# Simultaneous Localization and Mapping (SLAM)

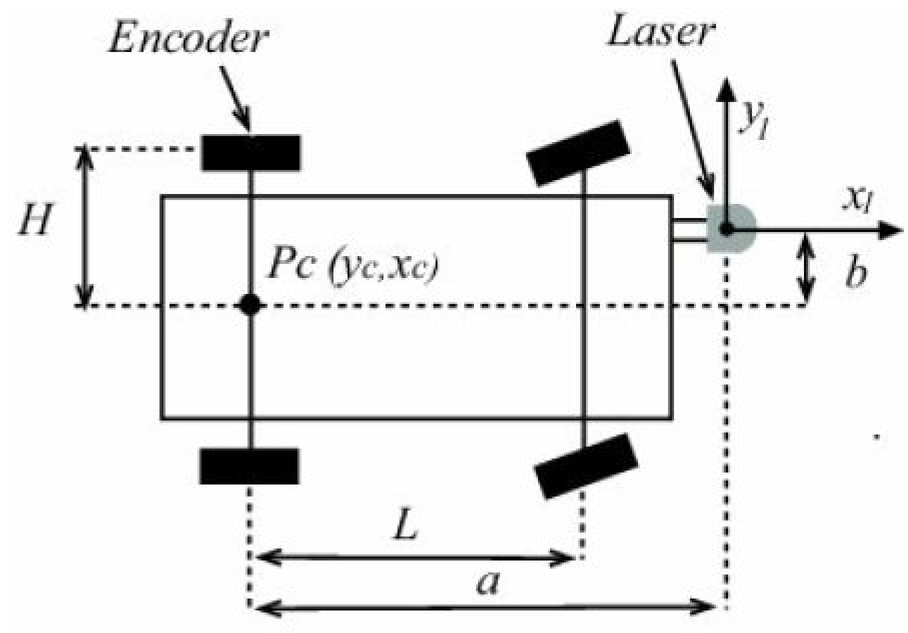
Lecture 04

# SLAM Implementation

## Step 1

- Derive vehicle model

$$\begin{bmatrix} \dot{x}_c \\ \dot{y}_c \\ \dot{\phi}_c \end{bmatrix} = \begin{bmatrix} v_c \cos\phi \\ v_c \sin\phi \\ {v_c}/{L} \tan\alpha \end{bmatrix}$$




- Translating our model to the laser point and differentiating

$$\begin{bmatrix} x_v \\ y_v \end{bmatrix} = \begin{bmatrix} x_c + a\cos\phi - b\sin\phi \\ y_c + a\sin\phi + b\cos\phi \end{bmatrix} \Rightarrow \begin{bmatrix} \dot{x}_v \\ \dot{y}_v \end{bmatrix} = \begin{bmatrix} \dot{x}_c - (a\sin\phi + b\cos\phi)\ \dot{\phi} \\ \dot{y}_c + (a\cos\phi - b\sin\phi)\ \dot{\phi} \end{bmatrix}$$

# SLAM Implementation

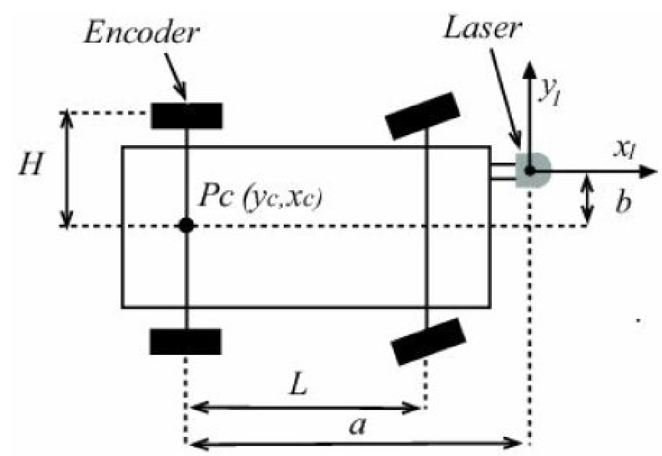
## Step 1

- The vehicle model in discrete time is

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \phi(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \Delta t v_c \cos\phi - \Delta t \frac{v_c}{L} \tan\alpha (a \sin\phi + b\cos\phi) \\ y(k) + \Delta t v_c \sin\phi + \Delta t \frac{v_c}{L} \tan\alpha (a \cos\phi - b\sin\phi) \\ \phi(k) + \Delta t \frac{v_c}{L} \tan\alpha \end{bmatrix}$$

- The velocity is measured with an encoder at the back wheel, thus $v_c$ is

$$v_c = \frac{v_e}{1 - \frac{h}{L}\tan\alpha}$$



# SLAM Implementation

## Step 2

- Calculate the Jacobian (δf/δx)

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \phi(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \Delta t v_c \cos\phi - \Delta t \dfrac{v_c}{L} \tan\alpha (a \sin\phi + b\cos\phi) \\ y(k) + \Delta t v_c \sin\phi + \Delta t \dfrac{v_c}{L} \tan\alpha (a \cos\phi - b\sin\phi) \\ \phi(k) + \Delta t \dfrac{v_c}{L} \tan\alpha \end{bmatrix}$$

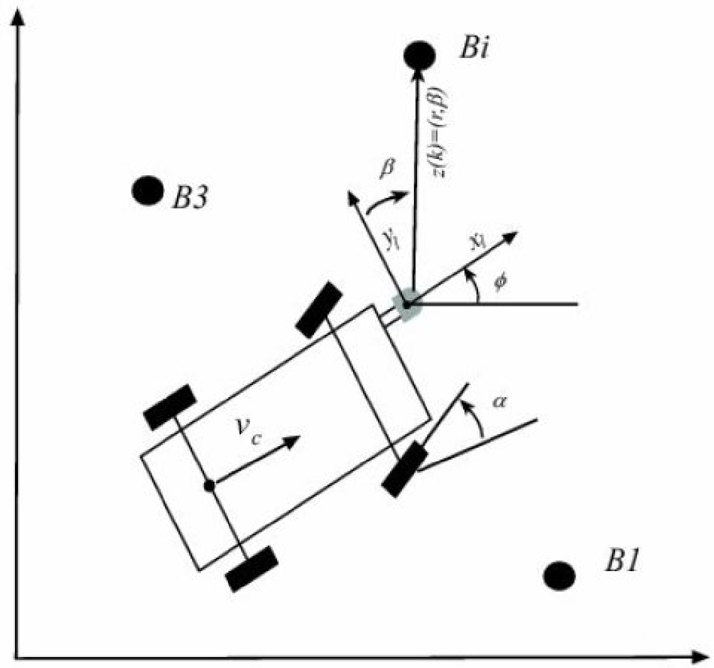
$$\frac{\partial f}{\partial x} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x} & \dfrac{\partial f_1}{\partial y} & \dfrac{\partial f_1}{\partial \phi} \\ \dfrac{\partial f_2}{\partial x} & \dfrac{\partial f_2}{\partial y} & \dfrac{\partial f_2}{\partial \phi} \\ \dfrac{\partial f_3}{\partial x} & \dfrac{\partial f_3}{\partial y} & \dfrac{\partial f_3}{\partial \phi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta t v_c \sin\phi - \Delta t \dfrac{v_c}{L} \tan\alpha (a\cos\phi - b\sin\phi) \\ 0 & 1 & \Delta t v_c \cos\phi - \Delta t \dfrac{v_c}{L} \tan\alpha (a\sin\phi + b\cos\phi) \\ 0 & 0 & 1 \end{bmatrix}$$

# SLAM Implementation

## Step 3

- Derive sensor model

$$\begin{bmatrix} z_r \\ z_\beta \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i - x_v)^2 + (y_i - y_v)^2} \\ \tan^{-1}\left(\dfrac{y_i - y_v}{x_i - x_v}\right) - \phi + \dfrac{\pi}{2} \end{bmatrix}$$




where $x_i$ and $y_i$ are the landmark locations

# SLAM Implementation

## Step 4

- Calculate the Jacobian (δh/δx)

$$\begin{bmatrix} z_r \\ z_\beta \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i - x_v)^2 + (y_i - y_v)^2} \\ \tan^{-1}\left(\dfrac{y_i - y_v}{x_i - x_v}\right) - \phi + \dfrac{\pi}{2} \end{bmatrix}$$

$$J_h(k) = \begin{bmatrix} \dfrac{\partial h_1}{\partial x} & \dfrac{\partial h_1}{\partial y} & \dfrac{\partial h_1}{\partial \phi} \\ \dfrac{\partial h_2}{\partial x} & \dfrac{\partial h_2}{\partial y} & \dfrac{\partial h_2}{\partial \phi} \end{bmatrix} = \begin{bmatrix} \dfrac{x_v - x_i}{r} & \dfrac{y_v - y_i}{r} & 0 \\ \dfrac{y_i - y_v}{r^2} & \dfrac{x_v - x_i}{r^2} & -1 \end{bmatrix}$$

where $r = \sqrt{(x_i - x)^2 + (y_i - y)^2}$

# SLAM Implementation

## Step 5

- Initialize state vector

$$\begin{bmatrix} x(0) \\ y(0) \\ \phi(0) \end{bmatrix} = \begin{bmatrix} 5.1 \\ 4.2 \\ -2.2 \end{bmatrix}$$

- Starting from t(0), make first prediction of where vehicle will be at t(0 + $\Delta$t)

$$\begin{bmatrix} x(k+1) \\ y(k+1) \\ \phi(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + \Delta t v_c \cos\phi - \Delta t \frac{v_c}{L} \tan\alpha (a \sin\phi + b \cos\phi) \\ y(k) + \Delta t v_c \sin\phi + \Delta t \frac{v_c}{L} \tan\alpha (a \cos\phi - b \sin\phi) \\ \phi(k) + \Delta t \frac{v_c}{L} \tan\alpha \end{bmatrix}$$

# SLAM Implementation

## Step 6

- Initialize covariance matrix

$$P(0) = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 15\frac{\pi}{180} \end{bmatrix}$$

Initial estimates for x,y could be 0.1 meters off while the angle could be 15 degrees off

- Initialize Q and W matrices

$$Q = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & (15\frac{\pi}{180})^2 \end{bmatrix}$$

$$W = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Calculate prediction for covariance matrix

$$Pest = AP(0)A^T + WQW^T$$

# SLAM Implementation

## Step 7

- Check to see if new velocity and steering measurements are available
  - If so, then update $v_e$ and $\alpha$ before next prediction

$$v_c = \frac{v_e}{1 - \frac{h}{L}\tan\alpha}$$

  - If not, then use previous measurements for next prediction

- Check to see if a new laser scan is available
  - If so, then move to step 8
  - If not, then move to next iteration (i.e. $t + \Delta t$)

# SLAM Implementation

## Step 8

- From a single 180 degree scan (SICK laser), determine
  - The number of landmarks in a scan
  - The center of each landmark

- For i = 1 to the number of landmarks in a scan
  - If current landmark is the first observed landmark (i.e. # states=3)
    - Add x and y coordinate of new landmark to state vector

$$x_4 = x + range * \cos(\beta + \phi - \pi / 2)$$

$$x_5 = y + range * \sin(\beta + \phi - \pi / 2)$$

    - Increase the size of all matrices (i.e. A is now a 5x5 matrix)
    - Update state and covariance matrices

$$xest(:,k) = xest(:,k) + K * innov$$

$$Pest = (I - K * J_h) * Pest$$

Where innov = actual measurement – predicted measurement (from model)

# SLAM Implementation

## Step 9

- For i = 1 to the number of landmarks in a scan
  - If current landmark is NOT the first observed landmark (i.e. # states>3)
    - Estimate position of observation

$$x_4 = x + range * \cos(\beta + \phi - \pi / 2)$$
$$x_5 = y + range * \sin(\beta + \phi - \pi / 2)$$

    - Compute distance between observation and all landmarks already incorporated into state vector
    - Find landmark which is closest to observation
    - If landmark is within 1 meter of observation, assume it is the same landmark
    - Otherwise add observation as new landmark
    - Update!

$$xest(:,k) = xest(:,k) + K * innov$$
$$Pest = (I - K * J_h) * Pest$$

Where innov = actual measurement – predicted measurement (from model)