RRT*

CAS CS599 N1 - Milestone 1: Project Proposal

Jonas Raedler, Prithviraj Khelkar, Shiven Sharma

1. Context of Algorithm in Robotics

The robotics computational pipeline consists of three crucial components: perception, mapping and localization, and motion planning and control. After gaining an insight into its current environment (perception) and determining where it is in relation to this environment and its obstacles (mapping and localization), the robot needs to calculate what path it should take to reach its goal state in a way that prevents any collisions (motion planning and control).

In order to achieve this task of determining such a safe path in an efficient way, many algorithms have been proposed. One big subgroup of these motion-planning algorithms are ones based on sampling, which has been found to be a lot more efficient than just outright calculating the best path (Lamiraux and Laumond). While the Rapidly-exploring Random Tree (RRT) algorithm quickly emerged as a probabilistically complete and fast - especially for single-query problems - sampling-based algorithm, it often provided suboptimal, inefficient paths (Elbanhawi and Simic).

To remedy this issue, RRT* has been developed, a variation that provides asymptotic optimality. Currently, RRT* is considered to be a state-of-the-art motion planning algorithm.

2. Implementation of Algorithm

The RRT* algorithm essentially works in the same way the normal RRT algorithm does. The key differences that provide the asymptotic optimality are the consideration of path costs when a new sample is added to the existing tree and the rewiring step that calculates whether this new sample, as a potential parent node for neighboring nodes, reduces the path cost for these nodes. The original RRT* paper provides a pseudocode implementation of the algorithm (Karaman and Frazzoli) and the Open Motion Planning Library (OMPL) provides a C++ implementation.

3. Computing Systems

RRT* is usually run on a fairly typical system. In our search, we have found that the algorithm is often run on an Intel Core i5 processor with 2 to 3 GHz and 4+ GB of RAM (Ding et al., Qureshi and Ayaz, Karaman and Frazzoli).

4. Limitations

As previously mentioned, the RRT* algorithm provides probabilistic completeness and asymptotic optimality, both of which are very desirable properties in motion planning. However, there is a cost for these additional benefits.

The first big cost is that RRT* has a significantly lower convergence rate than the original RRT algorithm. The two additions that set RRT* apart from RRT add a lot of computational complexity: in order to find the most optimal path, RRT* performs a nearest neighbor search whenever a new point has been sampled in order to determine which existing path provides the lowest path cost to this new sample. This, however, means that there is an additional nearest neighbor search for every sample that is added, which leads to a 49% increase in latency from this particular calculation in comparison to the original RRT algorithm (Bakhshalipour et al.). Furthermore, the latency of RRT* is further increased due to the many collision checks that are performed. Essentially, whenever the path cost of a nearest neighbor to a new sample is considered, RRT* also checks whether an edge between these two nodes would lead to a collision. These two computational additions to the RRT algorithm lead to a significantly lower convergence rate in RRT*: up to 8x slower in the experiments conducted by the authors of the RTRBench paper (Bakhshalipour et al.).

This finding of a slower convergence rate is further supported by Ding et al. and Qureshi and Ayaz, who also bring up the point that the slower convergence rate can further be explained by the large amount of samples that RRT* requires in order to determine the optimal path (Ding et al., Qureshi and Ayaz). This is especially true in high-dimensional spaces, which require a lot of samples for full exploration, as RRT* needs to explore a larger and more complex search space in such environments.

This increased number of required samples also has a large negative effect on the memory usage of the RRT* algorithm (Ding et al., Qureshi and Ayaz). These high memory requirements appear even worse when one considers the fact that a lot of the sampled nodes do not really contribute to finding an optimal path, meaning that the node utilization rate is fairly low (Ding et al.).

Thus, RRT* has two significant limitations: a slow convergence rate (i.e. high latency) and high memory requirements.

5. Possible Benefits from Addressing Limitations

Addressing the slow-convergence-rate-limitation of the RRT* algorithm would have the somewhat obvious benefit of a faster robot. The less time the robot spends on calculating a good, collision-free path, the lower the overall latency in the robotic computational pipeline (of which motion planning is a significant component), which in turns means that it will be easier for robots to perform tasks in “real-time”. This would have the result that we are a step closer to implementing robots in our daily lives (after all, making decisions in real-time is a necessity for most autonomous robots), which could be a huge benefit to society, as the advantages of robots could then be more reliably offered to humans.

Moreover, faster and more efficient motion planning would also directly impact power consumption. With reduced computational demands in RRT* and a higher convergence rate, robots would consume less energy during the path planning process, which would, for battery-powered robots, result in an extended operational time and a reduction in the frequency of recharging, again resulting in better circumstances for using robots in the real world.

Last but not least, if the memory usage limitation could be addressed, robots/systems that use the RRT* algorithm for motion planning would require not only a less expensive setup, but also a smaller overall memory, allowing RRT* to be used effectively on smaller robots that are incapable of using extensive amounts of memory.

All in all, addressing the limitations of the RRT* algorithm could significantly help in making robots more viable for real-world applications, which would not only lead to a drastic increase in innovation in the robotic research area, but we would also be closer to using robots to assist humans.

References

Bakhshalipour et al., *RTRBench: A Benchmark Suite for Real-Time Robotics*. https://www.pdl.cmu.edu/PDL-FTP/associated/rtrbench-ispass22.pdf

Karaman and Frazzoli, *Sampling-based Algorithms for Motion Planning*. https://arxiv.org/pdf/1105.1186.pdf

Ding et al., *An improved RRT* algorithm for robot path planning based on path expansion heuristic sampling*. https://www.sciencedirect.com/science/article/pii/S1877750322002964

Qureshi and Ayaz, *Potential functions based sampling heuristic for optimal path planning*. https://link.springer.com/article/10.1007/s10514-015-9518-0?utm_source=getftr&utm_medium=getftr&utm_campaign=getftr_pilot

Elbanhawi and Simic, *Sampling-Based Robot Motion Planning: A Review*. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6722915

Lamiraux and Laumond, *On the expected complexity of random path planning*, https://ieeexplore.ieee.org/document/509170

OMPL, *Open Motion Planning Library*. https://ompl.kavrakilab.org/classompl_1_1geometric_1_1RRTstar.html#gRRTstar

Individual contributions

We divided up the work for this milestone into sections.

Jonas worked on section 1: Context of Algorithm in Robotics, on section 4: Limitations, and 5: Possible Benefits from Addressing Limitations.
Prithvi worked on section 4: Limitations and on section 5: Possible Benefits from Addressing Limitations.
Shiven worked on section 3: Implementation of Algorithm and section 4: Computing Systems.

Furthermore, all three members looked for relevant papers in order to find information for their sections. These papers were shared and main points were highlighted that could be of interest to the other members.