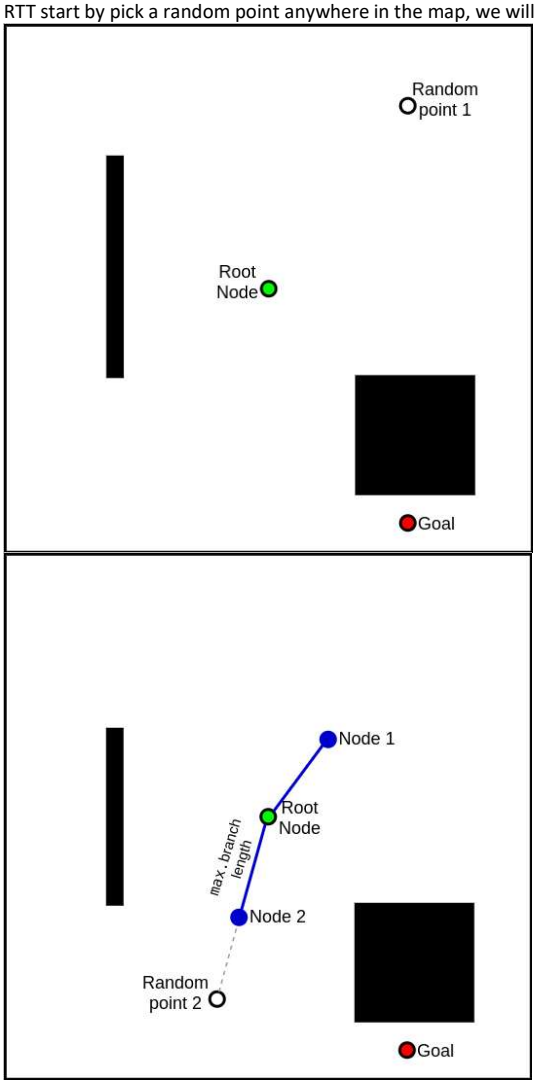
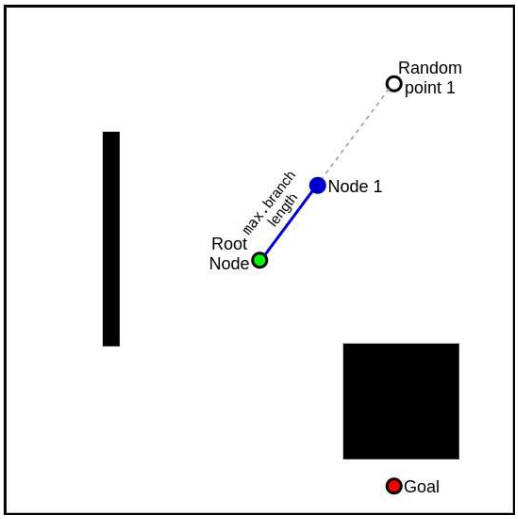
Rapid-Exploring Random Tree (RTT)

Saturday, January 28, 2023 2:48 PM

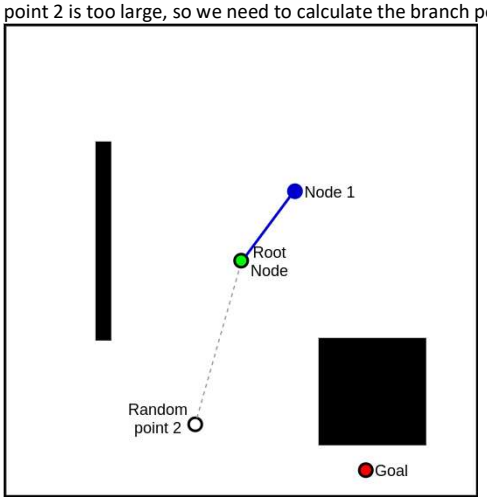
The map shown below displays a white area, the map's free space, which is limited by surrounding frame, the map boundaries. There are two obstacles present in the map which are represented by black rectangles. RTT initiates from the robot's current pose (let's call that position root node), and the goal is explicitly known.
RTT start by pick a random point anywhere in the map, we will call it random point 1

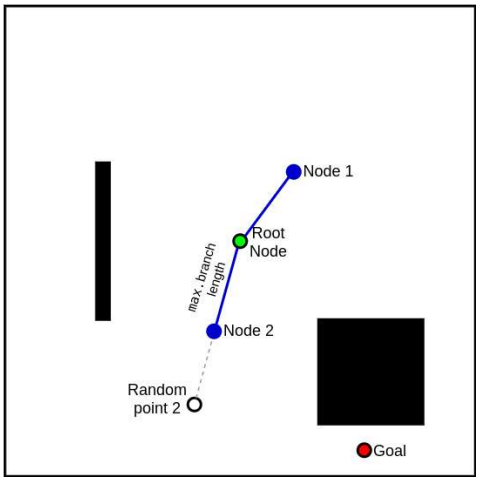


Then we search for the closest node to the random point (in this case the root node), once that closest node is found we connect it with the random point, we can see that random point is too far from root node, so we need a variable named "Max Branch Length", which will help us to create a point with the same direction as the previous random point but a "Max Branch Length" magnitude. Then we connect that branch point node to the root node (closest node in this case) .



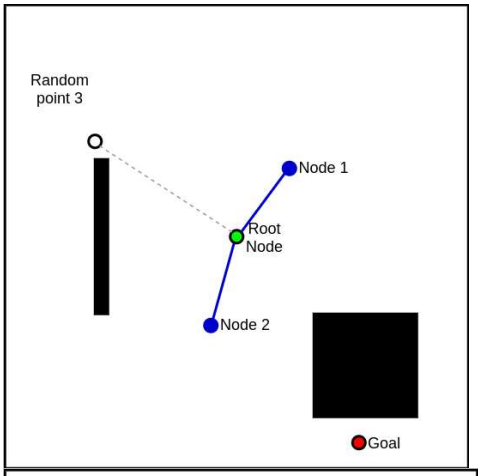
The algorithm continues generating other random point in the map, then we need to find the closest node to the random point (in this case root node), as we see that distance from root node to random point 2 is too large, so we need to calculate the branch point limited by the "Max Branch Length"

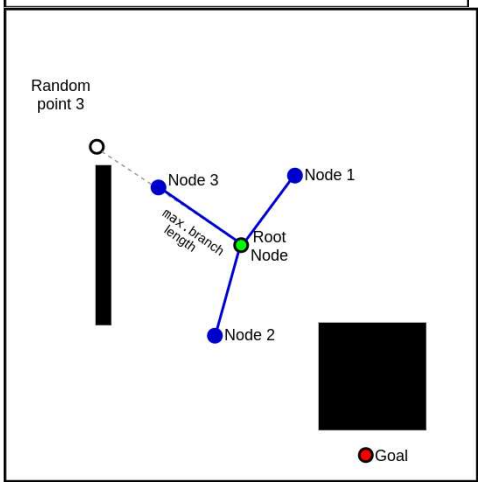




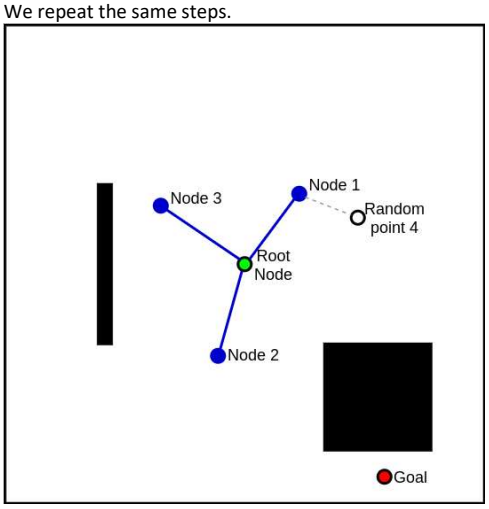
Then we add another random point, we repeat the same steps:

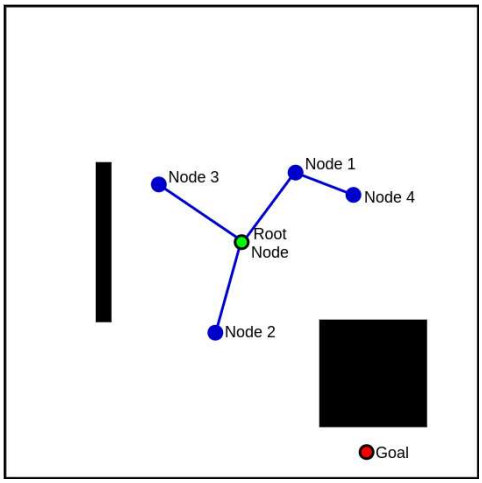
1) Find closest node to the random point
2) Verify if the distance from closest node to random point is greater than "Max Branch Length"
3) If it's greater, generate the branch point with the same orientation as the line between both nodes, otherwise use the random point as branch point
4) Repeat steps



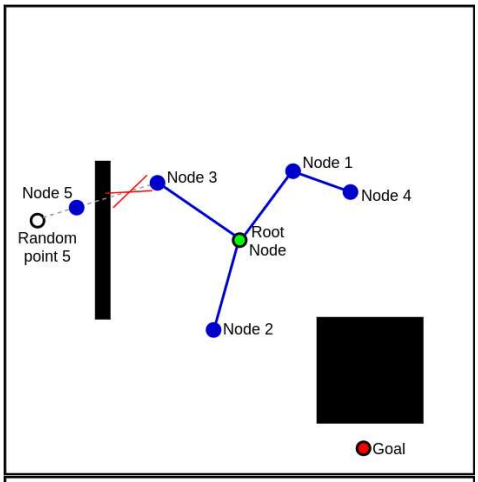


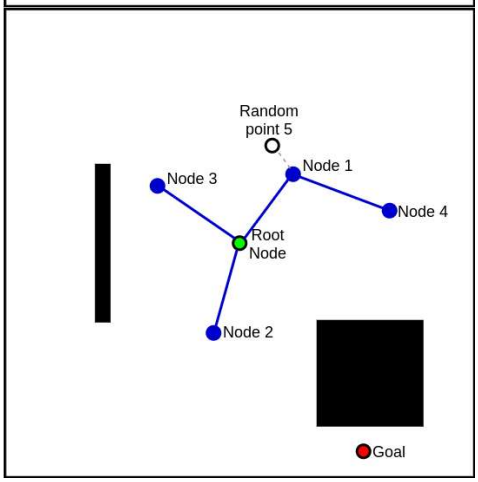
We repeat the same steps.



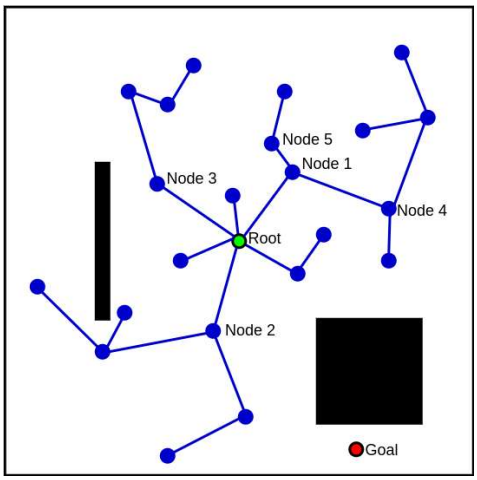


What happens if the line that connects the random point with the closest node crosses an osbtacle, we have to discard that node and cancel the current cycle iteration.

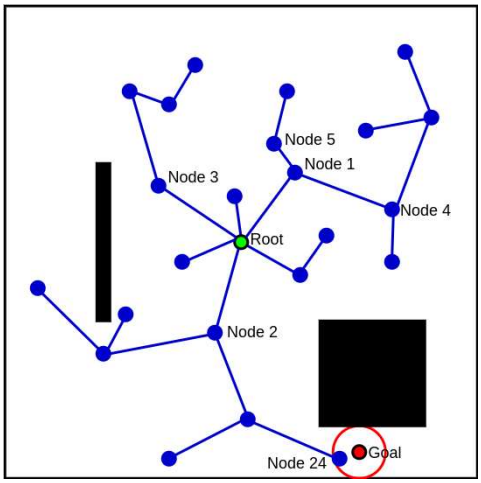




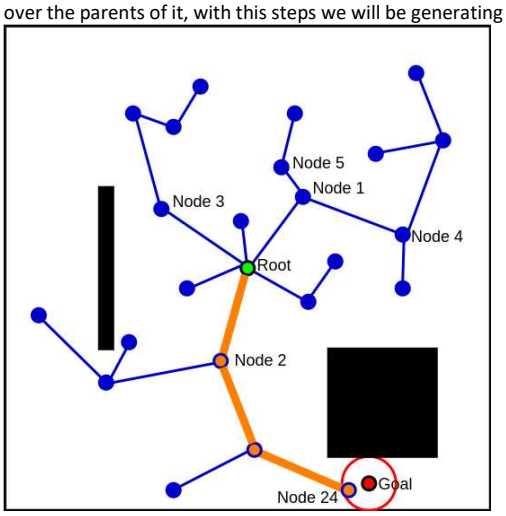
Then we continue exploring:



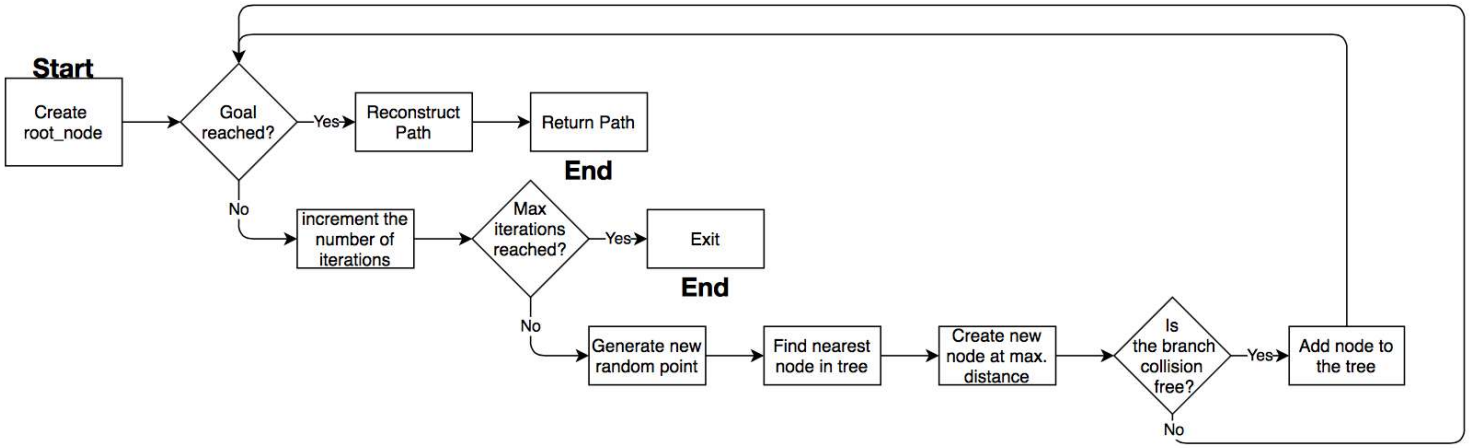
The expansion of this algorithm finishes until a connection to the goal node is found or a maximum number of iterations is exceed. Since its almost impossible that a random point can be generated in the same coordinates as the goal node, we need to define certain tolerance to verify if a branch point is near to it.



The final step is to build the path, so we need to get the closest node to the goal node, then we iterate over the parents of it, with this steps we will be generating a path. Finally we need to reverse the path.



This algorithm can be described in the following diagram:


Start
Create root_node
Goal reached?
Yes
Reconstruct Path
Return Path
End
No
increment the number of iterations
Max iterations reached?
Yes
Exit
End
No
Generate new random point
Find nearest node in tree
Create new node at max. distance
Is the branch collision free?
Yes
Add node to the tree
No