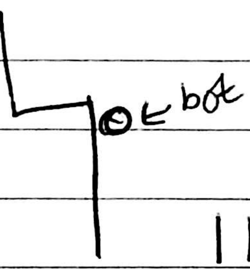
Algorithm ideas

Wall hugging : Stay at the walls and follow them around the map when we ~~lose~~ a wall we have to re-find the wall e.g in a map like:

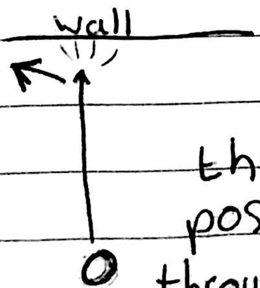


we go out of boundaries with the wall and then have our robot having to re-find the wall using the smallest distance in its path ; this should be the wall .

I hypothesise that the robot will begin to detect ~~ob~~ obstacles in the LG floor as 'walls' and start to follow them around. A problem I can see with this is that the robot may end up not exploring much of the centre of the environment ; which in a task is most likely to be used. Given proximity to objects of interest e.g. walls , I can see it being good for localisation. (Also , detect when we have gone full circle)

Hoover method : Move in a straight line until we come close to an obstacle . At which point we turn 45° (I think) .

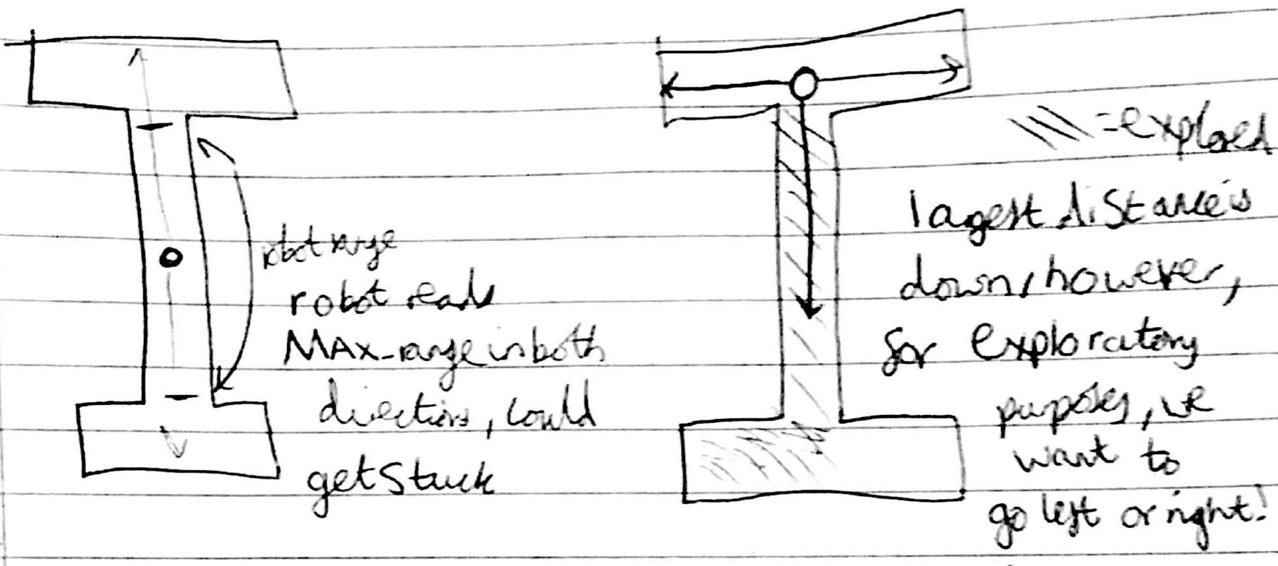
E.g.



Advantages: I ~~think~~ hypothesise this method will explore more and get stuck less than other methods, because if the robot gets positioned near the centre of the map, it could travel through multiple sectors at once. Could get stuck in a more confined space however .

The direction of most distance:

The robot goes in the direction of the largest distance it finds with one of, (atleast) of its sensors. This should keep it out of confined spaces. However at the basic level there are some issues with it, like this (going on layout of LG)


robot range
robot reads MAX-range in both directions, could get stuck
= explored
largest distance is down, however, for exploratory purposes, we want to go left or right!

We could try t implement 'knowing where ~~one~~ we have already been', this relies heavily on trusting the robots odometry etc. However the chance of getting stuck should be quite slim (especially if we forced the robot to make a move)

Random:

This method is fairly self-explanatory; we make the robot change it's direction pseudo-randomly only if it is about to hit a wall. This would be very computationally easy, however, I hypothesise this would be a highly unreliable method which also would become apparent when plotting results. I hypothesise this method would get stuck much more often than any other method being tested. I would only recommend using this method IF we do >2 methods, because there is a high probability of this method being of no future use. However, it would be highly useful for comparison with other methods.

Things to consider:

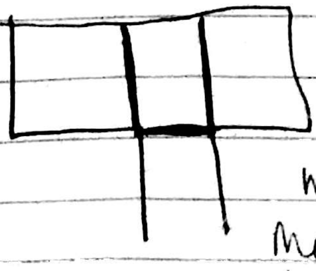
We have multiple sensors which could give us ideas of the distance: lidar + sonar (I will ignore kinect for now ☺) Obviously, both have their disadvantages + advantages; I would predict for the LG floor lases would be better. However, is there any we could combine them to get better information to the robot (given the nature of the programming this could be a pain, but I reckon it could be really cool!)

We get more credit for innovative design features. Here are some of the things we could do that I think may satisfy this:

- Using multiple sensors + combining them
- knowing where we've already been + including that!
- Cleaning up sensor readings (I saw last years group did this and it was a really cool idea).

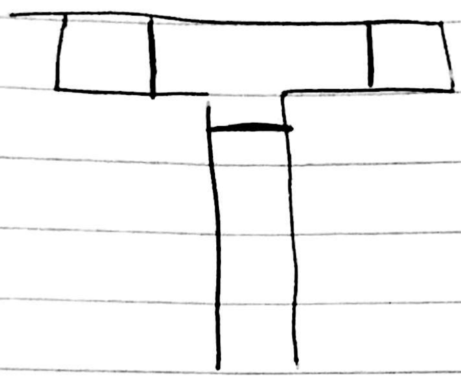
Testing + evaluating

~~At~~ I have read the stuff already in the report + it seemed good.
We have a distinct definition for going into a new chunk.
I would still like to record no. of times stuck. This is a recording very much to be done by eye but I would like to have a very clear definition of what it means 'to be stuck'
We have to choose our splitting points on the map VERY carefully. For example, choosing a subset of LG and ignoring size for now, this may be a bad idea:



because the robot could very much just move through these very slightly and count as moving through multiple chunks in a short ~~the~~ time which makes it look like a good explorer when it isn't.

This would be an improvement (albeit small)



What I am Saying is that having touching boundaries could be a no-go

AS well as all other tests, I think we could combine all of the values from the experiment together and weight them (choosing the weights scientifically) to get some judgement of 'overall system performance'. This would be quite good for an abstract / overview of our experiments.

Although I have ~~been~~ given 4 algorithms / methods here, after seeing the ones which can be extended well, my suggestion would be the wall hugger + maxdistance methods (maybe with random if we have time).