RRS: Rapidly exploration Random SNAKE a New Method for Mobile Robot Path Planning

Khelifa Baizid
Determent of Computer Graphics and Multimedia
Brno University of Technology (BUT), Brno, Czech Rep.
baizid.khelifa@gmail.com

Radim Luza
Determent of Intelligent System
Brno University of Technology (BUT) Brno, Czech Rep.
iluza@fit.vutbr.cz

Ryad Chellali
PAVIS Lab
Italian Institute of technology (IIT), Genova, Italy
ryad.chellali@iit.it

Beran Vitezslav
Determent of Computer Graphics and Multimedia
Brno University of Technology (BUT) Brno, Czech Rep.
beranv@fit.vutbr.cz

***Abstract*— During the last decade sampling-based path planning algorithms have been implemented in many practical robotics tasks. However, little improvements have been dedicated to the returned solution (from the point of view quality) and sampling process. The aim of this paper is to introduce a new technique that enhances the classical RRT algorithm. First, the sampling step is modified in order to increase the number of the possible solutions in the free space. Second, within possible solutions we apply an optimization scheme that gives the best solution in term of safety and shortness. The proposed solution, namely, the Rapidly exploring Random SNAKE (RRS) is a combination of Potential Fields (PF) and the RRT. The RRS takes the advantage of both RRT and PF in respectively: rapidly searching new candidate nodes in the free space and the to circumnavigating obstacles by calculating a safe sub-path in the free space towards the new node created by the RRT. In comparison to the classical RRT, the proposed algorithm increases the probability of completeness, accelerates the convergence and generates a much safer and shorter open-loop solution.**

***Index Terms*—Motion planning, sampling-based algorithms, optimal path planning, potential field**

I. INTRODUCTION

Robotic path planning problem has received a considerable amount of attention over the last years, where applications implementing real robots increase dramatically [1], [2]. The main issue that path planning solves is the possibility of driving robots from *Initial* to *Final* locations without colliding with any obstacle (safety condition of the path) in a minimum time. Such algorithms are judged efficient if they find a solution even in complex and cluttered environments. Moreover, the computational effort in this finding should be bounded, e.g., the algorithm provides at least one solution (completeness) if it exists in a finite amount of time. Cell decomposition [3] and visibility roadmaps [1] are known to guaranty the completeness. However, in practice the later algorithms are computationally expensive and hard to implement.

More recently, a new approach based on Sampling process of *C-space* has been introduced [1] and became very popular. The main advantage of this algorithm is to rely on random exploration to avoid visiting the whole working environment in order to derive an acceptable solution. Sampling Motion Planning (SMP) algorithms probe the *C-space* following an incremental sampling scheme and use a collision detector to find feasible paths [1]. The samples not verifying the collision-free conditions are not considered and the sampling process continue till the solution is found. One of SMP based algorithms is the Rapidly exploration Random Tree (RRT) [1] which generates random samples called *Nodes* and build a *Tree* from the *Start* to the *Goal* locations. RRT has almost two main conditions of collision free, one is the *Node* itself and the second is the *Edge* linking the current node and the candidate one. An improvement of the algorithm (OBRRT) was proposed in [4]. Unfortunately, these algorithms can only guarantee asymptotic completeness and no time-to-solution upper bound can be known a priori. On the other hand, the generated path is not optimal in length neither in smoothness.

In this paper we propose a new algorithm named "Rapidly exploration Random SNAKE (RRS)" based on the combination of SMP with Potential Field in 2D environment. The idea behind is to increase the number of accepted samples in E_{free} by circumnavigating locally obstacles by applying a local deformation to the colliding edges, thanks to the adaptation of the Active Contour Model (known also as "SNAKE" [5]). This combination leads to near-optimal paths in terms of number of probes, in length and in safety. Indeed, the *number of feasible paths* is increased for the same computational effort allowing additional operations that improves the safety and reduces the distance traveled by robots.

This paper is organized as follow; we first give related works to path planning approaches. We give after, the algorithm we proposed and its performances compared to the RRT algorithm. We finish by conclusion and future works.

II. RELATED WORKS

The fundamental issue in mobile robot path planning is to drive the system, robustly, from a known initial state S_{init} to the final state S_{goal} through a feasible trajectory, which links

both states [2]. By considering the prior knowledge of the environment $E = E_{free} + E_{obs}$, which contains free space E_{free} and obstacles E_{obs}. Generating a solution must fulfill three main conditions: 1) a feasible path S_{init} S_{goal}, a safe path that minimizes the risk of colliding with obstacles, 3) a short path minimizing the traveled distance. For cluttered and complex environments, the problem is known to be hard and subject to deadlock situations.

To address this problem, basically, four different classes path-planning techniques are used: Cell decomposition, Potential field, Probabilistic algorithms and Sampling methods [1], [2].

A. Cell decomposition or Grid-based algorithms

The configuration space is decompressed into squares forming a grid (or cell) [1] [6] in which every configuration corresponds to a certain grid. The robot moves from a point to an adjacent point, as long as that grid is in the free space E_{free} using a search algorithm such as A* [7] to find a path to get from start to the goal. These techniques handle complex problems in high dimensional spaces but usually operating in binary representation of the search space made up of free regions constrained by obstacles. Due to their nature solution paths, generally, have low quality and a post processing phase could be needed to improve specific criteria, like: the length or the distance to the obstacles. Regarding those inconvenient, some approaches which deal with these issues have been proposed in [8]. In addition, free space grid regions can be merged into hierarchical tree structure. This technique has been successfully implemented, but leads to more burdens in term of development cost and computation time [9]. An example falling in this category is the Bio-inspired algorithm. It relies on genetic algorithm as presented in [10] or Particle Swarm Optimization [11].

B. Potential fields algorithms

Potential fields-based algorithms built functions that present a global minimum, which attracts agents to the S_{goal} where E_{obs} generates repulsive forces. Originally, this approach has been proposed by Khatib in [12]. The robot is considered as a point robot in the configuration space under the influence of an artificial potential field depending on the distance from the target (S_{goal}) and the obstacles. Then, the robot moves only through the lowest adjacent potential value, which guides it toward S_{goal}. A global path planning approach was proposed by Charles [13], which is less sensitive to local minima. However, these methods need much more computation time, and the risk of getting trapped in local minima is very high.

C. Combinatorial algorithms

These algorithms construct structures within $E = E_{free} + E_{obs}$ to get necessary information needed for planning [1]. This information is obtained after *trapezoidal* or *triangular* decomposition or *Voronoi tessilation* [14]. The output of the applied processes is a visibility graph (VG), a generalized Voronoi graph (GVG) or *Roadmaps* [14], which are curves in E_{free}. Another important query step (e.g. using Dijkstra's algorithm to find connectivity schema which links the initial and the final states) is necessary. Usually, these methods are used off-line to prepare possible routes and paths that are checked during the searching phase. These methods deal better with completeness so that the existence of a solution is guarantied. However, having great number of edge and/or obstacles (E_{obs}) make the running time much higher. Preprocessing algorithm, namely the *plane sweep principle* [15] can reduce the query time dramatically.

D. Sampling-based algorithms

These algorithms avoid the explicit characterization of E_{obs} and E_{free} spaces by probing the environment E with a sampling schema, followed by a collision detection phase [1]. Clearly, the whole map is not visited and only randomly generated steps within E_{free} are considered. Probabilistic RoadMaps (PRM) [16] and Rapidly exploring Random Tree [17] which was introduced in 2001 are the most popular sampling based algorithms. Basically, a point in the space E is randomly chosen and free-collision test is performed. For positive answers, the nearest point in the tree is connected to this candidate. In addition to the advantages cited earlier, the RRT algorithm performs simultaneously the classical preprocessing and searching steps, which make it well adapted for real-time executions to handling uncertainties and sensory uncertainties. The Box-RRT was proposed by Pepy et al in [18] to deal with such uncertainties.

In general, RRT-like based algorithms are only implemented to find a possible path efficiently without considering the inherent costs. Recently, a new improvement of sampling methods mainly a guided utility function which guides the expansion algorithm towards regions having higher utility based on information of the state space [19]. Another method called RRT* to improve the returned solution and the internal process like increasing the number of samples and reducing the cost of the solution [20]. Regarding to limitations of RRT* in high dimensional space a heuristics method was proposed in [21] to improve the initial path and decrease iteratively the computational efforts.

To handle such issues, we investigated a new formulation of the basic RRT. Like in the classical RRT algorithm, first we generate a random sample (q_{rand}). In addition to this step, considering the local effects of obstacles allows creating a local potential field. The later is used to deform (using the adapted SNAKE) the original straight path to circumnavigate dynamically the closer obstacles. This deformation can handle as well safety considerations by adding a supplementary constraint that allows minimizing the length of the obtained sub-segment. We intend to extend the sampling searching strategy by giving the possibility of accepting samples

considered unreachable by the basic RRT. This increases the change to find optimal solutions in less iterations.

III. Rapidly Exploration Random Snake (RRS)

The basic idea behind our RRS is to combine the classical RRT together with SNAKE, an active contour model used previously in computer vision for image segmentation.

A. Active Contour Model for Path Planning

SNAKE was proposed initially for Image Segmentation in Computer Vision [5]. SNAKE deforms a line under the effects of forces derived from image grey levels or colors. These forces (or equivalently the potential field) push the line towards the equipotential curves, which actually the contour between two adjacent regions. This approach is adapted here to cope with the path generation and obstacle avoidance: it pushes the line far enough from the present object and finds a safe path in case of cluttered environment.

We build the energy of the SNAKE model based on the obstacles and the robot path. Based on the energy minimization, our SNAKE line is subject to the influence of the environment and the internal trajectory forces. The external forces (Obstacles) push the contour (path) far from zones of high potential field (higher collision risk probability). The internal force maximizes the smoothness of the contour. Assuming that the trajectory parameters are given by: $v(s)=(x(s),y(s))$. The SNAKE model is defined as follow:

$$E_{Snake}=\int_0^1 E_{IntFrc}(v(s))+\int_0^1 E_{ExtFrc}(v(s))+\int_0^1 E_{Con}(v(s))+\int_0^1 E_{OpFrc}(v(s)) \quad (1)$$

Where, E_{IntFrc} represents the internal energy of the contour (path), E_{ExtFrc} represents the external energy of the contour (obstacles), E_{Con} represents the external force constraints (which is not considered in this case) and E_{OpFrc} represents the energy to optimize the line (stretching the path).

The internal energy is defined as follow:

$$E_{IntFrc}=\left(\alpha(s)|v(s)|^2+\beta(s)|v(s)|^2\right)/2 \quad (2)$$

Where the first part (function of α) is a membrane force of the contour, and the second part (function of β) is a force acting to counter to be thin.

For the external energy we are using the term of "$1-E_{ExtFrc}$" defined in [5]. Where $E_{ExtFrc}=-\left(G_\sigma * \nabla^2 I\right)^2$.

Where I represents the data information containing the obstacles primitives.

The Optimization energy $E_{OpFrc}=\int_0^1 \sum \vec{f}(x,y)dt$ is defined by:

$$F=N*\vec{n}(s) \quad (3)$$

Where $\vec{n}(s)$ represents the normal unitary vector to the curve at point $v(s)$ and N is the weight, for a large value of N the curve converges to straight line very quickly. The normal of each point is strongly related to all other points' position in the path.

To make the path safer, we introduce a gaussian distribution representing robot's movements (Δm) and sensors uncertainties (Δo) (Fig. 1). It forces the second term of the Eq. 1 to consider these uncertainties by deforming the original path.

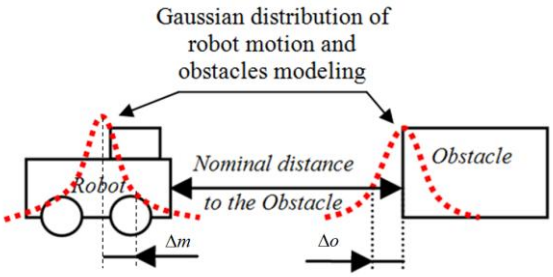


Fig. 1. Probabilistic model of the robot motion and obstacles modeling

Figure 2 shows an example of path deformation by the SNAKE algorithm. The blue color zone in Fig.2(a) is the force field exerted by obstacles. The found path after deformation is represented in Fig.2 (b) by green color.

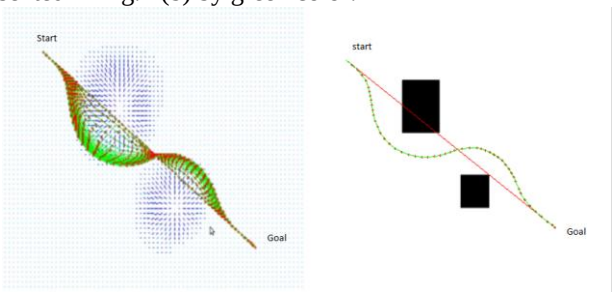


Fig. 2. Adopted SNAKE algorithm, (a) Deformation process, (b) Path before and after deformation

B. Sampling Active Contour Model: the RRS-Core

RRS builds a random tree similar to RRT algorithm. However, instead of taking the decision about invalid Nodes (accept or reject) the algorithm creates a local potential field based on the adapted SNAKE model to deform the curve which links the corresponding node with the mother tree. The RRS core algorithm is shown below:

Input: Initial and final configurations q_{init} and q_{fin}, maximum number of vertices *K*, incremental distance Δq.

Output: RRS graph G

1. G_{init} (q_{init})
2. for k = 1 to K
3. $q_{rand} \leftarrow$ RAND_CONF()
4. $q_{near} \leftarrow$ NEAREST_VERTEX(q_{rand}, G)
5. $q_{new} \leftarrow$ NEW_CONF(q_{near}, Δq)
   if q_{new} lies in E_{free}
   * *SNAKE curve deformation*
6. * G.add _vertex(q_{new})
7. * G.add _curve(q_{near}, q_{new})
8. end if
9. return G

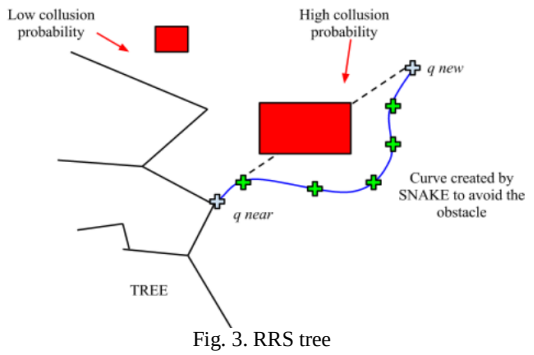


Fig. 3. RRS tree

After initialization, the algorithm starts the main loop. In the loop it generates a random node q_{rand} in a configuration space E during ***Step 3***. The conventional RRT checks if the random configuration q_{rand} in E belongs to E_{free} (by using a collision detection algorithm to reject samples in E_{obs}). If node does not belong to E_{free}, it is rejected. Only accepted nodes which satisfy the collision free condition are processed in ***Step 5***. If node does not satisfy the condition of collision free the current iteration is canceled and algorithm continues with next iteration.

The collision free condition is satisfied if and only if q_{rand} belongs to E_{free} and the edge which links q_{rand} to the previous node q_{near} in the TREE not crossing an existing obstacles. By modifying locally this condition, the SNAKE sets up the collision free condition to be satisfied by creating a set of new nodes in E_{free} circumnavigating obstacles. The local SNAKE starts with an initial "curve" C_{init} corresponding to the straight tree segment. In fact, the SNAKE gives more chance to accept nodes which do not satisfy the collision-free condition by providing a new curve C_{new} which links q_{near} and q_{rand}. SNAKE is involved only if the $edge(q_{near},q_{new})$ has a high collision probability P_{col} with the surrounding obstacles and then deform it from: E_{obs} to: E_{free}. After deformation the EDGE(q_{near},q_{new})= C_{init} becomes CURVE($C_{new} = f(q_{near}, q_{new}, E_{Ext})$). The probability P_{col} is calculated based on the force field generated by obstacles and it is defined by the following equation:

$$P_{col} = \sum_{0}^{n} E_{Ext} * C_{init} \quad (4)$$

Where *n* is the number of the points in $C(s)$.

The sub-path given by curve C_{new} is accepted if it satisfies the safety probability condition represented by P_{safe} which is defined in Eq. 5.

$$P_{safl} = \sum_{0}^{n} E_{Ext} * C_{new} \quad (5)$$

An example of a path generated by RRS algorithm from the a *start* to the *goal* locations is shown in the Fig. 3 by red color. Curves with pink color are edges deformed by SNAKE to circumnavigating obstacles (the path includes around 6 curves). In case of conventional RRT all these edges are rejected. The min probability of collision P_{col} , which was considered in this example is 0.95.

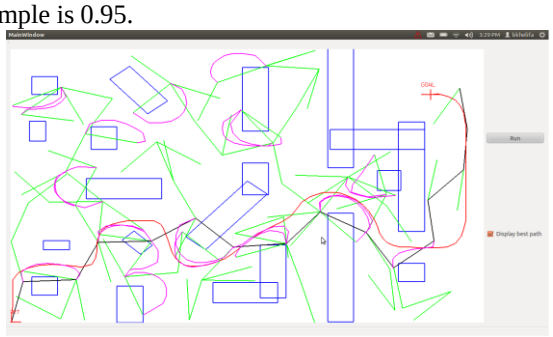
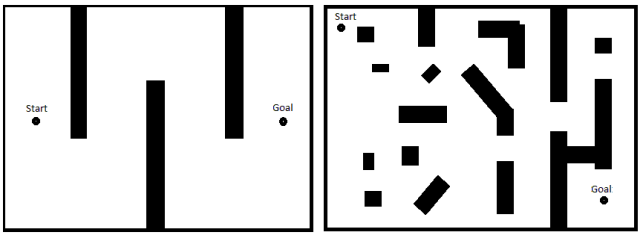


Fig. 4. Path Generated by RRS algorithm including all sub-paths created by adapted SNAKE (pink color)

IV. Resultes and Dicsussion

In our experiment we evaluate RRT and RRS in three different environments. The first environment is simple, the second is rather complicated and the third one is a randomly generated environment. Similar to [19] we see how the two algorithms perform in terms of ***Cost*** (Iteration taken to find solution), ***Optimality*** (Path shortness), ***Number of Samples*** and ***Time of Convergence***. Figure 5 shows those environments.



(a) (b)

Fig. 5. Experiment Environment, (a) Simple, (b) Complex

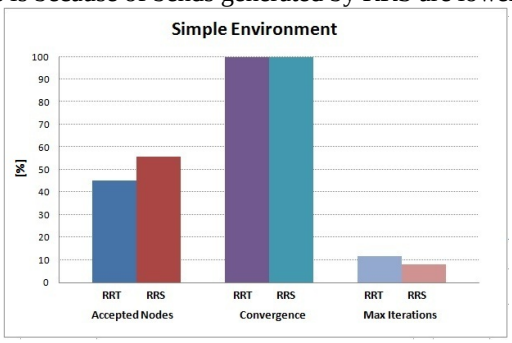
The two algorithms have the same test conditions in the three setups: Simple (Fig. 5(a)), Complex (Fig. 5(b)) and Random (objects are placed randomly, potentially cluttered and with potential deadlocks) environment.

- We consider that a solution is accepted (satisfied condition) if and only if the returned path links S_{init} and S_{goal} without collision.
- If the condition above is not satisfied, we allow both algorithms to reach the max iteration number by trying to find other possibilities (creating new nodes) to reach S_{goal} .
- Since both algorithms are based on random generation we pre-specified 100 random test to minimize the effect of chance.

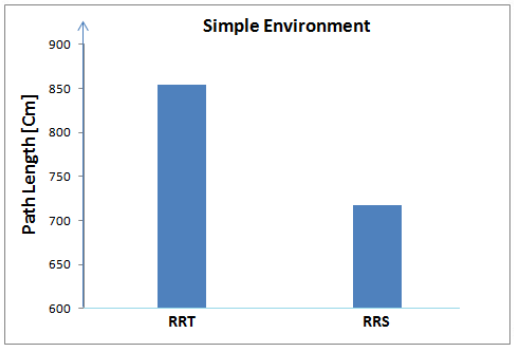
Basically, the RRS has good performances regarding all parameters cited before in comparison with RRT.

Figure 6 shows a comparison between RRT and RRS algorithms in simple environment. In Fig.6 (a) we can see clearly the good performances of RRS compared to RRT in term of accepted nodes: RRS has almost 10% (over 2000 nodes) grater then RRT. However, the percentage of convergence is the same because the environment is quite simple and the solution is easily found. Also both algorithms have small number of iterations: 231 (11.55%) and 149 (7.45%) for RRT and RRS respectively. The RRS needs almost the half of iterations that RRT takes.

For the path length the RRS has better solution compared to RRT. This is because of bends generated by RRS are lower.



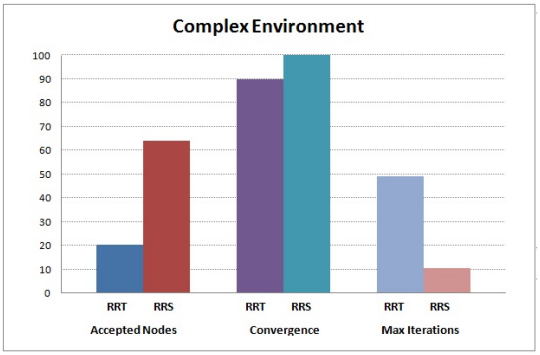
(a)



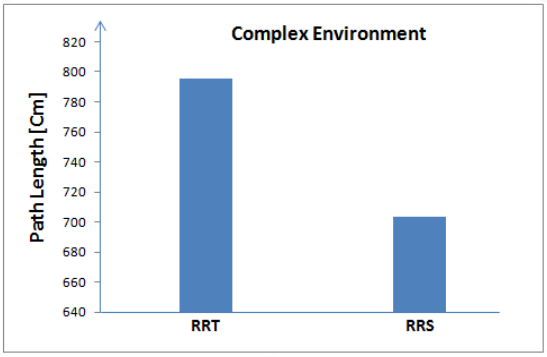
(b)

Fig. 6. Comparison between RRT and RRS in simple environment, (a) Percentage of Accepted Nodes, Percentage of Convergences and Percentage of max iterations taken to find solution, (b) Length of the path.

For complex environment (Fig. 7) with lots of obstacles, the task is more challenging. In this case, RRS performs better in term of node acceptance ratio, which is bigger than 60% while for RRT it is only about 20%. RRS in fact delays the decision making of choosing a given path by tolerating more possibilities to the obstacles avoidance process. Also there are some difficulties for the RRT to converge in all tests (only 90 times from 100) while the RRS succeeded to find a solution in all tests. In average, the maximum number of iterations taken to find a solution for the RRS algorithm was four times smaller than for RRT, which was 980,23. Also the paths given by RRS in all experiments are smoother and shorter. Again, allowing more accepted nodes, the RRS allows finding shorter paths in many cases. We noticed that the difference is almost 2 meters.



(a)



(b)

Fig. 7. Comparison between RRT and RRS in complex environment, (a) Percentage of Accepted Nodes, Percentage of Convergences and Percentage of max iterations taken to find solution, (b) Length of the path.

Figure 8 shows two solution - one found by the RRT (blue color) and one found by the RRS (red color). We can see that the RRS found a smoother and shorter path.

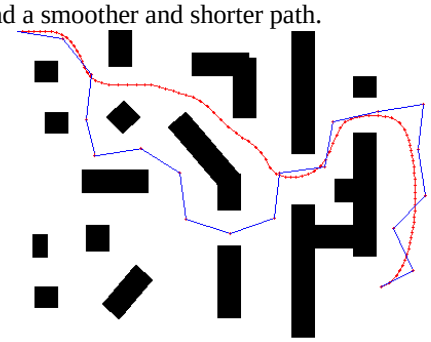


Fig. 8. Two examples of paths, RRT (blue color) and RRS (read color) in term of pat shortness.

The second part of the experiment is designed to evaluate the efficiency of the proposed algorithm in any environment by generating randomly a set of obstacles and their relative location in the scene. The min and max number of obstacles are 5 and 20 respectively. Regarding to time of finding solutions, differences are very small. However, RRT performs better then RRS in complex and simple environment (Table 1). It is probably caused by narrow and long obstacles appearing in both environments, which take lots of time for SNAKE to deform around them. However RRS is better in random environment. Time in Table 1 is calculated only for successful test. The percentage of convergence in random environment was a bit lower than the others, is around 89% for RRS and

88% for the RRT. This is caused by blocked situations where obstacles close the way to the goal.

Also we can notice the participation ration of adapted SNAKE in Node acceptance is higher in complex environment (51%) because is much encumbered.

TABLE I. EVALUATION RESULT ACCORDING TO TIME OF CONVERGENCE TAKEN BY EACH ALGORITHM IN ALL ENVIRONMENTS

Environment	Type	Mean Number of accepted Nodes [Integer]	Percentage of involved SNAKE in node acceptance [%]	Mean Time
Simple	RRT	99,19	---	0.082
	RRS	78,90	30,74	0.091
Complex	RRT	188,65	---	0.087
	RRS	125,88	51,07	0.110
Random	RRT	280,51	---	0.141
	RRS	297,75	29,04	0.123

V. CONCLUSION

In this paper we proposed a new path planning method based on probabilistic sampling and potential fields techniques. Mainly, it combines the Rapidly exploration Random Tree (RRT) and the Active contour Model (SNAKE). The RRT probes the environment in a relaxed way, while the SNAKE adjusts the candidate paths following three goals: guarantying safety, smoothing the path and shortening it. Our Method performs well in simple and complex environments in comparison to the classical RRT algorithm. Indeed, we demonstrated how the local deformations generated by the SNAKE enhance the capabilities of the original sampling algorithm and converge in almost all cases and with less iterations. Moreover, the retuned solution is optimized in terms of traveled distance regarding the RRT solutions.

Our future works are focusing on kinematic issues. Indeed, we are integrating non-holonomic constraints and more DoF's to allow handling more robots, including mobile manipulators for real time and real life scenarios with unknown and challenging environments.

ACKNOWLEDGMENT

The research leading to these results has received fund-grant 247772 - SRS, Artemis JU grant 100233 - R3-COP, and the IT4Innovations Center of Excellence, grant n. CZ.1.05/1.1.00/02.0070, supported by Operational Program "Research and Development for Innovations" funded by Structural Funds of the European Union and the state budget of the Czech Republic.

REFERENCES

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

[2] J.C. Latombe, "Robot Motion Planning", *Vol. SECS 0124*, Kluwer, Dordrecht, The Netherlands, 1991

[3] B. Chazelle. Approximation and decomposition of shapes. In J. T. Schwartz and C. K. Yap, editors, Algorithmic and Geometric Aspects of Robotics, pages 145-185. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.Samuel Rodriguez, Xinyu Tang, Jyh-Ming Lien, Nancy M. Amato, "An Obstacle-Based Rapidly-Exploring Random Tree" , IEEE Int. Conference on Robotics and Automation. (ICRA), pp. 895-900, Orlando, FL, May 2006.

[4] Michael. Kass, A. Witkin and D, Terzopoulos: "SNAKEs Active contour models", *Journal of Computer Vision*, pp.321-331, 1987.

[5] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., and Thrun, S. (2005). Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents). The MIT Press.

[6] Hart, P.E.; Nilsson, N.J.; Raphael, B.; , "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol.4, no.2, pp.100-107, July 1968.

[7] Han-dong Zhang; Bao-hua Dong; Yu-wan Cen; Rui Zheng; Hashimoto, S.; Saegusa, R.; , "Path Planning Algorithm for Mobile Robot Based on Path Grids Encoding Novel Mechanism," *Third International Conference on Natural Computation, 2007*, vol.4, no., pp.351-356, Aug. 2007

[8] Ferguson, D. and Stentz, A., Multi-resolution field D*," in Proceedings of the International Conference on Intelligent Autonomous Systems (IAS), March 2006.

[9] Yanrong Hu; Yang, S.X.; , "A knowledge based genetic algorithm for path planning of a mobile robot," *2004 IEEE International Conference on Robotics and Automation*, vol.5, no., pp. 4350- 4355, May 2004.

[10] Wu Xianxiang; Ming Yan; Wang Juan; , "An improved path planning approach based on Particle Swarm Optimization," *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, vol., no., pp.157-161, 5-8 Dec. 2011

[11] Khatib, O.; , "Real-time obstacle avoidance for manipulators and mobile robots," *1985 IEEE International Conference on Robotics and Automation*. Proceedings. , vol.2, no., pp. 500-505, Mar 1985.

[12] Warren, C.W.; , "Global path planning using artificial potential fields," *Proceedings of 1989 IEEE International Conference on Robotics and Automation*, vol.1, pp.316-321, 14-19 May 1989.

[13] Bhattacharya, P.; Gavrilova, M.L.; , "Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path," *IEEE Robotics & Automation Magazine,*, vol.15, no.2, pp.58-66, June 2008.

[14] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Computational Geometry: Algorithms and Applications, 2nd Ed. Springer-Verlag, Berlin, 2000.

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

[16] S. M. Lavalle and J. J. Kuffer, "Rapidly-exploring random trees: Progress and prospects", *Workshop on the Algorithmic Foundations of Robotics*, 2000.

[17] R. Pepy and M. Kieffer and E. Walter, "Reliably Safe Path Planning Using Interval Analysis", Laboratoire des Signaux et Syst_emes CNRS - SUPELEC - Univ Paris-Sud

[18] B. Burns and O. Brock. "Single-query motion planning with utility guided random trees". *IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007.

[19] Sertac Karaman, "Sampling-based algorithms for optimal motion planning", *International Journal of Robotics*. 2011.

[20] Baris Akgun and Mike Stilman, "Sampling Heuristics for Optimal Motion Planning in High Dimensions", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2011.