An Introduction to Mobile Robotics and the e-puck Robot
Outline

• Motivation and general concepts for robotics: autonomy, controllers, control loop

• The e-puck miniature robot
  – General architecture
  – Hardware features and limitations
  – Reality vs. simulation

• Control architectures for mobile robots:
  – A simple taxonomy for robotic control architectures
  – The obstacle avoidance example
Motivation and General Concepts for Robotic Systems
Motivation

• Automation is progressing also in environmental and civil engineering
• Because of the size of the area in which mission must be accomplished (e.g., sensing, monitoring, acting) mobility is key
• Being mobile add complexity and cost at the node level but extend coverage of potentially expensive assets
• Mobile robots become progressively essential tools for automating environmental and civil engineering missions
Open-Loop vs. Closed-Loop Control

• **Open-loop example:** An heating system, programmed to turn on at set times: it does not measure temperature as a form of feedback. Even if the sun is warming the room, the heating system would activate on schedule, wasting energy.

• **Closed-loop example:** An heating system which adjust the heating time as a function of the measured temperature. If the sun is warming the room, the heating system will be activated less often than in a rainy day.
What is a Controller?

• [From wikipedia] In control theory, a **controller** is a device which monitors and affects the operational conditions of a given **dynamical system**. The operational conditions are typically referred to as output variables of the system which can be affected by adjusting certain input variables.

• Example, the heating system of a room (closed-loop):
  – Sensing: temperature probe
  – Controller: thermostat
  – Actuation: heater
  – Dynamical system: room
What is a Controller?

• In this course (in **embedded systems**), a **controller** is a piece of software which monitors and affects the operational conditions of a given **dynamical system** consisting of the device hardware and the environment.

• Example, the heating system of a room (closed-loop):
  – Sensing: temperature probe
  – Controller: algorithm running on a programmable thermostat (measure temperature and control the heater)
  – Actuation: heater
  – Dynamical system: room + programmable thermostat (e.g., microcontroller-based)
Perception-to-Action Loop

- Perception
- Computation (controller)
- Environment
- Action

sensors  actuators

Note: real-time aspect emphasized!
Autonomy

• Different levels/degrees of autonomy
  – Energetic level
  – Sensory, motor, and computational level
  – Decisional level

• Needed degree of autonomy depends on task/environment in which the unit has to operate

• Environmental unpredictability is crucial: robot manipulator vs. mobile robot vs. sensor node
Autonomy – The Impact of Controllable Mobility

State of the Art in Mobile Robotics

Task Complexity

Human-Guided Robotics

Autonomous Robotics

Research

Industry

Autonomy
The e-puck miniature robot
A few selected and re-elaborated slides from:

Microinformatique

Introduction to the e-puck robot

Francesco Mondada
Robotics Systems Laboratory
IMT - STI - EPFL
The e-puck Mobile Robot

Main features

- Cylindrical, Ø 70mm
- dsPIC processor
- Two stepper motors
- Ring of LEDs
- Many sensors:
  - Camera
  - Sound
  - IR proximity
  - 3D accelerometer
- Li-ion accumulator
- Bluetooth wireless communication
- Open hardware (and software)
The e-puck Open Hardware License

The specifications of the e-puck mobile robot are "open source hardware". You can redistribute them and/or modify them under the terms of the e-puck Robot Open Source Hardware License as published by EPFL. You should have received a copy of the EPFL e-puck Robot Open Source Hardware License along with these specifications; if not, write to the Ecole Polytechnique Fédérale de Lausanne (EPFL), Industrial Relations Office, Station 10, 1015 Lausanne, Switzerland.

These specifications are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. For more details, see the EPFL e-puck Robot Open Source Hardware License.
Hardware Overview
e-puck Overview

IR receiver
(remote control)

Accelerometer

Programming and
debug connector

ON-OFF

microphones

Wheels with
stepper motor

Speaker

Reset

Mode selector

RS232

Ring of LEDs

IR proximity
sensors

CMOS camera

Li-Ion accumulator
e-puck Block Schema

Computation and memory
Communication
Actuators
Sensors
Light and Proximity Sensors

- Light and proximity sensors incorporated in the same electronic component.
- Khepera II has essentially the same components of e-pucks for this functionality.
- e-puck sensor numbering slightly different from Khepera II (e.g., IR7 and IR0 correspond to Sensor 2 and 3), but same layout.

Khepera II in front of a 50 W bulb

Khepera II in front of white paper
PIC/dsPIC Family

from www.microchip.com
DSP and Specialized Variants

- dsPIC is a family of chips combining microcontroller and Digital Signal Processor (DSP) structure and features
- For each dsPIC family member there are three variants: general purpose, motor control and power conversion, sensor processor

<table>
<thead>
<tr>
<th>Device</th>
<th>Pins</th>
<th>Program Memory</th>
<th>Instructions</th>
<th>SRAM Bytes</th>
<th>EEPROM Bytes</th>
<th>Timer 16-bit</th>
<th>Input Capture</th>
<th>Output Capture</th>
<th>Codec Interface</th>
<th>UART</th>
<th>SPI™</th>
<th>PC™</th>
<th>CAN</th>
<th>I/O Pins (Max.)</th>
<th>Packages</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsPIC30F3014</td>
<td>40/44</td>
<td>24K</td>
<td>8K</td>
<td>2048</td>
<td>1024</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>—</td>
<td>13 ch</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>30</td>
<td>PG, PT</td>
</tr>
<tr>
<td>dsPIC30F4013</td>
<td>40/44</td>
<td>48K</td>
<td>16K</td>
<td>2048</td>
<td>1024</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>AC’97, I2S</td>
<td>13 ch</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>30</td>
<td>PG, PT</td>
</tr>
<tr>
<td>dsPIC30F5011</td>
<td>64</td>
<td>68K</td>
<td>22K</td>
<td>4096</td>
<td>1024</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>AC’97, I2S</td>
<td>16 ch</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>52</td>
<td>PT</td>
</tr>
<tr>
<td>dsPIC30F6011A</td>
<td>64</td>
<td>132K</td>
<td>44K</td>
<td>6144</td>
<td>2048</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>—</td>
<td>16 ch</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>52</td>
</tr>
<tr>
<td>dsPIC30F6012A</td>
<td>64</td>
<td>144K</td>
<td>48K</td>
<td>8192</td>
<td>4096</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>AC’97, I2S</td>
<td>16 ch</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>52</td>
</tr>
<tr>
<td>dsPIC30F5013</td>
<td>80</td>
<td>69K</td>
<td>22K</td>
<td>4096</td>
<td>1024</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>AC’97, I2S</td>
<td>16 ch</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>66</td>
</tr>
<tr>
<td>dsPIC30F6013A</td>
<td>80</td>
<td>132K</td>
<td>44K</td>
<td>6144</td>
<td>2048</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>—</td>
<td>16 ch</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>66</td>
</tr>
<tr>
<td>dsPIC30F6014A</td>
<td>80</td>
<td>144K</td>
<td>48K</td>
<td>8192</td>
<td>4096</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>AC’97, I2S</td>
<td>16 ch</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>66</td>
</tr>
</tbody>
</table>
Memory and Computing Limitations in Embedded Systems
Data storage

- Three types of data storage:
  - **Primary storage:**
    - Directly accessible to the microprocessor
    - **Fastest** data storage (1-10 ns), but also the **smallest** (MB order)
    - Generally **volatile**
    - Examples: RAM, cache memory, registers, ROM (non-volatile)
  - **Secondary storage:**
    - Accessed through input/output channels
    - **Large** (GB order), but quite **slow** (1-10 ms)
    - Generally **non-volatile**
    - Often **removable**
    - Examples: hard disks, USB sticks, floppy disks, etc…
  - **Tertiary storage:**
    - **Very large** size (TB order)
    - **Very slow** access (5-60 s)
    - Generally **non-volatile**
    - Examples: tape libraries, optical jukeboxes
Primary storage

- There are generally three levels of primary storage:
  1. **Processor registers**
  2. **Cache**
  3. **Main memory**

- **Main memory:**
  - Connected to the CPU via a memory bus
  - **Von Neumann architecture:** single chip memory for programs and data
  - **Harvard architecture:** separated program and data memory
  - **Program memory:** can afford to be sequential access and read-only during execution.
  - **Data memory:** random access
Memory on the e-puck

- **Primary storage**: 8 KB of RAM (random-access, volatile) for data and 144 KB of Flash Memory (sequential-access, non-volatile) for programs (Harvard Architecture)
- All memory **inside the dsPIC**: microcontroller!
- **Secondary and tertiary storage**: none (but one could develop an extension)
- **Permanent storage of data**: communication with the base station, which will take care of it
- Please note the **very small amount** of RAM (8 KB) available on the e-puck robot!
- Memory is a **very precious resource** in embedded systems!
E-puck camera: 640x480 pixels, RGB color (3 bytes)

The microcontroller has not enough memory available for storing even a single image.

To be able to acquire the camera information, the image rate has to be reduced and the resolution too.
An Example with the On-Board Camera

- Note that the on-board computation/processing resources are also not following the data flow:
  - Pixels H x V x RGB x fps
  - $640 \times 480 \times 3 \times 30 = 27$Mbytes/second
  - A dsPIC can execute max 15MIPS (millions of instructions/second, i.e. 1 instruction takes at least 66 ns)
Solution: Image Processing

- By using **downsampling**, you can reduce the size of the image while retaining useful information.
- Typically, we can acquire a 40x40 downsampled color image at 4 frames per second.
- Without color, we can go up to 8 frames per second.
Simulation vs. Reality
Simulated and Real e-puck

Simulated e-puck (Webots)
- sensor- and actuator-based
- noise, nonlinearities of S&A reproduced
- kinematic (e.g., speed, position) and dynamic (e.g., mass, forces, friction)

Real e-puck
Simulated and Real e-puck

- Submicroscopic, high-fidelity simulator Webots looks very powerful and realistic but:
  - Manufacturing heterogeneities not reproduced (e.g., all sensor of a certain type are the same, all the robots as well).
  - Noise distributions are typically uniform or Gaussian
  - Sensor field of view simplified (e.g., ray instead of cone)
  - Real-time emulation very crudely approximated
  - Limitation in computational resources and internal electrical/computational architecture not reproduced
  - World physics approximated (e.g., geometry, communication channel, fluid dynamics) or not reproduced (e.g., chemical dispersion, thermal dissipation, etc.)
  - …
Example: Real-time Emulation

Delays are everywhere!

sensors

analog-digital conversion time

processing time

actuator delay

actuators

Environment

propagation time

Perception

tsampling rate
Real-time Emulation

R = possible reality situation; S: possible simulation parametrization

R1
- Read and convert sensor data
- Normalize data
- Filter data
- Update actuators
- 100µs 60µs 135µs 120µs

R2
- Read and convert sensor data
- 30ms

S1
- Read and convert sensor data
- Normalize data
- Filter data
- Update actuators
- 32ms (sim step k-1) 32ms (sim step k) 32ms (sim step k+1)

S2
- Read and convert sensor data
- Normalize data
- Filter data
- Update actuators
- 64ms (sim step k-1) 64ms (sim step k) 64ms (sim step k+1)
A Simple Taxonomy for Control Architecture in Mobile Robotics
Perception-to-Action Loop for a Mobile Robot

- sensors

Controller:
- Reactive vs. Deliberative
- Proximal vs. Distal

- actuators

Environment
Important Notes

• Although the following examples are focused on mobile robotics, the architecture considerations are valid for any system endowed with \( m \) sensors and \( n \) actuators, with \( m \geq n \) (typically \( m \gg n \))

• Effects on mobile robots are easier to visualize because of actuator action results in continuous movement in the physical space

• Alternative example: sprinkler system for irrigation (e.g., single output with varying amplitude) endowed with sensors for soil moisture, air humidity, air temperature, etc.
Reactive vs. Deliberative Architectures

• Reactive controller:
  – 1 perception-to-action loop horizon
  – No planning, no history stored

• Deliberative controller
  – Multiple perception-to-action loop horizon
  – Planning and history exploitation

• Reactive-deliberative boundary zone:
  – Short history, short look-ahead horizon
  – A few state variables and little memory
Proximal vs. Distal Architectures

• Proximal:
  – close to sensor and actuators; no control hierarchy or layering
  – high flexibility in shaping the behavior by changing parameters and keeping the structure fixed
  – light architecture, fast execution time
  – works well when few resources available
  – difficult to engineer in a “human-guided” way
Proximal vs. Distal Architectures

• Distal:
  – farther from sensor and actuators; some control hierarchy or layering possible
  – less flexibility in shaping the behavior (behavioral module or basic behavior definition; flexibility only in the module “wiring” typically)
  – not always light architecture
  – easier to engineer in a “human-guided” way because of the existence of modules (often hand coded)
Selected Reactive Architectures for Mobile Robots and their Application to Obstacle Avoidance
Overview

• Five “classical” examples of reactive control architecture for solving the same problem: obstacle avoidance.
• Two proximal: Braitenberg and Artificial Neural Network
• Three distal: Rule-based, Subsumption and Motor Schema, both behavior-based
Ex. 1: Braitenberg’s Vehicles

- Work on the **difference** (gradient) between sensors
- Originally **omni-directional** sensors; works also with **directional** sensors (sharper, potentially discontinuous differences at the sensory level -> more jerky movement)
- Originally: **light** sensors
- + excitation, - inhibition; **linear** controller (out = signed coefficient * in)
- Symmetry axis along main axis of the vehicle (----)
- See also lecture week 8 and lab 8
Ex. 1: Braitenberg’s Vehicles

With proximity sensors:
- $m = 8$
- local sensors $\rightarrow$ must add a bias otherwise no stimulus

**Linear summing & normalizing operator**

$$O_i = Kf(x_i) = Kx_i$$

$$x_i = \sum_{j=1}^{m} w_{ij} I_j + I_0$$

**With proximity sensors**: inhibitory conn. (negative weight) excitatory conn. (positive weight)

**Weight**

**Output**

**Input**
Ex. 2: Artificial Neural Network

**Neuron (output layer)** with sigmoid transfer function $f(x)$

$O_i = Kf(x_i)$

$$f(x) = \frac{2}{1 + e^{-x}} - 1$$

$$x_i = \sum_{j=1}^{m} w_{ij} I_j + I_0$$

$K$: normalization constant

**Synaptic weight**

$w_{ij}$

**Input**

$I_j$

**Output**

$O_i$

**Inhibitory conn.**

**Excitatory conn.**

EPFL

Disal
Ex. 3: Rule-Based

Rule 1:
if (proximity sensors on the left active) then
   turn right

Rule 2:
if (proximity sensors on the right active) then
   turn left

Rule 3:
if (no proximity sensors active) then
   move forwards
Subsumption Architecture

• Rodney Brooks (1986)
• Precursors: Braitenberg (1984), Walter (1953)
• Behavioral modules (basic behaviors) represented by Augmented Finite State Machines
• Response encoding: predominantly discrete (rule based)
• Behavioral coordination method: competitive (priority-based arbitration via inhibition and suppression)
Subsumption Architecture

Classical paradigm (serial); emphasis on deliberative control

Subsumption (parallel); emphasis on reactive control
Augmented Finite State Machine

**Behavioral Module**

- **Input lines**
- **Input**
- **Inhibitor**
  - **Inhibitor**: block the transmission
- **Reset**
- **Suppressor**
  - **Suppressor**: block the transmission and replace the signal with the suppressing message
- **Output lines**
Ex. 4: Behavior-Based with Subsumption

Concrete implementation within basic behaviors:
Obstacle avoidance: Braitenberg without bias, rule-based, etc.
Wander: bias on motors (straightforward motion), random walk, etc.
Motor Schemas

• Ronald Arkin 1987, Georgia Tech
• Precursors: Arbib (1981), Khatib (1985)
• Parametrized behavioral libraries (schemas)
• Response encoding: continuous using potential field analog
• Behavioral coordination method: cooperative via vector summation and normalization
Ex. 5: Behavior-Based with Motor Schemas

Note: each motor schema generate an output vector that then gets summed up and normalized for controlling the actuators
Visualization of Vector Field for Ex. 5

**Avoid-obstacle**

Vector = [magnitude, direction]

\[ V_{\text{magnitude}} = \begin{cases} 
0 & \text{for } d > S \\
\frac{S - d}{S - R} G & \text{for } R < d \leq S \\
\infty & \text{for } d \leq R
\end{cases} \]

\( S \) = obstacle’s sphere of influence
\( R \) = radius of the obstacle
\( G \) = gain
\( D \) = distance robot to obstacle’s center

\( V_{\text{direction}} = \) radially along a line between robot and obst. center, directed away from the obstacle
Move-to-goal

Vector = [magnitude, direction]

\[ V_{\text{magnitude}} = \text{fixed gain value} \]

\[ V_{\text{direction}} = \text{towards perceived goal} \]
Visualization of Vector field for Ex. 5

Avoid-obstacle + move-to-goal

Linear superposition (vectorial weighted sum)
Ex. 5: Issue with Motor Schemas

To avoid getting stuck in local minima of the vector field (typical problem of vector field approaches)

Alternative more complex approach: use harmonic potential functions (explicitly designed for not generating local minima)
Conclusion
Take Home Messages

• The complexity of a controller depends on the degree of autonomy, mission to accomplish, and environmental conditions.

• The e-puck uses a dsPIC microcontroller (programmable in C), has a rich sensory set, actuation capabilities, and bidirectional wireless and wired links.

• Computation on board has limits: memory and computing resources limits data processing and in turn control algorithms.

• Robotic controllers can be classified along two main axes: proximal vs. distal, reactive vs. deliberative.

• A given overall behavior of the device can be obtained with different control architectures.

• Controllers are characterized by parameters and a structure (architecture).
Additional Literature – Week 9

Manuals and technical documentation
• MPLAB C30 C Compiler User’s Guide
• dsPIC datasheet
• dsPIC30F Programmer’s Reference Manual
• e-puck website: http://www.e-puck.org/

Articles

Books