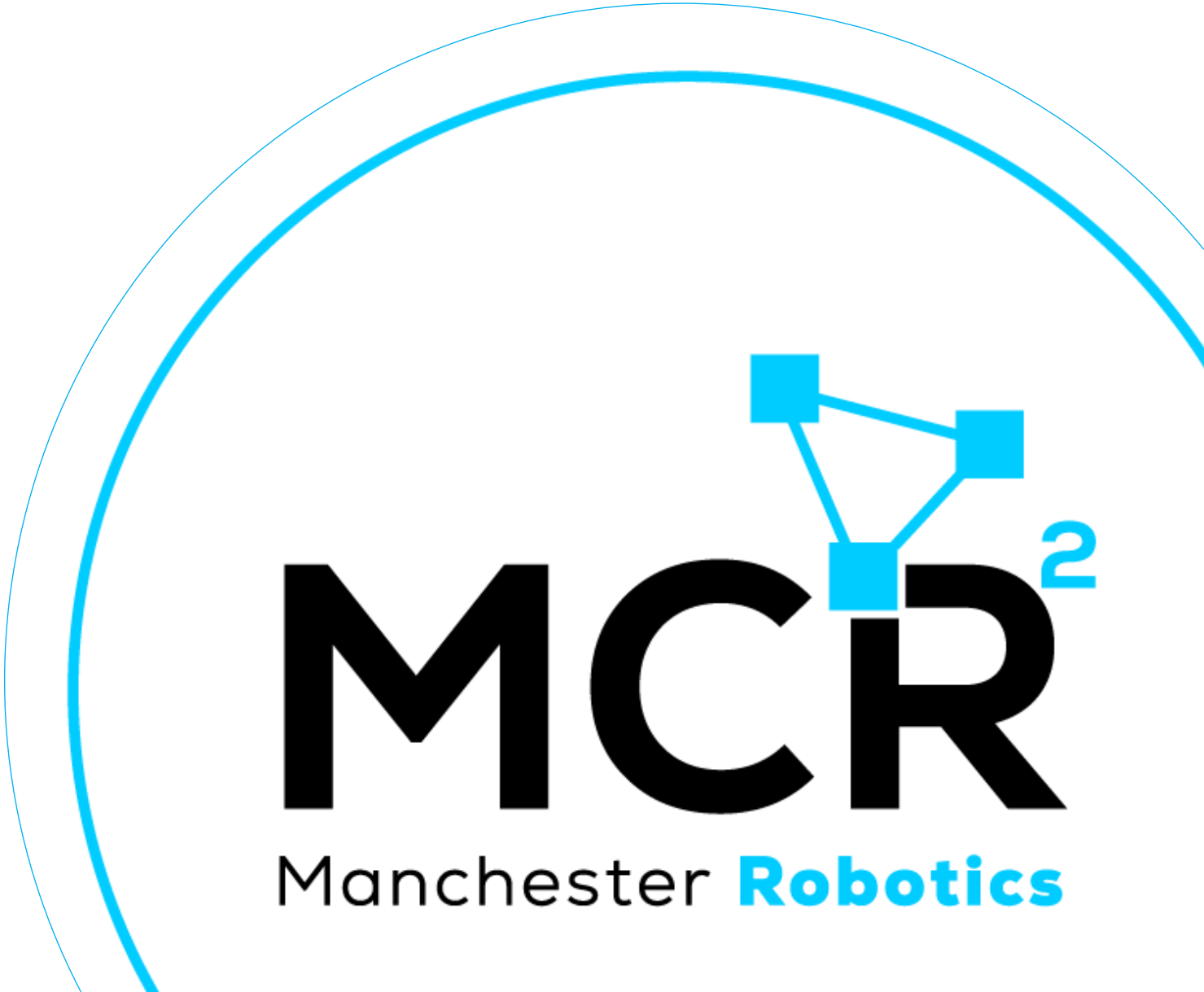
{Learn, Create, Innovate};

Modelling and Simulation

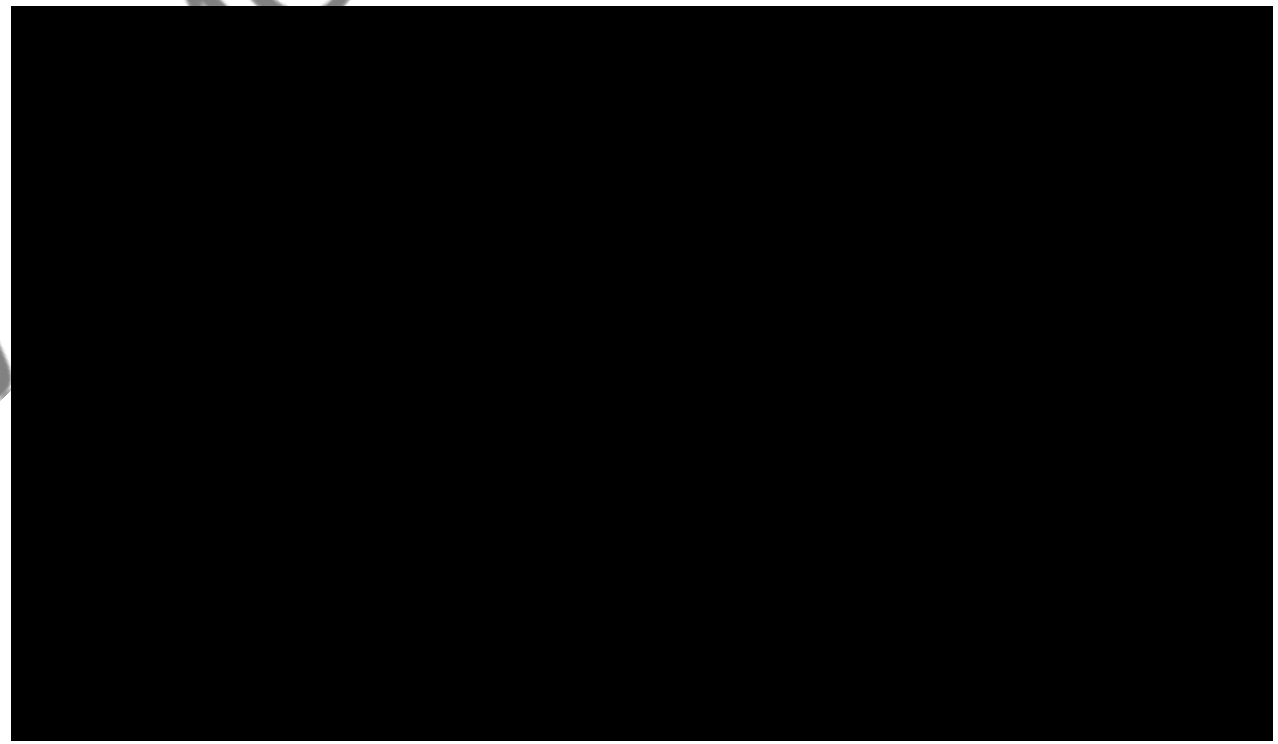
Introduction



Simulation

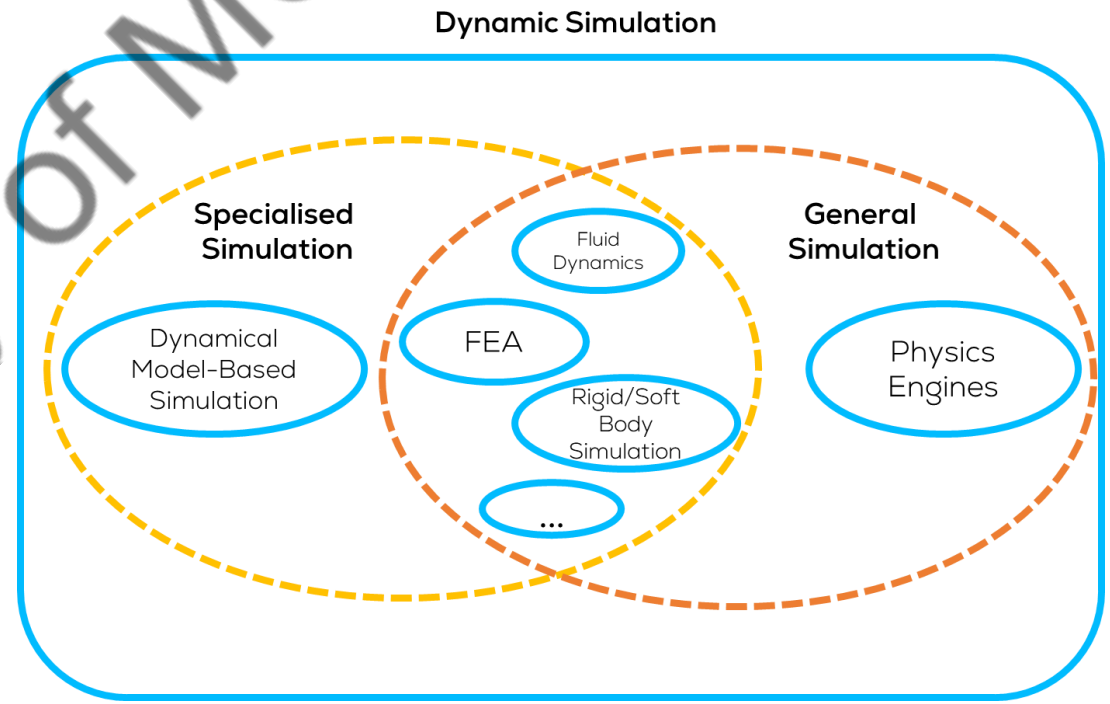
- **Simulation** refers to the use of a computer program to model the time-varying behaviour of a system.
- Usually, ordinary differential equations or partial differential equations describe the systems.
- The simulator solves these equations to determine the behaviour of state variables over a specified time.
- Creating a model of a system allows for predicting the values of the model-system state variables based on past state values.

Lorenz Attractor



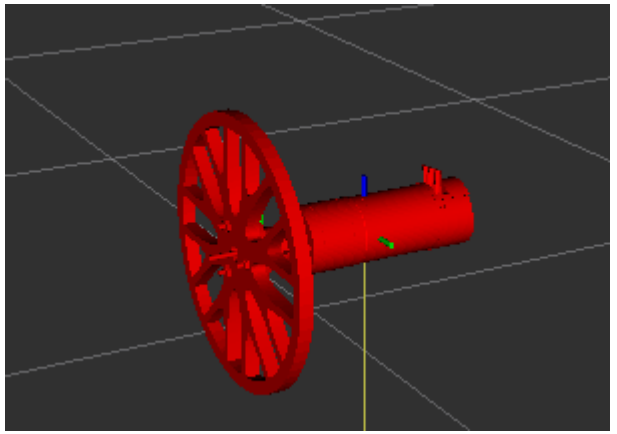
Categories of Simulations

- Simulations can be subdivided into different categories. In robotics and mechanics, in general, two main categories are used:
  - **Specialised**: designed for one particular mechanism or system.
  - **General:** uses different (rigid, soft, fluid, etc.) body physics engines that simulate just about any configuration.



Specialised Simulations

- Specialised simulation are done as follows:
    1. Deriving the governing equations (mathematical model) of the system at a particular configuration.
    2. Define the States of the systems to be analysed (some systems can have thousands of state variables).
    3. The next step is using a computer to solve the governing equations, using different computational tools.
        - This process involves numerical analysis and numerical methods, such as Euler Integration or Runge Kutta Methods.
    4. For simulations that involve collisions, additional steps are required to define the collisions such as backing up in time to the moment before the collision to modify the velocities, based on the collisions input.
    5. Finally, specify details about how to display the results and the user input.
- Usually, this type of simulation is used for simple or very complex systems where physical engines fail due to dynamical behaviour or computational power.



Specialised Simulations

- As stated before, specialised simulation can be used for simple systems, dedicated simulations or very complex systems where the user requires more control over the dynamical behaviour.
- Specialised simulators, although widely used in robotics and other engineering fields, they have some disadvantages, shown in the following table.

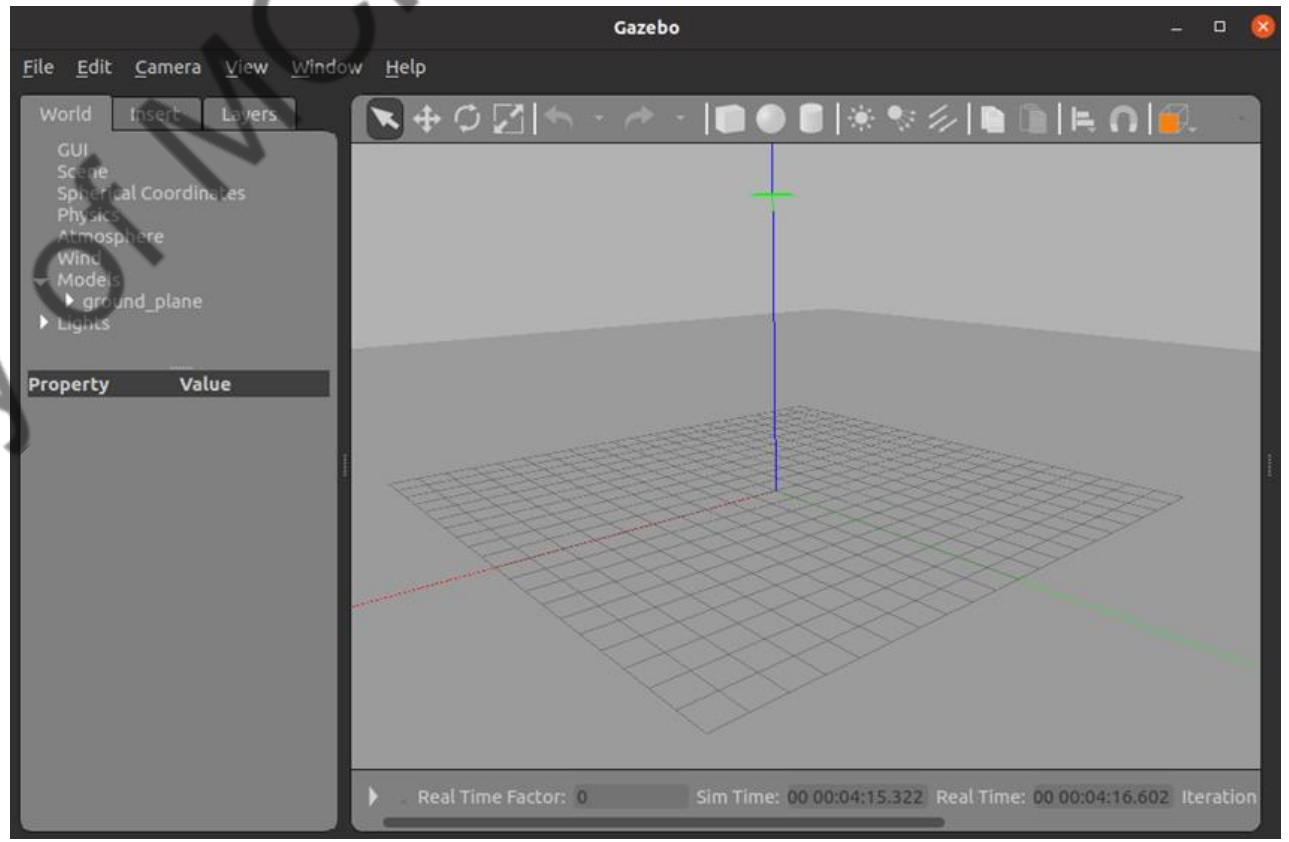
Pros	Cons
• **Flexibility:** They can be used to model a wide range of systems and phenomena, not limited to physics • Customization: You have greater control over the behaviour of the simulated objects and can define custom rules and interactions to suit your specific needs. • **Interdisciplinary:** Can be applied in fields beyond physics, including engineering, biology, economics, and social sciences. • **Realism:** Dynamic simulations can sometimes achieve higher realism for specific scenarios than generic physics engines.	• **Complexity:** Building and fine-tuning dynamic simulations can be complex and time-consuming. • **Performance:** Due to their custom nature, dynamic simulations may not always be as efficient as physics engines for real-time applications like video games and some robotic applications. • **Limited Physics Realism:** Dynamic simulations may lack some physical accuracy, especially when interacting with other components, unlike some physical engines.

Specialised Simulations

- An example in the “specialised" category can be a simple pendulum system.
  - That simulation can only simulate that particular mechanism.
  - The length of the pendulum, the mass of the objects, or spring stiffness can be modified.
  - The problems of this simulation how the parts are connected (change unions), apply new forces, or collide with another object. For such cases, new governing equations must be derived or using another type of simulator.

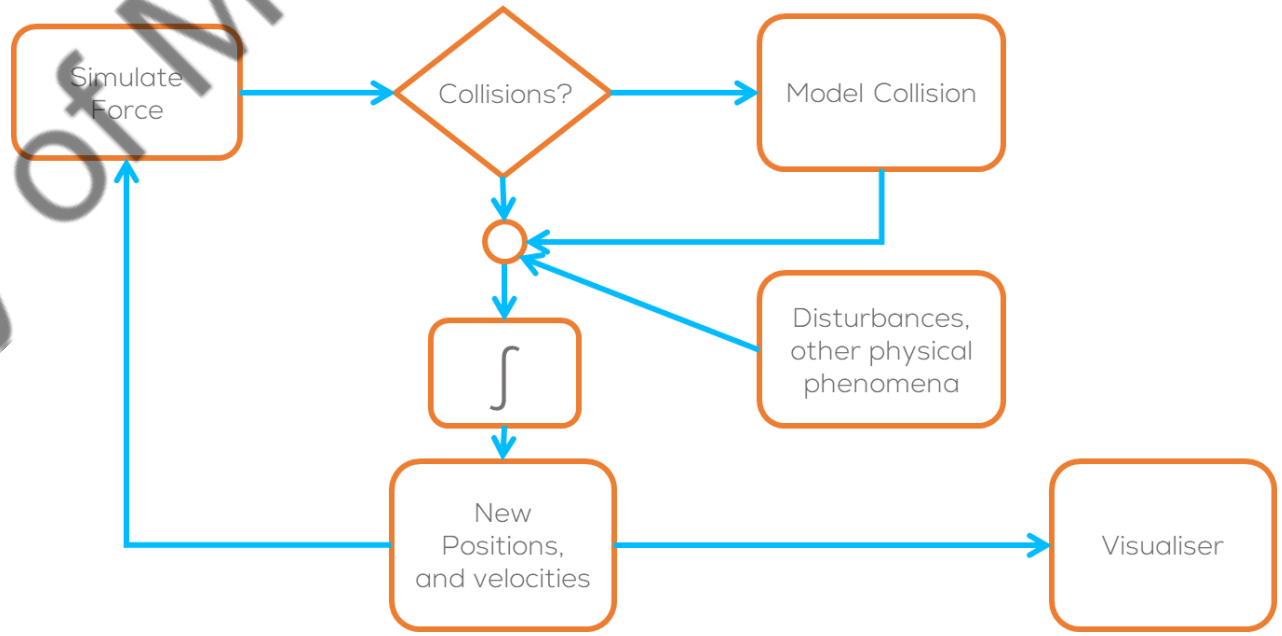
General Simulations

- In contrast, general simulators such as physics engines can model a wide variety of mechanisms.
- The physics engine calculates an object's acceleration, velocity, and displacement based on the forces and torques acting upon it.
- A physics engine is responsible of forming the equations of motion according to the systems' characteristics, solving the equations of motion, and detecting collisions and interactions with other components.
- Some software using physics engines are CoppeliaSim, Gazebo, Unity, etc.



General Simulations

- Physics engines work as a continuous loop.
  - Initialises by checking the joints and establishing the dynamical equations to solve.
  - Identifies all the forces and moments acting on the objects.
  - Solve the equations, by integrating the accelerations to obtain the velocities and positions of the objects.
  - At each new time step, if it detects any collisions, then recalculates the velocity and position of an object.
  - Finally, send the location data to the visualiser.
- This continuous loop is repeated until the simulation is complete.



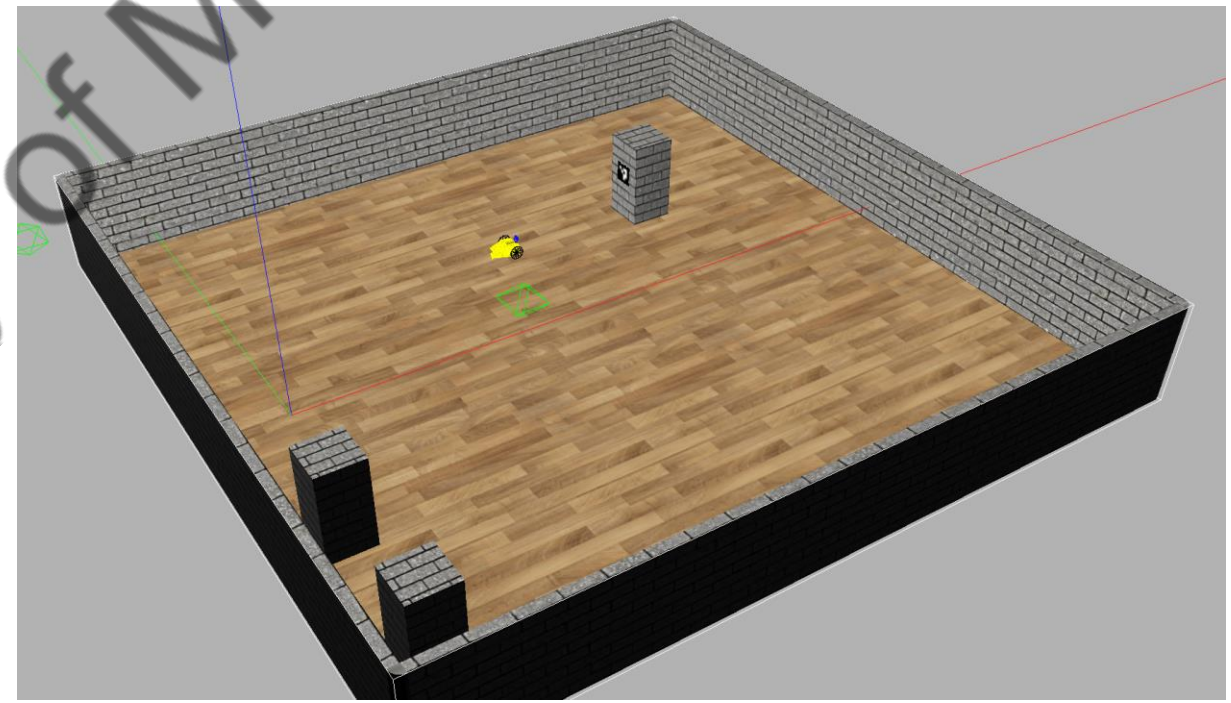
General Simulations

- As stated before, general simulation can be used for a wider range of systems where the user prioritises the behaviour of the system and its interaction with the environment and other systems.
- Physics engine simulators, although widely used in robotics and computer science, have some disadvantages, as shown in the following table.

Advantages	Disadvantages
• **Realistic Physics:** Physics engines provide realistic physics simulations out of the box, making them well-suited for applications where physics interactions are critical. • **Efficiency:** They are optimized for performance and can handle real-time simulations, making them ideal for video games and interactive simulations. • **Collision Detection:** Physics engines excel at collision detection and response, which is essential for creating realistic interactions between objects. • **Pre-built Components:** They come with pre-built components for common physical phenomena, saving development time	• **Limited Flexibility:** Physics engines are specialized and may not be as flexible as dynamic simulations. Modifying their behavior or adding custom rules can be challenging. • **Complex Integration:** Integrating a physics engine into a project can be complex, especially for beginners. • **Resource Intensive:** Realistic physics simulations can be resource-intensive and may require optimization for performance.

General Simulations

- One example is a robot simulated in Gazebo, using the physics engine to interact with the environment.
  - The simulator takes into consideration the light, forces applied to each joint of the robot, collisions, friction, inertias, etc.
  - Also simulate the interaction with cameras, LiDARS, and other sensors.



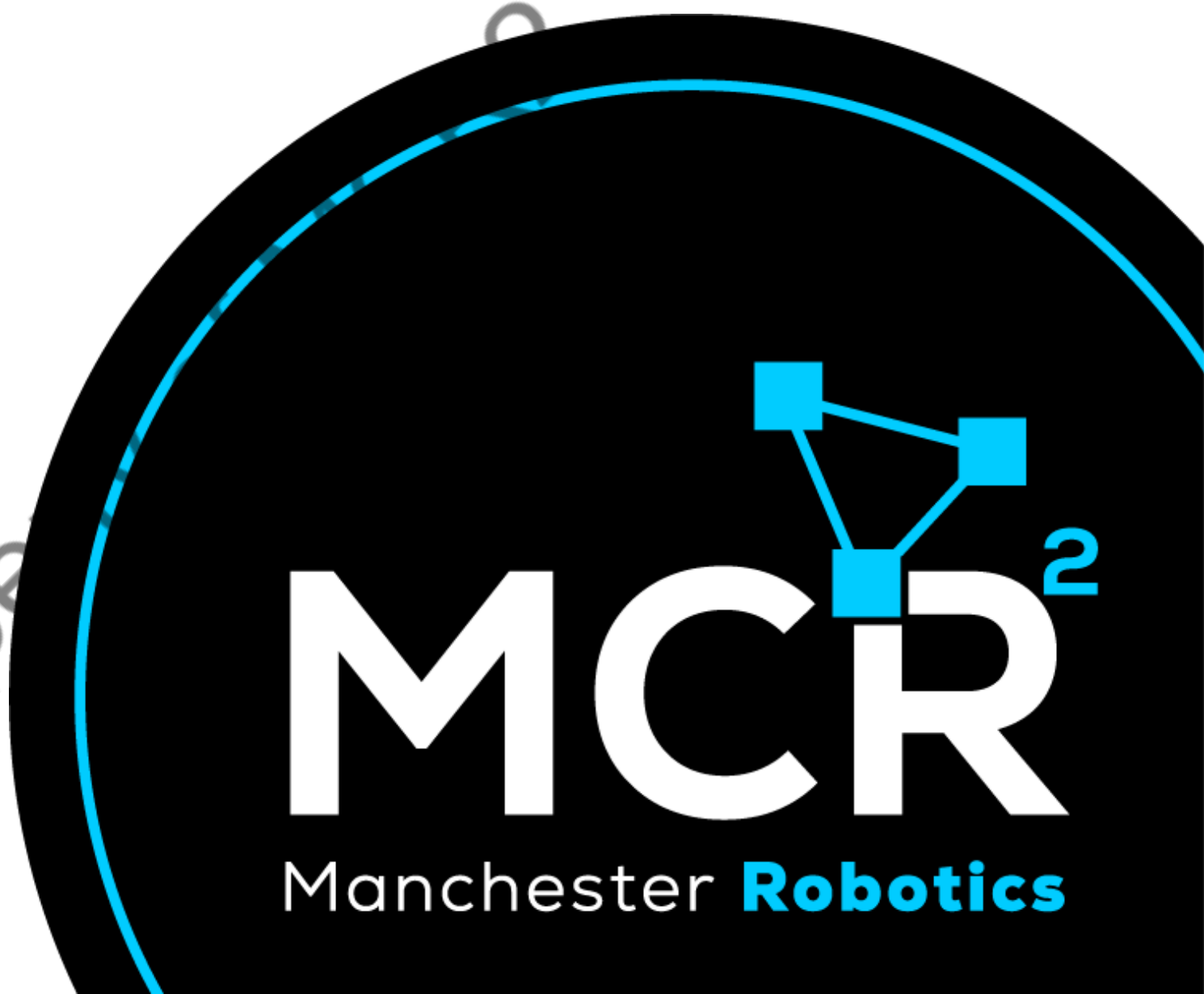
One vs. the other?

- In practice, the choice between specialised simulation and a physics engine depends on the specific requirements of the application.
- For video games and interactive simulations that rely heavily on realistic physics, a physics engine is often the preferred choice.
- Dynamic Model simulations are more suitable when flexibility and customization are paramount or when modelling complex systems that go beyond traditional physics.
- In some cases, a combination of both approaches may be used to achieve the desired level of realism and customisation.

{Learn, Create, Innovate};

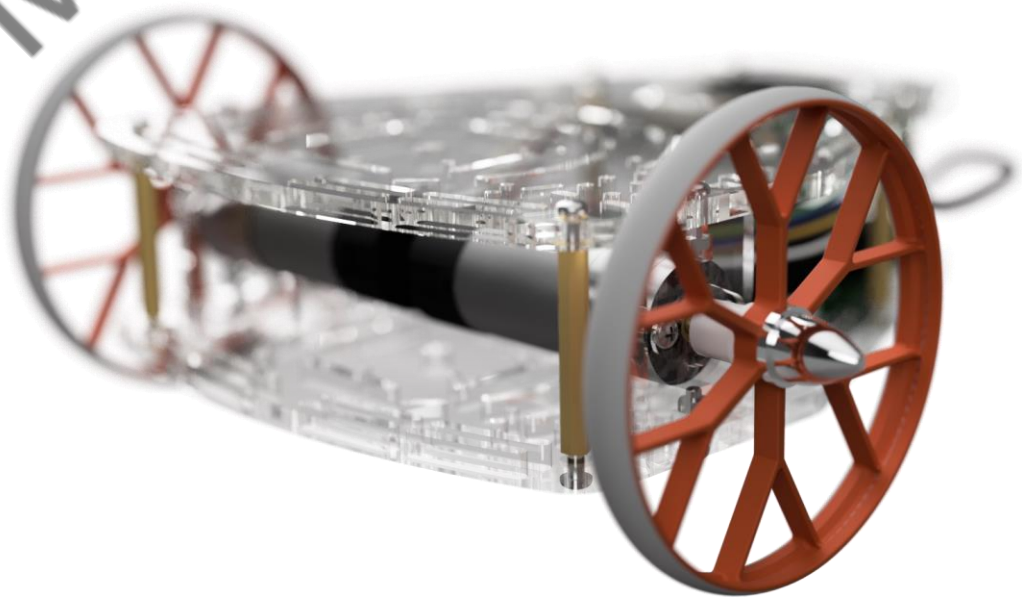
Mobile Robots

System Modelling



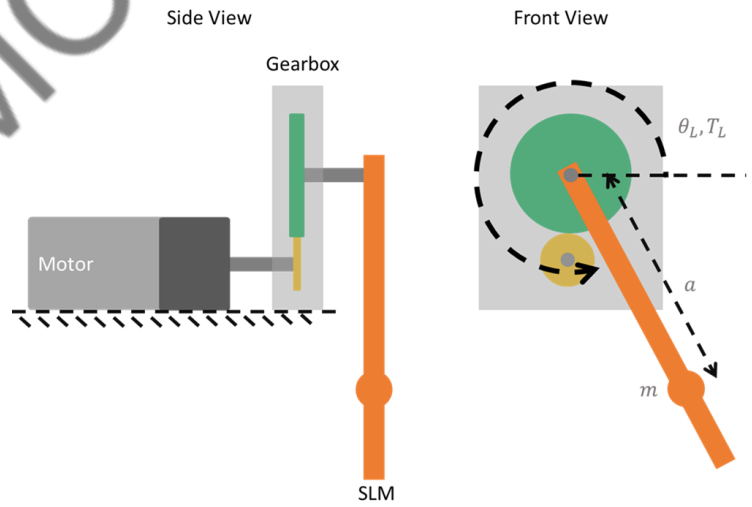
System Modelling

- In engineering, a system is an entity that consists of interconnected components built with a desired purpose. In this case, the system is said to be the differential drive robot.
- Systems can be modelled depending on their behavior.
- Models describe the behavior of a system using mathematical concepts and language.
    - Outputs depend on the present and past values of the inputs.
    - Changes over time.
    - Sometimes called dynamic systems or sequential systems.
    - Mathematically described with differential or difference equations.



System Modelling: Dynamical Models

- Considers different time-varying phenomena and the interaction between motions, forces and material properties.
- Incorporates physical parameters such as mass, friction, inertia, and forces/torques.
- Characteristics:
  - Mass and inertia
  - Force and torque inputs
  - Acceleration
  - Energy consumption



$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = (GK_m x_3 - mga\cos(x_1))\left(\dfrac{1}{G^2 J_m + J + ma^2}\right) \\ \dot{x}_3 = \dfrac{V_a}{L} - \dfrac{K_b G}{L}\, x_2 - \dfrac{R}{L} x_3 \end{cases}$$

System Modelling: Dynamical Models

Dynamic Model of a Differential Drive Robot*:

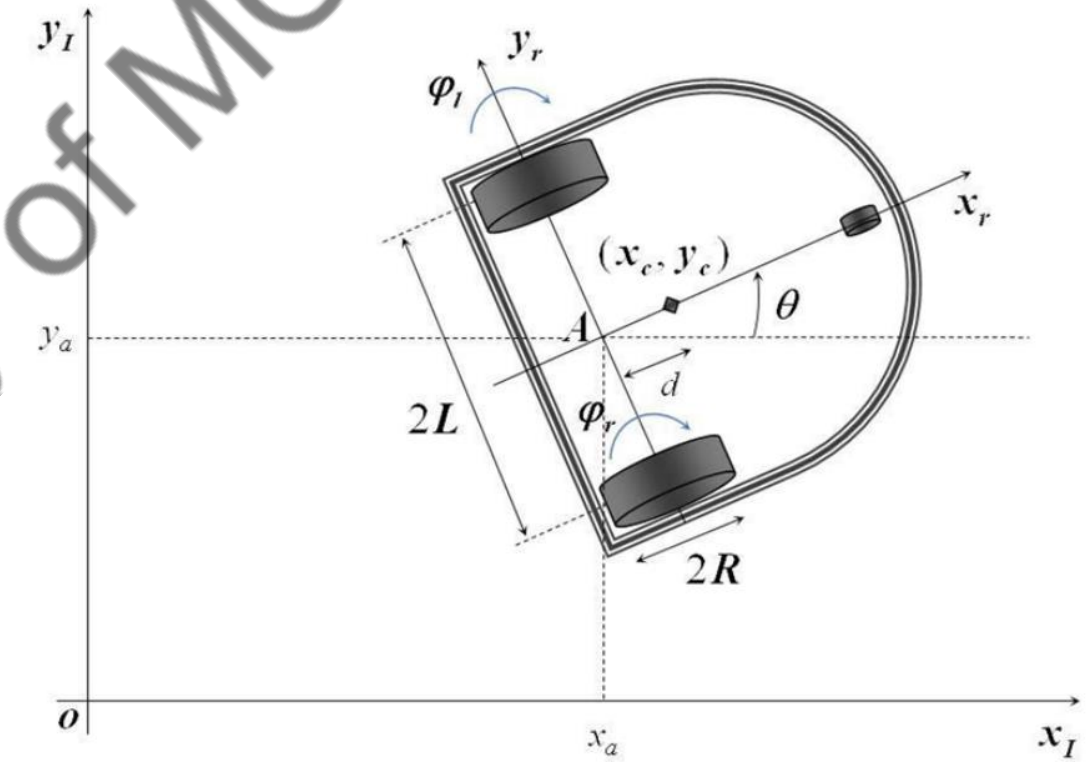
$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{\eta}} + \boldsymbol{V}(\boldsymbol{q},\dot{\boldsymbol{q}})\boldsymbol{\eta} = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{\tau}$$

$$\boldsymbol{M}(\boldsymbol{q}) = \begin{bmatrix} I_w + \frac{R^2}{4L^2}(mL^2 + I) & \frac{R^2}{4L^2}(mL^2 - I) \\ \frac{R^2}{4L^2}(mL^2 - I) & I_w + \frac{R^2}{4L^2}(mL^2 + I) \end{bmatrix}$$

$$\boldsymbol{V}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \begin{bmatrix} 0 & \frac{R^2}{2L} m_c d\dot{\theta} \\ -\frac{R^2}{2L} m_c d\dot{\theta} & 0 \end{bmatrix}$$

$$\boldsymbol{B}(\boldsymbol{q}) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\boldsymbol{\eta} = \begin{bmatrix} \varphi_r \\ \varphi_l \end{bmatrix}, \boldsymbol{\tau} = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}$$

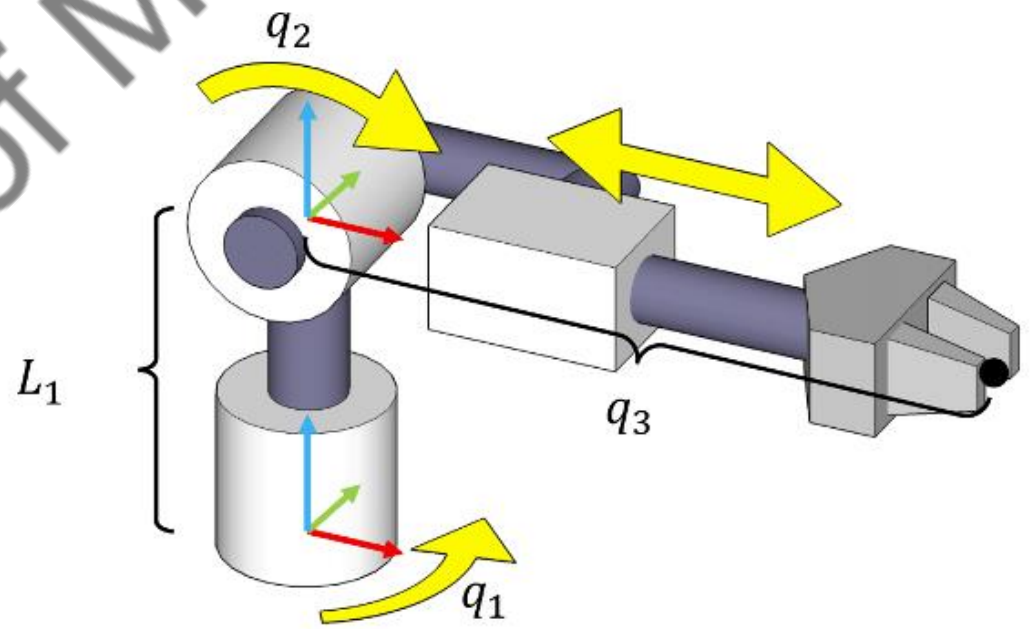


Differential Drive Robot

* Dhaouadi R, Hatab AA (2013) Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework. Adv Robot Autom 2: 107. doi: 10.4172/2168-9695.1000107

System Modelling: Kinematic Models

- Kinematic modelling: Studies the motion of a robot or mechanism under a set of constraints, regardless of the forces that cause it.
- A mathematical representation of a robot's motion that overlooks the forces that affect motion and focuses on the geometrical relationship among components.
- Describes robot motion based purely on geometry and constraints, without considering forces or torques.
- Characteristics:
  - Position
  - Velocity
  - Orientation
  - Constraints (holonomic and nonholonomic)



Robot Modelling: Kinematic Models

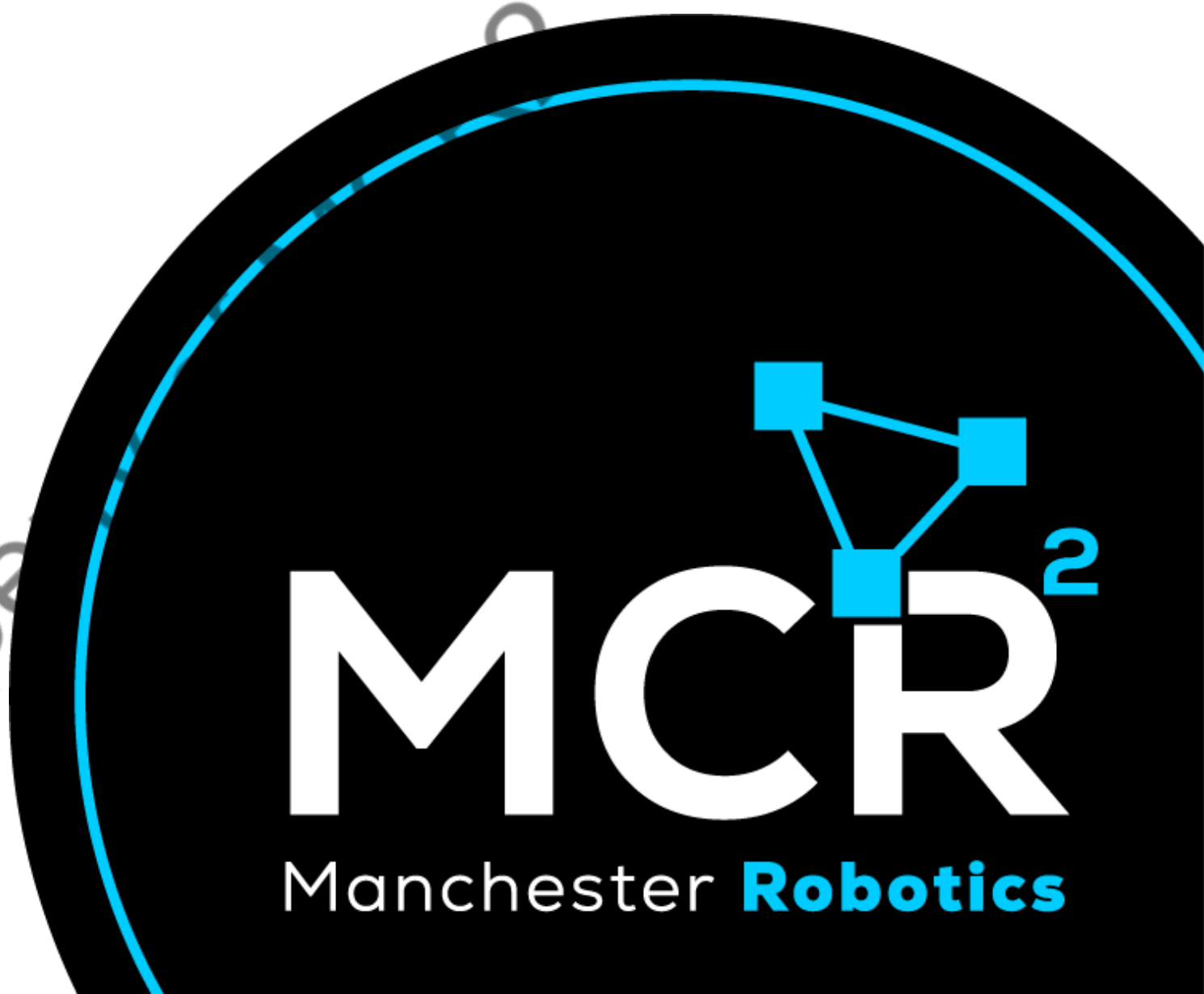
Differences Summarized

Aspect	Kinematic Model	Dynamical Model
Complexity	Lower	Higher
Parameters	Geometry, velocities	Mass, inertia, forces, torques
Applications	Path planning, simple control	Detailed control, stability analysis
Computation	Fast and efficient	Computationally intensive

{Learn, Create, Innovate};

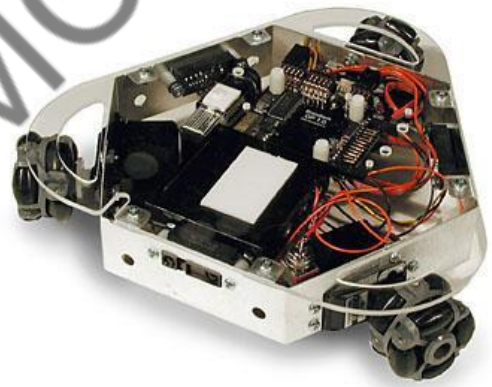
Mobile Robots

Kinematic Models of Mobile Robots

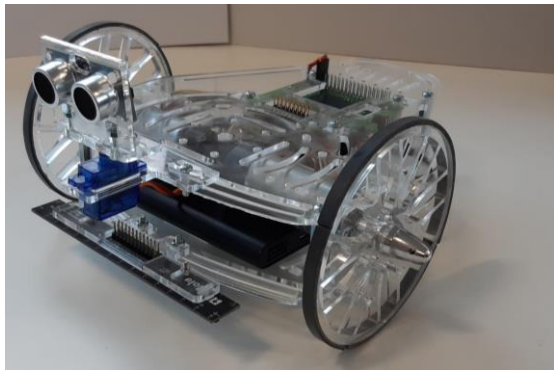


Mobile Robots

- 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 ©.

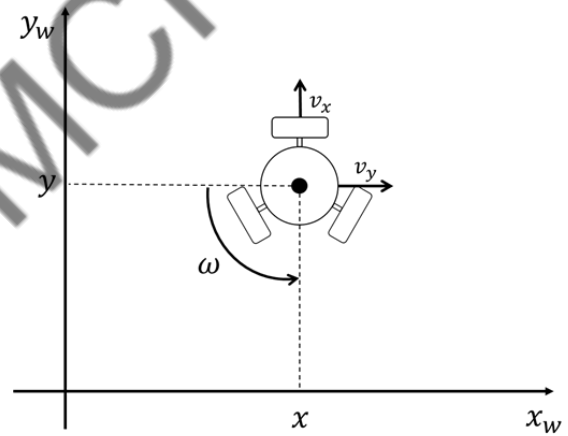


Differential-drive
Puzzlebot ©.

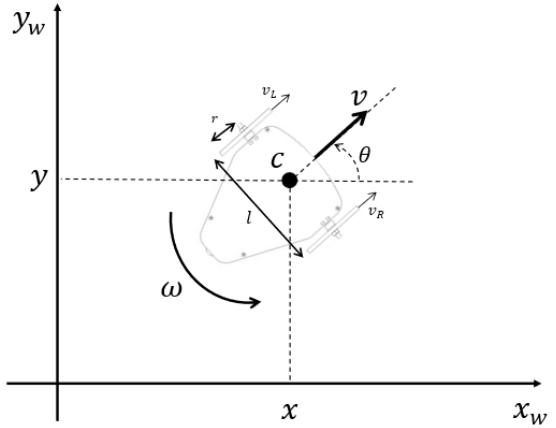
Mobile Robots

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.



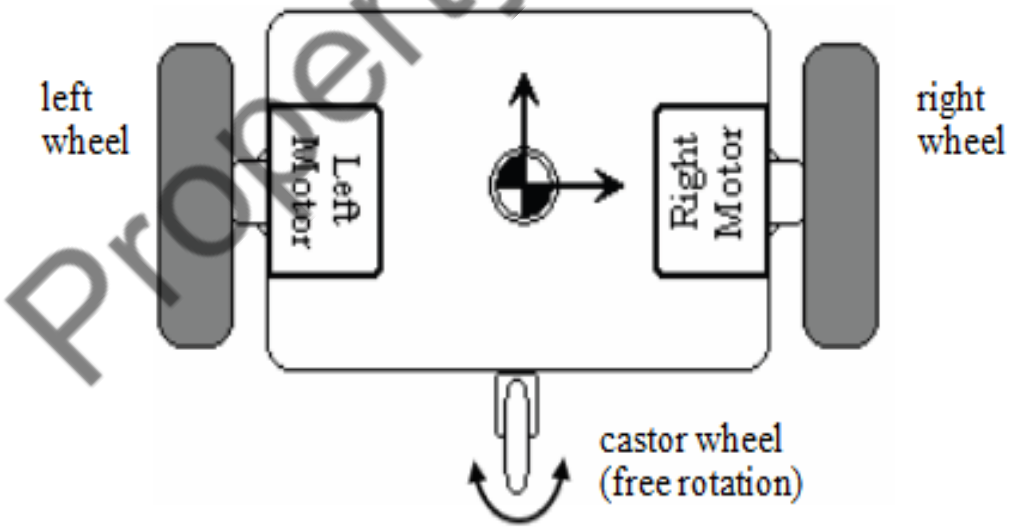
Holonomic Robot



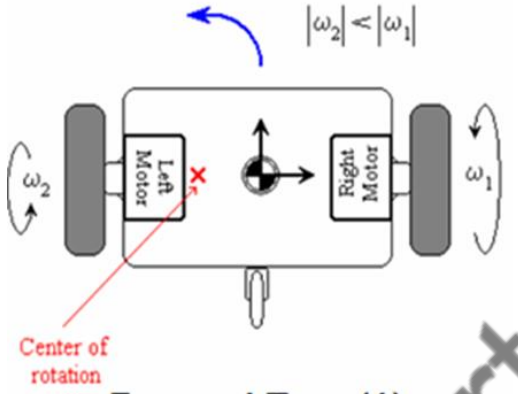
Nonholonomic Robot

Differential drive robots

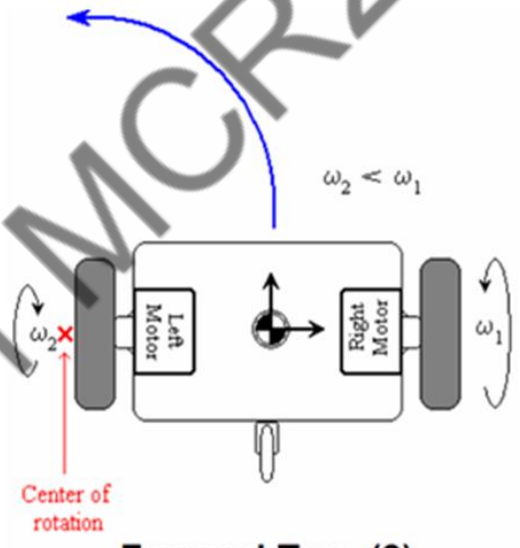
- Also known as differential wheeled robots, these are mobile robots whose movement is based on two separately driven wheels placed on either side of the robot body. It can thus change its direction by varying the relative rate of rotation of its wheels, thereby requiring no additional steering motion.



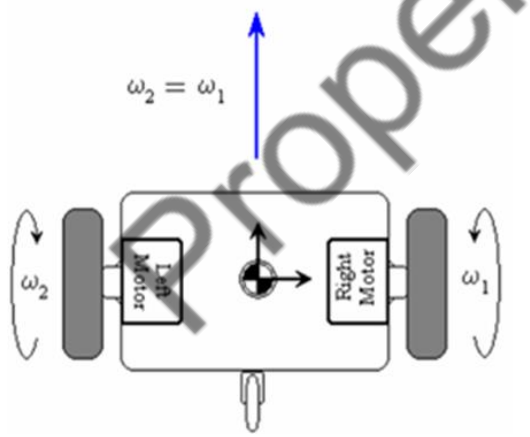
Differential drive kinematics

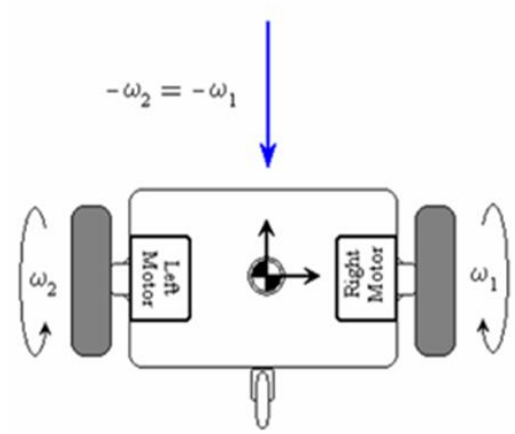


Forward Turn (1)



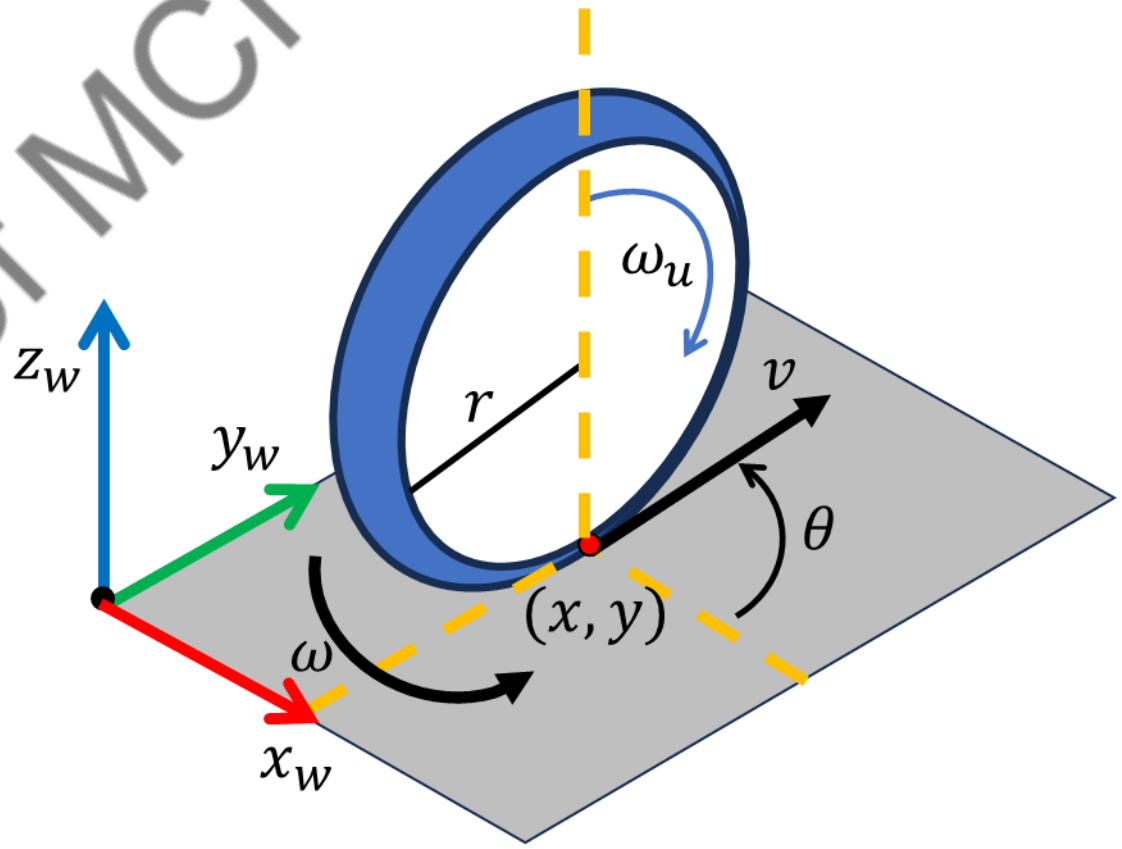
Forward Turn (2)





The Unicycle

- To derive the kinematics of a differential drive robot, it is instructive to first consider a simpler system: the unicycle.
- The state of the vehicle is defined as a three-element vector, $[x\ y\ \theta]$.
- The xy-position, specified in meters, and a vehicle heading angle, θ, specified in radians.
- Ignoring balancing concerns, there are two action variables $\omega_u \Rightarrow v, and\ \omega$, i.e., direct inputs to the system in the x_w, y_w plane.



Unicycle

The Unicycle

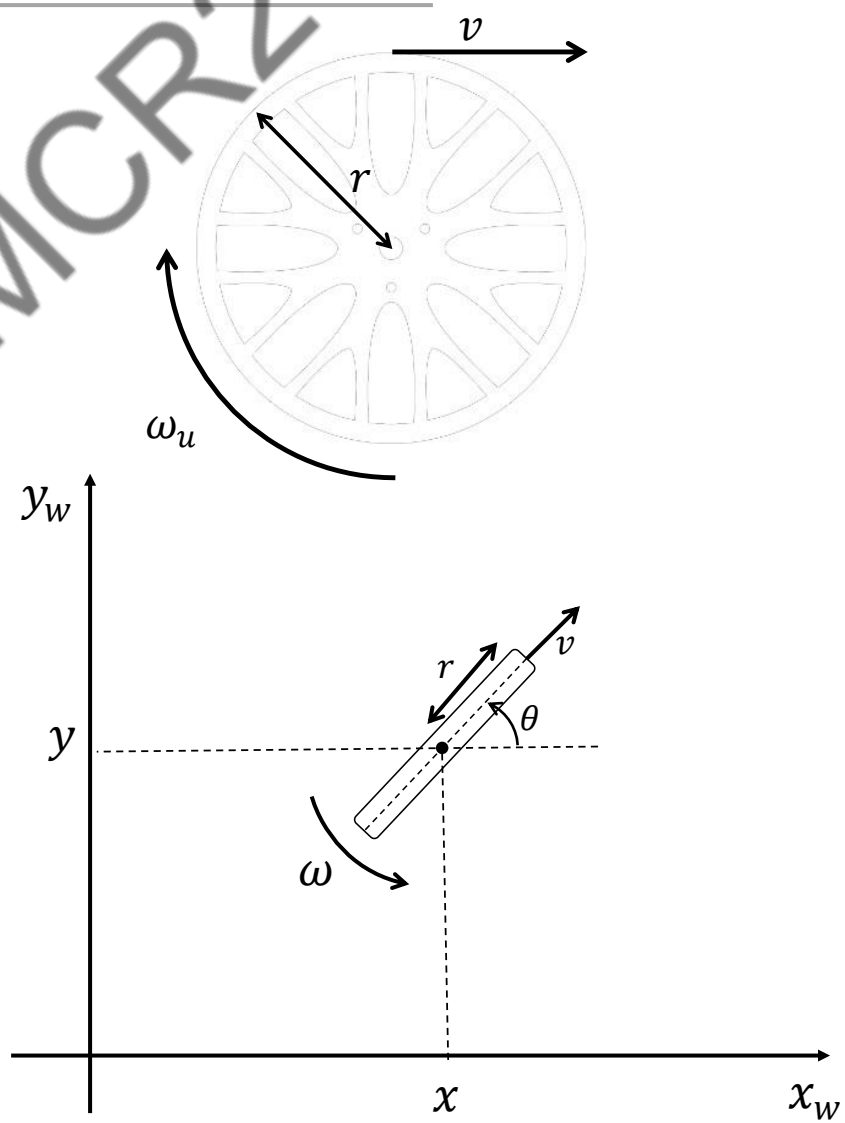
- The kinematic equations of the unicycles can be given as follows:

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

where $v = \omega_u \, r$ is the linear velocity of the unicycle, ω_u is the wheel angular velocity, r is wheel radius whereas the second one is the steering velocity denoted by ω.

- It can be seen, that velocity component in the direction perpendicular to the direction of movement is always zero this is known as the non-slip condition

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



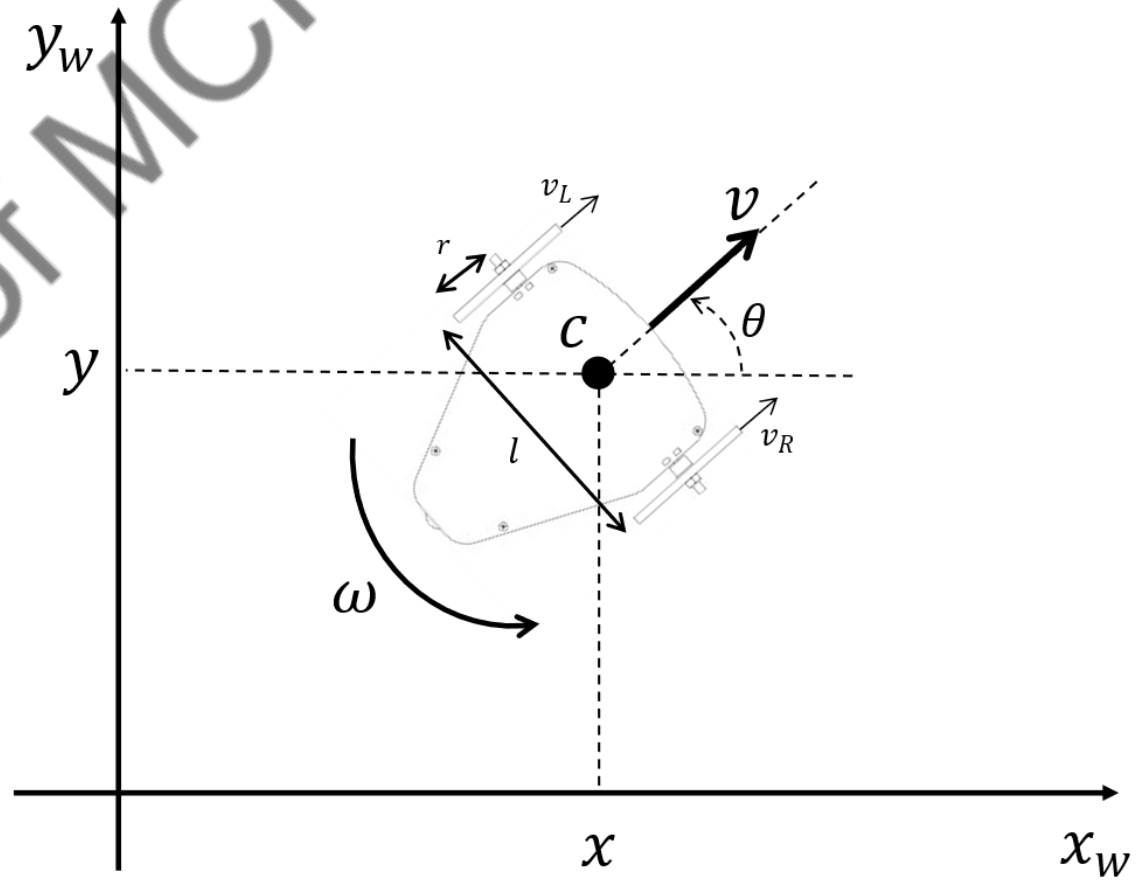
Unicycle side and top views

Differential Drive Kinematics

- For the case of a differential drive system, we can extend the idea of the Unicycle.
- To do this it is necessary to estimate the forward velocity v and the angular velocity ω.
- For this case, the resultant forward velocity v through C (centre of mass) may be reasoned as an average of the two forward wheel velocities given by

$$v = \left(\frac{v_R + v_L}{2}\right) = r\left(\frac{\omega_R + \omega_L}{2}\right)$$

where r is the radius of the wheel and ω_R, ω_L are the angular velocities of the left and right wheels, respectively.



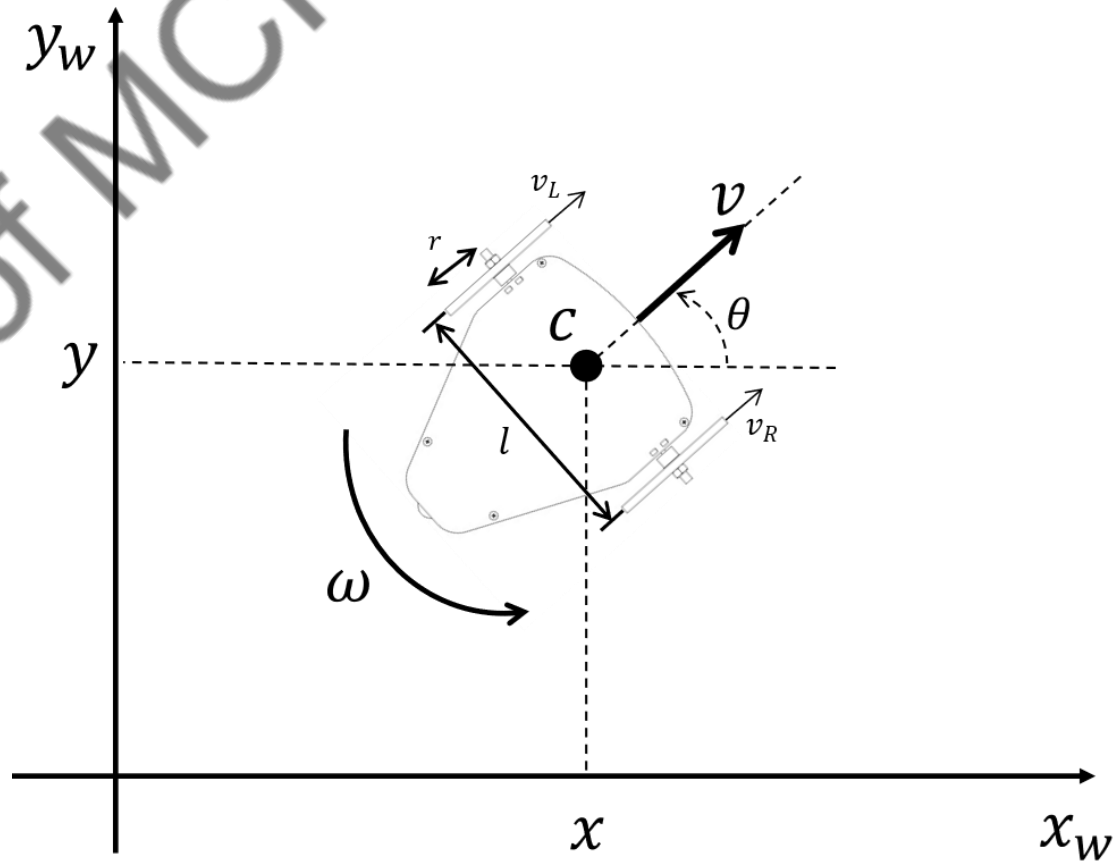
Differential Drive Robot
Puzzlebot ©.

Differential Drive Kinematics

- The resultant angular velocity ω (steering velocity), may also be reasoned as proportional to the difference between wheel velocities but inversely proportional to distance between the wheels, i.e.,

$$\omega = \left(\frac{v_R - v_L}{l}\right) = r \cdot \left(\frac{\omega_R - \omega_L}{l}\right)$$

where r is the radius of the wheel, l is the distance between wheels (wheelbase) and ω_R, ω_L are the angular velocities of the left and right wheels, respectively.



Differential Drive Robot
Puzzlebot ©.

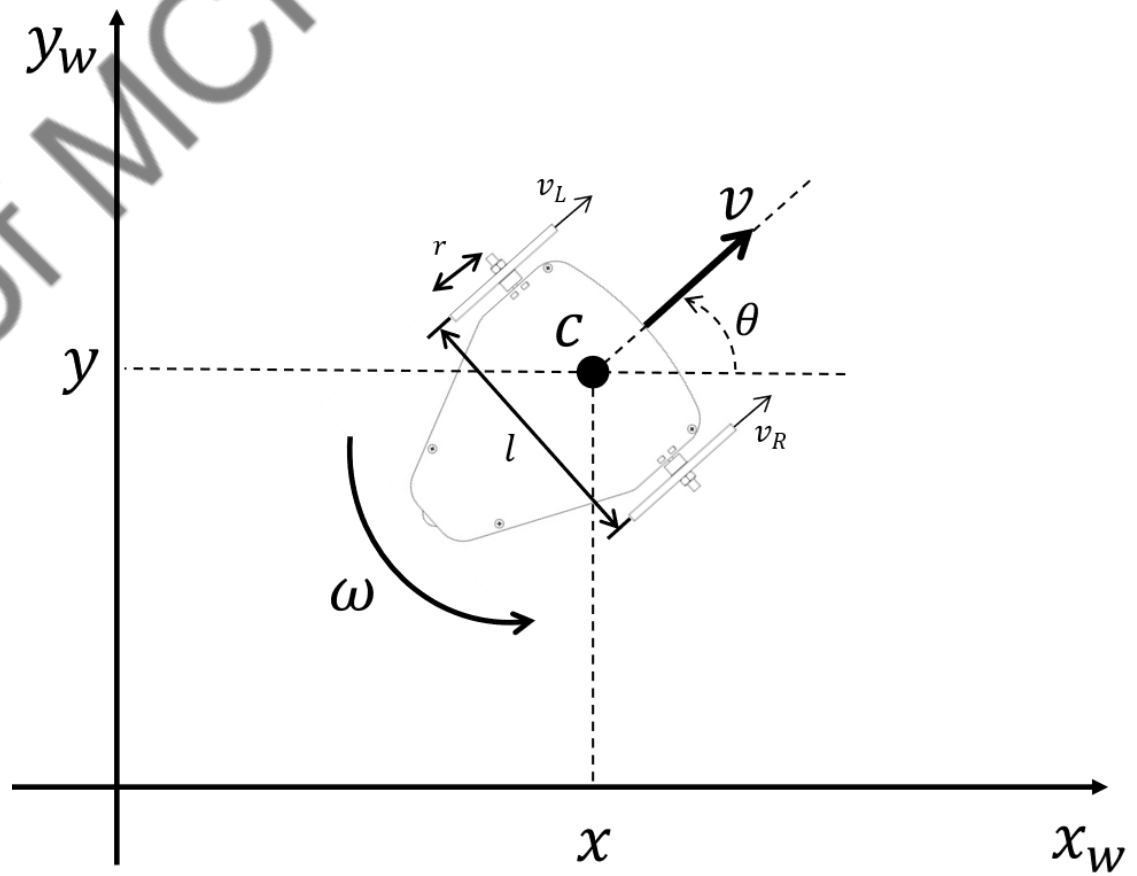
Differential Drive Kinematics

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

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

- Replacing v and ω we obtain

$$\begin{cases} \dot{x} = r\left(\frac{\omega_R + \omega_L}{2}\right) \cdot \cos\theta \\ \dot{y} = r\left(\frac{\omega_R + \omega_L}{2}\right) \cdot \sin\theta \\ \dot{\theta} = r\left(\frac{\omega_R - \omega_L}{l}\right) \end{cases}$$



Differential Drive Robot
Puzzlebot ©.

Differential Drive Kinematics

- In general, in the mobile robotic community, the state of a robot is denoted by $\boldsymbol{s}$, namely the *pose* or *posture*, for this case it consists of the robot position and orientation with respect to a frame of reference (world frame).

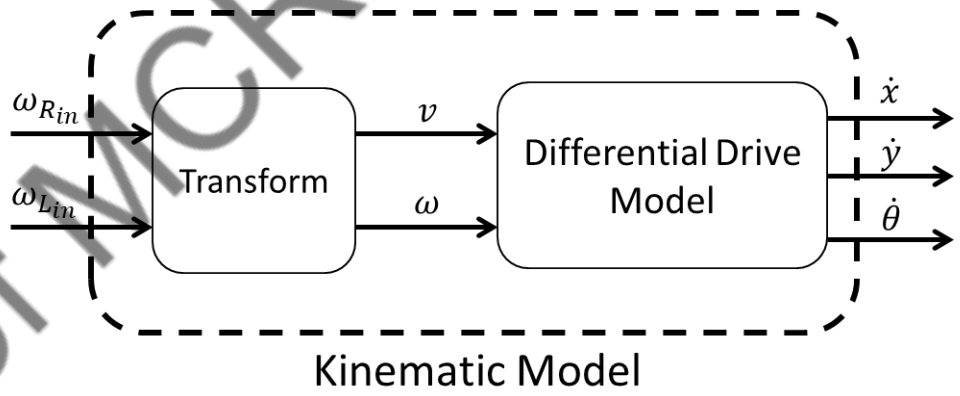
$$\mathbf{s} = [s_x \quad s_y \quad s_\theta]^T$$

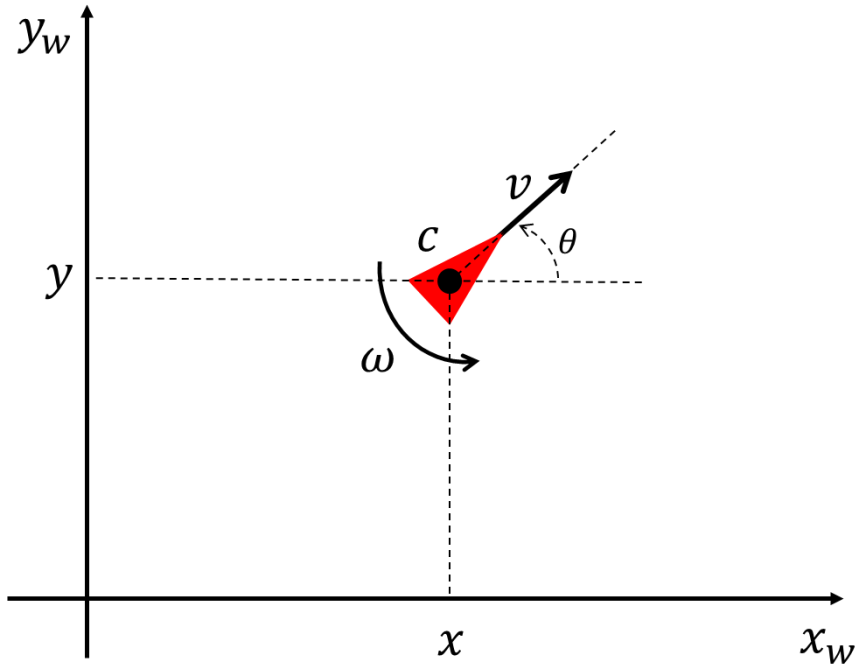
- The kinematic model of a differential drive robot can then be stated in terms of the robot *pose* ($\mathbf{s}$), as follows

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

- where, the inputs to the system $\mathbf{u} = [v \quad \omega]^T$, are given by

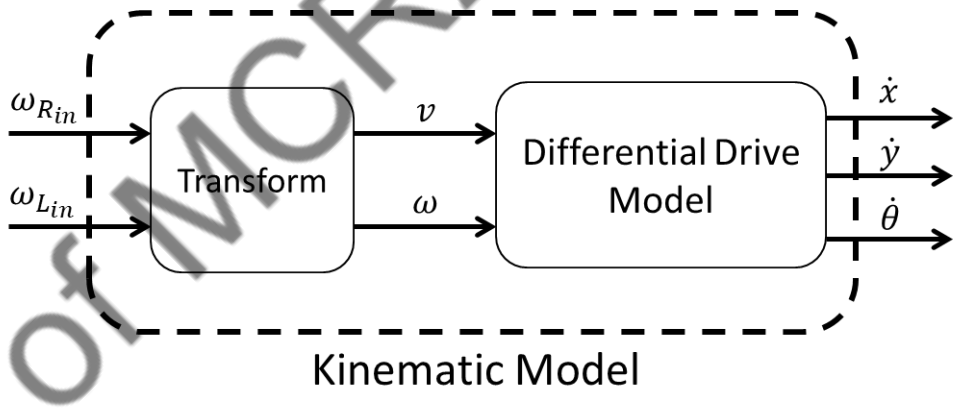
$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{l} & -\frac{r}{l} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix}$$



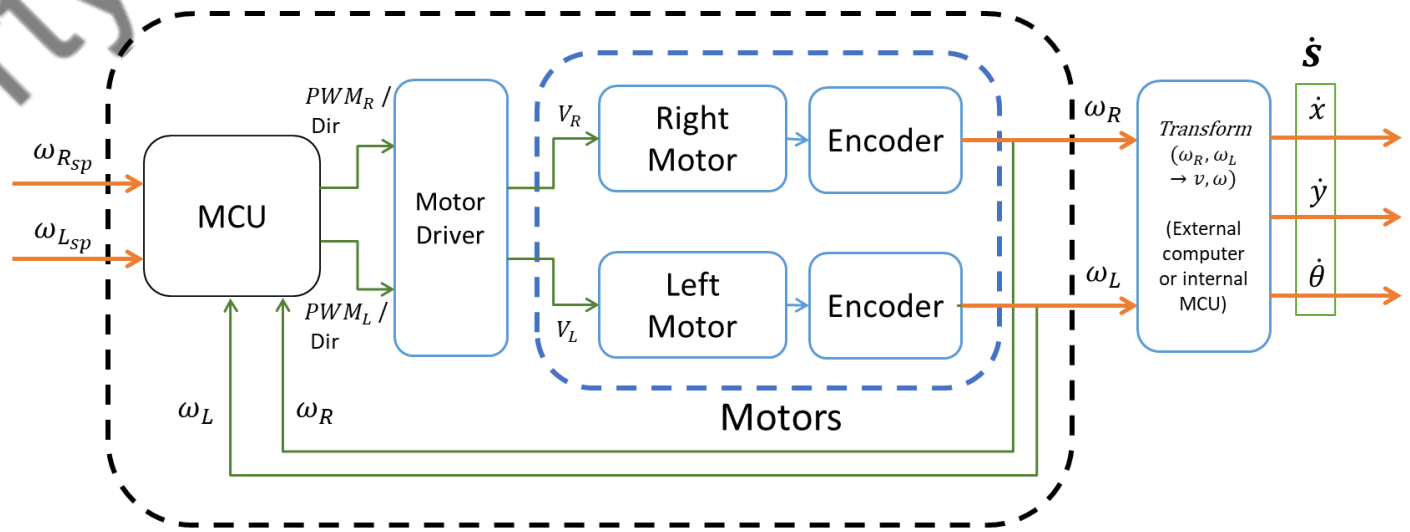


The Mobile Robot

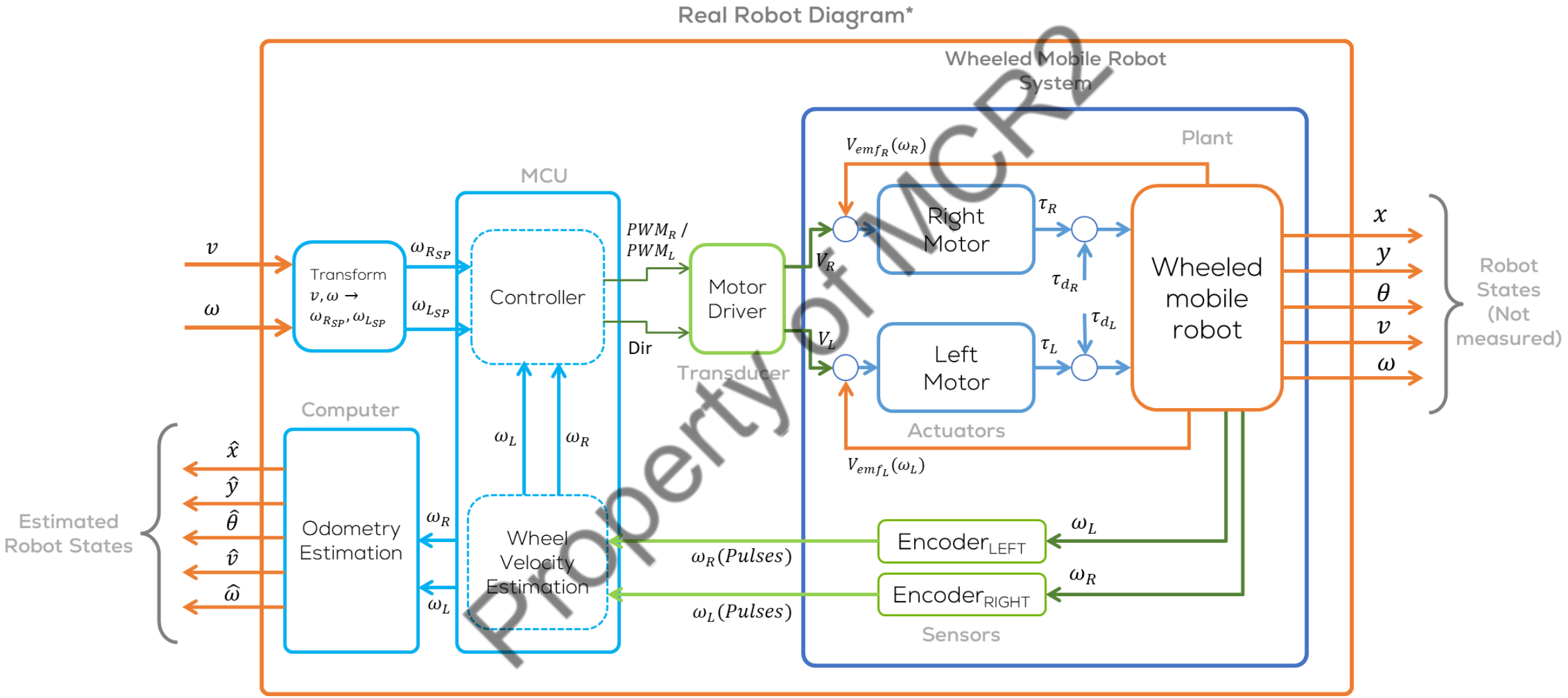
- The real robot, in comparison with the kinematic model, requires the usage of sensors and actuators to work.
- For the real robot, a closed control loop for each of the motors is required, to reach the required velocities, set by the user.
- In robotics this is called low level control. For the case of a wheeled mobile robot is a common practice to implement a PID control.
- It can be observed that the inputs and outputs are the same, but the inner loop controller and actuators will determine how close the Real Robot will resemble the Kinematic Model.



Kinematic Model



Real Robot Diagram*


Real Robot Diagram*
Wheeled Mobile Robot System
MCU
Plant
Transform
$v, \omega \rightarrow$
$\omega_{R_{SP}}, \omega_{L_{SP}}$
Controller
PWM_R /
PWM_L
Dir
Motor Driver
Transducer
V_R
V_L
Right Motor
Left Motor
τ_R
τ_L
τ_{d_R}
τ_{d_L}
Wheeled mobile robot
Actuators
$V_{emf_R}(\omega_R)$
$V_{emf_L}(\omega_L)$
x
y
θ
v
ω
Robot States (Not measured)
Computer
Odometry Estimation
Wheel Velocity Estimation
ω_L
ω_R
Estimated Robot States
$\hat{x}$
$\hat{y}$
$\hat{\theta}$
$\hat{v}$
$\hat{\omega}$
$\omega_R(Pulses)$
$\omega_L(Pulses)$
Encoder$_{LEFT}$
Encoder$_{RIGHT}$
Sensors
Property of MCR2