AI for Mobile Robots - CSIP5202 -

Actuators

Overview

- Robot actuators
- Mobility considerations
- Legs, wheels and tracks
- Arms and Grippers
- Measuring motion

Mobility Considerations

- As in this module we consider mainly mobile robots, the ability to move around autonomously is one of the main considerations in terms of actuators.
  - We can consider many of these actuators and abilities for instance in the context of autonomous vehicles (e.g. self-driving cars), so basically their drive considerations.
- You must note that there are many additional types of actuators which serve as tools for other tasks (e.g. industrial robots)

Mobility Considerations

- Consider the following in terms of a self-driving car
    - Manoeuvrability: ability to alter direction/speed, so that the required environment can be navigated
    - Controllability: practical and not too complex in computational terms suitable for an onboard computer
    - Traction: sufficient to minimise slippage/ skidding while not energy-inefficient
    - Climbing: the ability to navigate environmental discontinuities (kerbs, gaps), slopes, rough surfaces, etc.

Mobility Considerations

- Issues that impact mobility:
  - Stability: must not fall/roll over in the environment it is intended for
  - Efficiency: power consumption reasonable to provide a suitable range and cost
  - Maintenance: easy to maintain, reliable, reasonable cost and availability for repairs
  - Environmental impact: does not damage or change the environment i.e. efficient use of energy
  - Navigation: accuracy of dead-reckoning and controllability of its trajectory

Legs

- Two legs seem the most familiar configuration
  - But balance is a problem that requires significant computational resources
  - Need very fast and supple joints to move and balance
  - Standing still is an active task (can require the use of energy!!)
- Four legs are much easier to balance and move … are they?
- Six legs are much easier to balance and move
  - Stable when not moving
  - Can work with simple cams and rigid legs
  - Processing is much simpler!

Wheels

- Again, any number of wheels is possible
  - there are many different configurations that are useful
- Two individually driven wheels on either side
  - usually with one or more idler wheels for balance
  - independently driven wheels allow zero turning radius
    - one wheel drives forwards, one wheel drives backwards
- Rear/front-wheel drive, with front-wheel steering
  - the vehicle will have a non-zero turning radius
  - for two front wheels, turning geometry is complex
  - the traction wheels’ shaft needs a differential to prevent slippage
- 4WD is possible, but it is even more complex

Wheels

- Autonomous vehicles (e.g. self-driving cars) can use the same configuration as petrol/diesel cars as we already have extensive technology and manufacturing infrastructure for this
  - We also have a structured environment where the navigation of this configuration is suited
- For other applications in autonomous robots, two individually driven wheels on either side are much more common
  - May be much more efficient for operation
  - May be much more cost-effective when building it

Exotic Wheels & Tracks

- Tracks can be used in the same way as 2 wheels
  - good for rough terrain (as compared to wheels)
  - tracks must slip to enable turns (skid steering) so are inefficient in terms of energy use (friction) and may damage the surface on which they move
- In synchro drive three or more wheels are coupled
  - drive in the same direction at the same rate
  - pivot in unison about their respective steering axes
  - allows the body of the robot to remain in the same orientation
- Tri-star wheels are composed of 3 sub wheels
  - The entire wheel assembly rolls over a large obstacle
- There are many other exotic wheel configurations
  - multiple degrees of freedom (MDOF), e.g. Kilough

Actuator Types

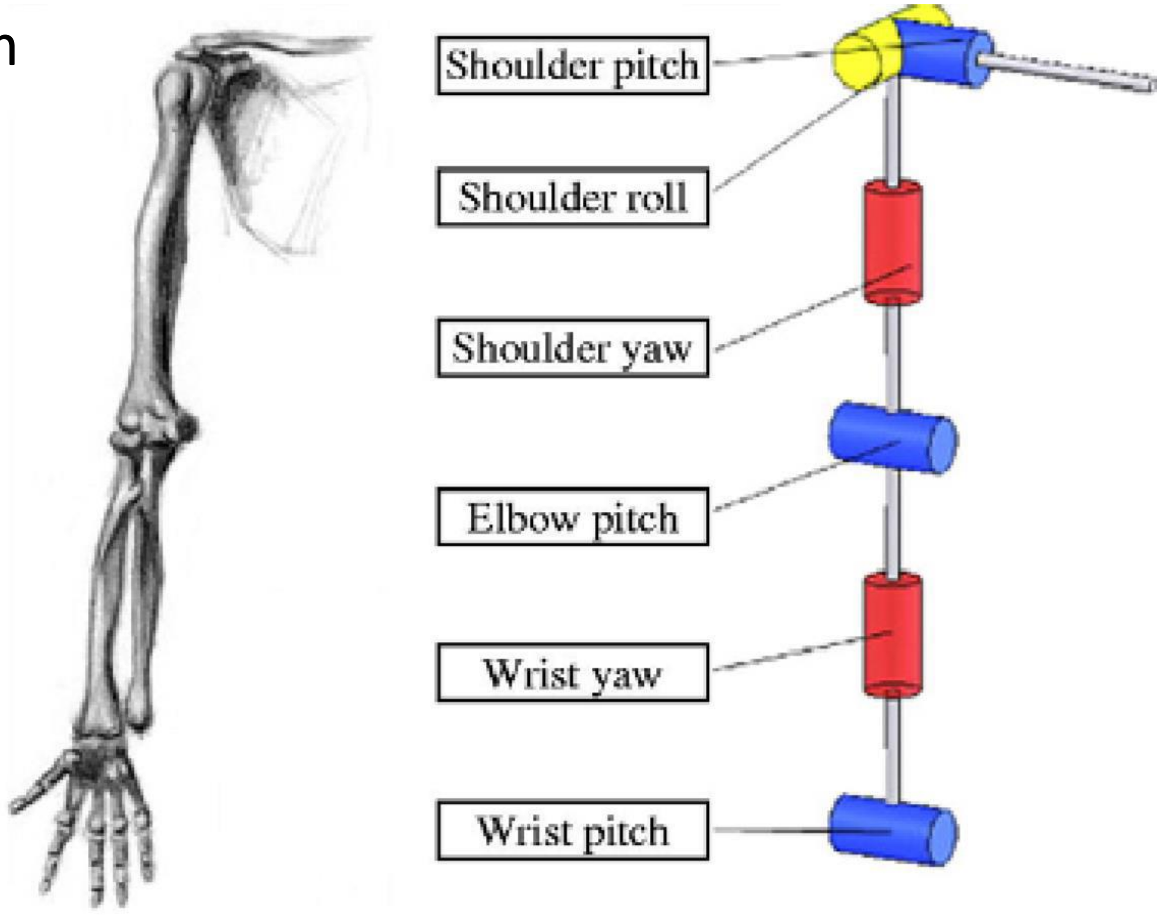
- Electric
  - A DC motor is the most common type used in mobile robots
  - Stepper motors turn a certain amount with a voltage pulse
- Pneumatic
  - operate by pumping compressed air through chambers: slow
- Hydraulic
  - pump oil under pressure: usually too heavy, dirty and expensive to be used on mobile robots
- Shape Memory Alloys
  - metallic alloys that deform under heat and then return to their previous shape: e.g. used for artificial muscles

Arm & Grippers

- Degrees of freedom
    - Independently controllable components of motion
- Arm
    - convenient method to allow full movement in 3D
    - more often used in fixed robots due to power and weight
    - even more difficult to control due to the extra degree of freedom
- Grippers
    - may be very simple (two rigid arms) to pick up objects
    - may be a complex device with fingers on the end of an arm
    - probably need feedback to control grip force e.g. picking up an egg

Arms & Grippers

- Degrees of freedom of the human arm
- And in the Hand? …



Measuring Motion: Odometry

- If wheels are being used, then the distance travelled can be calculated by measuring the number of turns
  - dead reckoning or odometry is the name given to the measure of distance (for navigation) in this way
- Motor speed and timing are very inaccurate
  - measuring the number of wheel rotations is better
  - shaft encoders, or rotation sensors, measure this
  - Can be used in Mobile Robots as well as Manipulators

Odometry

- There are many different types and technologies of shaft encoder
  - Optical encoders (relative)
  - Potentiometers (absolute for each turn)
- In practice, slippage, and inaccuracies (such as wheel size, sensor noise, etc..) produce an error
- And this error is relative

Odometry: A Relative Measurement

- A problem that odometry (and other “dead reckoning” based measurements) has is that it is a relative measurement:
    - We start at a “known location”
    - Relative to this, we measure the distance that the robot “seems” to have travelled according to internal state sensors
    - We calculate then where, relative to the starting point, we think we are

Odometry: A Relative Measurement

- The iRobot simulator provides us with odometry:
  - `DistanceSensorRoomba(SerialPort)` - returns the distance travelled since the last (previous) reading.
  - The manual states: “The wheels on the Create have encoders that keep track of how many times they spin. Knowing the radius of the wheel and the width of the robot allows the Create to keep track of how far it has travelled and how much it has turned.”

Odometry: A Relative Measurement

- Every time we calculate a new position, we get a small (hopefully!) error
- Next time we calculate a new position, we do it relative to the previous one, and hence, it’s previous error
- So in every new calculation, we accumulate the error of the previous cycles

Odometry: A Relative Measurement

- Eventually, this error grows to an unacceptable level
- Therefore odometry is only useful for small distances
    - You would have already observed this in your labs and discussed it in the discussion board

Odometry: A Relative Measurement

- So the iRobot simulator manual clarifies:
  - "… However, this does not account for wheels slipping, which makes the odometry inaccurate over large distances. It is not recommended to navigate using only 'dead reckoning' from odometry."
- Or... we need to reset the error periodically

Absolute Measurements

- Absolute measurements also have errors
- But the error does not accumulate
- It will be the same irrespective of the distance travelled
    - e.g. a GPS measurement will have an error of, say, 1m around the point we are at, but it will be 1m regardless of how much we travel, move, go in circles, slip, etc…

Absolute Measurements

- If we cannot use absolute measurements all the time, we can combine them with relative ones e.g. Landmarks
- Odometry could be the primary/short-term source of distance measurement
- Regularly, the error could be reset by checking an absolute distance (e.g. sonar) to a landmark
- But we need to know about them!

Absolute Measurements

- The iRobot simulator provides 2 ways of absolute measurements:
  - `OverheadLocalizationCreate` - returns the absolute (and very accurate) localization of the robot on the map, this could be seen as a type of GPS
  - `CameraSensorCreate` - The camera is used for finding beacons, so if you know the actual position of a beacon, you can use it to support odometry-based localization and correct accumulated error

Measuring Motion as we do

- When we (humans) navigate we have exactly the same techniques!
  - We use odometry for rough calculation of how much distance we have travelled
  - We use beacons (both implicit and explicit) to correct any relative/accumulated errors
  - We use absolute positioning systems (e.g. SatNav) to navigate over long distances
  - We use beacons to correct the error as the final detail…

Measuring Motion as we do

- We use odometry for a rough calculation of how much distance we have travelled we all have it in our cars. The odometer provides distance by counting the turns of the wheels and calculating relative to its size
- The “trip” odometer is used to measure short distances
- The odometer is used to measure long-term distances but where various miles of error are not critical!

Measuring Motion as we do

- We use beacons to correct any relative/accumulated errors
- You don’t just turn left when the odometer indicates e.g. “after 2 miles turn left
  - Implicit: You observe where there is a recognizable turning point (junction)
  - Explicit: You find the exact route with the sign!

Measuring Motion as we do

- We use absolute positioning (e.g. SatNav) to navigate over long distances, but once you arrive you don’t just blindly stop when it says “you’ve reached your destination”!
- We use beacons to correct the error at the end… again implicit (driveway, obstacles, …) or explicit (signs, demarcation, …)

Questions?