Turtlebot Path Planning using Rapidly-Exploring Random Trees (RRT) and its Variants

Gourab Ghosh Roy, Raabid Hussain and Kibrom Berihu Girum

May 17, 2016

Outline

- Introduction
- Path Planning Methods
  - RRT
  - Multiple-Restart RRT
  - RRT*
  - RRT-Connect
- Collision Avoidance
  - Dilation
  - RRT and Brushfire
- Experiments and Results
- Conclusion

1. Sampling-based Path Planning

- Path planning is an integral aspect of robotic applications
- Sampling-based algorithms consider few samples instead of searching the entire environment
- Optimal path found by sequence of connected free samples from start to goal

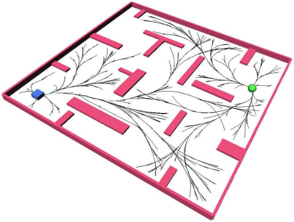


Figure: Sampling-based Path Planning

Image: Randomized Kinodynamic Planning, LaValle

2.1.1 RRT Algorithm

- Proposed by LaValle and Kuffner
- Tree exploring in the configuration space
- Two main steps - Build and Extend

Algorithm 1 Build (q_{init}, N)

$V \leftarrow \{q_{init}\}$
$E \leftarrow \phi$
$T = (E, V)$
for $i = 1$ to N **do**
    Generate q_{rand}
    Extend RRT (T, q_{rand})
end for
return T

2.1.2 RRT Algorithm

Algorithm 2 Extend (T, q)

$q_{near} \leftarrow$ Nearest Neighbor (T, q)
Get q_{new} by moving ϵ from q_{near} to q
if path from q_{near} to q_{new} is free **then**
    $V \leftarrow V \cup \{q_{new}\}$
    $E \leftarrow E \cup \{(q_{near}, q_{new})\}$
end if

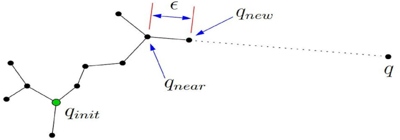


Figure: RRT Extend Operation

Image: RRT-Connect, Kuffner

2.2 Multiple-Restart RRT

- Proposed by Luna et al
- Path length not optimal in standard RRT
- Run RRT multiple times and select the one with shortest path

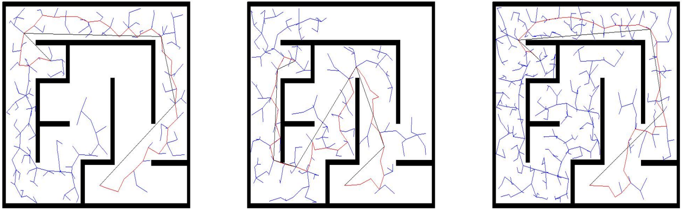


Figure: MRRT Algorithm

2.3.1 RRT*

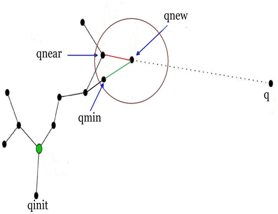
- Proposed by Karaman and Frazzoli
- Objective is to reduce path length/cost
- Provable optimality properties
- Differs from RRT in the Extend operation

2.3.2 RRT* Algorithm

Algorithm 3 RRT* Extend (T, q)

$q_{near} \leftarrow$ Nearest Neighbor (T, q)
Get q_{new} by moving ϵ from q_{near} to q, $q_{min} \leftarrow q_{new}$
$Q_{near} \leftarrow$ Neighbors (T, q_{new}, d)
for q_n in Q_{near} **do**
    if $\text{Cost}(q_n) + \text{dist}(q_n, q_{new}) < \text{Cost}(q_{new})$ **then**
        $q_{min} \leftarrow q_n$
    end if
end for
$V \leftarrow V \cup \{q_{new}\}$, $E \leftarrow E \cup \{(q_{min}, q_{new})\}$

2.3.2 RRT* Algorithm (Continued)



Algorithm 3 RRT* Extend (T, q)

for q_n in $Q_{near} \setminus q_{min}$ **do**
    if $\mathrm{Cost}(q_{new}) + \mathrm{dist}(q_{new}, q_n) < \mathrm{Cost}(q_n)$ **then**
        $E \leftarrow E \setminus \{(Parent(q_n), q_n)\}, E \leftarrow E \cup \{(q_{new}, q_n)\}$
    end if
end for
return T

2.4.1 RRT-Connect

- Proposed by Kuffner and LaValle
- Two trees at start and goal configurations
- Faster in finding a path

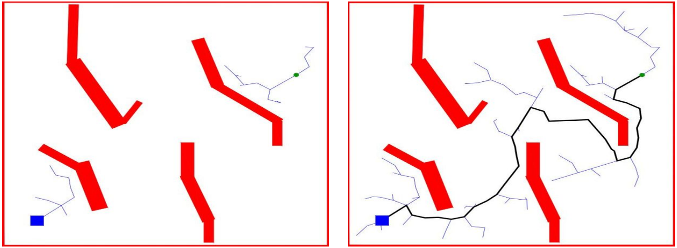


Figure: RRT-Connect Algorithm

Image: RRT-Connect, Kuffner

2.4.2 RRT-Connect Algorithm

Algorithm 4 RRT-Connect Planner(q_{init}, q_{goal})

Initialize $T_{q_{init}}$, $T_{q_{goal}}$
for $i = 1$ to N **do**
  Generate q_{rand} , Extend RRT ($T_{q_{init}}, q_{rand}$)
  if Connect RRT ($T_{q_{goal}}, q_{new}$) $=$ Success **then**
    return PATH($T_{q_{init}}$, $T_{q_{goal}}$)
  end if
  SWAP ($T_{q_{init}}$, $T_{q_{goal}}$)
end for

Algorithm 5 Connect (T, q)

while No Collision **do**
  EXTEND(T, q)
end while

3. Collision avoidance

- Maps have an approximated view of obstacles
- Even with perfect maps, the planner does not take into account the robot size
- Even slight collisions with obstacles need to be avoided

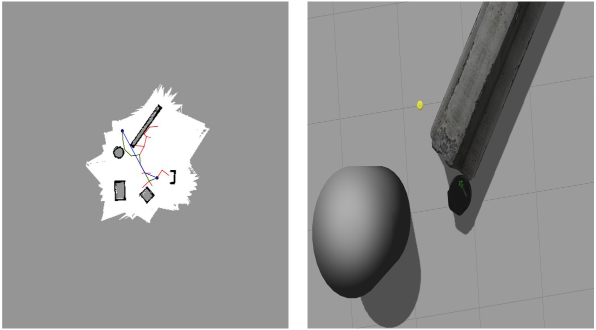


Figure: Collision

3.1 Dilation

- Size of the obstacles increased by application of morphological (dilation) operation
- Size of dilation element depends on size of the robot

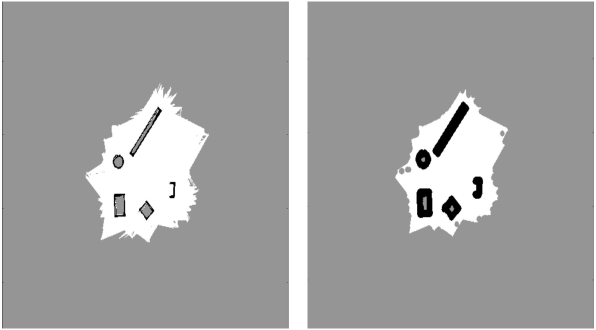


Figure: Left: Input map, Right: Map after dilation operation

3.2 RRT and Brushfire

- Brushfire algorithm gives the potential map of the environment
- In the RRT Extend operation, q_{rand} is replaced by a configuration with the lowest potential in its neighborhood
- During smoothing, a direct path is only considered if all points on it have a potential lower than a threshold

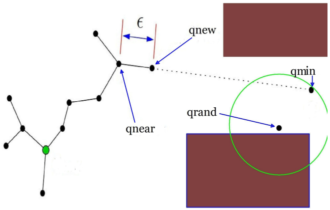


Figure: RRT with Brushfire

4.1 Experiments and Results

Turtlebot motion in a simulated environment consists of 3 parts

- Mapping
- Planning
- Movement

4.2 Mapping

- Create a world in Gazebo simulator
- Turtlebot teleop to move the turtlebot using keyboard commands
- Using Gmapping and Rviz, create and save a 2-D occupancy grid map (as yaml and image files)
- Read the map from ROS map_server and convert to a 2D matrix

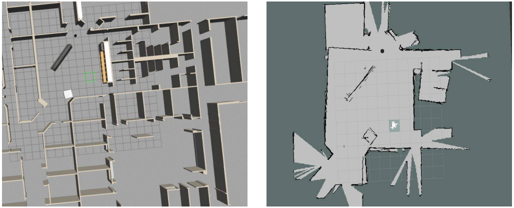


Figure: Left: Gazebo world, Right: Building map using Gmapping in Rviz

4.3 Turtlebot Driver

- Subscribe to the */odom* topic for robot position
- Compute linear and angular velocity to go to next waypoint
- Publish velocity commands to the */mobile_base/commands/velocity* topic

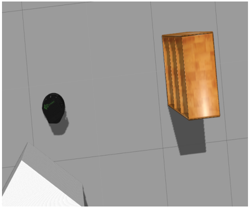


Figure: Turtlebot Movement

4.4 Testing

- Two environments used for presenting results

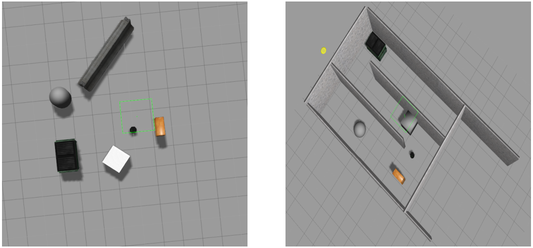


Figure: Left: Environment 1, Right: Environment 2

4.5 Planning Time

Environment	RRT	MRRT	RRT*	RRT-Connect
Environment 1	0.799	1.69	0.711	**0.615**
Environment 2	9.35	23.29	6.88	**5.71**

Table: Table showing Planning times in seconds for different variants of RRT for same start and goal in all cases (averaged over 5 runs)

4.6 Path Length

Environment	RRT	MRRT	RRT*	RRT-Connect
Environment 1	64.03	59.05	**56.35**	59.04
Environment 2	319.95	316.46	**295.523**	311.54

Table: Table showing Path length for different variants of RRT for same start and goal (averaged over 5 runs)

4.7 Collision Avoidance with RRT and Brushfire

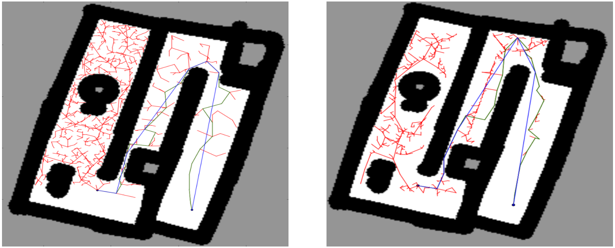


Figure: Left: RRT, Right: RRT with Brushfire

4.8 Demo

5. Conclusion

- Different variants of RRT implemented and compared in terms of planning efficieny
- Sampling based method combined with potential based method for obstacle avoidance
- Turtlebot movement in different simulated environments using the planned path

References

- S. M. LaValle and J. J. Kuffner. *Randomized Kinodynamic Planning*. International Journal of Robotics Research, 20(5):378400, May 2001.
- J. J. Kuffner and S. M. LaValle. *RRT-connect: An efficient approach to single-query path planning*. In Proceedings IEEE International Conference on Robotics and Automation, pages 9951001, 2000.
- S. Karaman and E. Frazzoli. *Incremental sampling-based algorithms for optimal motion planning*. In Robotics: Science and Systems (RSS), Zaragoza, Spain, June 2010.
- B.J. Verwer, P.W. Verbeek and S.T. Dekker. *An Efficient Uniform Cost Algorithm Applied to Distance Transforms*. IEEE Transactions on Pattern Analysis and Machine Intelligence archive Volume 11 Issue 4 Page 425-429, April 1989.
- R. Luna, I. Sucan, M. Moll and L. Kavraki. *Anytime solution optimization for sampling-based motion planning*. In IEEE Int. Conf. Rob. Aut. , pages 5068-5074, 2013.

Thank You