RRS: Rapidly-exploring Random Snakes a New Method for Mobile Robot Path Planning

K. Baizid[1], R. Chellali[2], R. Luza[3], B. Vitezslav[3] and F. Arrichiello[1]

[1]University of Cassino and Southern Lazio, Via G. Di Biasio 43, 03043 Cassino (FR), Italy
baizid@unicas.it
[2]Fondazione Instituto Italiano di Technologia (IIT), via Morego 30, Genova, Italy
[3]University of Technology Brno (BUT), Czech Republic

Abstract. Recently, sampling-based path planning algorithms have been implemented in many practical robotics tasks. However, little improvements have been dedicated to the returned solution (quality) and sampling process. The aim of this paper is to introduce a new technique that improves the classical Rapidly-exploring Random Trees (RRT) algorithm. First, the sampling step is modified in order to increase the number of possible solutions in the free space. Second, within the possible solutions, we apply an optimization scheme that gives the best solution in term of safety and shortness. The proposed solution, namely, Rapidly-exploring Random Snakes (RRS) is a combination of a modified deformable Active Contours Model (called Snakes) and the RRT. The RRS takes the advantage of both RRT and deformable Snakes contours, respectively, in: rapidly searching new candidate nodes in the free space and 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, hence, increasing considerably the efficiency of the classical RRT. The proposed approach has been validated via numerical simulations and experimental results with a mobile robot.

Keywords: Path planning, Active Counter Model, Sampling algorithms, Snake

1 INTRODUCTION

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

More recently, the Sampling-based Motion Planning (SMP) 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. The SMP probes the configuration space following an incremental sampling scheme, and uses a collision detector to find feasible paths. The samples not verifying the collision-free conditions are not considered and the sampling process continues till a solution is found. One of the SMP is the Rapidly exploring Random Trees (RRT) [1], which generates random samples called *Nodes* and builds a Tree from the *Start* to the *Goal* locations. Numerous variants of this algorithm have been introduced and developed to improve its performances in solving different path planning problems, e.g. [4][5],[6],[7],[8],[9]. Unfortunately, some of these algorithms can only guarantee asymptotic completeness and no upper bound for the *time-to-solution* can be known a priori. On the other hand, the generated paths are not optimal in length neither in safety.

In this paper we propose a new RRT extension, called the Rapidly-exploring Random Snakes (RRS). Our RSS is based on the combination of the classical SMP together with a modified Deformable Active Contours Model (known also as Snakes [10]), where the later refining the obstacle avoidance process (To the best of our knowledge no similar work has been proposed combining deformable Snakes with SMP methods). The main idea in this contribution is to increase the number of accepted samples in the free space by circumnavigating locally closest obstacles by applying a deformation process to the edges not satisfying the RRT free conditions. This combination leads to near-optimal path in terms of length, number of probes and safety. Indeed, the *number of feasible paths* is increased for the same computational effort, allowing more operations towards improving the safety and reducing the lengths of the possible trajectories. Moreover, we show trough simulation results that the proposed algorithm maximize the probability of finding solution (hereafter it is called the *probabilities completeness*) compared to classical procedures with less computational efforts.

This paper is organized as follows; we give, first, related to path planning. In the second part, we describe the proposed algorithm and its performances compared to the classical RRT algorithm. We finish by a conclusion and future works.

2 Related works

The fundamental issue in mobile robotics path planning is to drive the system, robustly, from a known initial state S_{init} to a final state S_{goal} through a feasible trajectory within a known environment. Any solution P must fulfill two main conditions: 1) P is safe by minimizing the risk of colliding with obstacles, 2) P is a short path by minimizing the traveled distance. For cluttered and complex environments, finding P is known to be hard and subject to deadlock situations. Basically, such issue is addressed in four different classes of path planning approaches: Cell decomposition [7],[8], Potential field [11],[12], Probabilistic algorithms [13],[14] and Sampling methods [1],[2]. For brevity hereafter we give reference to Sampling methods, which are much related to our contribution.

Sampling based algorithms, avoid the explicit characterization of C_{obs} and C_{free} by probing the configuration space C with a sampling schema, followed by a collision

detection phase. Clearly, the whole map is not visited and only randomly generated steps within C_{free} are considered. Probabilistic Roadmaps [15] and RRT [16], introduced since 2001 are the most popular sampling based algorithms. Basically, a point in the space C is randomly chosen and a 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 makes it well adapted for real-time executions to handle sensory uncertainties. The Box-RRT was proposed by Pepy et al in [6] 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 variant of sampling methods based on random tree expansion strategy has been presented in [17]. Mainly an utility-guided algorithm guides expansion towards regions having higher utility based on local information of the environment. Another method called RRT* has been proposed in [7] to improve the returned solution and the internal process by increasing the number of samples and reducing the computational costs. Regarding to limitations of this method in high dimensional space a heuristics method was proposed in [18] to improve the initial path and decrease iteratively the computational efforts. Also, the an Obstacle-Based RRT (OBRRT) was proposed in [19].

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 we consider the local effects of obstacles to create a local deformation process of the edge to connect the new candidate q_{rand} to the mother tree. This process uses a modified Snakes algorithm to circumnavigate dynamically the closest obstacles. Furthermore, it handles the safety issues and adds a supplementary constraint to minimize the length of the obtained sub-segment. We intend to extend the sampling searching strategy by giving it the possibility of accepting more samples, which are considered unreachable by the classical RRT. This increases the probability of completeness and minimizes the number of iterations.

3 Rapidly exploring Random Snakes (RRS)

The RRS combines the classical RRT and a modified Snakes algorithm in order to increase the number of samples which leads to increase the probability of completeness, accelerate the convergence and generate a safer and shorter paths. Hence, its aim is to increase the efficiency of the RRT basic algorithm.

3.1 Applying Active Contour Model for Safe Path Planning

Snakes algorithm was proposed initially for object boundaries detection in Computer Vision [10] and is used in several applications as, for example, [20]. Mainly, it deforms closed loop contour under the effects of forces (energy minimization process) derived from image grey levels or colors. This contour moves under the influence of internal forces coming from the contour itself and external forces computed from the

data of the image, towards equipotential zones of this last. This allows the contour to matches the object boundaries. In our method, this concept is modified to cope with the path generation and obstacle avoidance: it allows to circumnavigate obstacles keeping a safe distance from them even in cluttered environments.

The proposed Snakes algorithm calculates the energy model (Eq.1) based on the obstacles and the robot path projected on a bright image, which its size represents the environment boundaries. Similar to original Snakes model, and based on the energy minimization concept, the contour is subjected to the influence of the environment and the internal forces of the contour itself. The external forces push the contour far from zones having higher collision risk probability (high potential field), the internal forces ensure the connectivity and maximize the smoothness of the contour, while the optimization force reduces curves bands. Assuming that the path parameters (contour) are given by: $v(s) = (x(s), y(s))$. The Snakes energy is defined as follow:

$$E_{Snakes} = \int_0^1 E_{Int}(v(s))ds + \int_0^1 E_{Ext}(v(s))ds + \int_0^1 E_{Con}(v(s))ds + \int_0^1 E_{Opt}(v(s))ds \quad (1)$$

where, E_{Int} represents the internal energy of the contour (path), E_{Ext} represents the external energy of the contour, E_{Con} represents the external force constraints (which is not considered in our case) and E_{Opt} represents the energy to optimize the contour (stretching the path). The internal energy is defined as follow:

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

where the first part (function of α) of the equation makes the contour act like a membrane and the second part (function of β) makes it act like a thin plate. We modified the external energy E_{Ext} to be repulsive as defined bellow:

$$E_{Ext}(s) = (1 - (G_\sigma * \nabla^2 I)^2)v(s) \quad (3)$$

where G_σ is a two-dimensional Gaussian function with standard deviation sigma ∇ is the gradient operator and I represents the image data (for a complete review regarding the Snakes model please see [10]). The Optimization energy is defined as:

$$E_{Opt}(s) = \sum \vec{f}(x, y) \quad (4)$$

where $f = \gamma * \vec{n}(s)$, $\vec{n}(s)$ represents the normal unitary vector of the curve at points *v(s)* and γ is a weight; for a large value of γ the curve converges toward a straight line very quickly.

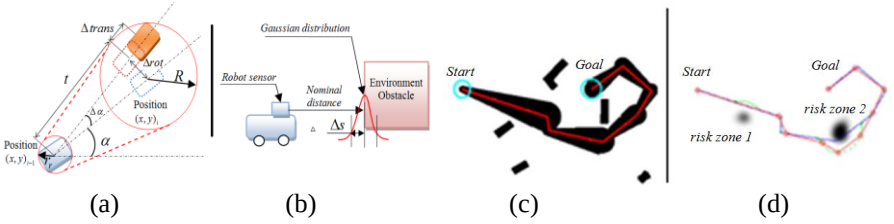
To insure the feasibility of the path, the original RRT algorithm generates a "new sample" using a normal distribution where all robots' directions are with equal probability. This may interfere with the consideration of the robot motions anisotropy (the rotation and translation errors). Indeed, the errors in executing rotations and translations are not uniform. There are many ways to model the uncertainties produced by motions and sensors [21]. In our case, we construct a local function to describe the

probability for the robot to pass by the position $(x, y)_i$ while performing a motion step ($\Delta trans$, $\Delta\alpha$). This function is obtained as the combination of two Gaussians, respectively the translation and the rotation ones (Fig. 1 (a)).

Likewise and due to the sensing errors, obstacles are transformed into probability distributions of being at the expected positions (Fig. 1 (b)). The product of the motion distributions and the sensing ones leads to a landscape representing the probability for the robot to hit an obstacle (risk zones in Fig 1.(d)).

Authors in [21] provide a complete review regarding to motion uncertainties. For brevity our method is very simplified. Moreover, an experimental study was performed to model motion and sensor uncertainties using Snakes algorithm in [22].

3.2 Active Contour Model for Safe Path Planning



(a) (b) (c) (d)

Fig. 1. A simplified probabilistic model of the robot motion and obstacles uncertainties, (a) Envelope of the robot performing rotation and translation, (b) Distance sensor uncertainties, (c) Obstacles and path primitives that define the Image data *I*, (d) Collisions risk zones.

Fig. 1 shows an example of the modified Snakes model applied to the whole path generated by RRT (not to the generated samples in real-time as designed for). The green dashed path in Fig. 2(d) represents the deformed segments, of the original RRT path (close to the two risk zones found).

3.3 Sampling Active Contour Model: the RRS-Core

RSS builds a random tree similar to RRT algorithm, however, instead of taking the decision about invalid samples (accept or reject) it creates a local *deformable Snakes contour* derived from the curve that links the corresponding sample to the mother tree. This contour is expected to circumnavigate local obstacles safely. To this end, a collision test is performed to judge (accept or reject) the corresponding sample.

After the initialization, the algorithm starts to generate a random sample q_{rand} in the configuration space C (Step 3). The conventional RRT checks if q_{rand} belongs to C_{free} based on a collision detection algorithm. Samples not belonged to C_{free} are rejected, while those which satisfy the collision free condition are considered in Step 5. In case the sample is rejected, the algorithm processes the next iterations till it finds a feasible path. Hence, the collision free condition is satisfied if and only if q_{rand} belongs to C_{free} and the edge that links q_{rand} to the previous sample q_{near} (which belong to the mother tree) is not colliding with any existing obstacle (C_{obs}). By modifying locally this condition, the RRS trays to rescue the current sample, to not be rejected, by creating a set of new samples based on the deformable Snakes contour in C_{free} circumnavigating close obstacles safely.

RRS Algorithm

Input: Initial and final configurations q_{init} and q_{fin}, maximum number of samples K, incremental distance Δq.
Outputs: 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)
   Create initial curve CV_{init} (q_{near}, q_{new})
   if CV_{init}(q_{near}, q_{new}) lies in C_{free}
   * G.add_vertex(q_{new})
   * G.add_curve(CV_{init})
   else
6. * Snakes curve deformation process ($CV_{new} = v(s)$)
7. if CV_{new} lies in C_{free}
   * G.add_vertex(q_{new})
   * G.add_curve(CV_{new})
   end if
   end if
   end for
8. return G

Therefore, in step 6 the local Snakes contour starts with an initial curve CV_{init}(q_{near}, q_{new}) derived from the edge connecting q_{new} and q_{near}. After the deformation, CV_{init} becomes $CV_{new} = v(s)$ where the new deformed curve will be accepted if it belongs to C_{free} (step 7). Hence, the sample rejected by the classical RRT is recovered. However, the mother tree becomes a function of samples (nodes) and Snakes curves instead of edges. It is worth to mention that the Snake algorithm uses terms of the external and internal energies, defined above and detailed in [10] to deform curves generated by the RRT basic algorithm. While, it uses the optimization energy to reduce curves bands of the path. Fig. 2 shows an example of a deformation process of a collided edge (links nodes *i-1* and *i*) as an output of our modified Snakes algorithm. The blue color zone represents the external force generated around obstacles, while the green color is the deformation of the Snakes contour.

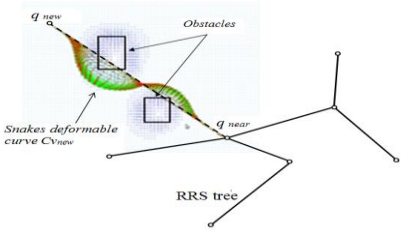


Fig. 2. Integration of deformable Snakes contours into RRT tree.

4 Simulations

In this section we present the evaluation schema used in the simulation analysis. We considered four different environments to evaluate the performance of RRT and RRS (Fig.2). The first one, named Maze-like and it contains a very reduced number of

obstacles, the second has a low number of obstacles (17 obstacles), the third is rather complicated with a high number of obstacles (48 obstacles), the fourth environment have high number of obstacles (potentially cluttered and with potential deadlocks). The obstacles number of this environment varies from 50 to 80 and those obstacles are placed randomly. Moreover, in this environment S_{init} and S_{goal} are randomly generated.

Similar to [17] we evaluate the performance of both algorithms (Fig.4), according to an interval of incremental distance Δq varying from 30cm to 150cm for each environment. Parameters of evaluation are detailed bellow (the average of all parameters is calculated over 100 trials).

- **Percentage of the accepted samples:** the ratio of the number of accepted samples to the number of generated samples;
- **Cost**: the maximal number of iterations to find a solution;
- **Percentage of the completeness probability (convergence)**: a metric that quantify how much the algorithm is capable to find at least one solution;
- **Optimality**: the average of path length;
- **Time of convergence:** time needed to find the solution as the average of Δq;
- **Safety**: estimated by the minimum distance from obstacles, as the average of Δq.

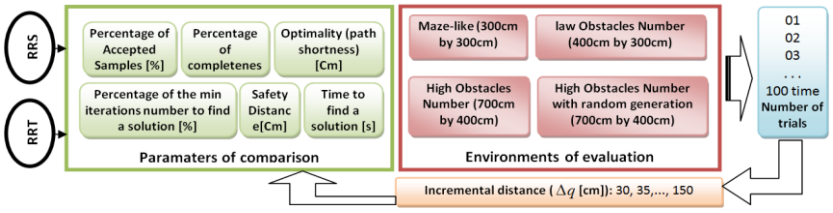


Fig. 3. Simulation parameters used to evaluate RSS and RRT.

Both algorithms have the same test conditions. However, during the random environment evaluation S_{init} and S_{goal} are generated randomly. Conditions are cited bellow:

- We consider a solution is accepted if and only if the returned path links S_{init} and S_{goal} without occurring any collision with the environment.
- If the condition above is not satisfied, we allow both algorithms to reach the max iterations number (5000).
- To minimize the effect of chance, we run each algorithm 100 test.

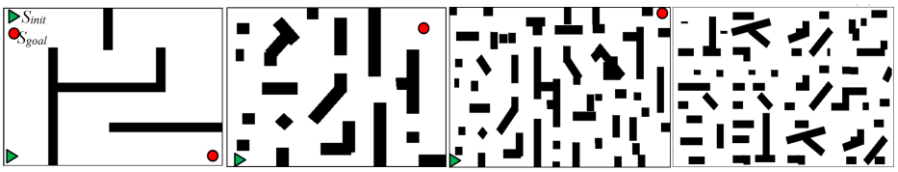
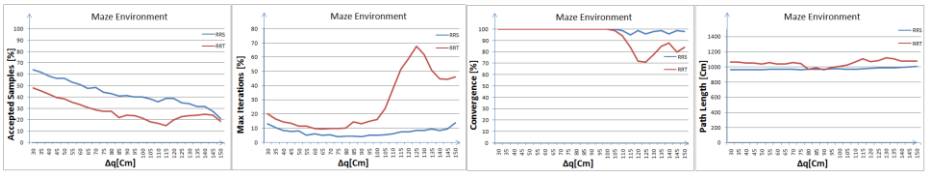


Fig. 4. Simulation Environments, (a) Maze-like (3m*3m), (b) Low dense obstacle (4m*3m), (c) High dense obstacles (7m*4m), (d) A sample of random obstacles (7m*4m).

5 Results and Discussion

In this section we present the results of the RRT and RRS evaluation. Basically this evaluation is for objective to validate the effectiveness of the proposed algorithm. We compare both algorithms' performances according to each test environments.

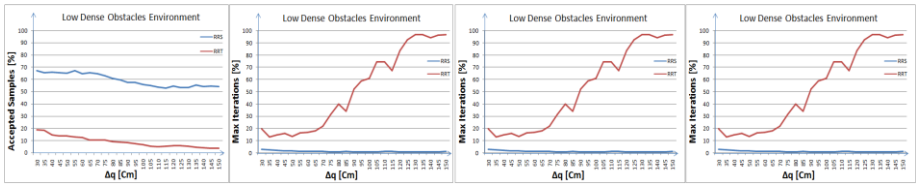


(a) (b) (c) (d)

Fig. 5. Comparison between RRT and RRS in Maze-like environment, (a) Percentage of accepted samples,(b) Percentage of the max number of iterations to find a solution, (c) The percentage of convergences, (d) Length of the path.

Fig. 5 presents a comparison between the RRT and RRS according to a set of incremental distance Δq in Maze-like environment. In Fig.4(a) we can see clearly the high performances of RRS compared to RRT in term of accepted samples, e.g. at 30cm of Δq the RRS reaches a percentage of 65% while at the same Δq the RRT remains below the 50%. However the overall accepted samples performance decreases when the Δq increases. In the same figure the RRS uses less number of iterations to find a solution, it is around 8% from 30cm to 100cm of Δq, while the percentage is double for the RRT; moreover, it increases dramatically in the remaining Δq interval. E.g. at Δq = 65cm the RRS needs 262 iterations (5.32%) while the RRT needs 478 iteration (9.56%).

In term of convergence, the RRS converges in all trials at almost all Δq intervals except at some values from 105cm to 150cm which remains higher than 96%. However the RRT dropped down till 70% at some lengths. Since the environment allows the passage through, only, one way there is no much differences in term of path length, however the RRS returns, slightly, better solutions.



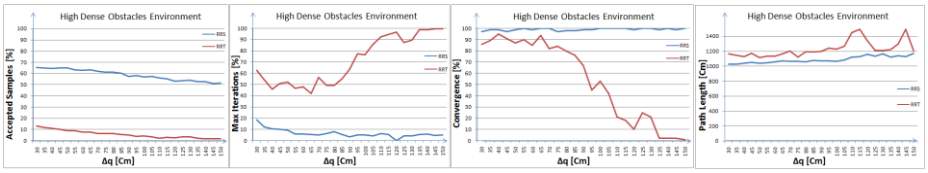
(a) (b) (c) (d)

Fig. 6. Comparison between RRT and RRS in Low Dense Obstacles environment, (a) Percentage of accepted samples,(b) Percentage of the max number of iterations to find a solution, (d) The percentage of convergences, (c) Length of the path.

The results of the Low Dense Obstacles environment are presented in Fig. 6. This environment is low encumbered by obstacles. The RRS performs better in term of acceptance samples ratio, which is around 67.07%, and it keeps a value greater than

50% during the whole Δq interval. However, the best percentage for RRT is 18.8%. Also there are some difficulties for the RRT to converge in all tests (dropped down from 100% to 5% in the last half of Δq interval) while the RRS succeeded to find a solution in all tests (100%). The average of the maximum number of iterations needed to find a solution by RRS was much less than the one of RRT (average percentage at all Δq interval is 2% for RRS, and 52% for RRT). Also solutions given by RRS are shorter; the average of the length path for all Δq values is 732.75cm and 554.98cm for RRT and RRS respectively.

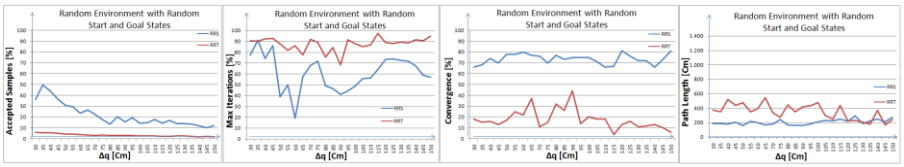
Now we add more challenging conditions by increase the number of obstacles (Fig. 5(c)) as well as the environment size. Always, the RRS presents better performances in all compared parameters (Fig. 7). The accepted samples percentage of the RRS algorithm is more than 50%, while the best value is 13.24% for RRT (at Δq = 30cm). The maximum number of iterations needed to find a solution is small and it is varying from 3.42% (171) to 18.45% (922) for RRS, and from 41.98% (2099) to 99.56% (4978) for RRT. For a higher Δq, RRT uses almost all the available iterations (5000).



(a) (b) (c) (d)

Fig. 7. High Dense Obstacles Environment, (a) Percentage of accepted samples,(b) Percentage of the max number of iterations to find a solution, (c) The percentage of convergences, (d) Length of the path.

Within this environment we can see a large difference between RRS and RRT performances in term of convergence percentage, where RRS seems stable (97% to 100%) during the whole Δq interval, while RRT dropped down dramatically after Δq = 65cm till 1%. Moreover, the RRS algorithm presents good results in term of path-length considering the average at all Δq values; the RRS has 1086.85cm as best solution where the RRT has 1228.83cm



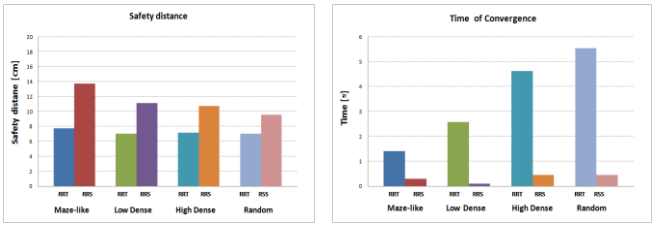
(a) (b) (c) (d)

Fig. 8. Random Environment with random S_{start} and S_{goal}, (a) Percentage of Accepted Samples,(b) Percentage of the max number of iterations to find a solution, (c) The percentage of the Convergences, (d) Length of the path.

In the 4th environment, we keep the same size, however we increase the number of obstacles (vary 50 to 80) and we set the S_{init} and S_{goal} randomly over each trial. Since

this environment is potentially cluttered, it can be very challenging for both algorithms.

Regarding the obtained results (Fig. 8), more or less, we have the same reading of the accepted samples percentage where RRS is always higher. At some Δq is more than 40% however is less than 5% for RRT. For the maximum number of the iteration percentage is almost less than 70% at all Δq values (except the first values) while is more than 70% for RRT. Comparing with the previous environment this percentage might show the level of complexity that this environment has. The convergence percentage is very stable for RRS (75% to 82%) however is much less for RRT (<44%). RRS provide better solution in term of path length and it is stable with all Δq values.



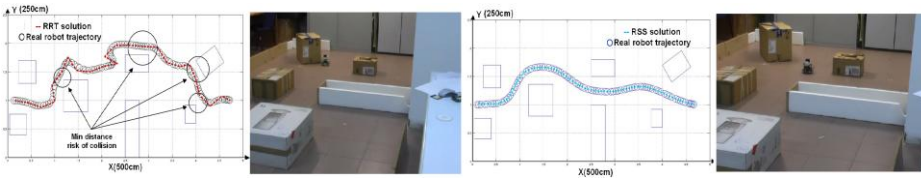
(a) (b)

Fig. 9. Safety distance and time w.r.t each environment, (a) Average of the safety distance, (b) Average of the minimum time needed to find a solution

Fig.9 shows the average of the safety distance and the time of convergence respectively. It is noticeable that the proposed algorithm enhances RRT in term of safety and speed up the process of convergence.

6 Real Experiment

In this section we present an experimental scenario of a Khepera robot following two path solutions (selected randomly from 100 trials) of RRS and RRT in a simple environment of 500cm by 200cm. The robot is equipped with a Hokuyo URG-04LX-UG01 Laser Range Finder and running on-board localization module (developed in our laboratory). This scenario shows the safety of RRS path compared to RRT one. Fig.10 represents both the generated and the real path of the robot. It is obvious that the real path of RRS is much safer than the RRT path; as we can see in the RRT path the robot is very close to obstacles in 4 areas (even after smoothing its original path).



(a) (b) (c) (d)

Fig. 10. Two examples of robot path following, (a) A smoothed RRT solution and the real robot trajectory, (b) A snapshot of the real robot following the RRT path, (c) RRS solution and the real robot trajectory, (d) A snapshot of the real robot following the RRS path.

7 Conclusion

In this paper we proposed a new variant of the RRT algorithm named Rapidly exploring Random Snakes (RRS). The proposed approach combines the RRT and a modified Active Contour Model (Snakes). This later was modified to cope with path planning issues. The RRT probes the environment in a relaxed way, while the Snakes adjusts the candidate paths following three goals: guarantying safety, increasing the probability of accepting samples and shortening the path. Our method performs well in all environments of tests, in comparison to the classical RRT algorithm. Indeed, we demonstrated how the local deformations of RRS's rigid edges, generated by the Snakes, enhance its capabilities of the original sampling algorithm and converge in almost all cases and with less number of 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

This research received funding from the European Community's 7th Framework Programme under grant agreement n. 287617 (IP project ARCAS - Aerial Robotics Cooperative Assembly System).

References

1. LaValle, S. M., Planning algorithms. University of Illinois 1999–2004.
2. J.C. Latombe, "Robot Motion Planning", Norwell, MA: Kluwer, 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.

4. Kuffner, J.J.; LaValle, S.M., "RRT-connect: An efficient approach to single-query path planning," *IEEE International Conference on Robotics and Automation*, vol.2, no., pp.995-1001, 2000.
5. Ryad Chellali, Emmanuel Bernier, Khelifa Baizid, Mohamed Zaoui, "Interface for Multi-robots Based Video Coverage", *International Conference on Human-Computer Interaction*, Vol.6769, 2011, pp 203-210.
6. R. Pepy and M. Kieffer and E. Walter, "Reliably Safe Path Planning Using Interval Analysis", Progress in Industrial Mathematics at ECMI 2008, Mathematics in Industry 2010, pp 583-588.
7. Karaman, S., Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning. IJRR 30(7), 846–894, 2011.
8. Bry, A.; Roy, N., "Rapidly-exploring Random Belief Trees for motion planning under uncertainty," *Robotics and Automation (ICRA), 2011 IEEE International Conference on* , vol., no., pp.723,730, 9-13 May 2011.
9. Garcia, I.; How, J.P., "Improving the Efficiency of Rapidly-exploring Random Trees Using a Potential Function Planner," *44th IEEE Conference on Decision and Control and European Control Conference*, vol., no., pp.7965,7970, 12-15 Dec. 2005.
10. M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models", *Int. J. Computer Vision*, vol. 1, pp.321 -331 1988.
11. Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots," *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," *International Conference on Robotics and Automation, 1989. Proceedings.* , vol., no., pp.316,321 vol.1, 14-19 May 1989.
13. Bhattacharya, P.; Gavrilova, M.L., "Roadmap-Based Path Planning - Using the Voronoi Diagram for a Clearance-Based Shortest Path," *Robotics & Automation Magazine, IEEE* , vol.15, no.2, pp.58,66, June 2008.
14. Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008. *Computational Geometry: Algorithms and Applications*, TELOS, Santa Clara, CA, USA.
15. Kavraki, L.E.; Svestka, P.; Latombe, J.-C.; Overmars, M.H., "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol.12, no.4, pp.566,580, Aug 1996.
16. S. M. Lavalle and J. J. Kuffer, "Rapidly-exploring Random Trees: Progress and prospects", *Workshop on the Algorithmic Foundations of Robotics*, 2000.
17. Burns, B.; Brock, O., "Single-Query Motion Planning with Utility-Guided Random Trees," *International Conference on Robotics and Automation,* , vol., no., pp.3307,3312, 10-14 April 2007.
18. Akgun, B.; Stilman, M., "Sampling heuristics for optimal motion planning in high dimensions," *International Conference on Intelligent Robots and Systems (IROS),* , vol., no., pp.2640,2645, 25-30 Sept. 2011.
19. Samuel Rodriguez, Xinyu Tang, Jyh-Ming Lien and Nancy M. Amato, "An Obstacle-Based Rapidly-Exploring Random Tree," *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, vol.* pp.895-900, Orlando, Florida - May 2006.
20. Chenyang Xu; Prince, J.L., "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol.7, no.3, pp.359,369, Mar 1998.
21. Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT press.
22. Khelifa Baizid, PhD thesis (2011) "*Multi-robots Tele-operation Platform: Design and Experiments*" Italian Institute of Technology & University of Genova, Italy.