Building and Programming Home Robots with Raspberry Pi and Python

Prof. Dr.-Ing. Gerald Schuller
Ilmenau University of Technology
Institute for Media Technology
gerald.schuller@tu-ilmenau.de

Introduction

- It started when the bumper sensors of my **Roomba Vacuum** robots failed.
- First I fixed them (there where dust accumulating in a light breaker switch).
- Since that problem repeated, and the repair was tedious, I looked for a more permanent and **more fun solution**:
- The Roomba robots have a (sometimes hidden) **serial interface**, for which documentation can be found online.
- Raspberry Pi as a **serial interface among its GPIO Pins**
- The corresponding **Python library is "serial"**
- So we have all we need :-)

My Approach

- Replace the internal control with a **Raspberry Pi** 2b
- Needs **low power** consumption, for power supply from the serial interface control over the serial interface,
- Replace the bumper sensors by a **cheap USB webcam** and video signal processing.
- Alternatively, instead of the USB webcam we could use the **smaller Raspberry Pi camera**

The challenging part: The video signal processing

- Goal:
  - Detect an object in the moving path of the robot (possible collision detection),
  - if an object is detected, change the path.
- Problem: video processing, particularly object recognition, is usually very **computational complex**.
- Here it has to fit on the Raspberry Pi 2b.

The Video processing Library

- For the video processing, I use Python OpenCV.
- In Linux (e.g. Raspbian), it can be installed with the command
  - sudo apt install python3-opencv
- This also works on a Raspberry Pi!

Processing the Video Signal

- **Approach: Simplify** algorithms until they fit and still reasonably work.
- Take “**Discrete Cosine Transform**” (DCT) "**Cepstrum**", known from **speech processing**, apply it to the video.
- **Low complexity** because it uses an efficient dct function from library **scipy.fftpack**
- Principle:
  - A=(scipy.fftpack.dct(currframe,axis=1))
  - A=np.abs(scipy.fftpack.dct(A,axis=0)) #magnitude here removes position information
  - #take the "inverse" dct (identical to forward dct):
  - #Keep just inner 100 rows and cols for "cepstrum":
  - A=(scipy.fftpack.dct(A[0:100,0:100],axis=1))
  - A=np.abs(scipy.fftpack.dct(A,axis=0))

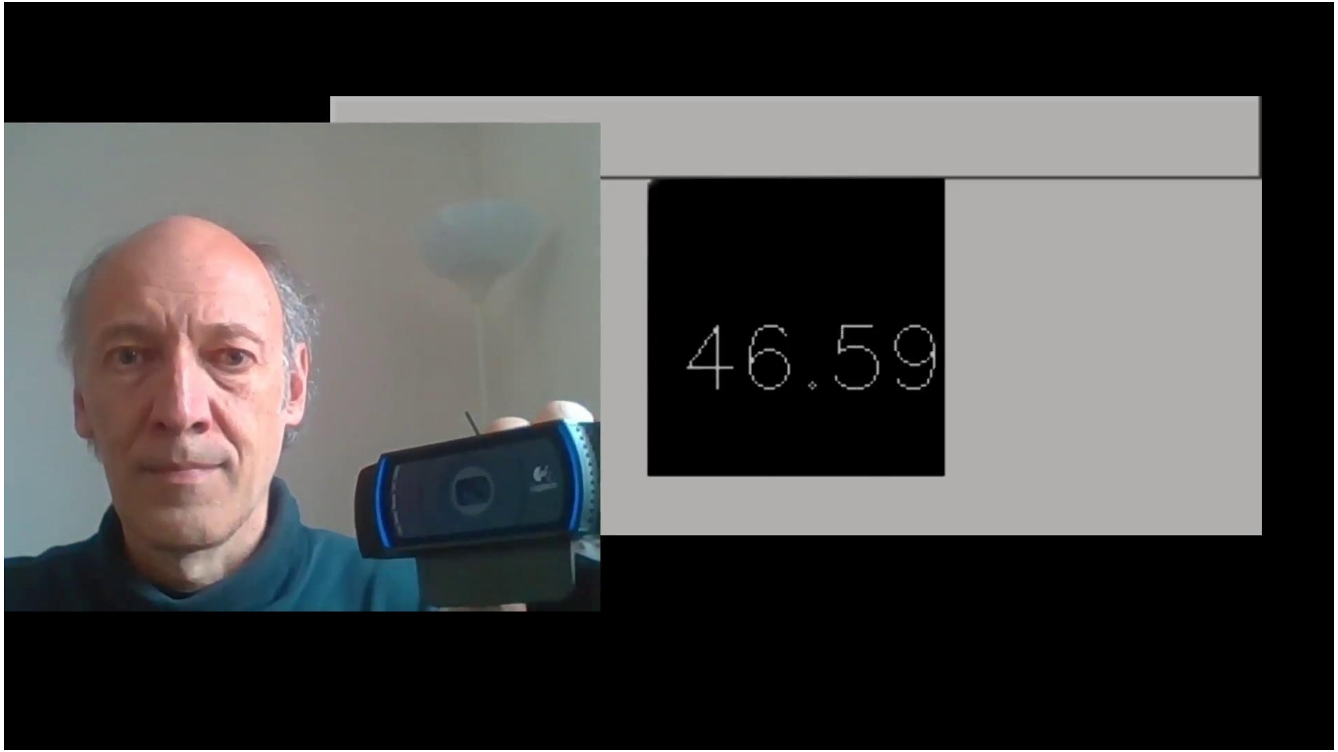
Processing the Video Signal

- This processing outputs a **smaller, faster to process**, 100x100 pixel image
- Objects are transformed into **“blobs” of similar size in the upper left hand corner!**
- This allows us to easily **estimate the change of the average size** of objects in the camera view.
- This is done by computing the “**center of mass**” (equilibrium point of an object) of the cepstrum image,
- and its **distance from the upper left hand corner**.
- If this **value increases quickly**, the robot is **nearing an obstacle.**

Video Demo of this Processing

- I made a small **Python demo for the Cepstrum computation**.
- It processes a webcam image and **display the life Cepstrum** on small window, and displays the **distance of the resulting center of mass from the upper left hand corner** as an overlaid number.
- Observe this number **becoming larger when I move my hand towards the camera.**
- Start with command: `python3 videorecdisp_dctcepstrum.py`

Video Demo of this Processing

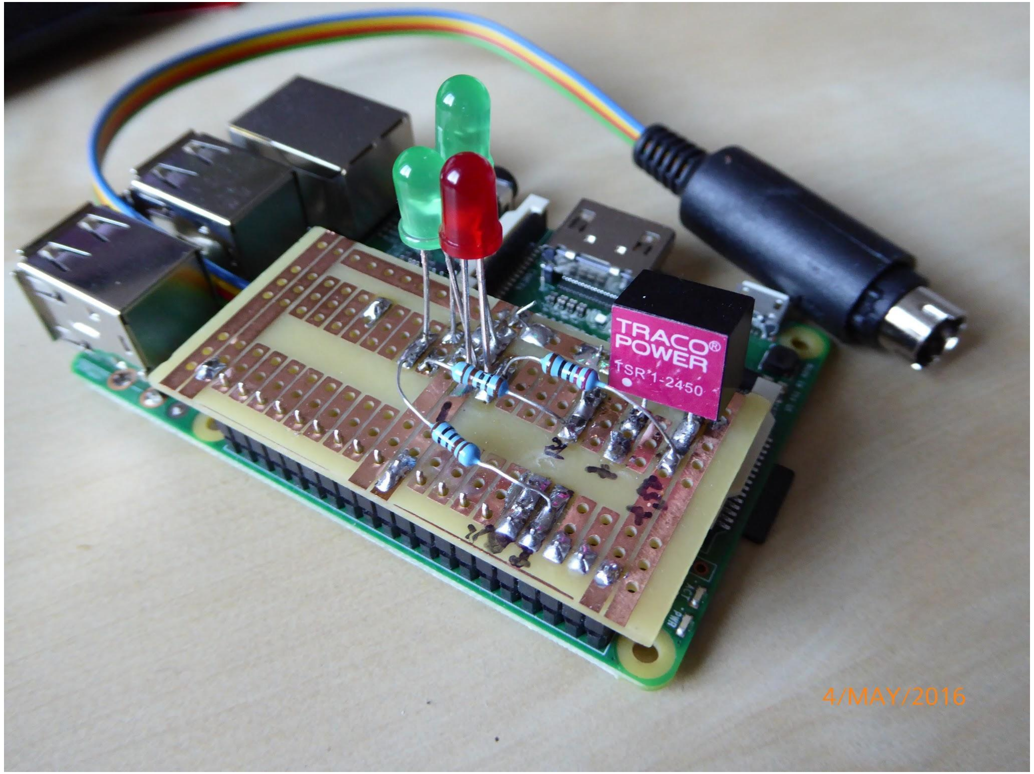


Video Link

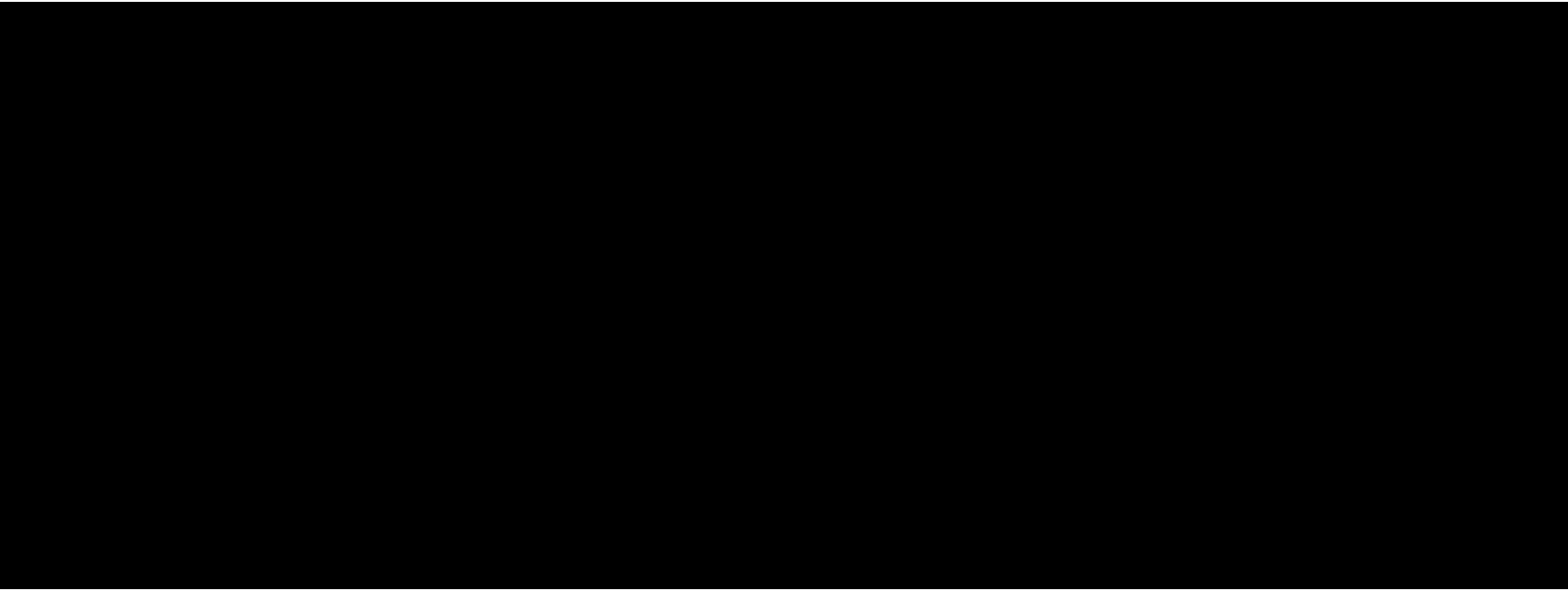
Raspberry Pi - Roomba Interface

- There is **no commercial interface** board for the Raspberry Pi to the Roomba. Hence we need to do it yourself.
- It needs to **convert 5V logic level on the Roomba to and from 3.3V logic level** on the Raspberry Pi, using resistors as voltage dividers.
- It needs to **convert the 12V supply voltage** from the Roomba **to the 5V** needed by the Raspberry Pi, using a highly efficient **switching voltage converter.**
- Connectors: The Roomba has a standard round DIN connector, the Raspberry Pi has GPIO connector
- https://www.tu-ilmenau.de/en/applied-media-systems-group/research/previous-projects/raspberry-pi-serial-interface-for-roomba-robots/

Raspberry Pi - Roomba Interface



Py-Roomba in Action



[Video 1]

[Video 2]

Two-Legged Robots

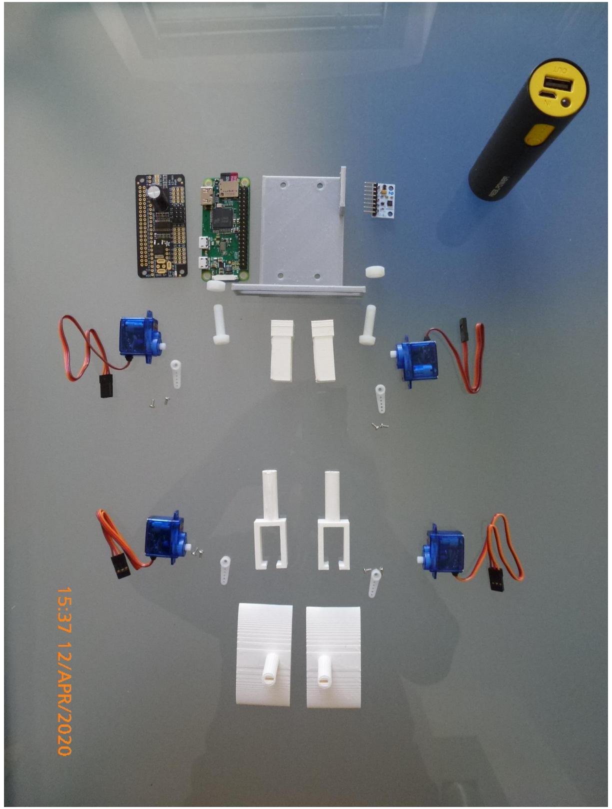
- I thought It would be practical if the vacuum robot **could walk stairs**
- -> **2-legged robot.**

Two-Legged Robots

Hardware:

- The 2-legged robot needs small **servo motors for its joints**.
- A 2-legged robot needs the **fewest number of servo motors (4)**,
- but is more difficult to balance, for which we need an **accelerometer**.
- Hip servo motors: **forward-backward**
- Knee servo motors: **sideways tilt**.
- A **Raspberry Pi Zero W** is used because it is smaller and lightweight (ca. 15 eur)
- Power supply: small **usb power bank**
- Body parts: from a **3D printer** (service e.g. from Conrad Elektronik), ca. 5 eur per piece.

Hardware, 3D Printed Parts



Interfaces

- For servo motors, commercial interfaces are available for the Raspberry Pi, for instance from Pimoroni.
- Python library: **adafruit_servokit** for the servos
- Servo Motors: e.g. Reely Analog- Servo S0008 from Conrad
- 3- axis digital accelerometer sensor,
- uses an **I2C interface**, which the Raspberry Pi fortunately also has.
- Python library: **smbus**
- See also:
- https://www.tu-ilmenau.de/en/applied-media-systems-group/research/project-robotics/

The difficult part: The Algorithms

Two parts:

- The upright balancing
- The foot steps

The Balancing

- a **"Proportional-Integral-Differential" (PID)** controller.
- This uses the information from the accelerometer (which indicate where is "down).
- It balances the robot upright, with some forward tilt to balance the relatively heavy battery pack.
- The tricky part here are the **coefficients of the controller**.
- They can be obtained **experimentally** (that takes some time but is fun :-)).
- An advanced possibility is to use **numerical optimization or reinforcement learning** (a machine learning approach) for it.

The Foot Steps

- first make the robot oscillate somewhat horizontally (left-right),
- by shifting the balancing goal left to right and back, using the knee servos.
- This lifts the opposite foot for movement.

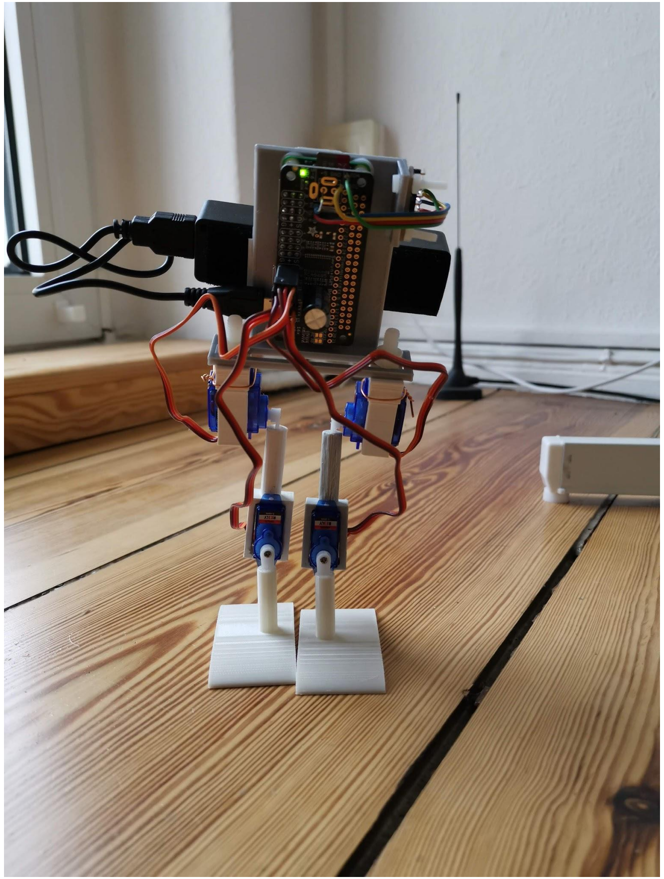
2 Modes:

- This oscillation can be given by a **periodic function**,
- Or it can be an excitation with a **random number generator (noise)** which makes the robot oscillate at its natural oscillation frequency. This is more energy efficient and leads to faster steps.

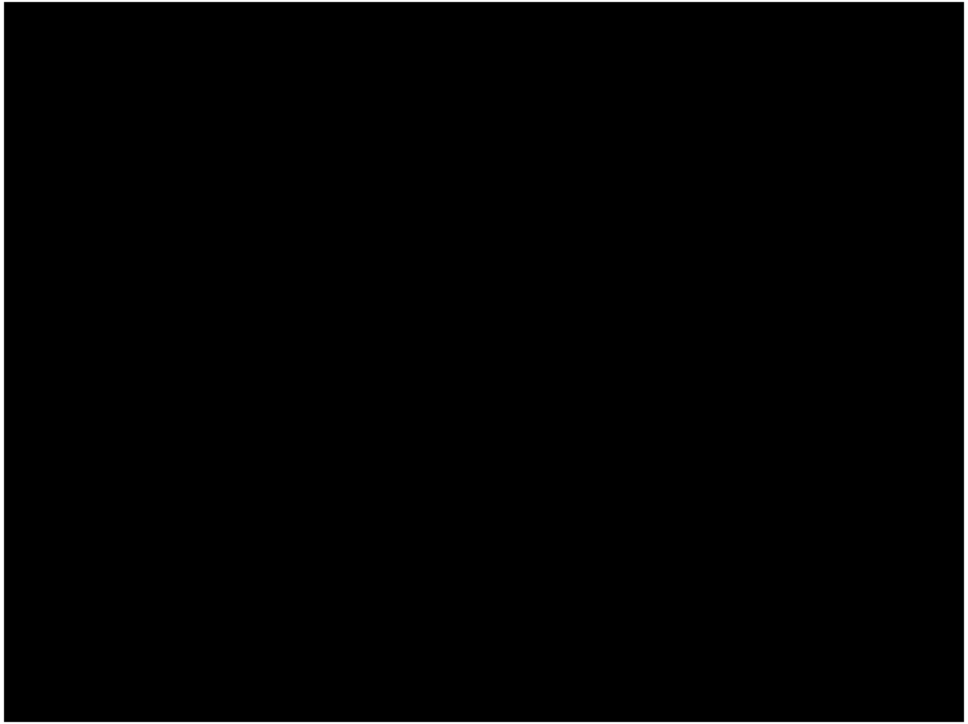
The Forward Move

- Each time the **robot moves to one side** (information from the accelerometer),
- the **opposite foot is moved forward** at constant speed,
- and the foot on the **same side is moved backward** at the same speed.

The Robot

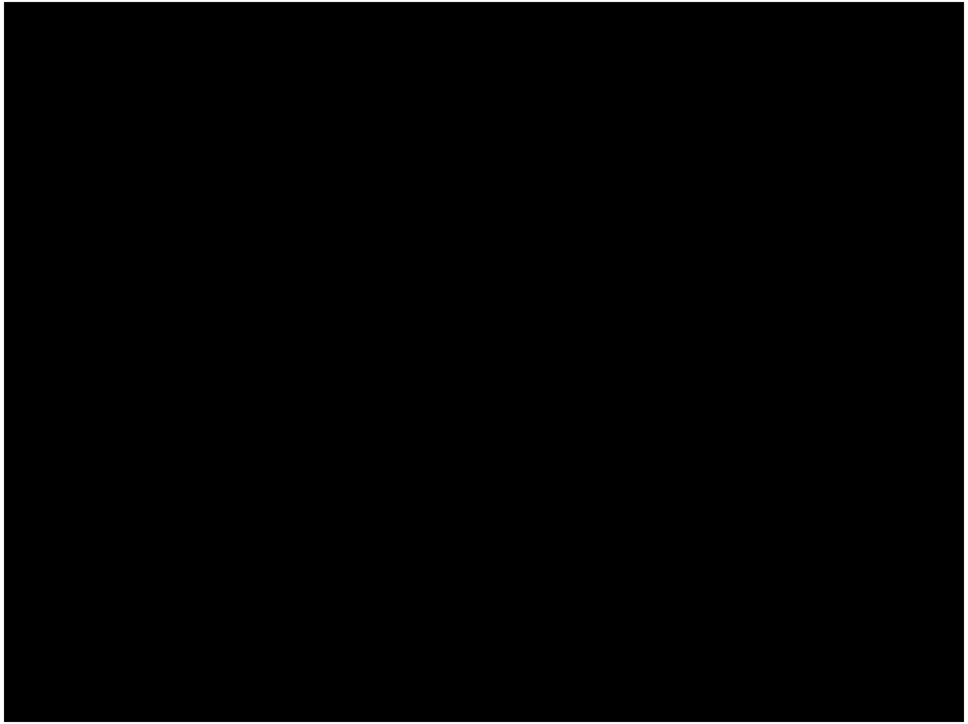


Robot Slow Walking



Video Link

Robot Fast Walking, Learning



[Video Link]

Conclusions

- With simplifications, even computer vision and machine learning algorithms can run on a Raspberry Pi in Python
- The Raspberry Pi allows easy connections to external hardware and interfaces
- This makes DIY building home robots doable

Links

See also: https://www.tu-ilmenau.de/en/applied-media-systems-group/

For Roomba robot:

- https://www.tu-ilmenau.de/en/applied-media-systems-group/research/previous-projects/raspberry-pi-serial-interface-for-roomba-robots
- https://github.com/TUIlmenauAMS/AutonomousRobotsWithCamera

2-legged Robots:

- https://www.tu-ilmenau.de/en/applied-media-systems-group/research/project-robotics/

Contact: gerald.schuller@tu-ilmenau.de