



JOHNS HOPKINS

WHITING SCHOOL  
of ENGINEERING

# Introduction to Robotics

Wheeled Vehicle Kinematics

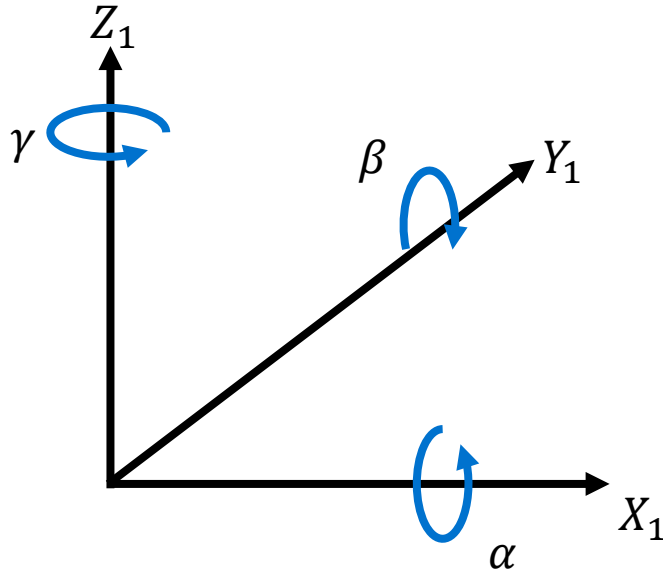
# Kinematics

---

- Kinematics is the study of motion
  - Kinematic model provides the equations of motion for a robotic system
  - Forces and torques that impact a robot do not play part in its kinematics
  - For unmanned systems, kinematics are at times referred to as kinematic constraints on the vehicle

# Definitions

- Degrees of Freedom (DOF)
  - Defined as number of independent dimensions in which motion a can be achieved
  - In free space, there are a maximum of 6 DOF of motion



# State Space

---

- A framework to represent the dynamic relationships between the critical characteristics (the states) of a system
- Provides framework for the first order differential relationships between inputs, outputs, and states – state equations
- Abstraction of properties enables vector representation – state vector
- If state equations are linear and time invariant (LTI) then space can be represented with matrices

# State Space (cont.)

- Provides a framework to describe the vehicle's kinematics

## Continuous Case

$\dot{X} = F(X, u)$  System Process Equation

$Y = h(X, u)$  Measurement Equation

## Discrete Case

$$X_{k+1} = F(X_k, u_k)$$

$$Y_k = h(X_k, u_k)$$

Where  $X_k, Y_k$  are the current state and output and  $X_{k+1}$  is the next state

Where  $X$  is the state vector  $[x_1, x_2, \dots, x_N]$

The input to the system is  $u$

$F(X, u)$  is the state transfer function

$Y$  is the output function (sensor measurement)

$h(X, u)$  is the output function of states and inputs

# State Space – Vehicles

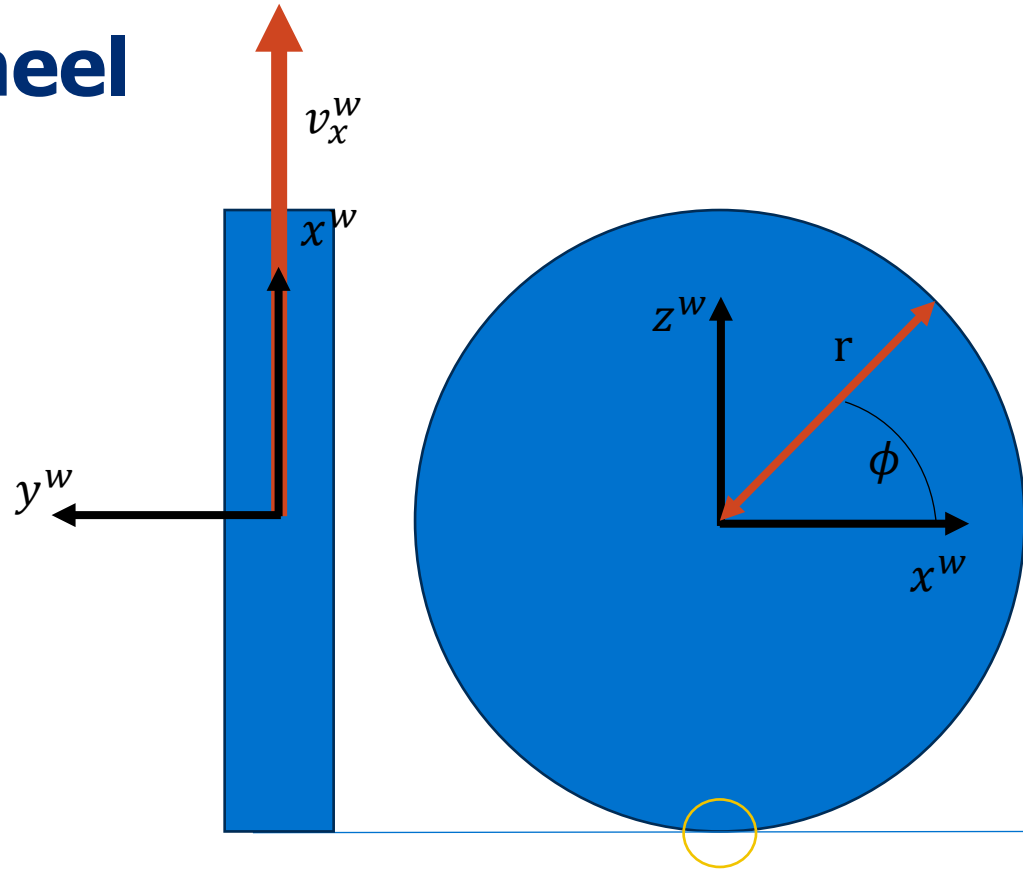
- First order of business is to choose the characteristics of the vehicle that will define its state – state variables
  - For a position-based vehicle the state variables will be the description of the vehicles pose (position/orientation), so choose them wisely
  - The state variables form the state vector
- Identify what can be measured on the vehicle to help with the calculation of state variables
- Finally, determine the equations of motion with respect to the state variables and the measured variables

# Properties of a Wheel

- Moves on a horizontal plane
- Point contact on wheel
- Wheel will not deform
- No slip, skid, or sliding
- No friction around contact point

$$v_x^w = r * \dot{\phi}$$

$$v_y^w = 0$$



# Unicycle Model

$$\text{State Vector} = X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

$$\text{Control} = u = \begin{bmatrix} \dot{\phi} \\ \omega \end{bmatrix}$$

$$v_x^w = v = r * \dot{\phi} \quad v_y^w = 0$$

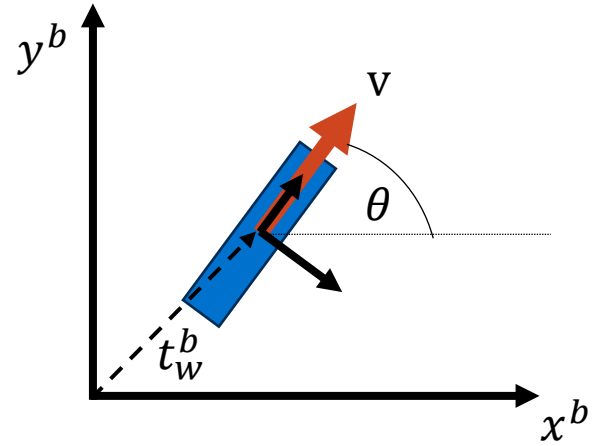
$$\dot{\theta} = \omega$$

$$v^b = R_w^b v^w$$

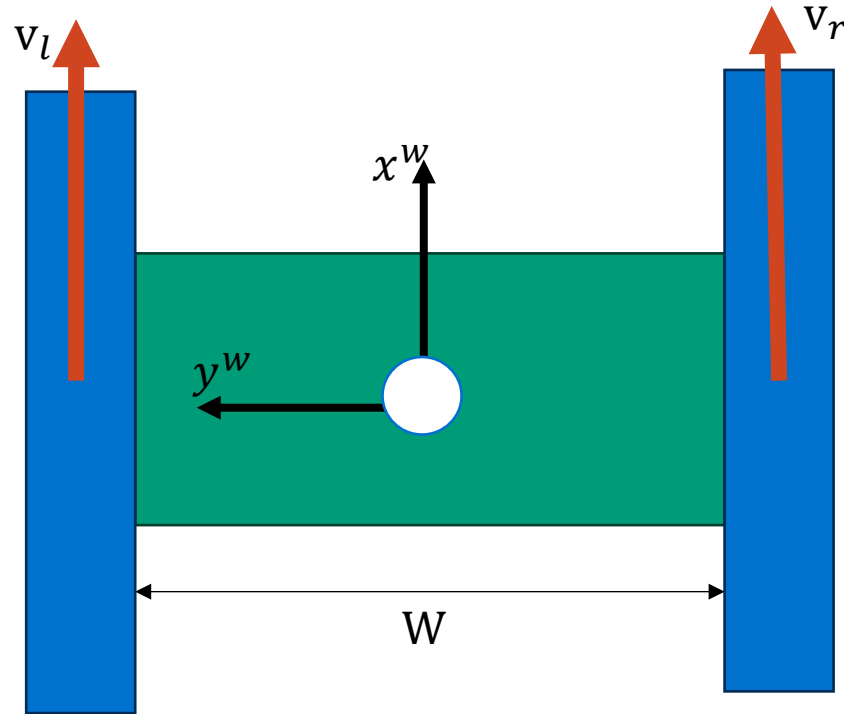
$r$  – Wheel radius  
 $\dot{\phi}$  – Wheel turn rate  
 $\omega$  – Yaw turn rate

$$v^b = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} v * \cos \theta \\ v * \sin \theta \end{bmatrix}$$

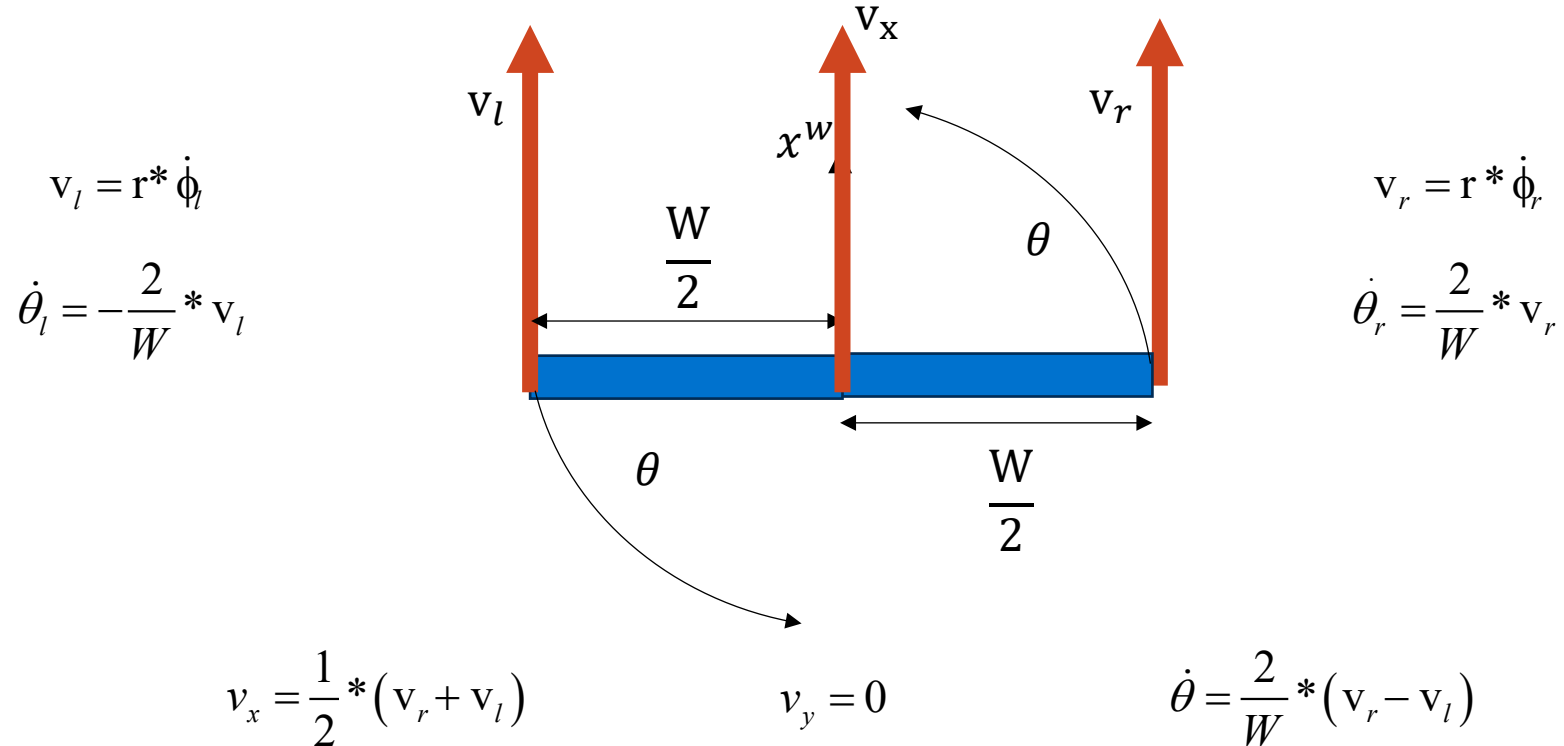
$$\dot{X} = \begin{bmatrix} v * \cos \theta \\ v * \sin \theta \\ \omega \end{bmatrix}$$



# Differential Drive (1)



# Differential Drive (2)



# Differential Drive (3)

$$\text{State Vector} = X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

$$\text{Control} = u = \begin{bmatrix} \dot{\phi}_r \\ \dot{\phi}_l \end{bmatrix}$$

$$v_L = r * \dot{\phi}_l$$

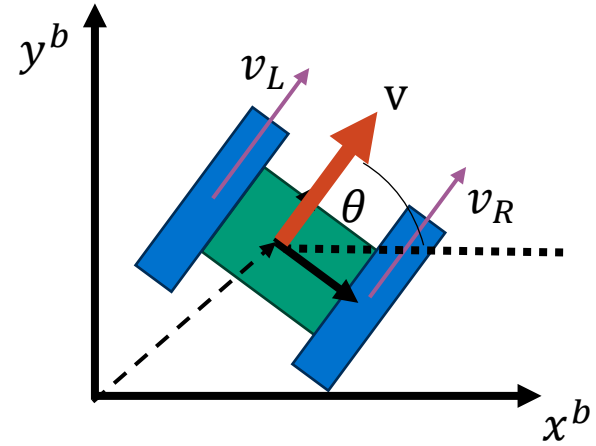
$$v_R = r * \dot{\phi}_r$$

$$v_x = \frac{1}{2} * (v_L + v_R)$$

$$\omega = \frac{2}{W} * (v_R - v_L)$$

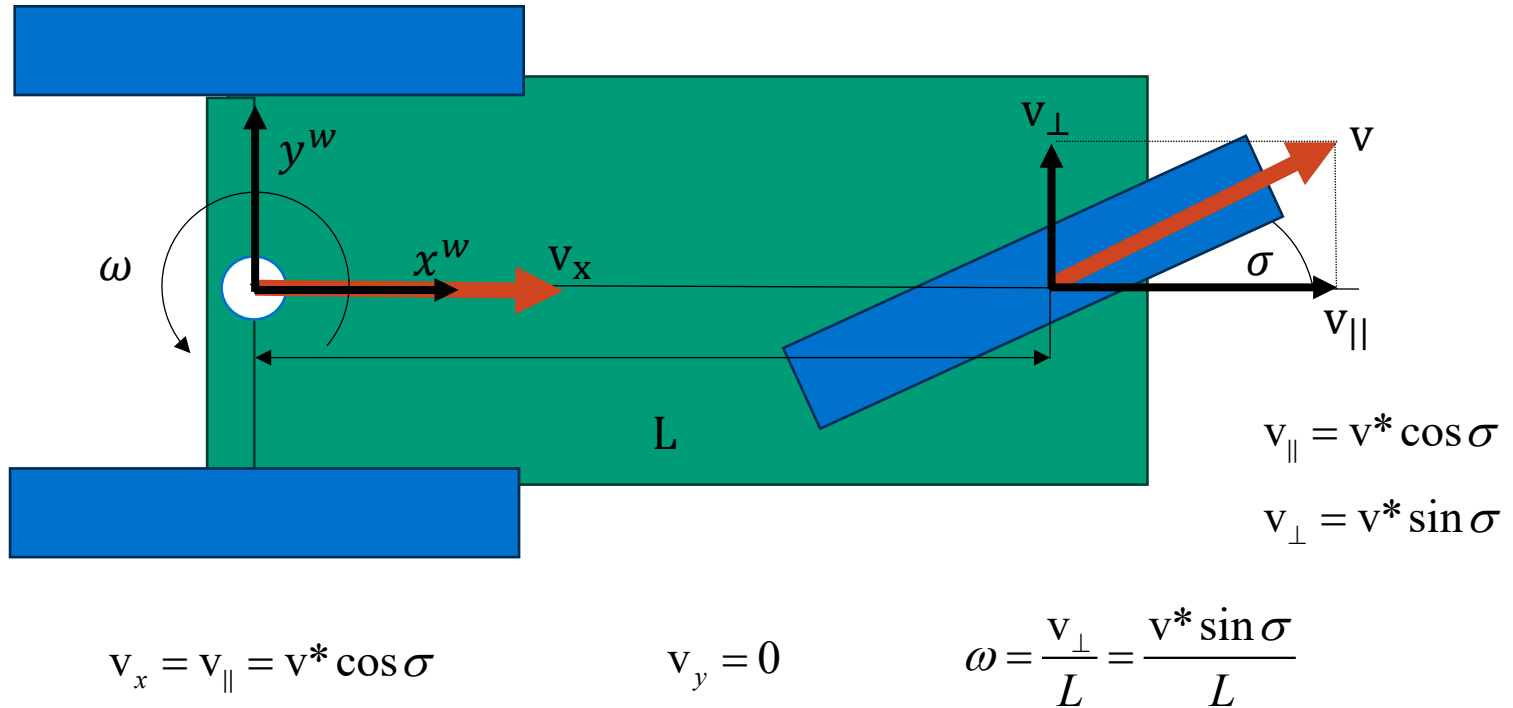
$r$  – Wheel radius  
 $W$  – Vehicle width  
 $\dot{\phi}_r$  – Right wheel turn rate  
 $\dot{\phi}_l$  – Left wheel turn rate

$$v^b = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} v_x \\ 0 \end{bmatrix}$$



$$\dot{X} = \begin{bmatrix} \frac{r}{2} (\dot{\phi}_r + \dot{\phi}_l) * \cos \theta \\ \frac{r}{2} (\dot{\phi}_r + \dot{\phi}_l) * \sin \theta \\ \frac{2r}{W} * (\dot{\phi}_r - \dot{\phi}_l) \end{bmatrix}$$

# Tricycle – Front Wheel Drive



# Tricycle - Front Wheel Drive

State Vector  $= X = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$

Control  $= u = \begin{bmatrix} \dot{\phi} \\ \sigma \end{bmatrix}$

$$v = r * \dot{\phi}$$

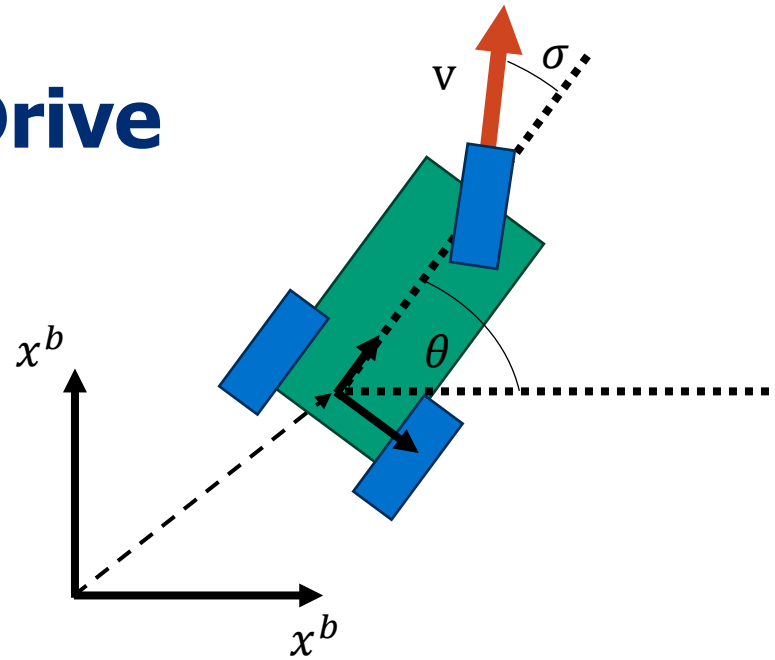
$$v_x = v^* \cos \sigma$$

$$\omega = \frac{v^* \sin \sigma}{L}$$

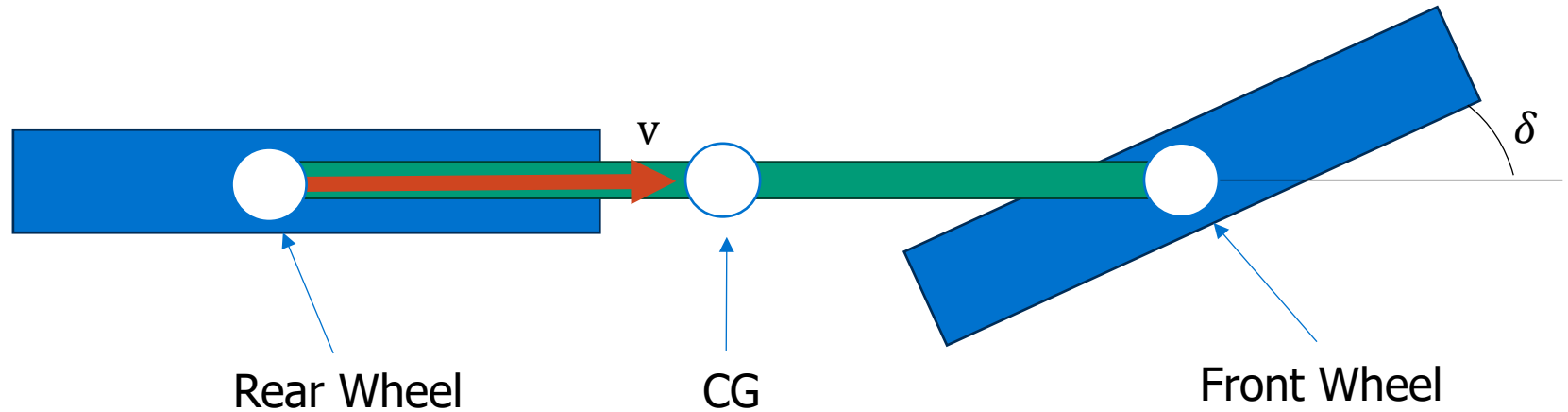
$r$  - Wheel radius  
 $L$  - Vehicle length  
 $\dot{\phi}$  - Front wheel turn rate  
 $\sigma$  - Steering angle

$$v^b = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} v_x \\ 0 \end{bmatrix}$$

$$\dot{X} = \begin{bmatrix} v^* \cos \sigma * \cos \theta \\ v^* \cos \sigma * \sin \theta \\ \frac{v^* \sin \sigma}{L} \end{bmatrix}$$



# Bicycle – Rear Wheel Drive



Multiple Choices for Frame of Reference

# Bicycle (1)

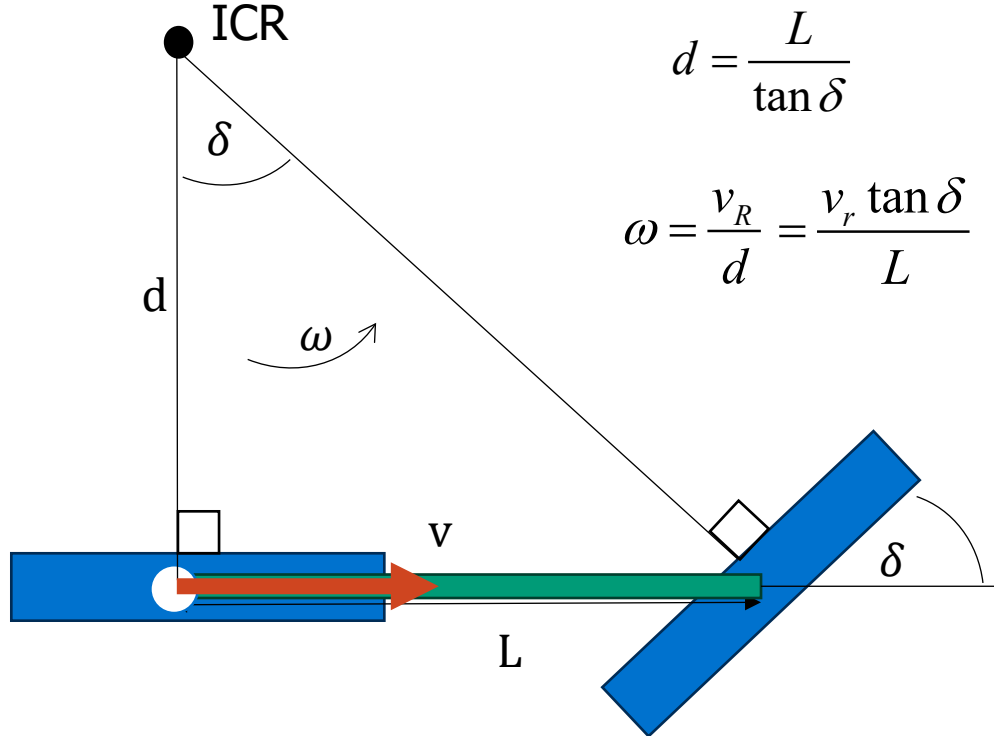
Reference Frame  
Rear Wheel

$$\dot{X} = \begin{bmatrix} v^* \cos \theta \\ v^* \sin \theta \\ \frac{v^* \tan \delta}{L} \end{bmatrix}$$

$$u = \begin{bmatrix} v_R \\ \delta \end{bmatrix}$$

$L$  – Vehicle length  
 $v_R$  – Rear wheel speed  
 $\delta$  – Steering angle

Instantaneous center of rotation (ICR)



# Bicycle (2)

Reference Frame  
Front Wheel

$$\dot{X} = \begin{bmatrix} \frac{v_R}{\cos \delta} * \cos \theta \\ \frac{v_R}{\cos \delta} * \sin \theta \\ \frac{v_R * \tan \delta}{L} \end{bmatrix}$$

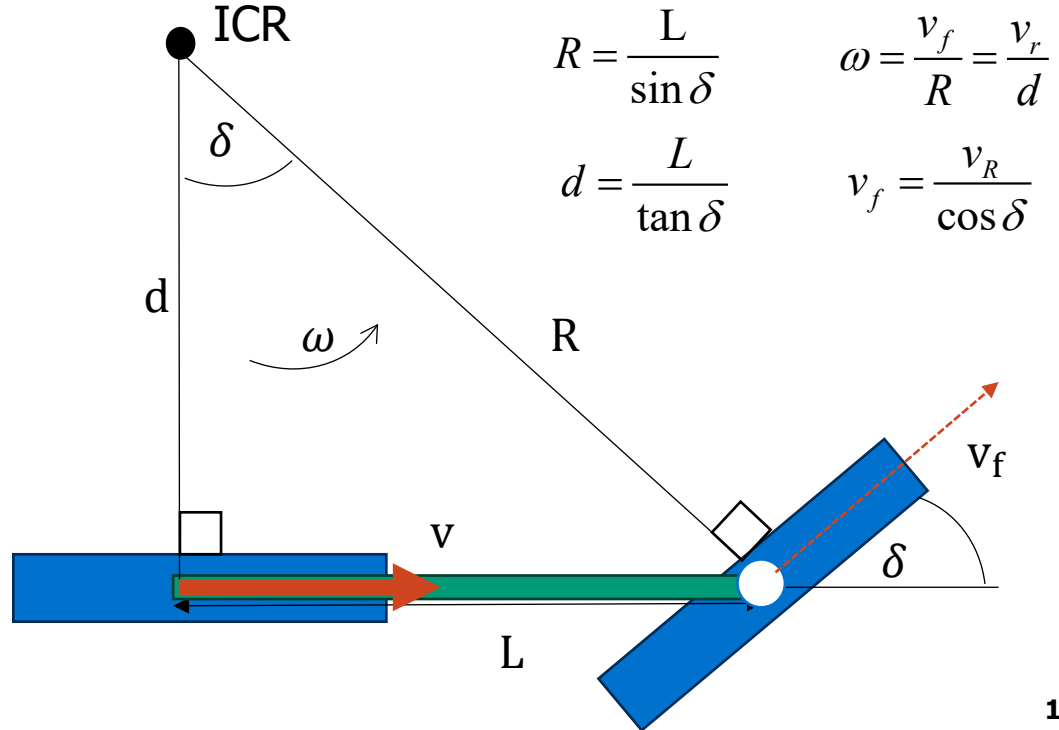
$L$  – Vehicle length

$v_R$  – Rear wheel speed

$\delta$  – Steering angle

$$u = \begin{bmatrix} v_R \\ \delta \end{bmatrix}$$

Instantaneous center of rotation (ICR)



# Bicycle (3)

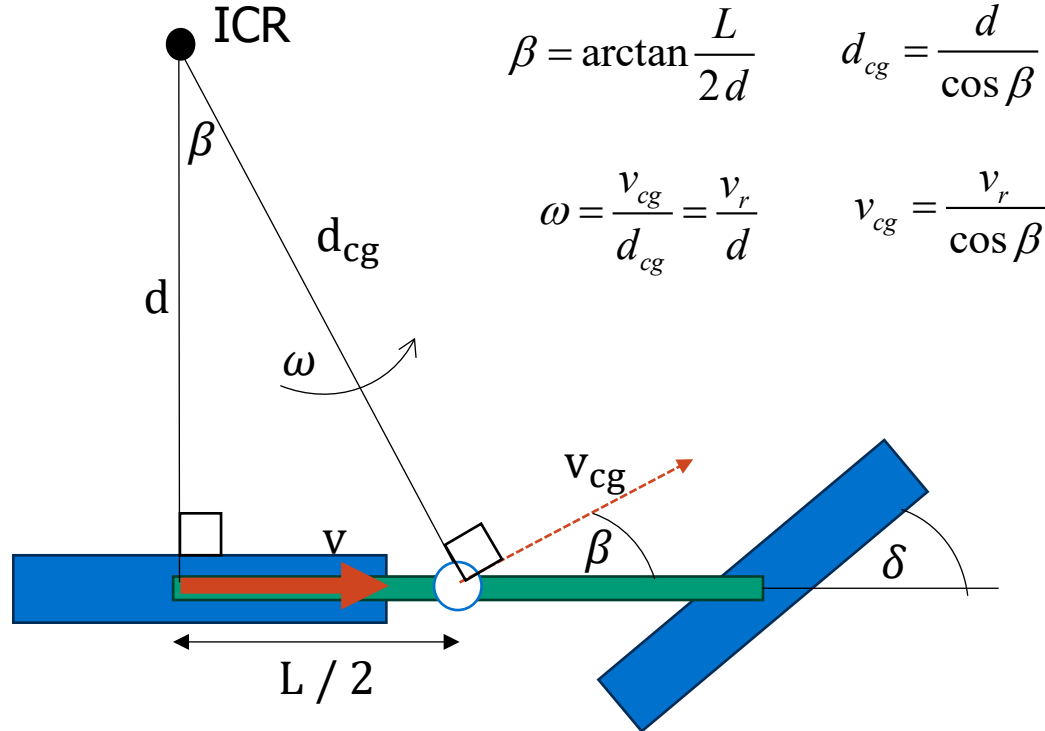
Reference Frame  
Center of Gravity

$$\dot{X} = \begin{bmatrix} v * \cos \sigma * \cos \theta \\ v * \cos \sigma * \sin \theta \\ \frac{v * \tan \delta}{L} \end{bmatrix}$$

$$u = \begin{bmatrix} v_R \\ \delta \end{bmatrix}$$

$L$  – Vehicle length  
 $v_R$  – Rear wheel speed  
 $\delta$  – Steering angle

Instantaneous center of rotation (ICR)





# JOHNS HOPKINS

WHITING SCHOOL  
*of* ENGINEERING

© The Johns Hopkins University 2025, All Rights Reserved.



JOHNS HOPKINS

WHITING SCHOOL  
of ENGINEERING

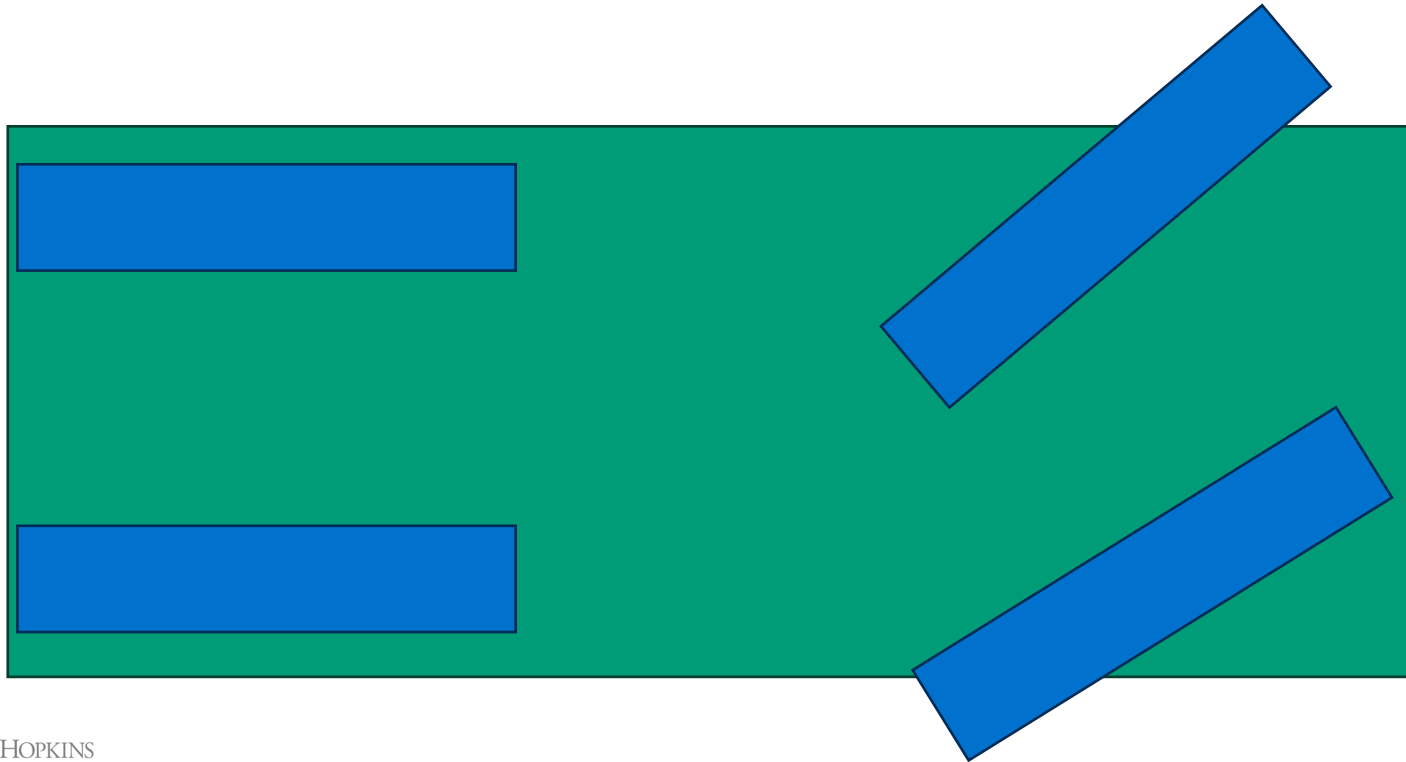
# Introduction to Robotics

Wheeled Vehicle Kinematics

4 Wheeled Vehicles

# Ackerman Steering

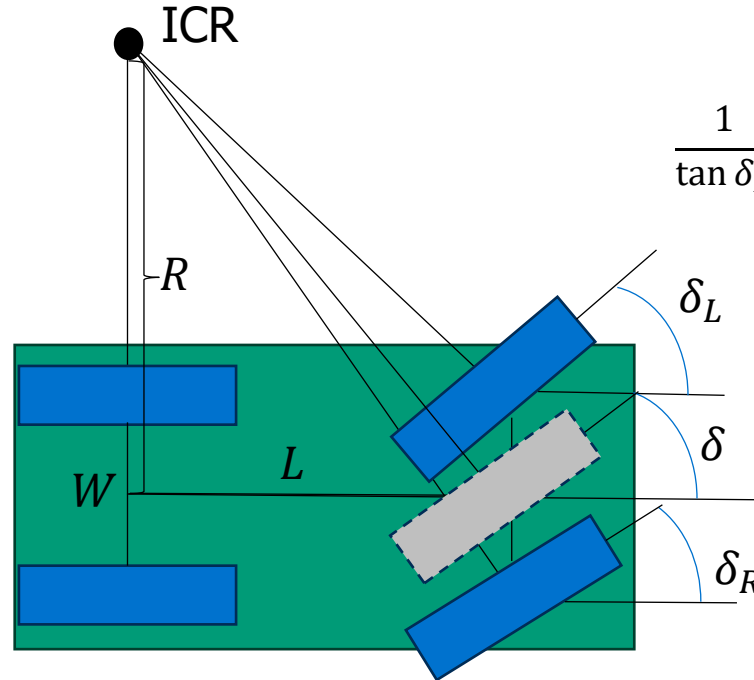
---



# Ackerman Steering (cont.)

$$\dot{X} = \begin{bmatrix} v * \cos \theta \\ v * \sin \theta \\ \frac{v * \tan \delta}{L} \end{bmatrix}$$

$L$  – Vehicle length  
 $W$  – Vehicle width  
 $v$  – Rear wheel speed  
 $\delta$  – Effective steering angle  
 $\delta_l$  – Left wheel angle  
 $\delta_r$  – Right wheel angle



$$\tan \delta = \frac{L}{R} \quad \tan \delta_L = \frac{L}{R - \frac{W}{2}} \quad \tan \delta_R = \frac{L}{R + \frac{W}{2}}$$

$$\frac{1}{\tan \delta_R} - \frac{1}{\tan \delta_L} = \frac{R + \frac{W}{2}}{L} - \frac{R - \frac{W}{2}}{L} = \frac{W}{L}$$

$$\cot \delta_L = \cot \delta - \frac{W}{2L}$$

$$\cot \delta_R = \cot \delta + \frac{W}{2L}$$

$$\delta_L = \text{atan}\left(\frac{2L \sin \delta}{2L \cos \delta - W \sin \delta}\right)$$

$$\delta_R = \text{atan}\left(\frac{2L \sin \delta}{2L \cos \delta + W \sin \delta}\right)$$

# Mecanum Wheel (1)

$$V_x = V_x^{\text{wheel}} + V_x^{\text{roller}}$$

$$V_y = V_y^{\text{wheel}} + V_y^{\text{roller}}$$

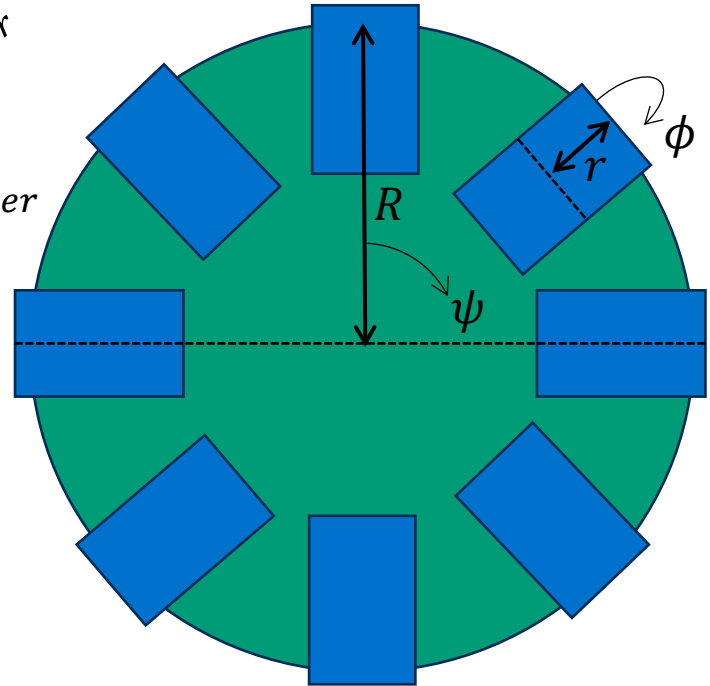
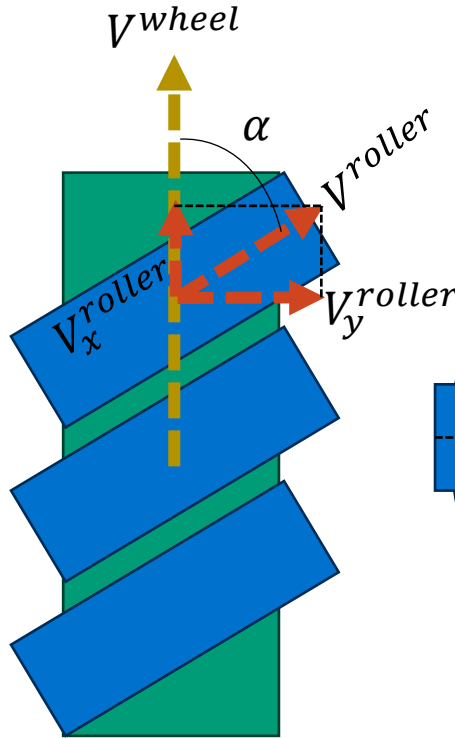
$$V_x^{\text{wheel}} = R\psi_i \quad V_y^{\text{wheel}} = 0$$

$$V_x^{\text{roller}} = r\phi \cos \alpha$$

$$V_y^{\text{roller}} = r\phi \sin \alpha$$

$$V_x = R\psi + r\phi \cos \alpha$$

$$V_y = -r\phi \sin \alpha$$



# Mecanum Wheel (2)

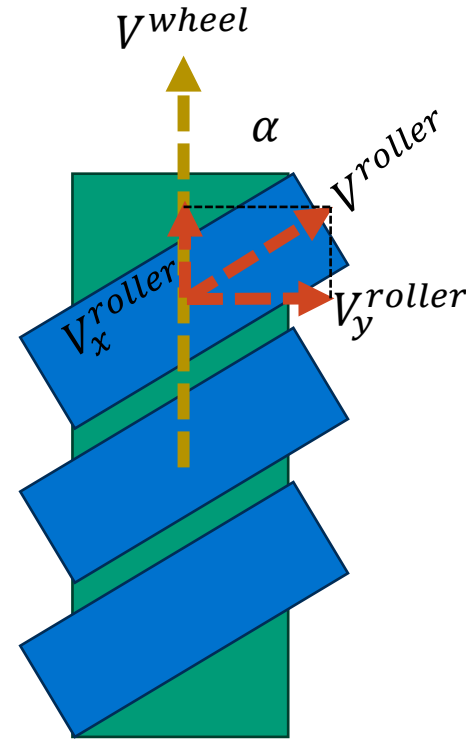
$$V_x = R\psi + r\phi \cos \alpha \quad V_y = -r\phi \sin \alpha$$

Therefore, we can remove  $\phi_i$  by substitution

$$\frac{V_y}{\sin \alpha} = -r\phi$$

Leaving us with the following relationship

$$V_x = R\psi - \frac{V_y}{\sin \alpha} \cos \alpha \rightarrow V_x + \frac{V_y}{\tan \alpha} = R\psi$$

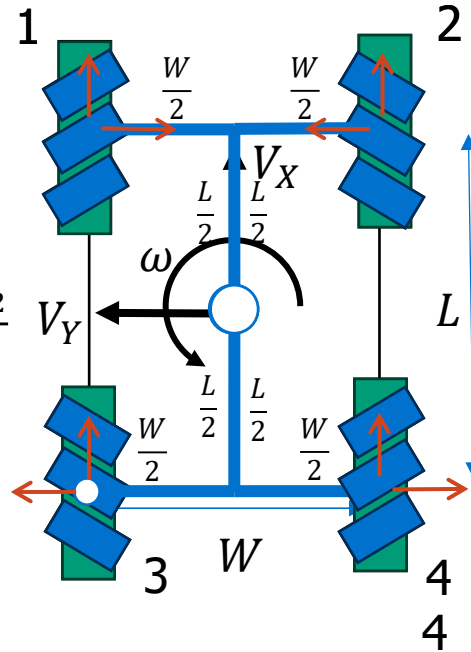


# Mecanum Wheel – Forward Kinematics (1)

$$V_X = V_{X1} + V_{X2} + V_{X3} + V_{X4}$$

$$V_Y = -V_{Y1} + V_{Y2} + V_{Y3} - V_{Y4}$$

$$\omega_z = -\frac{2V_{X1}}{W} - \frac{2V_{Y1}}{L} + \frac{2V_{X2}}{W} + \frac{2V_{Y2}}{L} - \frac{2V_{X3}}{W} - \frac{2V_{Y3}}{L} + \frac{2V_{X4}}{W} + \frac{2V_{Y4}}{L}$$



$$\omega_{z1} = \frac{-2}{W} V_{x1} + \frac{-2}{L} V_{y1}$$

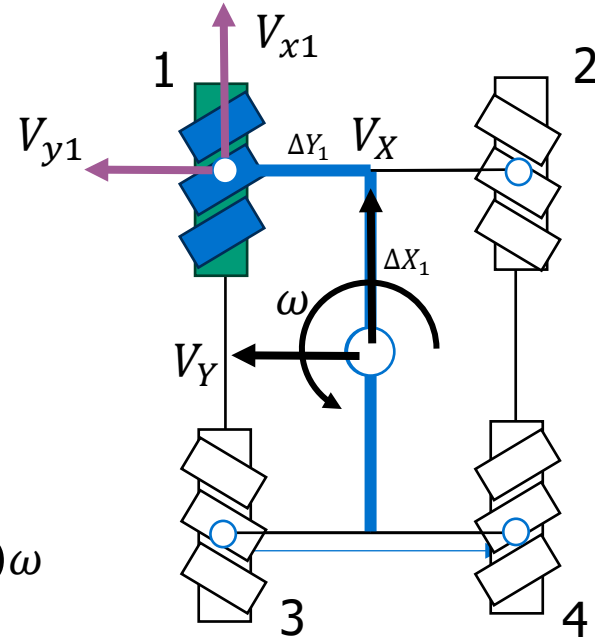
$$\omega_{z2} = \frac{2}{W} V_{x2} + \frac{2}{L} V_{y2}$$

$$\omega_{z3} = \frac{-2}{W} V_{x3} + \frac{-2}{L} V_{y3}$$

$$\omega_{z4} = \frac{2}{W} V_{x4} + \frac{2}{L} V_{y4}$$

# Mecanum Wheel – Inverse Kinematics (1)

$$\begin{aligned}
 V_{xi} &= V_X + \Delta Y_i * \omega \\
 V_{yi} &= V_Y + \Delta X_i * \omega \\
 R\psi_i &= V_{xi} + \frac{V_{yi}}{\tan \alpha} \\
 &= V_X + \frac{V_Y}{\tan \alpha_i} + \left( \Delta Y + \frac{\Delta X_i}{\tan \alpha_i} \right) \omega \\
 &= V_X + \cot \alpha_i V_Y + (\Delta Y + \Delta X_i \cot \alpha_i) \omega
 \end{aligned}$$



$$\begin{aligned}
 \Delta X_1 &= \frac{L}{2}, \quad \Delta X_2 = \frac{L}{2}, & \Delta Y_1 &= \frac{-W}{2}, \quad \Delta Y_2 = \frac{W}{2}, & \alpha_1 &= \alpha, \quad \alpha_2 = -\alpha, \\
 \Delta X_3 &= \frac{-L}{2}, \quad \Delta X_4 = \frac{-L}{2}, & \Delta Y_3 &= \frac{-W}{2}, \quad \Delta Y_4 = \frac{W}{2}, & \alpha_3 &= -\alpha, \quad \alpha_4 = \alpha
 \end{aligned}$$

# Mecanum Wheel – Inverse Kinematics (2)

$$A = \cot \alpha \quad B = \left( \frac{W}{2} + \frac{L}{2} \cot \alpha \right) \quad V_{y1}$$

$$\begin{aligned} R\psi_i &= V_X + \cot \alpha_i V_Y + (\Delta Y + \Delta X_i \cot \alpha_i) \omega \\ &= V_X \pm AV_Y \pm B\omega \end{aligned}$$

$$R\psi_1 = V_X + AV_Y - B\omega$$

$$R\psi_2 = V_X - AV_Y + B\omega$$

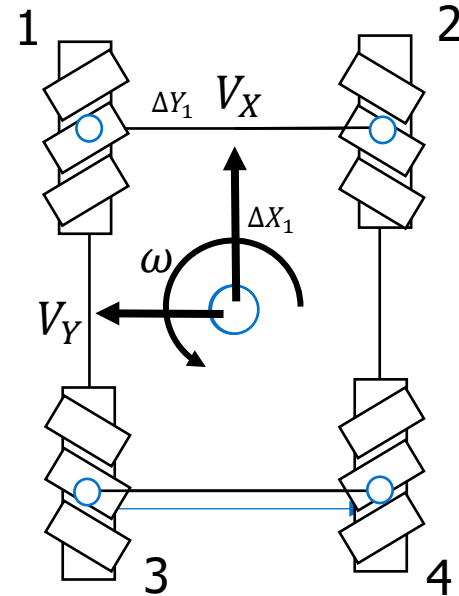
$$R\psi_3 = V_X - AV_Y - B\omega$$

$$R\psi_4 = V_X + AV_Y + B\omega$$

$$\begin{aligned} \Delta X_1 &= \frac{L}{2}, \quad \Delta X_2 = \frac{L}{2}, \\ \Delta X_3 &= \frac{-L}{2}, \quad \Delta X_4 = \frac{-L}{2} \end{aligned}$$

$$\begin{aligned} \Delta Y_1 &= \frac{-W}{2}, \quad \Delta Y_2 = \frac{W}{2}, \\ \Delta Y_3 &= \frac{-W}{2}, \quad \Delta Y_4 = \frac{W}{2} \end{aligned}$$

$$\begin{aligned} \alpha_1 &= \alpha, \quad \alpha_2 = -\alpha, \\ \alpha_3 &= -\alpha, \quad \alpha_4 = \alpha \end{aligned}$$



# Mecanum Wheel – Inverse Kinematics (3)

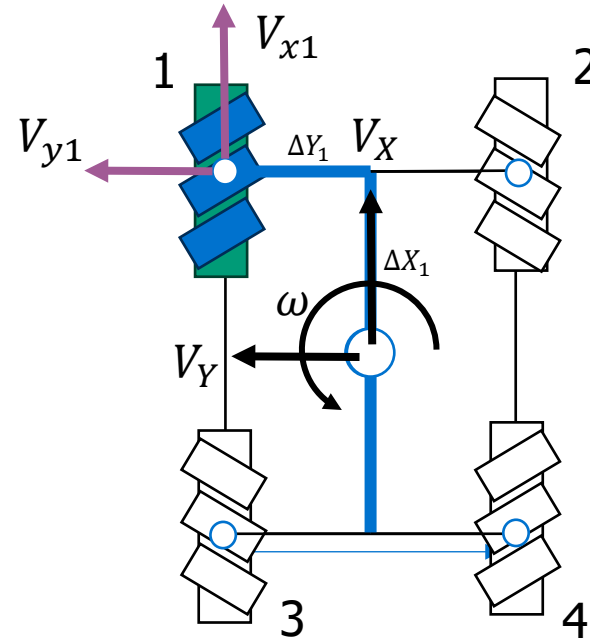
$$A = \cot \alpha \quad B = \left( \frac{W}{2} + \frac{L}{2} \cot \alpha \right)$$

$$R\psi_i = V_X + \cot \alpha_i V_Y + (\Delta Y + \Delta X_i \cot \alpha_i) \omega$$

Inverse Kinematic Equations

$$\Psi = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & A & -B \\ 1 & -A & B \\ 1 & -A & -B \\ 1 & A & B \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ \omega \end{bmatrix}$$

$$\begin{aligned} \Delta X_1 &= \frac{L}{2}, \Delta X_2 = \frac{L}{2}, & \Delta Y_1 &= \frac{-W}{2}, \Delta Y_2 = \frac{W}{2}, & \alpha_1 &= \alpha, \alpha_2 = -\alpha, \\ \Delta X_3 &= \frac{-L}{2}, \Delta X_4 = \frac{-L}{2}, & \Delta Y_3 &= \frac{-W}{2}, \Delta Y_4 = \frac{W}{2}, & \alpha_3 &= -\alpha, \alpha_4 = \alpha \end{aligned}$$



# Mecanum Wheel – Forward Kinematics (2)

To get the forward kinematics in terms of  $\Psi$  instead of  $V_{xi}, V_{yi}$  we invert the Jacobian  $J$

$$\Psi = \frac{1}{R} J V$$

$$V = \begin{bmatrix} V_X \\ V_Y \\ \omega_z \end{bmatrix} = R (J^T J)^{-1} J^T \Psi$$

$$J^T J = \begin{bmatrix} 1 & 1 & 1 & 1 \\ A & -A & -A & A \\ -B & B & -B & B \end{bmatrix} \begin{bmatrix} 1 & A & -B \\ 1 & -A & B \\ 1 & -A & -B \\ 1 & A & B \end{bmatrix}$$

$$J^T J = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4A^2 & 0 \\ 0 & 0 & 4B^2 \end{bmatrix} \quad (J^T J)^{-1} = \frac{1}{4} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/A^2 & 0 \\ 0 & 0 & 1/B^2 \end{bmatrix}$$

$$(J^T J)^{-1} J = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1/A & -1/A & -1/A & 1/A \\ -1/B & 1/B & -1/B & 1/B \end{bmatrix}$$

$$A = \cot \alpha \quad B = \left( \frac{W}{2} + \frac{L}{2} \cot \alpha \right)$$

# Mecanum Wheel – Forward Kinematics (3)

To get the forward kinematics in terms of  $\Psi$  instead of  $V_{xi}, V_{yi}$  we invert the Jacobian  $J$

$$\Psi = \frac{1}{R} J V$$
$$V = \begin{bmatrix} V_X \\ V_Y \\ \omega_Z \end{bmatrix} = R (J^T J)^{-1} J^T \Psi$$

Inverse Kinematic Equations

$$\Psi = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \psi_3 \\ \psi_4 \end{bmatrix} = \frac{1}{R} \begin{bmatrix} 1 & A & -B \\ 1 & -A & B \\ 1 & -A & -B \\ 1 & A & B \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ \omega \end{bmatrix}$$

Forward Kinematic Equations

$$V = \frac{R}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1/A & -1/A & -1/A & 1/A \\ -1/B & 1/B & -1/B & 1/B \end{bmatrix}$$

$$A = \cot \alpha \quad B = \left( \frac{W}{2} + \frac{L}{2} \cot \alpha \right)$$



JOHNS HOPKINS

WHITING SCHOOL  
of ENGINEERING

# Introduction to Robotics

Setting up a ROS2 Project

# Creating a ROS2 Package

Create a workspace directory and source directory and enter it

```
mkdir -p ~/dev_ws/src  
cd ~/dev_ws/src
```

Initialize a package (this auto-populates a folder with everything you need)

C++/cmake

```
ros2 pkg create --build-type ament_cmake <package_name>
```

Python

```
ros2 pkg create --build-type ament_python <package_name>
```

You can also define the name of the node you want to create

```
ros2 pkg create --build-type ament_python --node-name my_node my_package
```

# Building a ROS2 Package

Enter your workspace folder and build everything in the workspace

```
cd ~/dev_ws  
colcon build
```

You can also build a specific package in the workspace

```
colcon build --packages-select my_package
```

To utilize the files you need add them to your path using the following command

```
source install/setup.bash
```

# Package.xml

- The Package.xml file provides information on the packages provenance
- It also defines the packages dependencies for COLCON
- It primarily handles the C++ dependencies for CMAKE

```
<?xml version="1.0"?> <?xml-model
href="http://download.ros.org/schema/package_format3.
xsd"
schematypens="http://www.w3.org/2001/XMLSchema"?>
<package format="3">
  <name>my_package</name>
  <version>0.0.0</version>
  <description>TODO: Package description</description>
  <maintainer email="user@todo.todo">user</maintainer>
  <license>TODO: License declaration</license>
  <buildtool_depend>ament_python</buildtool_depend>
  <test_depend>ament_copyright</test_depend>
  <test_depend>ament_flake8</test_depend>
  <test_depend>ament_pep257</test_depend>
  <test_depend>python3-pytest</test_depend>
  <export>
    <build_type>ament_python</build_type>
  </export>
</package>
```

# setup.py

- The setup.py file provides the information on the python code being developed in the package
- To add commands that can be executed via `ros2 run` you must define them as entry\_points
- To add files that will be installed with your package (e.g. robot models, launch files, configuration files). They must be added to the data\_files list.

```
from setuptools import setup
package_name = 'my_py_pkg'
setup( name=package_name,
      version='0.0.0', packages=[package_name],
      data_files=[
        ('share/ament_index/resource_index/packages',
         ['resource/' + package_name]),
        ('share/' + package_name,
         ['package.xml']), ],
      install_requires=['setuptools'],
      zip_safe=True, maintainer='TODO',
      maintainer_email='TODO',
      description='TODO: Package description',
      license='TODO: License declaration',
      tests_require=['pytest'],
      entry_points={ 'console_scripts': [
        'my_py_pkg.my_node:main' ],
        },
      )
```

# Creating a Publisher Node

wget  
[https://raw.githubusercontent.com/ros2/examples/master/rclpy/topics/minimal\\_subscriber/examples\\_rclpy\\_minimal\\_subscriber/subscriber\\_member\\_function.py](https://raw.githubusercontent.com/ros2/examples/master/rclpy/topics/minimal_subscriber/examples_rclpy_minimal_subscriber/subscriber_member_function.py)

Creates a publisher for this node, defining one topic that the node publishes to and its type

Creates a timer that will periodically execute the callback function

Creates a new message object

Publishes the message to /topic

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String

class MinimalPublisher(Node):
    def __init__(self):
        super().__init__('minimal_publisher')
        self.publisher_ = self.create_publisher(String, 'topic', 10)
        timer_period = 0.5 # seconds
        self.timer = self.create_timer(timer_period, self.timer_callback)
        self.i = 0

    def timer_callback(self):
        msg = String()
        msg.data = 'Hello World: %d' % self.i
        self.publisher_.publish(msg)
        self.get_logger().info('Publishing: "%s"' % msg.data)
        self.i += 1
```

# Creating a Publisher Node (cont.)

wget  
[https://raw.githubusercontent.com/ros2/examples/master/rclpy/topics/minimal\\_subscriber/examples\\_rclpy\\_minimal\\_subscriber/subscriber\\_member\\_function.py](https://raw.githubusercontent.com/ros2/examples/master/rclpy/topics/minimal_subscriber/examples_rclpy_minimal_subscriber/subscriber_member_function.py)

Initialize the ROS2 environment

Creates the node object

Trigger the node to execute all its callbacks, publishing any pending messages and pulling messages from topics it is subscribed to

```
def main(args=None):
    → rclpy.init(args=args)
    → minimal_publisher = MinimalPublisher()
    → rclpy.spin(minimal_publisher)
      # Destroy the node explicitly
      # (optional - otherwise it will be done automatically
      # when the garbage collector destroys the node object)
      minimal_publisher.destroy_node()
      rclpy.shutdown()
if __name__ == '__main__':
    main()
```

# Creating a Subscriber Node

Creates a subscriber for this node, defining the topic name, topic type, and callback to execute when a message is received

Callback function that is called when a new message is received

```
import rclpy
from rclpy.node import Node
from std_msgs.msg import String
class MinimalSubscriber(Node):
    def __init__(self):
        super().__init__('minimal_subscriber')
        self.subscription = self.create_subscription( String, 'topic',
                                                    self.listener_callback,
                                                    10)
    def listener_callback(self, msg):
        self.get_logger().info('I heard: "%s"' % msg.data)
def main(args=None):
    rclpy.init(args=args)
    minimal_subscriber = MinimalSubscriber()
    rclpy.spin(minimal_subscriber)
    # Destroy the node explicitly
    # (optional - otherwise it will be done automatically
    # when the garbage collector destroys the node object)
    minimal_subscriber.destroy_node() rclpy.shutdown()
if __name__ == '__main__':
    main()
```

# Adding Nodes to ROS2 Run

Update the settings.py file to add the nodes as available entrypoints

```
entry_points={
    'console_scripts': [
        'talker = py_pubsub.publisher_member_function:main',
        'listener = py_pubsub.subscriber_member_function:main',
    ],
},
```

Build the package

```
colcon build --packages-select py_pubsub
```

Launch the listener/talker nodes

```
ros2 run py_pubsub talker
```

```
ros2 run py_pubsub listener
```

# Resources

---

Creating a ROS2 package

<https://index.ros.org/doc/ros2/Tutorials/Creating-Your-First-ROS2-Package/>

Creating a Publisher/Subscriber

<https://index.ros.org/doc/ros2/Tutorials/Writing-A-Simple-Py-Publisher-And-Subscriber/>

# Assignment 3

- Goal: Create a ROS2 node that can control a robot with Mecanum wheels. Turning a desired velocity command into individual wheel commands.
  - Create a new ROS2 package
  - Write a node that subscribes to a `cmd_vel` topic and publishes the four wheel velocities to achieve that velocity
    - Subscribes to:
      - `/cmd_vel` : `geometry_msgs.Twist` – The desired velocity (linear and angular) for the center of the vehicle.
    - Publishes:
      - `/lf_wheel_vel_cmd` : `geometry_msgs.Float64` – The angular velocity for the left front wheel
      - `/rf_wheel_vel_cmd` : `geometry_msgs.Float64` – The angular velocity for the right front wheel
      - `/lb_wheel_vel_cmd` : `geometry_msgs.Float64` – The angular velocity for the left back wheel
      - `/rb_wheel_vel_cmd` : `geometry_msgs.Float64` – The angular velocity for the right back wheel
  - This node should publish the wheel velocity commands whenever it receives a desired velocity command
  - Test using the following command to generate command messages

```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```



# JOHNS HOPKINS

WHITING SCHOOL  
*of* ENGINEERING

© The Johns Hopkins University 2025, All Rights Reserved.