CMPUT503: Experimental Mobile Robotics

Samuel Neumann

January 23, 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 limited by the range of its sensors
    * Cannot *undo* an incorrect action
    * Impossible to plan ahead
    * Unable to recover from actions which fail silently

- **Deliberative Architecture**
  - Organized by decomposing the required system functionality into *concurrent modules* or components
    * Map building
    * Path planning
    * Navigation
    * ...
  - Problems: overall complexity of the system grows with required components, 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
  - One kind of decomposition is **temporal decomposition** (see below)

- **Subsumption Architecture**
  - Formed using a collection of concurrent behaviours placed in layers
  - The higher-level behaviours always, if triggered, subsume the output of lower behaviours and therefore have **overall control**.
  - Problem: hard to have many layers as goals begin interfering with each other
  - **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.

- **ROS**
  - A *meta* operating system for robots

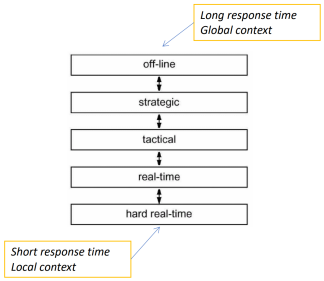

Figure 1: Types of time demands for *temporal decomposition*.

- 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. Each node is typically run as a *separate process*.
- **Not** a programming language
- **Not** a *hard real-time* architecture
- ...

Definition

Temporal Decomposition: A way to decompose a system's required functionality which distinguishes between process that have varying real-time and non-real-time demands.

Definition

Hard Real-Time Requirement: A real-time requirement where, if not met, results in system failure or the system being broken (e.g. driving off a cliff).

2.2 What is ROS?

2.2.1 ROS Nodes

Typically, we try to divide software into different modules, each of which performs a specific task. In ROS, these modules are run over a single or multiple *Nodes*.

Definition

Node: a process that performs some computation.

Nodes are combined together into a graph and communicat with each other using streaming *topics*. Nodes **do not** communicate with each other directly, otherwise we can get into the craziness and complications of the other architectures discussed above.

2.2.2 ROS Topics

Definition

Topic: a named bus over which nodes exchange messages

- Topics have **anonymous** publish/subscribe semantics, i.e. no one knows who published a message or read a message
- Can be *multiple* publisher and subscribers to a topic
- Strongly typed

2.2.3 ROS Messages

Definition

Message: a simple data structure comprising strongly typed fields which is used for one node to communicate to other nodes

3 Locomotion

Definition

Locomotion: enabling a robot to move

Wheeled locomotion is simple, safe, and stable. Key issues for wheeled locomotion include

- 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

Definition

Holonomic: refers to the relationship between controllable and total degrees of freedom of a robot. If the controllable degrees of freedom is equal to the total degrees of freedom, then the robot is said to be **holonomic**. If a robot is holonomic, it can move in any direction in its configuraiton space.

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. hwo 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

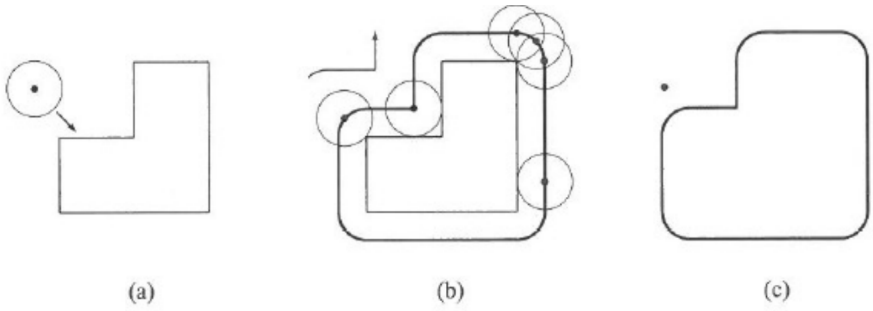

Figure 3.4 (a) The circular mobile robot approaches the workspace obstacle. (b) By sliding the mobile robot around the obstacle and keeping track of the curve traced out by the reference point, we construct the configuration space obstacle. (c) Motion planning for the robot in the workspace representation in (a) has been transformed into motion planning for a point robot in the configuration space.

Figure 2: (a)A circular mobile in the workspace. The configuration space is the set of all points in space that the robot can be in. By re-representing the circular robot as a point, we can (b) adjust the workspace to result in the same configuration space and (c) consider the robot as a single point which simplifies calculations.

Definition

Configuration: a complete specification of the position of every point of the robotic system (position and orientation). This is sometimes also called a **pose**.

Definition

Configuration Space: The space of all possible configurations, which can be thought of as one of the robot's frame of references.

Definition

Workspace: the 2D or 3D **ambient** space the robot is in

4 Kinematics

Definition

Kinematics: computing how a robot will move given motor inputs

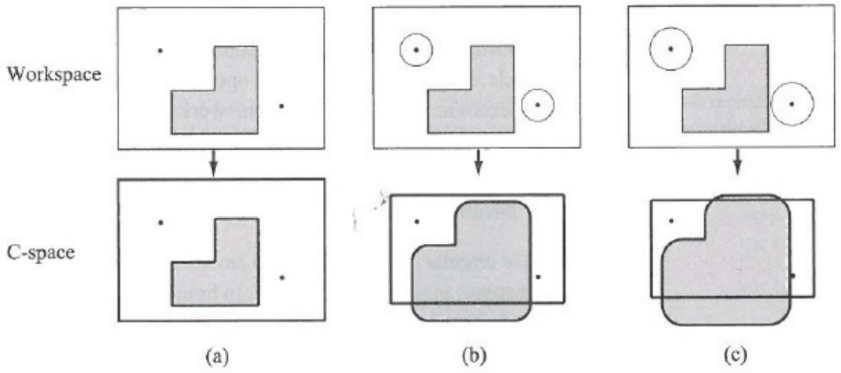

Figure 3.5 The top row shows the workspace and the bottom row shows the configuration space for (a) a point mobile robot, (b) a circular mobile robot, and (c) a larger circular mobile robot.

Definition

Inverse Kinematics: computing how to move motors to get a robot to do what we want

There are two frames that we need to worry about:

- **Initial frame**: the world - $\xi_i = (x_i, y_i, \theta_i)$
- **Robot frame**: think of yourself as the robot - $\xi_r = (x_r, y_r, \theta_r)$

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. Instead, the robot can only control ξ_r.

Consider a wheeled robot where the robot can know the speed of wheel i: $\dot{\phi}_i \forall i = 1, \ldots, n$, steering angle of steerable wheel j: $\beta_j \forall j = 1, \ldots, m$, and speed with which steering angle for wheel j is changing: $\dot{\beta}_j$. These define the forward motion of the robot or the *forward kinematics*.

In the robot frame, we have $\xi_r = (x_r, y_r, \theta_r)$. But, since a wheeled robot is always facing straight forward, we have $\theta_r = 0$. We will simply re-define $\theta_r = \theta_i$. We have the forward kinematics to be:

$$f(\dot{\phi}_i, \ldots, \dot{\phi}_n, \beta_1, \ldots, \beta_m, \dot{\beta}_1, \ldots, \dot{\beta}_m) = [\dot{x}_i \quad \dot{y}_i \quad \dot{\theta}_i]^\top.$$

Inverse kinematics would give us:

$$[\dot{\phi}_i \ \ldots \ \dot{\phi}_n \ \beta_1 \ \ldots \ \beta_m \ \dot{\beta}_1 \ \ldots \ \dot{\beta}_m]^\top = f^{-1}(\dot{x}_i, \dot{y}_i, \dot{\theta}_i).$$

What we want to do now is:

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

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

$$\dot{x}_r = \dot{x}_i \cos(\theta) + \dot{y}_i \sin(\theta) \quad (2)$$

$$\dot{y}_r = -\dot{x}_i \sin(\theta) + \dot{y}_i \cos(\theta) \quad (3)$$

$$\dot{\theta}_r = \dot{\theta}_i \quad (4)$$

Still, this isn't what we want, we want the inverse-kinematics, not the kinematics:

$$\begin{pmatrix} \dot{x}_i \\ \dot{y}_i \\ \dot{\theta}_i \end{pmatrix} = \boldsymbol{R}_\theta^{-1} \begin{pmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{pmatrix} \qquad \text{where } \boldsymbol{R}_\theta^{-1} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5)$$

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

$$\dot{x}_i = \dot{x}_r \cos(\theta) - \dot{y}_r \sin(\theta) \quad (6)$$

$$\dot{y}_i = \dot{x}_r \sin(\theta) + \dot{y}_r \cos(\theta) \quad (7)$$

$$\dot{\theta}_i = \dot{\theta}_r \quad (8)$$

So, if we know the relative changes in x, y, and θ, we can find the global position. How do we know what these values are?

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

$$\begin{pmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{pmatrix} = \begin{pmatrix} \frac{r\dot{\phi}_r}{2} + \frac{r\dot{\phi}_l}{2} \\ 0 \\ \frac{r\dot{\phi}_r}{2l} - \frac{r\dot{\phi}_l}{2l} \end{pmatrix} \tag{9}$$

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.

Thu 19 Jan 2023

Actuator: generates motion or force, usually a motor

For wheels on a robot, we want them synchronized so we typically want to use a servo motor and not a DC motor.