Mobile Robot Systems Mini Project 5

Sam Sully (sjs252), Paul Durbaba (pd452), Luke Dunsmore (ldd25)

Lent 2020

Project Outline

Project Outline

- LIDAR based localisation (ex1)

Project Outline

- LIDAR based localisation (ex1)
- Improve with range and bearing of other robots (sjs252)

Project Outline

- LIDAR based localisation (ex1)
- Improve with range and bearing of other robots (sjs252)
- Centralised approach to world coverage (ldd25)

Project Outline

- LIDAR based localisation (ex1)
- Improve with range and bearing of other robots (sjs252)
- Centralised approach to world coverage (ldd25)
- Decentralised approach to world coverage (pd452)

Localisation

Localisation

- Particle filter

Localisation

- Particle filter
- LIDAR

Localisation

- Particle filter
- LIDAR
- Range & bearing

LIDAR

$$w_i = \prod_{s_j \in \text{Sensors}} \Phi(R(i,j), s_{ij}, \sigma^2)$$

- w_i = LIDAR weight of particle i
- s_{ij} = distance recorded by sensor j on the robot
- $\Phi(x, \mu, \sigma)$ = Gaussian PDF with mean μ and standard deviation σ
- $R(i,j)$ = ray traced distance from particle i in the direction of sensor j

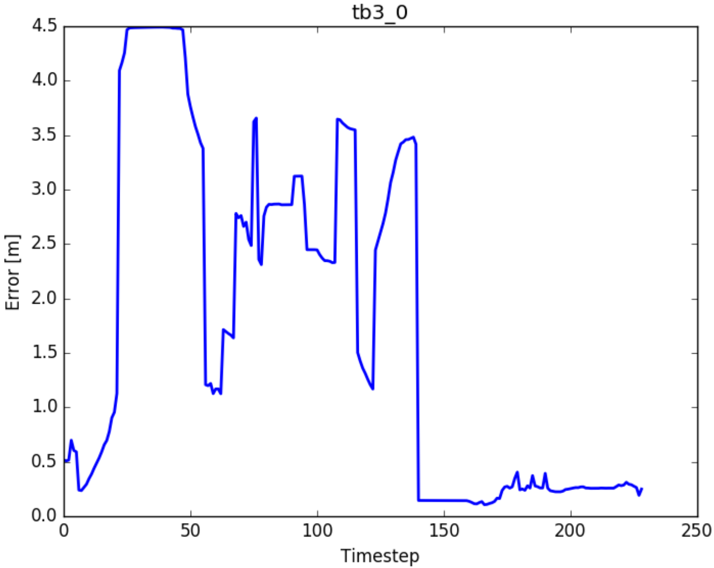
Range & Bearing

$$\bar{w}_i = \prod_{r_j \in N_i} \sum_{p_k \in r_j} \Phi\left(\begin{bmatrix} D_i(p_k) \\ \Theta_i(p_k) \end{bmatrix}, \begin{bmatrix} d_j \\ \theta_j \end{bmatrix}, \xi \right) \cdot w_{p_k}$$

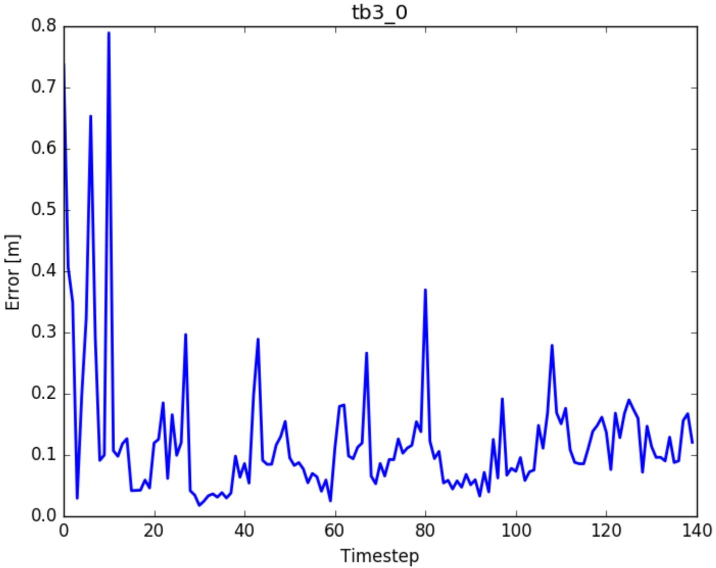
- $\bar{w}_i$ range & bearing weight of particle i
- N_i = robot i's neighbours
- p_k ranges over the set of particles from robot r_j
- d_j = received distance between this robot and robot r_j
- θ_j = received bearing of this robot from r_j
- $D_i(p_k)$ = distance between the particle i on this robot and the particle p_k from the other robot
- $\Theta_i(p_k)$ = bearing between the particle i and the particle p_k on the other robot
- w_{p_k} = weight of particle k
- ξ = covariance matrix

Normalising factors omitted.

Performance Without Enhancement



Performance With Enhancement



Demo

https://drive.google.com/file/d/
1VfTZwqM-bqTKb0AGtHgcXKm1kq8-nVVY/view?usp=sharing

Centralised Approach to World Coverage

Centralised Approach to World Coverage

- Divide the world into equal regions.

Centralised Approach to World Coverage

- Divide the world into equal regions.
- Plan a path for each robot within its region.

Centralised Approach to World Coverage

- Divide the world into equal regions.
- Plan a path for each robot within its region.
- Follow the paths.

DARP

Divide world, $\mathcal{L}$, into regions, L_i, such that:

DARP

Divide world, $\mathcal{L}$, into regions, L_i, such that:

- $L_i \cap L_j = \phi, \forall i, j \in 1..n_r, i \neq j$

DARP

Divide world, $\mathcal{L}$, into regions, L_i, such that:

- $L_i \cap L_j = \phi, \forall i, j \in 1..n_r, i \neq j$
- $L_1 \cup L_2 \cup \cdots \cup L_{n_r} = \mathcal{L}$

DARP

Divide world, $\mathcal{L}$, into regions, L_i, such that:

- $L_i \cap L_j = \phi, \forall i, j \in 1..n_r, i \neq j$
- $L_1 \cup L_2 \cup \cdots \cup L_{n_r} = \mathcal{L}$
- $|L_1| \approx |L_2| \cdots \approx |L_{n_r}|$

DARP

Divide world, $\mathcal{L}$, into regions, L_i, such that:

- $L_i \cap L_j = \phi, \forall i, j \in 1..n_r, i \neq j$
- $L_1 \cup L_2 \cup \cdots \cup L_{n_r} = \mathcal{L}$
- $|L_1| \approx |L_2| \cdots \approx |L_{n_r}|$
- L_i is connected $\forall i \in 1..n_r$

DARP

Divide world, $\mathcal{L}$, into regions, L_i, such that:

- $L_i \cap L_j = \phi, \forall i, j \in 1..n_r, i \neq j$
- $L_1 \cup L_2 \cup \cdots \cup L_{n_r} = \mathcal{L}$
- $|L_1| \approx |L_2| \cdots \approx |L_{n_r}|$
- L_i is connected $\forall i \in 1..n_r$
- $x_i(t_0) \in L_i$ (each robot starts in its own region)

DARP

The algorithm:

DARP

The algorithm:

- For each robot, weight each cell in the world based on distance to the robot.

DARP

The algorithm:

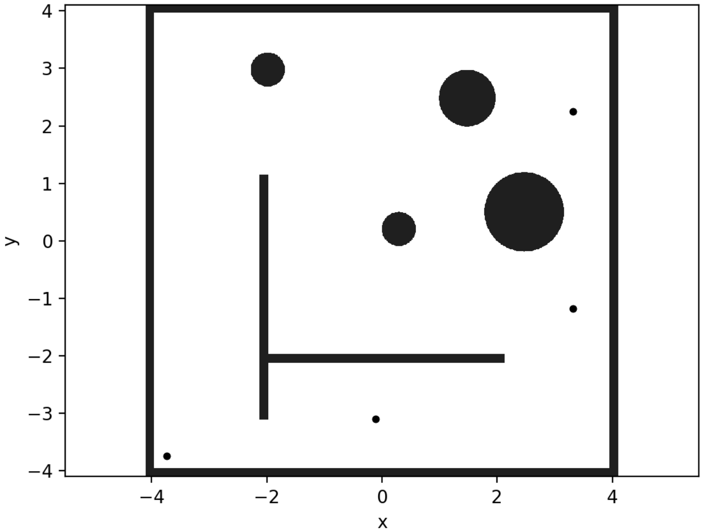
- For each robot, weight each cell in the world based on distance to the robot.
- Assign each cell to the robot that gives it the smallest weight.

DARP

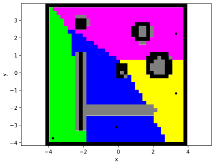
The algorithm:

- For each robot, weight each cell in the world based on distance to the robot.
- Assign each cell to the robot that gives it the smallest weight.
- Iteratively adjust weights to balance the sizes of the regions and ensure all regions are single connected components.

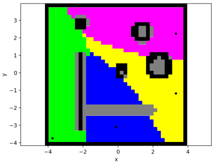
DARP



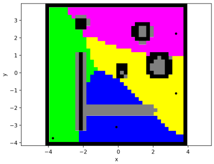
DARP



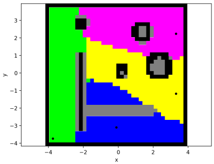
r=1



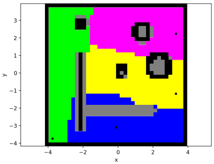
r=10



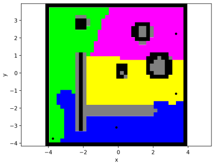
r=50



r=100



200



r=346

Regions after r iterations.

Path Planning

Path Planning

- Each cell is 2x the diameter of the robot.

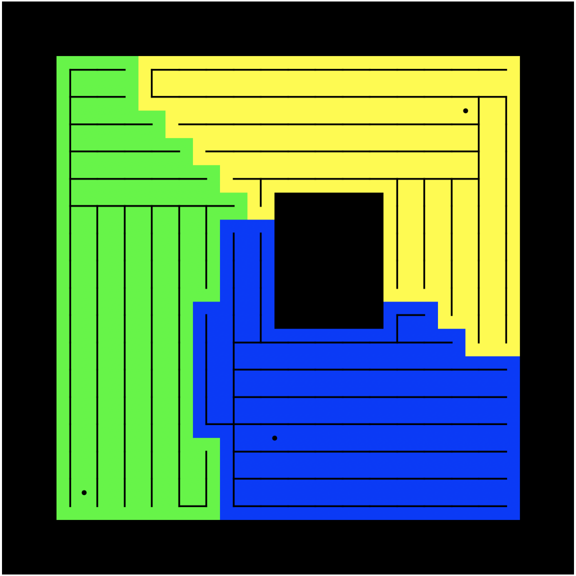
Path Planning

- Each cell is 2x the diameter of the robot.
- For each region, construct a spanning tree between the cells.

Path Planning

- Each cell is 2x the diameter of the robot.
- For each region, construct a spanning tree between the cells.
- Trace a path around the edges of the spanning tree.

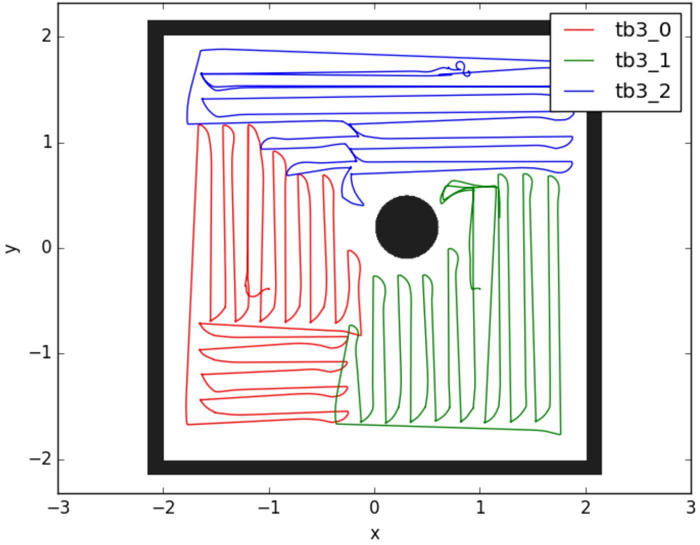
Path Planning



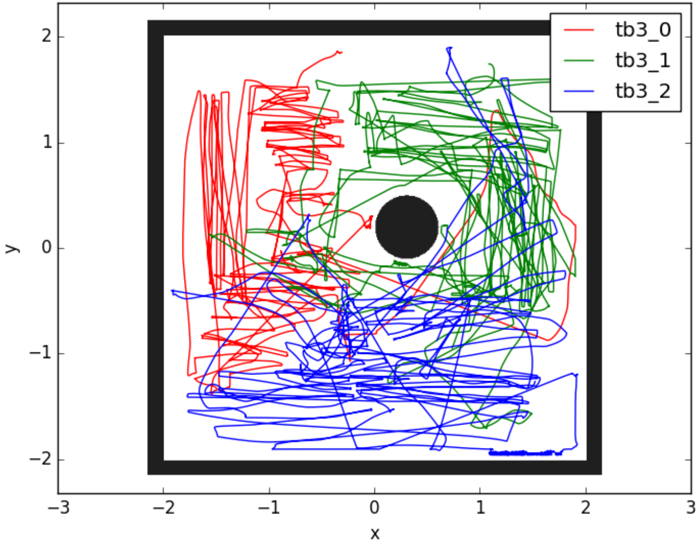
Path Following

- Extract the points on the path where a change of direction is required.
- Travel to each of these points in turn in a straight line, then rotate to face the next direction of travel.

Path Following - Ground Truth



Path Following - Localisation



Decentralized Coverage

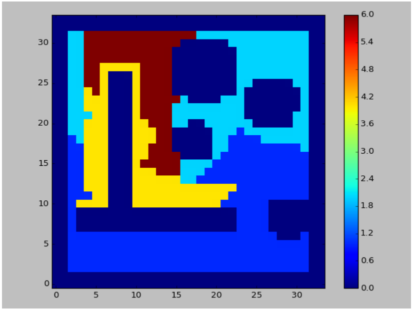
- Region Trading
- Navigation within region

Region Trading

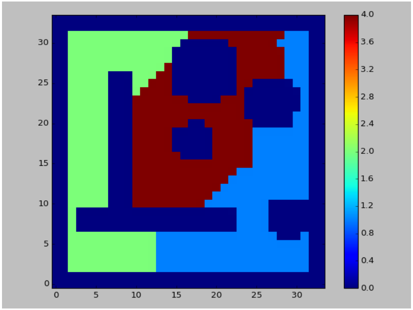
- Each robot starts out believing that it ‘owns’ the whole world
- When robots meet, they trade individual grid cells to try and balance the amount they both have
- Converges after about 10-20 random robot pairing trades.

Region Trading

After 3 trades - some area still multiply owned



After 10 trades - regions are separate



Region Trading

- Robots start by buying their positions
- Then take turns buying cells using BFS
- Cells already purchased by one robot can still be purchased by the other, if it has run out of unowned cells to buy
  - Simple strategy used to avoid one robot accidentally splitting the other's region in two by doing this

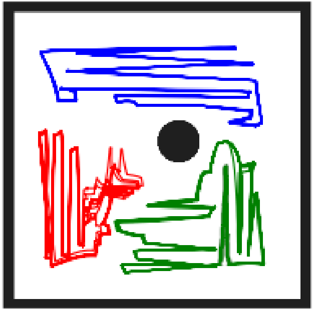
Navigation within region

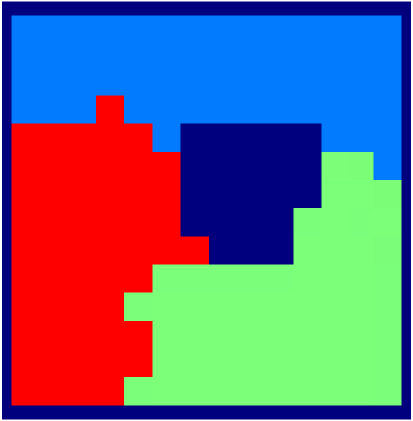
- Two approaches tried
  - Originally used RRT to each position - RRT leads to pathing everywhere
  - Luke's strategy: Following around the MST of the region
- The robots will always use RRT anyway to get back inside their region should they find themselves outside - hence paths outside of their own regions

Collision avoidance

- Phase 1 - modify speed / bearing to avoid potential collision
  - Robots moving towards each other will bear away from each other
  - Rotate away from line between the two robots
- Phase 2 - cancel paths and switch to rule based movement away from obstacle / other robot

Following MST





With Localization