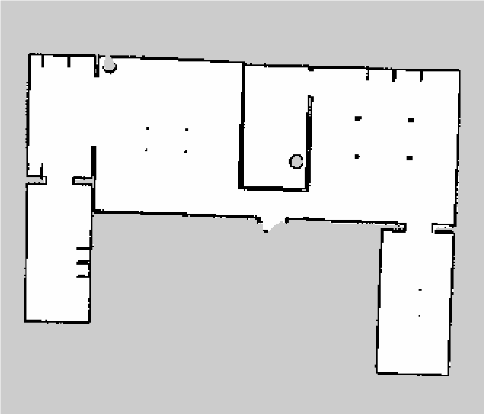
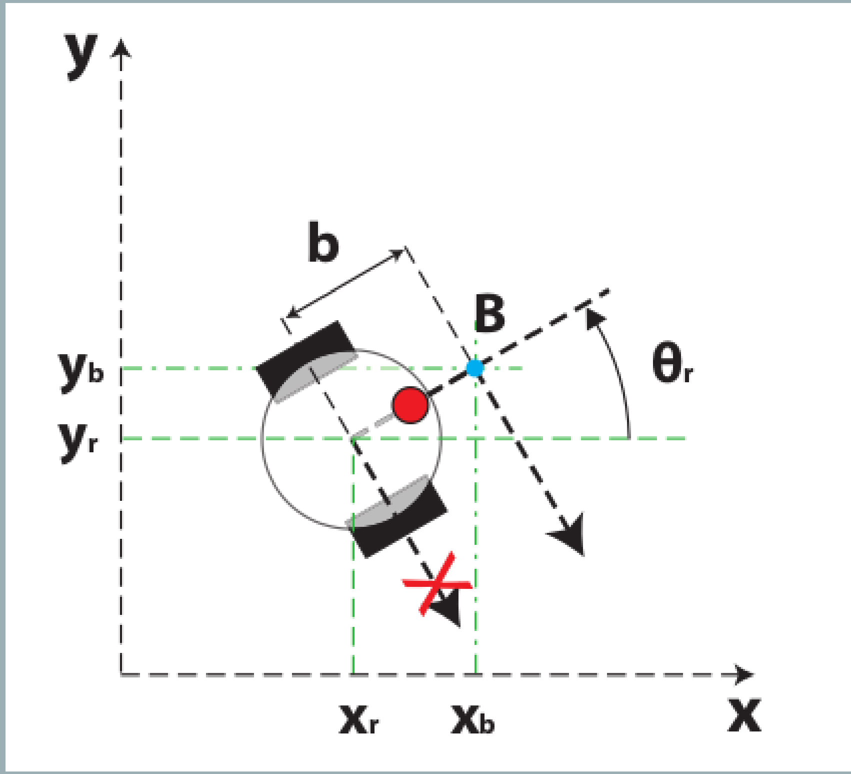
SIMULATION EXERCISES WITH ROS AND TURTLEBOT3 BURGER

SARIM MEHDI

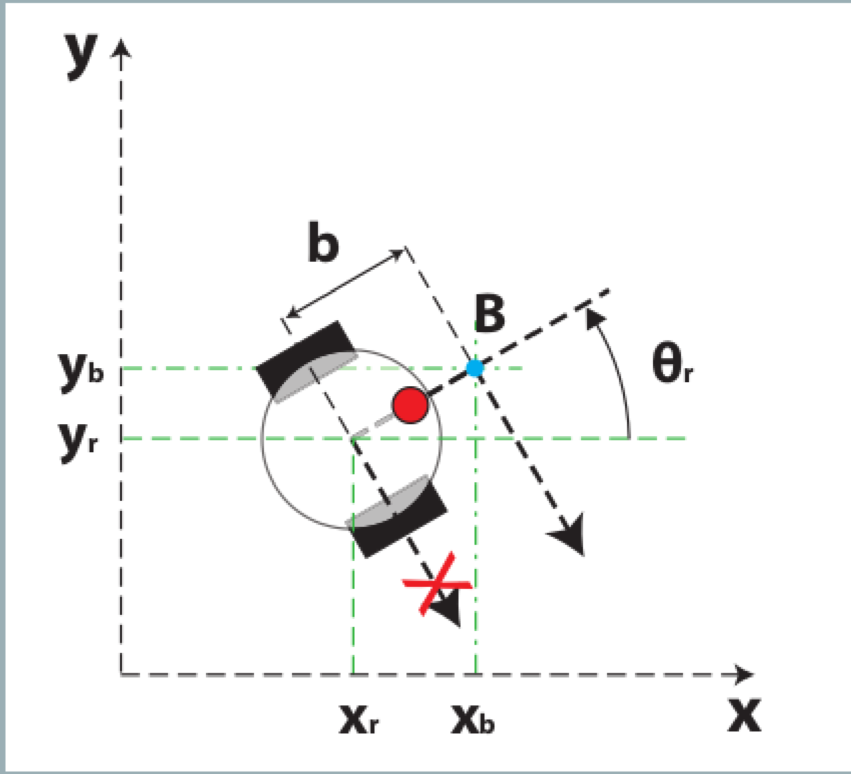
ZARA TORABI



FOLLOWING TRAJECTORY

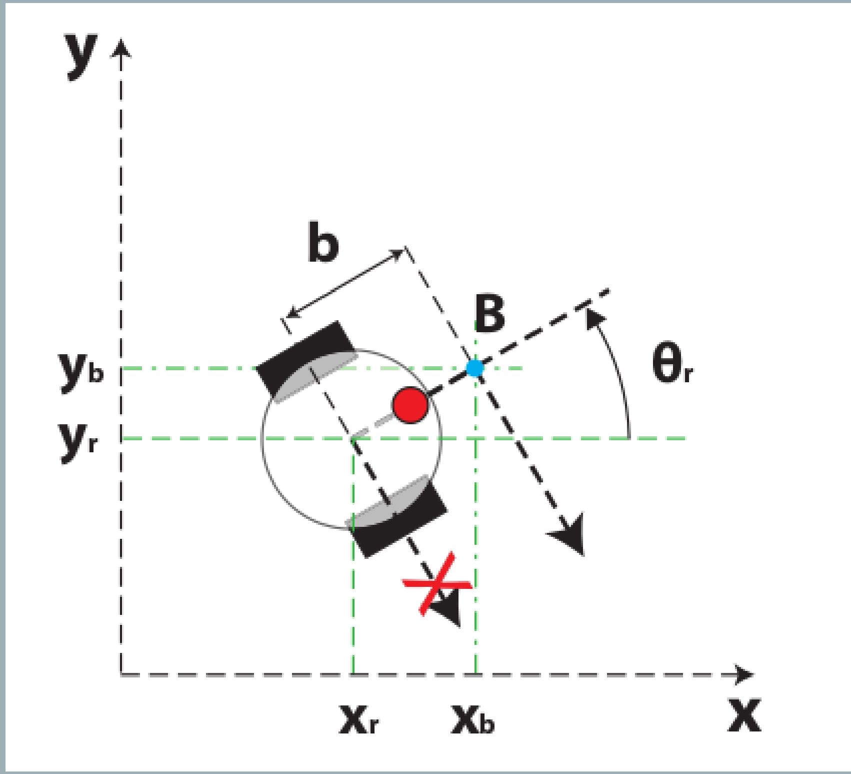


FOLLOWING TRAJECTORY



$$\begin{cases} x_b = x_r + b\cos\theta_r \\ y_b = y_r + b\sin\theta_r \end{cases}, \quad b \neq 0$$

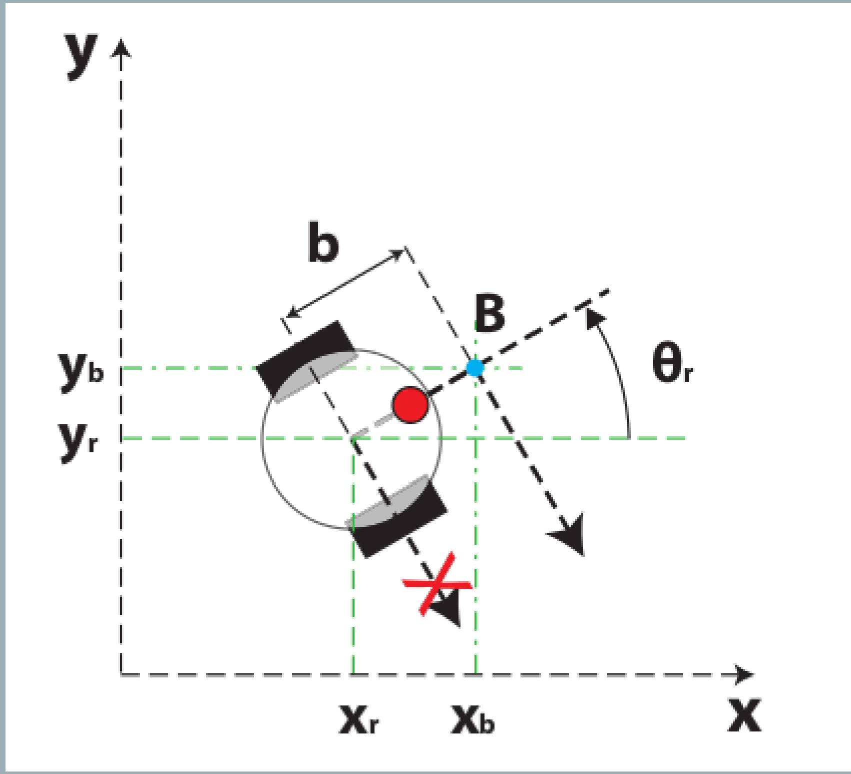
FOLLOWING TRAJECTORY



$$\begin{cases} x_b = x_r + b\cos\theta_r \\ y_b = y_r + b\sin\theta_r \end{cases}, \quad b \neq 0$$

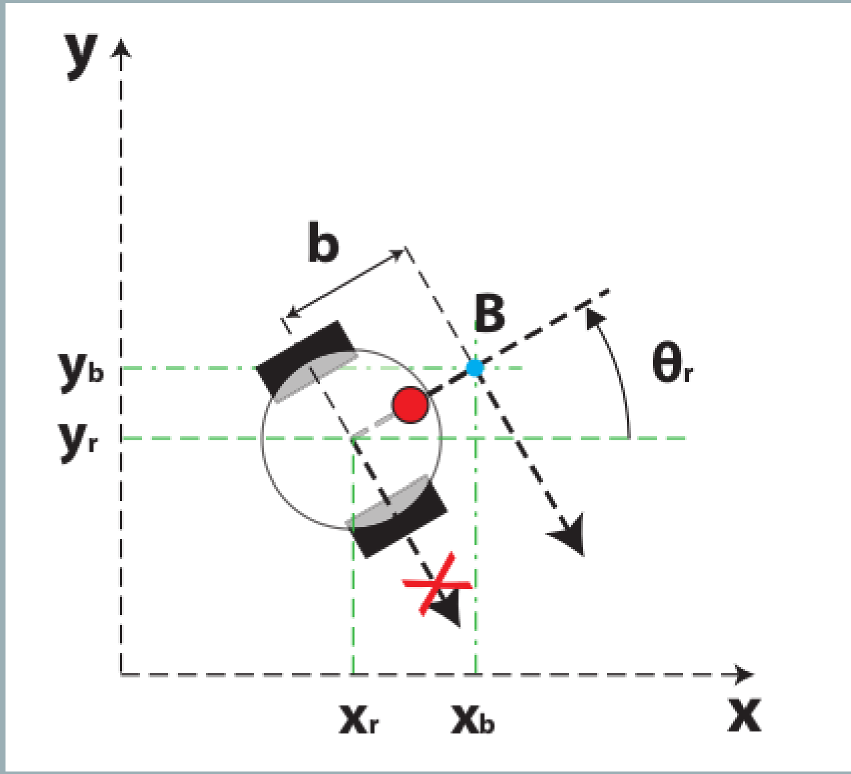
$$\begin{aligned} \dot{x}_b &= v_{x,b} \\ \dot{y}_b &= v_{y,b} \end{aligned}$$

FOLLOWING TRAJECTORY



$$
\begin{aligned}
\dot{x}_b &= \dot{x}_r - b \cdot \omega \sin \theta_r \\
\dot{y}_b &= \dot{y}_r + b \cdot \omega \cos \theta_r
\end{aligned}
$$

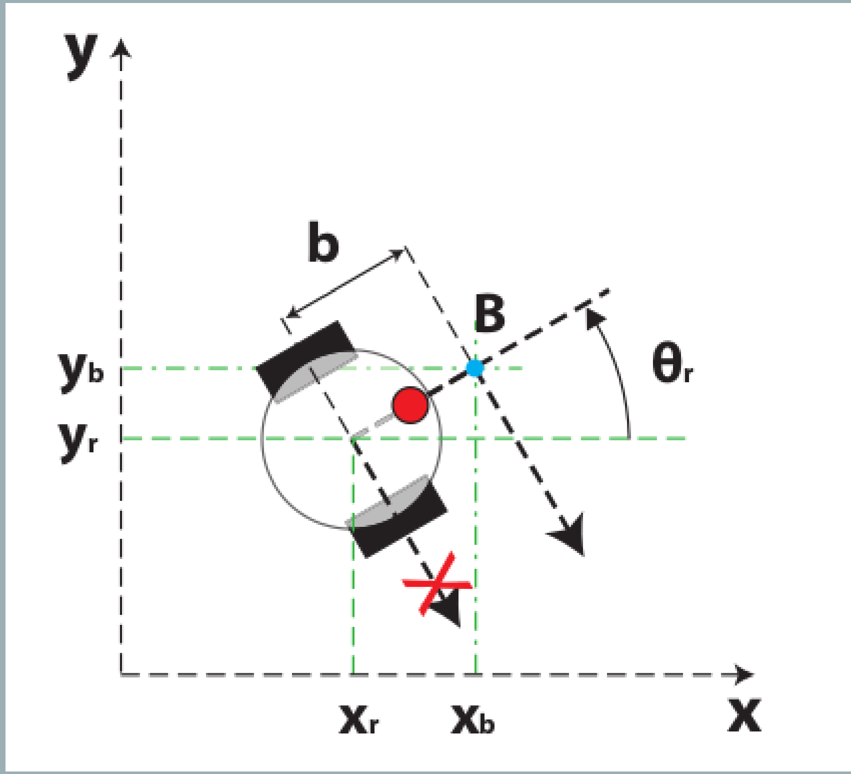
FOLLOWING TRAJECTORY



$$
\begin{aligned}
\dot{x}_b &= \dot{x}_r - b \cdot \omega \sin\theta_r \\
\dot{y}_b &= \dot{y}_r + b \cdot \omega \cos\theta_r
\end{aligned}
$$

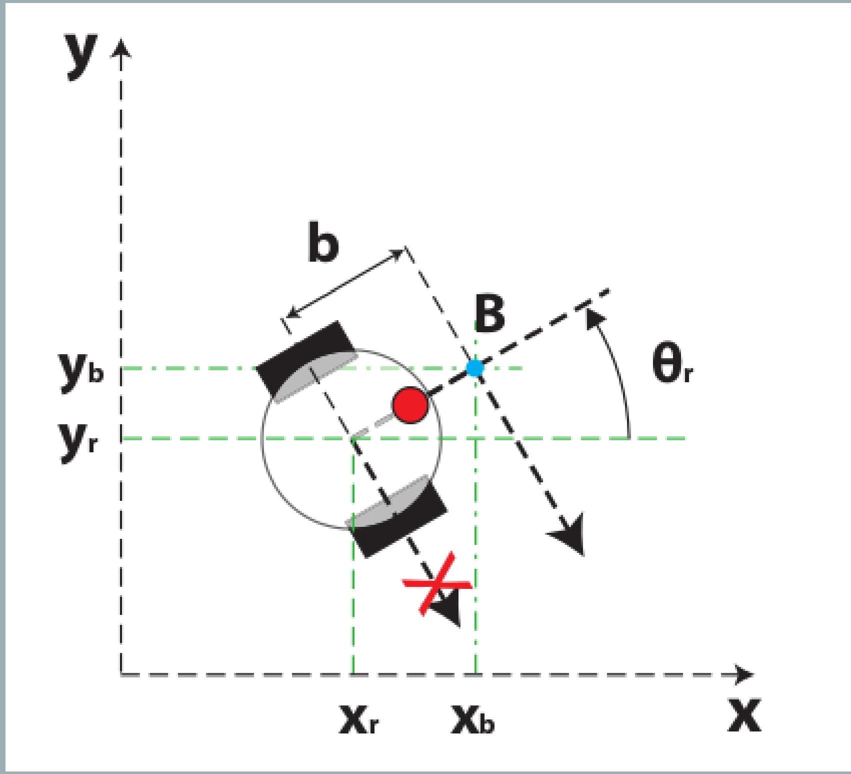
$$
\begin{aligned}
\dot{x}_b &= v\cos\theta_r - b \cdot \omega \sin\theta_r \\
\dot{y}_b &= v\sin\theta_r + b \cdot \omega \cos\theta_r
\end{aligned}
$$

FOLLOWING TRAJECTORY



$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \begin{bmatrix} \cos\theta_r & -b\sin\theta_r \\ \sin\theta_r & b\cos\theta_r \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

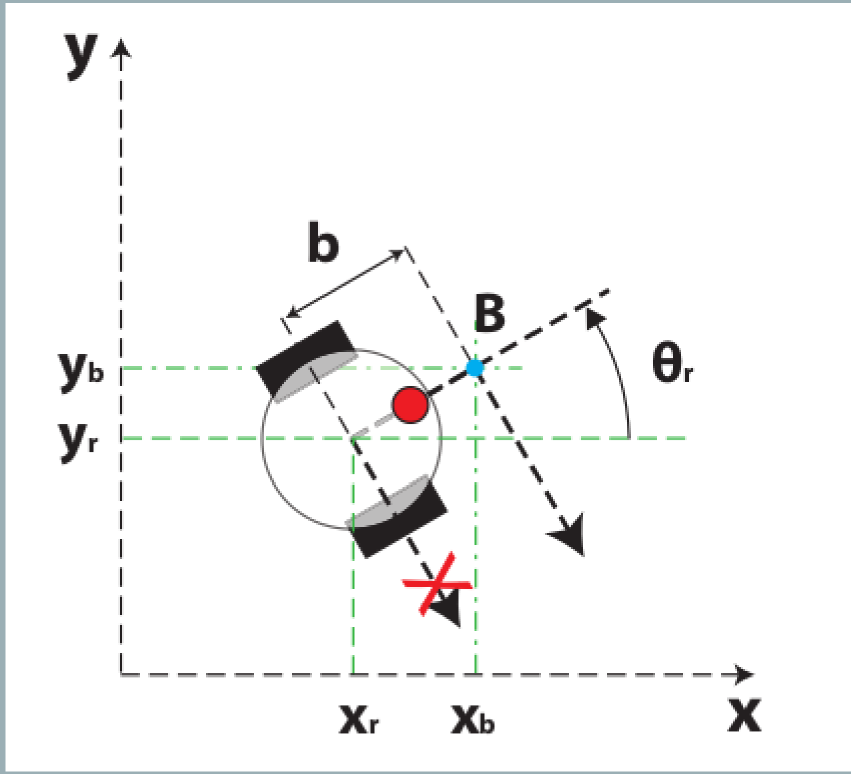
FOLLOWING TRAJECTORY



$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \begin{bmatrix} \cos\theta_r & -b\sin\theta_r \\ \sin\theta_r & b\cos\theta_r \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta_r & \sin\theta_r \\ -\frac{1}{b}\sin\theta_r & \frac{1}{b}\cos\theta_r \end{bmatrix} \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix}$$

FOLLOWING TRAJECTORY

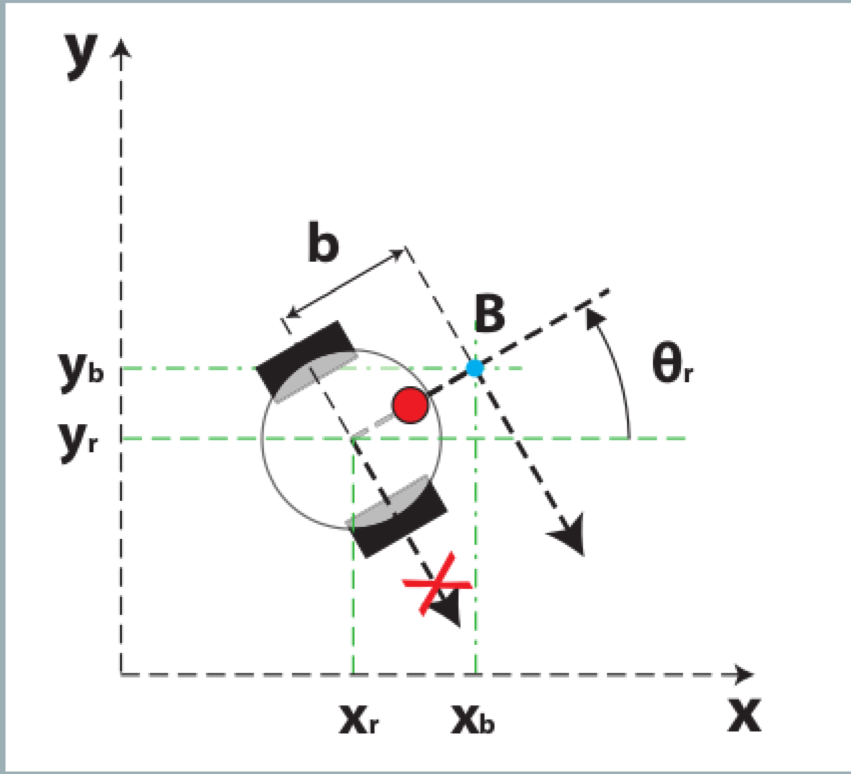


$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \begin{bmatrix} \cos\theta_r & -b\sin\theta_r \\ \sin\theta_r & b\cos\theta_r \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos\theta_r & \sin\theta_r \\ -\frac{1}{b}\sin\theta_r & \frac{1}{b}\cos\theta_r \end{bmatrix} \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix}$$

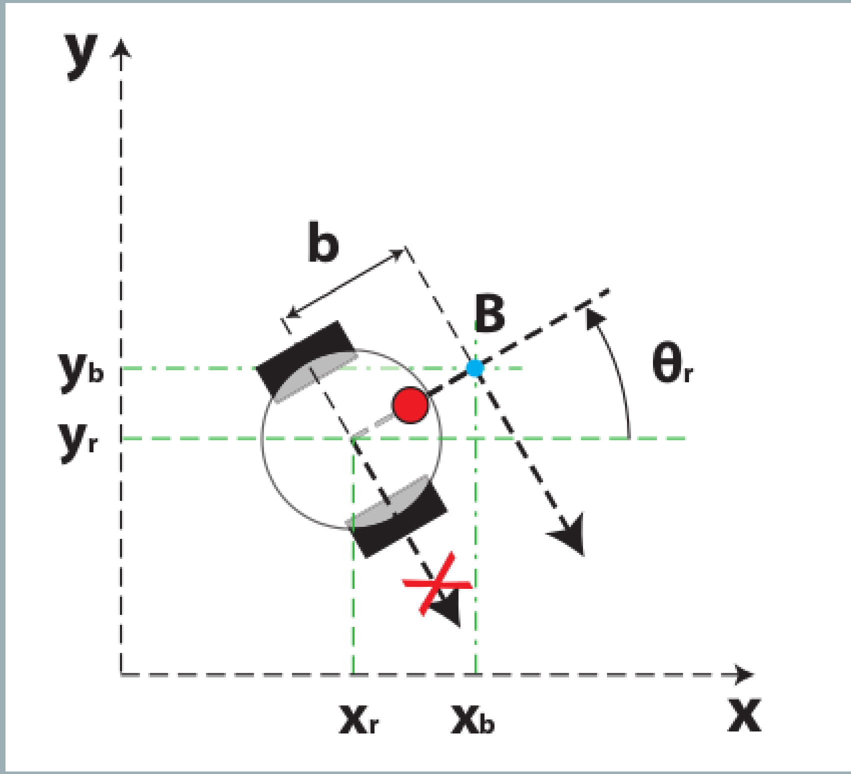
v = linear velocity input to Turtlebot3
w = angular velocity input to Turtlebot3

FOLLOWING TRAJECTORY



GENERATE TRAJECTORY HAS POINTS [q1*, q2*, …, qn*]

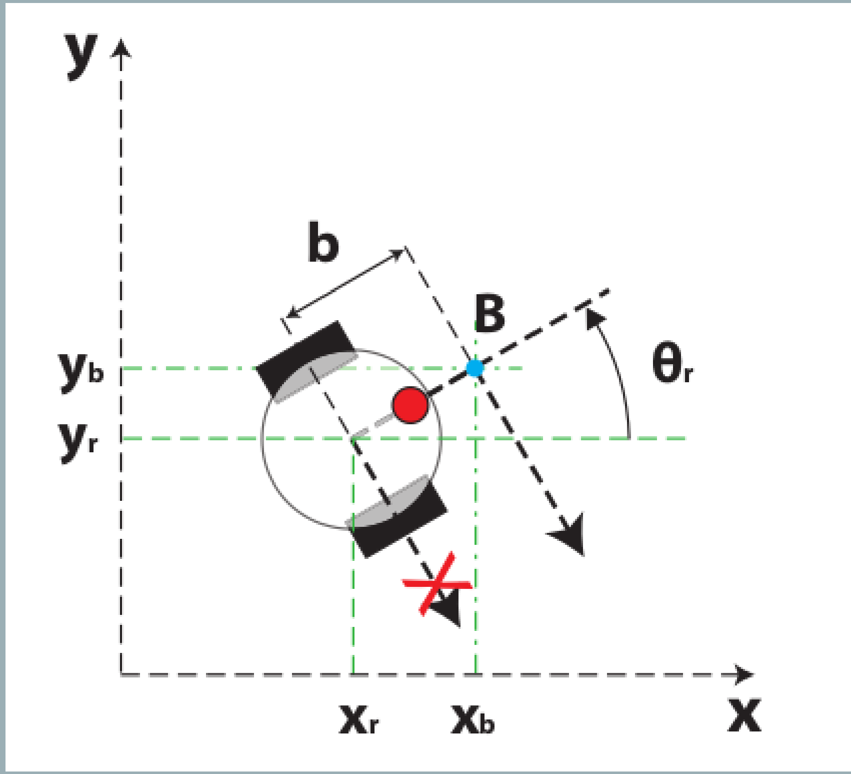
FOLLOWING TRAJECTORY



GENERATE TRAJECTORY HAS POINTS [q1*, q2*, …, qn*]

LET q* BE A POINT FROM THIS LIST

FOLLOWING TRAJECTORY

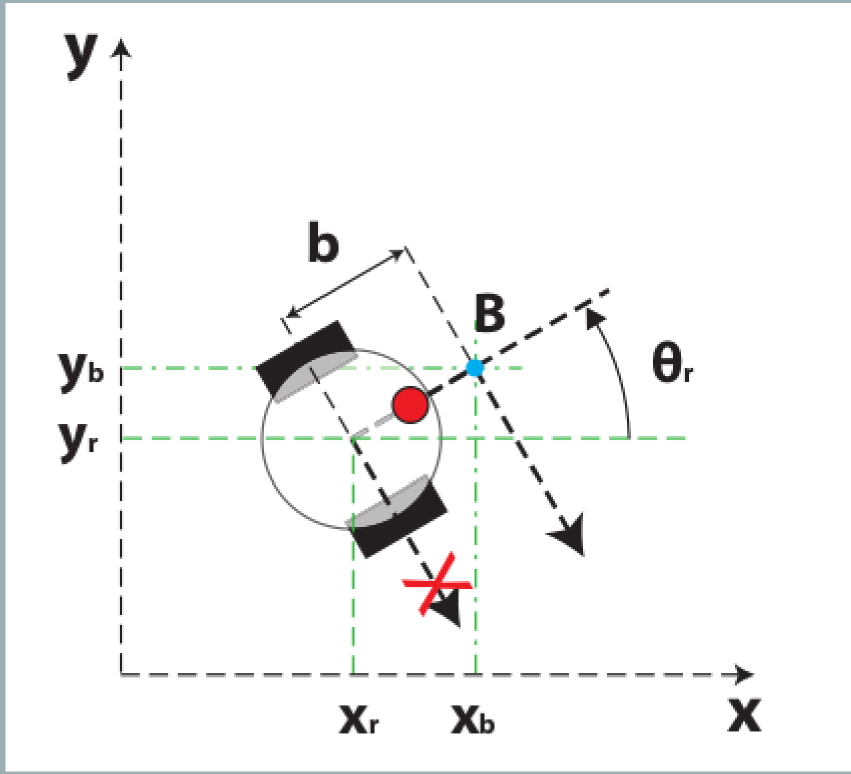


GENERATE TRAJECTORY HAS POINTS [q1*, q2*, …, qn*]

LET q* = [x*, y*] BE A POINT FROM THIS LIST THAT ROBOT MUST GET TO

LET q = [xb, yb] BE CURRENT POSITION OF ROBOT POINT B

FOLLOWING TRAJECTORY



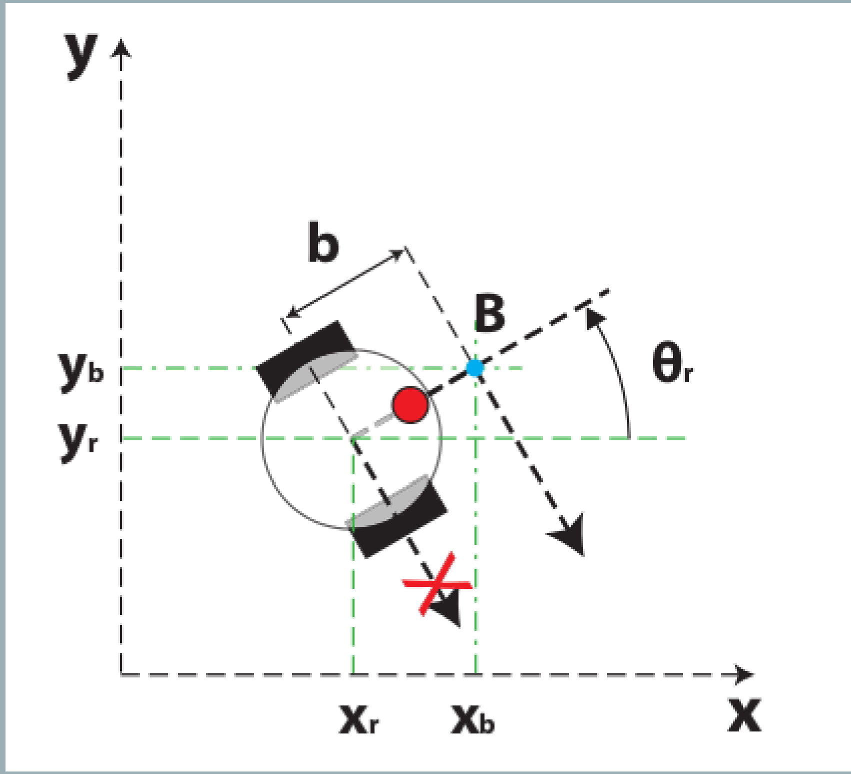
GENERATE TRAJECTORY HAS POINTS [q1*, q2*, …, qn*]

LET q* = [x*, y*] BE A POINT FROM THIS LIST THAT ROBOT MUST GET TO

LET q = [xb, yb] BE CURRENT POSITION OF ROBOT POINT B

USE POTENTIAL FIELD ALGORTHM TO GET REQUIRED vxb and vyb

FOLLOWING TRAJECTORY



Attractive potential

$U_{att}(q)$

- It has the minimum in q_{goal}
- Its job is to attract the robot to the goal

Repulsive potential

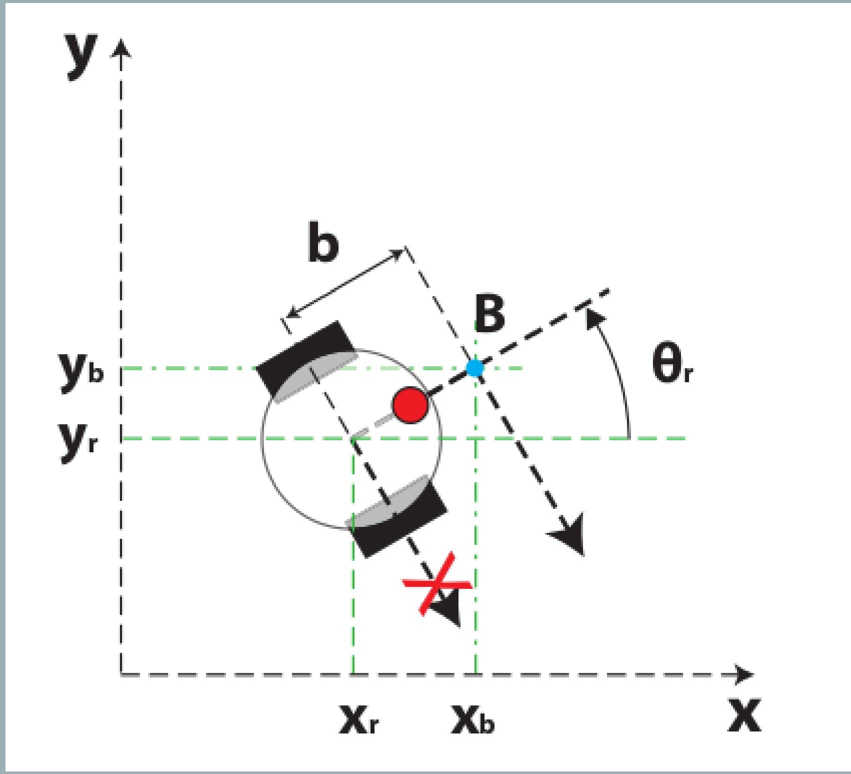
$U_{rep} = \sum_{i=1}^{N_{obst}} U_{rep_i}(q)$

- Sum of the repulsive potentials for each obstacle
- It takes the robot away from obstacles

Control law

$$\dot{q}(t) = -\nabla U(q), \qquad \text{dove: } U(q) = U_{att}(q) + U_{rep}(q)$$

FOLLOWING TRAJECTORY



Possible choice

$$U_{att} = \frac{1}{2}K_a\Delta(q, q_{\text{goal}}) \qquad \nabla U_{att} = \frac{1}{2}K_a\nabla\Delta^2(q, q_{\text{goal}}) = K_a(q - q_{\text{goal}})$$

Possible choice

$$U_{rep_i}(q) = \begin{cases} \frac{1}{2}K_{r_i}\left(\frac{1}{d_i(q)} - \frac{1}{q^*}\right)^2 & d_i(q) \leq q^* \\ 0 & d_i(q) > q^* \end{cases}$$

$$\nabla U_{rep_i}(q) = \begin{cases} K_{r_i}\left(\frac{1}{q^*} - \frac{1}{d_i(q)}\right)^2 \nabla d_i(q) & d_i(q) \leq q^* \\ 0 & d_i(q) > q^* \end{cases}$$

OBSERVATIONS

Values set after experimentation:

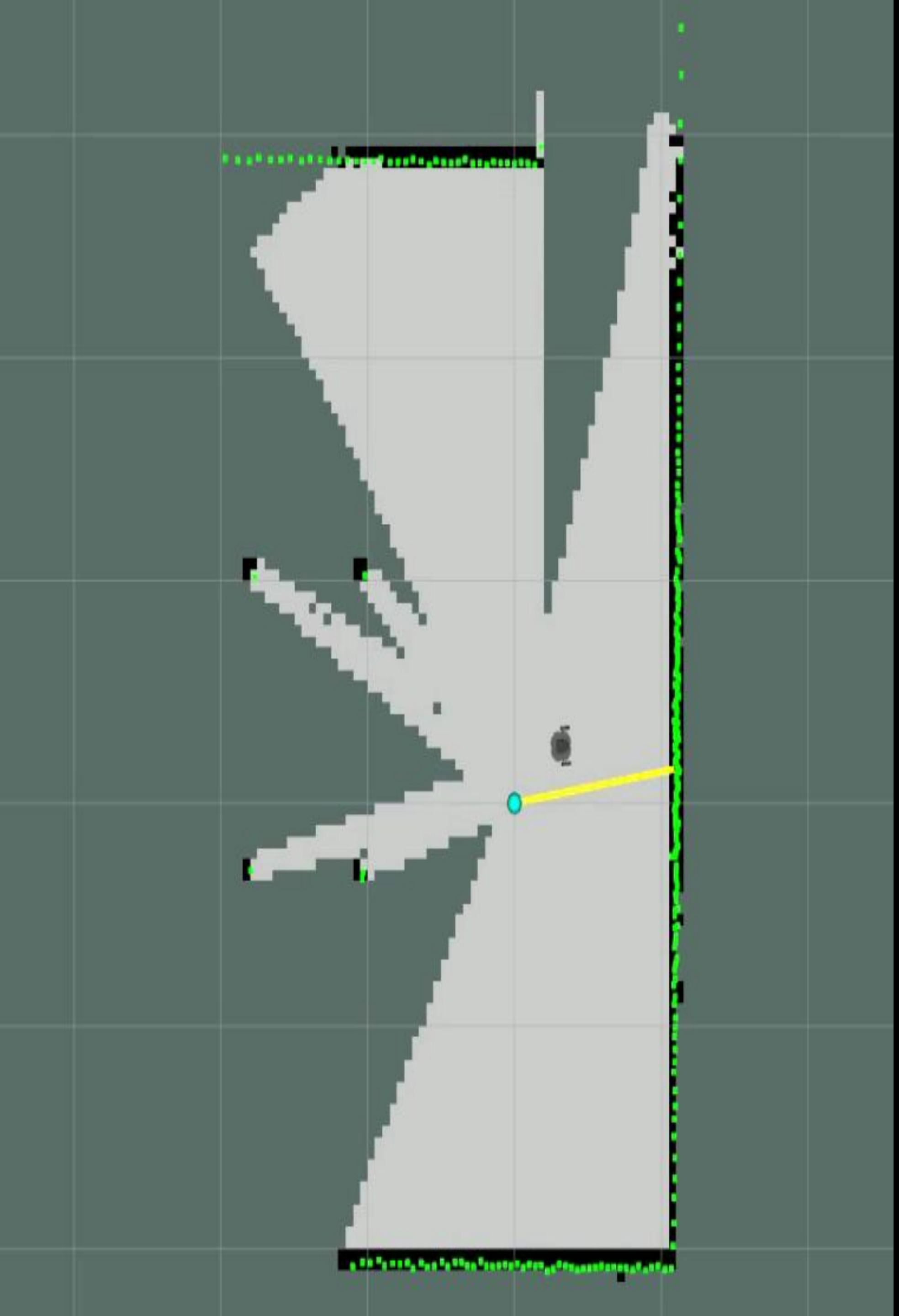
'b' = 0.1

'Ka' = 0.2

'Kr' = 0.05

'q*' = 0.5


CORNER CASES



MAPPING

- GET CLOSEST OBSTACLE IN FRONT OF TURTLEBOT AND GENERATE STRAIGHT PATH TOWARDS IT

MAPPING

- GET CLOSEST OBSTACLE IN FRONT OF TURTLEBOT AND GENERATE STRAIGHT PATH TOWARDS IT
- WHEN YOU GET TO POINT, MARK IT AS VISITED SO THAT YOU DON'T REVISIT IT IN THE NEAR FUTURE, AND THEN MOVE TO NEXT CLOSEST OBSTACLE IN FRONT OF YOUR ROBOT

MAPPING

- GET CLOSEST OBSTACLE IN FRONT OF TURTLEBOT AND GENERATE STRAIGHT PATH TOWARDS IT
- WHEN YOU GET TO POINT, MARK IT AS VISITED SO THAT YOU DON'T REVISIT IT IN THE NEAR FUTURE, AND THEN MOVE TO NEXT CLOSEST OBSTACLE IN FRONT OF YOUR ROBOT
- WHAT IF YOU WISH TO BACKTRACK AND EXPLORE OTHER AREAS OF THE HOUSE?

EXPERIMENTS

- ROBOT SPAWNED IN EACH OF THE 6 ROOMS AND GIVEN 10 MINUTES TO EXPLORE AS MUCH OF THE HOUSE AS POSSIBLE
- ROBOT FULLY EXPLORES 4 ROOMS IN BEST ATTEMPT AND 2 ROOMS IN WORST ATTEMPT

NAVIGATION

- USE PRM (PROBABILISTIC ROADMAP)

NAVIGATION

- USE PRM (PROBABILISTIC ROADMAP)
- 500 POINTS RANDOMLY SAMPLED FROM FREE SPACE

NAVIGATION

- USE PRM (PROBABILISTIC ROADMAP)
- 500 POINTS RANDOMLY SAMPLED FROM FREE SPACE
- EACH POINT CAN HAVE AT MOST 10 NEIGHBORS WITHIN A DISTANCE OF 0.3 CM WHICH ARE CONNECTED TO IT WITH AN UNDIRECTED EDGE (EDGE COST SAME AS DISTANCE BETWEEN POINTS)

NAVIGATION

- USE PRM (PROBABILISTIC ROADMAP)
- 500 POINTS RANDOMLY SAMPLED FROM FREE SPACE
- EACH POINT CAN HAVE AT MOST 10 NEIGHBORS WITHIN A DISTANCE OF 0.3 CM WHICH ARE CONNECTED TO IT WITH AN UNDIRECTED EDGE (EDGE COST SAME AS DISTANCE BETWEEN POINTS)
- CONNECT GOAL POSITION TO NEAREST POINT IN THIS GRAPH.

NAVIGATION

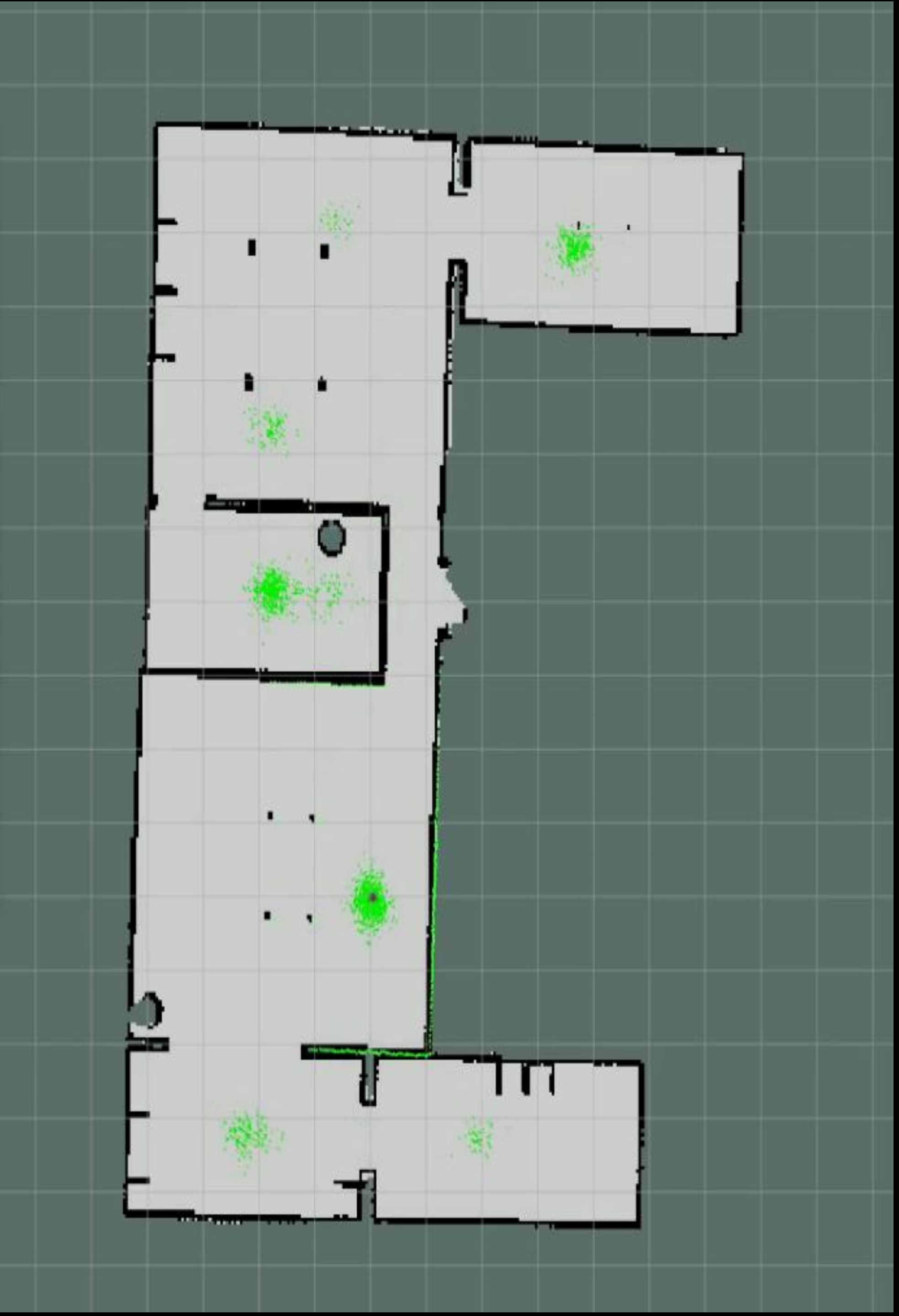
- USE PRM (PROBABILISTIC ROADMAP)
- 500 POINTS RANDOMLY SAMPLED FROM FREE SPACE
- EACH POINT CAN HAVE AT MOST 10 NEIGHBORS WITHIN A DISTANCE OF 0.3 CM WHICH ARE CONNECTED TO IT WITH AN UNDIRECTED EDGE (EDGE COST SAME AS DISTANCE BETWEEN POINTS)
- CONNECT GOAL POSITION TO NEAREST POINT IN THIS GRAPH.
- CONNECT ROBOT POSITION TO CLOSEST POINT IN GRAPH

NAVIGATION

- USE PRM (PROBABILISTIC ROADMAP)
- 500 POINTS RANDOMLY SAMPLED FROM FREE SPACE
- EACH POINT CAN HAVE AT MOST 10 NEIGHBORS WITHIN A DISTANCE OF 0.3 CM WHICH ARE CONNECTED TO IT WITH AN UNDIRECTED EDGE (EDGE COST SAME AS DISTANCE BETWEEN POINTS)
- CONNECT GOAL POSITION TO NEAREST POINT IN THIS GRAPH.
- CONNECT ROBOT POSITION TO CLOSEST POINT IN GRAPH
- USE DIJKSTRA'S TO TRAVERSE GRAPH TO GOAL

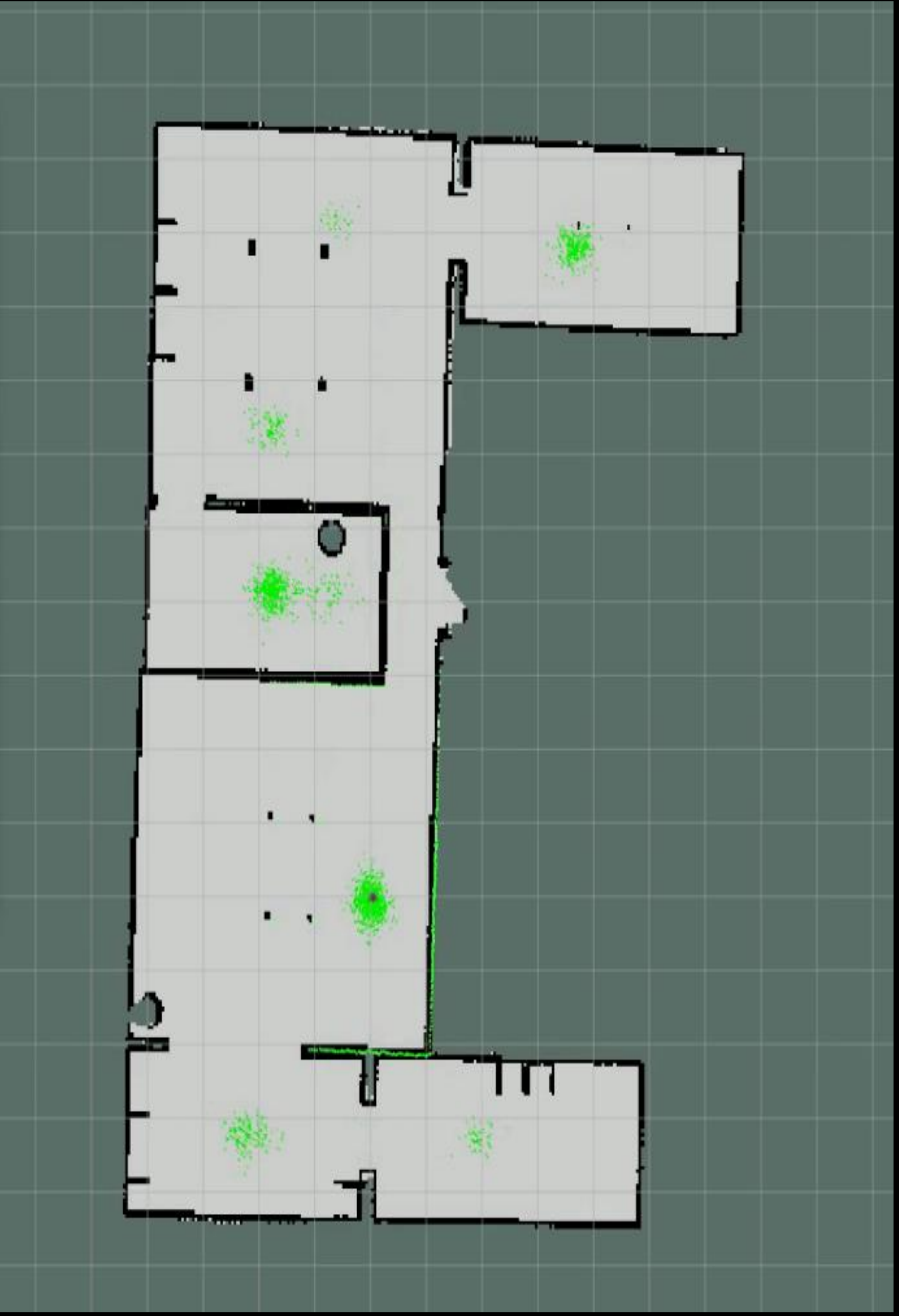
NAVIGATION

- USE PRM (PROBABILISTIC ROADMAP)
- 500 POINTS RANDOMLY SAMPLED FROM FREE SPACE
- EACH POINT CAN HAVE AT MOST 10 NEIGHBORS WITHIN A DISTANCE OF 0.3 CM WHICH ARE CONNECTED TO IT WITH AN UNDIRECTED EDGE (EDGE COST SAME AS DISTANCE BETWEEN POINTS)
- CONNECT GOAL POSITION TO NEAREST POINT IN THIS GRAPH.
- CONNECT ROBOT POSITION TO CLOSEST POINT IN GRAPH
- USE DIJKSTRA'S TO TRAVERSE GRAPH TO GOAL
- IN CASE OF FAILURE, ALGORITHM CAN REPEAT FOR A TOTAL OF 10 TIMES BEFORE GIVING UP COMPLETELY



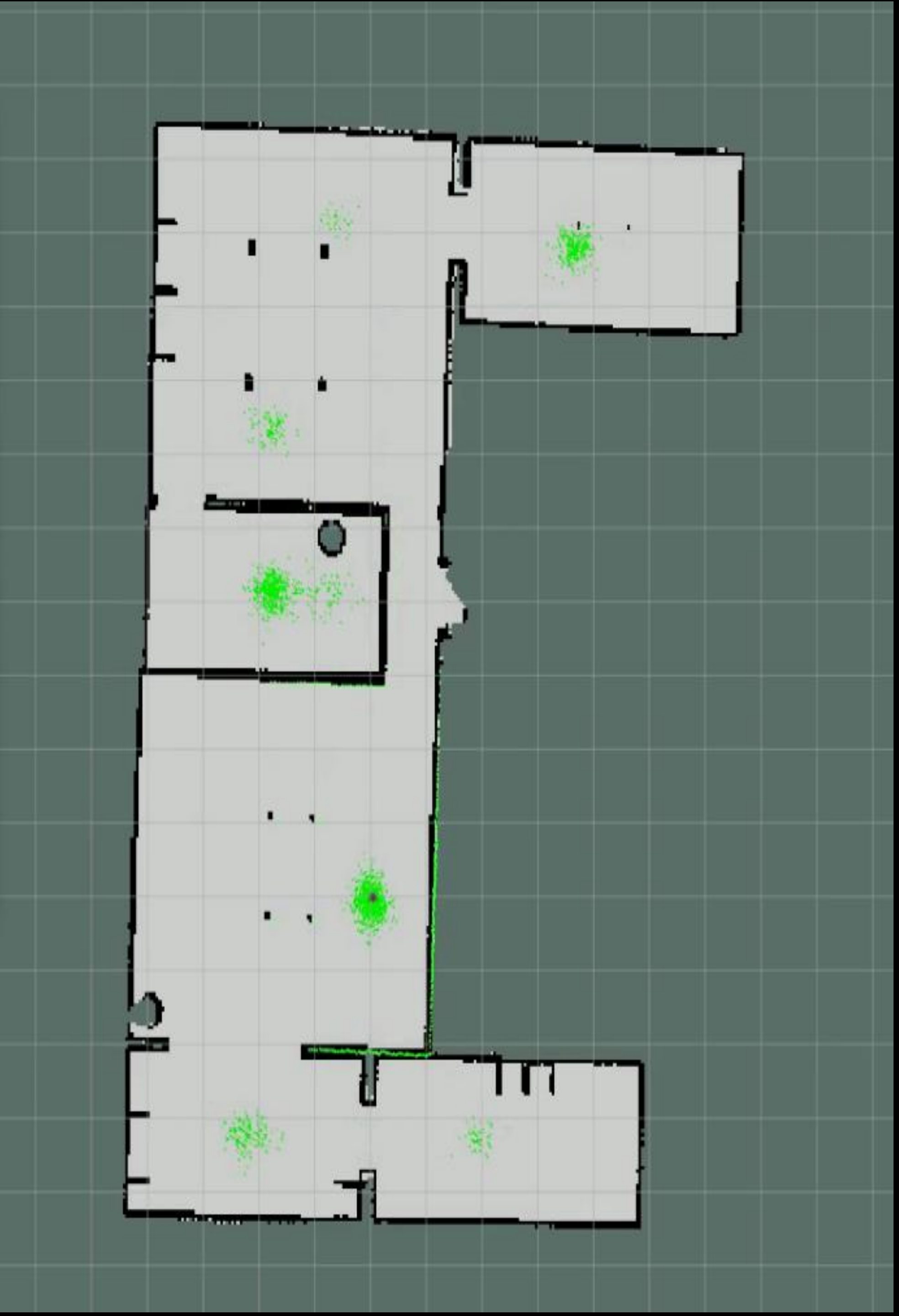
LOCALIZATION (DONE BEFORE NAVIGATION)

- AMCL PARTICLES EVENLY SPREAD ALL OVER THE MAP IN BEGINNING



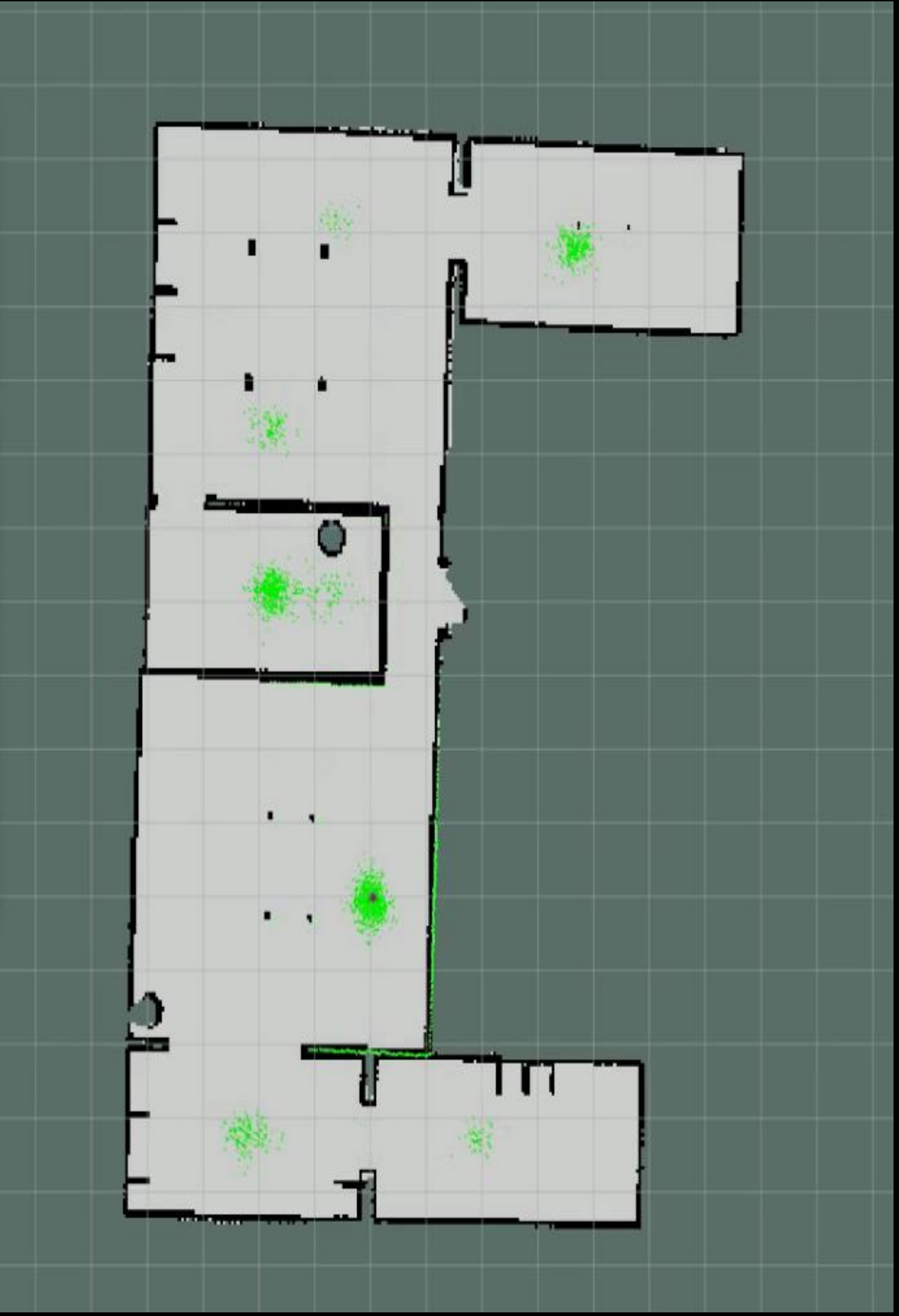
LOCALIZATION (DONE BEFORE NAVIGATION)

- AMCL PARTICLES EVENLY SPREAD ALL OVER THE MAP IN BEGINNING
- ROBOT HAS EQUAL PROBABILTY OF BEING AT ANY OF THE PARTICLE LOCATIONS



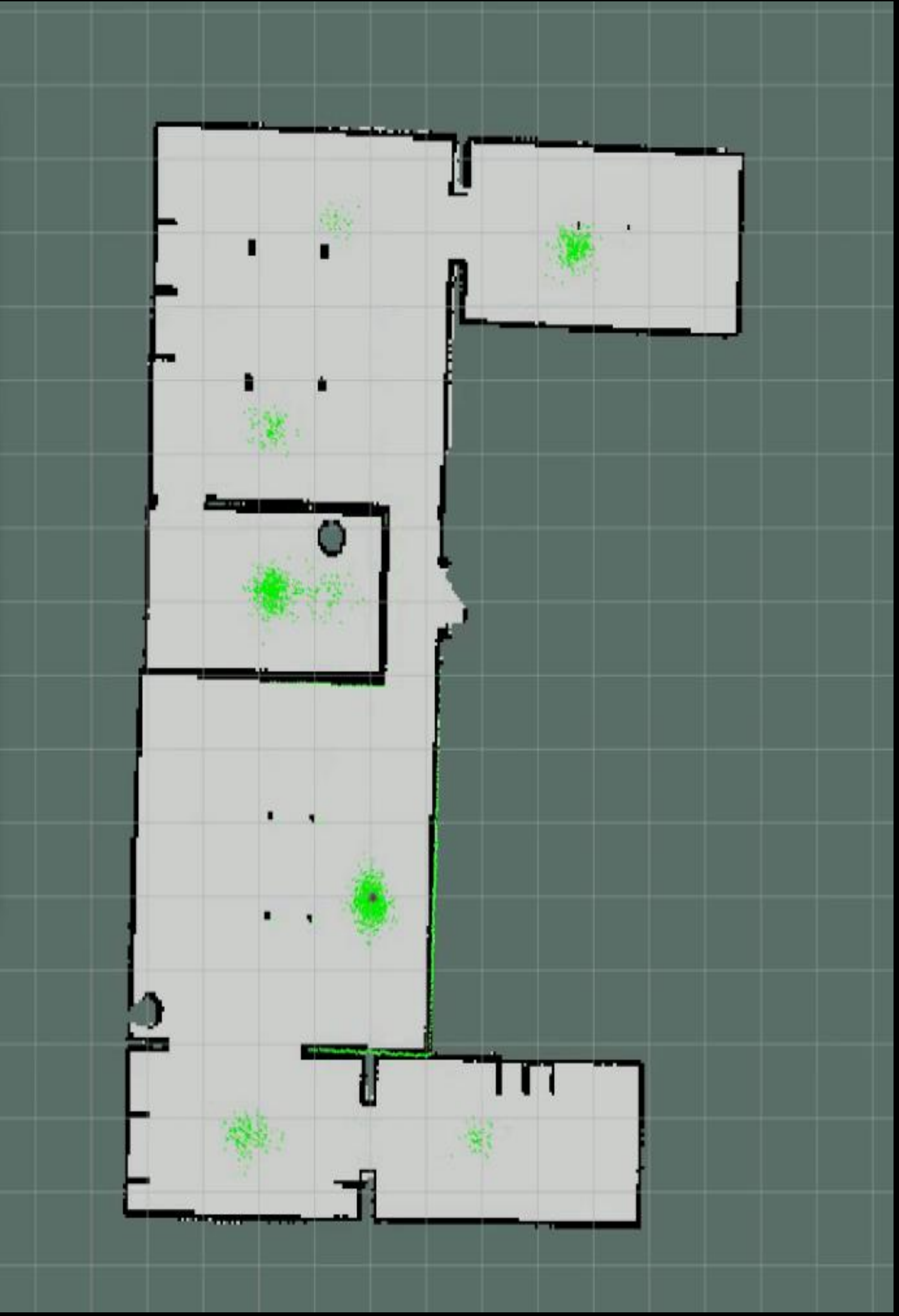
LOCALIZATION (DONE BEFORE NAVIGATION)

- AMCL PARTICLES EVENLY SPREAD ALL OVER THE MAP IN BEGINNING
- ROBOT HAS EQUAL PROBABILTY OF BEING AT ANY OF THE PARTICLE LOCATIONS
- ROBOT MADE TO SPUN AROUND FOR 60 SECONDS



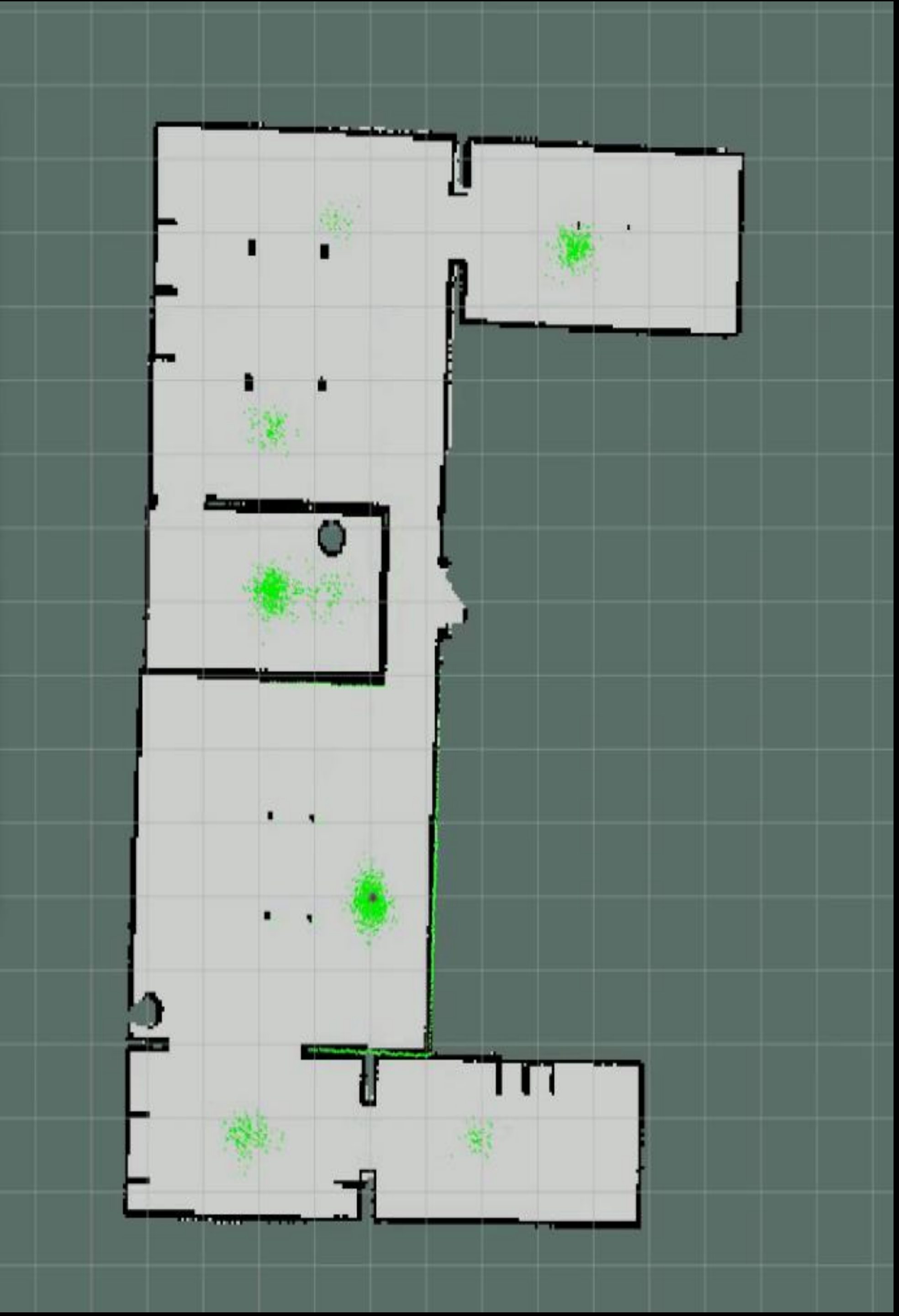
LOCALIZATION (DONE BEFORE NAVIGATION)

- AMCL PARTICLES EVENLY SPREAD ALL OVER THE MAP IN BEGINNING
- ROBOT HAS EQUAL PROBABILTY OF BEING AT ANY OF THE PARTICLE LOCATIONS
- ROBOT MADE TO SPUN AROUND FOR 60 SECONDS
- BASED ON LASER SCANNER READINGS, PROBABILITY OF ROBOT BEING AT EACH OF THE PARTICLES IS RECOMPUTED



LOCALIZATION (DONE BEFORE NAVIGATION)

- AMCL PARTICLES EVENLY SPREAD ALL OVER THE MAP IN BEGINNING
- ROBOT HAS EQUAL PROBABILTY OF BEING AT ANY OF THE PARTICLE LOCATIONS
- ROBOT MADE TO SPUN AROUND FOR 60 SECONDS
- BASED ON LASER SCANNER READINGS, PROBABILITY OF ROBOT BEING AT EACH OF THE PARTICLES IS RECOMPUTED
- LOTS OF PARTICLES GATHER AROUND THE LIKELY TRUE POSITION OF THE ROBOT



LOCALIZATION (DONE BEFORE NAVIGATION)

- AMCL PARTICLES EVENLY SPREAD ALL OVER THE MAP IN BEGINNING
- ROBOT HAS EQUAL PROBABILTY OF BEING AT ANY OF THE PARTICLE LOCATIONS
- ROBOT MADE TO SPUN AROUND FOR 60 SECONDS
- BASED ON LASER SCANNER READINGS, PROBABILITY OF ROBOT BEING AT EACH OF THE PARTICLES IS RECOMPUTED
- LOTS OF PARTICLES GATHER AROUND THE LIKELY TRUE POSITION OF THE ROBOT
- ALGORITHM GIVES EXACT LOCATION OF ROBOT IN MAP 99% OF THE TIMES