E-Puck – a mobile robot

Stephan Zibner
Outline

• Introducing the E-Puck
• Communication
• Motors
  • Odometry
• Sensors
  • Calibration
• Case study Braitenberg vehicle

Always intermingled with MATLAB knowledge!
Introducing the E-Puck
Introducing the E-Puck

Reset button

On/off switch

Battery
Introducing the E-Puck

Two wheels with separate motors and encoders
Introducing the E-Puck

Accelerometer

Three microphones

VGA camera

Eight active infrared sensors
Introducing the E-Puck

- Power LED
- Communication LED
- Surrounding lights
- Body light
- Speaker
Introducing the E-Puck

On-board camera is mainly useless!
Communication

serial
Communication in MATLAB

- Get handle with `kOpenPort`
  - you get a confirmation message
  - on Windows: insert proper COM port first
- Only on Linux: store handle in variable
- Close serial port with `kClose`

OR

- Use initialization scripts
  - `setupEPuckRobot.m`, `closeEPuckRobot.m`
Motors

- Two separately controllable motors
- Encoders on both wheels
- Unit of velocity and distance: encoder pulses
- For E-Puck: 0.13 mm per pulse
- That's 7.692307692 pulses per mm
- Velocity is specified as pulses per second
Motors in MATLAB

- **kGetSpeed, kSetSpeed**
  - Get or set the robot's velocity
  - No timing parameter, executed ad infinitum

- **kGetEncoders, kSetEncoders** (Linux only)
  - Get or set the robot's integrated encoder values
  - Should be resetted once in a while

- **kStop**
  - Never forget to set the robot's velocity back to zero!
Motors and odometry

\[ x' \]
\[ y' \]
\[ \Delta x \]
\[ \Delta y \]
\[ \phi' \]
\[ \phi_{\text{new}} \]
\[ x_{\text{old}} \]
\[ y_{\text{old}} \]
\[ x_{\text{new}} \]
Motors and odometry

\[ \Delta x = x' \cdot \cos(\phi) - y' \cdot \sin(\phi) \]

\[ \Delta y = x' \cdot \sin(\phi) + y' \cdot \cos(\phi) \]

\[ \phi_{\text{new}} = \phi + \phi' \]
Motors and odometry in MATLAB

- Call `integrateForwardKinematics` with
  - Travelled distance for both wheels
  - Last position and heading returned by this function
  - Wheel distance in mm
- Return value contains
  - New x,y coordinates
  - New heading direction
Sensors

- 8 infrared sensors
  - All around the robot's body
  - Passive mode: measure ambient light
  - Active mode: measure reflected infrared light
- 3 microphones
  - At -90, 90, and 180 degrees
- PAL camera
  - Maximum resolution of 720 x 576
  - External battery, not part of e-puck
Sensors in MATLAB

- **kProximity, kAmbient**
  - Active and passive mode of IR sensors
  - All eight sensors are returned for each call
- **kGetMicrophones**
  - All three microphone amplitudes are returned
- **Image acquisition toolbox (Windows only)**
  - `vid = videoinput('winvideo', 1, 'YUY2_640x480');`
  - Initialization...
  - `trigger(vid); getdata(vid);`
Sensor Calibration

• IR sensors are not identical
  • Manufacturing errors
  • Vertical alignment

• We provide a simple solution
  • Determine weights for all sensors with GUI
  • Weaken or strengthen specific sensors
Sensor Calibration in MATLAB
Case Study Braitenberg

Braitenberg 1986
Case Study Braitenberg

• Open communication
• In a loop
  • Call `kGetAmbient` to receive current sensor values
  • Take only the front sensors (e.g. 1 and 8)
  • Translate from sensor signal to motor strength
  • Call `kSetSpeed` to set a new velocity for each wheel
  • Insert a pause
• Close communication
The Wrapup

• Learning by doing
• We provide you with
  • Cheat sheet for constants
  • Cheat sheet for functions
  • Templates
    – Odometry
    – Sensor calibration
    – Getting camera images
• If you experience any difficulties, just ask!