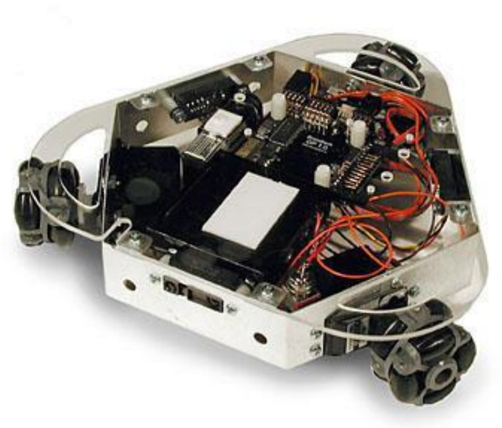
{Learn, Create, Innovate};

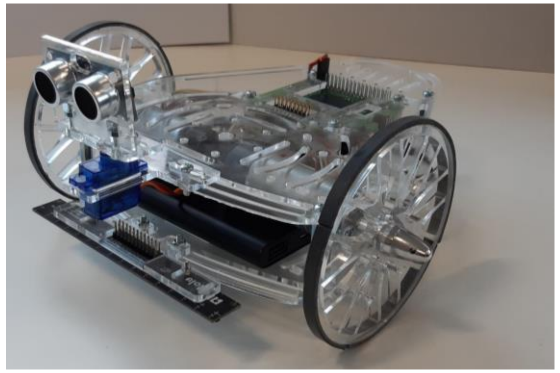
Autonomous Systems

Dr Alexandru Stancu

Mobile Robots

- There exists many types of wheeled robotic platforms
  - Differential-Drive robots
  - Omnidirectional robots
  - Ackermann-steering robots
  - and many others…
- In this course we will focus on differential drive robots, also known as “differential wheeled robots”.


Holonomic Robot
Acroname ©.

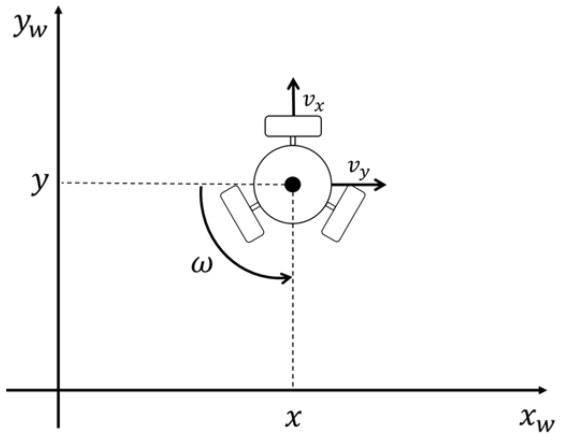

Differential-drive
Puzzlebot ©.

Mobile Robots

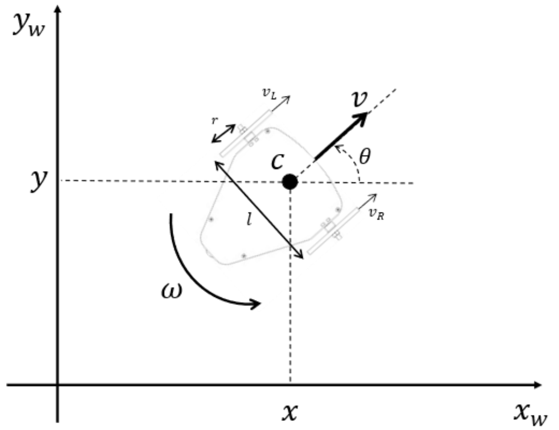
Mobile robots can be classified as "Holonomic" or "Nonholonomic“.

- This classification depends on the relationship between controllable and total degrees of freedom of a robot.

- **Holonomic Robots**: If the controllable degree of freedom is equal to total degrees of freedom, then the robot
- **Nonholonomic Robots**: If the controllable degree of freedom is less than the total degrees of freedom. Such systems are therefore called underactuated. Differential Drive Systems fall into this category.



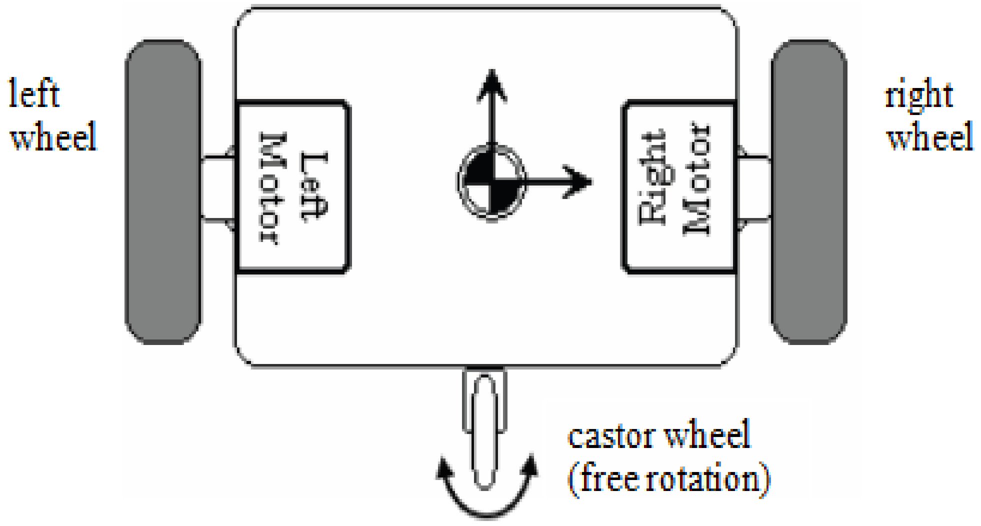
Holonomic Robot



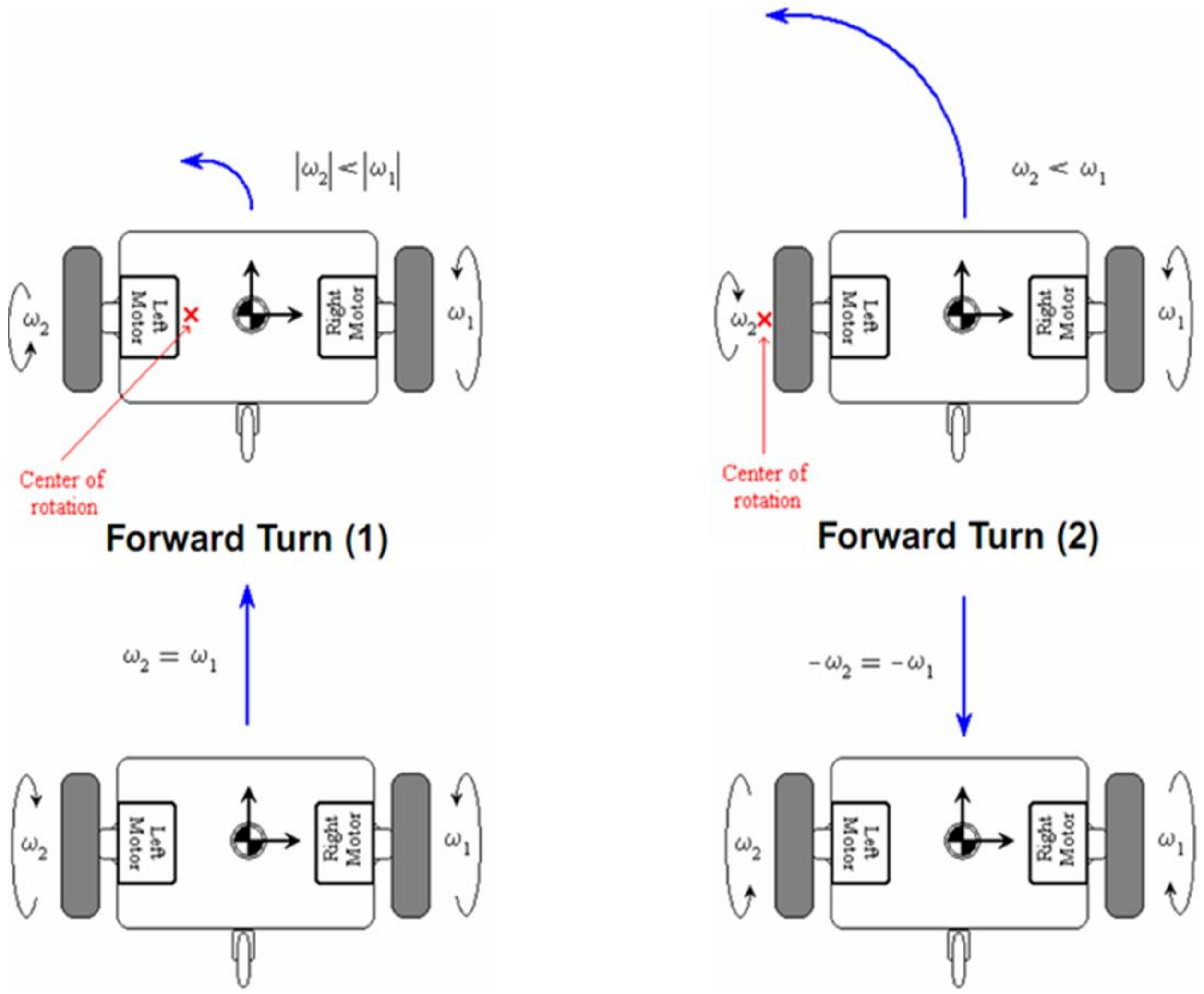
Nonholonomic Robot

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.

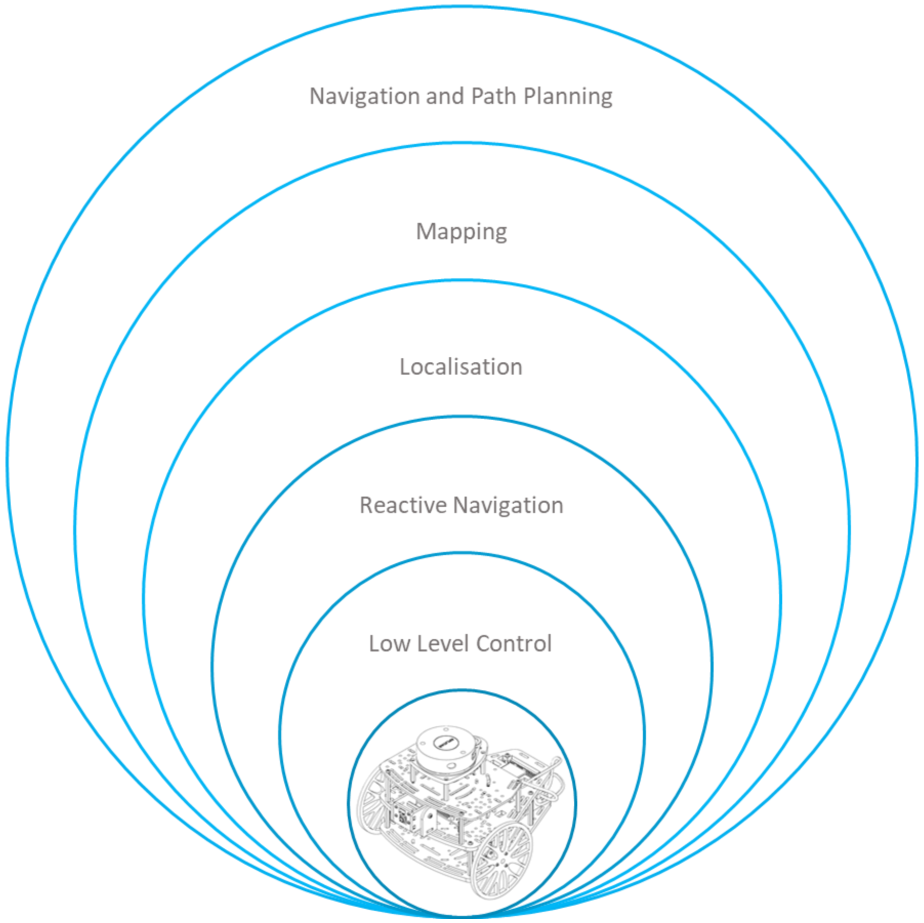


Differential drive kinematics



The hierarchy of the autonomy

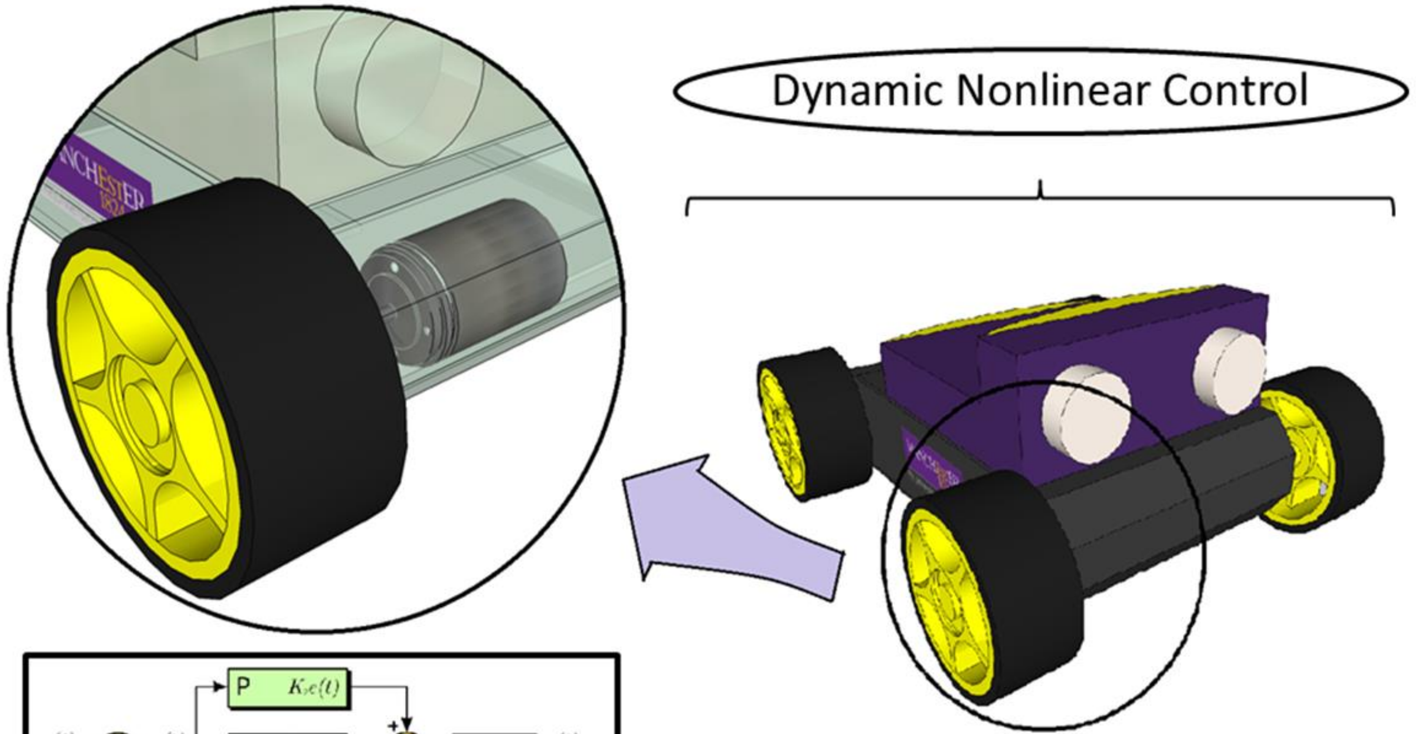
How much information and support must be provided by human to ensure that the robot is able to achieve its goals.

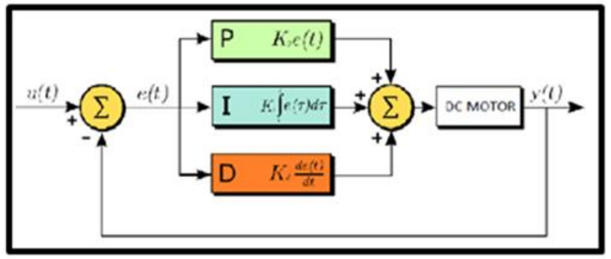


This lecture (First level of autonomy)

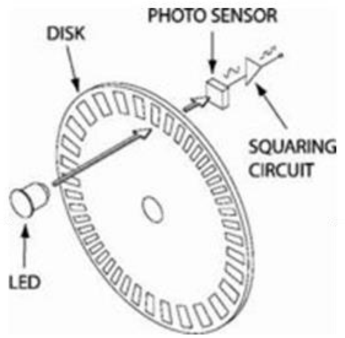
- Dead reckoning localisation
- First level of autonomy: Motion control
- Application: Point-to-point navigation

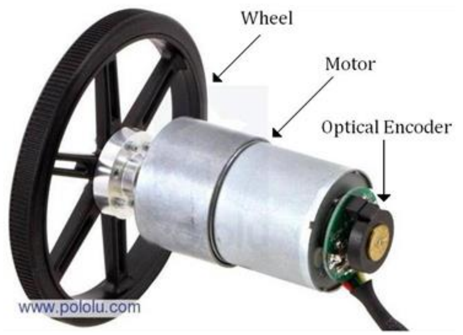
Low Level Control





Wheel velocity control using PID





First level of autonomy: Motion-based Localisation (Dead Reckoning)

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

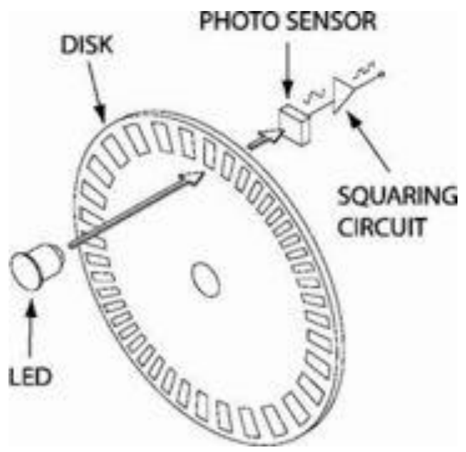
The robot may also employ some proprioceptive sensors such as: encoders or IMUs, to provide a better estimate of the pose change over time. This method is simple to implement and does not require sophisticated sensors.

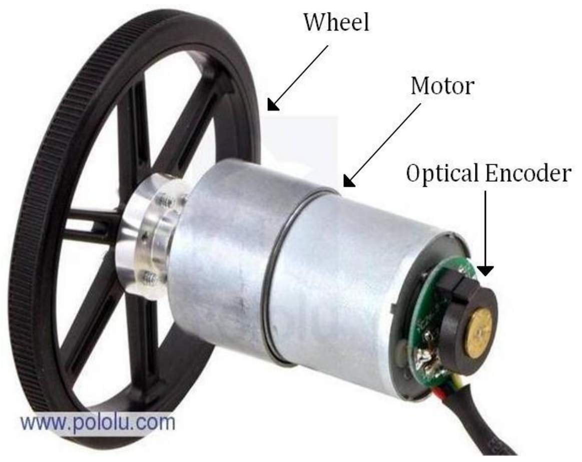
However, such technique suffers from the unbounded growth of uncertainty about the robot pose over time due to the numerical integration and accumulation of error.

Proprioceptive sensors Rotary Encoders

- Rotary optical encoders, are proprioceptive sensors, used to determine the angular position of the shaft they are attached to.
- VERY IMPORTANT!!! Encoders in mobile robots are considered proprioceptive sensors because they only acquire information about the robot itself, not the structure of the environment.

Essentially, it is a mechanical light chopper that produces a certain number of pulses for each shaft revolution.





Proprioceptive sensors Rotary Encoders

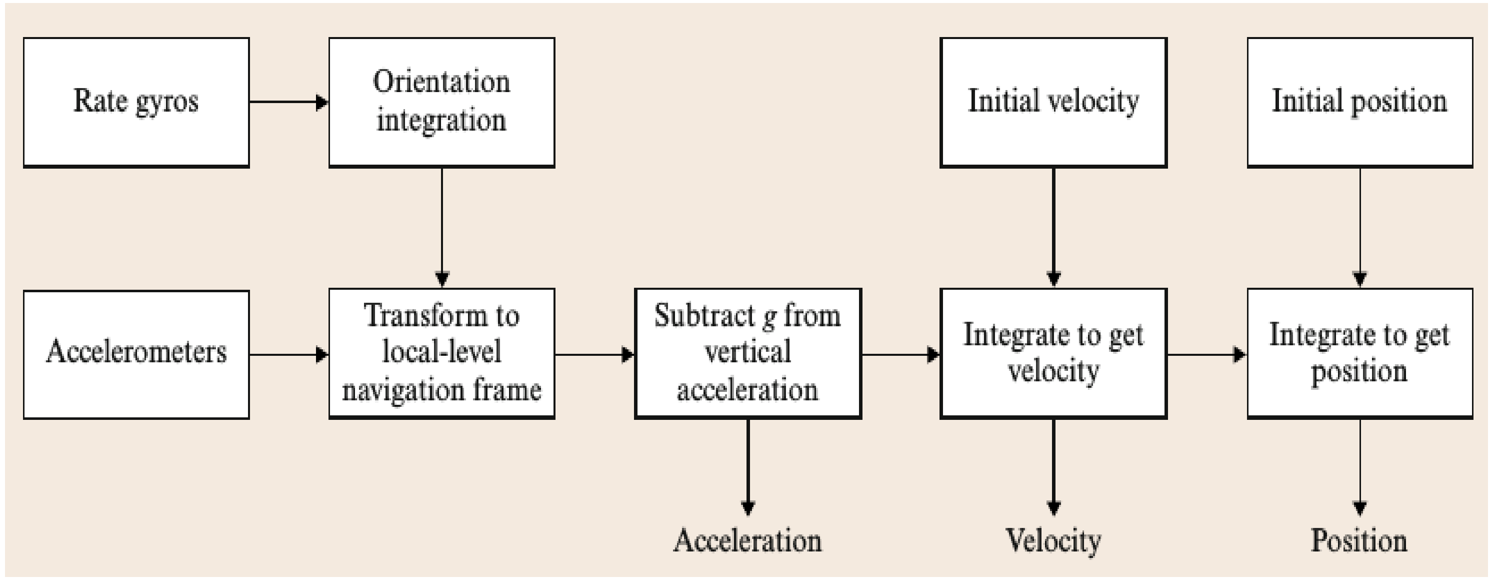
- When an encoder is attached to the axle of each wheel in a differential-drive robot, it is possible to convert the number of pulses into useful information, such as the distance travelled by each wheel. By measuring the wheel diameter, the distance travelled by each wheel is calculated as follows:

$$distance = \frac{counts}{CPR}(\pi d)$$

- CPR – counts per revolution
- d – wheel diameter

Proprioceptive sensors IMUs

Another type of proprioceptive sensing device is the *inertial measurement unit* (IMU). An IMU is an electronic sensor that maintains a 6-degree-of-freedom (DOF) estimate of the relative pose of a mobile vehicle, this is, position in *x*, *y* and *z*, and orientation roll, pitch and yaw. Such task is achieved using a set of gyroscopes and accelerometers. IMUs are a common navigational component of aircrafts and ships, particularly for their capability to determine changes in the yaw, pitch and roll.


Rate gyros
Orientation integration
Initial velocity
Initial position
Accelerometers
Transform to local-level navigation frame
Subtract g from vertical acceleration
Integrate to get velocity
Integrate to get position
Acceleration
Velocity
Position

Motion-based Localisation (Dead Reckoning)

Kinematic model for a differential robot model

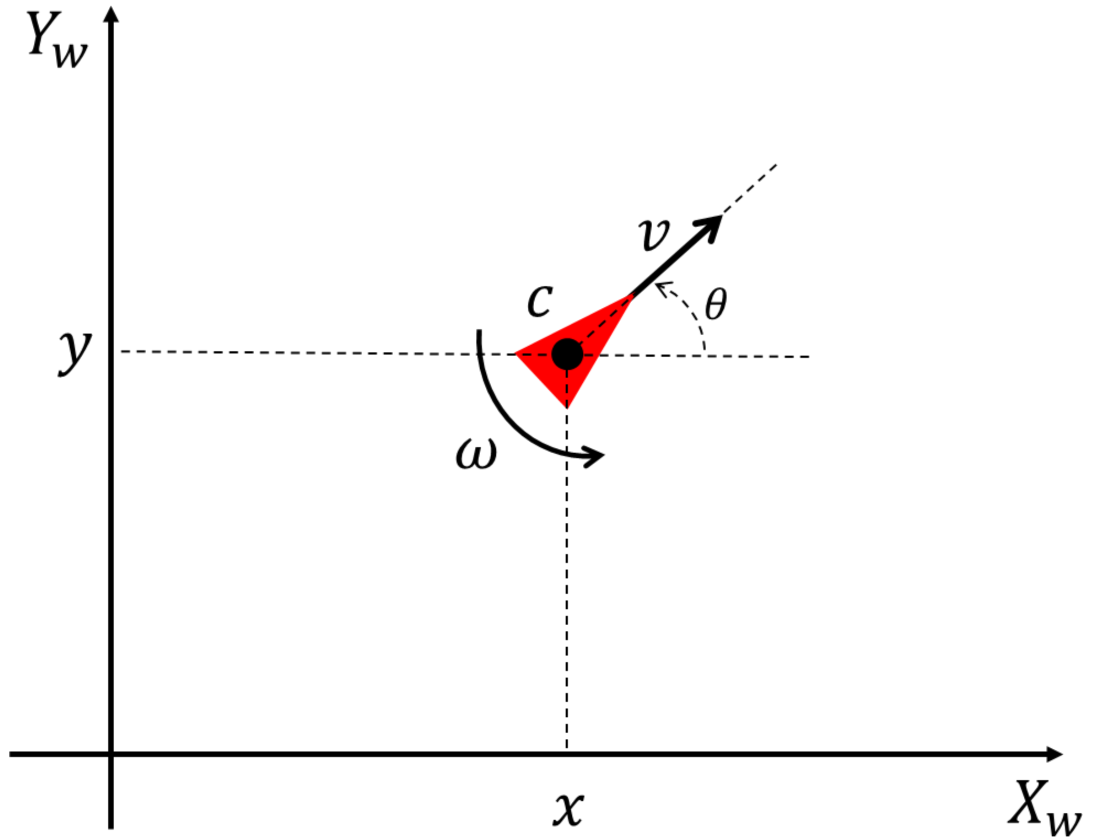
$$\frac{d}{dt}\begin{bmatrix} s_x \\ s_y \\ s_\theta \end{bmatrix} = \begin{bmatrix} \cos(s_\theta) & 0 \\ \sin(s_\theta) & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} v \\ \omega \end{bmatrix}$$

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:

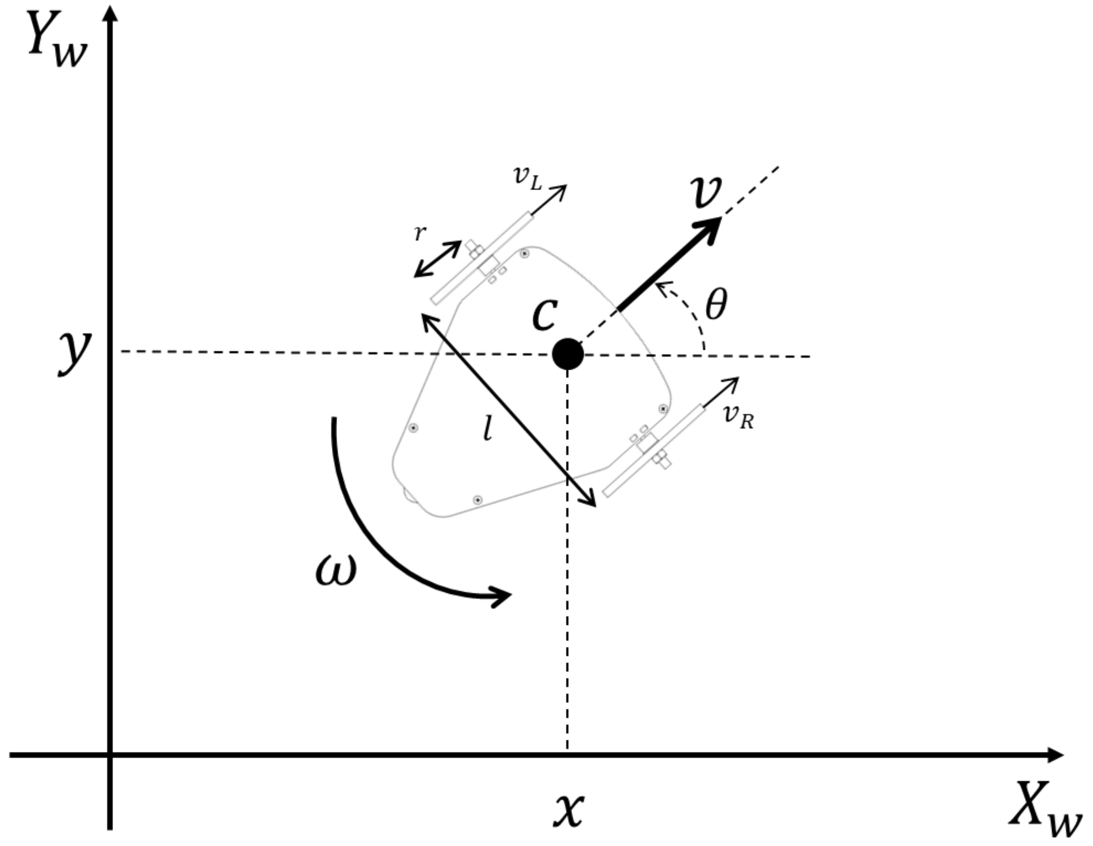
$$\Delta d = v \cdot \Delta t \qquad\qquad \Delta\theta = \omega \cdot \Delta t$$

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

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

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

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



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.

Autonomous Systems

Motion Control

{Learn, Create, Innovate};

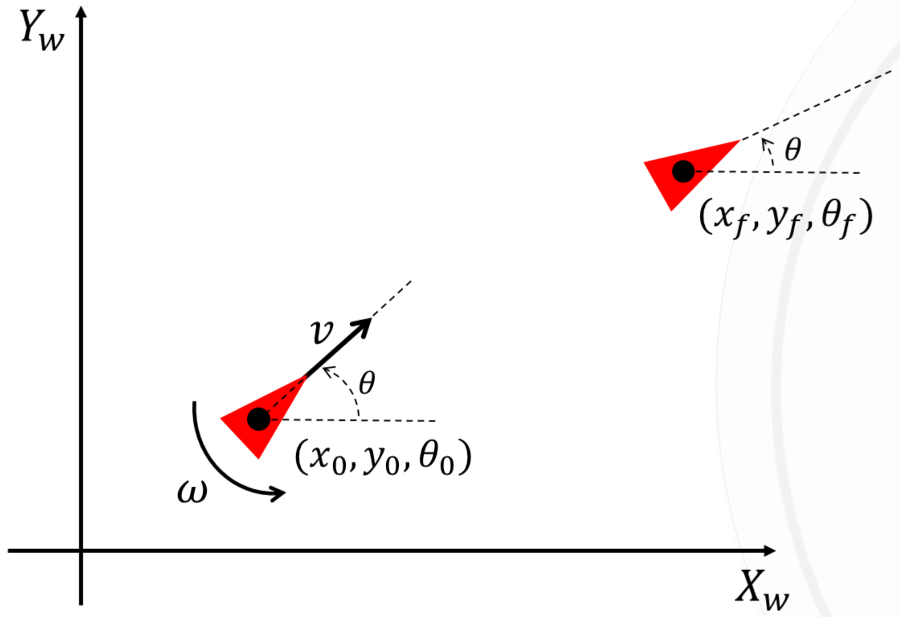
MCR2 Manchester Robotics

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.

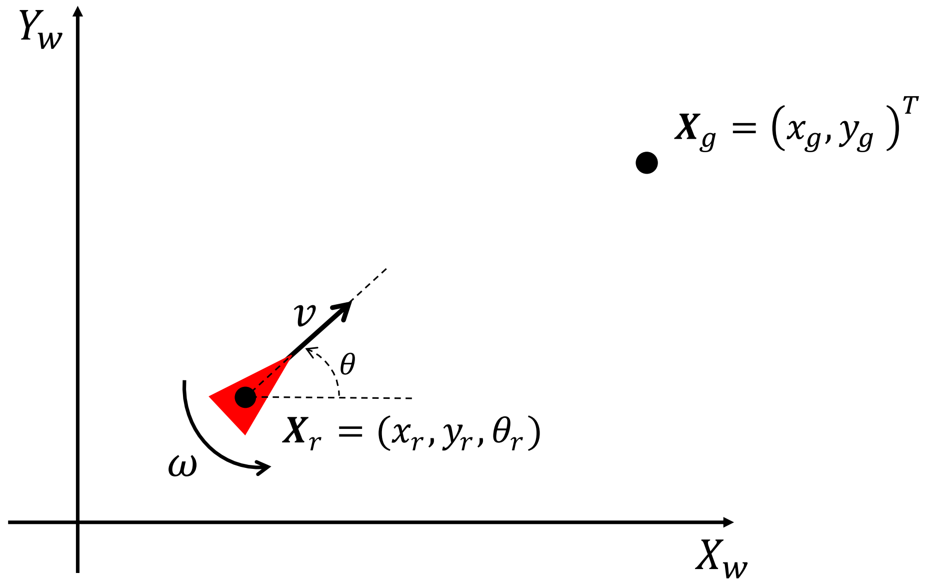


Point Stabilisation

- 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:

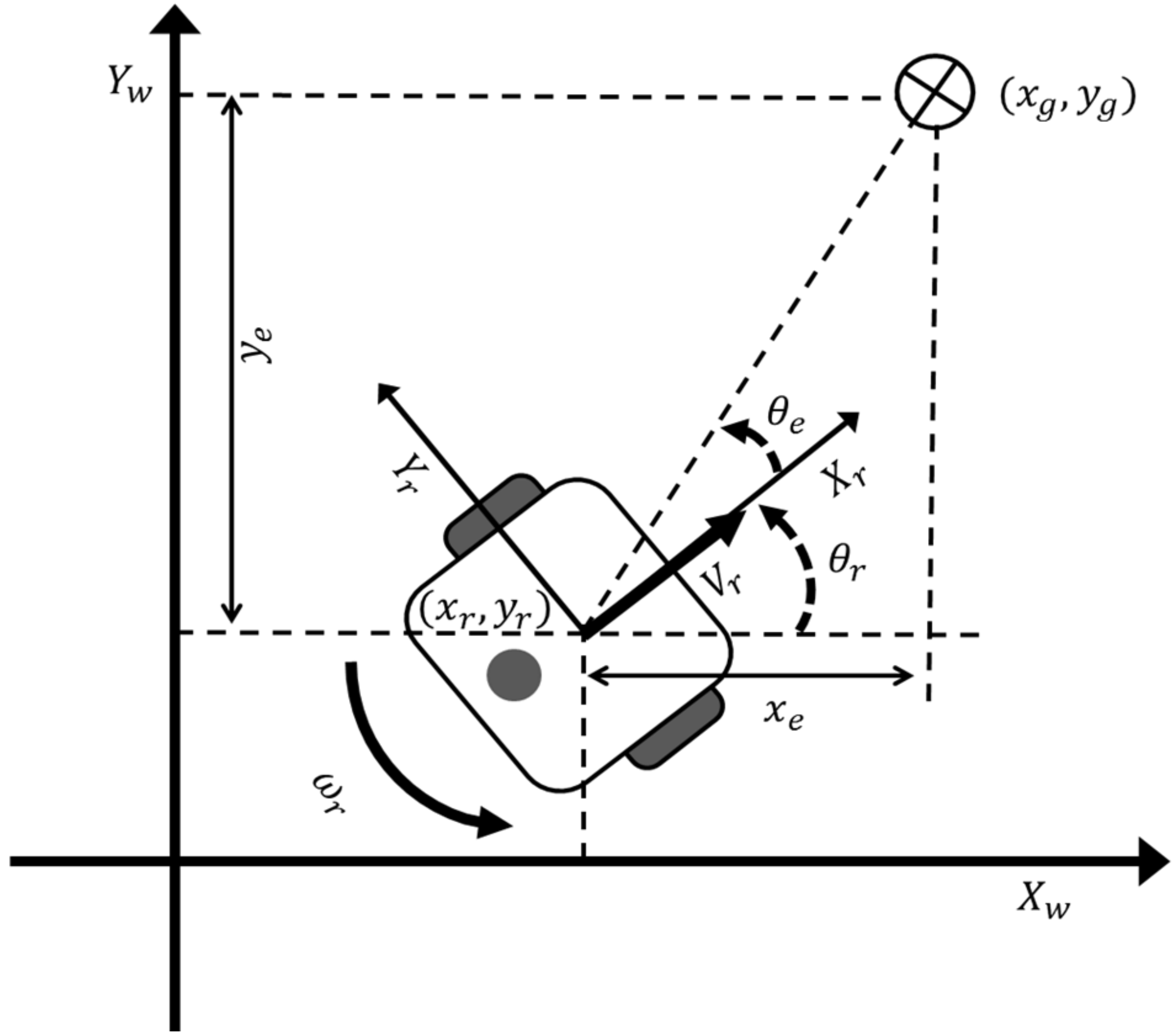
- $\boldsymbol{X}_g = \begin{pmatrix} x_g \\ y_g \end{pmatrix}$ if the position of the robot needs to be controlled, or

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



Y_w
$\boldsymbol{X}_g = (x_g, y_g)^T$
v
θ
ω
$\boldsymbol{X}_r = (x_r, y_r, \theta_r)$
X_w

Point Stabilisation

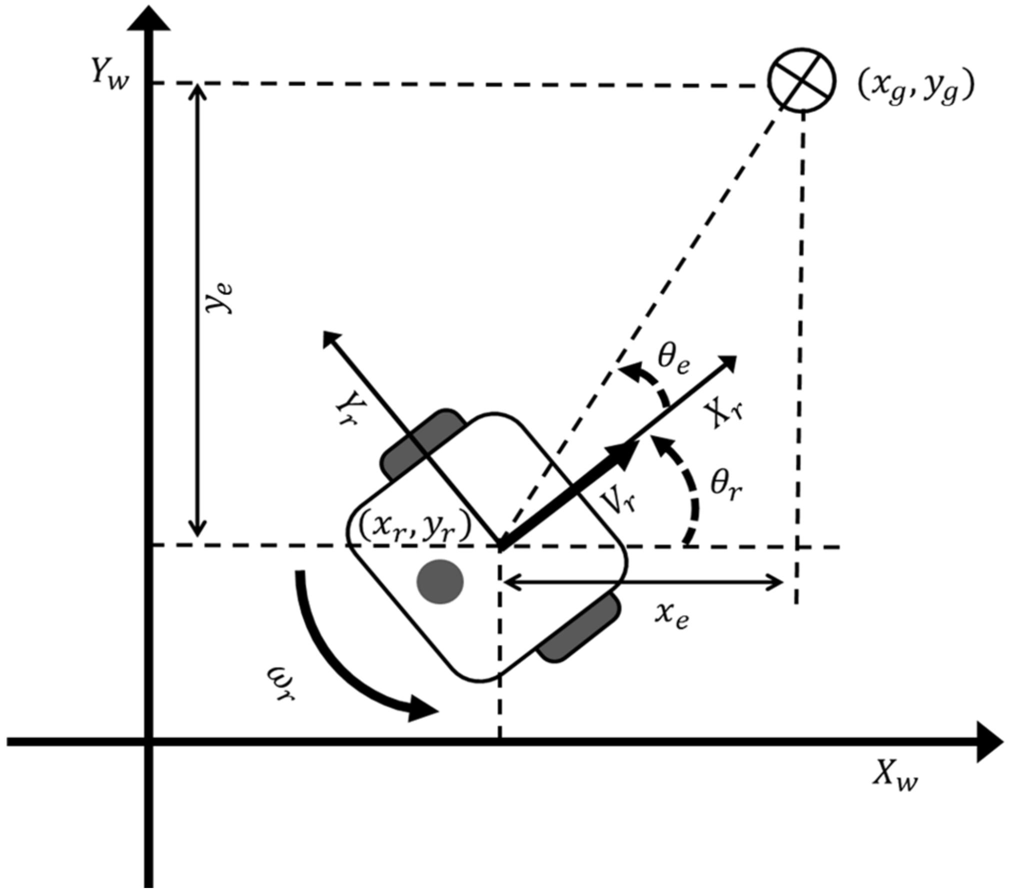


Point Stabilisation

- 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 $\boldsymbol{v}_r = \begin{pmatrix} v_r \\ \omega_r \end{pmatrix}$. The linear velocity of the robot, v_r, is always oriented in the direction of x axis of the robot reference frame because of the non-holonomic constraint .



Point Stabilisation

- For the sake of simplicity, in this course, the goal is defined only by a 2D set of coordinates

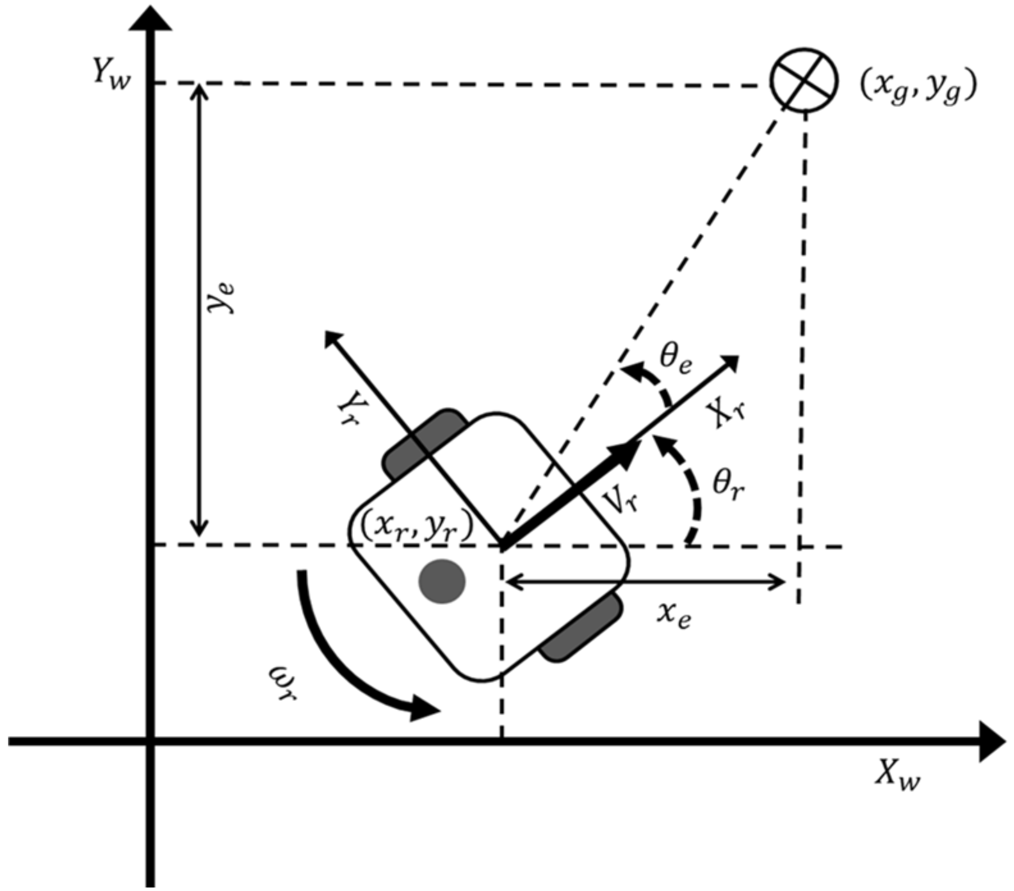
$$\boldsymbol{X}_g = \begin{pmatrix} x_g \\ y_g \end{pmatrix}.$$

- 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.

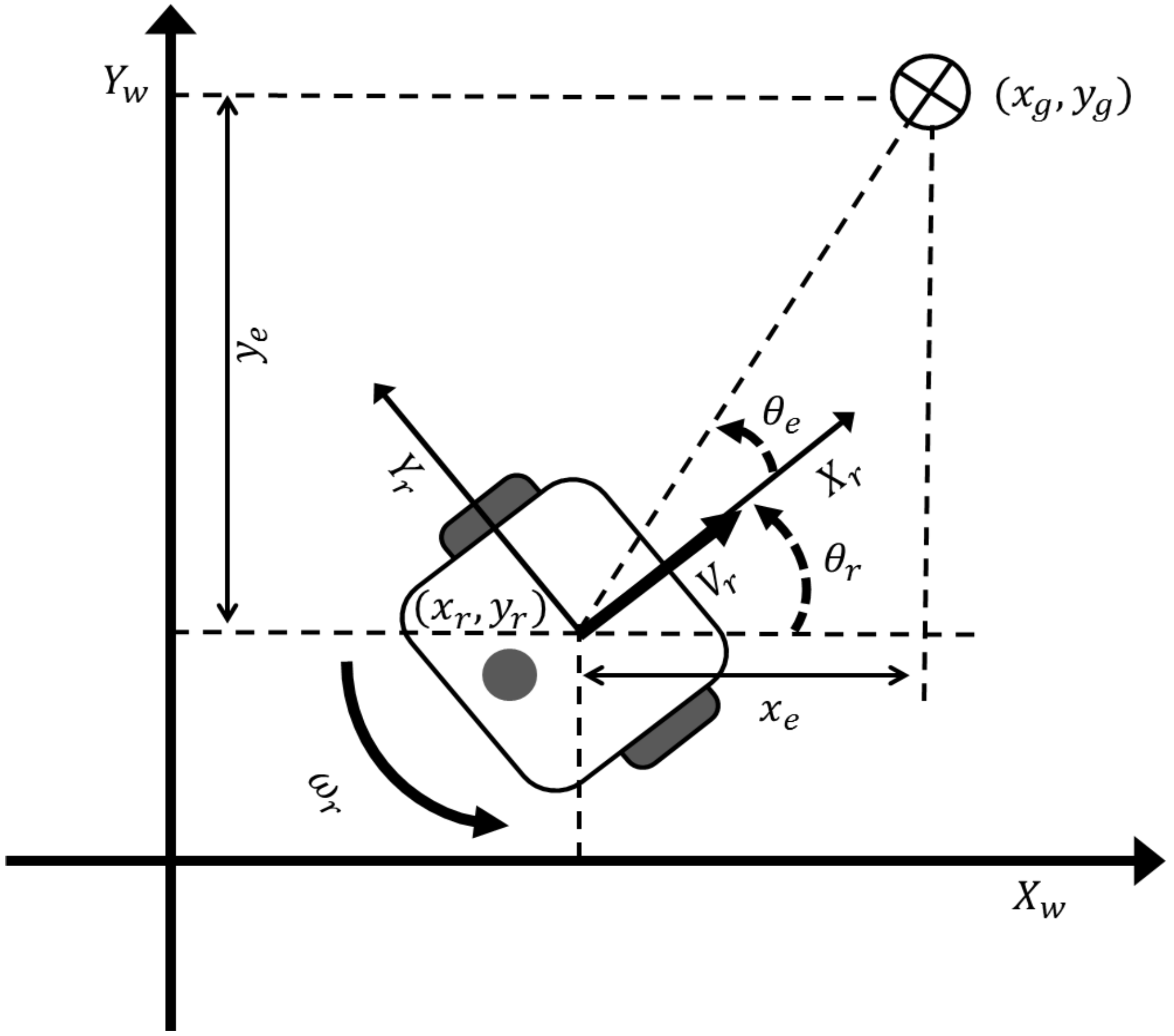
Point Stabilisation

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

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

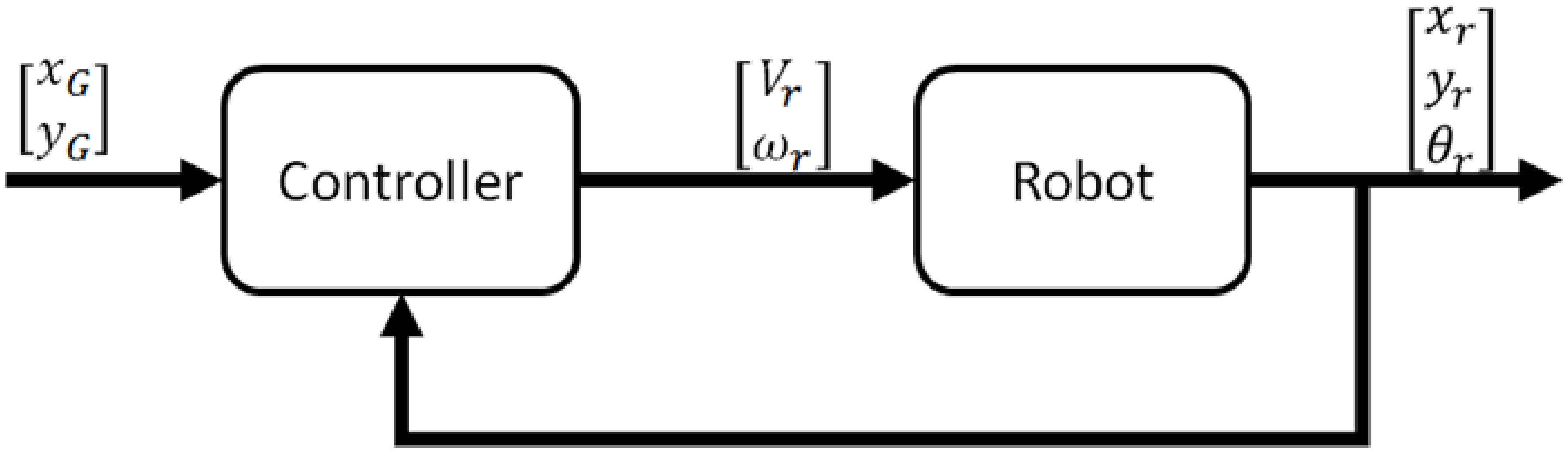
- The control law can now be written as:

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



Point Stabilisation

- Point Stabilisation block diagram:


$\begin{bmatrix} x_G \\ y_G \end{bmatrix}$
Controller
$\begin{bmatrix} V_r \\ \omega_r \end{bmatrix}$
Robot
$\begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}$