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
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...?

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

- 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.*
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
Exotic Wheeled Robots

- Segway platform with dynamic balance gives good height with small footprint and high acceleration. Self-balancing Lego Robots built by DoC students in 2008:
  http://www.youtube.com/watch?v=fQQctJz7ap4
- Mars Rover has wheels on stalks to tackle large obstacles.
Standard Wheel Configurations

Rack and Pinion  Differential Drive  Skid-Steer  Synchro Drive

- Simple, reliable, robust mechanisms suitable for robots which essentially move in a plane.
- All of these robots 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.
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 $\omega_L$ 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 $\Delta t$ 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:

$$\Delta \theta = \frac{v_L \Delta t}{R - \frac{W}{2}} = \frac{v_R \Delta t}{R + \frac{W}{2}}$$

$$\Rightarrow \frac{W}{2} (v_L + v_R) = R(v_R - v_L)$$

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

These equations are the basis for *odometry*: given certain control inputs, how does the robot move?
Car/Tricycle/Rack and Pinion Drive

- 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:

\[ R_d = \frac{L}{\sin s} \]
In time $\Delta t$ 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:

$$\Delta \theta = \frac{v \Delta t}{R_d} = \frac{v \Delta t \sin s}{L}.$$
Actuation of Driving Wheels: DC Motors

- DC motors are available in all sizes and types.
- Simple control with voltage or Pulse Width Modulation (PWM).
- For precision, encoders and feedback can be used for servo control (the NXT 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:

\[
t_2 = r_2 F = \frac{r_2}{r_1} t_1 .
\]
The change in angular velocity between Gear 1 and Gear 2 is calculated by considering velocity at the point where they meet:

\[ v = \frac{\omega_1 r_1}{r_2} = \frac{\omega_2 r_2}{r_1} \]

\[ \Rightarrow \omega_2 = \frac{r_1}{r_2} \omega_1 \]

- 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.
Motor Control — Open Loop

Let $P$ be a Single-Input-Single-Output (SISO) dynamic system (e.g. Lego Mindstorms geared DC motor). It is described by:

- an input $u$,
  *here:* voltage $V$ or corresponding PWM value,
- internal states $x$, whose dynamics follow differential equations,
  *here:* $x = [x_1, x_2]^T = [\omega, \phi]^T$, with rotation speed $\omega$ and angle $\phi$,
- an output $y$ as a function of $x$,
  *here:* angle $\phi$, i.e. $y = x_2$.

Qualitative open-loop response on input (voltage) step: input signal $V$ leads to output motor angle $\phi$, which changes with time.
Motor Control — Closed Loop

- How about if we want to accurately reach and keep a reference angle?

  → **Closed-loop control!**

- Let $C$ be a controller, possibly with internal states:

  $r$ is a reference (desired) output, and 
  $e$ is the error between reference and actual output.

- The controller runs at a high frequency. At each iteration, it checks the current error, and calculates a control value to the motor which aims to reduce the error.

- Simplest controller: binary on/off (like a thermostat). If the error is positive, go forward; if it is negative, go back.

- Next idea: proportional control: set a control signal proportional to the error.
More General Motor Control — PID

• PID (Proportional-Integral-Differential): a controller:

\[ C : u(t) = k_p e(t) + k_i \int_{t_0}^{t} e(\tau) d\tau + k_d \frac{de(t)}{dt}, \]

  • with \(k_p\): proportional gain, reduces the error,
  • \(k_i\): integral gain, removes steady-state error,
  • and \(k_d\): differential gain, can reduce settling time.

• Qualitative closed-loop system response to reference step (well-tuned controller):

• A simple (heuristic) tuning rule to try: Ziegler-Nichols:
  • set \(k_i\) and \(k_d\) to zero. Increase \(k_p\) until the system starts oscillating with period \(P_u\) (in seconds) — remember this gain as \(k_u\);
  • set \(k_p = 0.6k_u\), \(k_i = 2k_p/P_u\), and \(k_d = k_pP_u/8\).
Motor Control — Additional Tweaks

- Reference filtering: respect physical limits already in the reference:
  \[ r, r_f \]

- Anti-Reset-Windup: stops integrating the error for the I-part, when \( u \) is at its physical limit.

- Dead-band compensation: add offset to \( u \) to compensate friction.

- Feed-forward controller \( C_f : u_f(t) = k_f \frac{dr(t)}{dt} \), reduces "work" for the feed-back controller:
  \[ r, r_f \]

\[ u \]

\[ + \]

\[ u_f \]

\[ C_f \]

\[ C \]

\[ y \]
• What is the robot speed, when the wheels turn?

\[ v = r_w \omega \]

• 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* \( \mathbf{x} \) 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.
- \( \theta \) 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 \((x, y, \theta)\) with \(-\pi < \theta \leq \pi\).
- Consider a robot which only drives ahead or turns on the spot:

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

- During a straight-line period of motion of distance \(D\):

- During a pure rotation of angle angle \(\alpha\):

\[
\begin{pmatrix}
    x_{\text{new}} \\
    y_{\text{new}} \\
    \theta_{\text{new}}
\end{pmatrix} = \begin{pmatrix}
    x \\
    y \\
    \theta + \alpha
\end{pmatrix}
\]
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:

\[
\begin{pmatrix}
    x_{\text{new}} \\
    y_{\text{new}} \\
    \theta_{\text{new}}
\end{pmatrix}
= 
\begin{pmatrix}
    x + R(\sin(\Delta \theta + \theta) - \sin \theta) \\
    y - R(\cos(\Delta \theta + \theta) - \cos \theta) \\
    \theta + \Delta \theta
\end{pmatrix}
\]
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 \((x, y, \theta)\) 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:

\[
\begin{pmatrix}
    dx \\
    dy
\end{pmatrix} = \begin{pmatrix}
    W_x - x \\
    W_y - y
\end{pmatrix}
\]

The absolute angular orientation \(\alpha\) the robot must drive in is therefore given by:

\[
\alpha = \tan^{-1} \frac{dy}{dx}
\]

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

The robot should then drive forward in a straight line through distance $d = \sqrt{d_x^2 + d_y^2}$. 

Position-Based Path Planning
• 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.
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 next week’s practical. Your whole group should be there to demonstrate and discuss your robot.