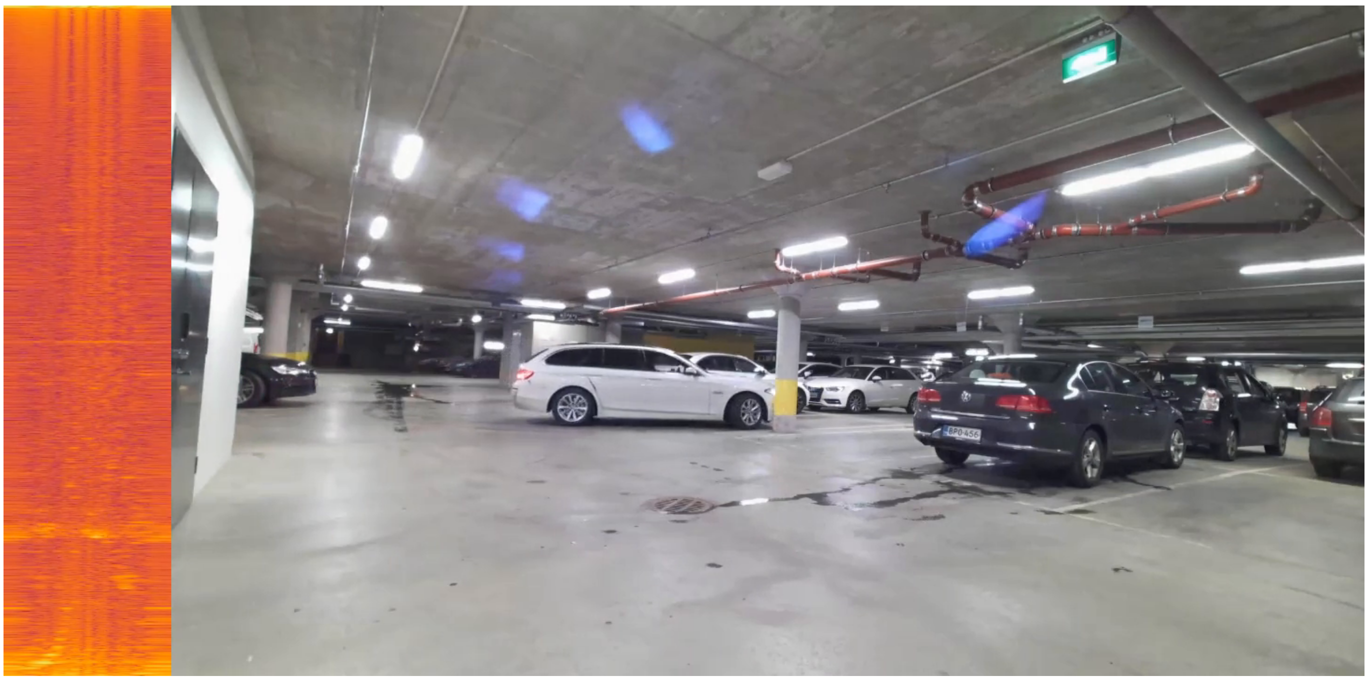
README

Table of content

Parking dataset

This dataset is collected inside a parking lot with 60 GHz radar. There are people and cars entering and leaving the parking lot.



amplitude (folder)

This folder contains the amplitude information.

```
-amplitude
 |-amplitude00
 |-amplitude01
 |-ampltiude02
```

- Text files that contain radar data from 60 GHz radar
- Each row is a single frame from the radar

```
130.12345,156.56712,120.12423,.. // 178200 elements each row
```

- Frame has **990** rows and **180** columns
- Framerate is **50 Hz**
- Data is in polar coordinate system
    - Range information in rows (min range 0.5 m, max range 50 m)
    - Angle information in columns (min angle -90 deg, max angle 90 deg)

In python the data can be read:

```
data = np.loadtxt('amplitude/amplitude00',delimiter=',')
data = np.reshape(data,(data.shape[0],180,990))
```

bounding.json

- json formatted text file that contains bounding boxes of detected objects for each RGB image frame
- Annotated using RetinaNet so **accuracy not guaranteed**
    - There can be multiple bounding boxes per image
- Data is in image coordinates
- Image size **1280,720** (needed if you want to normalize the bounding boxes [0,1])
- **15 fps**

```
{
    "category_id": 0, // 0=human, 1=car
    "id": 10,
    "area": 197772,
    "image_id": 10, // rgb frame
    "bbox": [847.7, 249.0, 1280.0, 706.5] // x,y,x2,y2
}
```

timestamps.log

Timestamps from the 60 GHz radar. (Quite useless.)

occupancy.txt

- Number of persons in each radar frame

```
1
2
1
1
1
```

parking.mp4

Video containing the RGB frames.

- 15 fps
- Image size **1280,720**

Meeting room dataset

This dataset is collected inside a meeting room where people are entering and leaving the room. The dataset is collected with 60 GHz radar.

Radar data starts roughly 6 seconds before RBG data. If you want to time-syncronize the data, skip the first 600 frames from the radar!

amplitude (folder)

```
-amplitude
 |-amplitude00
 |-amplitude01
 |-ampltiude02
```

- Text files that contain radar data from 60 GHz radar
- Each row is a single frame from the radar

```
130.12345,156.56712,120.12423,.. // 19800 elements each row
```

- Frame has **110** rows and **180** columns
- Framerate is **100 Hz**

- Data is in polar coordinate system
    - Range information in rows (min range 0.5 m, max range 6 m)
    - Angle information in columns (min angle -90 deg, max angle 90 deg)

In python the data can be read:

```
data = np.loadtxt('amplitude/amplitude00',delimiter=',')
data = np.reshape(data,(data.shape[0],180,110))
```

bounding.json

- json formatted text file that contains bounding boxes of detected objects for each RGB image frame
- Annotated using RetinaNet so **accuracy not guaranteed**
    - There can be multiple bounding boxes per image
- Data is in image coordinates
- Image size **1280,720** (needed if you want to normalize the bounding boxes [0,1])
- **15 fps**

```
{
    "category_id": 0, // 0=human
    "id": 10, // rgb frame
    "area": 197772,
    "image_id": 10,
    "bbox": [847.7, 249.0, 1280.0, 706.5] // x,y,x2,y2
}
```

timestamps.log

Timestamps from the 60 GHz radar. (Quite useless.)

office.mp4

Video containing the RGB frames.

- 15 fps

occupancy.txt

- Number of persons in each radar frame

```
1
2
1
1
1
```

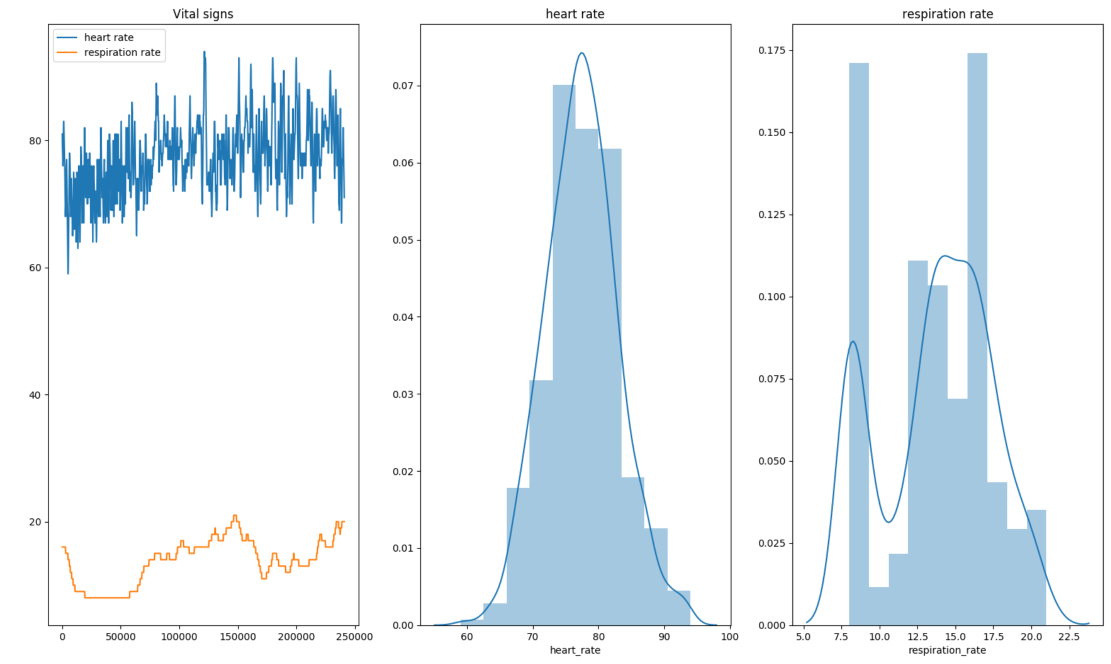
skeleton.json

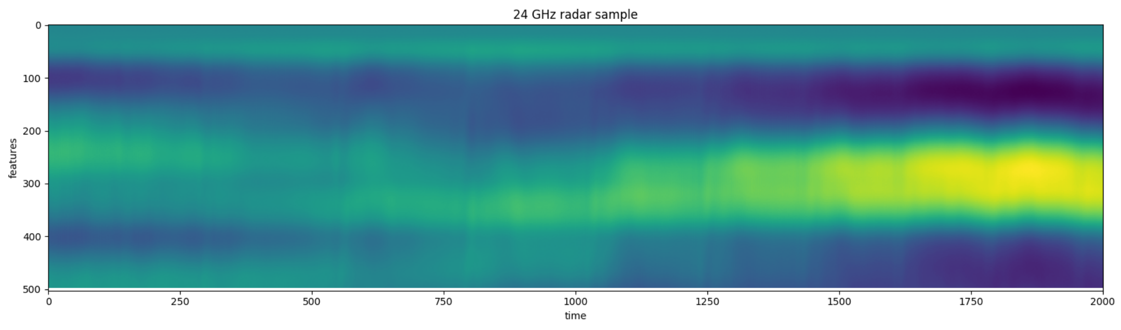
- persons' joint positions in (RGB) image coordinates
- Annotated using OpenPose so **accuracy not guaranteed**
- **15 fps**
- Image size **1280,720** (needed if you want to normalize the skeleton coordinates [0,1])
- Check the keypoint ordering here

```
{
    "image_id":62, // RGB frame
    "category_id":1, // human
    "keypoints":[
        0,0,0,
        0,0,0,
        0,0,0,
        0,0,0,
        0,0,0,
        1183.69,289.538,1, // x,y,visibility: 0=invisible, 1=visible
        0,0,0,
        1134.77,412.798,1,
        0,0,0,
        1164.15,477.371,1,
        0,0,0,
        0,0,0,
        0,0,0,
        0,0,0,
        0,0,0,
        0,0,0,
        0,0,0
    ],
    "score":0.0
    }
```

Workplace dataset

This dataset is collected at a workstation with 24 Ghz radar. The data contains a single person in front of the radar. The radar can be used to detect micromovements.





A single **csv** called **workplace.csv** containing the following data:

```
time // timestamps
r_0 // radar measurements
r_1
...
r_496
heart_rate // avg heart rate
respiration_rate // avg respiration rate
left_shoulder_x
```

```
left_shoulder_y
left_shoulder_z
left_elbow_x
left_elbow_y
left_elbow_z
left_wrist_x
left_wrist_y
left_wrist_z
neck_x
neck_y
neck_z
nose_x
nose_y
nose_z
right_shoulder_x
right_shoulder_y
right_shoulder_z
right_elbow_x
right_elbow_y
right_elbow_z
right_wrist_x
right_wrist_y
right_wrist_z
good_posture // 0: bad posture, 1: good posture
moving // 0: still, 1: working/moving
```

You can load the data in python:

```
import pandas as pd
import numpy as np

data = pd.read_csv('workplace.csv')

# Additionally you can transform the data into phase and amplitude data
with
# these functions.
def calculate_phase(raw_data, bin=2):
    """Calculate phase from raw radar data sample.
       Phase information tells about
       target micromovements"""
    s = np.array(raw_data)
    S = np.fft.fft(s)
    ph = np.angle(S)
    phase = ph[bin]
    return phase

def calculate_phases(raw_data, start=1, end=6):
    """Calculate phases from raw radar data sample
       and return multiple bins. Phase information tells about
       target micromovements"""
    s = np.array(raw_data)
    S = np.fft.fft(s)
    ph = np.angle(S)
    return ph[start:end]

def calculate_amplitude(raw_data, bin=2):
    """Calculate amplitude from raw radar data sample.
       Amplitude depends on the target size."""
    s = np.array(raw_data)
    S = np.fft.fft(s)
    amplitude = np.abs(S)
    return amplitude
```

Note that there are a lot of missing values, because the radar is capable of capturing with higher fps than the other sensors.

Live Access

Folder `live` contains scripts to access both radars live. You need to connect your computer via ethernet cable to our computer to get the data.

Instructions for Linux:

```
pip install -r requirements.txt
cd live

# 24 Ghz
python Streamer24.py

# 60 Ghz
python Streamer60.py
```

Simulator

The script **simulator.py** contains a class Simulator. You can initialise it using any of the two environments, 'garage' or 'meeting'. You can install the required packages using the following command.

```
# if you haven't already installed requirements
pip install -r requirements.txt
```

You can integrate your algorithm output easily with the animation. The workflow is like this:

1. You have a set of N samples from the radar dataset. For each sample, your algorithm is reponisble for producing some value, for a total of N outputs. Based on this value, you wish to change the settings of the luminaires. It can be brightness (0-1.0), temperature (0-1.0, warm to cool) or both.
2. The function responsible in the simulator class is called populate_data. It accepts a list of items. Each item corresponds to the N output samples. Each item itself is a dictionary, where it can contain multiple

objects corresponding to different luminaires.
3. The key of each object is the luminaire ID, and the value is another object which contains 'b' and 't' values. At each instance, you are assigning a set of luminaires different brightness and temperature values. If you do not wish to change any of the values, you can just append an empty dict object.
4. Once you have created this list (called frames in the sample), you only need to call the function populate_data with the list as its argument, and then run the simulator. It will open the animation in the browser. You can optionally set the speed of the animation as well.

The structure of the data is given below. It shows two luminaires which are both at maximum brightness and their color is cool. At the next instance, we wish to power down Luminaire 1, so we set its brightness value to 0.0.

```
frames = array of N size

frames[0] = {
    'Luminaire 1' : {
        'b' : 1.0,
        't' : 1.0
    },
    'Luminaire 2' : {
        'b' : 1.0,
        't' : 1.0
    }
}

frames[1] = {
    'Luminaire 1' : {
        'b' : 0.0,
        't' : 1.0
    },
    'Luminaire 2' : {
        'b' : 1.0,
        't' : 1.0
    }
}
```

Live Demos

You can control light level (brightness) and color temperature by sending control command to usb dongle (control unit) via serial interface. Control unit - is a gateway to same bluethooth mesh network as luminaire box

In **lightControl.py** you will find example and functions that implements serial communication with a luminaire. You will need to install pyserial:

```
pip install pyserial
```

Don't forget to specify `LUMINAIRE_ID` in script (`string 18`). On box you can find the luminaire ID and wall panel (WP) identifier.

How to use the function:

```
set_light_level_color_temperature(_serialDevice, _lightLevelValue = 50,
_colourTemperatureValue = 3900)
```

- `_serialDevice` - configured serial port where to send command
- `_lightLevelValue` - light level value in percent from 0 to 100 (lowest brightness to highest)
- `_colourTemperatureValue` - color temperature in Kelvin from 6500K (coolest temperature), 2700K (warmest)

FAQ

24 Ghz radar

- horizontal beamwidth: 80 deg
- elevation beamwidth: 25 deg
- range resolution: 60 cm
- angular resolution: not applicable as this is one-channel
- micromotion detection accuracy: 1-5 um

60 Ghz radar

- horizontal beamwidth: 160 deg
- elevation beamwidth: 25 deg
- range resolution: 5 cm
- angular resolution: 3.5 deg
- micromotion detection accuracy: < 700 nm

What causes the sidelobes in the radar data?

> The sidelobes in the radar image are mainly caused by two effects:
>
> - Leakage signal from transmitter to receiver
> - Numerical leakage in Fourier transform-based image formation

Why there is flickering in the radar response?

> Human form fluctuates in radar image because at millimeter wavelenghts (60 GHz frequency), humans are large targets consisting of many small point targets. The cumulative effect of these many point targets as human moves (even breaths) causes the fluctuation.

What is the difference between 24 GHz and 60 Ghz material penetration?

> Lower frequency signals typically penetrate materials easier. However, from this point of view 24 GHz and 60 GHz are quite close to each other and there are no large differences in material penetration. For example clothing materials, plastics, dry wood etc. are penetrated quite well.

Does the depth resolution decrease as a function of distance?

> Range resolution remains the same regardless of target distance. This is an effect of the frequency-modulated continuous wave radar approach. The signal-to-noise ratio of target does of course get worse when targets are far away as received signal power is smaller.