Durham University

Robotics – Planning and Motion

COMP52815

Dr Junyan Hu & Prof Farshad Arvin

Email: junyan.hu@durham.ac.uk
Room: MCS 2060

Durham University

COMP52815 Robotics - Planning and Motion

Lecture: Learning Objectives

The aim of this lecture is to learn the dynamics, control, and localisation of wheeled mobile robots

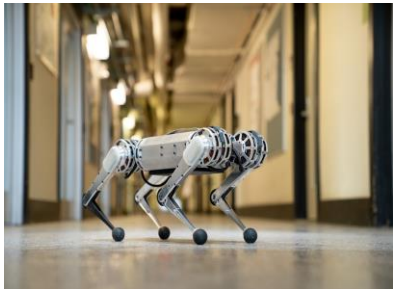
- Objectives:
  1. Differential drive robots
  2. Localisation
  3. Motion control

See also:

- Introduction to Mobile Robot Control, Spyros G. Tzafestas, 2014

Mobile robots

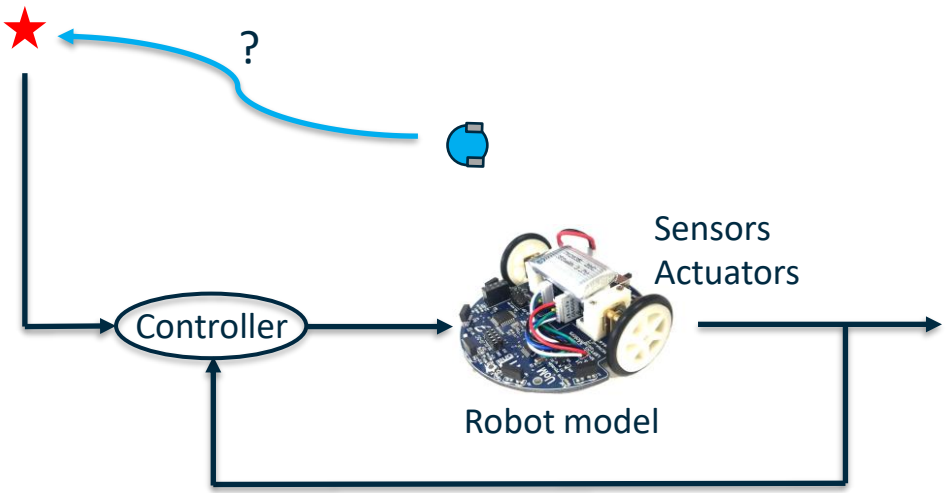
- *A robot is an artificial physical agent that perceives its environment through sensors and acts upon that environment through actuators*



Driving robot around?

- What does it take to drive a robot from point A to B?

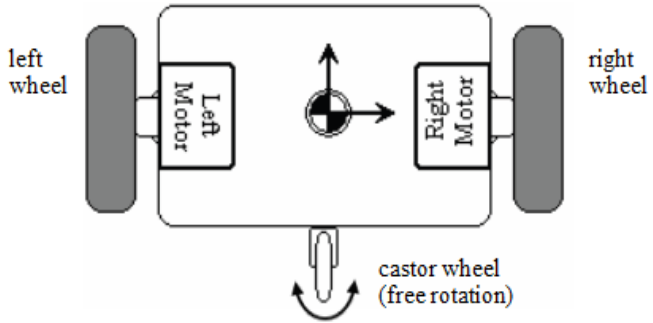


Divide and Conquer

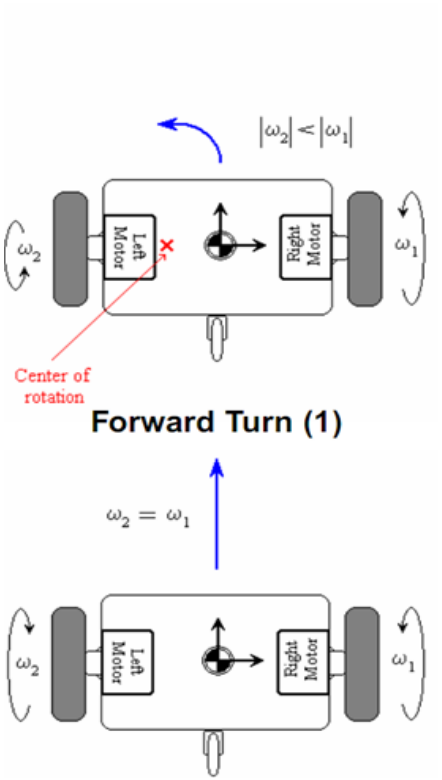
- The world is dynamic and fundamentally unknown
- The controller must be able to respond to environmental conditions
- Instead of building one complicated controller – divide and conquer: Behaviors
  - Go-to-goal
  - Avoid-obstacles
  - Follow-path
  - Track-target
  - …

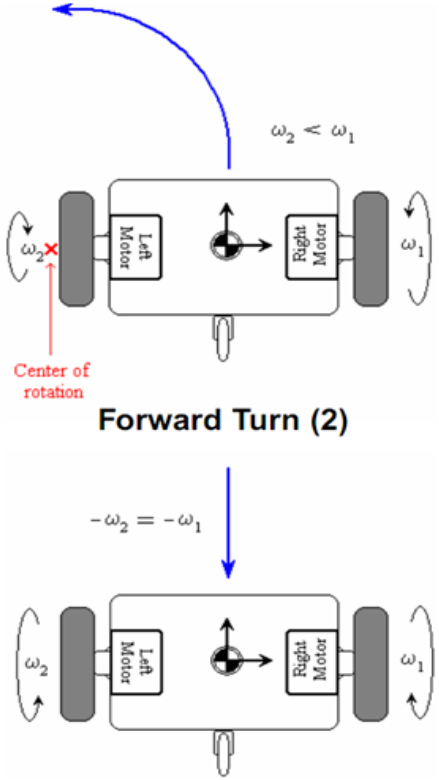
Differential drive robots

- Also known as differential wheeled robots, these are mobile robots whose movement is based on two separately driven wheels placed on either side of the robot body. It can thus change its direction by varying the relative rate of rotation of its wheels, thereby requiring no additional steering motion.



Differential drive kinematics



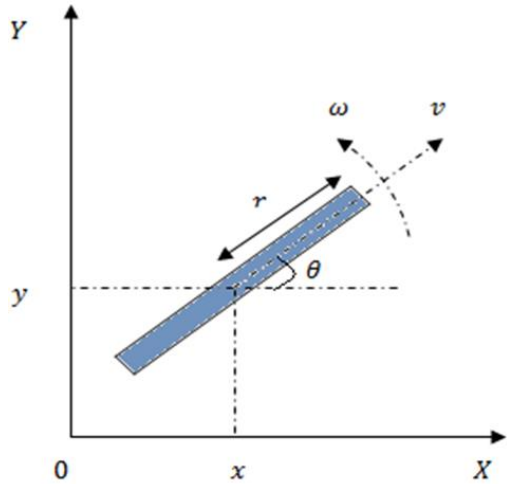


Durham University

Kinematics of a unicycle

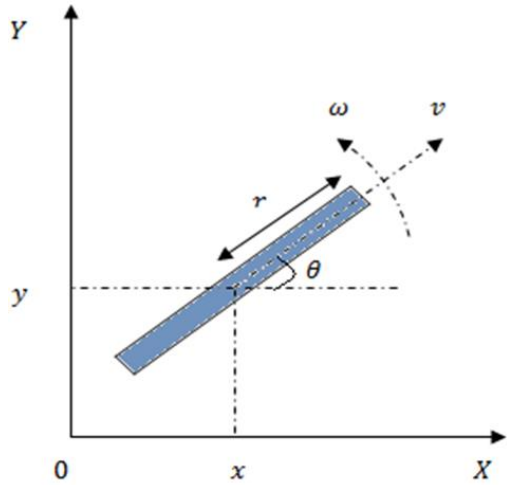
Ignoring balancing concerns, there are two action variables, i.e., direct inputs to the system in the XY plane.

- The first one is the forward/linear velocity: $v=\omega_u r$, where ω_u is the wheel angular velocity, r is wheel radius
- The second one is the steering velocity denoted by ω.



Durham University

Kinematics of a unicycle



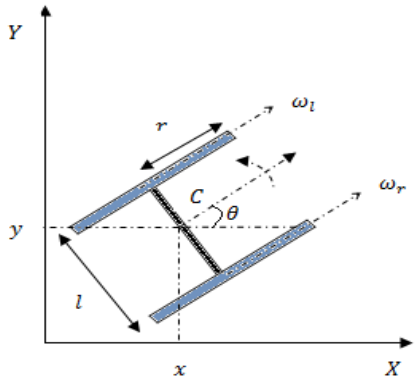
Dynamics:

$$\dot{x} = v \cdot \cos\theta$$
$$\dot{y} = v \cdot \sin\theta$$
$$\dot{\theta} = \omega$$

The nonholonomic constraint:

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0$$

Kinematics of a differential drive



- The resultant forward velocity through C (the centre of mass) is $v = r(\frac{\omega_r + \omega_l}{2})$.
- The steering velocity is $\omega = r(\frac{\omega_r - \omega_l}{l})$.

Kinematics of a differential drive

- Thus, just like the unicycle, the configuration transition equations may be given as

$$\dot{x} = r\frac{\omega_r + \omega_l}{2}\cos\theta$$
$$\dot{y} = r\frac{\omega_r + \omega_l}{2}\sin\theta$$
$$\dot{\theta} = r\frac{\omega_r - \omega_l}{l}$$

- Comparing the equations for unicycle and for differential drive yields the transformation

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

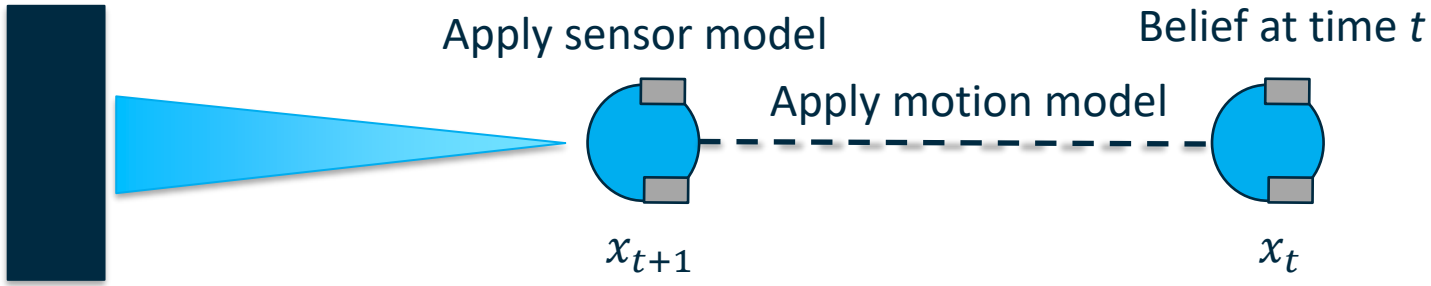
Localisation

- The robot needs to know its location in the environment in order to make proper decisions.

- Localisation can be achieved using:
  - Proprioceptive sensors (encoders, IMU). This type of localisation is named **dead reckoning localisation**.
  - Exteroceptive sensors (sonar, LiDAR, camera). This type of localisation is named **map-based localisation**.
  - External sensors (GPS). Not suitable for indoor applications.

Probabilistic Localisation

- In robotics, we deal with localization probabilistically
- Three key components:
  - A robot's belief of where it is (its state)
  - A robot's motion model
  - A robot's sensor model



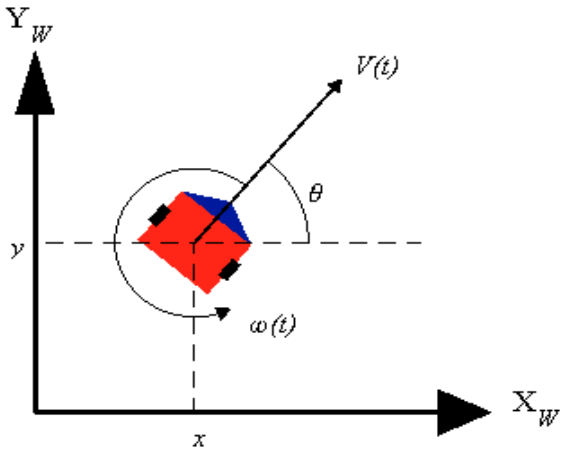
Motion-based Localisation (Dead Reckoning)

This technique uses the internal kinematics of the robot to localise it in the environment.

This method is simple to implement, and does not require sophisticated sensors.

However, such technique suffers from the unbounded growth of uncertainty about the robot pose over time due to the numerical integration and accumulation of error.

Motion-based Localisation (Dead Reckoning)



Kinematic model for a differential robot model

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

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

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

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:

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

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

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

This approximation follows the **Markov assumption** where the current robot pose depends only on the previous pose and the input velocities.

$$\begin{bmatrix}s_{x,k} \\ s_{y,k} \\ s_{\theta,k}\end{bmatrix} = \begin{bmatrix}s_{x,k-1} \\ s_{y,k-1} \\ s_{\theta,k-1}\end{bmatrix} + \begin{bmatrix}\Delta d \cos\left(s_{\theta,k-1}\right) \\ \Delta d \sin\left(s_{\theta,k-1}\right) \\ \Delta\theta\end{bmatrix}$$

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.

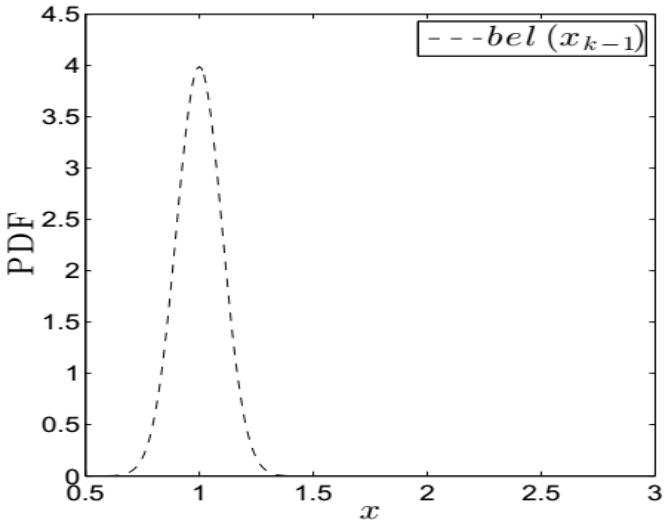
The Gaussian distribution (or normal distribution) is denoted by

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

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.

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:

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

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

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 with Markov assumption 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.

Motion-based Localisation (Dead Reckoning)

The function $\mathbf{h}(\mathbf{s}_{k-1}, \mathbf{u}_k)$ is generally nonlinear, and in the case of a differential-drive robot , this function is defined as:

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

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:

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

Motion-based Localisation (Dead Reckoning)

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 $\mathbf{s}_{k-1} = \boldsymbol{\mu}_{k-1}$:

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

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

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

Motion-based Localisation (Dead Reckoning)

In the case of a differential-drive robot, the Jacobian $\mathbf{H}_k$ is computed as follows:

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

$$\mathbf{H}_k = \begin{bmatrix} \frac{\partial h_1}{\partial s_{x,k}} & \frac{\partial h_1}{\partial s_{y,k}} & \frac{\partial h_1}{\partial s_{\theta,k}} \\ \frac{\partial h_2}{\partial s_{x,k}} & \frac{\partial h_2}{\partial s_{y,k}} & \frac{\partial h_2}{\partial s_{\theta,k}} \\ \frac{\partial h_3}{\partial s_{x,k}} & \frac{\partial h_3}{\partial s_{y,k}} & \frac{\partial h_3}{\partial s_{\theta,k}} \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & -\Delta t \cdot v_k \cdot \sin(\mu_{\theta,k-1}) \\ 0 & 1 & \Delta t \cdot v_k \cdot \cos(\mu_{\theta,k-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

Motion-based Localisation (Dead Reckoning)

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

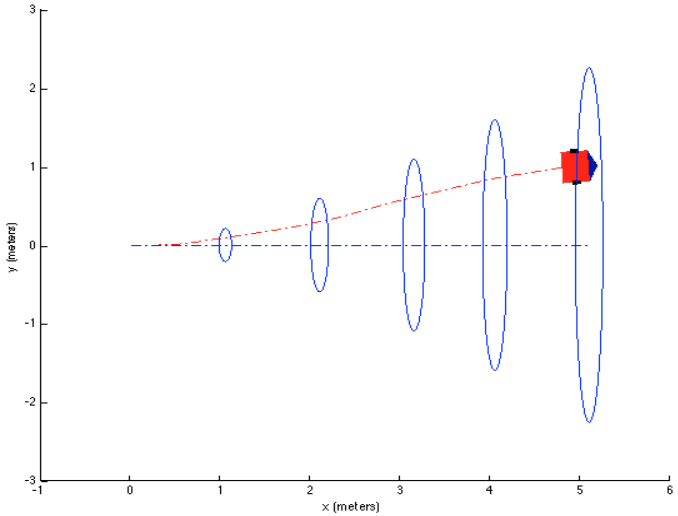
$$\boldsymbol{\Sigma}_k = \mathbf{H}_k \boldsymbol{\Sigma}_{k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{Q}_k$$

Thus, the estimated pose at time step k is Gaussian such that $\mathbf{s}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, 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 such that $\boldsymbol{\mu}_0 = \mathbf{0}$, and $\boldsymbol{\Sigma}_0 = \mathbf{0}$.

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.

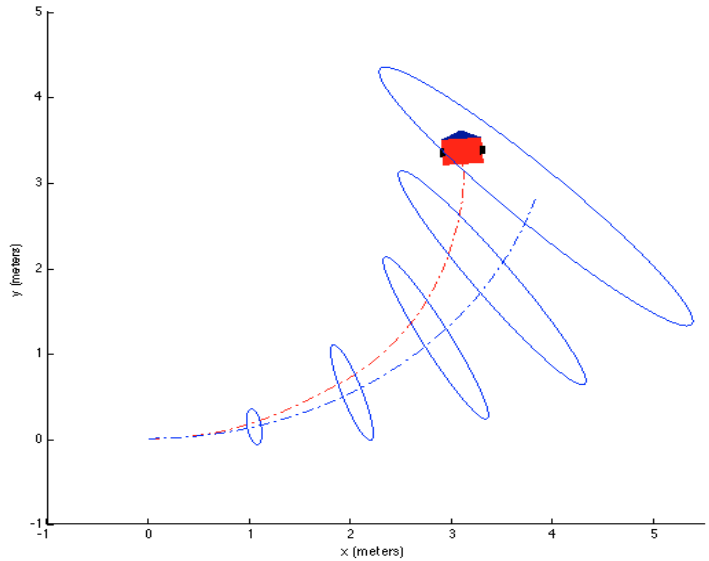
Motion-based Localisation (Dead Reckoning)

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.



Durham University

Pose covariance matrix

- 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

- 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 $k_r\left|\omega_{r,k}\right|$.

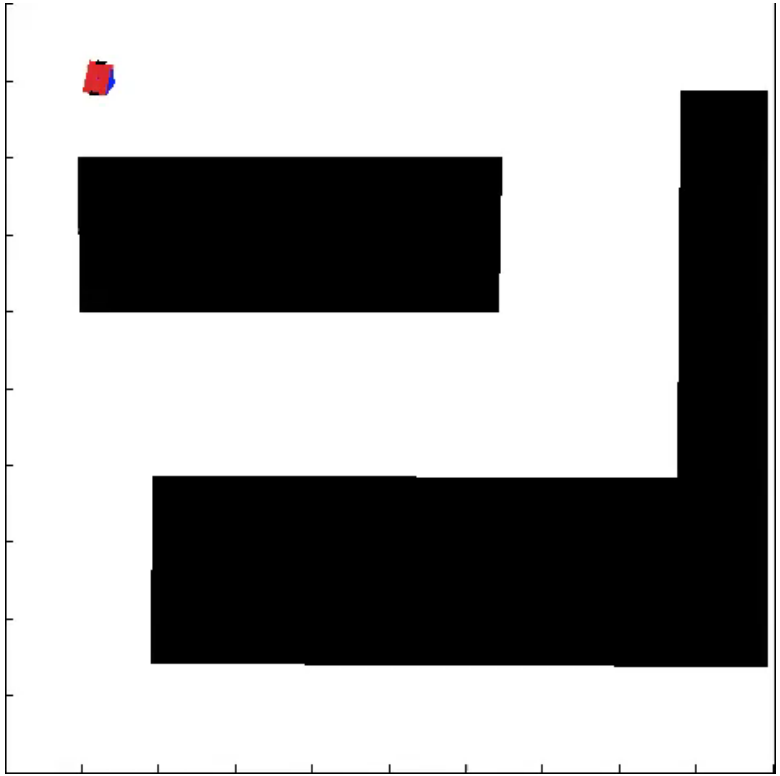
Pose Covariance Matrix

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

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

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

Example



Conclusion

With this type of localisation, where the robot depends solely on its proprioceptive sensors, the pose uncertainty will grow unbounded over time as the robot moves. This growth will soon make the estimated pose invalid for the robot to make proper navigational decisions, such as performing optimal path-planning.

One way to overcome this issue is to use a map; this approach is called map-based localisation, which will not be covered in this course.

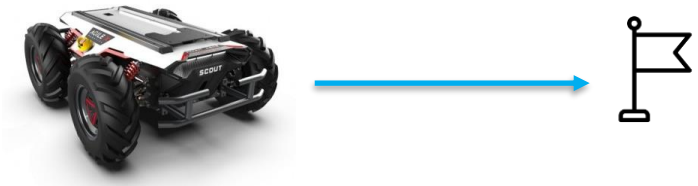
Motion Control

- The **motion control** for a mobile robot deals with the task of finding the control inputs that need to be applied to the robot such that a predefined goal can be reached in a finite amount of time.

- Control of differential drive robots has been studied from several points of view, but essentially falls into one of the following three categories: **point-to-point navigation (or point stabilisation), trajectory tracking, and path following**.

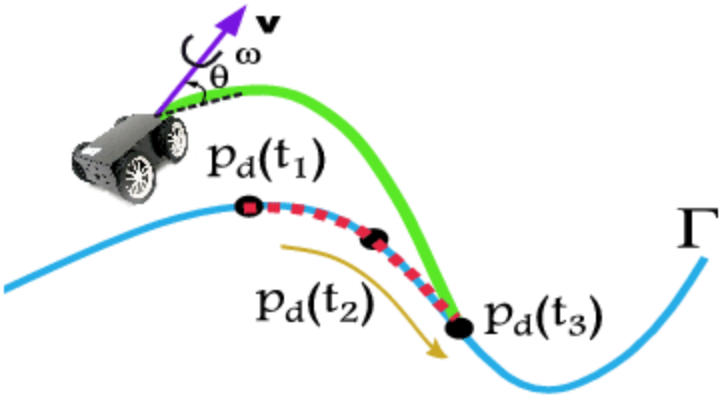
Motion Control
-Point Stabilisation-

- The objective here is to drive the robot to a desired fixed state, say a fixed position and orientation. Point stabilisation presents a true challenge to control system when the vehicle has nonholonomic constraints, since that goal cannot be achieved with smooth time-invariant state-feedback control laws. This control technique will be used in this course.



Motion Control
-Trajectory Tracking-

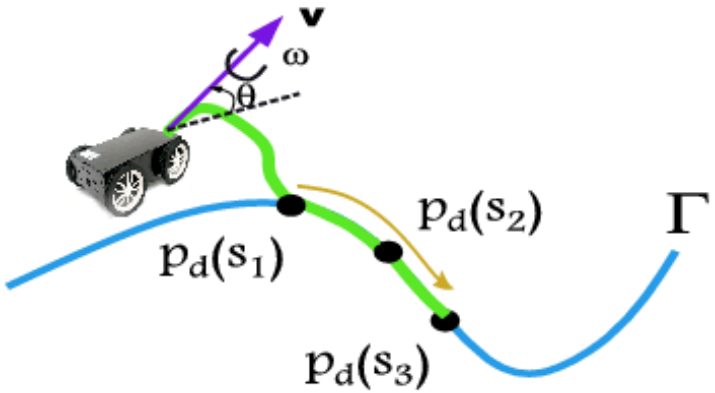
- The objective is driving the robot into following a time-parameterised state trajectory. In fact, the trajectory tracking problem for fully actuated systems is well understood and satisfactory solutions can be found in advanced nonlinear control textbooks. However, in case of underactuated systems, the problem is still a very active area of research.



Time constraints

Motion Control
-Path Following-

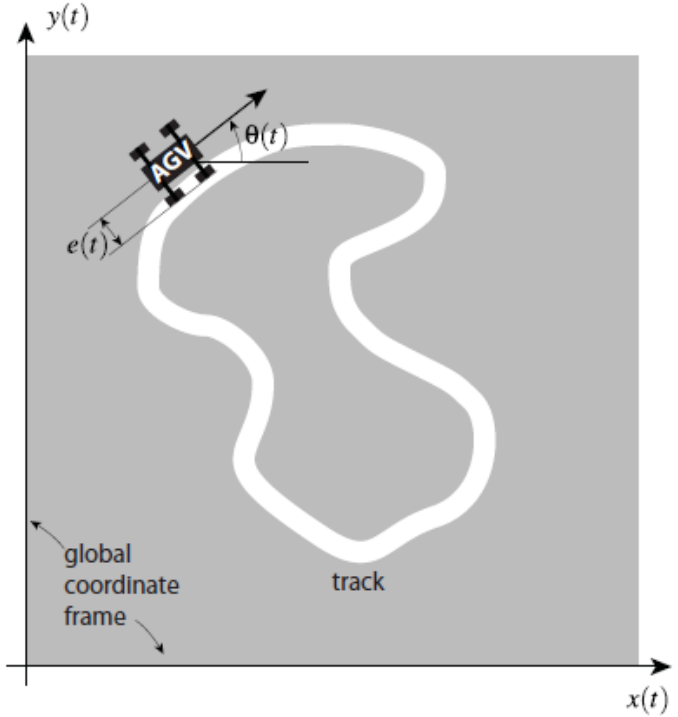
- In this case the vehicle is required to converge to and follow a path, without any temporal specifications. The underlying assumption in path following control is that the vehicle's forward speed tracks a desired speed profile, while the controller acts on the vehicle orientation to drive it to the path. Typically, smoother convergence to a path is achieved and the control signals are less likely pushed to saturation.



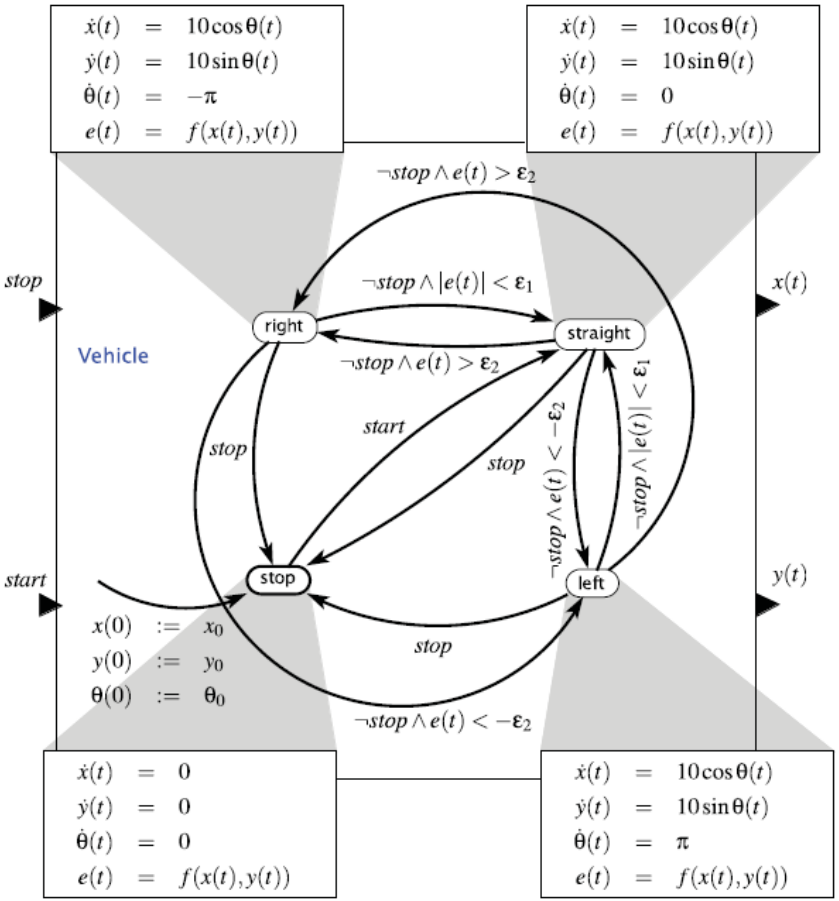
Without time constraints

Example

- Consider an automated guided vehicle (AGV) that moves along a closed track painted on a warehouse or factory floor. We will design a controller so that the vehicle closely follows the track.



Example



Dynamics: $\dot{x} = v \cdot cos\theta$

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

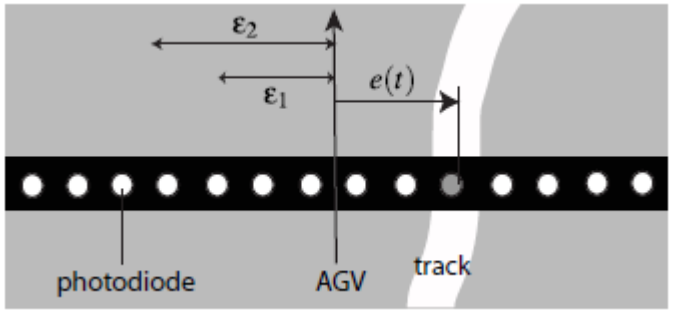
$\dot{\theta} = \omega$

We ignore the inertia of the vehicle, so we assume that we can instantaneously change the velocity or angular speed.

Durham University

Example

We design the supervisory control governing transitions between modes in such a way that the vehicle closely follows the track, using a sensor that determines how far the vehicle is to the left or right of the track.

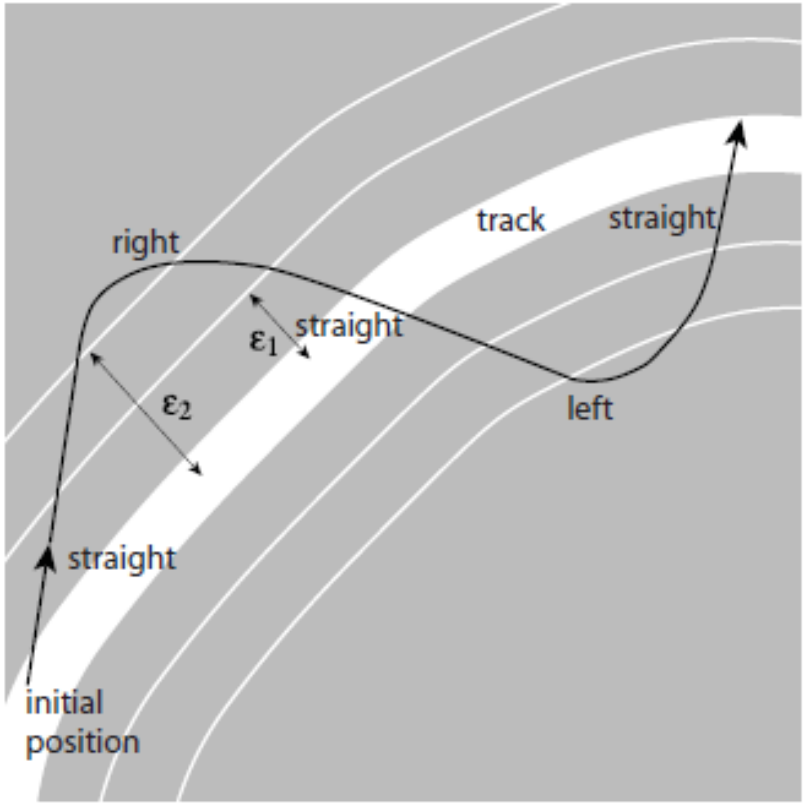


If $|e(t)| < \varepsilon_1$, move straight ahead

If $|e(t)| > \varepsilon_2$, turn right

If $|e(t)| < -\varepsilon_2$, turn left

Example


track
straight
right
straight
ε_1
ε_2
left
straight
initial
position

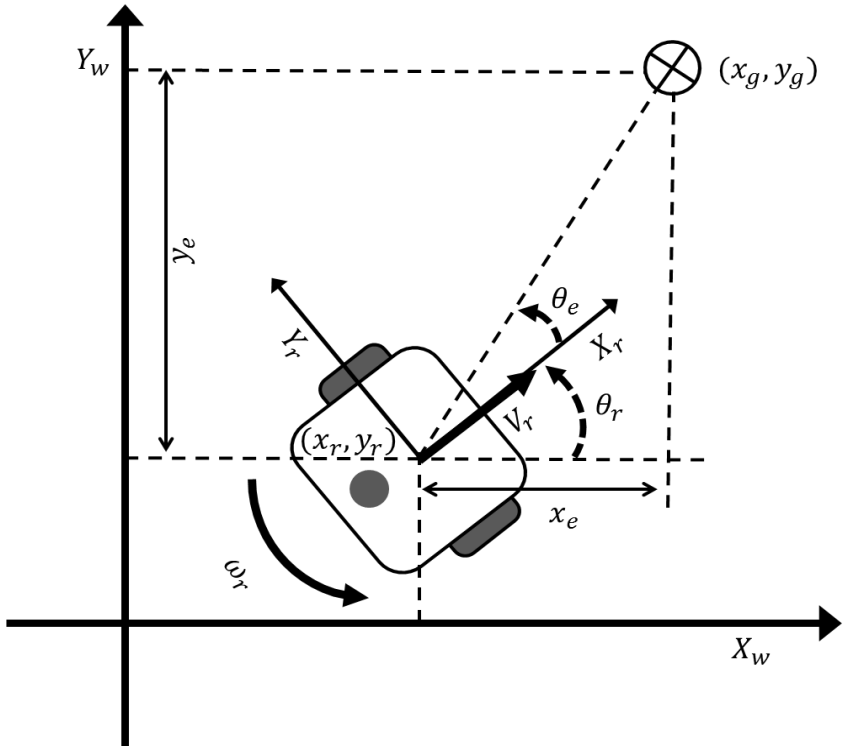
Point Stabilisation

The goal can be defined in the simplest way as a set of coordinates in a multidimensional space. For instance, if the robot is moving in a two dimensional space the goal is:

$\boldsymbol{X}_g = \begin{pmatrix} x_g \\ y_g \end{pmatrix}$ if the position of the robot needs to be controlled, or

$\boldsymbol{X}_g = \begin{pmatrix} x_g \\ y_g \\ \theta_g \end{pmatrix}$ if both position and heading need to be controlled.

Point Stabilisation



- Dynamics:

$$\begin{cases} \dot{x}_r = v_r \cdot cos\theta_r \\ \dot{y}_r = v_r \cdot sin\theta_r \\ \dot{\theta}_r = \omega_r \end{cases}$$

Point Stabilisation

In order to design the controller we first need to define the robot inputs, robot pose and the goal. For a non-holonomic robot moving in a 2D environment the robot pose can be represented by the vector

$$\boldsymbol{\rho}_r = \begin{pmatrix} x_r \\ y_r \\ \theta_r \end{pmatrix}.$$

The inputs are the linear and angular velocity of the robot

$$\boldsymbol{v}_r = \begin{pmatrix} v_r \\ \omega_r \end{pmatrix}.$$

Point Stabilisation

For the sake of simplicity, in this course, the goal is defined only by a 2D set of coordinates $\boldsymbol{X}_g = \begin{pmatrix} x_g \\ y_g \end{pmatrix}$.

Then compute the errors by using the goal coordinates and robot position as in the following:

$$e_x = x_g - x_r$$

$$e_y = y_g - y_r$$

$$e_\theta = atan2(e_y, e_x) - \theta_r$$

Point Stabilisation

The robot position is assumed to be known and can be computed using various localisation techniques as Dead Reckoning. The equations for the error can be represented in vector format as follows:

$$\boldsymbol{e} = \begin{pmatrix} e_x \\ e_y \\ e_\theta \end{pmatrix}$$

The general form of the control law can be written as:

$$\begin{pmatrix} v_r \\ \omega_r \end{pmatrix} = \boldsymbol{K} \begin{pmatrix} e_x \\ e_y \\ e_\theta \end{pmatrix}$$

where

$$\boldsymbol{K} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{pmatrix}$$, is the control matrix

Point Stabilisation

For simplicity the six controller gain parameters can be reduced to only two by defining the distance error:

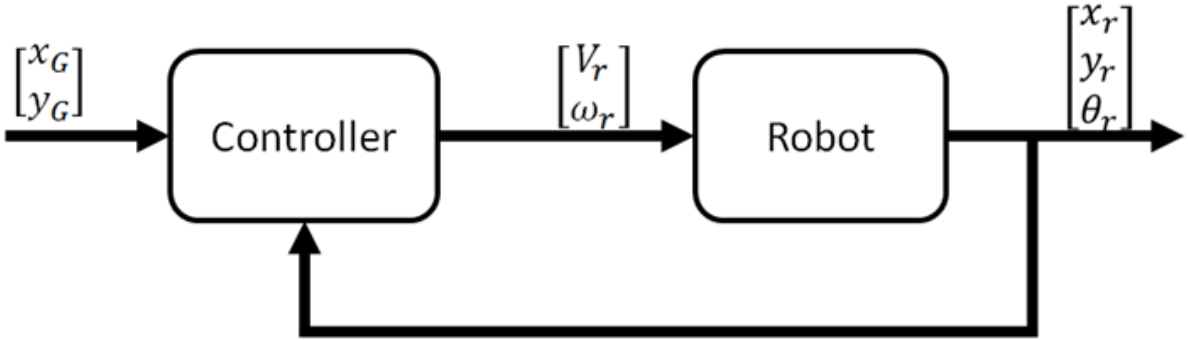
$$e_d = \sqrt{e_x^2 + e_y^2}$$

The control law can now be written as:

$$v_r = K_d e_d$$
$$\omega_r = K_\theta e_\theta$$

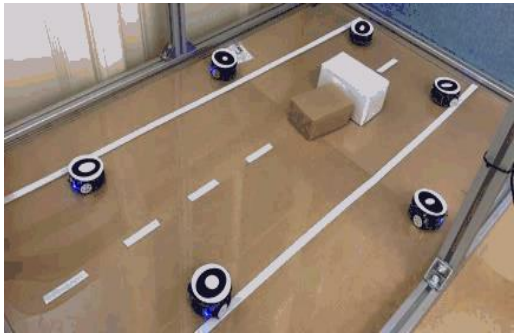
Point Stabilisation

Closed-loop control block diagram:



Motion control of multiple mobile robots

Multi-target enclosing



Object transportation

Collision avoidance

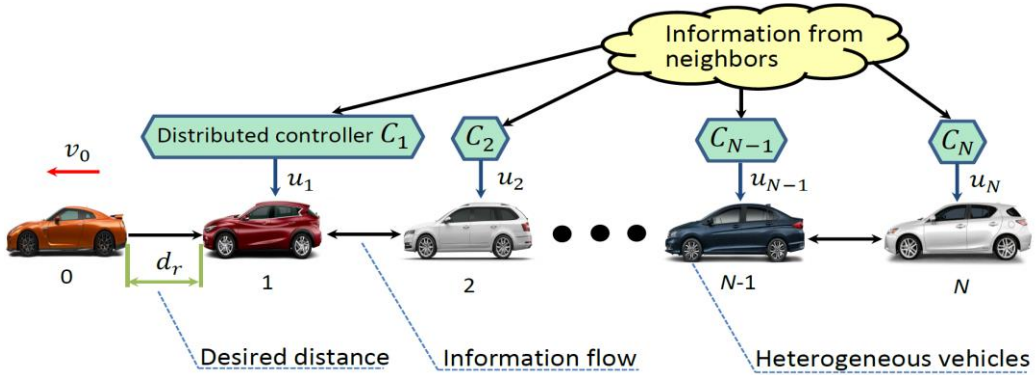
Dynamic formation

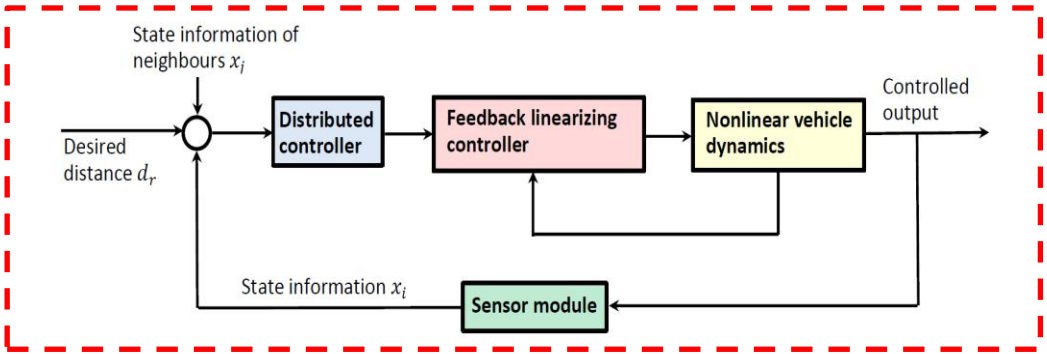
Durham University

Extension: Control of Autonomous Vehicles

Durham University

Control Scheme

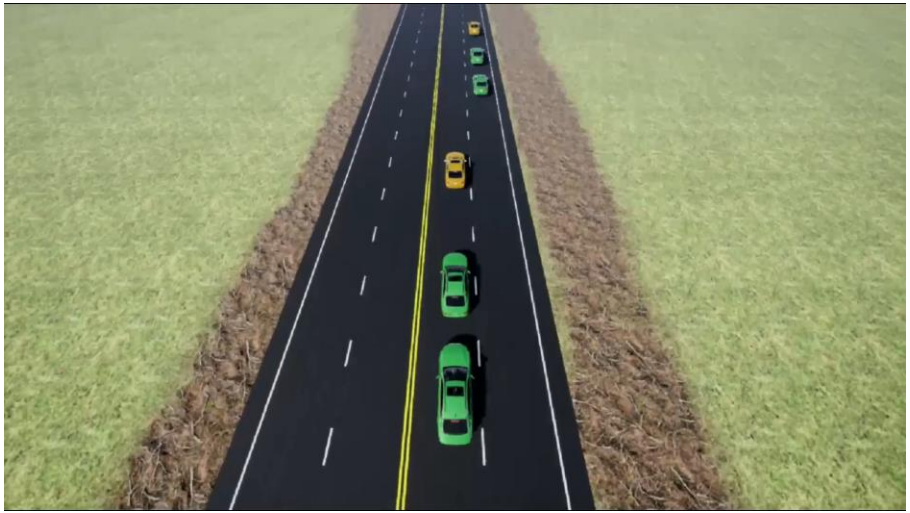




Durham University

J. Hu, P. Bhowmick, F. Arvin, A. Lanzon and B. Lennox, “Cooperative control of heterogeneous connected vehicle platoons: An adaptive leader-following approach,” ***IEEE Robotics and Automation Letters***, 2020.

Simulation Results



Vehicle platoon merging

Vehicle platoon separating

Durham University

J. Hu, P. Bhowmick, F. Arvin, A. Lanzon and B. Lennox, “Cooperative control of heterogeneous connected vehicle platoons: An adaptive leader-following approach,” ***IEEE Robotics and Automation Letters***, 2020.

Lecture Summary

- Differential drive robots
- Localisation
- Motion control of unicycles