *Bi-RRT* that consequently increase the *Bi-RRT* convergence towards a solution thanks to the enhancement of the two trees convergence. This has been settled to:

- ensure a fast resolution for “simple” problems (in a space without obstacle, the *RRT* growth should be faster than in a space with many obstacles);
- maintain the probabilistic convergence property. Using heuristics modify the probability convergence towards the goal and also should modify its evolving distribution. Modifying the random sampling can create local minima that could slow down the algorithm convergence.

```

connectT( $q, \Delta t, T$ )
1   $r \leftarrow \text{ADVANCED}$ ;
2  while  $r = \text{ADVANCED}$ 
3       $r \leftarrow \text{expandT}(q, \Delta t, T)$ ;
4  return  $r$ ;
```

ALG. 2: Connecting a configuration q to a graph T with *RRT-Connect*.

In *RRT-Connect*, the two graphs previously called T_{init} and T_{obj} are called now T_a and T_b (ALG. 3). T_a (respectively T_b) replaces T_{init} and T_{obj} alternatively (respectively T_{obj} and T_{init}). The main contribution of *RRT-Connect* is the `ConnectT` function which move towards the same configuration as long as possible (*i.e.* without collision). As the incremental nature algorithm is reduced, this variation is designed for non-differential constraints. This is iteratively realized by the expansion function (ALG. 2). A connection is defined as a succession of successful extensions. An expansion towards a configuration q becomes either an extension or a connection. After connecting successfully q_{new} to T_a , the algorithm tries as many extensions as possible towards q_{new} to T_b . The configuration q_{new} becomes the convergence configuration q_{co} (ALG. 3 lines 8 and 10).

Inherent relations inside the adequate construction of T in C_{free} shown in previous works are:

- the deviation of random sampling in the variations *Bi-RRT* and *RRT-Connect*. Variations include in *RRT-Connect* are called *RRT-ExtCon*, *RRT-ConCon* and *RRT-ExtExt*; they modify the construction strategy of one of the two trees of the method *RRT-Connect* by changing priorities of the extension and connection phases [19].

- the well-adapted q_{prox} element selected according to its collision probability in the variation *CVP* and the integration of collision detection since q_{prox} generation [20].
- the adaptation of C to the vicinity accessibility of q_{prox} in the variation *RC-RRT* [21].
- the parallel execution of growing operations for n distinct graphs in the variation *OR parallel Bi-RRT* and the growing of a shared graph with a parallel q_{new} sampling in the variation *embarrassingly parallel Bi-RRT* [22].
- the sampling adaptation to the *RRT* growth [23–27].

```

rrtConnect( $q_{init}, q_{obj}, k, \Delta t, C$ )
1  init( $q_{init}, T_a$ );
2  init( $q_{obj}, T_b$ );
3  for  $i \leftarrow 1$  to  $k$ 
4       $q_{rand} \leftarrow \text{randomState}(C)$ ;
5       $r \leftarrow \text{expandT}(q_{rand}, \Delta t, T_a)$ ;
6      if  $r \neq \text{TRAPPED}$ 
7          if  $r = \text{REACHED}$ 
8               $q_{co} \leftarrow q_{rand}$ ;
9          else
10              $q_{co} \leftarrow q_{new}$ ;
11             if connectT( $q_{co}, \Delta t, T_b$ ) =
                REACHED
12                  $\text{sol} \leftarrow \text{plan}(q_{co}, T_a, T_b)$ ;
13                 return sol;
14             swapT( $T_a, T_b$ );
15 return TRAPPED;

```

ALG. 3: Expanding two graphs T_a and T_b towards themselves with *RRT-Connect*. q_{new} mentionned line 10 corresponds to the q_{new} variable mentionned line 9 ALG. 4.

By adding the collision detection in the given space S during the expansion phase, the selection of nearest neighbor q_{prox} is realized in $S \cap C_{free}$ (ALG. 4). Although the collision detection is expensive in computing time, the distance metric evaluation ρ is subordinate to the collision detector. U defines the set of admissible orders available to the mobile \mathcal{M} . For each expansion, the function `expandT` (ALG. 4) returns three possible values: `REACHED` if the configuration q_{new} is connected to T , `ADVANCED` if q is only an extension of q_{new} which is not connected to T , and `TRAPPED` if q cannot accept any successor configuration q_{new} .

```

expandT( $q, \Delta t, T$ )
1   $q_{prox} \leftarrow \text{nearbyState}(q, T)$ ;
2   $d_{min} \leftarrow \rho(q_{prox}, q)$ ;
3   $success \leftarrow FALSE$ ;
4  foreach  $u \in U$ 
5       $q_{tmp} \leftarrow \text{integrate}(q, u, \Delta t)$ ;
6      if isCollisionFree( $q_{tmp}, q_{prox}, \mathcal{M}, C$ )
7           $d \leftarrow \rho(q_{tmp}, q_{rand})$ ;
8          if  $d < d_{min}$ 
9               $q_{new} \leftarrow q_{tmp}$ ;
10              $d_{min} \leftarrow d$ ;
11              $success \leftarrow TRUE$ ;
12 if  $success = TRUE$ 
13     insertState( $q_{prox}, q_{new}, T$ );
14     if  $q_{new} = q$ 
15         return REACHED;
14     return ADVANCED;
17 return TRAPPED;

```

ALG. 4: Expanding T with obstacles.

In the next section, we examine in detail some justifications of our algorithm and the inherent relations in the various components used. This study enables to synthesize a new algorithm named Rapidly exploring Sorted Random Tree (*RSRT*), based on reducing collision detector calls without modification of the classical random sampling strategy.

2 RSRT algorithm

Variations of *RRT* method presented in the previous section is based on the following sequence :

- generating q_{rand} ;
- selecting q_{prox} in T ;
- generating each successor of q_{prox} defined in U .
- realizing a colliding test for each successor previously defined;
- selecting a configuration called q_{new} that is the closest to q_{rand} among successors previously defined; This selected configuration has to be collision free.

The construction of T corresponds to the repetition of such a sequence. The collision detection discriminates the two possible results of each sequence:

- the insertion of q_{new} in T (*i.e.* without obstacle along the path between q_{prox} and q_{new});

- the rejection of each q_{prox} successors (*i.e.* due to the presence of at least one obstacle along each successors path rooted at q_{prox}).

The rejection of q_{new} induces an expansion probability related to its vicinity (and then also to q_{prox} vicinity); the more the configuration q_{prox} is close to obstacles, the more its expansion probability is weak. It reminds one of fundamentals *RRT* paradigm: free spaces are made of configurations that admit various number of available successors; good configurations admit many successors and bad configurations admit only few ones. Therefore, the more good configurations are inserted in T , the better the *RRT* expansion will be. The problem is that we do not previously know which good and bad configurations are needed during *RRT* construction, because the solution of the considered problem is not yet known. This problem is also underlined by the parallel variation *OR Bi-RRT* [22] (*i.e.* to define the depth of a search in a specific vicinity). For a path planning problem p with a solution s available after n integrations starting from q_{init} , the question is to maximize the probability of finding a solution; According to the concept of “rational action”, the response of *P3* class to adapt a on-line search can be solved by the definition of a formula that defines the cost of the search in terms of “local effects” and “propagations” [28]. These problems find a way in the tuning of the behavior algorithm like *CVP* did [20].

```

newExpandT( $q, \Delta t, T$ )
1   $q_{prox} \leftarrow \text{nearbyState}(q, T)$ ;
2   $S \leftarrow \emptyset$ ;
3  foreach  $u \in U$ 
4       $q \leftarrow \text{integrate}(q_{prox}, u, \Delta t)$ ;
5       $d \leftarrow \rho(q, q_{rand})$ ;
6       $S \leftarrow S + \{(q, d)\}$ ;
7  qsort( $S, d$ );
8   $n \leftarrow 0$ ;
10 while  $n < \text{Card}(S)$ 
11      $s \leftarrow \text{getTupleIn}(n, S)$ ;
12      $q_{new} \leftarrow \text{firstElementOf}(s)$ ;
13     if isCollisionFree( $q_{new}, q_{prox}, \mathcal{M}, C$ )
14         insertState( $q_{prox}, q_{new}, T$ );
15         if  $q_{new} = q$ 
16             return REACHED;
17         return ADVANCED;
18      $n \leftarrow n + 1$ ;
19 return TRAPPED;

```

ALG. 5: Expanding T and reducing the collision detection.

In the case of a space made of a single narrow passage, the use of bad configurations (which successors generally collide) is necessary to resolve such problem. The weak probability of such configurations extension is one of the weakness of the *RRT* method.

To bypass this weakness, we propose to reduce research from the closest element (ALG. 4) to the first free element of C_{free} . This is realized by reversing the relation between collision detection and distance metric; the solution of each iteration is validated by subordinating collision tests to the distance metric; the first success call to the collision detector validates a solution. This inversion induces:

- a reduction of the number of calls to the collision detector proportionally to the nature and the dimension of U ; Its goal is to connect the collision detector and the derivative function that produce each q_{prox} successor.
- an equiprobability expansion of each node independently of their relationship with obstacles;

The T construction is now based on the following sequence:

1. generating a random configuration q_{rand} in C ;
2. selecting q_{prox} the nearest configuration to q_{rand} in T (ALG. 5 line 1);
3. generating each successors of q_{prox} (ALG. 5 lines 3 to 6); each successor is associated with its distance metric from q_{rand} . It produces a couple called s stored in S ;
4. sorting s elements by distance (ALG. 5 lines 7);
5. selecting the first collision-free element of S and breaking the loop as soon as this first element is discovered (ALG. 5 lines 16 and 17);

3 Experiments

This section presents experiments performed on a Redhat Linux Cluster that consists of 8 Dual Core processor 2.8 GHz Pentium 4 (5583 bogomips) with 512 MB DDR Ram.

To perform the run-time behavior analysis for our algorithm, we have generated series of problems that gradually contains more 3D-obstacles. For each problem, we have randomly generated ten different instances. The number of obstacles is defined by the sequence 20, 40, 60, . . . , 200, 220. In each instance, all obstacles are cubes and their sizes are randomly varying between (5, 5, 5) and (20, 20, 20). The mobile is a cube with a fixed size (10, 10, 10). Obstacles and mobile coordinates are varying between $(-100, -100, -100)$ and $(100, 100, 100)$. For each instance, a set of 120 q_{init} and 120 q_{obj} are generated in C_{free} . By combining each q_{init} and each q_{obj} , 14400 configuration-tuples are available for each instance of each problem. For all that, our benchmark is made of more than 1.5 million problems. An instance with 20 obstacles is shown in FIG. 1 on the lower part and another instance with 100 obstacles in FIG. 1 on the left part. On these two examples, q_{init} and q_{obj} are also visible. We used the Proximity

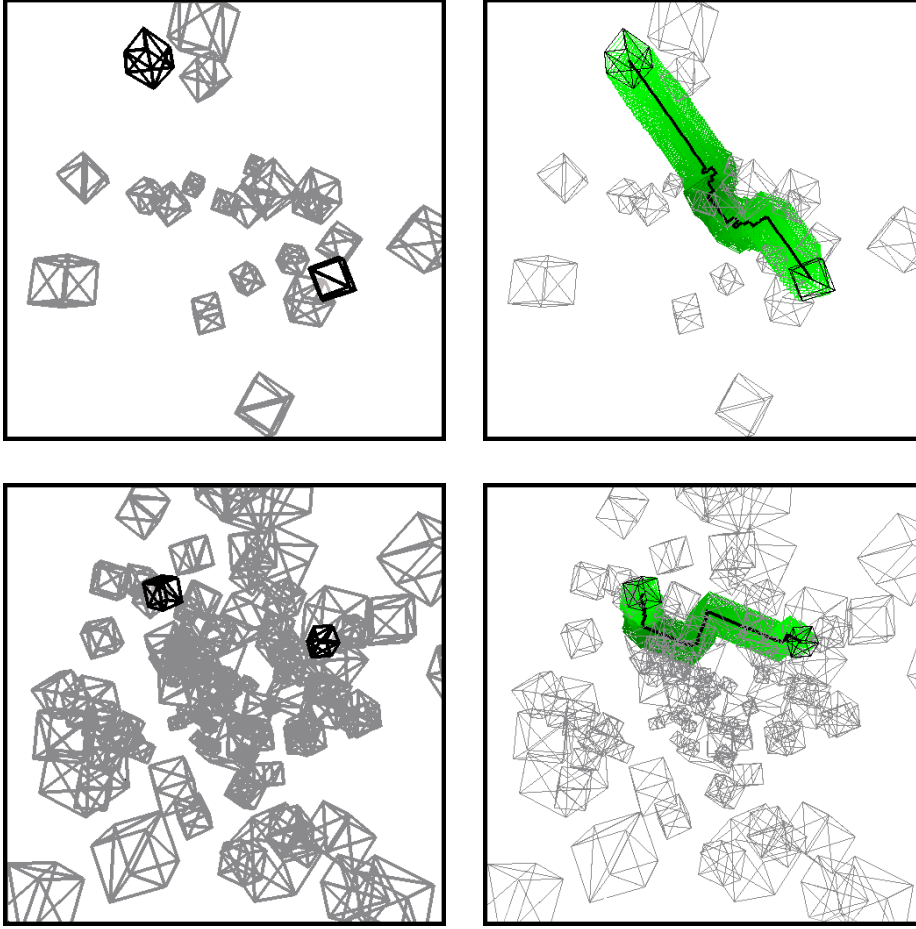


Fig. 1: 20 obstacles problem and its solution (upper couple). 100 obstacles problem and its solution (lower couple).

Query Package (*PQP*) library presented in [29] to perform the collision detection. The mobile is a free-flying object controlled by a discretized command that contains 25 different inputs uniformly dispatched over translations and rotations. The performance was compared between *RRT-Connect* (using the *RRT-ExtCon* strategy) and our *RSRT* algorithm (ALG. 5).

The choice of the distance metric implies important consequences on configurations' connexity in C_{free} . It defines the next convergence node q_{co} for the local planner. The metric distance must be selected according to the behavior of the local planner to limit its failures. The local planner chosen is the straight line in C . To validate the toughness of our algorithm regarding to *RRT-Connect*, we had use three different distance metrics. Used distance metrics are:

- the Euclidean distance (mentioned *Eucl* in FIG. 2 to 4)

$$d(q, q') = \left(\sum_{k=0}^i (c_k - c'_k)^2 + nf^2 \sum_{k=0}^j (\alpha_k - \alpha'_k)^2 \right)^{\frac{1}{2}}$$

where nf is the normalization factor that is equal to the maximum of c_k range values.

- the scaled Euclidean distance metric (mentioned *Eucl2* in FIG. 2 to 4)

$$d(q, q') = \left(s \sum_{k=0}^i (c_k - c'_k)^2 + nf^2(1-s) \sum_{k=0}^j (\alpha_k - \alpha'_k)^2 \right)^{\frac{1}{2}}$$

where s is a fixed value 0.9;

- the Manhattan distance metric (mentioned *Manh* in FIG. 2 to 4)

$$d(q, q') = \sum_{k=0}^i \|c_k - c'_k\| + nf \sum_{k=0}^j \|\alpha_k - \alpha'_k\|$$

where c_k are axis coordinates and α_k are angular coordinates.

For each instance, we compute the first thousand successful trials to establish average resolving times (FIG. 2), standard deviation resolving times (FIG. 3) and midpoint resolving times (FIG. 4). These trials are initiated with a fixed random set of seed. Those fixed seed assume that tested random suite are different between each other and are the same between instances of all problems. As each instance is associated to one thousand trials, each point of each graph is the average over ten instances (and then over ten thousands trials). On each graph, the number of obstacles is on x-axis and resolving time in *sec.* is on y-axis.

Figure 2 shows that average resolving time of our algorithm oscillates between 10 and 4 times faster than the original *RRT-Connect* algorithm. As the space obstruction grows linearly, the resolving time of *RRT-Connect* grows exponentially while *RSRT* algorithm grows linearly. Figure 3 shows that the standard deviation follows the same profile. It shows that *RSRT* algorithm is more robust than *RRT-Connect*. Figure 4 shows that midpoints' distributions follow the average resolving time behavior. This is a reinforcement of the success of the *RSRT* algorithm. This assumes that half part of time distribution are 10 to 4 times faster than *RRT-Connect*.

4 Conclusion

We have described a new *RRT* algorithm, called *RSRT* algorithm, to solve motion planning problems in static environments. *RSRT* algorithm accelerates consequently the resulting resolving time. The experiments show the practical performances of the *RSRT* algorithm, and the results reflect its classical behavior.

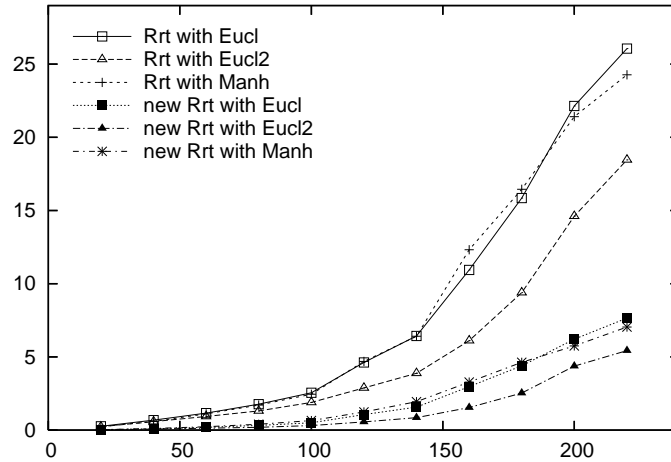


Fig. 2: Averages resolving times.

The results given above (have been evaluated on a cluster which provide a massive experiment analysis. The challenging goal is now to extend the benchmark that is proposed to every motion planning method. The proposed benchmark will be enhanced to specific situations that allow *RRT* to deal with motion planning strategies based on statistical analysis.

References

1. Canny, J.: The complexity of robot motion planning. PhD thesis, Massachusetts Institute of Technology. Artificial Intelligence Laboratory. (1987)
2. Schwartz, J., Sharir, M.: On the piano movers problem:I, II, III, IV, V. Technical report, New York University, Courant Institute, Department of Computer Sciences (1983)
3. Latombe, J.: Robot Motion Planning (4th edition). Kluwer Academic (1991)
4. Lozano-Pérez, T.: Spatial Planning: A Configuration Space Approach. In: Trans. on Computers. (1983)
5. Donald, B., Xavier, P., Canny, J., Reif, J.: Kinodynamic Motion Planning. Journal of the ACM (1993)
6. Fraichard, T.: Dynamic trajectory planning with dynamic constraints: a "state-time space" approach. In: Int. Conf. Robotics and Automation (ICRA'93). (1993)
7. Barraquand, J., Latombe, J.: A Monte-Carlo Algorithm for Path Planning with many degrees of Freedom. In: Int. Conf. on Robotics and Automation (ICRA'90). (1990)
8. Cheng, P.: Reducing rrt metric sensitivity for motion planning with differential constraints. Master's thesis, Iowa State University (2001)

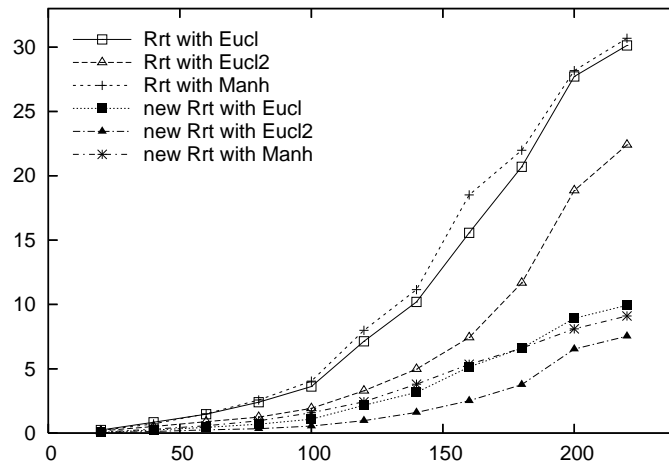


Fig. 3: Standard deviation resolving times.

9. LaValle, S.: Planning Algorithms. [on-line book] (2004)
<http://misl.cs.uiuc.edu/planning/>.
10. LaValle, S.: Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Dept. of Computer Science, Iowa State University (1998)
11. Ferré, E., Laumond, J.: An iterative diffusion algorithm for part disassembly. In: Int. Conf. Robotics and Automation (ICRA'04). (2004)
12. Kuffner, J., Nishiwaki, K., Kagami, S., Inaba, M., Inoue, H.: Motion planning for humanoid robots. In: Int'l Symp. Robotics Research (ISRR'03). (2003)
13. Williams, B.C., B.C., Kim, P., Hofbauer, M., How, J., Kennell, J., Loy, J., Ragno, R., Stedl, J., Walcott, A.: Model-based reactive programming of cooperative vehicles for mars exploration. In: Int. Symp. on Artificial Intelligence, Robotics and Automation in Space. 2001
14. Tovar, B., LaValle, S., Murrieta, R.: Optimal navigation and object finding without geometric maps or localization. In: Int. Conf. on Robotics and Automation (ICRA'03). (2003)
15. Simov, B., LaValle, S., Slutzki, G.: A complete pursuit-evasion algorithm for two pursuers using beam detection. In: Int. Conf. on Robotics and Automation (ICRA'02). (2002)
16. LaValle, S., Kuffner, J.: Randomized kinodynamic planning. In: Int. Conf. on Robotics and Automation (ICRA'99). (1999)
17. Russell, S., Norvig, P.: Artificial Intelligence, A Modern Approach (2ème édition). Prentice Hall (2003)
18. Kuffner, J., LaValle, S.: RRT-Connect: An efficient approach to single-query path planning. In: Int. Conf. on Robotics and Automation (ICRA'00). (2000)
19. LaValle, S., Kuffner, J.: Rapidly-exploring random trees: Progress and prospects. In: Workshop on the Algorithmic Foundations of Robotics (WAFR'00). (2000)

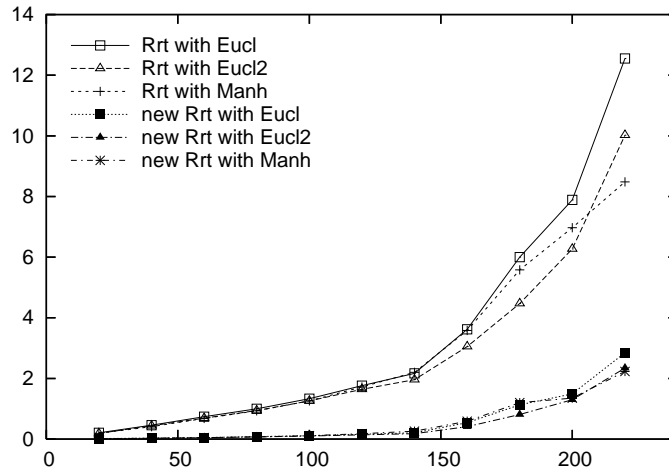


Fig. 4: Midpoint resolving times.

20. Cheng, P., LaValle, S.: Reducing Metric Sensitivity in Randomized Trajectory Design. In: *Int. Conf. on Intelligent Robots and Systems (IROS'01)*. (2001)
21. Cheng, P., LaValle, S.: Resolution Complete Rapidly-Exploring Random Trees. In: *Int. Conf. on Robotics and Automation (ICRA'02)*. (2002)
22. Carpin, S., Pagello, E.: On Parallel RRTs for Multi-robot Systems. In: *8th Conf. of the Italian Association for Artificial Intelligence (AI*IA'02)*. (2002)
23. Jouandeau, N., Chérif, A.A.: Fast Approximation to gaussian random sampling for randomized motion planning. In: *Int. Symp. on Intelligent Autonomous Vehicules (IAV'04)*. (2004)
24. Cortès, J., Siméon, T.: Sampling-based motion planning under kinematic loop-closure constraints. In: *Workshop on the Algorithmic Foundations of Robotics (WAFR'04)*. (2004)
25. Lindemann, S.R., LaValle, S.M.: Current issues in sampling-based motion planning. In: *Int. Symp. on Robotics Research (ISRR'03)*. (2003)
26. Lindemann, S., LaValle, S.: Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In: *Int. Conf. on Robotics and Automation (ICRA'04)*. (2004)
27. Yershova, A., Jaillet, L., Simeon, T., LaValle, S.M.: Dynamic-domain rrts: Efficient exploration by controlling the sampling domain. In: *Int. Conf. on Robotics and Automation (ICRA'05)*. (2005)
28. Russell, S.: *Rationality and Intelligence*. In Press, O.U., ed.: *Common sense, reasoning, and rationality*. (2002)
29. Gottschalk, S., Lin, M., Manocha, D.: Obb-tree: A hierarchical structure for rapid interference detection. In: *Proc. of ACM Siggraph'96*. (1996)