Hardware II Seminar

Sensors and Data Analysis

FAB LAB BCN

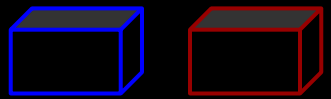
Iaac

Óscar González & Daniil Koshelyuk

Quick recap

Image and Video processing

Interfacing with cameras

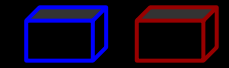


Basics of image processing

FAB LAB BCN

Iaac

Image and Video processing - Recap



CAMERA

COMPUTER

Video capture and storage of RGB camera example:

python

```
import cv2

cap = cv2.VideoCapture(0)
if (cap.isOpened() == False):
    print("Unable to read camera feed")

frame_width = int(cap.get(3))
frame_height = int(cap.get(4))

out =
cv2.VideoWriter('out.avi',cv2.VideoWriter_fourcc('M'
,'J','P','G'), 10, (frame_width,frame_height))

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        frame = cv2.flip(frame,0)
        out.write(frame)
        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
out.release()
cv2.destroyAllWindows()
```

OpenCV with python

Open video capture on device 0 (webcam in my case)

VideoWriter (see https://www.fourcc.org/fourcc.php)

Capture, store and show frame

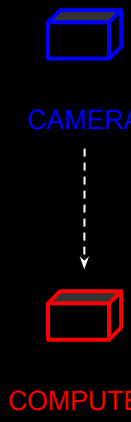
Close everything if `q` is pressed

Full example at: https://gitlab.fabcloud.org/barcelonaworkshops/code-club/-/tree/2020/04_computer_vision

FAB LAB BCN

Iaac

Image and Video processing - Some use cases



Here are some of the possible use cases:

- *Feature detection: descriptors techniques*
- *Image alignment: align images in pictures, homography*
- *Object detection: detect object in images (where something is, knowing what we are looking for) - solvable with DL, but ML works too*
- *Identification: identify if one object matches another*

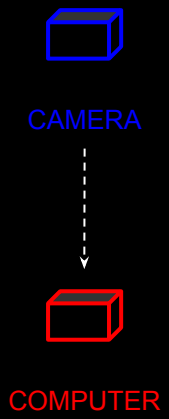
Understanding the basics

Descriptors

FAB LAB BCN

Iaac

Image and Video processing - Understanding basics

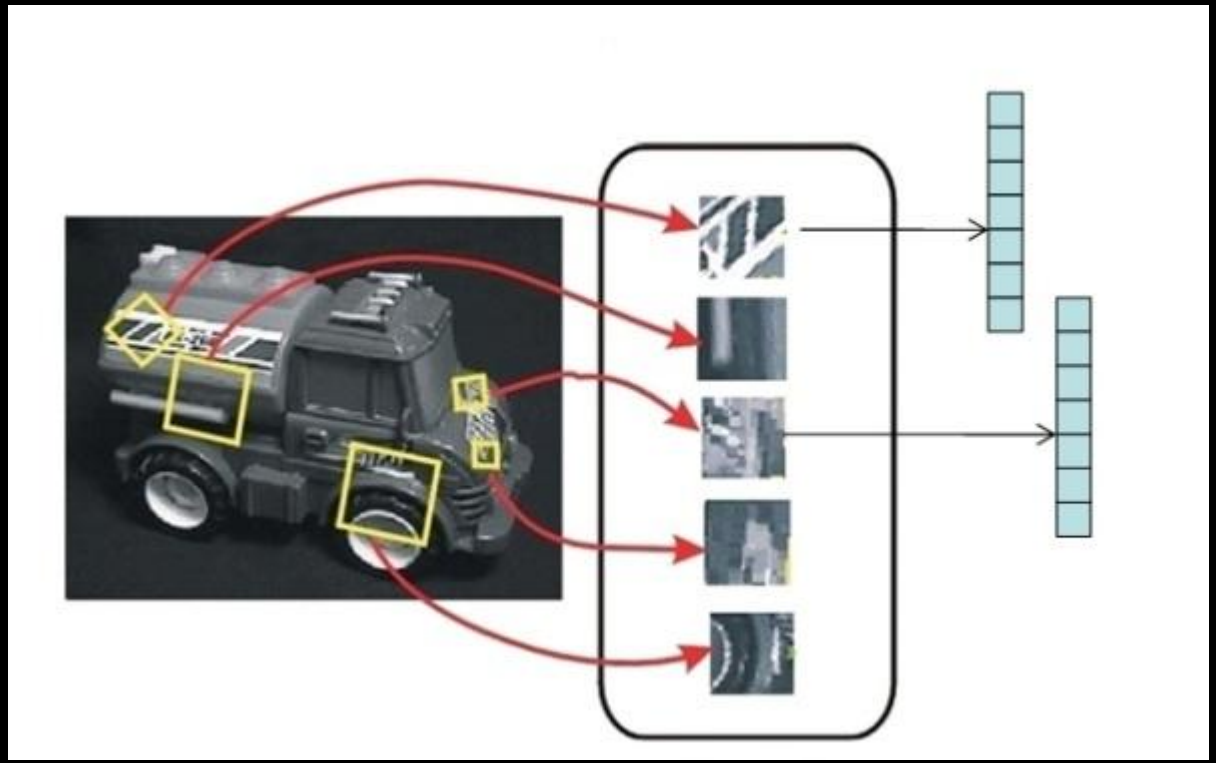


Descriptors

When dealing with image processing problems, a common approach is to divide the image in **subimages or patches**. Each of the patches is a small portion of the original image which can then be used and compared to other patches of other images and extract conclusions. Are the objects the same? Are the objects of the same type? Ideally, we would need of a method to determine if a patch is the same to another one, and that this method is independent of image changes: rotation, lighting, noise… Welcome **descriptors**.

More at:https://en.wikipedia.org/wiki/Visual_descriptor

Image and Video processing - Understanding basics



*> A descriptor is **some function that is applied on the patch to describe** it in a way that is invariant to all the image changes that are suitable to our application (e.g. rotation, illumination, noise etc.). A descriptor is "built-in" with a distance function to determine the similarity, or distance, of two computed descriptors. So to compare two image patches, we'll compute their descriptors and measure their similarity by measuring the descriptor similarity, which in turn is done by computing their descriptor distance.*

From:https://gilscvblog.com/

Image and Video processing - Understanding basics

Some descriptor types

- HOG: Histogram of Oriented Gradients. Nice wrap-up *here*
  - SIFT (non free)
  - SURF (non free)
- Binary descriptors: Are a way of representing a patch as a binary string, using only comparison of intensity (in separated channels if necessary)
  - BRIEF
  - ORB
  - BRISK
  - FREAK
  - ...

They are based on the following workflow. 1. Sampling pattern definition: where to sample points in the region around the descriptor. 2. Orientation compensation: some mechanism to measure the orientation of the keypoint and rotate it to compensate for rotation changes. 3. Sampling pairs: the pairs to compare when building the final descriptor.

FAB LAB BCN

Iaac

Image and Video processing - Understanding basics

Basic feature finding example

CAMERA

COMPUTER

```python
#!/usr/bin/python

import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('images/eagle.jpg',0)
img = cv.GaussianBlur(img, (3, 3), 0)

# Initiate ORB detector
orb = cv.ORB_create()
# find the keypoints with ORB
kp = orb.detect(img,None)
# compute the descriptors with ORB
kp, des = orb.compute(img, kp)
# draw only keypoints location,not size
and orientation
img2 = cv.drawKeypoints(img, kp, None,
color=(0,255,0), flags=0)
cv.imshow('Features', img2)

…
cv.imwrite("images/eaglefeatures.jpg",
img2)
```

OpenCV with python

Open image and filter it

Initiate ORB and detect keypoint

Show result keypoings, and save

FAB LAB BCN

Iaac

Full example: https://github.com/oscgonfer/sensors_dsp_lectures/blob/current/03_computer_vision/examples/03_feature.py

Image and Video processing - Understanding basics

Matching features in two images

CAMERA

COMPUTER

```
#!/usr/bin/python                                    python

import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

img1 =
cv.imread('images/eaglerot.jpg',cv.IMREAD_
GRAYSCALE)
img1 = cv.GaussianBlur(img1, (3,3), 0)
…
orb = cv.ORB_create()
kp1 = orb.detect(img1,None)
kp1, des1 = orb.compute(img1, kp1)
…
bf = cv.BFMatcher(cv.NORM_HAMMING,
crossCheck=True)
matches = bf.match(des1,des2)
matches = sorted(matches, key = lambda
x:x.distance)
img3 =
cv.drawMatches(img1,kp1,img2,kp2,matches[:
10],None,flags=cv.DrawMatchesFlags_NOT_DRA
W_SINGLE_POINTS)
plt.imshow(img3),plt.show()
```

OpenCV with python

Open image 1 (and also image 2)

Initiate ORB and detect keypoints in both images

Get matches. Sort them in the order of their distance and draw them

Full example: https://github.com/oscgonfer/sensors_dsp_lectures/blob/current/03_computer_vision/examples/04_matches.py

FAB LAB BCN

Iaac

Image and Video processing - Understanding basics

Recommended content

- *OpenCV Documentation: https://docs.opencv.org/3.4/db/d27/tutorial_py_table_of_contents_feature2d.html*
- *Learn OpenCV: https://learnopencv.com*
- *GilsCV Blog:https://gilscvblog.com/*

> Try to understand well these basics to know what you are doing!

FAB LAB BCN

Iaac

Some techniques

Homography

Image and Video processing - Some techniques

Homography is a transformation (a matrix) that maps the points in one image to the corresponding points in the other image.

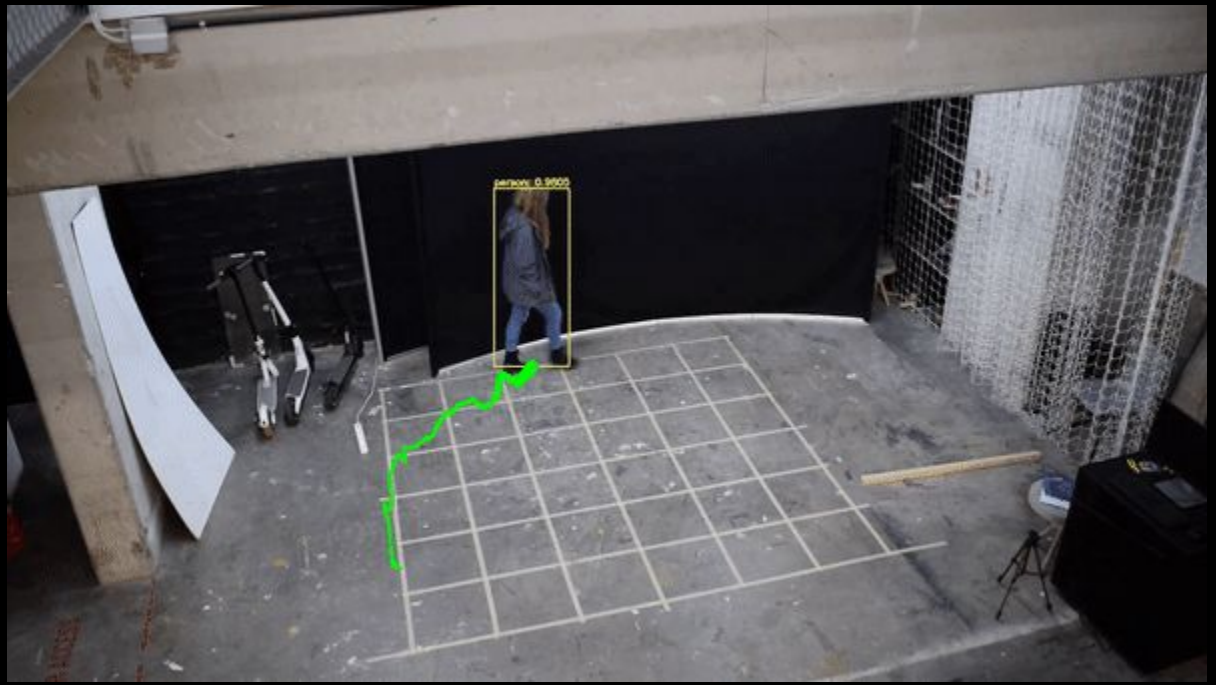


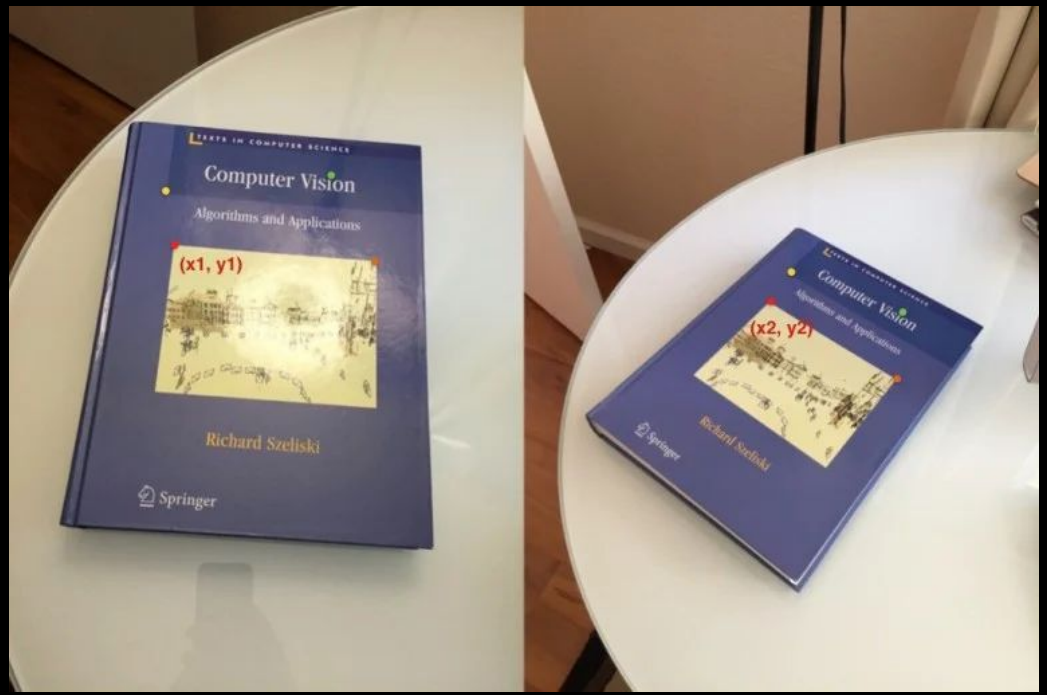
Image and Video processing - Some techniques

Homography is a transformation (a matrix) that maps the points in one image to the corresponding points in the other image

Some examples:

https://www.learnopencv.com/homography-examples-using-opencv-python-c/

https://www.learnopencv.com/image-alignment-feature-based-using-opencv-c-python/



Some techniques

Tracking things

FAB LAB BCN

Iaac

Image and Video processing - Tracking things

Ultimately, we can try to detect things in a image, or video frame (the same thing). We can track objects in an image by tracking color, pixel changes, or trying to build more complex trackers, all the way up to deep learning algorithms.

Remember, tracking, in general, doesn't imply identifying what the tracked object is, so normally it's a more light weight process.

Image and Video processing - Color tracking

Source:
https://docs.opencv.org/trunk/da/d97/tutorial_threshold_inRange.html

CAMERA

COMPUTER

```python
#!/usr/bin/python

import cv2 as cv
…

cap = cv.VideoCapture(args.camera)
while True:
    ret, frame = cap.read()
    if frame is None:
        break
    frame_HSV = cv.cvtColor(frame, cv.COLOR_BGR2HSV)
    frame_threshold = cv.inRange(frame_HSV, (low_H,
low_S, low_V), (high_H, high_S, high_V))

    bn_img = cv.erode
(frame_threshold,cv.getStructuringElement(cv.MORPH_RE
CT,(3,3)),iterations = 1)
    bn_img = cv.dilate
(bn_img,cv.getStructuringElement(cv.MORPH_RECT,(5,5))
,iterations = 1)
    M = cv.moments(bn_img)
    if M['m00']>50000:
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        cv.circle (frame,(cx,cy),20,(255,0,0), 2)

    cv.imshow(window_capture_name, frame)
    cv.imshow(window_detection_name, frame_threshold)

    key = cv.waitKey(30)
    if key == ord('q') or key == 27:
        break
```

OpenCV with python

Open video and start loop

Convert to HSV, get pixels within Range. Clean Image

Get moments, centroid and draw a circle. Then show

Full example:
https://github.com/oscgonfer/sensors_dsp_lectures/blob/current/03_computer_vision/examples/06_color_tracking.py

FAB LAB BCN

Iaac

Image and Video processing - Color tracking / use cases

For an image, we get all the pixels that fall between a certain range of HSV or RGB values.

Then, we calculate the centroid of those coordinates.

This technique is only useful if there are very distinct colors, but is very fast if images are not super-mega high-resolution.

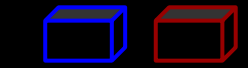
https://www.newscientist.com/article/dn21877-go-for-a-jog-with-a-helicopter-drone/

FAB LAB BCN

Iaac

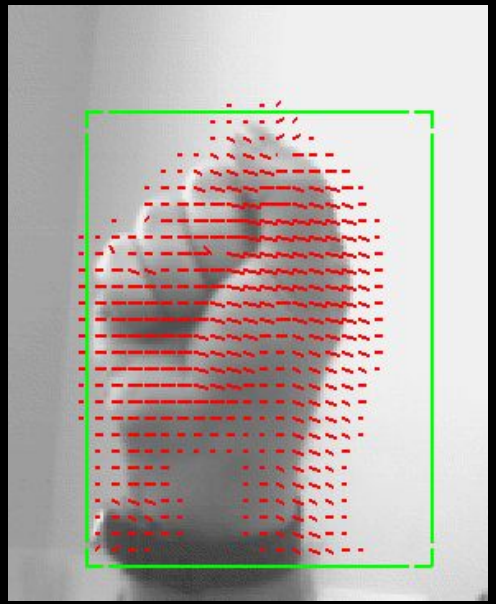
Source:
https://docs.opencv.org/4.2.0/d4/dee/tutorial_optical_flow.html

Image and Video processing - Optical Flow



OpenCV with python

Optical flow or optic flow *is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene.*



FAB LAB BCN

Iaac

Full example:
https://github.com/oscgonfer/sensors_dsp_lectures/blob/current/03_computer_vision/examples/08_optical_flow_lk.py

Source:
https://docs.opencv.org/4.2.0/d4/dee/tutorial_optical_flow.html

Image and Video processing - Optical Flow

CAMERA

COMPUTER

```python
#!/usr/bin/python

import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)
ret, old_frame = cap.read()
old_gray = cv.cvtColor(old_frame, cv.COLOR_BGR2GRAY)
p0 = cv.goodFeaturesToTrack(old_gray, mask = None,
**feature_params)

while(1):
    ret,frame = cap.read()
    frame_gray = cv.cvtColor(frame,
cv.COLOR_BGR2GRAY)
    p1, st, err = cv.calcOpticalFlowPyrLK(old_gray,
frame_gray, p0, None, **lk_params)
    if p1 is not None:
      good_new = p1[st==1]
      good_old = p0[st==1]

      for i,(new,old) in enumerate(zip(good_new,
good_old)):
          a,b = new.ravel()
          c,d = old.ravel()
          mask = cv.line(mask, (a,b),(c,d),
color[i].tolist(), 2)
            frame =
cv.circle(frame,(a,b),5,color[i].tolist(),-1)
        img = cv.add(frame,mask)
        cv.imshow('frame',img)
        old_gray = frame_gray.copy()
        p0 = good_new.reshape(-1,1,2)
```

OpenCV with python

Open video and get features to track

Get optical flow and get good points.
Draw lines

Replace new frame and start over

Full example:
https://github.com/oscgonfer/sensors_dsp_lectures/blob/current/03_computer_vision/examples/08_optical_flow_lk.py

FAB LAB BCN

Iaac

Some techniques

Identifying things

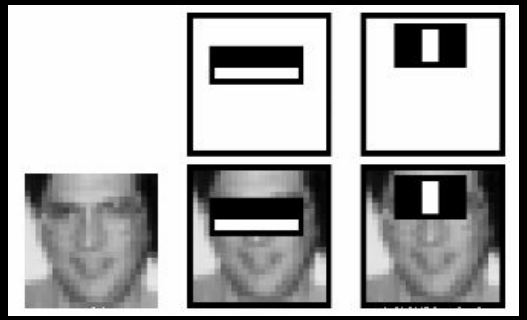
Source:
https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

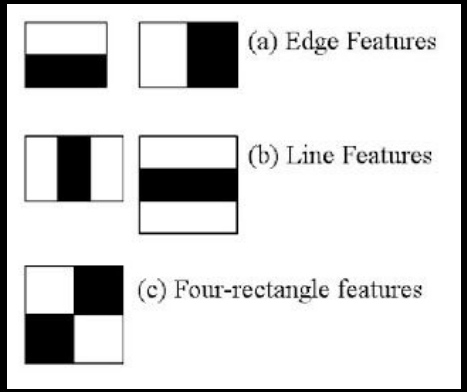
Image and Video processing - Identifying things

When identifying things, it's better to try to find classifiers that try to find something very specific.

The more specific, the faster it normally will be, for instance, using ML technique of Cascade Classifiers.

> Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.





FAB LAB BCN

Iaac

Image and Video processing - Haar Cascade Classifier

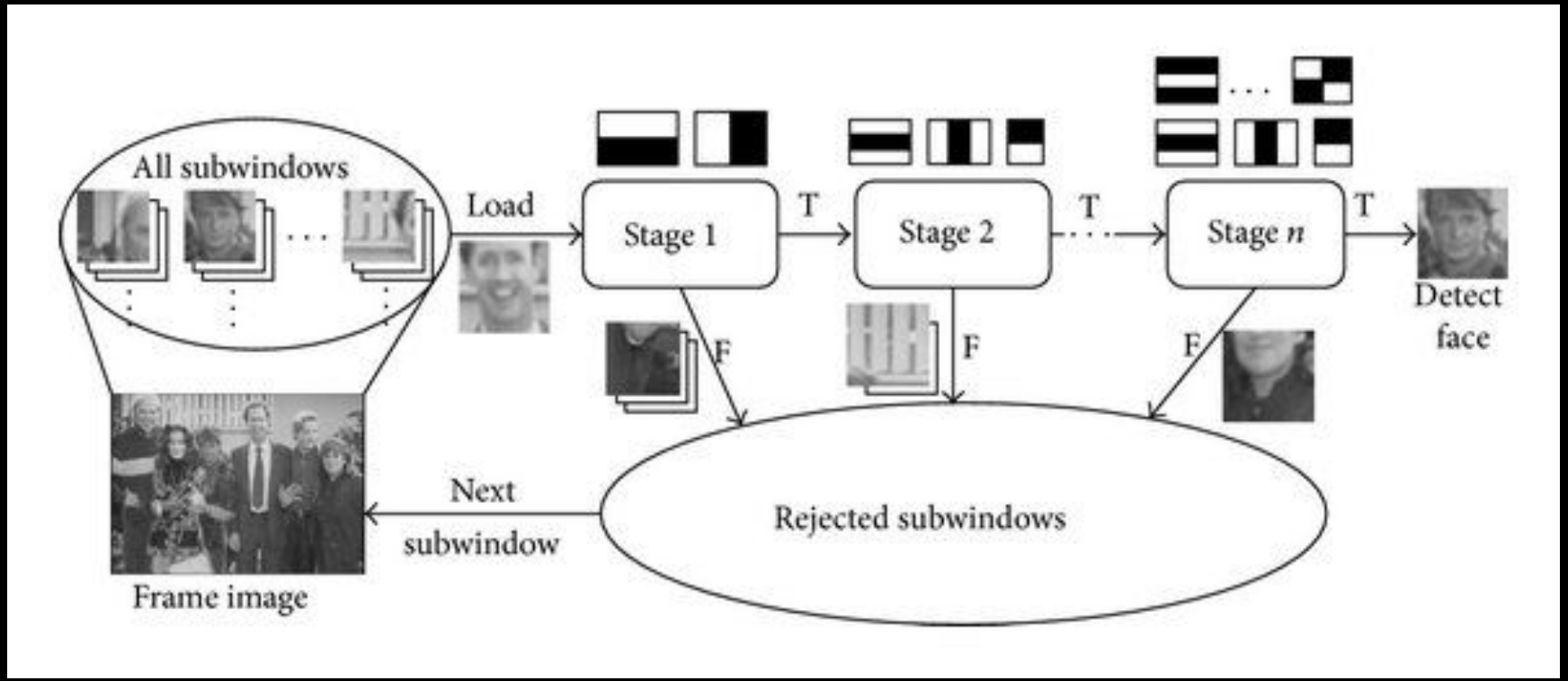


Image and Video processing - Identifying things

FAB LAB BCN

Iaac

Image and Video processing - Identifying things

Source:
https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

Image and Video processing - Cascade classifier

CAMERA

COMPUTER

python

Full example:
https://github.com/oscgonfer/sensors_dsp_lectures/blob/current/03_computer_vision/examples/07_classifier.py

OpenCV with python

FAB LAB BCN

Iaac

Some techniques

Identifying things - the DL way

FAB LAB BCN

Iaac

Image and Video processing - YOLO

> YOLO performs object detection by creating a rectangular grid throughout the entire image. Then creates bounding boxes based on (x,y) coordinates. Class probability gets mapped by using random color assignment. To filter out weak detections, a 50% confidence is being used (this can change) which helps eliminate unnecessary boxes.

The paper: https://arxiv.org/abs/1804.02767
The source: https://pjreddie.com/darknet/yolo/

Image and Video processing - YOLO

Installation:

1. Install Darknet: https://pjreddie.com/darknet/install/
   a. For weird image formats, use it with OPENCV (optional). For this, modify Makefile.
2. If using `opencv` and `opencv-python`, it comes already with darknet

Weights and cfg (remember, this is DL)

Get the **weights** from here:

```
wget https://pjreddie.com/media/files/yolov3.weights
```

Get the yolov3.cfg from here:

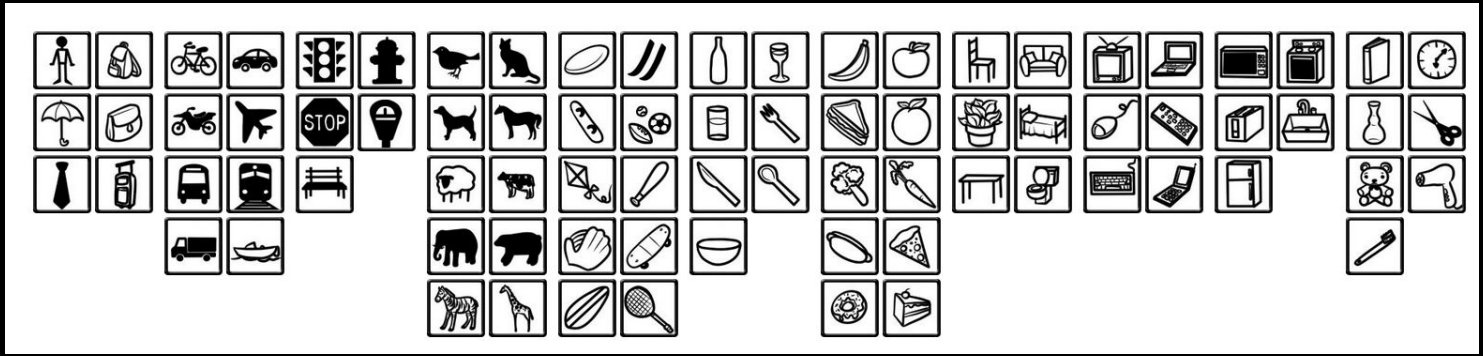
```
wget https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg
```

Check in the Darknet github for more cfgs.

Image and Video processing - YOLO

COCO dataset (COmmon objects in COntext) - *https://cocodataset.org/*

COCO dataset is trained on 200,000 images of 80 different categories (person, suitcase, umbrella, horse etc…). It contains images, bounding boxes and 80 labels. COCO can be used for both object detection, and segmentation, we are using it for detecting people.



We will be using the COCO dataset. You will find coco.names in the extras folder of the repository

FAB LAB BCN

Iaac

Image and Video processing - YOLO

Full image example: python

https://github.com/oscgonfer/sensors_dsp_lectures/blob/current/03_computer_vision/examples/09_yolo.py

Full video example: python

https://github.com/oscgonfer/sensors_dsp_lectures/blob/current/03_computer_vision/examples/10_yolo_video.py

OpenCV with python

FAB LAB BCN

Iaac

References

- *COCO - Common Objects in Context*
- *Learning Computer Vision Basics in Excel*
- *Corner Detector*
- *Real time person removal*
- *Introduction to descriptors* and *Tutorial on descriptors*
- *GilCV Code Examples*
- *Viola-Jones face detection and tracking explained*
- *Using non-free detectors*

References - Working on a pi

- *Rpi camera do's and dont's*
- *Optimizing opencv for the pi*
- *Face Recognition*
- *Object Detection with DL*
- *Facial landmarks*

Thanks!

FAB LAB BCN

Iaac