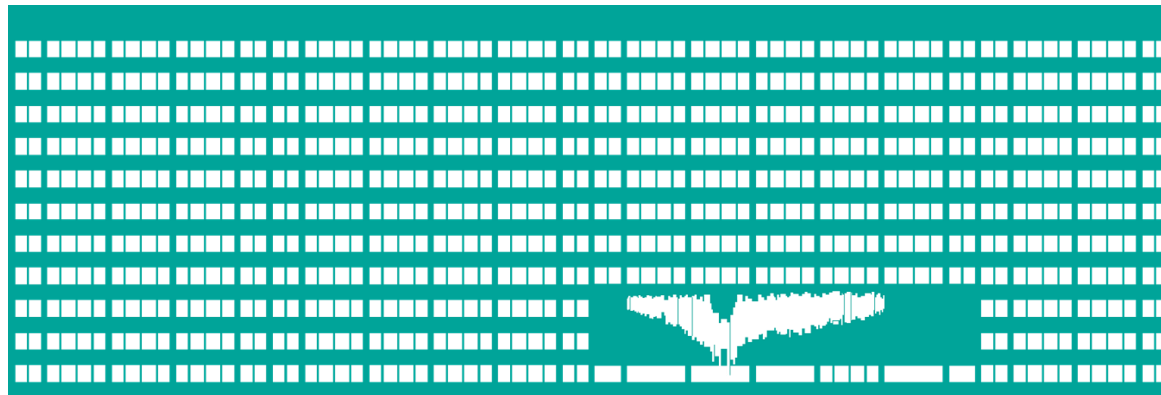




VŠB TECHNICKÁ
UNIVERZITA
OSTRAVA

VSB TECHNICAL
UNIVERSITY
OF OSTRAVA



www.vsb.cz

Základy analýzy obrazu (ZAO)

Radovan Fusek

The most common tasks in image processing:

- Enhancement/Filtering

Filtering



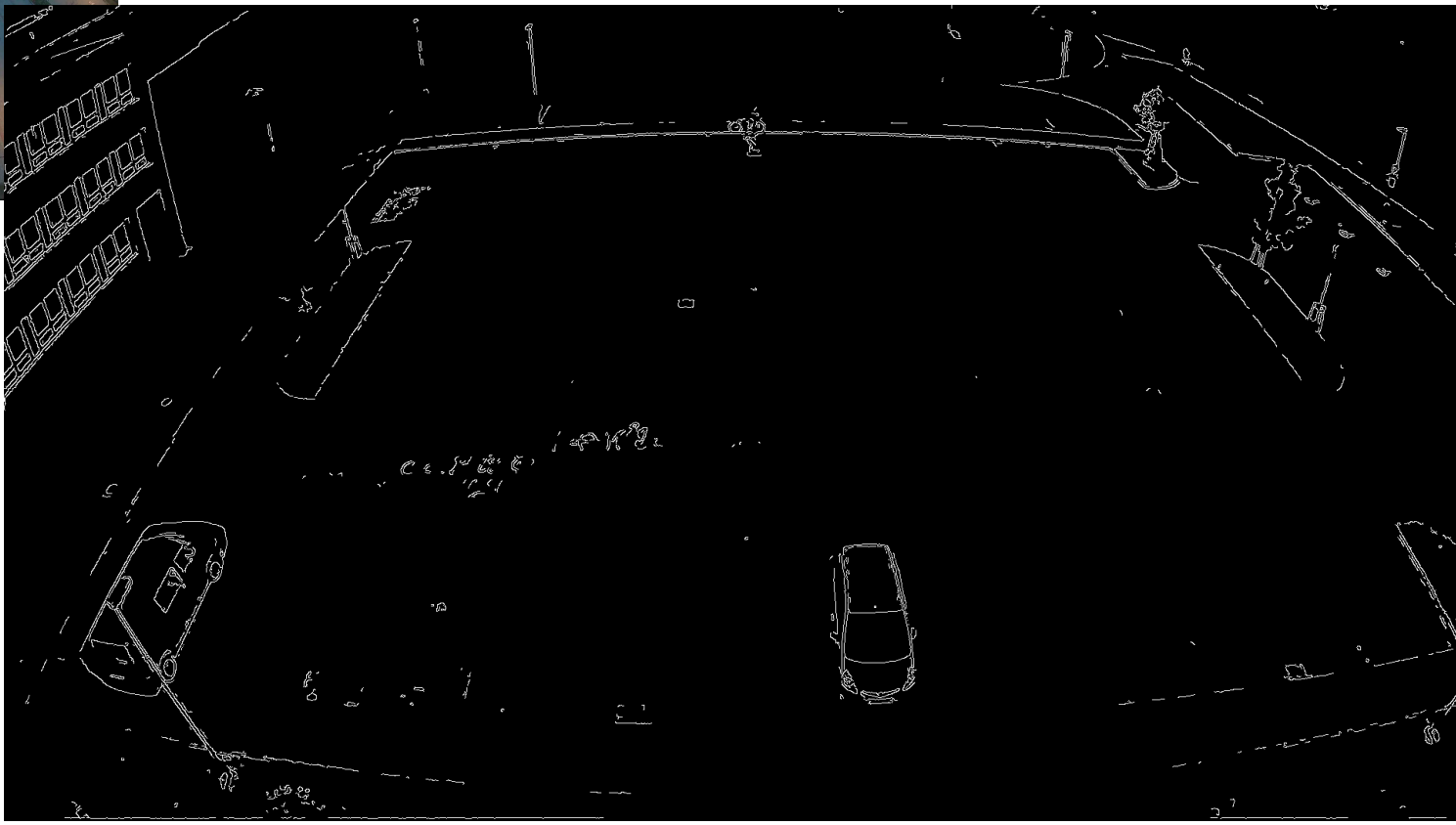
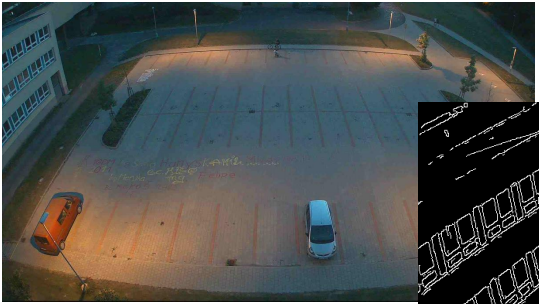
Filtering



The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection

Edge Detection



Edge Detection

Video Example

The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection

Object Detection

What is goal of object detection methods?



Object Detection

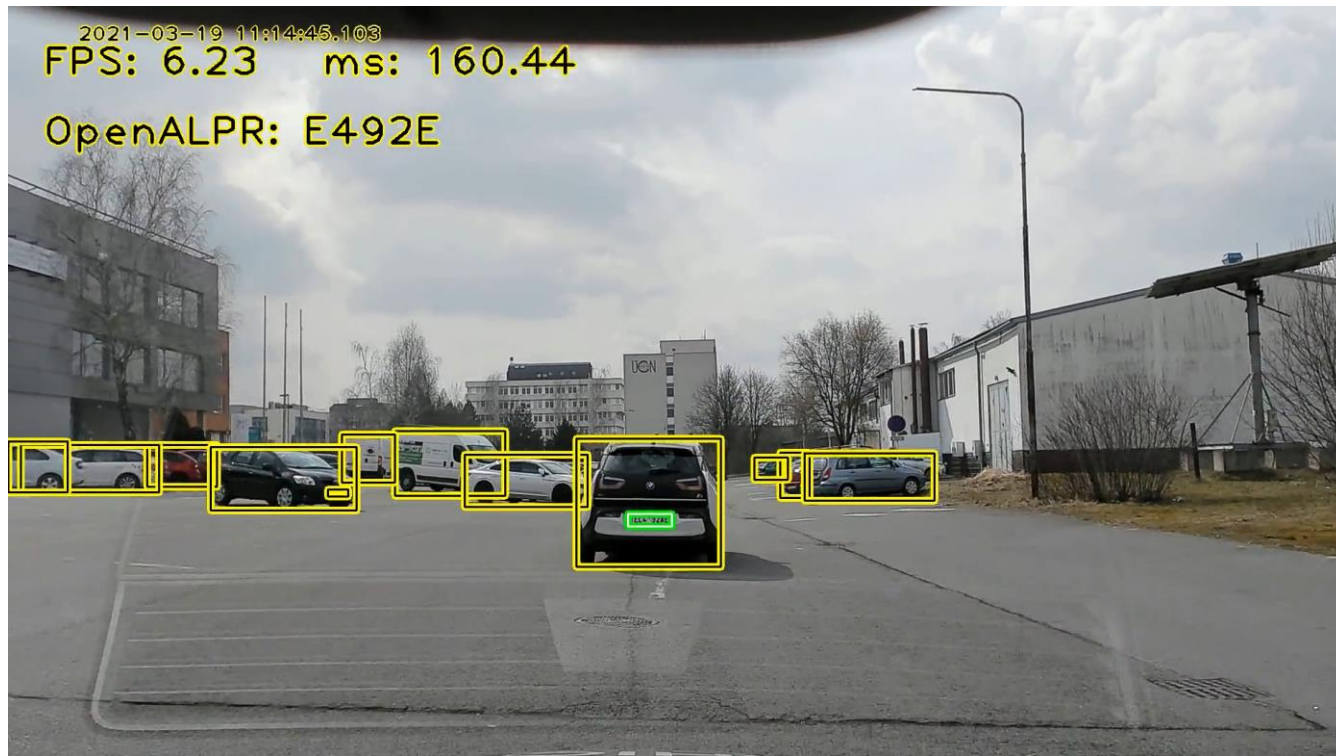
What is goal of object detection methods?

- It is clear that the images contain many objects of interest. The goal of the object detection systems is to find the location of these objects in the images (e.g. cars, faces, pedestrians).
- For example, the vehicle detection systems are crucial for traffic analysis or intelligent scheduling, the people detection systems can be useful for automotive safety, and the face detection systems are a key part of face recognition systems.

Object Detection

What is goal of object detection methods?

- position of the objects (coordinates) + scale of the objects



Object Detection

Video Example

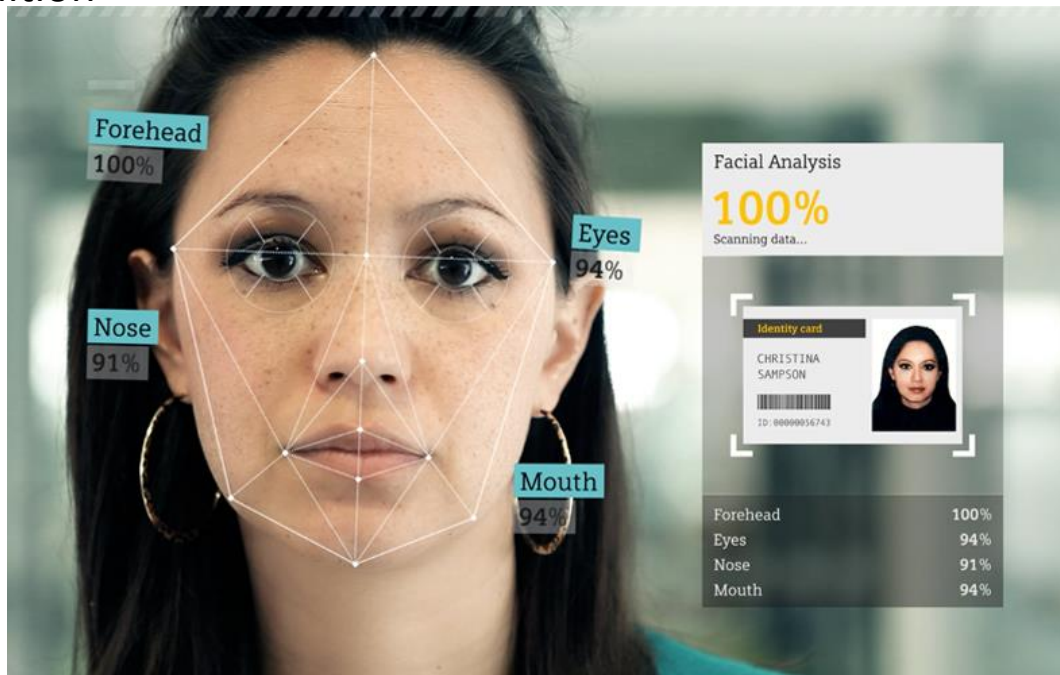
The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection
- Object Recognition

Object Recognition

What is goal of object recognition methods?

- identifying objects (categories)
- e.g. face recognition



<https://rexfaces.com/articles/how-accurate-is-facial-recognition>

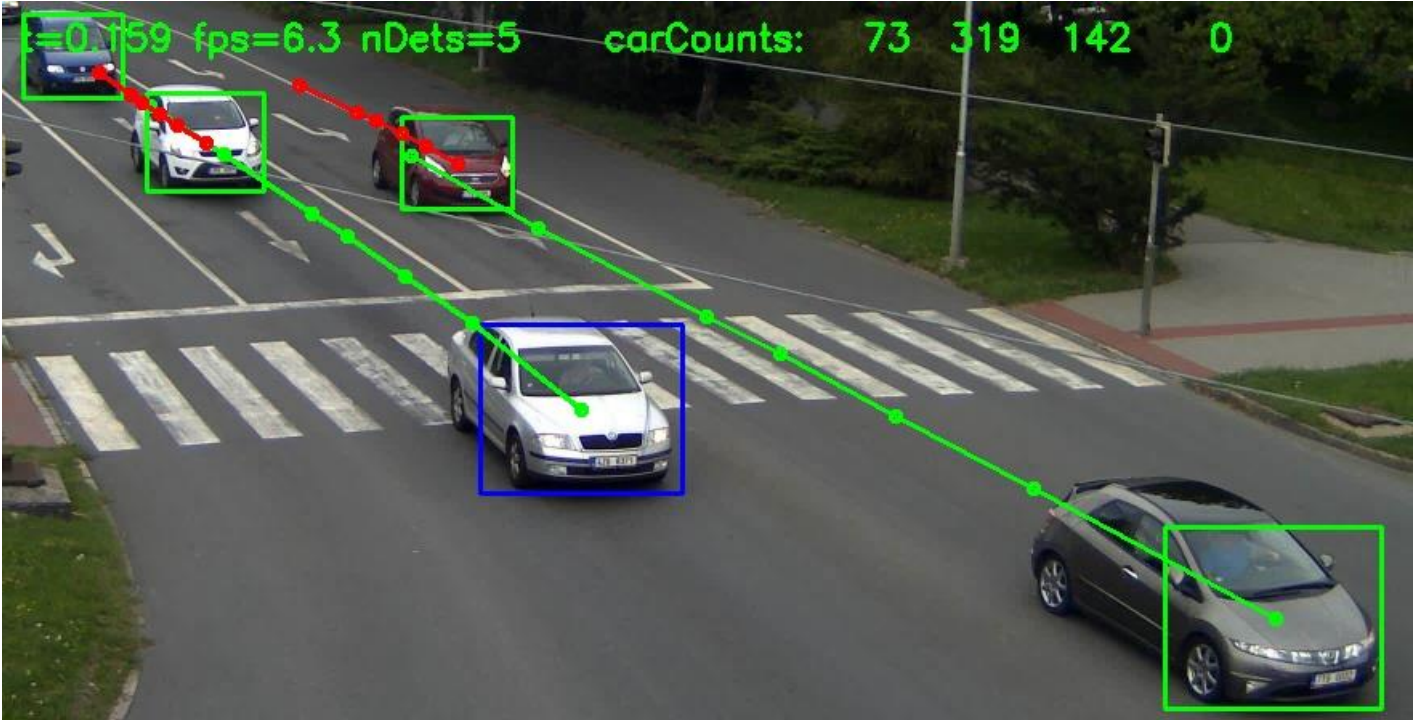
The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection
- Object Recognition
- Object Tracking

Object Tracking

Object Tracking

- e.g. estimation of trajectory/unique ID to each tracked object

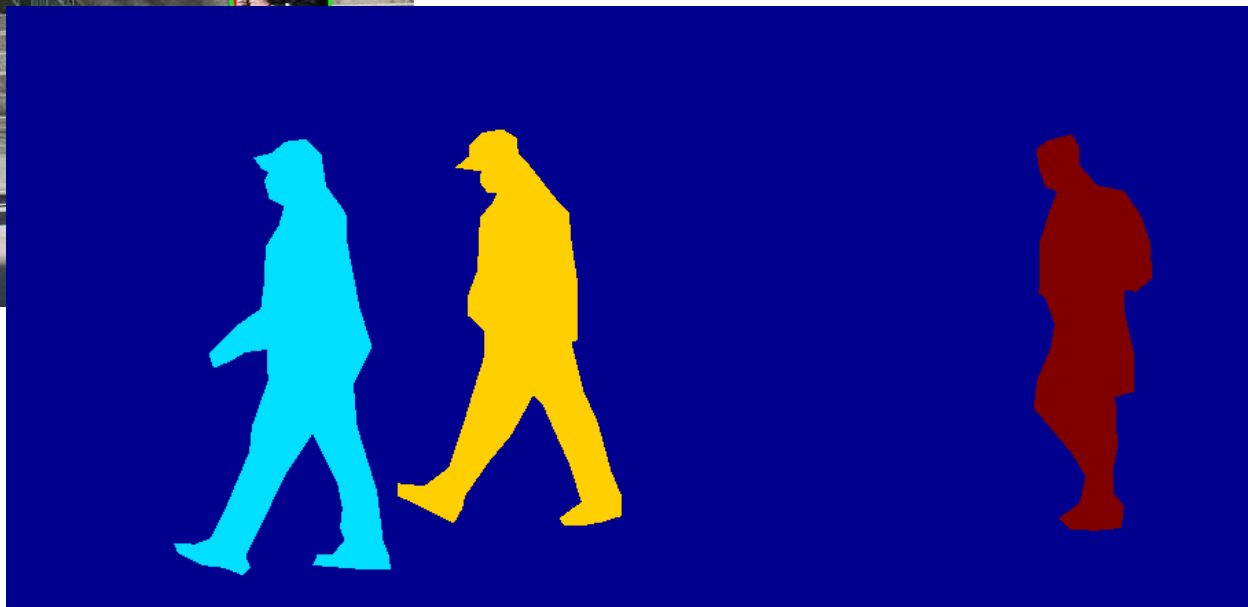
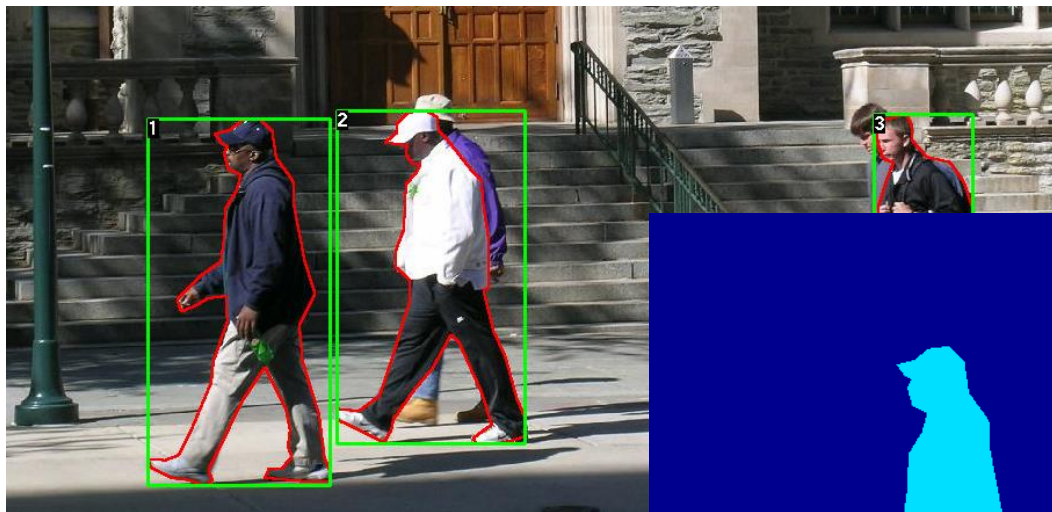


The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection
- Object Recognition
- Object Tracking
- Segmentation

Object Segmentation

Image is partitioned into multiple regions



Object Segmentation

Image is partitioned into multiple regions

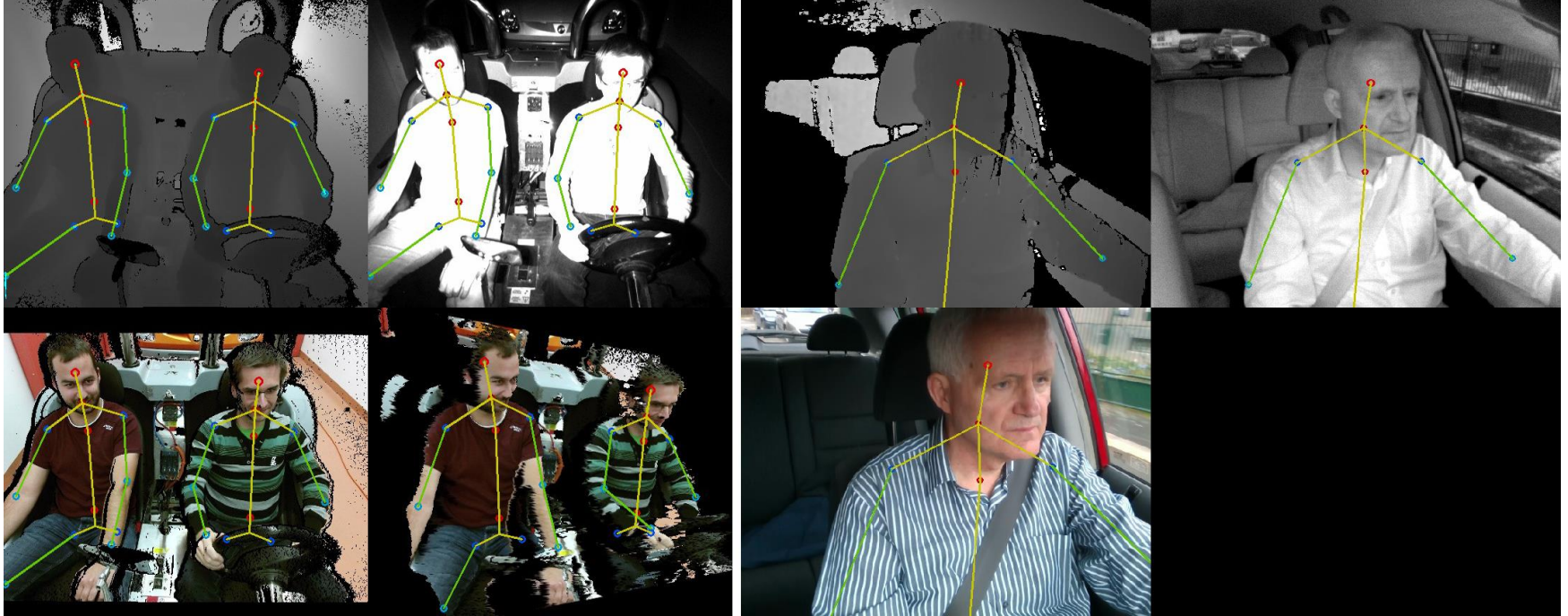


Video Example

The most common tasks in image processing:

- Enhancement/Filtering
- Edge Detection
- Object Detection
- Object Recognition
- Object Tracking
- Segmentation
- Human Pose Detection
- ... many others

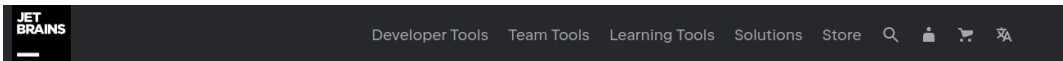
Pose Detection



Pose Detection

Video Example

Opencv + Python



PyCharm What's New Features Learn Buy [Download](#)

PC **PyCharm**

The Python IDE
for Professional Developers

[DOWNLOAD](#)

Full-fledged Professional or Free Community



[Library](#) [Forum](#) [Courses](#) [Services](#) [Store](#) [Partnership](#) [Resources](#)

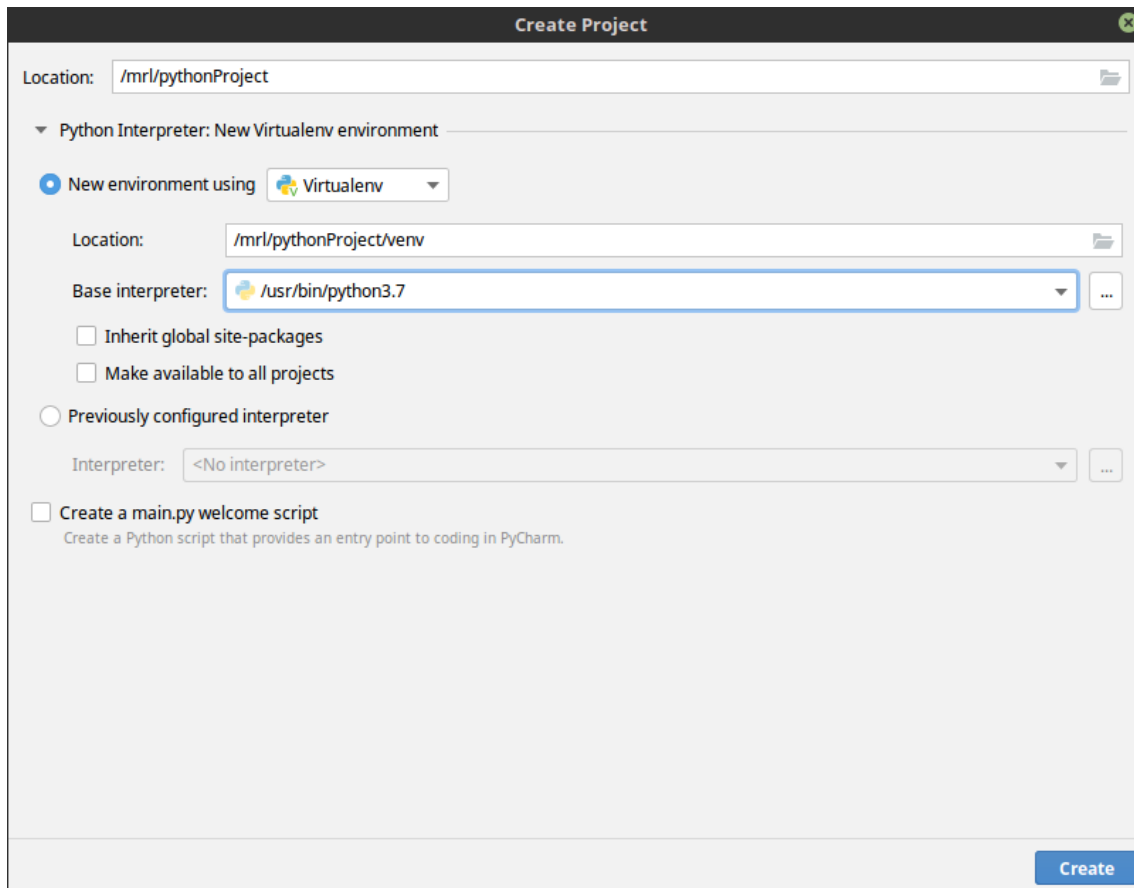
Introducing OAK-D-LITE

The latest OpenCV AI Kit, now on Indiegogo!

[Visit Site](#)

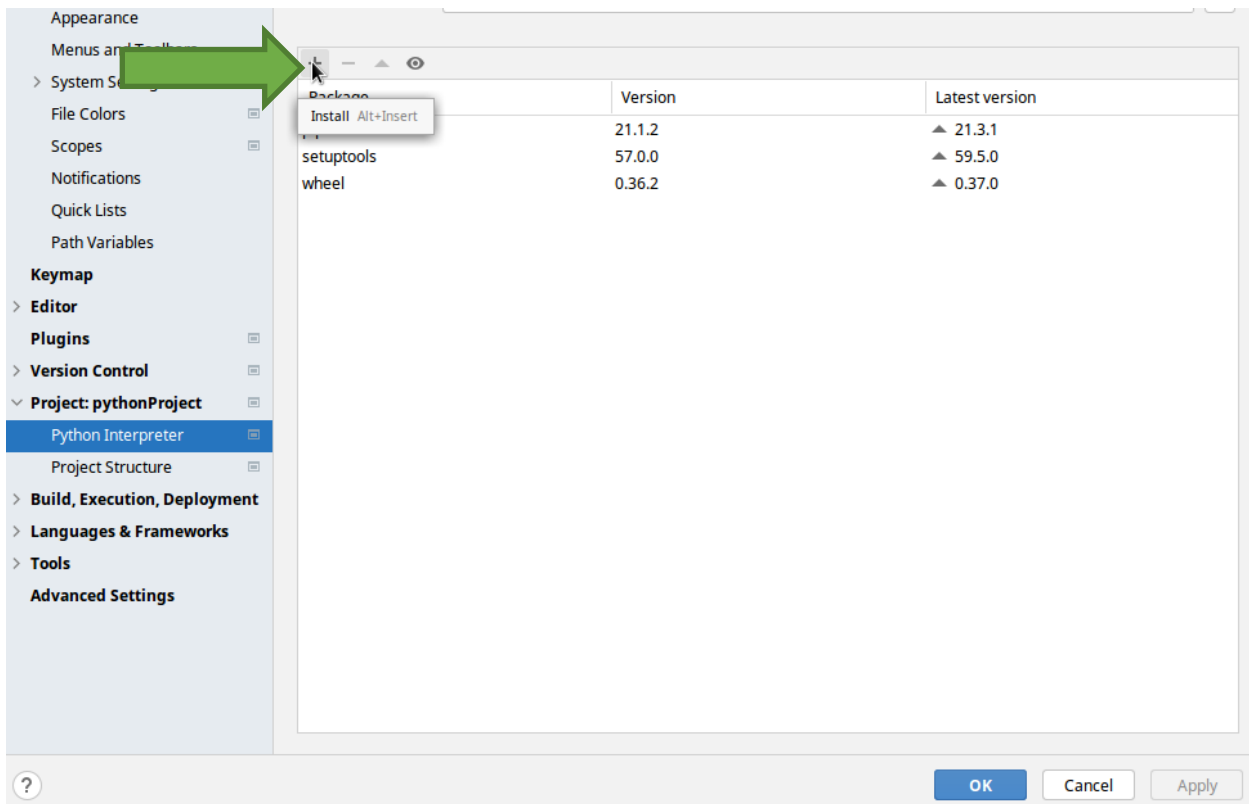
python-3.7.9-amd64
(OK with Dlib)

Opencv + Python



Opencv + Python

Ctrl+Alt+S



Opencv + Python

Available Packages

Q opencv-contrib-

opencv-contrib-python
opencv-contrib-python-headless

Description

Wrapper package for OpenCV python bindings.

Version
4.5.4.60
<https://github.com/skvarok/opencv-python>

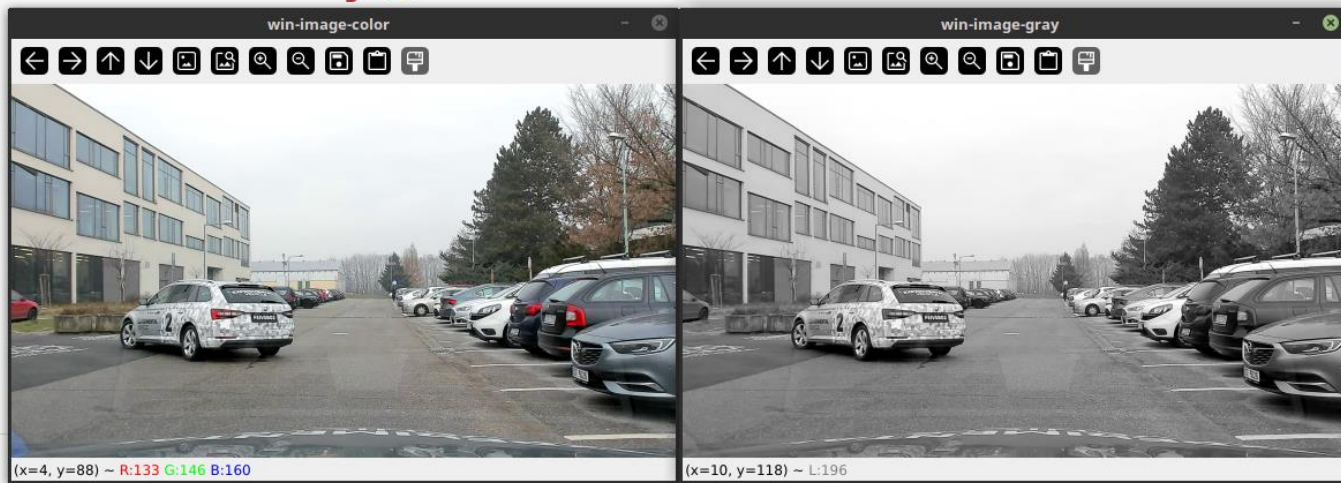
Specify version 4.5.4.60

Options

Install Package Manage Repositories

Opencv + Python

```
1 import cv2
2
3 image_gray = cv2.imread("img.png", 0)
4 image_color = cv2.imread("img.png", 1)
5 cv2.imshow("win-image-gray", image_gray)
6 cv2.imshow("win-image-color", image_color)
7 cv2.waitKey()
```



Opencv + Python

```

1 import cv2
2
3 image_gray = cv2.imread("img.png", 0)
4 image_color = cv2.imread("img.png", 1)
5 cv2.imshow("win-image-gray", image_gray)
6 cv2.imshow("win-image-color", image_color)
7 cv2.waitKey()
    
```



Opencv + Python

```
pythonProject - main.py
File Edit View Navigate Code Refactor Run Tools VCS Window Help

pythonProject | main.py
Project | pythonProject /media/mr10/4TB_e...
| venv
| main.py
| test_image.png
| External Libraries
| Scratches and Consoles

1 import cv2
2
3 def load_show_image(name):
4     print(f'image: {name}')
5     in_mat = cv2.imread(name, 1)
6     h, w, c = in_mat.shape
7     print(f'size: {h}, {w}, {c}')
8
9
10 if __name__ == '__main__':
11     img_name = "test_image.png"
12     load_show_image(img_name)
13
14
15

Run: | main |
| /media/mr10/4TB_ext4/vyuka/2021_let0/zao/pythonProject/venv/bin/python /media/mr10/4TB_ext4/vyuka/2021_let0/zao/pythonProject/main.py
| image: test_image.png
| size: 1080, 1920, 3
| Process finished with e...
```


Opencv + Python

The screenshot shows an IDE window titled "pythonProject - test_image.png". The interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help), a toolbar, and a sidebar with a Project Explorer showing a file structure: "pythonProject" containing "venv", "main.py", and "test_image.png", along with "External Libraries" and "Scratches and Consoles". The main workspace displays a photograph of a white car with a camouflage pattern and the license plate "FEIVS802". The car is parked in a lot with other vehicles and buildings in the background. The image metadata in the top right corner reads "1,920x1,080 PNG (24-bit color) 1.87 MB". At the bottom, a Run window shows the command: `/media/mr10/4TB_ext4/vyuka/2021_let0/zao/pythonProject/venv/bin/python /media/mr10/4TB_ext4/vyuka/2021_let0/zao/pythonProject/main.py` with the output: `image: test_image.png` and `size: 1080, 1920, 3`. The terminal also displays "Process finished with exit code 0". The bottom status bar includes "Version Control", "Run", "TODO", "Problems", "Terminal", "Python Packages", "Python Console", and "Event Log".

Opencv + Python

```

1 import cv2
2
3
4 def load_show_image(name):
5     print(f'image: {name}')
6     in_mat = cv2.imread(name, 1)
7     h, w, c = in_mat.shape
8     print(f'size: {h}, {w}, {c}')
9
10
11     win_name = "image"
12     cv2.namedWindow(win_name, 0)
13     cv2.resizeWindow(win_name, int(w/3), int(h/3))
14     cv2.imshow(win_name, in_mat)
15     cv2.waitKey(0)
16
17 if __name__ == '__main__':
18     img_name = "test_image.png"
19     load_show_image(img_name)
20
21

```

Run: main

```

/media/mr10/4TB_ext4/vyuka/2021_Leto/zao/pythonProject/venv/bin/python /media/mr10/4TB_ext4/vyuka/2021_Leto/zao/pythonProject/main.py
image: test_image.png
size: 1088, 1920, 3

```

image

(x=1206, y=1059) ~ R:140 G:149 B:151

Opencv + Python

The image shows a screenshot of an IDE (likely PyCharm) with a Python project named 'pythonProject'. The main file, 'main.py', contains the following code:

```

20 def load_show_video(name):
21     video_cap = cv2.VideoCapture(name)
22     win_video = "video"
23     cv2.namedWindow(win_video, 0)
24     while True:
25         ret, frame = video_cap.read()
26         if ret is True:
27             cv2.imshow(win_video, frame)
28             if cv2.waitKey(1) == ord('q'):
29                 break
30
31
32 if __name__ == '__main__':
33     img_name = "test_image.png"
34     vid_name = "test_video.mp4"
35     #load_show_image(img_name)
36     load_show_video(vid_name)
37
38
39
40 if __name__ == '__main__':

```

The code is highlighted in a green box. A yellow highlight is under the function call on line 36. To the right of the code editor, a window titled 'video' is open, displaying a video frame of a street scene with cars and buildings. The window has a status bar at the bottom showing coordinates: '(x=791, y=2) ~ R:170 G:209 B:236'. At the bottom of the IDE, a run console shows an error: 'QApplication: invalid style override 'gtk' passed, ignoring it. Available styles: Windows, Fusion'.

Opencv + Python

```

1 import cv2
2
3
4 def convert_video_to_gray(in_video, out_video):
5     video_cap = cv2.VideoCapture(in_video)
6     # if not video_cap.isOpened():
7     if video_cap.isOpened() == False:
8         print("video_cap: False")
9
10    fps = video_cap.get(cv2.CAP_PROP_FPS)
11    frame_count = video_cap.get(cv2.CAP_PROP_FRAME_COUNT)
12    frame_width = video_cap.get(cv2.CAP_PROP_FRAME_WIDTH)
13    frame_height = video_cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
14
15    print(f'fps: {fps}, frame_count: {frame_count}')
16    print(f'frame_width: {frame_width}, frame_height: {frame_height}')
17
18
19 def main():
20     in_video = "test_video.mp4"
21     out_video = "test_video_gray.mp4"
22     convert_video_to_gray(in_video, out_video)
23
24
25 if __name__ == "__main__":
26     main()
27
    
```

Read video properties

```

Run: main
/media/nr1/SSD_500GB/VSB/vyuka_2021_letni/PytorchProjects/venv_zao/bin/python /media/nr1/SSD_500GB/VSB/vyuka_2021_letni/PytorchProjects/01_zao_video/main.py
fps: 24.0, frame_count: 6748.0
frame_width: 1920.0, frame_height: 1080.0
Process finished with exit code 0
    
```

Opencv + Python

```

4 def convert_video_to_gray(in_video, out_video):
5     video_cap = cv2.VideoCapture(in_video)
6     # if not video_cap.isOpened():
7     if video_cap.isOpened() == False:
8         print("video_cap: False")
9
10    fps = video_cap.get(cv2.CAP_PROP_FPS)
11    frame_count = video_cap.get(cv2.CAP_PROP_FRAME_COUNT)
12    frame_width = video_cap.get(cv2.CAP_PROP_FRAME_WIDTH)
13    frame_height = video_cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
14
15    print(f'fps: {fps}, frame_count: {frame_count}')
16    print(f'frame_width: {frame_width}, frame_height: {frame_height}')
17
18    frame_size = (int(frame_width), int(frame_height))
19    video_writer = cv2.VideoWriter(out_video,
20                                  cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),
21                                  fps,
22                                  frame_size,
23                                  False)
24
25    frame_num = 0
26    while (video_cap.isOpened()):
27        ret, frame = video_cap.read()
28        if ret:
29            frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
30            video_writer.write(frame_gray)
31            frame_num = frame_num + 1
32            print("frame_num: ", frame_num)

```

Save video

◆ VideoWriter() [2/3]

```

cv::VideoWriter::VideoWriter ( const String & filename,
                               int fourcc,
                               double fps,
                               Size frameSize,
                               bool isColor = true
                               )

```

Python:

```

cv.VideoWriter( ) -> <VideoWriter object>
cv.VideoWriter( filename, fourcc, fps, frameSize[, isColor] ) -> <VideoWriter object>
cv.VideoWriter( filename, apiPreference, fourcc, fps, frameSize[, isColor] ) -> <VideoWriter object>

```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- filename** Name of the output video file.
- fourcc** 4-character code of codec used to compress the frames. For example, `VideoWriter::fourcc('P','I','M','1')` is a MPEG-1 codec, `VideoWriter::fourcc('M','J','P','G')` is a motion-jpeg codec etc. List of codes can be obtained at [Video Codes by FOURCC page](#). FFMPEG backend with MP4 container natively uses other values as fourcc code: see `ObjectType`, so you may receive a warning message from OpenCV about fourcc code conversion.
- fps** Framerate of the created video stream.
- frameSize** Size of the video frames.
- isColor** If it is not zero, the encoder will expect and encode color frames, otherwise it will work with grayscale frames (the flag is currently supported on Windows only).

Tips:

- With some backends `fourcc=-1` pops up the codec selection dialog from the system.
- To save image sequence use a proper filename (eg. `img_%02d.jpg`) and `fourcc=0` OR `fps=0`. Use uncompressed image format (eg. `img_%02d.BMP`) to save raw frames.
- Most codecs are lossy. If you want lossless video file you need to use a lossless codecs (eg. FFMPEG FFV1, Huffman HFYU, Lagarith LAGS, etc...)
- If FFMPEG is enabled, using `codec=0; fps=0;` you can create an uncompressed (raw) video file.

Opencv + Python

```

4 def convert_video_to_gray(in_video, out_video):
5     video_cap = cv2.VideoCapture(in_video)
6     # if not video_cap.isOpened():
7     if video_cap.isOpened() == False:
8         print("video_cap: False")
9
10    fps = video_cap.get(cv2.CAP_PROP_FPS)
11    frame_count = video_cap.get(cv2.CAP_PROP_FRAME_COUNT)
12    frame_width = video_cap.get(cv2.CAP_PROP_FRAME_WIDTH)
13    frame_height = video_cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
14
15    print(f'fps: {fps}, frame_count: {frame_count}')
16    print(f'frame_width: {frame_width}, frame_height: {frame_height}')
17
18    frame_size = (int(frame_width), int(frame_height))
19    video_writer = cv2.VideoWriter(out_video,
20                                  cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),
21                                  fps,
22                                  frame_size,
23                                  False)
24
25    frame_num = 0
26    while (video_cap.isOpened()):
27        ret, frame = video_cap.read()
28        if ret:
29            frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
30            video_writer.write(frame_gray)
31            frame_num = frame_num + 1
32            print("frame_num: ", frame_num)

```

Save video

◆ VideoWriter() [2/3]

```

cv::VideoWriter::VideoWriter ( const String & filename,
                               int fourcc,
                               double fps,
                               Size frameSize,
                               bool isColor = true
                               )

```

Python:

```

cv.VideoWriter( ) -> <VideoWriter object>
cv.VideoWriter( filename, fourcc, fps, frameSize[, isColor] ) -> <VideoWriter object>
cv.VideoWriter( filename, apiPreference, fourcc, fps, frameSize[, isColor] ) -> <VideoWriter object>

```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

- filename** Name of the output video file.
- fourcc** 4-character code of codec used to compress the frames. For example, `VideoWriter::fourcc('P','I','M','1')` is a MPEG-1 codec, `VideoWriter::fourcc('M','J','P','G')` is a motion-jpeg codec etc. List of codes can be obtained at [Video Codecs](#) by FOURCC page. FFMPEG backend with MP4 container natively uses other values as fourcc code: see `ObjectType`, so you may receive a warning message from OpenCV about fourcc code conversion.
- fps** Framerate of the created video stream.
- frameSize** Size of the video frames.
- isColor** If it is not zero, the encoder will expect and encode color frames, otherwise it will work with grayscale frames (the flag is currently supported on Windows only).

Tips:

- With some backends `fourcc=-1` pops up the codec selection dialog from the system.
- To save image sequence use a proper filename (eg. `img_%02d.jpg`) and `fourcc=0` OR `fps=0`. Use uncompressed image format (eg. `img_%02d.BMP`) to save raw frames.
- Most codecs are lossy. If you want lossless video file you need to use a lossless codecs (eg. FFMPEG FFV1, Huffman HFYU, Lagarith LAGS, etc...)
- If FFMPEG is enabled, using `codec=0`; `fps=0`; you can create an uncompressed (raw) video file.