Unit 2: Locomotion
Kinematics of Wheeled Robots: Part 3

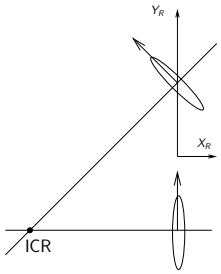
Computer Science 4766/6912

Department of Computer Science
Memorial University of Newfoundland

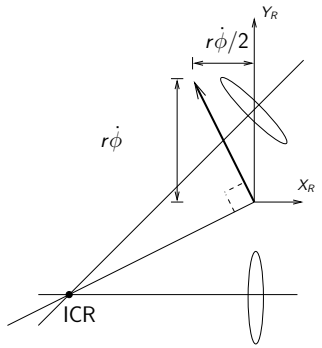
May 23, 2018

Manoeuvrability: Degree of Mobility

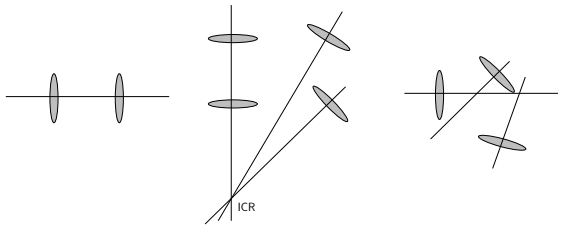
- Some robots are more manoeuvrable than others
  - Intuitively, we can see that the differential drive robot is more manoeuvrable than the turning bicycle
- A robot's degree of mobility is defined by its sliding constraints
- Sliding constraints can be visualized by drawing a *zero motion line* through the wheel's axis, perpendicular to the wheel plane
- The intersection of all zero motion lines defines the *instantaneous centre of rotation* (ICR)



- A robot's instantaneous velocity must be tangential to the circle centred at its ICR
- Recall the translatory component of the turning bicycle's velocity: $\dot{x}_R = -r\dot{\phi}_1/2, \quad \dot{y}_R = r\dot{\phi}_1$
- For any given r and $\dot{\phi}_1$ the velocity vector (heavy vector below) will be orthogonal to the line connecting P and ICR



ICR's for Various Wheel Configurations



- Left: For a differential-drive the two zero-motion lines are coincident; Thus, the ICR is constrained only to lie somewhere on that line
- Centre: For an Ackerman configuration (approximated by modern cars) the two rear wheels give only one zero-motion line; To prevent slipping, the two front wheels must be steered such that their zero motion lines intersect the rear line at a common point
- Right: A degenerate configuration; There is no ICR; If there is no slipping, there is also no movement

- We can formally characterize a robot's degree of mobility
- Consider a differential-drive robot; It has two wheels but only one *independent* sliding constraint
- To determine the degree of mobility we count the number of independent sliding constraints
- Define a matrix $\boldsymbol{C}$ that encodes the wheel direction component of the sliding constraint equations for all wheels
- The rank of this matrix is the number of independent constraints
  - *rank* = number of independent rows or columns (have to be equal)

- Examples:
  - Differential-drive:

$$\boldsymbol{C} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$rank\,[\boldsymbol{C}] = 1$$

  - Turning bicycle:

$$\boldsymbol{C} = \begin{bmatrix} -1 & 0 & -1 \\ -\sqrt{2}/2 & -\sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$$

$$rank\,[\boldsymbol{C}] = 2$$

- The maximum rank of an $N \times 3$ matrix is ____
- We define a robot's **degree of mobility** as follows,

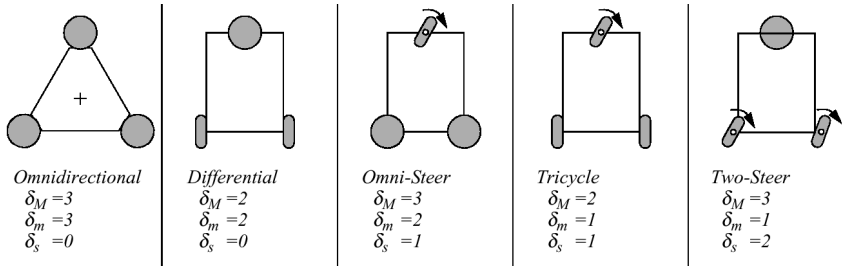
$$\delta_m = 3 - rank\,[\boldsymbol{C}]$$

  - Differential-drive: $\delta_m = 2$
  - Turning bicycle: $\delta_m = 1$
  - Robot with all omnidirectional wheels: $\delta_m = 3$
- Determining δ_m is an important part of determining how manoeuvrable a robot is; However, the fact that some wheels are steerable should also be considered...

- We define a robot's **degree of steerability**, δ_s, as the number of *independently steerable wheels* that yield a valid ICR
  - A normal bicycle: $\delta_s = 1$
  - A car: $\delta_s = 1$ (cannot independently steer both front wheels)
  - The maximum δ_s is 2: Once two wheels define the ICR, the choice of the third is not independent
- We define a robot's **degree of manoeuvrability** as follows,

$$\delta_M = \delta_m + \delta_s$$

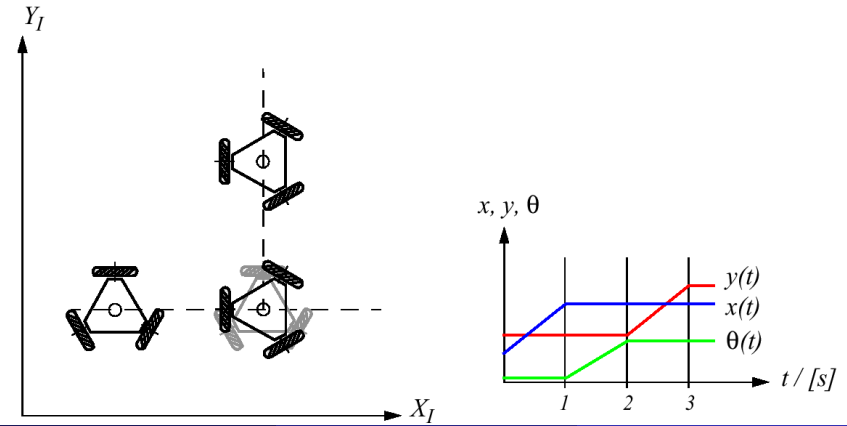
Degrees of Manoeuvrability, Mobility, and Steerability for Various Configurations



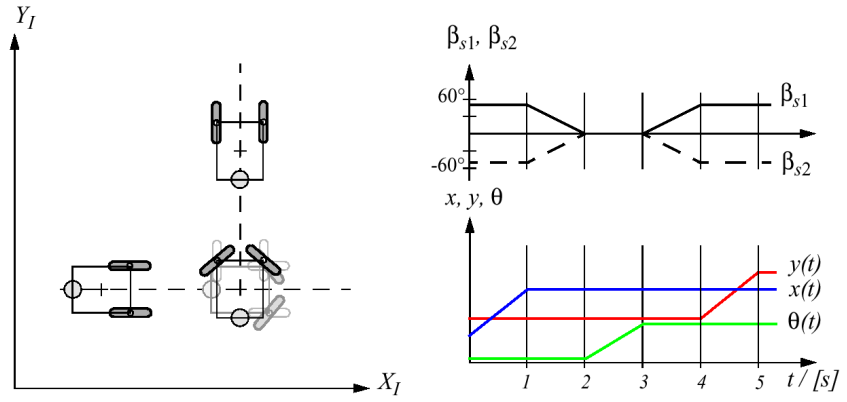
- The term *holonomic robot* usually refers to a robot with no constraints on its motion (a.k.a. an *omnidirectional robot*)
- Nonholonomic robots are subject to *nonholonomic constraints* (i.e. sliding constraints)
- Omnidirectional robots have $\delta_M = 3$ and exhibit the best possible manoeuvrability; However, the omnidirectional wheels required for such robots (i.e. Swedish, castor, or spherical) have some drawbacks:
  - increased complexity and expense
  - reduced accuracy for dead reckoning
  - reduced ground clearance for powered versions
  - standard wheels can passively counteract lateral forces; more efficient and stable for high-speed turns

What is the difference between an omnidirectional robot, and a non-omnidirectional robot such as the Two-Steer? Both have $\delta_M = 3$?

The difference is that it takes *time* for a steered robot to steer its wheels to the appropriate positions; Consider this omnidirectional robot following an 'L' shaped trajectory



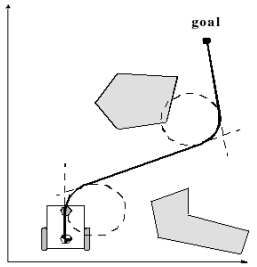
Now consider the trajectory of the Two-Steer



During time intervals 1-2 and 3-4 the robot was doing nothing but steering its wheels; The omnidirectional robot could transition between segments of the trajectory without any delay; Both robots take the same **path** but the **trajectories** (path + time dimension) differ

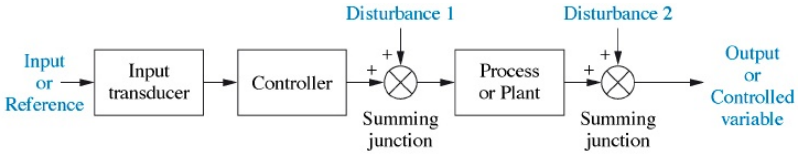
Motion Control: Trajectory Following

- One simple means of controlling the motion of a robot is to decompose its path into a sequence of elementary motions
  - Elementary motions may include lines and segments of circles which any robot with $\delta_M \geq 2$ can execute



- The robot's trajectory can be planned completely in advance without using any sensors → *Open-loop control*
- Alternatively, information from the sensors can be used to update the plan → *Closed-loop control*

Open-Loop System:



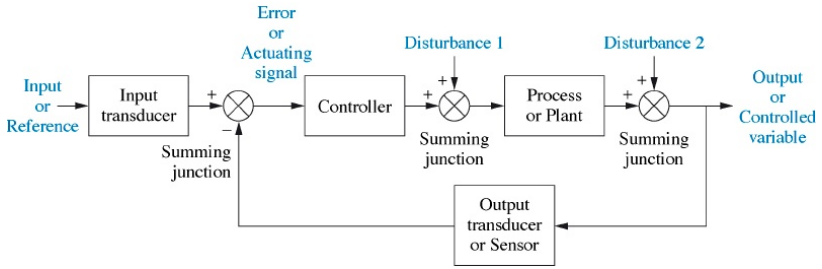
The input (a.k.a. *reference*) is first converted by the *input transducer* to the form required by the *controller*. The *process* or *plant* caries out the core function of the system (e.g. the furnace in a heating system, motors in a robot). The output (a.k.a. the *controlled variable*) differs from its desired value because of the two disturbances.

e.g. Open-loop heating system: The controller is an electronic amplifier and disturbance 1 is noise in the amplifier's output. Disturbance 2 might be variations in temperature due to the furnace itself.

Open-loop systems cannot correct for disturbances. In our example, the final temperature would deviate from the desired temperature.

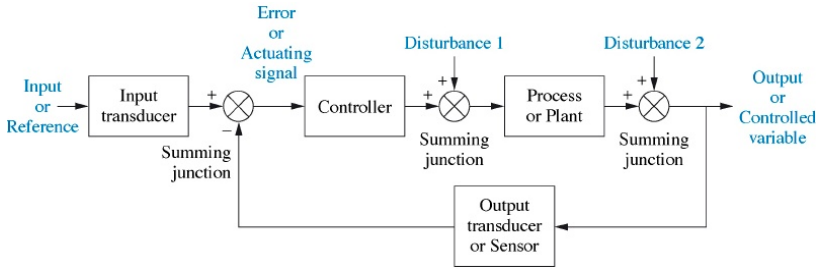
e.g. A toaster is an open-loop system. It cannot correct for the thickness of the bread or whether it is whole wheat or white.

Closed-Loop System:



In a closed-loop system there is an *output transducer* or *sensor* which converts the output into the form used by the controller. e.g. Position can be converted to an electrical signal by a potentiometer.

The first summing junction subtracts the output signal from the input. This is the *error signal*.



Closed-loop systems compensate for disturbances through **feedback**. If the actuating signal is zero then the output is correct and the plant does not need to be driven. Otherwise, the actuating signal describes how different the output is from what it should be. This drives the plant to correct this difference.

While open-loop systems fail to correct for disturbances or changes in the environment, they will tend to be simpler and cheaper than closed-loop systems. Thus, there is a trade-off to consider between them.

Closed-Loop Control

- Consider the problem of driving a differential-drive robot to goal position $\boldsymbol{g}_I = [\boldsymbol{g}_{Ix}\ \boldsymbol{g}_{Iy}]$, expressed in the global reference frame
- (Later we will consider the problem of arriving at the goal position with a particular orientation)
- We need to determine how to set the robot's forward speed $v(t)$ and rotational speed $\omega(t)$
- For a differential-drive robot we have the following,

$$\begin{aligned} v(t) &= \dot{x}_R = \frac{r(\dot{\phi}_r + \dot{\phi}_l)}{2} \\ w(t) &= \dot{\theta} = \frac{r(\dot{\phi}_r - \dot{\phi}_l)}{2l} \end{aligned}$$

- If we can obtain $\boldsymbol{g}_R$ then we can apply some control function f to get $v(t)$, the forward velocity component, and $\omega(t)$, the angular velocity component

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = f\left(\boldsymbol{g}_R\right)$$

- The control function should drive the robot such that,

$$\lim_{t\to\infty} \boldsymbol{g}_R(t) = [0\ 0]^T$$

which just means that the robot will eventually reach the goal

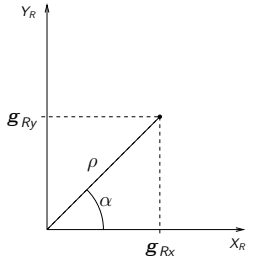
...Some Details

- We are given $\boldsymbol{g}_I$; How do we determine $\boldsymbol{g}_R$?
- COVERED ON BOARD

- $\boldsymbol{g}_R = R_{cw}(\theta)(\boldsymbol{g}_I - [x\ y]^T)$
- (We should use the 2×2 rotation matrix here)
- Clearly we need $[x\ y]^T$; How do we get that?
  - Odometry (previously covered), or by using a map (to be covered)

Two-step Controller

- We break the problem into two steps:
  - Turn to face the goal
  - Move towards goal
- First, it is convenient to express $\boldsymbol{g}_R$ using polar coordinates $[\rho\ \alpha]^T$



- The two steps can now be specified:
  - Minimize α
  - Minimize ρ

The Controller: Two States

parameters: k_α, ϵ_α, k_ρ, ϵ_ρ

1. $$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} 0 \\ k_\alpha\ \mathrm{sign}(\alpha) \end{bmatrix}$$
   Switch to state 2 if $|\alpha| < \epsilon_\alpha$
2. $$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} k_\rho \\ 0 \end{bmatrix}$$
   End if $\rho < \epsilon_\rho$

Note: angle α must be in $[-\pi, \pi]$

- There are problems with this controller:
  - If the first step fails, the second will also fail
  - It is difficult to choose appropriate values for the parameters: $k_\alpha, \epsilon_\alpha, k_\rho, \epsilon_\rho$
    - Smaller thresholds require high-precision localization and actuation (if too small, goal is never reached)
    - Larger thresholds reduce accuracy
  - Splitting the motion into two distinct phases is **inefficent**; We can save time by moving forwards while turning

Smooth Controller 1

- We try to minimize the quantities $\boldsymbol{g}_{Rx}$, $\boldsymbol{g}_{Ry}$, but now we minimize both simultaneously
- Consider the following control law

$$\begin{aligned} v(t) &= k_v \boldsymbol{g}_{Rx} \\ \omega(t) &= k_\omega \boldsymbol{g}_{Ry} \end{aligned}$$

  where the k parameters are positive

- The robot drives forward until $\boldsymbol{g}_{Rx} = 0$
- If $\boldsymbol{g}_{Ry}$ is positive, robot will turn CCW to face the goal; If negative it will turn CW.

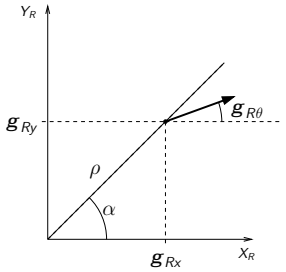
Smooth Controller 2

- Assume we now wish to drive the robot to a desired **pose**
  - *Pose* means (x, y) position and orientation θ
- We are given the goal pose $\boldsymbol{g}_I = [\boldsymbol{g}_{Ix}\ \boldsymbol{g}_{Iy}\ \boldsymbol{g}_{I\theta}]^T$, expressed in the inertial reference frame
- We need the goal pose in the robot reference frame

$$\boldsymbol{g}_R = R_{cw}(\theta)(\boldsymbol{g}_I - \boldsymbol{\xi}_I)$$

- (We should use the 3×3 rotation matrix here)

- Again it will be useful to express the goal pose using polar coordinates



- The final orientation is given by $\boldsymbol{g}_{R\theta}$
- We require a controller that minimizes α, ρ, and $\boldsymbol{g}_{R\theta}$ simultaneously

- Consider the following control law

$$\begin{aligned} v(t) &= k_\rho \rho \\ \omega(t) &= k_\alpha \alpha - k_\theta \boldsymbol{g}_{R\theta} \end{aligned}$$

  where the following conditions hold:

  - $\alpha \in [-\pi, \pi]$
  - all of the k parameters are positive
  - $k_\theta < k_\alpha$

- The robot drives forward until $\rho = 0$
- If α is positive, robot will turn CCW to minimize it
- $k_\theta < k_\alpha$ so θ does not have much influence until α becomes small; At this point the robot will be driven to turn away from the goal; This increases α so the robot will turn towards the goal again, only now $\boldsymbol{g}_{R\theta}$ will be reduced

Smooth Controller 2: Refinement

- If $\alpha \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$ then the robot will approach the goal directly (although its trajectory will be curved)
- If $\alpha \in \left(-\pi, -\frac{\pi}{2}\right] \cup \left(\frac{\pi}{2}, \pi\right]$ then the robot will first have to turn around before approaching the goal; We can detect this situation and modify the control law so that the robot backs up to the goal position, without turning around

$$\begin{aligned} v(t) &= -k_\rho \rho \\ \omega(t) &= -k_\alpha(\alpha - \pi) - k_\theta \boldsymbol{g}_{R\theta} \end{aligned}$$

(Here the angle $(\alpha - \pi)$ must be in $[-\pi, \pi]$)