Turtlebot3 Navigation Logic

The turtlebot creates a subscription to status, odom and map, and has a publisher to publish goal poses. Once the map is ready, the robot sends the its own position as the first waypoint to start off the feedback subscription.

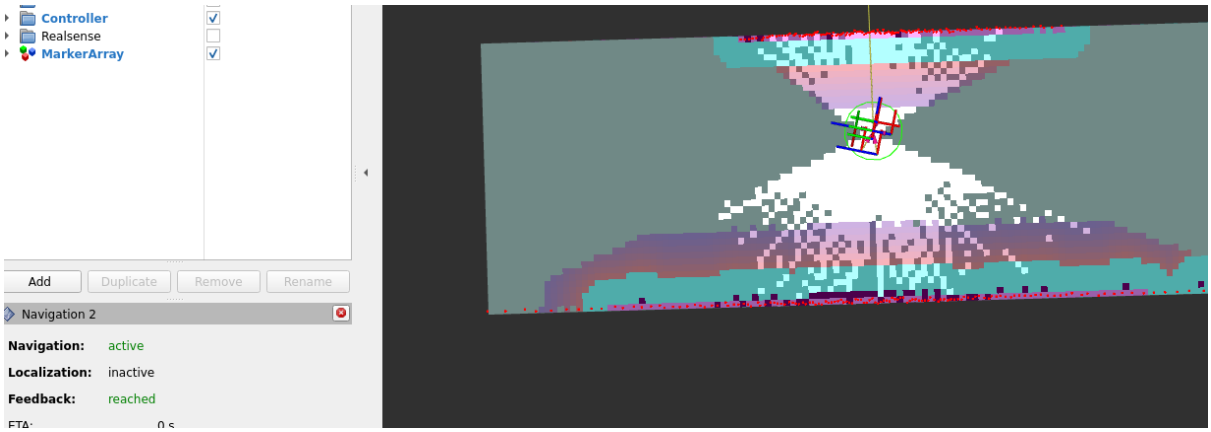

Figure 1: The robot must reach the first waypoint before status callbacks can be used

This is where the user is required to input 'enter' to continue the program

The first time the map subscriber is called, it initiates a list of unreachable positions the same size as the map, then searches over the current map for wall pixels that the robot detects and marks a square radius of 3 around the wall as unreachable positions.

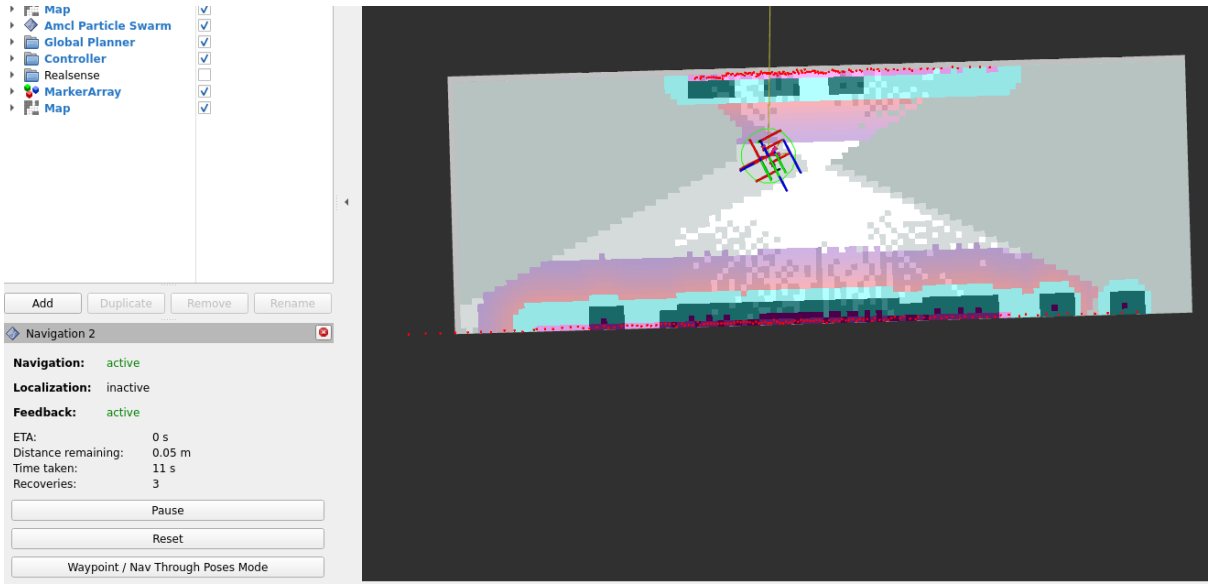

Figure 2: Initial map of unreachable points in black

The robot decides which pixels in range it wants to explore by searching over the whole map for unexplored pixels. If the function also finds a wall, it marks it as unreachable similar to when the first map subscription occurs. For every unexplored pixel, it checks a 5 x 5 square around the pixel, and if all pixels are unexplored, then it checks a 9 x 9 square around the pixel for explored pixels. If a pixel in the 9 x 9 square is in the unreachable list, then it checks a different unexplored pixel. If there are at least 5 explored pixels in the 9 x 9 square, and the distance between the robot and the waypoint is larger than 10 pixels, then the robot adds it to the frontier to be explored.

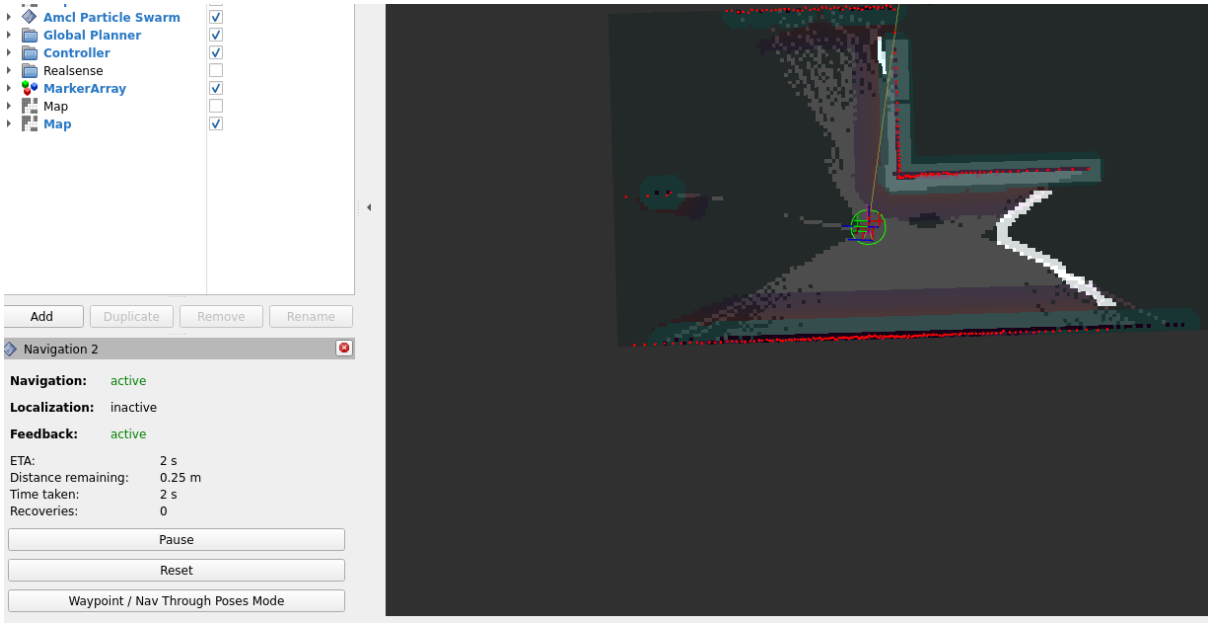

Figure 3: Frontier nodes in white

If the frontier is not empty, then the frontier is sorted according to the distance from the robot to the waypoint, preferring a set smaller (30 pixel) distance to the robot. A waypoint 20 pixels away from the robot has the same priority as one 40 pixels away, 10 same as 50, etc. This is to prefer closer pixels to the robot such that it explores in one direction, but not too close as to waste time.

The frontier is then popped to get the highest priority waypoint, but a path to the waypoint is first checked to see if it can be generated via basic navigator before being sent to the robot.

If the map size increases as the robot navigates, in the callback for the map the list of unreachable positions is expanded and the positions that are unreachable are re-checked.

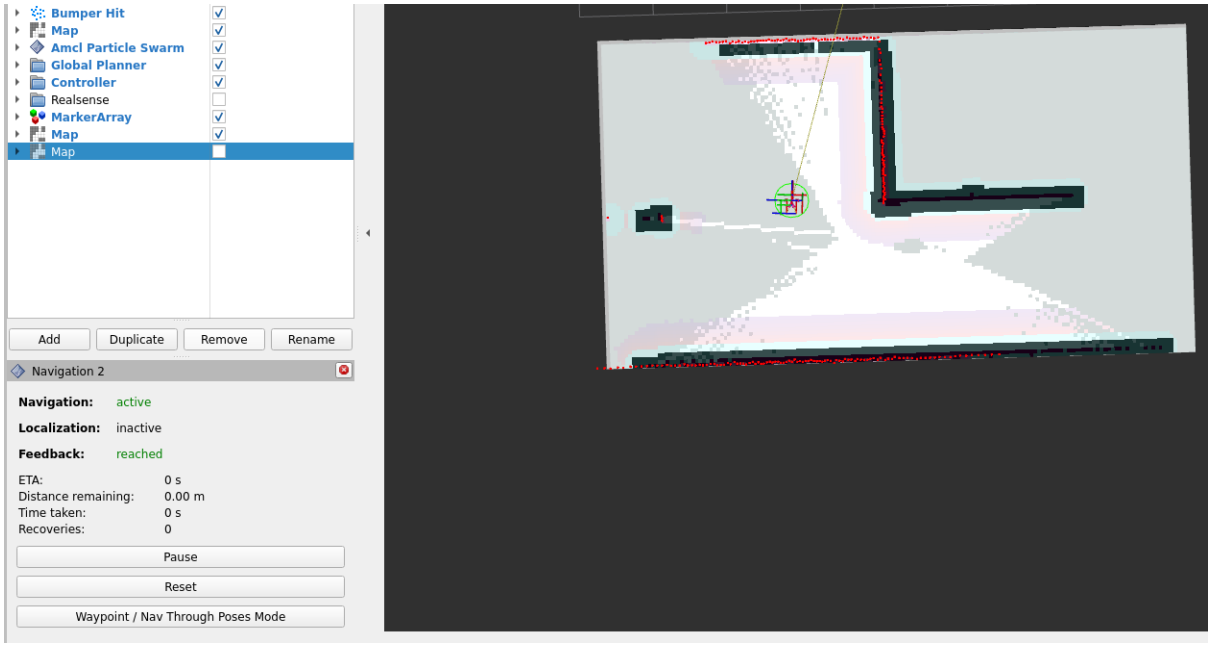

Figure 4: Expanding of unreachable map

Once the robot is within a small range (<0.1m) of the waypoint (or if the log callback detects that the robot is at the waypoint), it then re-searches for another unexplored pixel using the same method above. If the robot has not moved or turned within 3 seconds and is also not within that same small range of the waypoint, then the waypoint and the radius around it is also marked as an unreachable point.

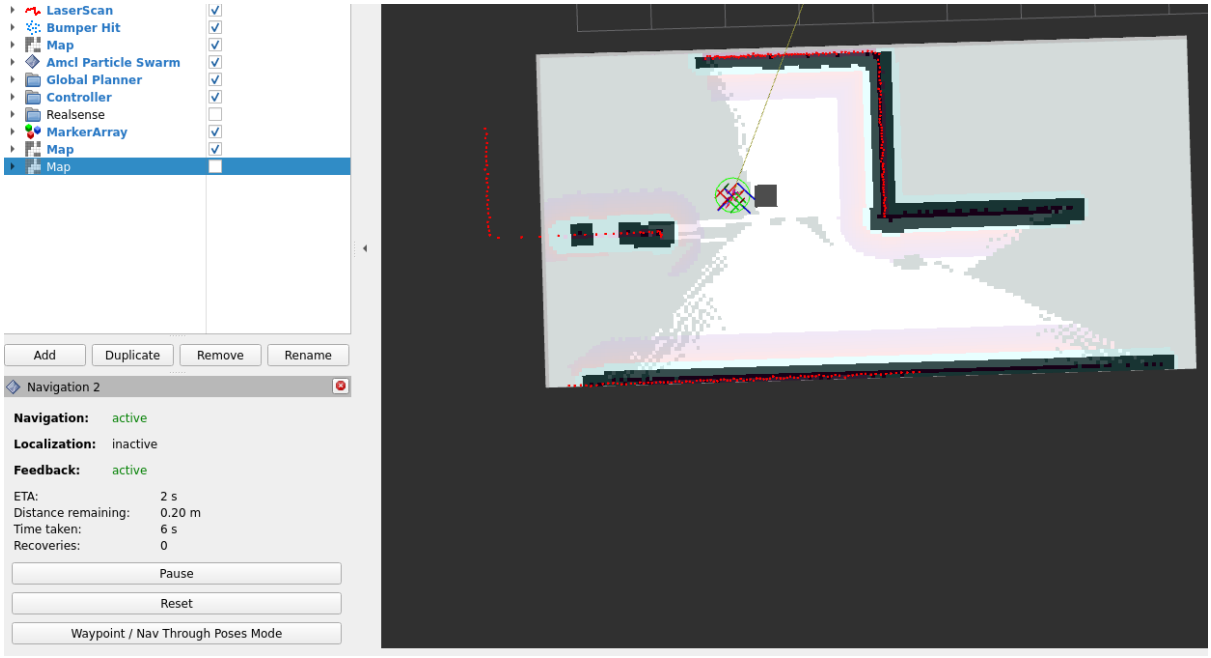

Figure 5: Marking a waypoint as unreachable after navigation

This repeats until the frontier is empty, to which the search ends and the robot concludes that the map is completely searched.