

CMPUT503: Experimental Mobile Robotics

Samuel Neumann

January 17, 2023

1 Introduction

Thu 05 Jan 2023

Goals: appreciate robotics, and understand what is possible and how, get all the basic knowledge you need in this class to jump into a robotics job (i.e. pass interviews)

Math is not the focus of this course.

If you do the work, you get the grade. But, you probably will have to come into the lab after lab hours (i.e. more than 3 in-lab hours)

Cut corners wherever you can, use already-built packages for assignments/labs

Lowest two lab grades will get dropped!

Look into DuckieTown before the first lab.

2 Robot Architecture and Locomotion

Tue 10 Jan 2023

Architecture can be thought of as the interaction of hardware and software.

2.1 Architectures

- Reactive Architecture
 - *Actions* are directly triggered by *sensors*.

- No representations of the environment
- Predefined, fixed response to situation
- *Fast* response to changes in the environment
- Limitations: no memory (unable to count), low computing power, no machine learning, knowledge of the world is limited by the range of its sensors, cannot *undo* and incorrect action, not possible to plan ahead, unable to recover from actions which fail silently, etc.
- Deliberative Architecture
 - Organized by decomposing the required system functionality into *concurrent modules* or components
 - Problems: overall complexity of the system may grow, hard to offer real-time guarantees on performance
 - * Solving any given problem takes longer than an equivalent *reactive* implementation
 - * Solving different problems take different amount of time
- Subsumption Architecture
- ROS?

Temporal Decomposition distinguishes between processes that have varying real-time and non-real-time demands

If you fail to meet a hard real-time requirement, then your system is broken. E.g.: driving off a cliff.

Rodney A. Brooks: tried to make a realistic robot with the abilities of an insect. He developed the *subsumption architecture*. This architecture puts different importance/prioritization on different tasks. E.g. I really don't want to fall off the cliff, but if I'm not falling off a cliff, then don't bump into any objects, etc.

2.2 What is ROS?

- A *meta* operating system for robots
- An **architecture** for distribution inter-process/inter-machine communication and configuration
- ROS is a *peer-to-peer* robot middleware package
- In ROS, all major functionality is broken into nodes that communicate with each other using messages sent on topics
- **Not** a programming language
- **Not** a *hard real-time* architecture
- ...

2.2.1 ROS Nodes

- **Node:** a process that performs some computation
- Typically we try to divide the entire software into different modules – each one is run on a single or multiple nodes.
- Nodes are combined together into graph and communicated with each other using streaming topics. Nodes **do not** communicate with each other directly, otherwise we get into the craziness and complications of the other architectures.

2.2.2 ROS Topics

- **Topic:** named buses over which nodes exchange messages to ROS master
- Topics have **anonymous** publish/subscribe semantics
- Can be *multiple* publisher and subscribers to a topic
- Strongly typed

2.2.3 ROS Messages

- **Message:** is a simple data structure, comprising typed fields.
- Can contain headers with metadata

3 Locomotion

Locomotion: enabling robots to move

Wheel locomotion is simple, safer, and stable:

Key issues for wheeled locomotion:

- Stability
 - Number and geometry of contact points
 - Centre of gravity
 - Static/dynamic stability
 - Inclination of terrain
- Characteristics of contact
 - Contact point/path size and shape
 - Angle of contact

- Friction
- Type of environment

Legged mobile robots can adapt to human terrain, but they have lots of limitations: power and mechanical complexity, high degrees of freedom, control system complexity, etc.

3.1 Leg Locomotion

Degree of Freedom (DOF): joints or axes in motion

A minimum of two DOFs is required to move a leg forward, but usually legs have three degrees of freedom. A fourth DOF is needed for the angle joint, which might improve walking.

As you add more DOFs, the complexity of the system increases very fast.

Arms need 6 or 7 DOFs.

Often, clever mechanical design can perform the same operations as complex active control circuitry.

Thu 12 Jan 2023

Gait control: how legs are coordinated for movement, e.g. crawl, trot, pace, bound, pronk, gallop, etc.

Number of gaits is $(2k - 1)!$ where k is the number of legs.

Cost of transportation:

- How much energy a robot uses to travel a certain distance
- Usually normalized by the robot weight
- Measured in $\frac{J}{N-m}$
- Driving on wheels has very low cost of transportation.

Legged robot control should be designed to better exploit the dynamics of the system. For example passive dynamic walking.

3.2 Wheeled Mobile Robots

Wheels are *the most popular locomotion mechanisms*:

- Highly efficient
- Simple mechanical implementation
- Balancing is not *usually* a problem, but a suspension system is needed to allow all wheel to maintain ground contact on uneven terrain.

We will focus on:

- Traction
- Stability
- Maneuverability
- Control – we will mostly focus on this

Wheel Designs:

- Standard wheels
 - 2 DOFs
- Castor wheels
 - 2 DOFs
 - Can move while the contact point of the wheel stays the same.
 - If something is super heavy, it's easier to get momentum with these wheels
 - Harder to control
- Swedish (Omni) wheels
 - 3 DOFs
- Ball or spherical wheel
 - 3 DOFs
 - Balled computer mice used these
 - Suspension issue – hard to get suspension on these

Stability of a vehicle is guaranteed with 3 wheels – centre of gravity is required to be within the triangle formed by ground contact points of wheels.

Stability is improved by 4 and more wheels

Holonomic: can move in any direction

Combining **actuation** and **steering** on one wheel makes design complex and adds additional errors for odometry.

Static stability with two wheels can be achieved by **ensuring centre of mass is below the wheel axis** or by using fancy controllers.

3.3 Motion Control

- Kinematic/dynamic model of the robot comes in here
- Model the interaction between the wheel and ground
- Definition of required motion – which motion is required:
 - Speed control
 - Position control
- Control law that satisfies the requirements

Kinematics: Description of mechanical behaviour of the robot for design and control. E.g. how a robot will move given motor inputs.

Mobile robots can move unbounded with respect to their environment:

- No direct way to measure robot's position
- Position must be integrated over time
- Leads to inaccuracies of the position (motion) estimate

Some important definitions:

- **Configuration:** complete specification of the position of every point of the system. Position and orientation. Also called a **pose**.
 - For use: (x, y, z)
- **Configuration space:** space of all possible configurations (can be thought of as one of the robot's frame of references, i.e. we can change the workspace and consider the robot as e.g. a point)
- **Workspace:** the 2D or 3D **ambient** space the robot is in

4 Kinematics

Robot knows how movement relative to centre of rotation (P). This is **not** the same as knowing how to robot moves in the real world.

Two frames we need to worry about:

- Initial frame: the world
- Robot frame: think of yourself as the robot

Robot is at initial frame $\xi_i = (x_i, y_i, \theta_i)$. We want to get to some location, but we can't control (x_i, y_i, θ_i) directly.

The robot can know the speed of wheel i : $\dot{\phi}_i$, steering angle of steerable wheel i : β_i , and speed with which steering angle for wheel i is changing: $\dot{\beta}_i$.

In the robot frame, we have $\xi_r = (x_r, y_r, \theta_r)$. Since in the robot frame $\theta_r = 0$, we simply define $\theta_r = \theta_i$. We have:

$$f(\dot{\phi}_i, \dot{\phi}_r) = [\dot{x}_i \quad \dot{y}_i \quad \dot{\theta}_i]^\top.$$

What we want to do now is:

$$\begin{pmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{pmatrix} = \mathbf{R}_\theta \begin{pmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{pmatrix} \quad \text{where } \mathbf{R}_\theta = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

where $\theta = \theta_i = \theta_r$.

Still, this isn't what we want, we want the reverse of the (inverse-kinematics):

$$\begin{pmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{pmatrix} = \mathbf{R}_\theta^{-1} \begin{pmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{pmatrix} \quad \text{where } \mathbf{R}_\theta^{-1} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2)$$

this tells us how to change the robot's wheels to get somewhere in the world. In our case, $\dot{y}_r = 0$ always.

Constraints and assumptions:

- Movement on a horizontal plane
- Point contact of wheels
- Wheels are not deformable
- Pure rolling: velocity is 0 at contact point

- ...

Differential Drive:

- The **differential drive** is a two-wheeled drive system with independent actuators for each wheel. The name refers to the fact that the motion vector of the robot is the sum of the independent wheel motions, and so turning can be accomplished by rotating the wheels at different speeds. The drive wheels are usually placed on each side of the robot and toward the front.
- Wheels rotate at $\dot{\phi}$
- Each wheel contributes $\frac{r\dot{\phi}}{2}$ to the motion of centre of rotation.
- Speed: sum of two wheels
- Rotation: due to the right wheel is $\omega_r = \frac{r\dot{\phi}}{2l}$ counterclockwise about left wheel, where l is the distance between the wheel and centre of rotation.
- Combining components

$$\begin{pmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{pmatrix} = \begin{pmatrix} \frac{r\dot{\phi}_r}{2} + \frac{r\dot{\phi}_l}{2} \\ 0 \\ \dots \frac{r\dot{\phi}_r}{2l} - \text{what is this? get from notes} \end{pmatrix} \quad (3)$$

now, what is the change in the initial frame of reference?

Tue 17 Jan 2023

Kinematics works in a perfect world, with all our above assumptions satisfied, and where the robot is a point mass. In this case, opened-loop controllers like kinematics work.

Sliding constraint:

- Standard wheel has no lateral motion, we can have steered standard wheels or steered caster wheels
- Move in circle whose centre is on *zero motion line* through the axis

Degree of Mobility: number of degrees of freedom of robot chassis that can be immediately manipulated through changes in wheel velocity.

Degree of Maneuverability: the overall degrees of freedom that a robot can manipulate: $\delta_M = \delta_m + \delta_s$

We may want a robot to be **redundant** if it needs to get to a certain point. If one path is blocked, it can take another path e.g. Think of a manipulator attempting to get its end effector to a certain position. If one trajectory is blocked, it can make another.

5 Manipulators

A **manipulator** is some robot that manipulates (physically alters) something in the real world, but not its own position, at least as a primary goal. Desirable in dangerous, dirty, or dull workspaces.