# EE211: Robotic Perception and Intelligence

## Lecture 1 Introduction

Jiankun WANG

Department of Electronic and Electrical Engineering
Southern University of Science and Technology

Undergraduate Course, Sep 2025

# Outline

# Outline

# Course Information

- **Instructor** Jiankun WANG
  - Email: wangjk@sustech.edu.cn
  - Address: Room 709, South Tower, College of Engineering
- **Teaching Assistants**
  - Siyuan Wang, 12012324@mail.sustech.edu.cn
  - Biru Zhang
  - Zhaoqing Zeng
- **Lectures**
  - Address: Room 123, Teaching Building 1
  - Time: Tue 1-2 weekly (1-16)
- **Lab**
  - Address: Room 120, South Tower, College of Engineering
  - Time: Tue 3-4 weekly (1-16)

# Course Description

- Introduce the commonly used sensors and their working principles in robots, including inertial sensing, GPS and odometry, 3D vision for navigation and grasping tasks, visual servoing, and multi-sensor data fusion.
- Introduce the intelligent planning methods in different robot tasks.
- (TBD) Introduce the commonly used robot learning algorithms.

# Course Description

- Understand the working principle of common sensors.
- Understand basic robot motion and path planning algorithms.
- (TBD) Understand basic robot learning algorithms.
- Use robotic perception and intelligence to complete a specific robot task through teamwork.

# Learning Material

- Lectures & Lab & Assignments
- Textbook and Supplementary Readings
  - Siciliano, B., & Khatib, O. (2016). Springer handbook of robotics
  - Lynch, K. M., & Park, F. C. (2017). Modern robotics
  - Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics
  - Bishop, C. M., & Nasrabadi, N. M. (2006). Pattern recognition and machine learning
- Academic Papers from ICRA, IROS, RAL, TRO, TASE, IJRR

# Course Contents

| No. | Dates | Contents |
|---|---|---|
| 1 | Sep.9 | Introduction |
| 2-4 | Sep.16-Sep.30 | Trajectory Generation, Motion Planning |
| 6-7 | Oct.14-Oct.21 | Basic & advanced planning algorithms |
| 8-10 | Oct.28-Nov.11 | Different sensors for perception |
| 11-14 | Nov.18-Dec.9 | Sensor information processing |
| 15-16 | Dec.16-Dec.23 | Robot learning if possible |

# Course Assessment

- Assignment and Sign-in 20%
  - Sign-in or quiz 5%
  - 3 assignments 15%
- Project 30%
  - Conduct real-world robot experiments
- Final Examination 50%
  - Closed-book exam

# Project Description

- Specific task: Conduct real-world robot experiments involving mobile navigation and grasping
- Teamwork: 4? persons in each group
- Evaluation metric: Announced later

# 机器人感知与智能课程（EE211）课程成果展

王建坤

机器人智能与感知重点实验室（rπ Lab）

# 机器人感知与智能课程（EE211）课程成果展

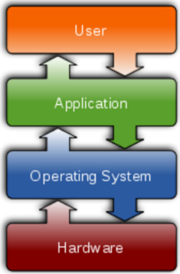
王建坤

机器人智能与感知重点实验室（rπ Lab）

# Outline

# Operating systems

- **operating system**: Manages activities and resources of a computer.
  - software that acts as an interface between hardware and user
  - provides a layer of abstraction for application developers
- features provided by an operating system:
  - ability to execute programs (and multi-tasking)
  - memory management (and virtual memory)
  - file systems, disk and network access
  - an interface to communicate with hardware
  - a user interface (often graphical)




- **kernel**: The lowest-level core of an operating system.

*CSE 391@UW, CMSC 121@Upeen*

# Unix

- The UNIX operating system was born in the late 1960s. It originally began as a one man project led by Ken Thompson of Bell Labs, and has since grown to become the most widely used operating system.
- In the time since UNIX was first developed, it has gone through many different generations and even mutations.
  - Some differ substantially from the original version, like Berkeley Software Distribution (BSD) or Linux.
  - Others, still contain major portions that are based on the original source code.
- An interesting and rather up-to-date timeline of these variations of UNIX can be found at http://www.levenez.com/unix/history.html.

# Linux

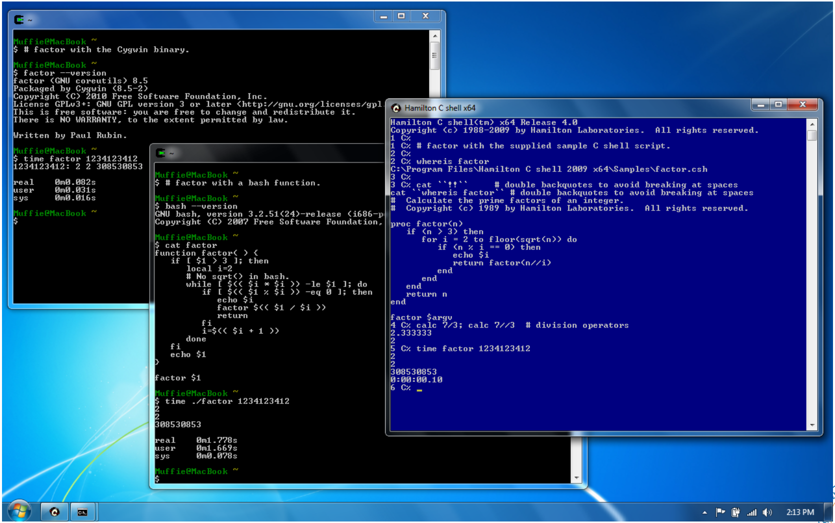
- Linux: A kernel for a Unix-like operating system.
  - commonly seen/used today in servers, mobile/embedded devices, ...
- GNU(a recursive acronym for "GNU's Not Unix!"): A "free software" implementation of many Unix-like tools
  - many GNU tools are distributed with the Linux kernel
- distribution: A pre-packaged set of Linux software.
  - examples: Ubuntu, Fedora
- key features of Linux:
  - open source software: source can be downloaded
  - free to use
  - constantly being improved/updated by the community

# Linux Distributions

# Shell

- Shell: An interactive program that uses user input to manage the execution of other programs.
  - A command processor, typically runs in a text window.
  - User types commands, the shell runs the commands
  - Several different shell programs exist. bash-the default shell program on most Linux/Unix systems. Other shells: Bourne, csh, tsch
- Why should I learn to use a shell when GUIs exist?

# Shell Example



# Graphical User Interfaces (GUIs)

- When you logon locally, you are presented with graphical environment.
- You start at a graphical login screen. You must enter your username and password. You also the have the option to choose from a couple session types. Mainly you have the choice between Gnome and KDE.
- Once you enter in your username and password, you are then presented with a graphical environment that looks like one of the following...

# Command Line Interface

- You also have access to some UNIX servers as well.
  - You can logon from virtually any computer that has internet access whether it be Windows, Mac, or UNIX itself.
- In this case you are communicating through a local terminal to one of these remote servers.
  - All of the commands actually execute on the remote server.
  - It is also possible to open up graphical applications through this window, but that requires a good bit more setup and software.

# Why use a shell?

- faster
- work remotely
- programmable
- customizable
- repeatable

# Shell commands

| command | description |
|---|---|
| exit | logs out of the shell |
| ls | lists files in a directory |
| pwd | print the current working directory |
| cd | changes the working directory |
| man | brings up the manual for a command |

```
$ pwd
/homes/iws/rea
$ cd CSE391
$ ls
file1.txt file2.txt
$ ls -l
-rw-r--r-- 1 rea    fac_cs 0 2016-03-29 17:45 file1.txt
-rw-r--r-- 1 rea    fac_cs 0 2016-03-29 17:45 file2.txt
$ cd ..
$ man ls
$ exit
```

# Relative directories

| directory | description |
|---|---|
| `.` | the directory you are in ("working directory") |
| `..` | the parent of the working directory<br>(`../..` is grandparent, etc.) |
| `~` | your **home** directory<br>(on many systems, this is `/home/`***`username`*** ) |
| `~`***`username`*** | ***username***'s **home** directory |
| `~/Desktop` | your desktop |

# Directory commands

| command | description |
|---|---|
| `ls` | list files in a directory |
| `pwd` | **p**rint the current **w**orking **d**irectory |
| `cd` | **c**hanges the working **d**irectory |
| `mkdir` | create a new directory |
| `rmdir` | delete a directory (must be empty) |

- some commands (`cd`, `exit`) are part of the shell ("builtins")
- others (`ls`, `mkdir`) are separate programs the shell runs

# Linux vs. Windows

- OS does not have to use a graphical interface.
  - The OS itself (the kernel) is incredibly small.
  - The GUI just another application (or set of applications) that can be installed and run on top the existing text-based OS.
- File system differences.
  - Windows typically uses FAT32 or NTFS file systems; Linux typically uses the ext2 or ext3 file systems.
  - Windows lists all drives separately (A:,C:,D:, etc. . . ), with "My Computer" at the highest level; UNIX starts its highest level at "/" and drives can be mounted anywhere underneath it.

# Outline

# Before ROS

- Lack of standards
- Little code reusability
- Keeping reinventing (or rewriting) device drivers, access to robot's interfaces, management of onboard processes, inter-process communication protocols, ...
- Keeping re-coding standard algorithms
- New robot in the lab (or in the factory) → start re-coding (mostly) from scratch

*16-311-Q@CMU*

# Robot Operating System (ROS)

- ROS is an open-source robot operating system
- A set of software libraries and tools that help you build robot applications that work across a wide variety of robotic platforms
- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory and development continued at Willow Garage
- Since 2013 managed by OSRF (Open Source Robotics Foundation)

# ROS Main Features

- The operating system side, which provides standard operating system services such as: hardware abstraction
  - low-level device control
  - implementation of commonly used functionality
  - message-passing between processes
  - package management
- A suite of user contributed packages that implement common robot functionality such as SLAM, planning, perception, vision, manipulation, etc.

# ROS Philosophy

- Peer to Peer: ROS systems consist of many small programs (nodes) which connect to each other and continuously exchange messages
- Tools-based: There are many small, generic programs that perform tasks such as visualization, logging, plotting data streams, etc.
- Multi-Lingual: ROS software modules can be written in any language for which a client library has been written. Currently client libraries exist for C++, Python, LISP, Java, JavaScript, MATLAB, Ruby...
- Thin: The ROS conventions encourage contributors to create stand-alone libraries/packages and then wrap those libraries so they send and receive messages to/from other ROS modules.
- Free & open source, community-based, repositories

# ROS Demo


ROS NAV2
TURTLEBOT 4

# TurtleBot 4 - An Out of This World Demonstration