Closed loop control with odometry error modeling

Beyond closed loop control

- Deterministic robotics tends to consider all information from all sensors as true information
- The closed loop control applied earlier is only perfect in a perfect world
- In the real world, all sensors have errors and the mathematical modeling of the system is not perfect
- Good example of erroneous feedback is the encoders of the wheels
- Encoders might be perfect, but slippage would result in a drift in orientation or translation
- Simple closed loop control does not consider these errors

Control inputs

- There are two different types of inputs.

1. The first specifies velocity commands given to the robot's motors (differential drive, synchro drive)
2. The second model assumes that one is provided with odometry information (distance traveled, angle turned).

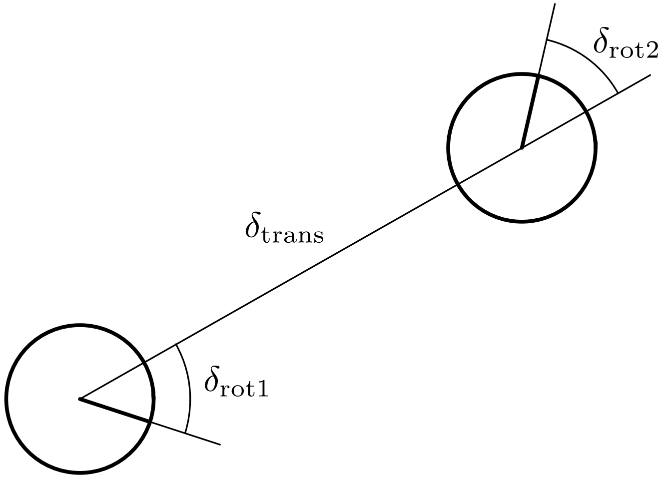
- The two models are applied differently for integrating such information.
- In practice, odometry models tend to be more accurate than velocity models.

Control inputs

- Odometry is only available post-the-fact, so it cannot be used for motion planning.
- Planning algorithms such as collision avoidance have to predict the effects of motion.
- Therefore, odometry models are usually applied for estimation, whereas velocity models are used for probabilistic motion planning.

Improving the closed loop control algorithm?

- We will only prepare the odometry model, at every reading from the robot, we will plot the location of the robot according to the deterministic model.
- We will also plot 100 points with an error in δ_{rot1}, δ_{trans}, δ_{rot2}



The logic

$\bar{x}_{t-1} = (\bar{x}, \bar{y}, \bar{\theta})$ $\;x, y, \theta$ are previous values from probabilistic model

$\bar{x}_t = (\bar{x}', \bar{y}', \bar{\theta}')$ calculated from odometry as done in the deterministic model

Algorithm sample_motion_model_odometry(u_t, x_{t-1})**:**

$\delta_{\text{rot1}} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$

$\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^s}$

$\delta_{\text{rot2}} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot1}}$

$\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \textbf{sample}(\alpha_1 \delta_{\text{rot1}} + \alpha_2 \delta_{\text{trans}})$

$\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \textbf{sample}(\alpha_3\, \delta_{\text{trans}} + \alpha_4(\delta_{\text{rot1}} + \delta_{\text{rot2}}))$

$\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \textbf{sample}(\alpha_1 \delta_{\text{rot2}} + \alpha_2 \delta_{\text{trans}})$

$x' = x + \hat{\delta}_{\text{trans}}\, \cos(\theta + \hat{\delta}_{\text{rot1}})$

$y' = y + \hat{\delta}_{\text{trans}}\, \sin(\theta + \hat{\delta}_{\text{rot1}})$

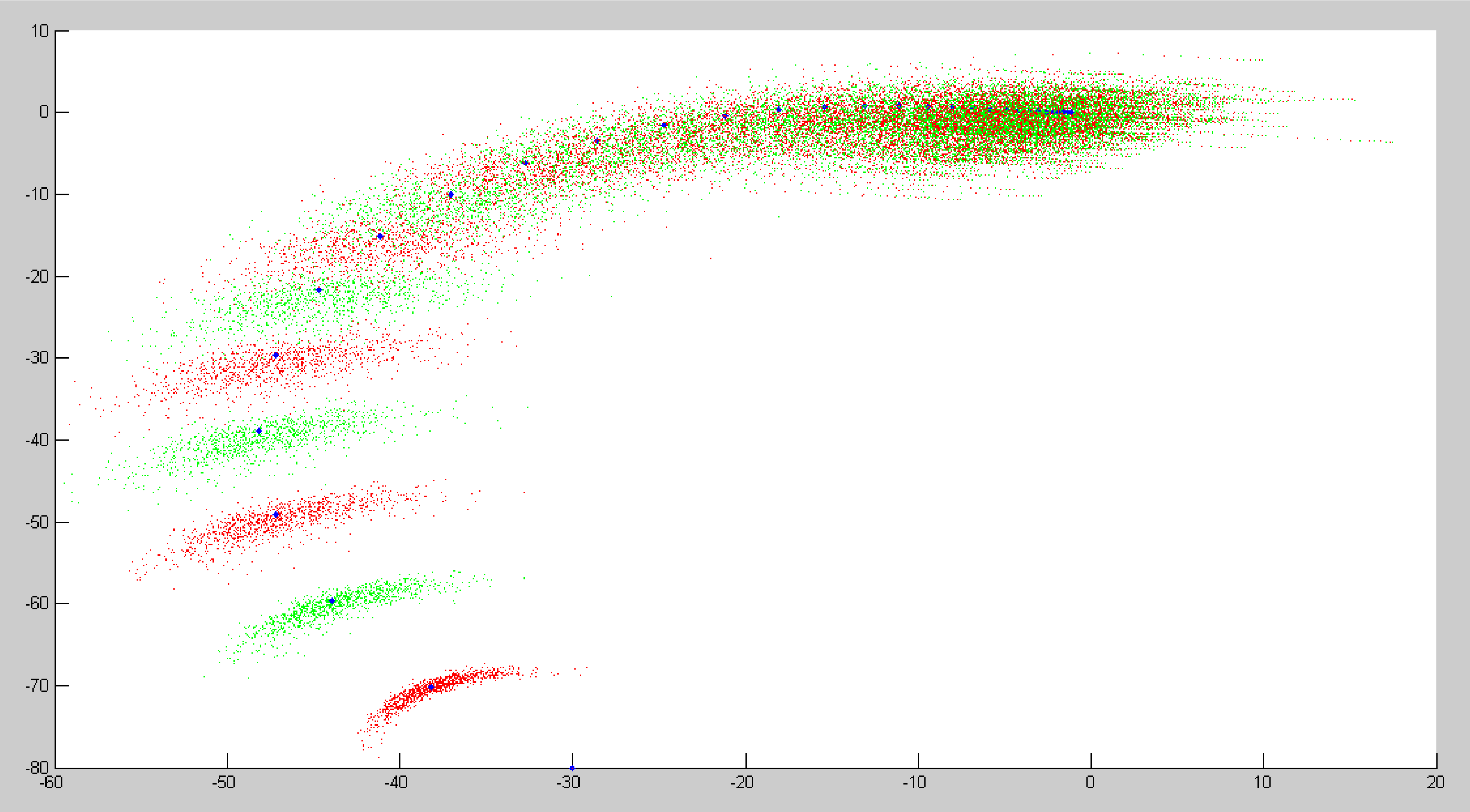
$\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$

return $x_t = (x', y', \theta')^T$

The challenge

- Modify the closed loop control algorithm and change the position x, y, theta of the robot into an array of 101 elements each.
- The first element x(1), y(1), theta(1) should show the position of the robot according to the deterministic model.
- The rest of the array should be filled by applying the algorithm shown above, the points should be plotted as the robot moves.

The outcome



Steps

- Initialize arrays x(101), y(101), theta(101) with initial value being the initial position of the robot as done in the previous challenge
- Calculate deltaX, deltaY, deltaTheta
- Calculate new position $\bar{x}_t = (\bar{x}', \bar{y}', \bar{\theta}')$ from deterministic model, do not update $x(1), y(1), theta(1)$
- Loop for array values i:2~100 where:
  $(x, y, \theta) = x(i), y(i), theta(i)$ (**previous** value)
  $\bar{x}_{t-1} = (\bar{x}, \bar{y}, \bar{\theta}) = [x(1), y(1), theta(1)]$
- The returned values should be saved in $x(i), y(i), theta(i)$
- Update x(1), y(1),theta(1)