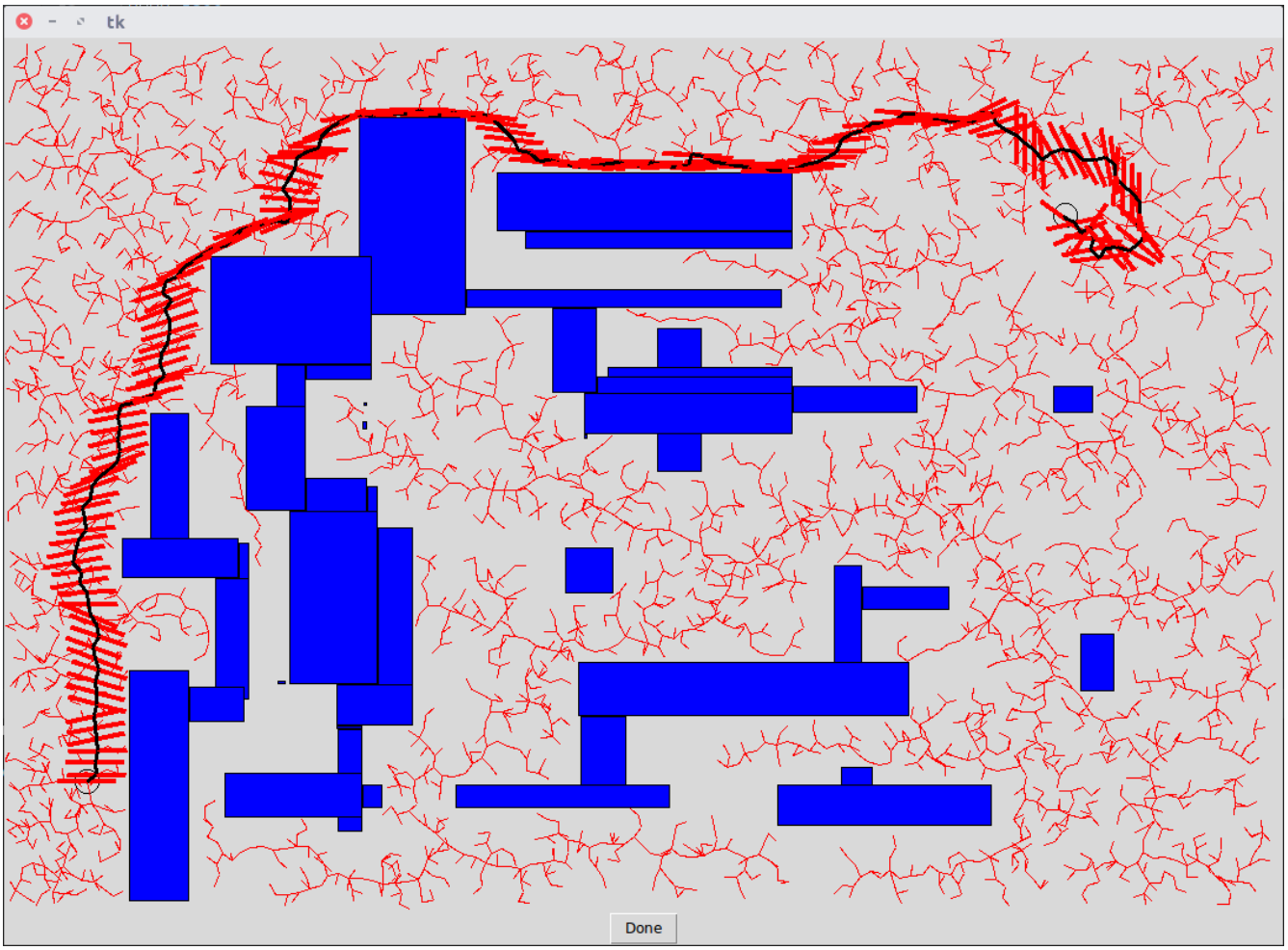
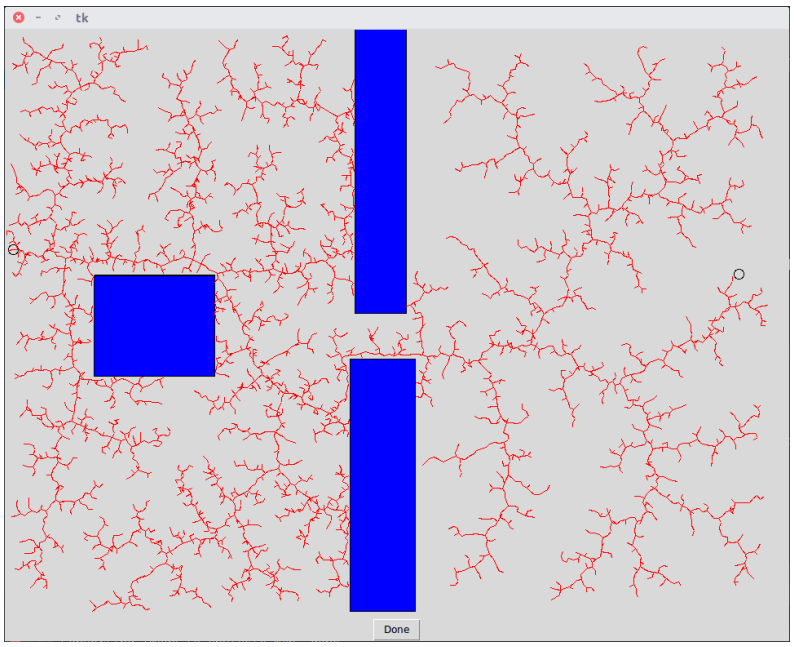
Yann Vonlanthen, 260808862
McGill, October 2017

# COMP 417 – Assignment 2
# Rapidly-exploring Random Trees

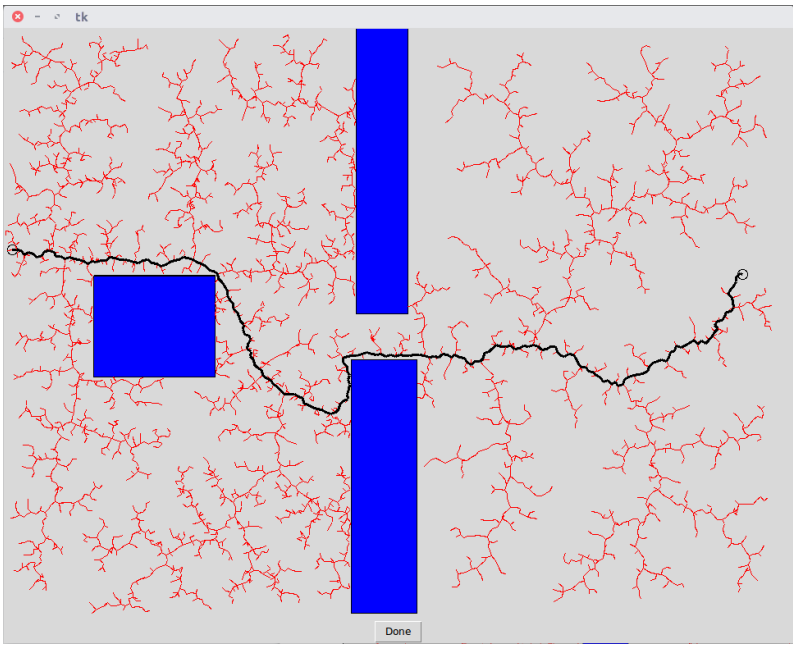


Yann Vonlanthen, 260808862
McGill, October 2017

# Part 1a – Uniform Sampling for Point Robot

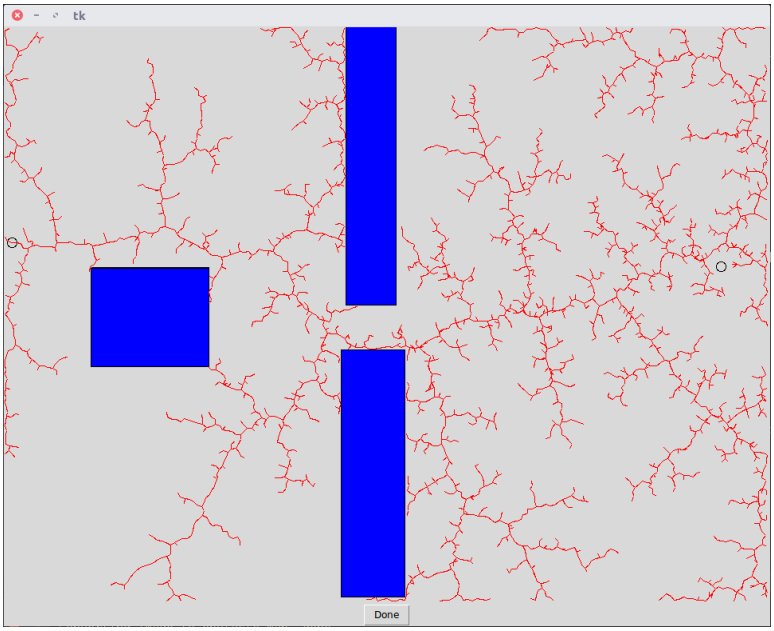



Planning Phase

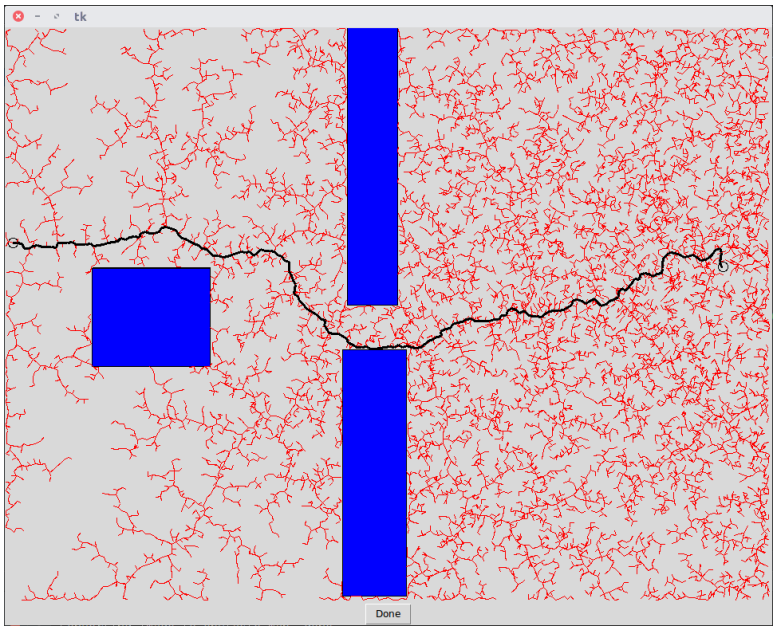



Final Path to destination

# Part 1b – Gaussian Sampling for Point Robot




Planning Phase




Final Path to destination

Yann Vonlanthen, 260808862
McGill, October 2017

We see that a uniform distribution creates an evenly spread out RRT, that is the tree expands in every direction of the map in the same way. Sometimes it will take the tree some time to pass the obstacle and it will thus expand further on the left side first.

A Gaussian distribution on the other side will tend to grow the tree quicker towards the goal state, while a large enough variance makes sure that obstacles can still be overcome. Note that I ran the experiment with the variances that were given (MAX_X/2, MAX_Y/2).

## Part 1c – Plots for Point Robot

As expected the number of iterations increases with decreasing step size. The increase is drastic for small step sizes. The high standard deviation and the experiences made while running the script indicate that this is most likely due to the presence of one or more outliers that took much longer to converge to the target.

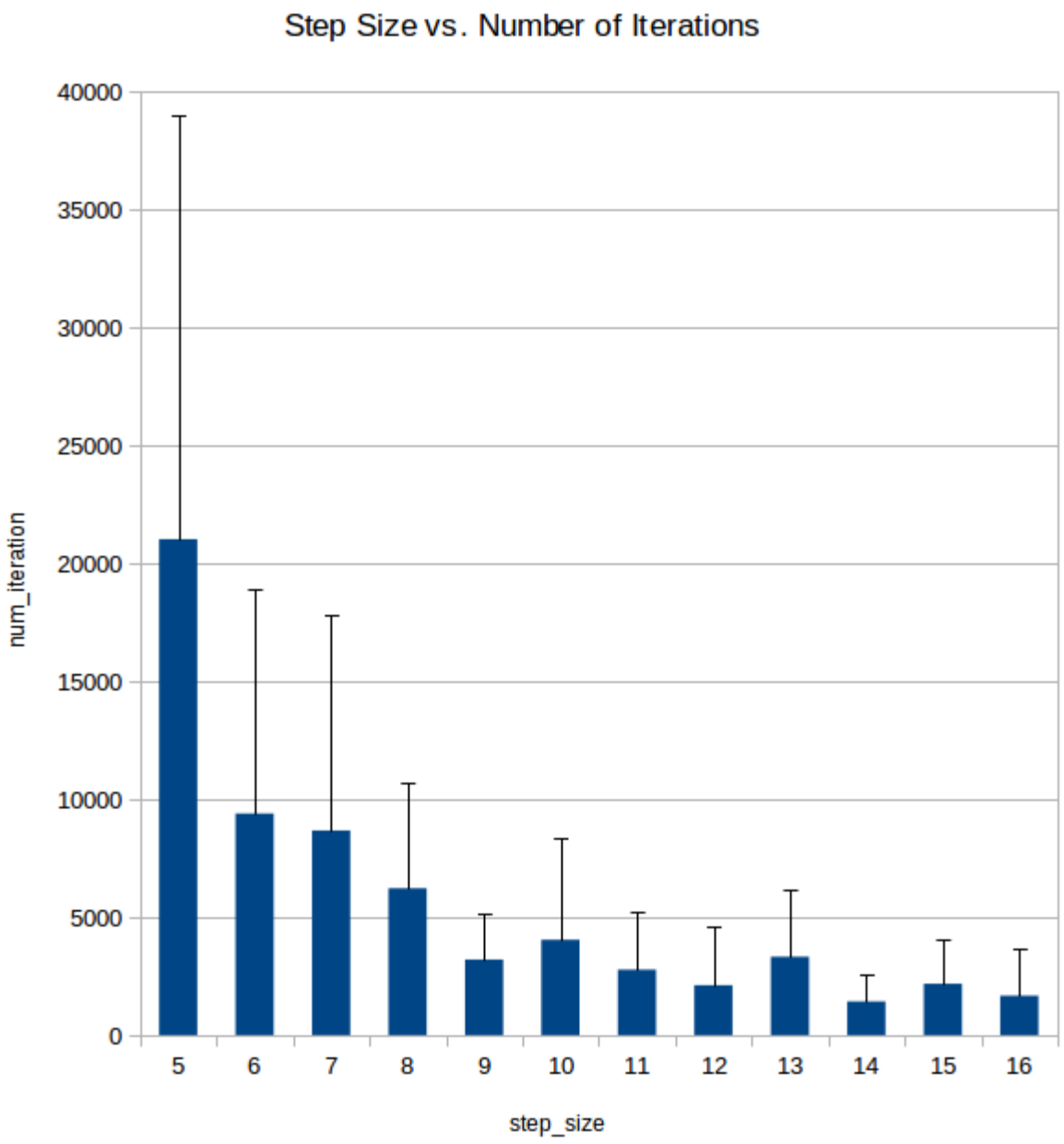
We can also remark that the standard deviation decreases at first but then stays around the same for larger step_sizes. This indicates that one can always be “lucky” or “unlucky” and that a prediction of the running time is very hard.

On the other hand the path size does not seem to increase with increasing path size. This is the opposite of what I would have expected. It is possible that it may be due to a small sample size.

It is also worth noting that the implementation for finding the closest vertex is linear in the number of vertices's, and thus slows down the algorithm considerably as the number of vertices's increases. Maybe if we had position sensitive hashing our conclusion would be different, but as is, choosing a small step size does not seem like a good choice.

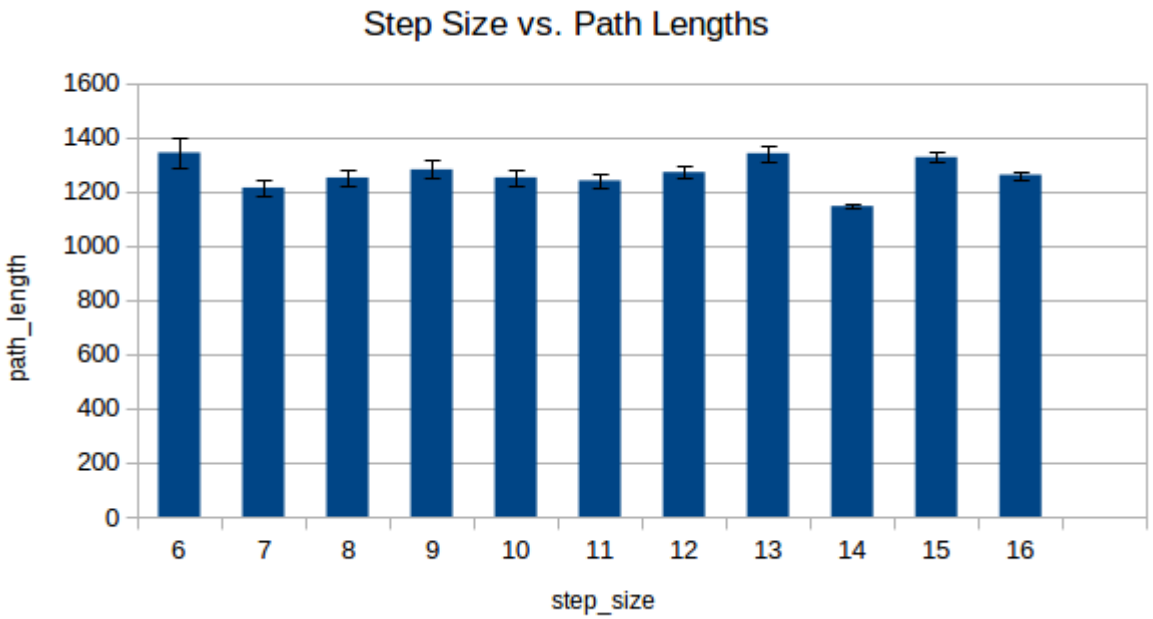
Finally, looking at the graphs a step_size of 14 seems like the best choice in my case, but changing the sample size and allowing for larger step_size would have a big effect on this choice.

Yann Vonlanthen, 260808862
McGill, October 2017




Note that I didn't include larger step_sizes since otherwise the algorithm tends to skip over obstacles that should hinder its movement in reality, even though I added an additional condition for collision detection. It can be noted however, that increasing the step_size will further reduce the number of iterations needed.

Yann Vonlanthen, 260808862
McGill, October 2017




## Part 1d – Additional Remarks

A heuristic that could be added is to decrease the standard deviation of the Gaussian sampling the closer the RRT gets to the point. This can be dangerous, as the RRT can get “trapped” close to the point in a U shape, and would take a very long time to find its way out.

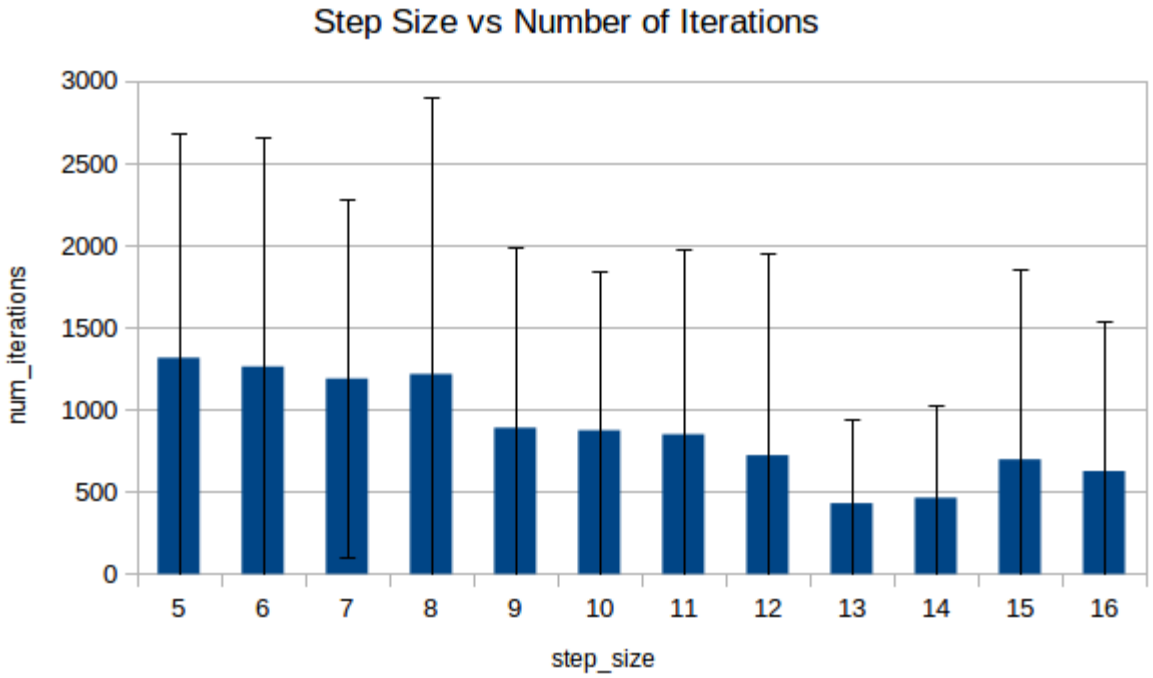
Another (maybe somewhat better) heuristic would be to take the goal / target as sample once in a while, for example once every ten times. This leads leads to much faster convergence in certain cases, but can also have negative effects that have to be weighted.

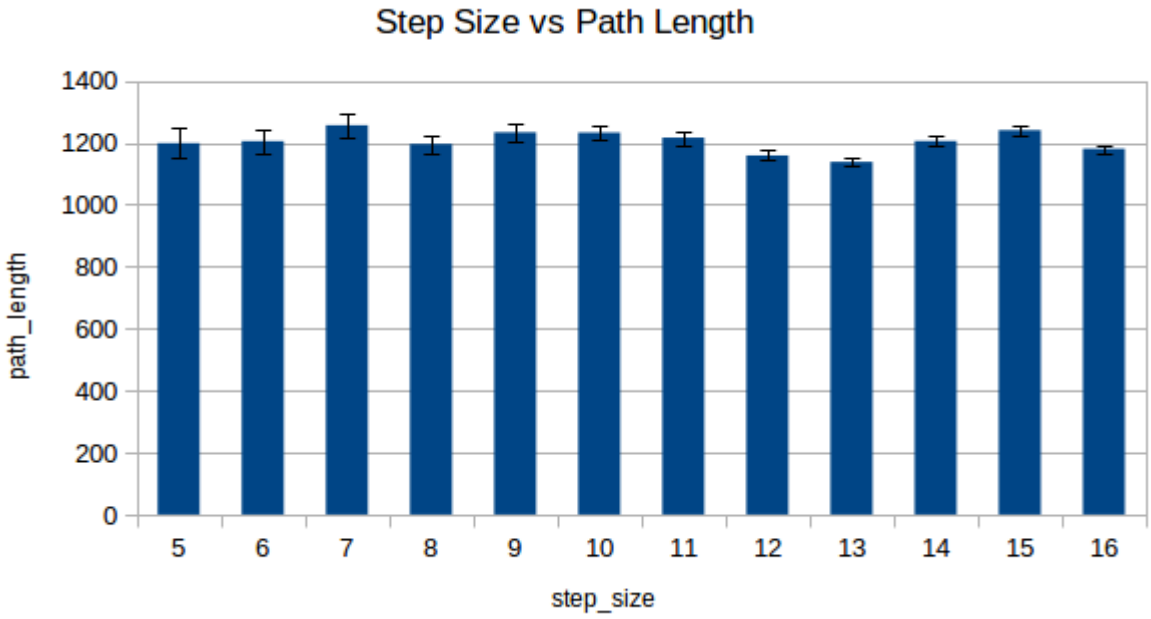
Finally at every iteration we could also look for a straight way to the target position. This means that once line of sight is established the algorithm would stop instantly.

I implemented the second suggestion and added the same graph as for 1c for completeness. Since the algorithm converges much faster I ran 20 trials for every step_size.

First we see that the path length is almost exactly the same as without the heuristic. But the iteration number gets decreased tenfold or more! The variance remains large since the RRT sometimes choses the longer way that passes below the obstacles, but all in all the speed-up is non-negligible. This would have to be tested on many more maps though, in order to see whether it is a valid strategy!

Yann Vonlanthen, 260808862
McGill, October 2017



Yann Vonlanthen, 260808862
McGill, October 2017

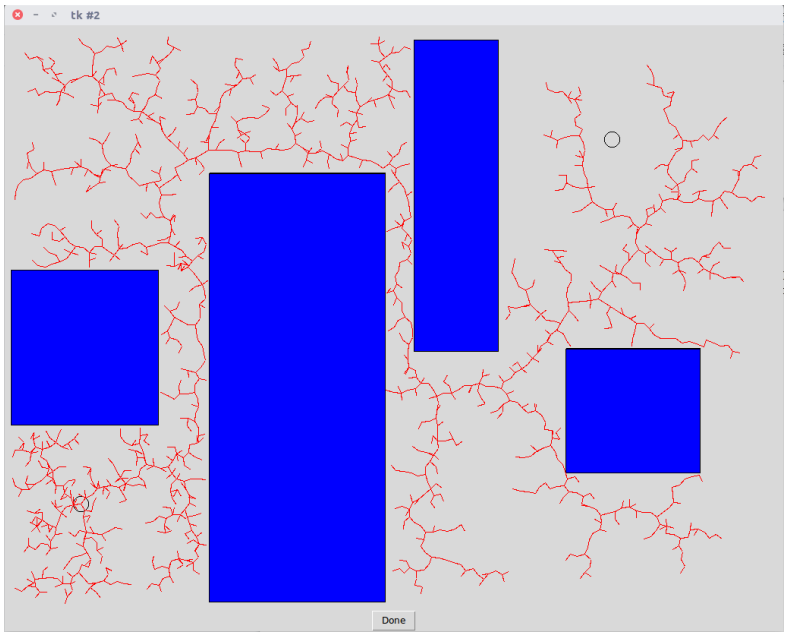
# Part 2 – Line Robot

Since it was not specified I used a step_size of 10. This allowed for pretty fast convergence, and resembles the pictures that we were given. I ran the experiment for lines of size 10 to 190, where 10 behaves almost like a point robot and 190 is almost the biggest size for a line to be able to navigate through the maze.
Note that since I only check that the line is not inside an object in the final state, it can happen that a path is actually not practically do-able, though this shouldn't happen very often.

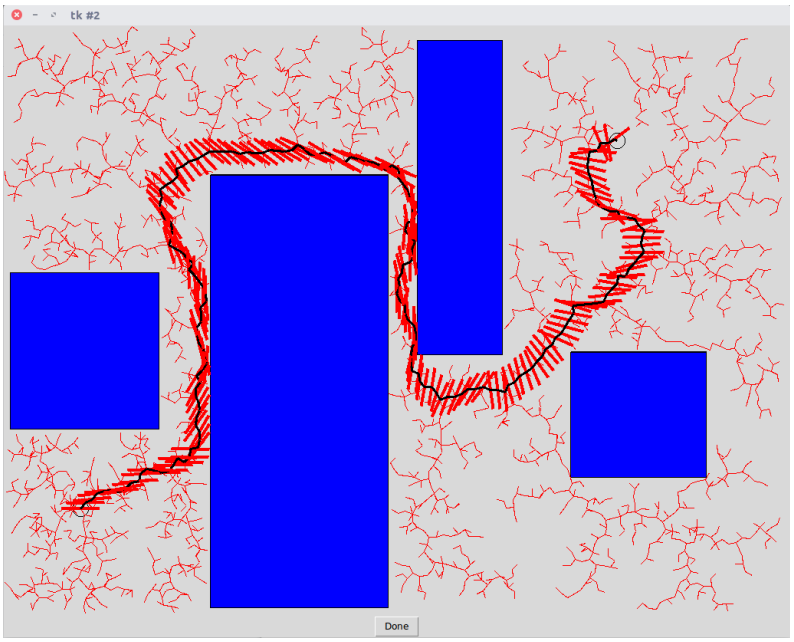
# Part 2a

The following pictures depict the movement of a line of size 50 and a step_size of 10 using uniform sampling.

Path Planning:




Final Path to destination:



Yann Vonlanthen, 260808862
McGill, October 2017

## Part 2b

We see that the number of iterations needed for the RRT to find the target increases very clearly with the size of the line. If we look closer we can also observe that there are two major bumps, first when passing from 50 to 70 and the second one when going from 170 to 190. I believe they can be explained by the topology of the map, since some turns and narrow passages can be passed in almost every orientation until a certain size, but get harder after a certain threshold is crossed.

A line of size 10 seems to behave almost like a point robot, with iteration numbers close to those observed under 1c) (Remember that for this part I used a step_size of 10), where as a line of size 190 takes around five times longer to converge, and has a much larger standard deviation on top of that. Finally I'd like to note that my algorithm makes some simplifications when checking if a state can be reached collision free, since it only checks the new position. I believe that without those convergence would be even harder for a line of large size.