SIT32004
ICT Application Development

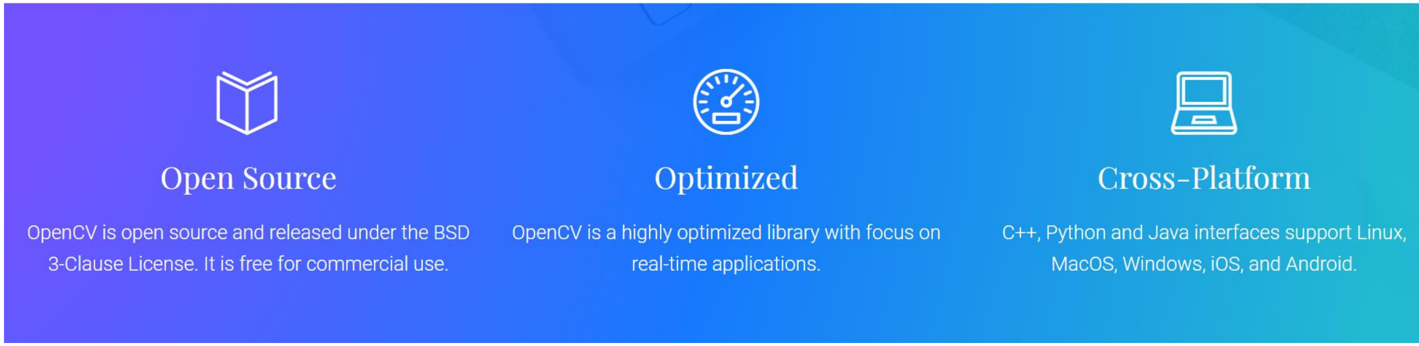
Advanced Image Processing and
Video Processing

Prof. Changbeom Choi

OpenCV[1]

- OpenCV is …
  - An open source computer vision and machine learning software library
  - OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
  - OpenCV was started at Intel in 1999
  - 2500+ optimized computer vision and machine learning algorithms

OpenCV



Open Source

OpenCV is open source and released under the BSD 3-Clause License. It is free for commercial use.

Optimized

OpenCV is a highly optimized library with focus on real-time applications.

Cross-Platform

C++, Python and Java interfaces support Linux, MacOS, Windows, iOS, and Android.

Mouse Event Handling in OpenCV

- Register callback functions
  - cv2.setMouseCallback(*window_name, callback_fn, param*)
    - » window_name: window that cv2 uses
      - Ex) cv2.namedWindow('frame’)
    - » callback_fn: callback function
      - Callback function’s argument should be (event,x,y,flags,param)
      - Event list
        - 'EVENT_FLAG_ALTKEY', 'EVENT_FLAG_CTRLKEY', 'EVENT_FLAG_LBUTTON', 'EVENT_FLAG_MBUTTON', 'EVENT_FLAG_RBUTTON', 'EVENT_FLAG_SHIFTKEY', 'EVENT_LBUTTONDBLCLK', 'EVENT_LBUTTONDOWN', 'EVENT_LBUTTONUP', 'EVENT_MBUTTONDBLCLK', 'EVENT_MBUTTONDOWN', 'EVENT_MBUTTONUP', 'EVENT_MOUSEHWHEEL', 'EVENT_MOUSEMOVE', 'EVENT_MOUSEWHEEL', 'EVENT_RBUTTONDBLCLK', 'EVENT_RBUTTONDOWN', 'EVENT_RBUTTONUP'
    - » param: additional parameters which you may pass

Mouse Click and Positions

- Requirements
  - Let a callback function to handle events from opencv window
    - Use setMouseCallback method
      - cv2.setMouseCallback(*NAMED_WINDOW*, *FUNCTION_OBJECT*, *PARAMS*)
  - Develop a logic to handle mouse click event
    - 'EVENT_LBUTTONDOWN’
    - 'EVENT_MOUSEMOVE’
    - 'EVENT_LBUTTONUP'

Mouse Event Handling in OpenCV

- Register callback functions
  - cv2.setMouseCallback(*window_name, callback_fn, param*)

```
import numpy as np
import cv2

def draw_circle(event,x,y,flags,param):
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(img,(x,y), 100,(0,255,255),-1)
        cv2.imshow('img', img)
        print(x, y)

# Create a black image
img = np.zeros((768,1024,3), np.uint8)

cv2.namedWindow('img')
cv2.setMouseCallback('img', draw_circle, None)

cv2.imshow('img',img)
cv2.waitKey(0)
```

Image Processing

- **Scaling**
  - Scaling is just resizing of the image
  - OpenCV provides **cv2.resize()**
    - The size of the image can be specified manually, or you can specify the scaling factor.
    - Different interpolation methods may be used
      - **cv2.INTER_AREA:** Suitable for shrinking
      - **cv2.INTER_LINEAR**: Suitable for zooming
      - **cv2.INTER_CUBIC:** Suitable for zooming but slow

```
import cv2
import numpy as np

img = cv2.imread('pikachu1.png')

res = cv2.resize(img,None,fx=2, fy=2, interpolation = cv2.INTER_CUBIC)
cv2.imshow('zoom', res)

res = cv2.resize(img,None,fx=0.5, fy=0.5, interpolation = cv2.INTER_AREA)
cv2.imshow('shrink', res)

cv2.waitKey(0)
```

Image Processing

- Perspective Transformation

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

img = cv2.imread('sudokusmall.png')
rows,cols,ch = img.shape

pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
pts2 = np.float32([[0,0],[300,0],[0,300],[300,300]])

M = cv2.getPerspectiveTransform(pts1,pts2)

dst = cv2.warpPerspective(img,M,(300,300))

plt.subplot(121),plt.imshow(img),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
```

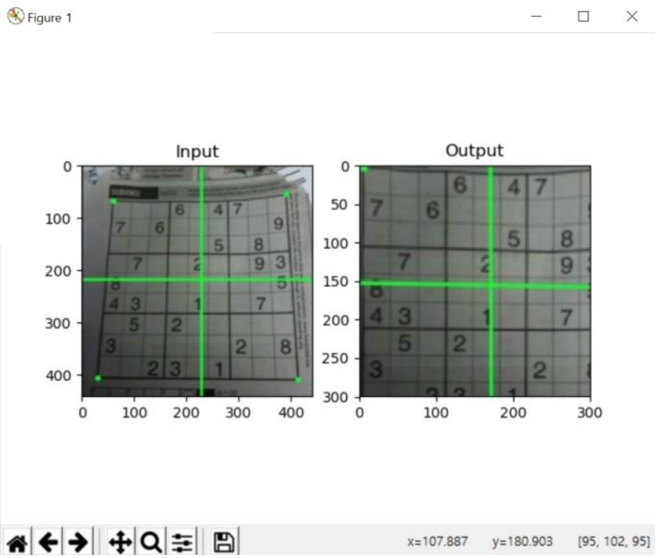
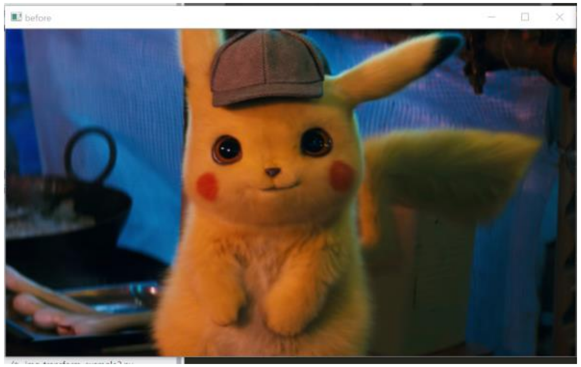
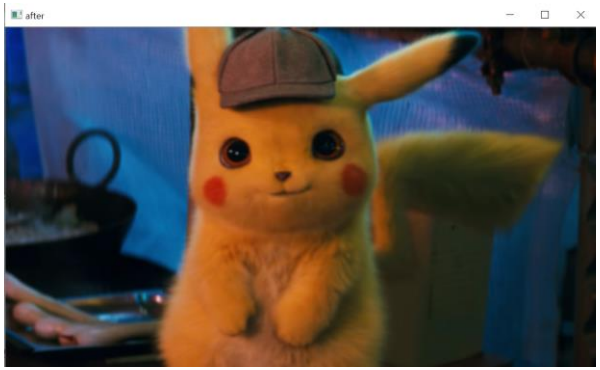


Image Processing

- Image Smoothing
  - **Averaging**
    - » cv2.blur(src, ksize)
      - src: source image
      - ksize: Kernel Size
  - **Gaussian**
    - » cv2.GaussianBlur(img, ksize, sigmaX)
      - Img: source image
      - ksize – (width, height), should be positive odd number
  - **Median**
    - » cv2.medianBlur(src, ksize)
      - src: source image
      - ksize: Kernel Size, an odd number greater than 1



Before



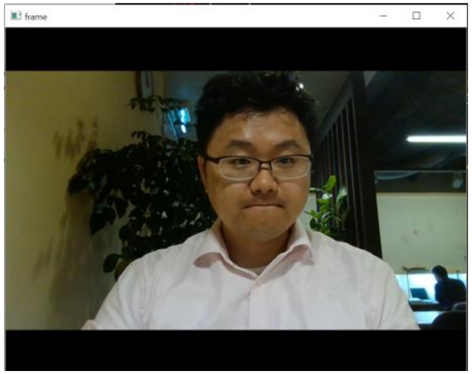
After: Gaussian Blur 5 X 5

Video Processing [2-4]

- Video is a Data Stream
- Terminology
  - Frame
    - a rectangular raster of pixels, which is a single still image
  - Frame rate
    - expressed in frames per second or fps
    - the frequency (rate) at which consecutive images(frames) appear on a display
  - Resolution
    - the detail an image holds
    - the frame is composed of picture elements, therefore, the resolution is equivalent to pixel count
  - CODEC
    - A codec is a device or computer program for encoding or decoding a digital data stream or signal.

Video

- OpenCV provides a very simple interface to capture livestream with camera
- VideoCapture object
  - Argument: device index or the name of a video file



```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()

    # Our operations on the frame come here
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2BGRA)

    # Display the resulting frame
    cv2.imshow('frame',gray)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

Video Capture

- Use VideoWriter object
  - You should specify the output filename and the CODEC(FourCC)
  - FourCC is a 4-byte code used to specify the video codec

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

# Define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi',fourcc, 20.0, (640,480))

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        frame = cv2.flip(frame,0) # flip the video source

        # write the flipped frame
        out.write(frame)

        cv2.imshow('frame',frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break

# Release everything if job is finished
cap.release()
out.release()
cv2.destroyAllWindows()
```

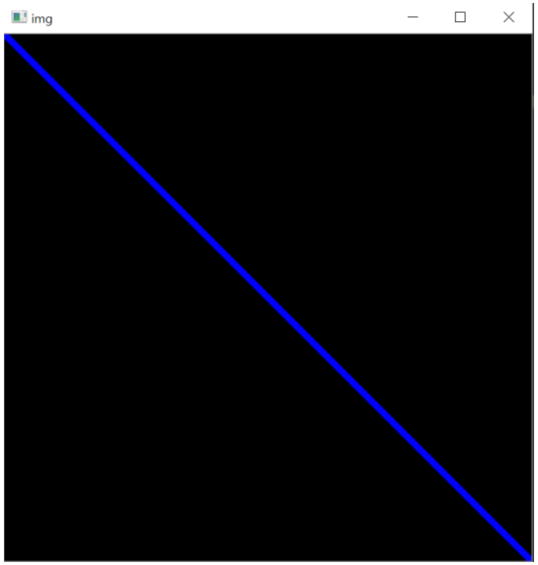
Drawing Functions in OpenCV

- **Drawing Line**

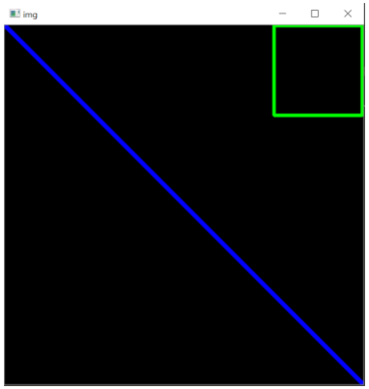
```
import numpy as np
import cv2

# Create a black image
img = np.zeros((512,512,3), np.uint8)

# Draw a diagonal blue line with thickness of 5 px
img = cv2.line(img,(0,0),(511,511),(255,0,0),5)
```

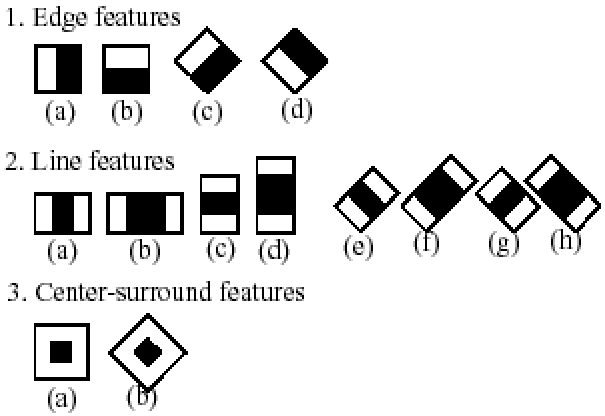


- **Drawing Rectangle**
  - `img = cv2.rectangle(img,(384,0),(510,128),(0,255,0),3)`



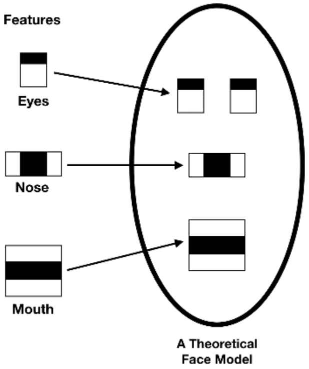
Face Recognition: Haar Cascade

- Viola–Jones algorithm
  - Haar Feature Selection
  - Creating an Integral Image
  - Adaboost Training
  - Cascading Classifiers



* Image from docs.opencv.org

- Haar Feature
  - All human faces share some similar properties
    - » The eye region is darker than the upper-cheeks.
    - » The nose bridge region is brighter than the eyes.
  - Composition of properties forming matchable facial features:
    - » Location and size: eyes, mouth, bridge of nose
    - » Value: oriented gradients of pixel intensities



* Image from https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177

Face Detection using Haar Cascade (1/2)

- Use CascadeClassifier
  - cv2.CascadeClassifier(CASCADE_XML’)
    - » You may change different Cascade Classifier
      - Face cascade classifier
      - Smile detect cascade classifier
      - Etc.
  - Example

```
face_cascade = cv2.CascadeClassifier('cascade.xml')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml’)

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (21, 21), 0)
faces = face_cascade.detectMultiScale(gray, 2, 2)
for (x,y,w,h) in faces:
    img = cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    smiles = smile_cascade.detectMultiScale(roi_gray, scaleFactor=1.2,
                                                  minNeighbors=22,
                                                  minSize=(25, 25))
        for (ex,ey,ew,eh) in smiles:
            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
```

Face Detection using Haar Cascade (2/2)

```
import numpy as np
import cv2

cv2.namedWindow('frame')

cap = cv2.VideoCapture(0)

face_cascade = cv2.CascadeClassifier('cascade.xml')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')

#cv2.imshow('face', f_img)

while(cap.isOpened()):
        ret, frame = cap.read()
        if ret==True:
                    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
                    gray = cv2.GaussianBlur(gray, (21, 21), 0)
                    faces = face_cascade.detectMultiScale(gray, 2, 2)
                    for (x,y,w,h) in faces:
                                img = cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
                                roi_gray = gray[y:y+h, x:x+w]
                                roi_color = img[y:y+h, x:x+w]
                                smiles = smile_cascade.detectMultiScale(roi_gray, scaleFactor=1.2,
                                                  minNeighbors=22,
                                                  minSize=(25, 25))
                                for (ex,ey,ew,eh) in smiles:
                                            cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

                    cv2.imshow('frame', frame)

                    if cv2.waitKey(1) & 0xFF == ord('q'):
                                break
        else:
                    break
```

Reference

- OpenCV, https://opencv.org/about/
- Film Frame, https://en.wikipedia.org/wiki/Film_frame
- Film Rate, https://en.wikipedia.org/wiki/Frame_rate
- Image Resolution, https://en.wikipedia.org/wiki/Image_resolution

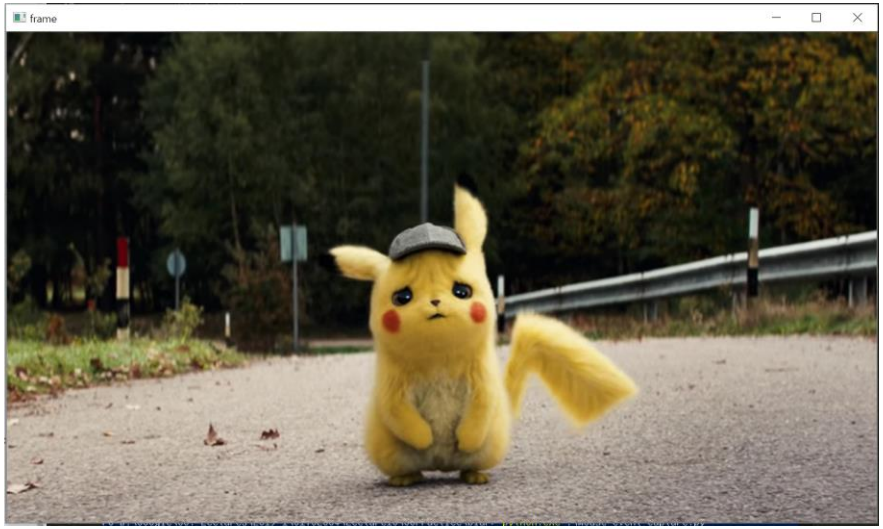
SIT32004
ICT Application Development

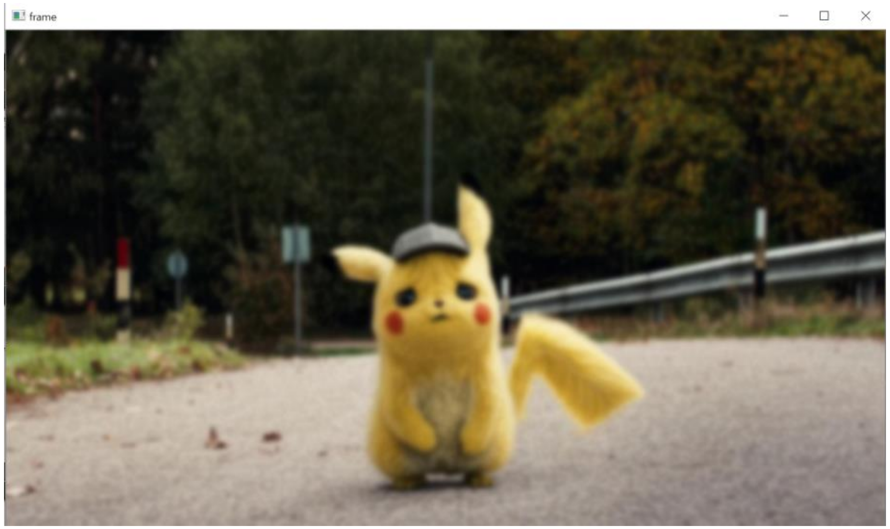
Practice

Prof. Changbeom Choi

Image blur using mouse click

- Your task is to develop an image processing software.
  - Your software should apply the blur effect to the image when you click the window
  - Use SetMouseCallback
    - » cv2.setMouseCallback(*SRC_WINDOW_NAME*, *CALLBACK_FN*, None)
  - Use Gaussian Blur
    - » cv2.GaussianBlur(*SRC*, (3, 3), 0)





Training Your own Haar Cascade

- To get accurate results, you should use
  - 1,000+ positive
  - 10,000+ negative images
- Parameter Tuning
  - npos <= (samples in vec file - 100)/(1+(nstages-1)*(1-minhitrate))) (Recommendation)

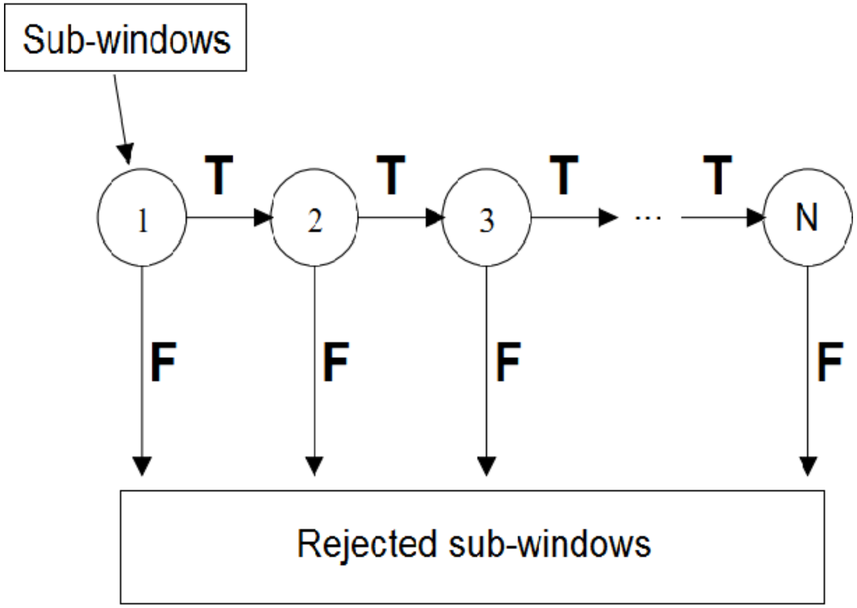
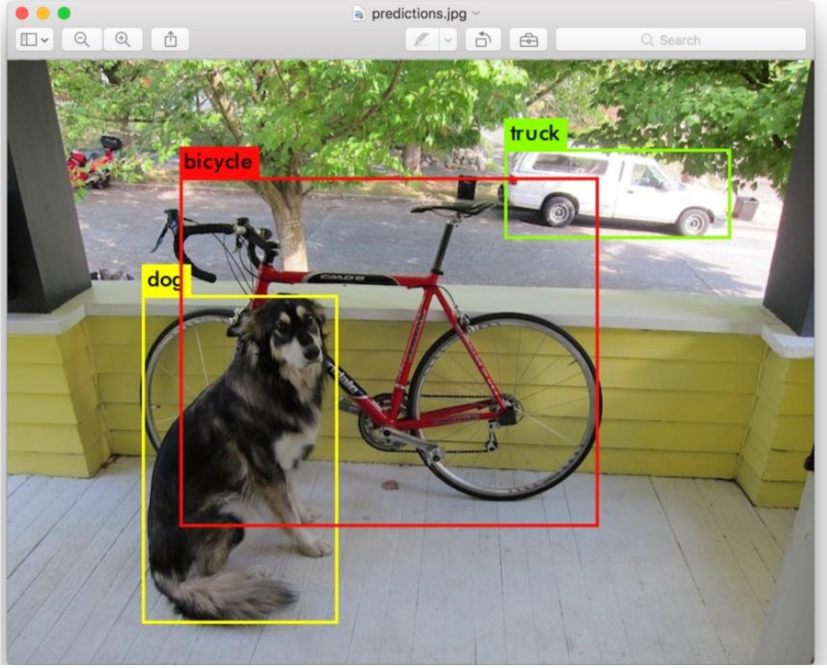


Fig. 3 Detection cascade

* Image from https://www.semanticscholar.org/paper/Evaluation-of-Haar-Cascade-Classifiers-Designed-for-Padilla-Filho/5b90bf3ebad1583beebcae5f892db2add248bcad

Object Detection with Deep Learning

- YOLO (You Only Look Once)



```
classes = None
  with open('yolov3.txt', 'r') as f:
      classes = [line.strip() for line in f.readlines()]

  # generate different colors for different classes
  COLORS = np.random.uniform(0, 255, size=(len(classes), 3))

  # read pre-trained model and config file
  net = cv2.dnn.readNet('yolov3.weights', 'yolov3.cfg')


  # create input blob
  blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)

  # set input blob for the network
  net.setInput(blob)

  # run inference through the network
  # and gather predictions from output layers
  outs = net.forward(get_output_layers(net))
```

Object Detection with Deep Learning

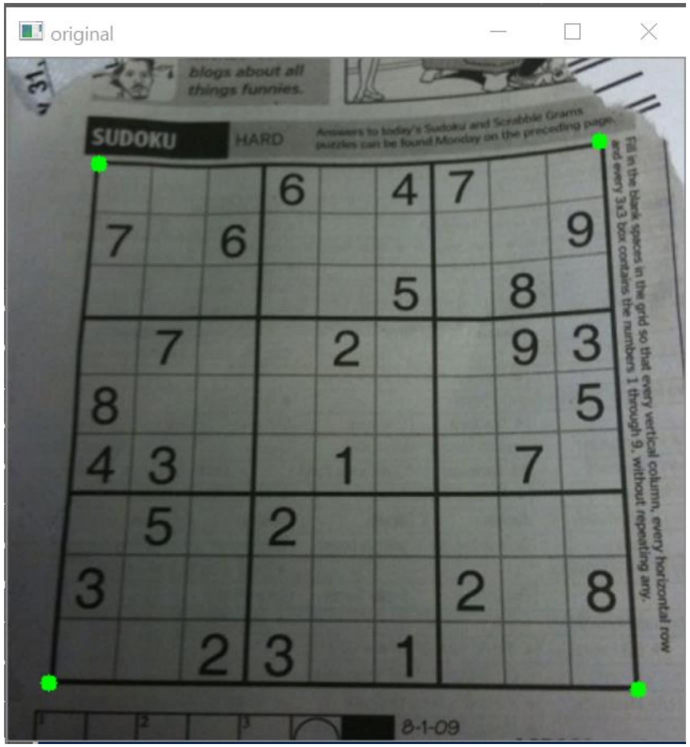
```
# for each detetion from each output layer
        # get the confidence, class id, bounding box params
        # and ignore weak detections (confidence < 0.5)
        for out in outs:
            for detection in out:
                scores = detection[5:]
                class_id = np.argmax(scores)
                confidence = scores[class_id]
                if confidence > 0.5:
                    center_x = int(detection[0] * Width)
                    center_y = int(detection[1] * Height)
                    w = int(detection[2] * Width)
                    h = int(detection[3] * Height)
                    x = center_x - w / 2
                    y = center_y - h / 2
                    class_ids.append(class_id)
                    confidences.append(float(confidence))
                    boxes.append([x, y, w, h])

        # apply non-max suppression
        indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

        # go through the detections remaining
        # after nms and draw bounding box
        for i in indices:
            i = i[0]
            box = boxes[i]
            x = box[0]
            y = box[1]
            w = box[2]
            h = box[3]
            draw_bounding_box(image, class_ids[i], confidences[i], round(x), round(y), round(x+w), round(y+h))
```

Homework01

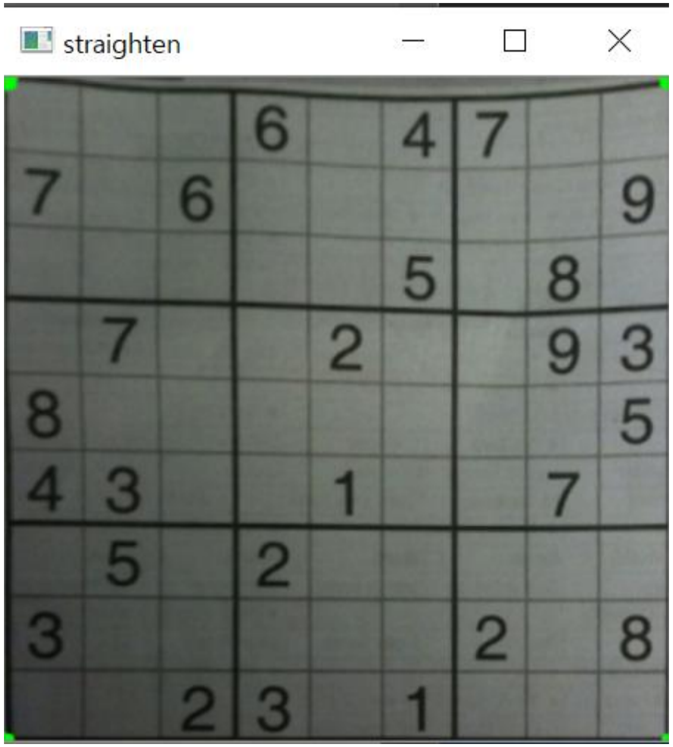
- StraightenImage
  - Use mouse to straighten an image
  - You should keep track four points
  - Use perspective transform
    - » `M = cv2.getPerspectiveTransform(pts1,pts2)`
    - » `dst = cv2.warpPerspective(img,M,(300,300))`



Answers to today's Sudoku and Scrabble Grams puzzles can be found Monday on the preceding page.

Fill in the blank spaces in the grid so that every vertical column, every horizontal row and every 3x3 box contains the numbers 1 through 9, without repeating any.

			6		4	7		
7		6						9
					5		8	
	7			2			9	3
8								5
4	3			1			7	
	5		2					
3						2		8
		2	3		1			



Homework02

- Train your own Haar Cascade Classifier