Embedded Motion Control

Maze Solving Challenge - Group 4

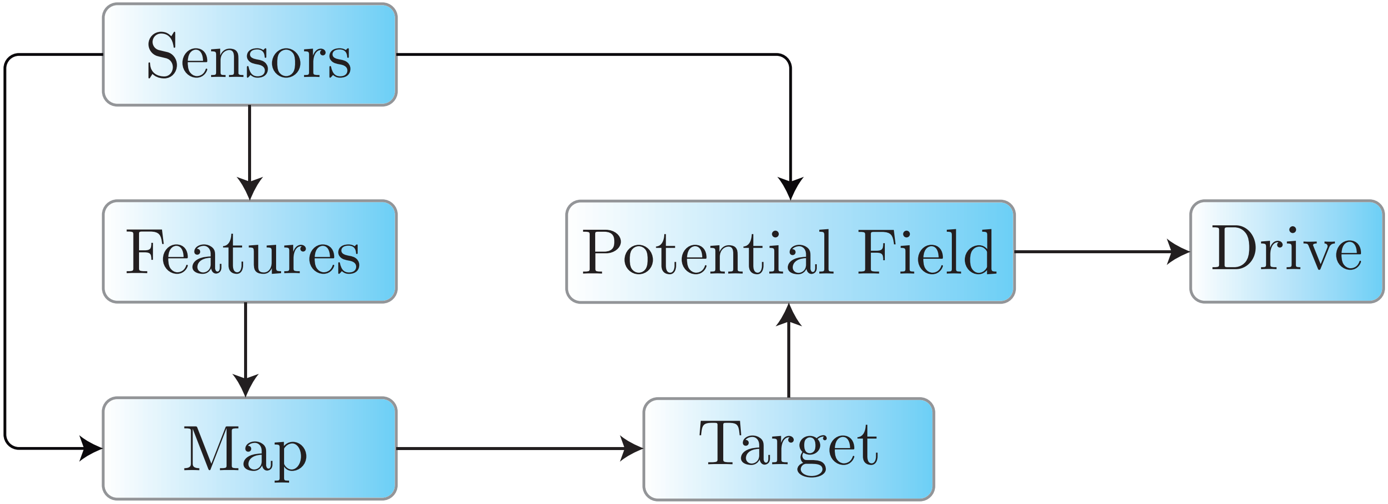
Tim Josten	0811028
Tom Leenen	0790737
Martin Plantinga	0832751
Joey Reinders	0816951

Technische Universiteit
Eindhoven
University of Technology

June 1, 2016

Contents

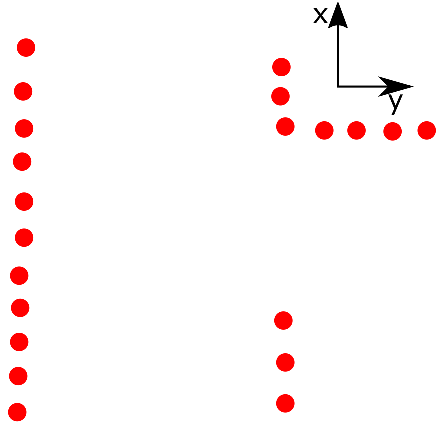
Introduction



Node detection

Line extraction (Split and Merge)

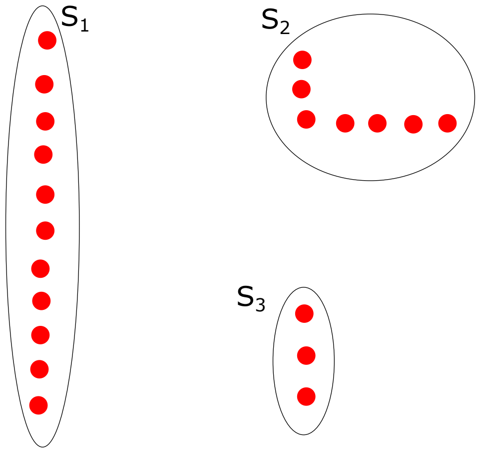
1. Compute local x and y coordinates based on LRF



Node detection

Line extraction (Split and Merge)

1. Compute local x and y coordinates based on LRF
2. Compute sets of data based on distance between points

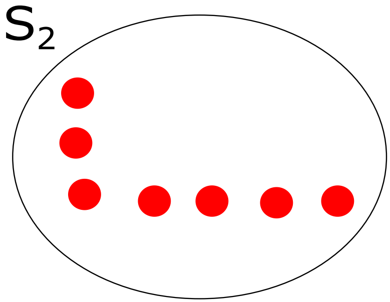


Technische Universiteit Eindhoven University of Technology

Node detection

Line extraction (Split and Merge)

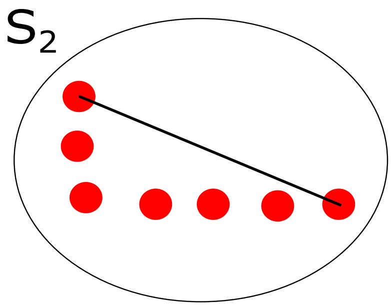
1. Compute local x and y coordinates based on LRF
2. Compute sets of data based on distance between points
3. Fit straight lines through separate sets



Node detection

Line extraction (Split and Merge)

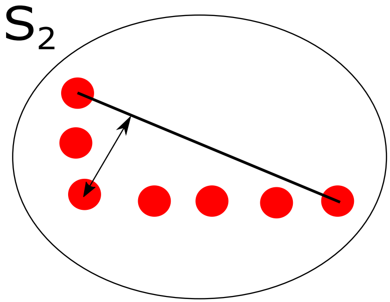
1. Compute local x and y coordinates based on LRF
2. Compute sets of data based on distance between points
3. Fit straight lines through separate sets



Node detection

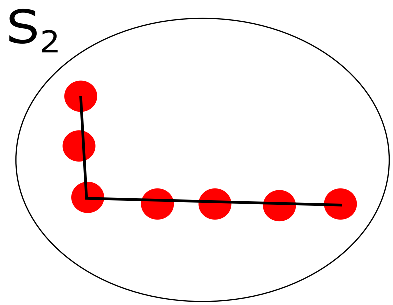
Line extraction (Split and Merge)

1. Compute local x and y coordinates based on LRF
2. Compute sets of data based on distance between points
3. Fit straight lines through separate sets
4. Determine maximal error and split set again



Line extraction (Split and Merge)

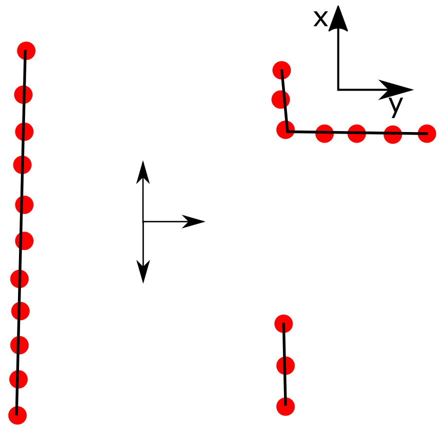
1. Compute local x and y coordinates based on LRF
2. Compute sets of data based on distance between points
3. Fit straight lines through separate sets
4. Determine maximal error and split set again
5. Repeat 3 and 4 until maximal error is small enough



Node detection

Line extraction (Split and Merge)

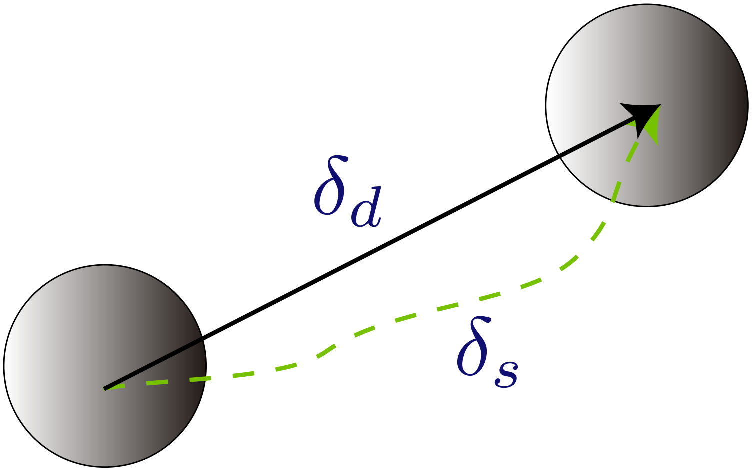
1. Compute local x and y coordinates based on LRF
2. Compute sets of data based on distance between points
3. Fit straight lines through separate sets
4. Determine maximal error and split set again
5. Repeat 3 and 4 until maximal error is small enough
6. Determine nodes local position and type



Position estimation

Position

- Using odometry data
- Small motions, updated at high frequency



Technische Universiteit Eindhoven University of Technology

Equations of Motion

- Discrete-time model

$$x_{\langle k+1\rangle} = f(x_{\langle k\rangle}, \delta_{\langle k\rangle}, v_{\langle k\rangle})$$

- New configuration in terms of previous configuration and odometry

$$\xi_{\langle k+1\rangle} = \begin{bmatrix} x_{\langle k\rangle} + (\delta_{d,\langle k\rangle} + v_d)\cos(\theta_{\langle k\rangle} + \delta_\theta + v_\theta) \\ y_{\langle k\rangle} + (\delta_{d,\langle k\rangle} + v_d)\sin(\theta_{\langle k\rangle} + \delta_\theta + v_\theta) \\ \theta_{\langle k\rangle} + \delta_\theta + v_\theta \end{bmatrix}$$

δ_d: movement in x direction
δ_θ: rotation
v_d, v_θ: error in odometry

Laser Range Finder

- Observation relative to robot

$$z = h(x_v, x_f, w)$$

x_v: robot state
x_f: location of feature
w: sensor error

- Observation of feature i

$$z = \begin{bmatrix} r \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{(y_i - y_v)^2 + (x_i - x_v)^2} \\ \tan^{-1} \frac{y_i - y_v}{x_i - x_v} - \theta_v \end{bmatrix} + \begin{bmatrix} w_r \\ w_\beta \end{bmatrix}$$

r: range
β: bearing angle

- Linearize equation

Technische Universiteit Eindhoven University of Technology

Extended Kalman Filter

1. EKF Prediction Equations

$$\hat{x}_{\langle k+1|k\rangle} = f(\hat{x}_{\langle k\rangle}, \delta_{\langle k\rangle}, 0)$$
$$\hat{P}_{\langle k+1|k\rangle} = F_{x,\langle k\rangle}\hat{P}_{\langle k|k\rangle}F^{\top}_{x,\langle k\rangle} + F_{v,\langle k\rangle}\hat{V}F^{\top}_{v,\langle k\rangle}$$

2. EKF Sensor Update Equations

$$\begin{aligned}
\hat{x}_{\langle k+1|k+1\rangle} &= \hat{x}_{\langle k+1|k\rangle} + K_{\langle k+1\rangle}\nu_{\langle k+1\rangle} \\
\hat{P}_{\langle k+1|k+1\rangle} &= \hat{P}_{\langle k+1|k\rangle}F^{\top}_{x,\langle k\rangle} - K_{\langle k+1\rangle}H_{x,\langle k+1\rangle}\hat{P}_{\langle k+1|k\rangle} \\
S_{\langle k+1\rangle} &= H_{x,\langle k+1\rangle}\hat{P}_{\langle k+1|k\rangle}H^{\top}_{x,\langle k+1\rangle} + H_{w,\langle k+1\rangle}\hat{W}_{\langle k+1\rangle}H^{\top}_{w,\langle k+1\rangle} \\
K_{\langle k+1\rangle} &= \hat{P}_{\langle k+1|k\rangle}H^{\top}_{x,\langle k+1\rangle}S^{-1}_{\langle k+1\rangle}
\end{aligned}$$

Map Making

- State vector with estimated coordinates fo m landmarks

$$\hat{x} = (x_1, y_1, \ldots, x_m, y_m)^\top$$

- Prediction equations

$$\begin{aligned}\hat{x}_{\langle k+1|k\rangle} &= \hat{x}_{\langle k|k\rangle}\\ \hat{P}_{\langle k+1|k\rangle} &= \hat{P}_{\langle k|k\rangle}\end{aligned}$$

- State vector evolution

$$g(x_v, z) = \begin{bmatrix} x_v + r_z \cos(\theta_v + \theta_z) \\ y_v + r_z \sin(\theta_v + \theta_z) \end{bmatrix}$$

- Adding landmarks

$$x^\star_{\langle k|k\rangle} = y(x_{\langle k|k\rangle}, z_{\langle k\rangle}, x_{v,\langle k|k\rangle}) = \begin{bmatrix} x_{\langle k|k\rangle} \\ g(x_{v,\langle k|k\rangle}, z_{\langle k\rangle}) \end{bmatrix}$$

Simultaneous Localization and Mapping

- State Vector

$$\hat{x} = (x_v, y_v, \theta_v, x_1, y_1, ..., x_m, y_m)$$

- Feature observation wrt state vectors:

$$H_x = (H_{x_v} \ldots 0 \ldots H_{x_i} \ldots 0)$$

Path planning

Maze solving

- Tremaux algorithm to solve the maze
  1. Initially at a node a random direction is chosen
  2. Node is marked
  3. When the robot comes to a node for the second time the unmarked direction is chosen
- Using the global map to determine where the robot has been

Path planning

- Pico is guided using the potential field
- Turns are taken using a point of attraction