Robotics

Lecture 2: Robot Motion

See course website `http://www.doc.ic.ac.uk/~ajd/Robotics/` for up to date information.

Andrew Davison
Department of Computing
Imperial College London

Robot Motion

- A mobile robot can *move* and *sense*, and must *process information* to link these two. In this lecture we concentrate on robot movement, or locomotion.

What are the possible goals of a robot locomotion system?

- Speed and/or acceleration of movement.
- Precision of positioning (repeatability).
- Flexibility and robustness in different conditions.
- Efficiency (low power consumption)?

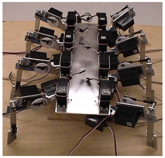
Locomotion

- Robots might want to move in water, in the air, on land, in space...?

AUV

Micro UAV

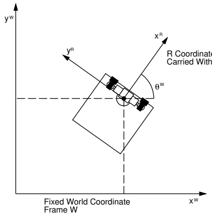
Zero-G Assistant


Spider

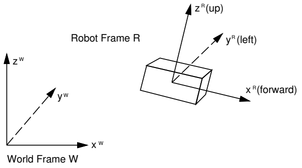
Humanoid

- In this course we will concentrate on wheeled robots which move on fairly flat surfaces.

Motion and Coordinate Frames



2D



3D

- We define two coordinate frames: a *world frame W* anchored in the world, and a *robot frame R* which is carried by and stays fixed relative to the robot at all times.
- Often we are interested in knowing the robot's *location*: i.e. what is the transformation between frames *W* and *R*?

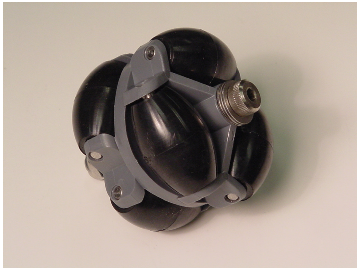
Degrees of Motion Freedom

- A rigid body which translates and rotates along a 1D path has 1 degree of freedom (DOF): translational. Example: a train.
- A rigid body which translates and rotates on a 2D plane has 3 DOF: 2 translational, 1 rotational. Example: a ground robot.
- A rigid body which translates and rotates in a 3D volume has 6 DOF: 3 translational, 3 rotational. Example: a flying robot.

- A *holonomic robot* is one which is able to move instantaneously in any direction in the space of its degrees of freedom.
- Otherwise a robot is called *non-holonomic*.

A Holonomic Ground Robot

- Holonomic robots do exist, but need many motors or unusual designs and are often impractical.
- Ground-based holonomic robots can be made using omnidirectional wheels; e.g. `http://www.youtube.com/watch?v=HkhGr7qfeT0`



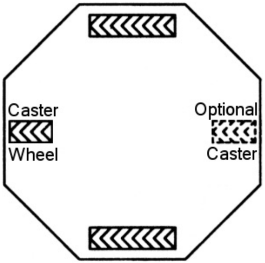
Standard Wheel Configurations

Drive and Steer

Differential Drive

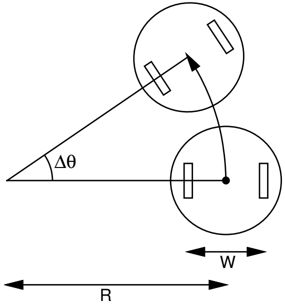
- Simple, reliable, robust mechanisms suitable for robots which essentially move in a plane.
- Both of these configurations are *non-holonomic* (each uses two motors, but has three degrees of movement freedom). For instance, a car-like robot can't instantaneously move sideways.

Differential Drive



- Two motors, one per wheel: steering achieved by setting different speeds.
- Wheels run at equal speeds for straight-line motion.
- Wheels run at equal and opposite speeds to turn on the spot.
- Other combinations of speeds lead to motion in a circular arc.

Circular Path of a Differential Drive Robot



We define the wheel velocities of the left and right wheels respectively to be v_L and v_R (linear velocities of the wheels over the ground: e.g. $v_L = r_L \omega_L$, where r_L is the radius of the wheel and ω_L is its angular velocity). The width between the wheels of the differential drive robot is W.

- Straight line motion if $v_L = v_R$
- Turns on the spot if $v_L = -v_R$
- More general case: moves in a circular arc.

Circular Path of a Differential Drive Robot

To find radius R of curved path: consider a period of motion Δt where the robot moves along a circular arc through angle $\Delta\theta$.

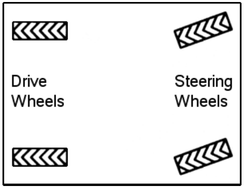
- Left wheel: distance moved $= v_L \Delta t$; radius of arc $= R - \frac{W}{2}$.
- Right wheel: distance moved $= v_R \Delta t$; radius of arc $= R + \frac{W}{2}$.
- Both wheel arcs subtend the same angle $\Delta\theta$ so:

$$\begin{aligned} \Delta\theta &= \frac{v_L \Delta t}{R - \frac{W}{2}} = \frac{v_R \Delta t}{R + \frac{W}{2}} \\ \Rightarrow \frac{W}{2}(v_L + v_R) &= R(v_R - v_L) \end{aligned}$$

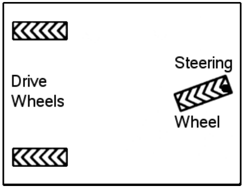
$$\boxed{\Rightarrow R = \frac{W(v_R + v_L)}{2(v_R - v_L)} \qquad \Delta\theta = \frac{(v_R - v_L)\Delta t}{W}}$$

These equations are the basis for *odometry*: given certain control inputs, how does the robot move?

Car/Tricycle/Rack and Pinion Drive



Car

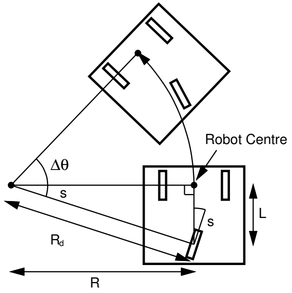


Tricycle

- Two motors: one to drive, one to steer.
- Cannot normally turn on the spot.
- With a fixed speed and steering angle, it will follow a circular path.
- With four wheels, need rear differential and variable (‘Ackerman’) linkage for steering wheels.

Circular Path of a Car-Like Tricycle Robot

This is a robot configuration which has a single steerable and drivable wheel at the back. The front wheels are free running.



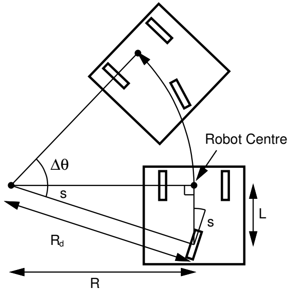
Assuming no sideways wheel slip, we intersect the axes of the front and back wheels to form a right-angle triangle, and obtain:

$$R = \frac{L}{\tan s} \ .$$

The radius of the path that the rear driving wheel moves in is:

$$R_d = \frac{L}{\sin s} \ .$$

Circular Path of a Car-Like Tricycle Robot



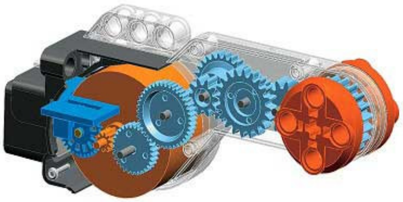
In time Δt the distance along its circular arc moved by the drive wheel is $v\Delta t$, so the angle $\Delta\theta$ through which the robot rotates is:

$$\Delta\theta = \frac{v\Delta t}{R_d} = \frac{v\Delta t \sin s}{L} \ .$$

$$\boxed{R = \frac{L}{\tan s} \qquad \Delta\theta = \frac{v\Delta t \sin s}{L}}$$

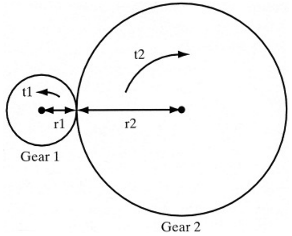
Actuation of Driving Wheels: DC Motors

- DC motors are available in all sizes and types.
- A power signal is sent to the motor (using Pulse Width Modulation PWM).
- For precision, encoders and feedback can be used for *servo* control using a PID control law. Our Lego motors have built-in encoders.



Gearing

- DC motors tend to offer high speed and low torque, so gearing is nearly always required to drive a robot



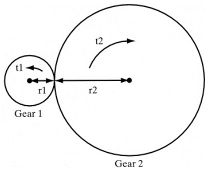
If Gear 1 is driven with torque t_1, it exerts tangential force:

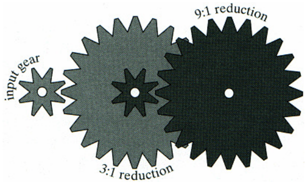
$$F = \frac{t_1}{r_1}$$

on Gear 2. The torque in Gear 2 is therefore:

$$t_2 = r_2 F = \frac{r_2}{r_1} t_1 \ .$$

Gearing





The change in angular velocity between Gear 1 and Gear 2 is calculated by considering velocity at the point where they meet:

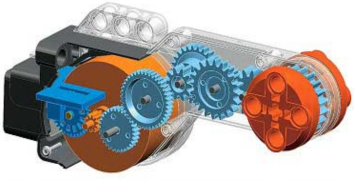
$$\begin{aligned} v &= \omega_1 r_1 = \omega_2 r_2 \\ \Rightarrow \omega_2 &= \frac{r_1}{r_2}\omega_1 \end{aligned}$$

- When a small gear drives a bigger gear, the second gear has higher torque and lower angular velocity in proportion to the ratio of teeth.
- Gears can be chained together to achieve compound effects.

Powering and Controlling Motors

- Most basically, we can set an amout of power to send to a motor, which will cause it to start to move. Most often, this will be a voltage signal with a fixed amplitude but with the amount of ‘fill-in’ set using Pulse Width Modulation (PWM). E.g. in the BrickPi interface we can set a power level in percent, from -100 to 100.
- A set power level will cause an angular response of the motor which depends on various things: the friction in the gears; the amount of load connected to the motor (e.g. the mass of the robot it has to move, or the resistance it is pushing).

Feedback or Servo Control Using an Encoder



- A Lego motor has an encoder which records angular position. (Actually the encoder is attached directly to the motor spindle, but the driver software scales this via the gearing to report the angular position of the orange end effector of the motor).
- So we can use feedback control (servo control) to make the motor do what we want.
- Principle: decide where we want the motor to be at every point in time. At a high rate, check where the motor actually is from the encoder. Record the difference (the error). Send a power demand to the motor depending on the error, aiming to reduce it.
- Our motors: record motion rotational position in degrees.
- Two main modes: position control (where demand is a constant) and velocity control (where demand increases linearly with time).

PID (Proportional/Integral/Differential) Control

- Error $e(t)$ is demand minus actual position.
- PID expression: sets power as a function of error:
  $$P(t) = k_p e(t) + k_i \int_{t_0}^{t} e(\tau) d\tau + k_d \frac{de(t)}{dt}$$
- k_p, k_i and k_d are gain constants which can be tuned.
- k_p is the main term: high values give rapid response but possible oscillation.
- k_i, integral term, can be increased to reduce steady state error.
- k_d, differential term, can be increased to reduce settling time.

BrickPi API

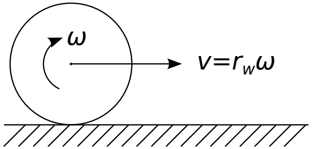
- `BP.set_motor_power(BP.PORT_A, power)`: set a raw power level (in the range -100 to 100) which will make the motor move without PID control.
- `BP.set_motor_power(BP.PORT_A, BP.MOTOR_FLOAT)`: set the motor to ‘float’ without power, such that it can be turned by hand (the encoder can still be read so this is an interesting way to interface with a robot).
- `BP.get_motor_encoder(BP.PORT_A)`: returns the current encoder position in degrees.
- `BP.offset_motor_encoder(BP.PORT_A, BP.get_motor_encoder(BP.PORT_A))`: resets the encoder count to zero.
- `BP.set_motor_position(BP.PORT_A, degrees)`: set a position demand for the motor in degrees, and start PID control to reach it.
- `BP.set_motor_dps(BP.PORT_A, dps)`: set a velocity demand for the motor in degrees per second, and start PID control to achieve it.

BrickPi API

- `BP.get_motor_status(BP.PORT_A)`: return a tuple of four values which indicate the current status flag, power in percent, encoder position in degrees and current velocity in degrees per second.
- `BP.set_motor_limits(BP.PORT_A, power, dps)`: set limits on the power and degrees per second that will be used in PID control. These are useful to protect your BrickPi and motors from overloading (we would recommend usually staying below 70% power).
- `BP.set_motor_position_kp(BP.PORT_A, kp)`: set PID proportional gain constant; default is 25.
- `BP.set_motor_position_kd(BP.PORT_A, kd)`: set PID differential gain constant; default is 70.
- `BP.reset_all()`: disable all motors and sensors.

Mapping Wheel Rotation Speed to Velocity

- What is the robot speed, when the wheels turn?



- In principle, we could measure the radius of each wheel r_w to turn angular velocity into linear motion. However, in practice (due to hard to model factors, such as surface slip and tyre softness) it is much better to *calibrate* such things empirically. i.e., via experiments (guided trial and error), work out the scaling between the motor reference angle and distance travelled over the ground.

Motion and State on a 2D Plane

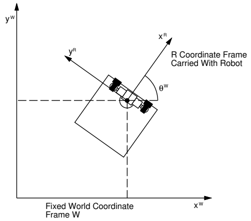
- If we assume that a robot is confined to moving on a plane, its location can be defined with a *state vector* **x** consisting of three parameters:

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

- x and y specify the location of the pre-defined ‘robot centre’ point in the world frame.
- θ specifies the rotation angle between the two coordinate frames (the angle between the x^W and x^R axes).
- The two coordinate frame coincide when the robot is at the origin, and $x = y = \theta = 0$.

Integrating Motion in 2D

- 2D motion on a plane: three degrees of positional freedom, represented by (x, y, θ) with $-\pi < \theta <= \pi$.
- Consider a robot which only drives ahead or turns on the spot:



- During a straight-line period of motion of distance D:

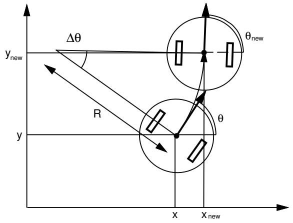
$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x + D\cos\theta \\ y + D\sin\theta \\ \theta \end{pmatrix}$$

- During a pure rotation of angle angle α:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta + \alpha \end{pmatrix}$$

Integrating Circular Motion Estimates in 2D

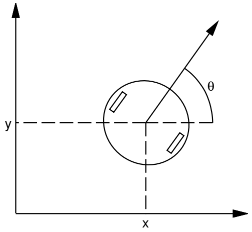
- Simple rotate, move motion like Bigtrak! `https://www.youtube.com/watch?v=due9mvuUL-I`.



- More generally, in the cases of both differential drive and the tricycle robot, we were able to obtain expressions for R and $\Delta\theta$ for periods of constant circular motion. Given these:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x + R(\sin(\Delta\theta + \theta) - \sin\theta) \\ y - R(\cos(\Delta\theta + \theta) - \cos\theta) \\ \theta + \Delta\theta \end{pmatrix}$$

Position-Based Path Planning



Assuming that a robot has *localisation*, and knows where it is relative to a fixed coordinate frame, then position-based path planning enables it to move in a precise way along a sequence of pre-defined waypoints. Paths of various curved shapes could be planned, aiming to optimise criteria such as overall time or power usage. Here we will consider the specific, simple case where we assume that:

- Our robot's movements are composed by straight-line segments separated by turns on the spot.
- The robot aims to minimise total distance travelled, so it always turns immediately to face the next waypoint and drives straight towards it.

Position-Based Path Planning

In one step of path planning, assume that the robot's current pose is (x, y, θ) and the next waypoint to travel to is at (W_x, W_y).

- It must first rotate to point towards the waypoint. The vector direction it must point in is:

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} W_x - x \\ W_y - y \end{pmatrix}$$

  The absolute angular orientation α the robot must drive in is therefore given by:

$$\alpha = \tan^{-1} \frac{d_y}{d_x}$$

Care must be taken to make sure that α is in the correct quadrant of $-\pi < \alpha \leq \pi$. A standard $\tan^{-1}$ function will return a value in the range $-\pi/2 < \alpha <= \pi/2$. This can be also achieved directly with an `atan2(dy, dx)` function (available in Python's `math` module).

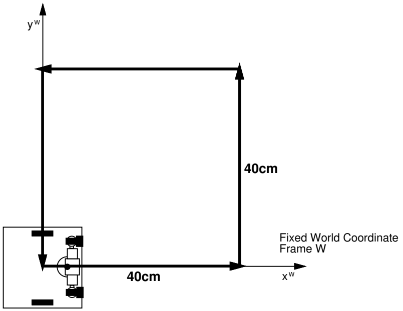
Position-Based Path Planning

- The angle the robot must rotate through is therefore $\beta = \alpha - \theta$. If the robot is to move as efficiently as possible, care should be taken to shift this angle by adding or subtracting 2π so make sure that $-\pi < \beta \leq \pi$.
- The robot should then drive forward in a straight line through distance $d = \sqrt{d_x^2 + d_y^2}$.

Week 2 Practical

- Please READ THE WHOLE PRACTICAL SHEET CAREFULLY.
- Lab location: teaching lab 219, one floor down.
- Organise yourselves into groups and come to us to fill in a form and get a kit.

Week 2 Practical: Getting Started with BrickPi; Accurate Robot Motion



- Today's practical is on accurate robot motion. How well is it really possible to estimate robot motion from wheel odometry?
- **Everyone should read the practical sheet fully!**
- This is an **ASSESSED** practical: we will assess your achievement next week at the start of Thursday's practical. Your whole group should be there to demonstrate and discuss your robot.