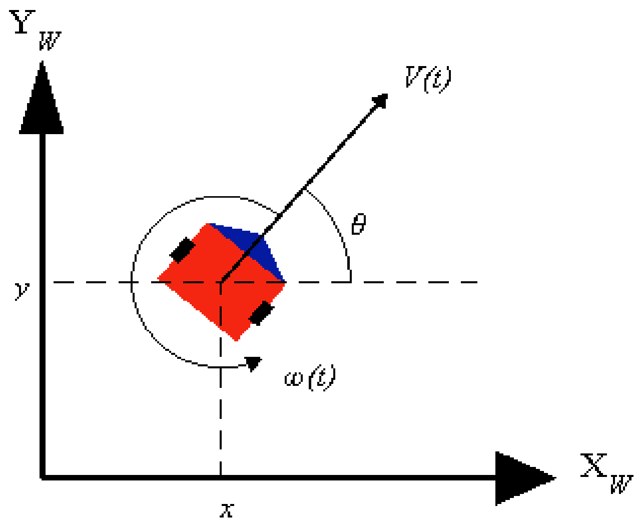
Mobile robots - Localisation in presence of uncertainties -

Dr Alexandru Stancu

Dr Mario Martinez

Kinematics of a differential drive

The Kinematic model for a non-holonomic mobile robot is



$$\dot{x} = v\, cos\theta$$

$$\dot{y} = v\, sin\theta$$

$$\dot{\theta} = \omega$$

Transformation

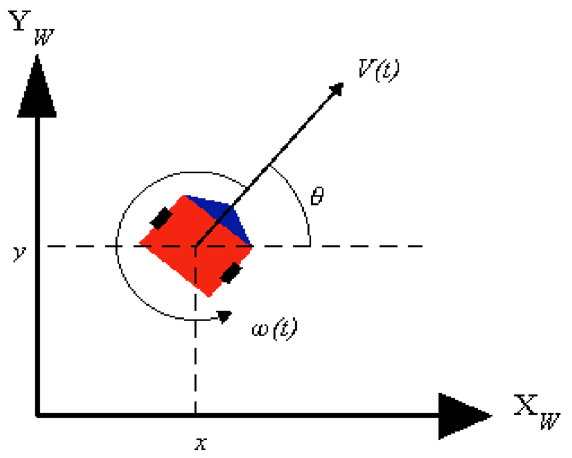
$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} {}^{r}/_{2} & {}^{r}/_{2} \\ {}^{r}/_{l} & -{}^{r}/_{l} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \longrightarrow$$

$$\dot{x} = r\frac{\omega_r + \omega_l}{2} cos\theta$$

$$\dot{y} = r\frac{\omega_r + \omega_l}{2} sin\theta$$

$$\dot{\theta} = r\frac{\omega_r - \omega_l}{l}$$

Motion-based Localisation (Dead Reckoning)



We use the notation s_x, s_y, and s_θ for the states of the mobile robot.

$$\dot{\mathbf{s}} = \mathbf{h}(\mathbf{s}, \mathbf{u})$$

$$\frac{d}{dt}\begin{bmatrix} s_x \\ s_y \\ s_\theta \end{bmatrix} = \begin{bmatrix} \cos(s_\theta) & 0 \\ \sin(s_\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

The robot pose $\mathbf{s}_k = [s_x \quad s_y \quad s_\theta]^T$

The robot inputs $\mathbf{u}_k = [v \quad \omega]^T$

Using Euler, we discretise the continuous time model

Motion-based Localisation (Dead Reckoning)

If Δt is the sampling time, then it is possible to compute the incremental linear and angular displacements, Δd and $\Delta\theta$, as follows:

$$\Delta d = v \cdot \Delta t \qquad\qquad \Delta\theta = \omega \cdot \Delta t$$

$$\begin{bmatrix} \Delta d \\ \Delta\theta \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 \\ 1/l & -1/l \end{bmatrix} \begin{bmatrix} \Delta d_r \\ \Delta d_l \end{bmatrix}$$

To compute the pose of the robot at any given time step, the kinematic model must be numerically integrated.

The current robot pose depends only on the previous pose and the input velocities.

$$\begin{bmatrix} s_{x,k} \\ s_{y,k} \\ s_{\theta,k} \end{bmatrix} = \begin{bmatrix} s_{x,k-1} \\ s_{y,k-1} \\ s_{\theta,k-1} \end{bmatrix} + \begin{bmatrix} \Delta d \cos\left(s_{\theta,k-1}\right) \\ \Delta d \sin\left(s_{\theta,k-1}\right) \\ \Delta\theta \end{bmatrix} \qquad\qquad \mathbf{s}_k = \mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)$$

But…

Motion-based Localisation (Dead Reckoning)

The pose estimation of a mobile robot is **always associated with some uncertainty** with respect to its state parameters.

From a geometric point of view, the error in differential-drive robots is classified into three groups:

- **Range error:** it is associated with the computation of Δd over time.
- **Turn error:** it is associated with the computation of $\Delta \theta$ over time.
- **Drift error:** it is associated with the difference between the angular speed of the wheels and it affects the error in the angular rotation of the robot.

Motion-based Localisation (Dead Reckoning)

Due to such uncertainty, it is possible to represent **the belief of the robot pose** by a Gaussian distribution, where

- the mean vector $\boldsymbol{\mu}_k$ is the best estimate of the pose, and
- the covariance matrix $\boldsymbol{\Sigma}_k$ is the uncertainty of the pose that encapsulates the errors presented in the previous slide.

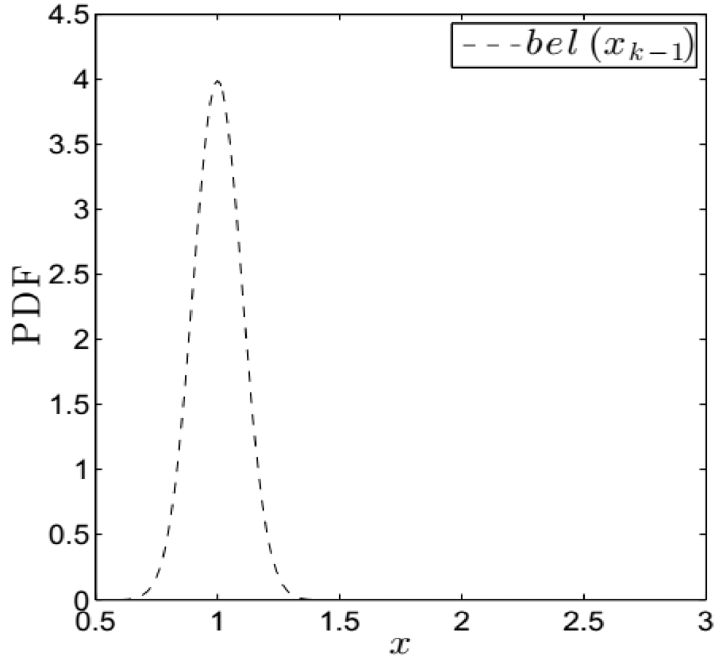
The Gaussian distribution (or normal distribution) is denoted by

$$\mathbf{s}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

Gaussian distributions

A random variable X is *normally distributed*, or *Gaussian*, if its probability density function is defined as:

$$p_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(x-\mu_X)^2}{2\sigma^2}\right)$$



where, μ_X, σ^2 are the *mean* and *variance*, respectively; they are the distribution parameters. The notation $X \sim \mathcal{N}\,(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$ means that the random variable *X* is Gaussian.

Motion-based Localisation (Dead Reckoning)

In the context of probability, the robot pose at time step k, denoted by $\mathbf{s}_k$, can be described as function of previous robot pose $\mathbf{s}_{k-1}$ and the current control input $\mathbf{u}_k = [v_k \quad \omega_k]^T$. This process is called the ***robot motion model***.

$$\mathbf{s}_k = \mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k) + \mathbf{q}_k$$

where $\mathbf{q}_k$ is an additive Gaussian noise such that $\mathbf{q}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$, and $\mathbf{Q}_k$ is a positive semidefinite covariance matrix. This noise is directly related to the error sources described before.

At this stage we assume that we already have obtained $\mathbf{Q}_k$ experimentally. However, we'll come back later and do the experiments to calibrate $\mathbf{Q}_k$.

Motion-based Localisation (Dead Reckoning)

If $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ and $\mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)$ is a nonlinear function and

$$\mathbf{s}_k = \mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k) + \mathbf{q}_k$$

then $\mathbf{s}_k$ is not a Gaussian distribution anymore.

Motion-based Localisation (Dead Reckoning)

If $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ and $\mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)$ is a nonlinear function and

$$\mathbf{s}_k = \mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k) + \mathbf{q}_k$$

then $\mathbf{s}_k$ is not a Gaussian distribution anymore.

Solution?

What we should do to preserve the propagation of the Gaussian distribution?

Motion-based Localisation (Dead Reckoning)

If $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{k-1})$ and $\mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)$ is a nonlinear function and

$$\mathbf{s}_k = \mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k) + \mathbf{q}_k$$

then $\mathbf{s}_k$ is not a Gaussian distribution anymore.

Linearisation!!!

- This is supported by the Affine Transformation result.

Gaussian distributions

Affine Transformation

Consider $X \sim \mathcal{N}(\boldsymbol{\mu}_X, \boldsymbol{\Sigma}_X)$ in $\mathbb{R}^n$, and let $Y = \mathbf{A}X + \boldsymbol{b}$ be the affine transformation, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$. Then, the random vector $Y \sim \mathcal{N}(\boldsymbol{\mu}_Y, \boldsymbol{\Sigma}_Y)$ such that:

$$\boldsymbol{\mu}_Y = \mathbf{A}\boldsymbol{\mu}_X + \mathbf{b}$$

$$\boldsymbol{\Sigma}_Y = \mathbf{A}\boldsymbol{\Sigma}_X\mathbf{A}^{\boldsymbol{T}}$$

Linearisation

$$\mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k) = \begin{bmatrix} s_{x,k-1} + \Delta t \cdot v_k . \cos\left(s_{\theta,k-1}\right) \\ s_{y,k-1} + \Delta t \cdot v_k . \sin\left(s_{\theta,k-1}\right) \\ s_{\theta,k-1} + \Delta t \cdot \omega_k \end{bmatrix}$$

The following equation represents the Jacobian matrix of $\mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)$ with respect to each variable in $\mathbf{s}_{k-1}$, evaluated at $\mathbf{s}_{k-1} = \boldsymbol{\mu}_{k-1}$:

$$\mathbf{H}_k = \nabla_{\mathbf{s}_k} \mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)\Big|_{\mathbf{s}_{k-1}=\boldsymbol{\mu}_{k-1}}$$

Linearisation

$$\mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k) = \begin{bmatrix} s_{x,k-1} + \Delta t \cdot v_k . \cos\left(s_{\theta,k-1}\right) \\ s_{y,k-1} + \Delta t \cdot v_k . \sin\left(s_{\theta,k-1}\right) \\ s_{\theta,k-1} + \Delta t \cdot \omega_k \end{bmatrix}$$

$$\mathbf{H}_k = \begin{bmatrix} \frac{\partial h_1}{\partial s_{x,k}} & \frac{\partial h_1}{\partial s_{y,k}} & \frac{\partial h_1}{\partial s_{\theta,k}} \\ \frac{\partial h_2}{\partial s_{x,k}} & \frac{\partial h_2}{\partial s_{y,k}} & \frac{\partial h_2}{\partial s_{\theta,k}} \\ \frac{\partial h_3}{\partial s_{x,k}} & \frac{\partial h_3}{\partial s_{y,k}} & \frac{\partial h_3}{\partial s_{\theta,k}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta t \cdot v_k \cdot \sin(\mu_{\theta,k-1}) \\ 0 & 1 & \Delta t \cdot v_k \cdot \cos(\mu_{\theta,k-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

Motion-based Localisation (Dead Reckoning)

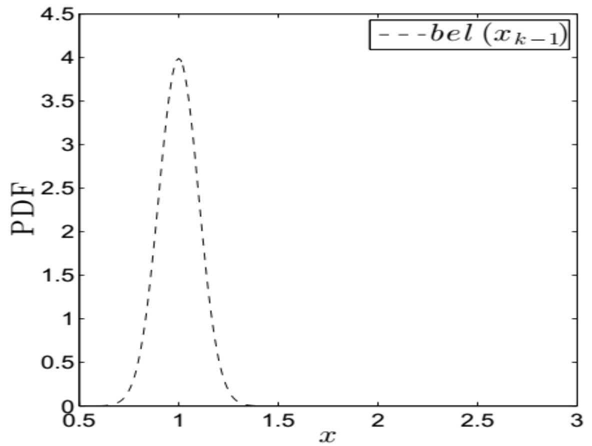
Assume that the robot pose at time step $k-1$ is given by a Gaussian distribution such that $\mathbf{s}_{k-1} \sim \mathcal{N}(\boldsymbol{\mu}_{k-1}, \boldsymbol{\Sigma}_{k-1})$, then, the above setup can be used to estimate the robot pose at time step k by linearising the robot motion model using first-order Taylor expansion around $\boldsymbol{\mu}_k$ as follows:

$$\boldsymbol{\mu}_k = \mathbf{h}(\boldsymbol{\mu}_{k-1}, \mathbf{u}_k)$$

and the pose $\mathbf{s}_k$ is computed using the linearised system:

$$\mathbf{s}_k \approx \boldsymbol{\mu}_k + \mathbf{H}_k(\mathbf{s}_{k-1} - \boldsymbol{\mu}_{k-1})$$

where $\mathbf{s}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$



Motion-based Localisation (Dead Reckoning)

Since the robot motion model is linearised and all uncertainties are Gaussians, it is possible to compute the covariance $\mathbf{\Sigma}_k$ associated with the robot pose at time step k using the properties of Gaussians:

$$\mathbf{\Sigma}_k = \mathbf{H}_k \mathbf{\Sigma}_{k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{Q}_k$$

Thus, the estimated pose at time step k is Gaussian such that $\mathbf{s}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$, and it is computed recursively using the pose at time step $k-1$ and the input vector $\mathbf{u}_k$. The initial robot pose is assumed known, the robot starts in zero, i.e., all states are zero.

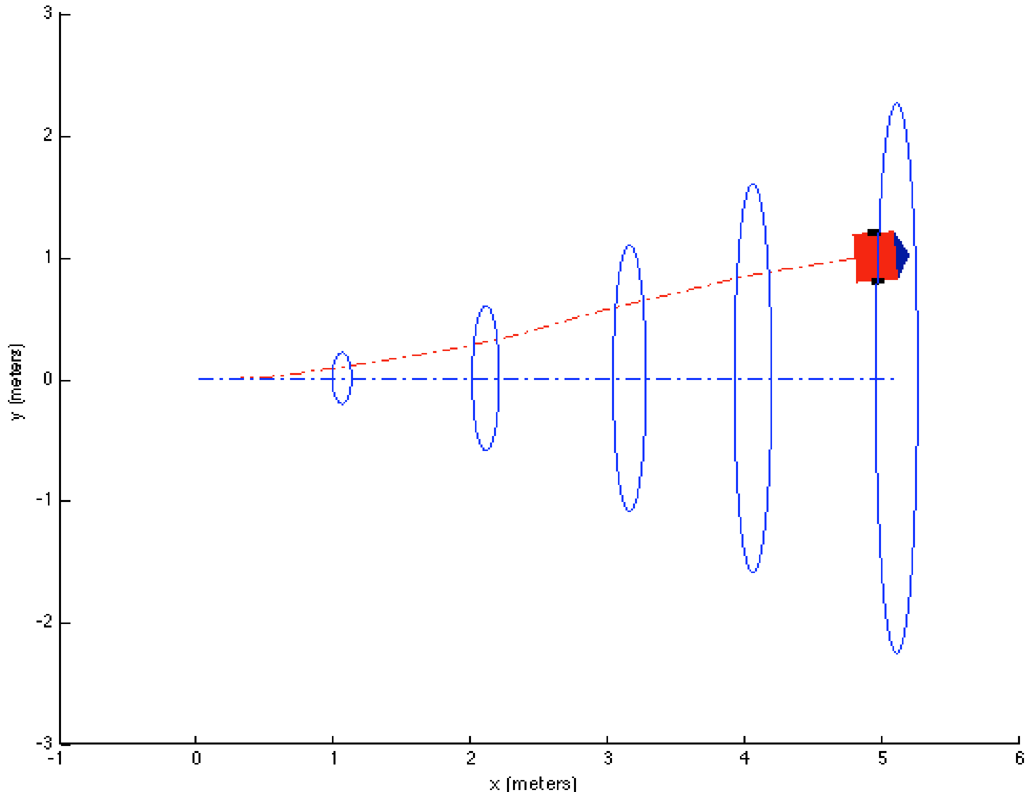
The pose uncertainty will always increase every time the robot moves due to the addition of the nondeterministic error represented by $\mathbf{Q}_k$, which is positive semi-definite.

Motion-based Localisation (Dead Reckoning)

$\boldsymbol{\mu}_0 = \mathbf{0}$, and $\boldsymbol{\Sigma}_0 = \mathbf{0}$

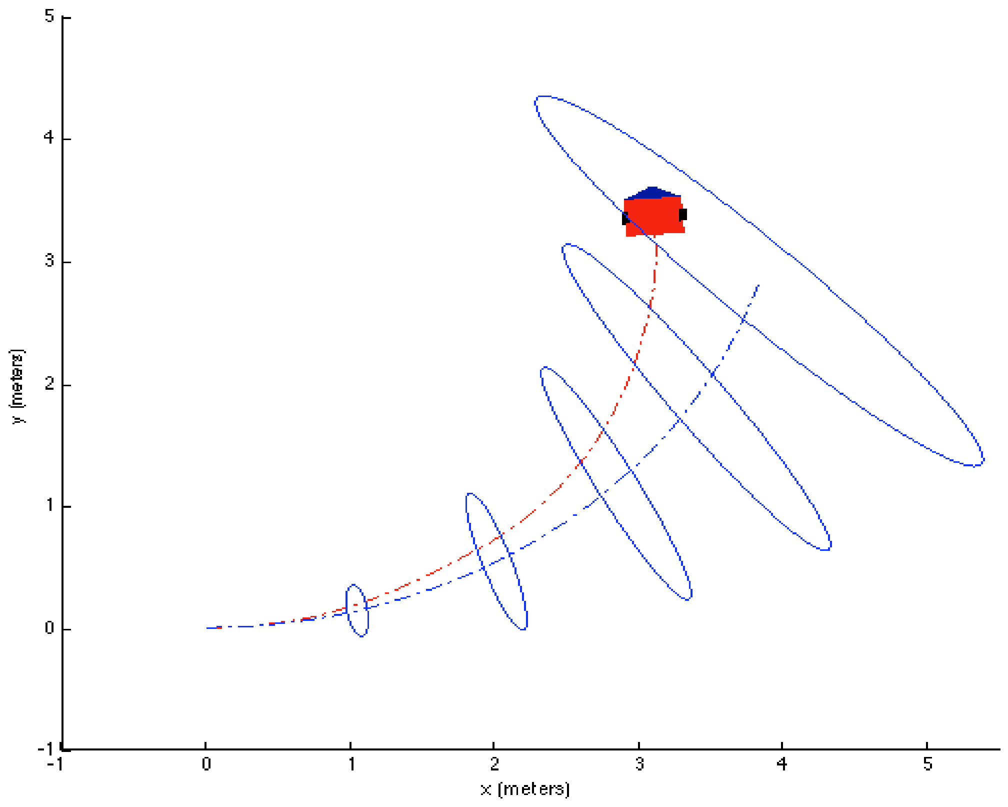
Motion-based Localisation (Dead Reckoning)

The joint uncertainty of s_x and s_y is represented by an ellipsoid around the robot. This ellipsoid is named **Ellipsoid of Confidence**.
As the robot moves along the x-axis, its uncertainty along the y-axis increases faster than the x-axis due to the drift error.



Motion-based Localisation (Dead Reckoning)

The uncertainty ellipsoid is no longer perpendicular to the motion direction as soon as the robot starts to turn.



Worked Example

Dead Reckoning Localisation

Worked Example

GIVEN

$$\boldsymbol{\mu}_0 = \begin{bmatrix} s_{x,0} \\ s_{y,0} \\ s_{\theta,0} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$ robot initial position

$$\boldsymbol{\Sigma}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$ initial covariance matrix

Assume the following conditions remain constant $\forall k$

$$\boldsymbol{Q_k} = \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix}$$ motion model covariance matrix

$V_k = 1m/s$ mobile robot linear velocity

$\omega_k = 1rad/s$ mobile robot angular velocity

$\Delta t = 0.1s$ sampling time

FIND

Using dead reckoning, estimate the position of the robot for two time steps, i.e., μ_1, μ_2 and Σ_1, Σ_2.

Worked Example

First Iteration

Step 1: Calculate the estimated position of the robot

Worked Example

First Iteration

Step 1: Calculate the estimated position of the robot

$$\boldsymbol{\mu}_1 = \mathbf{h}(\boldsymbol{\mu}_0, \mathbf{u}_1)$$

$$\boldsymbol{\mu}_1 = \begin{bmatrix} s_{x,1} \\ s_{y,1} \\ s_{\theta,1} \end{bmatrix} = \begin{bmatrix} s_{x,0} + \Delta t \cdot V_1 \cdot \cos\left(s_{\theta,0}\right) \\ s_{y,0} + \Delta t \cdot V_1 \cdot \sin\left(s_{\theta,0}\right) \\ s_{\theta,0} + \Delta t \cdot \omega_1 \end{bmatrix} = \begin{bmatrix} 0 + 0.1 \cdot 1 \cdot \cos(0) \\ 0 + 0.1 \cdot 1 \cdot \sin(0) \\ 0 + 0.1 \cdot 1 \end{bmatrix} = \begin{bmatrix} 0.1 \\ 0 \\ 0.1 \end{bmatrix}$$

Worked Example

First Iteration

Step 2: Calculate the linearised model to be used in the uncertainty propagation

Worked Example

First Iteration

Step 2: Calculate the linearised model to be used in the uncertainty propagation

$$\boldsymbol{H}_1 = \begin{bmatrix} 1 & 0 & -\Delta t \cdot V_1 \cdot \sin\left(s_{\theta,0}\right) \\ 0 & 1 & \Delta t \cdot V_1 \cdot \cos\left(s_{\theta,0}\right) \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.1 \\ 0 & 0 & 1 \end{bmatrix}$$

Worked Example

First Iteration

Step 3: Calculate the propagation of the uncertainty using Affine Transformation Result

Worked Example

First Iteration

Step 3: Calculate the propagation of the uncertainty using Affine Transformation Result

$$\mathbf{\Sigma}_1 = \mathbf{H}_1 \cdot \mathbf{\Sigma}_0 \cdot \mathbf{H}_1^T + \mathbf{Q}_1$$

$$\mathbf{\Sigma}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.1 & 1 \end{bmatrix} + \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix}$$

$$\mathbf{\Sigma}_1 = \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix}$$

Worked Example

Second Iteration

Step 1: Calculate the estimated position of the robot

Worked Example

Second Iteration

Step 1: Calculate the estimated position of the robot

$$\boldsymbol{\mu}_2 = \mathbf{h}(\boldsymbol{\mu}_1, \mathbf{u}_1)$$

$$\boldsymbol{\mu}_2 = \begin{bmatrix} s_{x,2} \\ s_{y,2} \\ s_{\theta,2} \end{bmatrix} = \begin{bmatrix} s_{x,1} + \Delta t \cdot V_2 \cdot \cos(s_{\theta,1}) \\ s_{y,1} + \Delta t \cdot V_2 \cdot \sin(s_{\theta,1}) \\ s_{\theta,1} + \Delta t \cdot \omega_2 \end{bmatrix} = \begin{bmatrix} 0.1 + 0.1 \cdot 1 \cdot \cos(0.1) \\ 0 + 0.1 \cdot 1 \cdot \sin(0.1) \\ 0.1 + 0.1 \cdot 1 \end{bmatrix} = \begin{bmatrix} 0.1995 \\ 0.01 \\ 0.2 \end{bmatrix}$$

Worked Example

Second Iteration

Step 2: Calculate the linearised model to be used in the uncertainty propagation

Worked Example

Second Iteration

Step 2: Calculate the linearized model to be used in the uncertainty propagation

$$\boldsymbol{H}_2 = \begin{bmatrix} 1 & 0 & -\Delta t \cdot V_2 \cdot \sin\left(s_{\theta,1}\right) \\ 0 & 1 & \Delta t \cdot V_2 \cdot \cos\left(s_{\theta,1}\right) \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -0.1 \cdot 1 \cdot \sin(0.1) \\ 0 & 1 & 0.1 \cdot 1 \cdot \cos(0.1) \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -0.01 \\ 0 & 1 & 0.0995 \\ 0 & 0 & 1 \end{bmatrix}$$

Worked Example

Second Iteration

Step 3: Calculate the propagation of the uncertainty using Affine Transformation Result

Worked Example

Second Iteration

Step 3: Calculate the propagation of the uncertainty using Affine Transformation Result

$$\boldsymbol{\Sigma}_2 = \mathbf{H}_2 \cdot \boldsymbol{\Sigma}_1 \cdot \mathbf{H}_2^{\boldsymbol{T}} + \mathbf{Q}_2$$

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} 1 & 0 & -0.01 \\ 0 & 1 & 0.0995 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -0.01 & 0.0995 & 1 \end{bmatrix} + \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix}$$

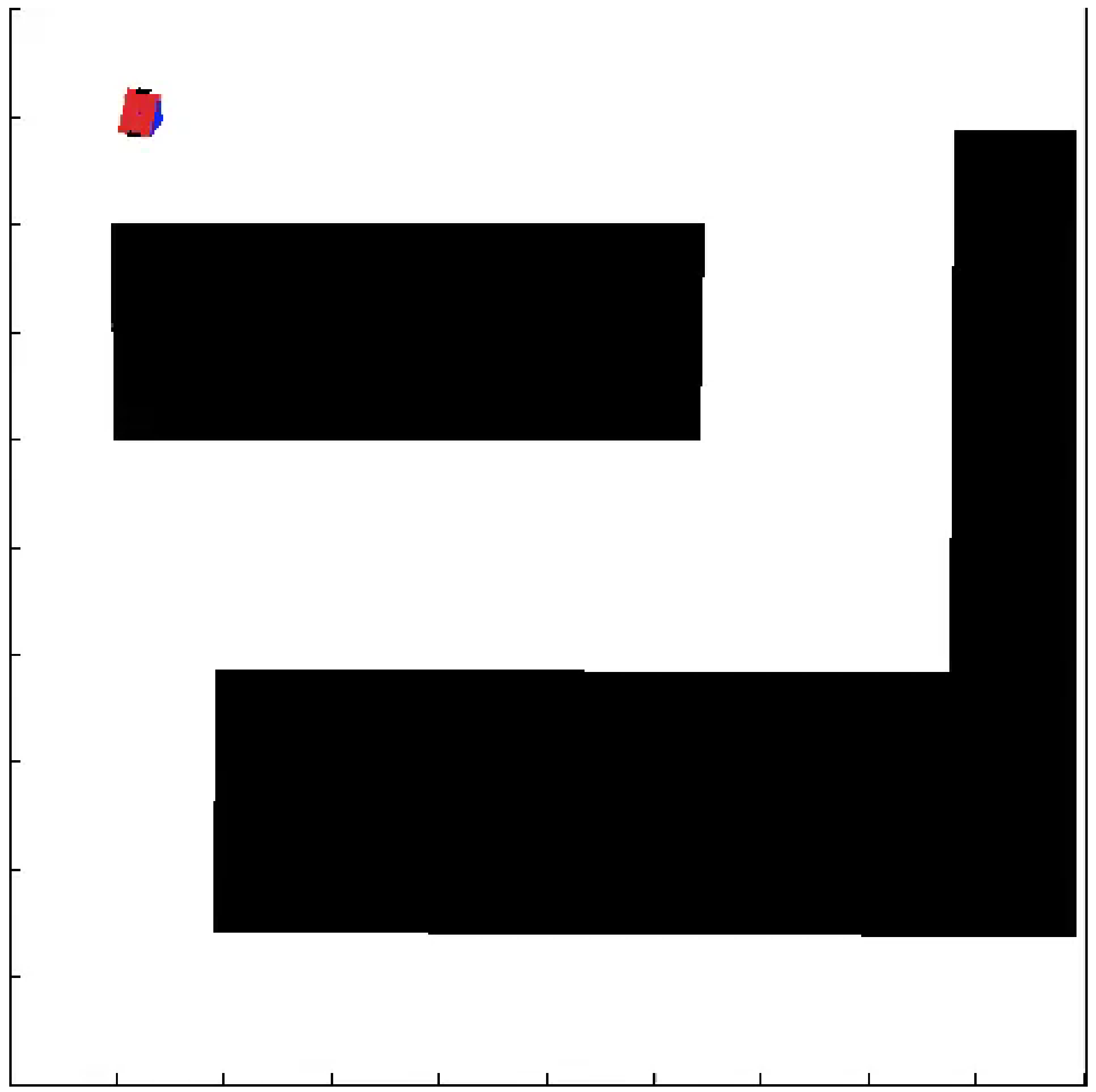
$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} 0.998 & 0.0207 & 0.0180 \\ 0.0207 & 1.0040 & 0.0399 \\ 0.0180 & 0.0399 & 0.4000 \end{bmatrix}$$

Worked Example

The uncertainty in robot localisation increases over time: $\mathbf{\Sigma_2} > \mathbf{\Sigma_1}$

$$\mathbf{\Sigma_1} = \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix} \qquad \mathbf{\Sigma_2} = \begin{bmatrix} 0.998 & 0.0207 & 0.0180 \\ 0.0207 & 1.0040 & 0.0399 \\ 0.0180 & 0.0399 & 0.4000 \end{bmatrix}$$

Dead Reckoning Localisation



Pose covariance matrix $\mathbf{Q}_k$ -preliminaries-

How can we obtain, experimentally, the covariance matrix $\mathbf{Q}_k$???

We do many experiments with the mobile robot.

Example:

- Run PuzzleBot many times in open loop, i.e., for the same number of seconds and at the same speed. For instance, for 4 seconds at half of the maximum speed.
- Save the end positions for each experiment
- Save the measured angular velocities of the wheels ω_r and ω_l
- Compute the ideal values for ω_r and ω_l
- Linearise the Kinematic model considering as variables ω_r and ω_l
- Compute the covariance $\mathbf{Q}_k$ by tunning k_r and k_l

Pose covariance matrix $\mathbf{Q}_k$ -preliminaries-

Consider the following motion model for a differential drive robot:

$$\boldsymbol{h}(\boldsymbol{s}_k, \omega_{r,k}, \omega_{l,k}) = \begin{bmatrix} s_{x,k-1} + r\Delta t \dfrac{\omega_{r,k} + \omega_{l,k}}{2} \cos(s_{\theta,k-1}) \\ s_{y,k-1} + r\Delta t \dfrac{\omega_{r,k} + \omega_{l,k}}{2} \sin(s_{\theta,k-1}) \\ s_{\theta,k-1} + r\Delta t \dfrac{\omega_{r,k} - \omega_{l,k}}{l} \end{bmatrix}$$

where $\omega_{r,k}$ and $\omega_{l,k}$ are the right and left wheel angular velocity at time step k.

Pose covariance matrix $\mathbf{Q}_k$ -preliminaries-

- Now assume the noise in both right and left wheel angular velocities to be zero-mean Gaussian distribution such that:

$$\begin{bmatrix} \omega_{r,k} \\ \omega_{l,k} \end{bmatrix} \sim \mathcal{N}\left(0, \Sigma_{\Delta,k}\right) \qquad \Sigma_{\Delta,k} = \begin{bmatrix} k_r\left|\omega_{r,k}\right| & 0 \\ 0 & k_l\left|\omega_{l,k}\right| \end{bmatrix}$$

where k_r and k_l are constants representing the error associated with computing the angular velocity by each wheel.

- These constants are related to the traction between the wheels and the floor surface or the encoder noise used to compute the wheel displacements.
- Larger angular speed of the right motor $\left|\omega_{r,k}\right|$ will lead to a larger variance of that motor $k_r\left|\omega_{r,k}\right|$.

Pose covariance matrix $\mathbf{Q}_k$ -preliminaries-

It is possible to propagate this noise $\Sigma_{\Delta,k}$ to be seen from the robot state prospective using Taylor series expansion as follows:

$$Q_k = \nabla_{\omega_k} \cdot \Sigma_{\Delta,k} \cdot \nabla_{\omega_k}^T$$

$$\nabla_{\omega_k} = \begin{bmatrix} \frac{\partial h_1}{\partial \omega_{r,k}} & \frac{\partial h_1}{\partial \omega_{l,k}} \\ \frac{\partial h_2}{\partial \omega_{r,k}} & \frac{\partial h_2}{\partial \omega_{l,k}} \\ \frac{\partial h_3}{\partial \omega_{r,k}} & \frac{\partial h_3}{\partial \omega_{l,k}} \end{bmatrix} = \frac{1}{2} r \Delta t \begin{bmatrix} \cos(s_{\theta,k-1}) & \cos(s_{\theta,k-1}) \\ \sin(s_{\theta,k-1}) & \sin(s_{\theta,k-1}) \\ \frac{2}{l} & -\frac{2}{l} \end{bmatrix}$$