CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

# DSECOP280: Automated Object Detection

Karan Shah
DSECOP Fellow
www.casus.science

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
HELMHOLTZ CENTRE FOR ENVIRONMENTAL RESEARCH – UFZ
Uniwersytet Wrocławski
SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT UND KUNST
Freistaat SACHSEN

# Goals

# Goals

- Relevant course: Introductory Classical Mechanics Lab Courses (also relevant Advanced Labs)

- Physics goals:
  - Basic numerical methods
  - Experiment setup
  - Dealing with noise

- Machine learning goals:
  - Introduction to Computer Vision (CV) from scratch
  - Application of CV for analyzing experiments

# Structure

# Structure

- Prerequisites:
  - Basic classical mechanics (already satisfied by being in lab)
  - `numpy` and `matplotlib`

- Lesson 1: Representation and manipulation of images in Python
- Lesson 2: Object tracking from scratch
- Lesson 3: Ready-to-use code for lab videos

- Components:
  - In-built interactive demonstrations and exercises
  - Mini-projects
  - In-lab use

# Submodules

# Lesson 1

## Introduction to CV
## (and manipulation using Linear Algebra)

```
[[1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1.],
[1., 1., 1., 1., 0., 0., 0., 1., 0., 0., 0., 1., 1., 1., 1.],
[1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.],
[1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.],
[1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1.],
[1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.],
[1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1.],
[0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0.],
[1., 0., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1., 1., 0., 1.],
[1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1.],
[1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1.],
[1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1.],
[1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.],
[1., 1., 1., 1., 0., 0., 0., 1., 0., 0., 0., 1., 1., 1., 1.],
[1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1.]]
```

Array representation

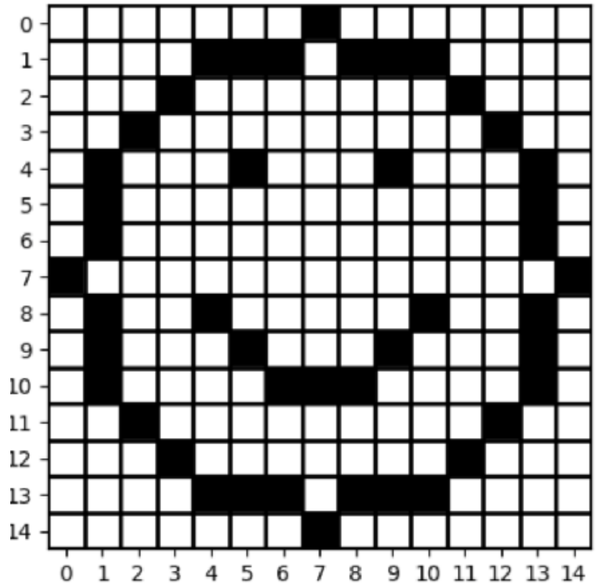
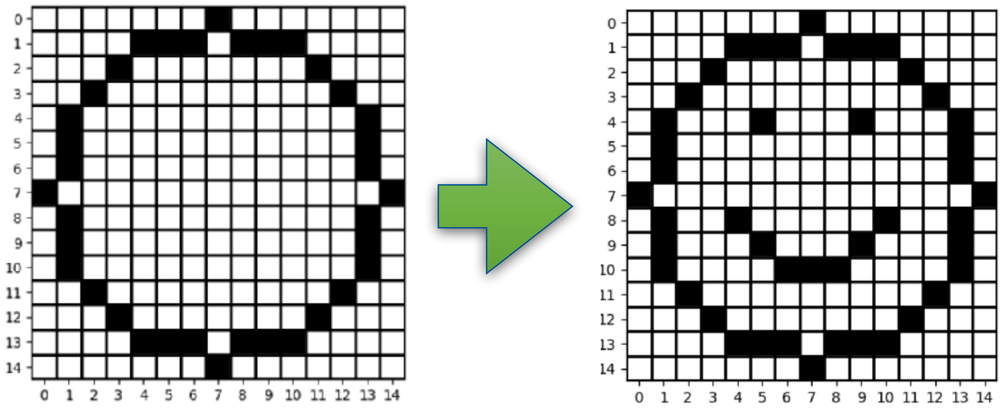



Image representation

# Lesson 1

## Introduction to CV - Pixel art warmup

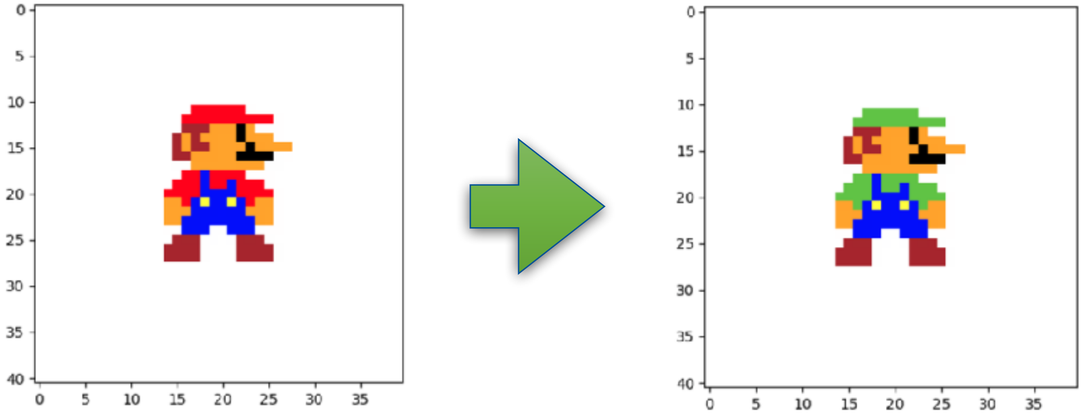


Manipulate pixels

```
# Define the positions of the eyes and the smile. Student code:
eye1 = (4, 5)
eye2 = (4, 9)


# Set these positions to 0 (black). Student code:
image[eye1] = 0
image[eye2] = 0

# Fill in the smile. Student code:
image[(10,7)] = 0
for i in range(1,4):
    image[(10-i+1, 7-i)] = 0
    image[(10-i+1, 7+i)] = 0
```

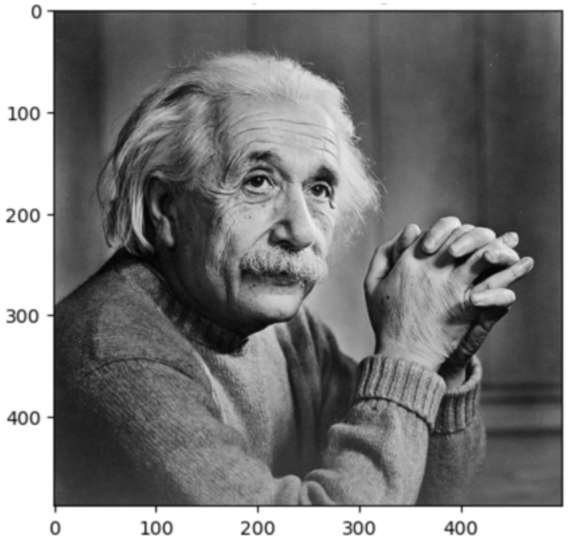


Manipulate pixel colors

```
# Fill in the code snippets, hints in comments:
# Pick a pixel.
hat_pixel = (11,18)
# Get pixel color.
mario_red = mario_image[hat_pixel][:]
# Set Luigi pixel color.
luigi_green = [96, 196, 69]
luigi_image = mario_image
# Select all red pixels, all three channels.
r_mask = mario_image[:,:,0] == mario_red[0]
g_mask = mario_image[:,:,1] == mario_red[1]
b_mask = mario_image[:,:,2] == mario_red[2]
# Combine the masks.
combined_mask = r_mask & g_mask & b_mask
# Change red pixels to green.
luigi_image[combined_mask] = luigi_green
```

# Lesson 1

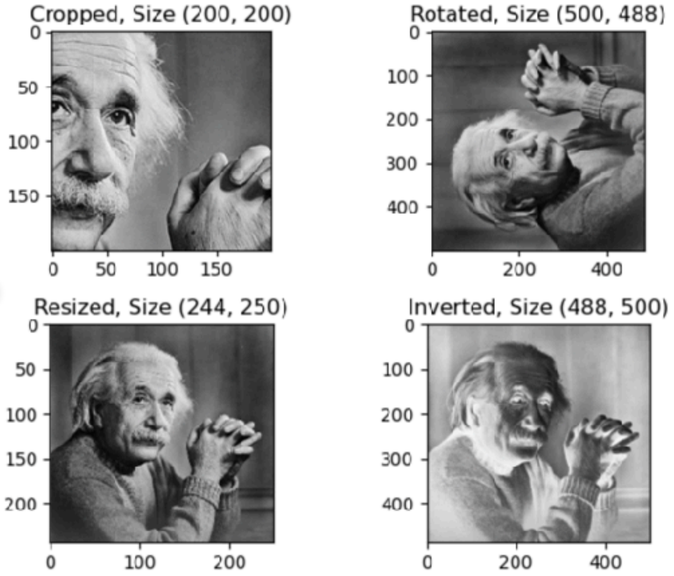
## Introduction to CV - Basic manipulations



Input image

```
# Fill in the following lines:
# Cropping
cropped_image = bw_image[100:300, 200:400]
# Rotation (90 degrees)
rotated_image = np.rot90(bw_image)
# Resize
resized_image = bw_image[::2, ::2]
# Invert
inverted_image = np.abs(255 - bw_image)
```

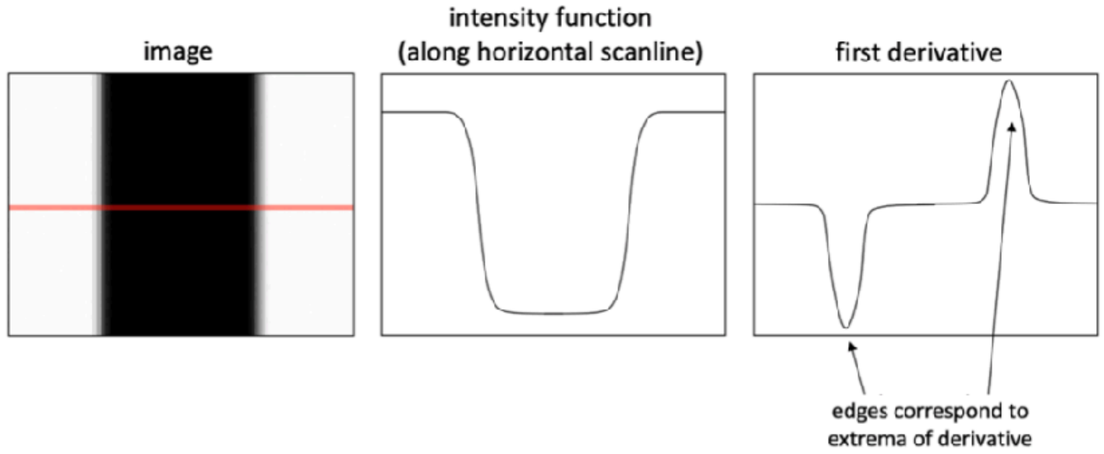
Student code




Basic manipulations

# Lesson 1

## Introduction to CV - Kernels and Convolutions - Example: Edge Detection




Build intuition

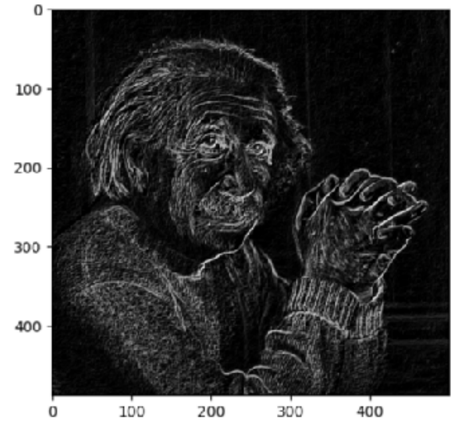
$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Operator definition

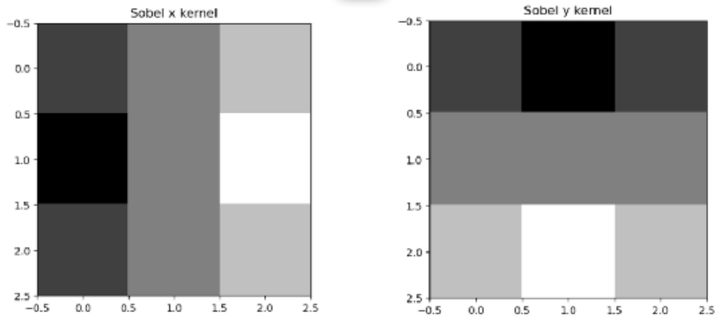
```python
def sobel_operator(image):
# Fill in the operators. Student code:
    sobel_x = np.array([[-1, 0, 1],
                        [-2, 0, 2],
                        [-1, 0, 1]])
    sobel_y = np.array([[-1, -2, -1],
                        [0, 0, 0],
                        [1, 2, 1]])
# End student code.
    grad_x = cv2.filter2D(image, -1, sobel_x)
    grad_y = cv2.filter2D(image, -1, sobel_y)

    grad_x = rescale(grad_x)
    grad_y = rescale(grad_y)
    return grad_x, grad_y
```

Student code (boilerplate provided)



Results



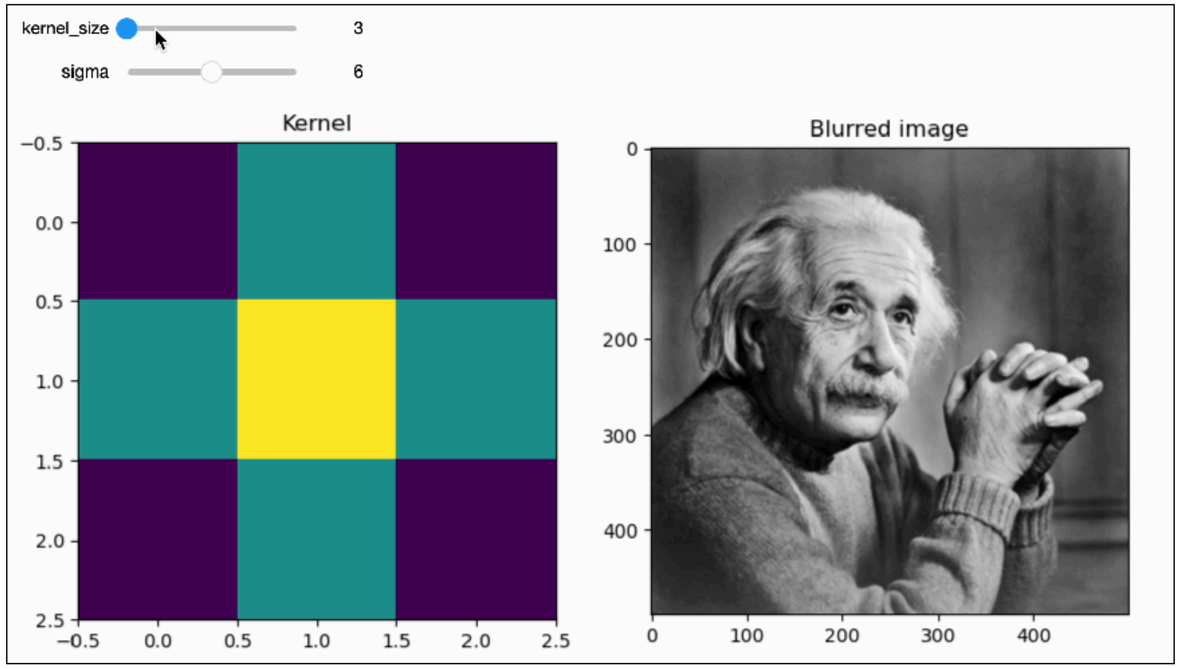

Visualize kernels

# Introduction to CV - Kernels and Convolutions - Example: Gaussian blur

Implement 2D Gaussian blur:

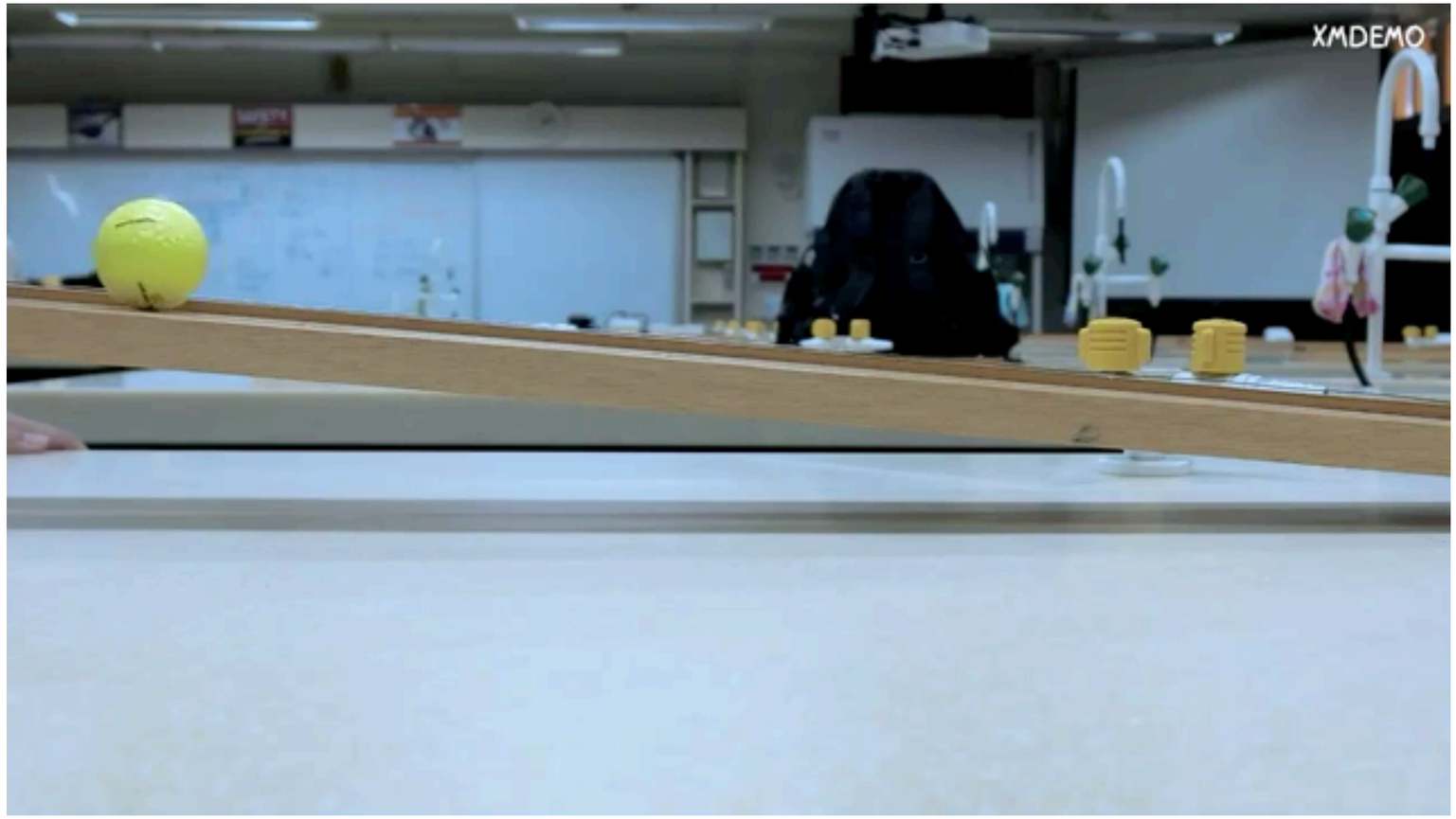
$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

```python
def gaussian_blur(kernel_size=20, sigma=5, ret = False):
    ax = np.arange(-kernel_size // 2 + 1., kernel_size // 2 + 1.)
    xx, yy = np.meshgrid(ax, ax)
    # Student code:
    kernel = np.exp(-(xx**2 + yy**2) / (2. * sigma**2))
    kernel = kernel / np.sum(kernel)
    # End student code
    blurred_image = cv2.filter2D(image, -1, kernel)
```



# Lesson 2

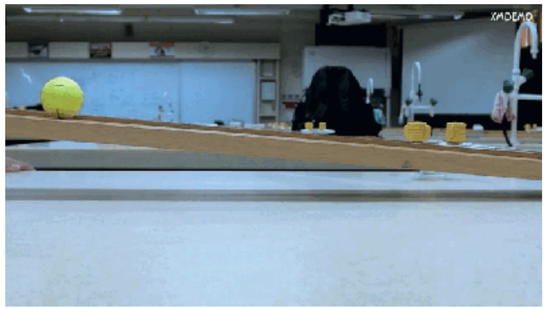
## Object tracking




Example experiment video

# Lesson 2

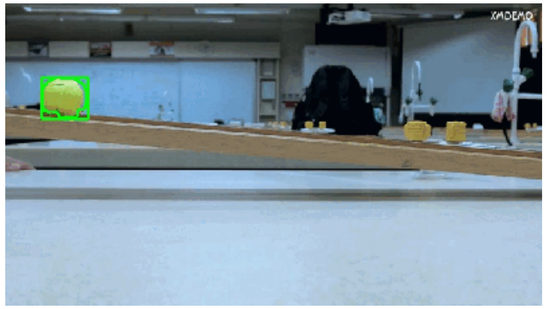
## Object tracking - Simple algorithm with OpenCV operations



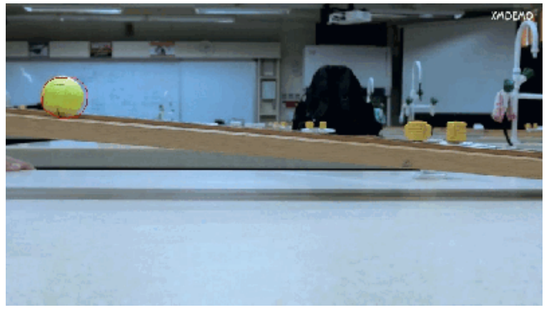

Input

Difference of frames

Grayscale and blur




Output




Convex hulls

Connected components

Binary threshold

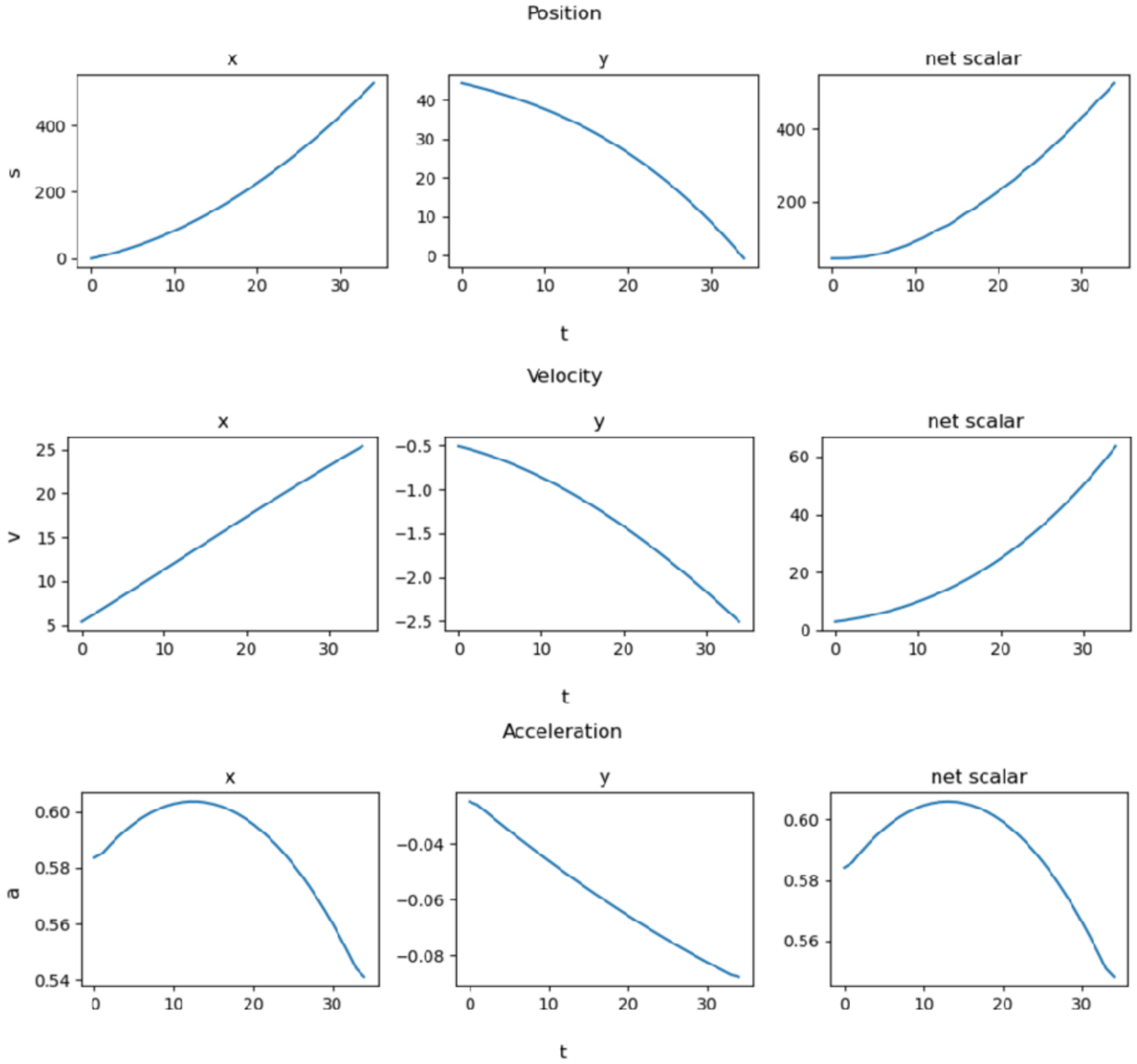
# Lesson 2

## Trajectory estimation

- Filtering noise
- Gradient calculation

```
i=0 | Coords: x=43, y=88, w=55, h=49 | Detections: 1
i=1 | Coords: x=47, y=88, w=56, h=51 | Detections: 1
i=2 | Coords: x=52, y=88, w=55, h=51 | Detections: 1
i=3 | Coords: x=57, y=88, w=56, h=52 | Detections: 1
i=4 | Coords: x=64, y=89, w=56, h=51 | Detections: 1
i=5 | Coords: x=71, y=89, w=56, h=53 | Detections: 1
i=6 | Coords: x=77, y=90, w=58, h=52 | Detections: 1
i=7 | Coords: x=85, y=90, w=59, h=54 | Detections: 1
i=8 | Coords: x=93, y=91, w=60, h=53 | Detections: 1
```

Sample output




Trajectory extracted from the example.

# Lesson 3

## Automatic Tracking

Input video

```
python 03_auto_tracking.py spring
spring
Found directory spring. Moving there.
[INFO] Read video file in spring
Detection mode:
1. Manual Region of Interest Selection
2.  Automatic using YOLO
(Default 1)1
[INFO] Detection mode 1
[INFO] Manual detection
Select a ROI and then press SPACE or ENTER button!
```

Run provided code

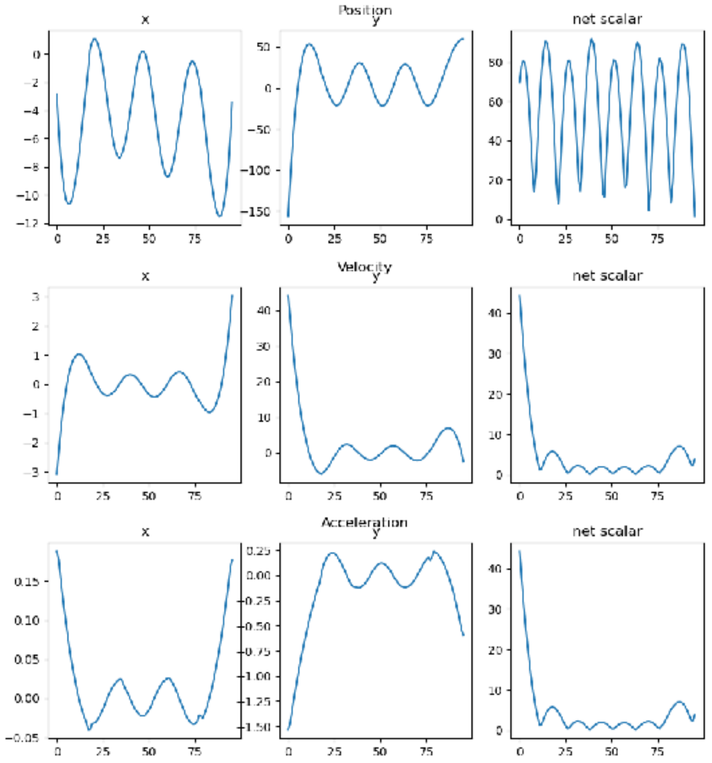
2 options:

1) OpenCV Tracking

2) Automatic NN tracking

(recognizes circular objects)


Video annotation




Trajectory extraction

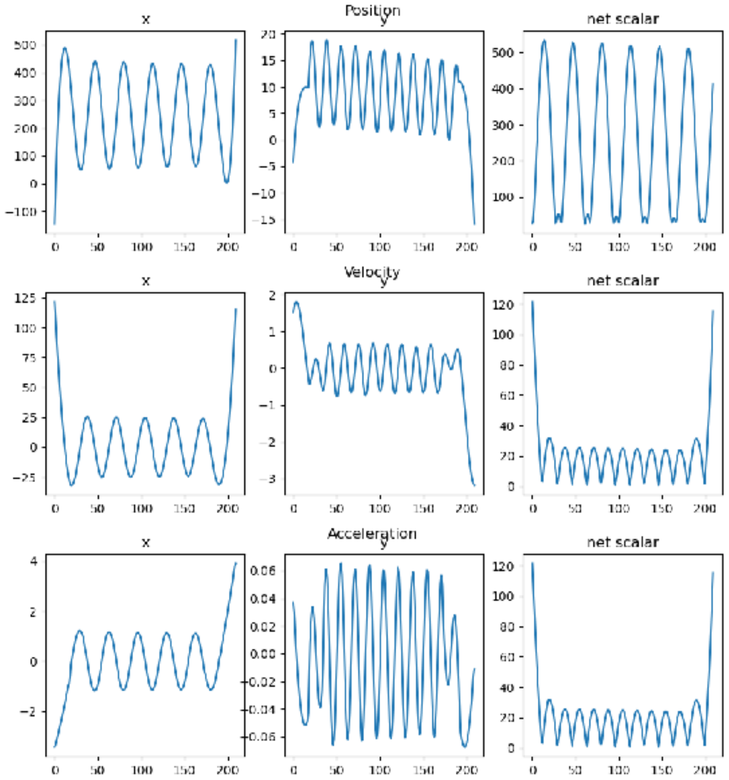
CSV + Recording + Plots

# Lesson 3

## Not so automatic

Example of bad annotation

Good annotation

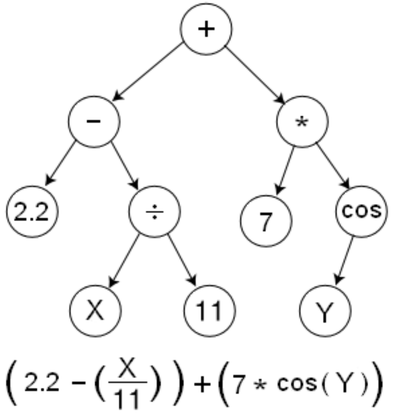



Trajectory for good annotation case

# Mini-projects

1. Hybrid Images (After Notebook 1)

2. Combine with Symbolic Regression and Numerical Methods modules (After Notebook 2)




$$\left(2.2 - \left(\frac{X}{11}\right)\right) + \left(7 * \cos(Y)\right)$$

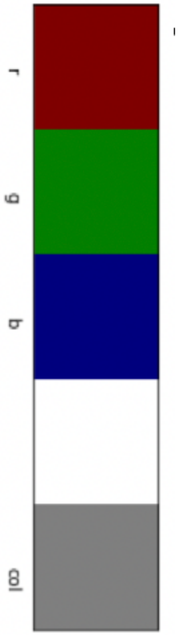
[Symbolic Regression](#)
- Joseph Dominicus Lap

$$\frac{1}{(\Delta X)^2}\begin{vmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -2 \end{vmatrix} \begin{vmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{vmatrix} = \begin{vmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{vmatrix}$$

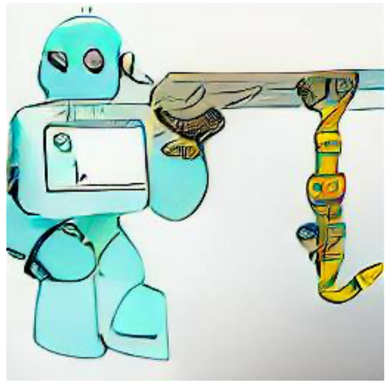
[Solving Differential Equations in Classical Mechanics with Neural Networks](#)
- Julie Butler

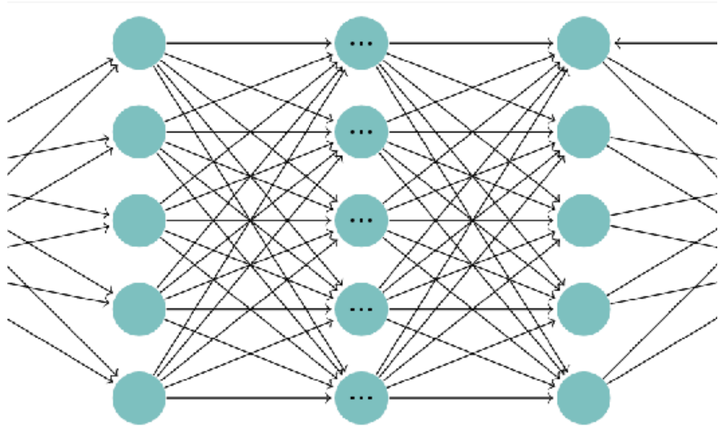
# Supplementary:

## Gentle Introduction to Neural Networks (with plumbing and colors)

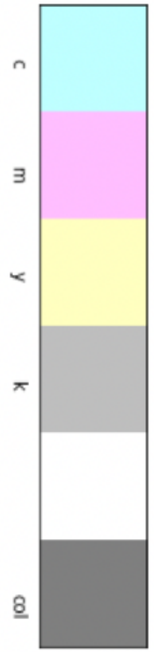



RGB





Neural Network
(Trainable Parameters)




CMYK

In module: Learning the Schrödinger Equation

# Sample Lesson Plan

- Take home - Lesson 1: Image Manipulation (2 hours)
- In class - General discussion of requisite Linear Algebra
  (1 hour)
- Take home - Lesson 2 and Lesson 3: With included examples (1 hour)
- Mini-Project
  (2 - 4 hours)
- In lab - Lesson 2 and Lesson 3 - With videos of own experiments

# Alternatives

- LoggerPro - Frame by frame annotation
- Physlets Tracker - Java based, Open source
- Vernier sensors - Need sensor hardware

# Request for videos

- If you use this module in your lab course, can you also share some sample videos that your students take?
- Can be used to train a custom neural network specialised for lab experiments.

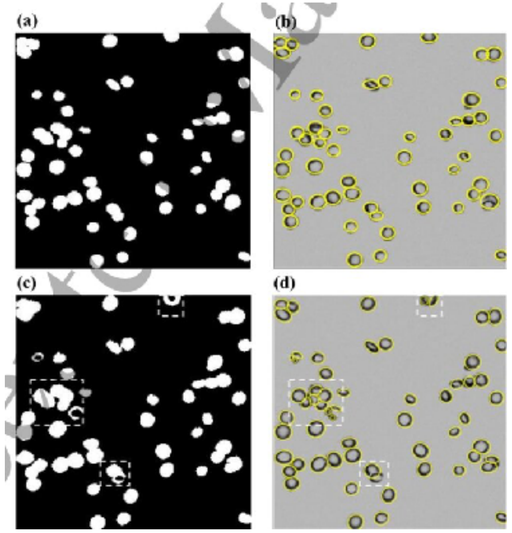
- Question: What tools are already being used in lab?

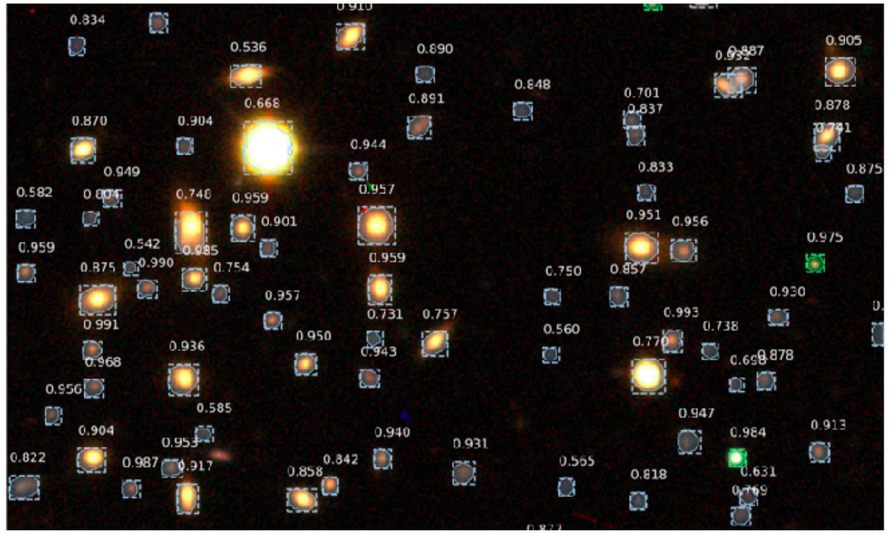
Please reach out on Github, Slack or at k.shah@hzdr.de.

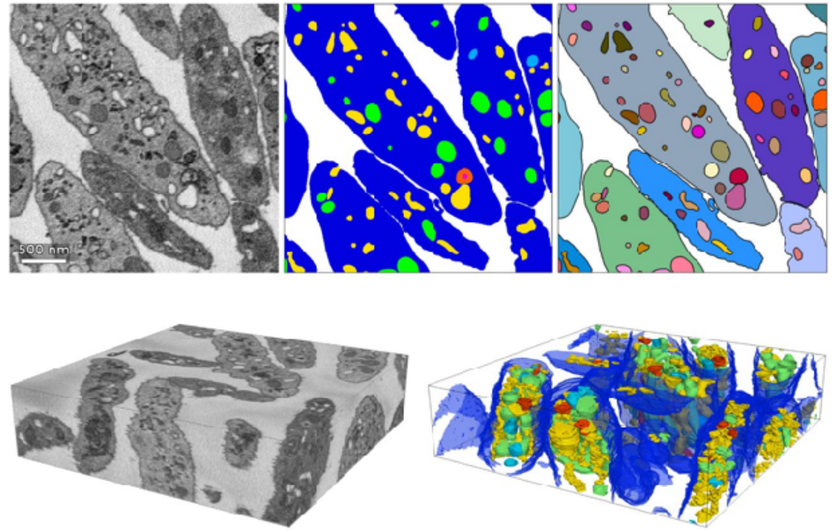
# Conclusion

# Conclusion

- Module can be used for a lab course
- Should be actively modified by students during labs
- Can be combined with DSECOP Modules: Symbolic Regression, Numerical Methods
- Computer vision can be useful in research too (eg image segmentation)



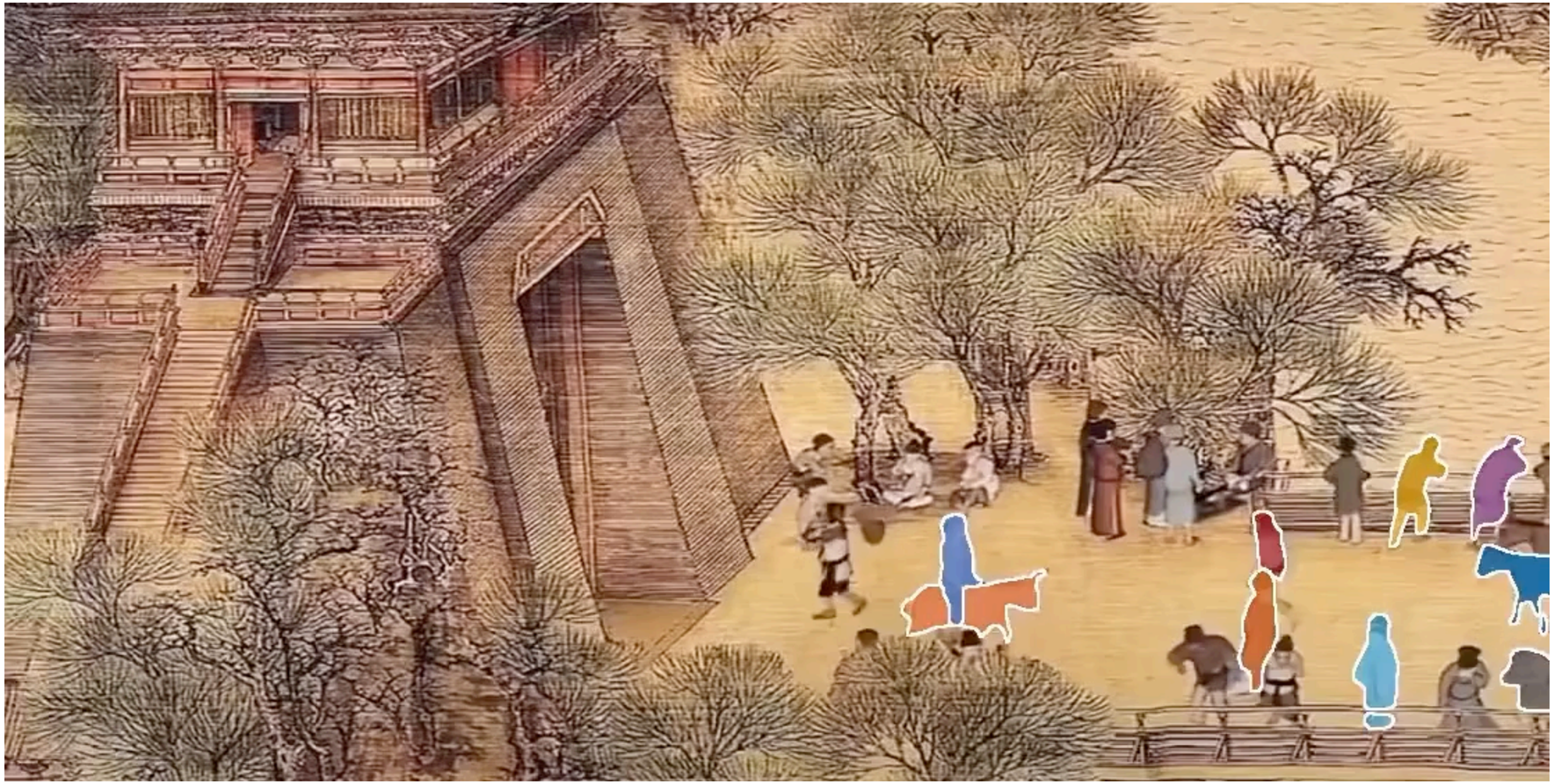






Please reach out on Github, Slack or at k.shah@hzdr.de.

# Future work

- Fine-tuning Yolo network used here to detect common lab objects
- Advanced models: “Track Anything”



# Thank you!
# Questions Comments Concerns?

Karan Shah
k.shah@hzdr.de

Feedback form:
*https://bit.ly/DSECOP-feedback*

Feedback

GitHub