Day 4

Teemu Sarapisto

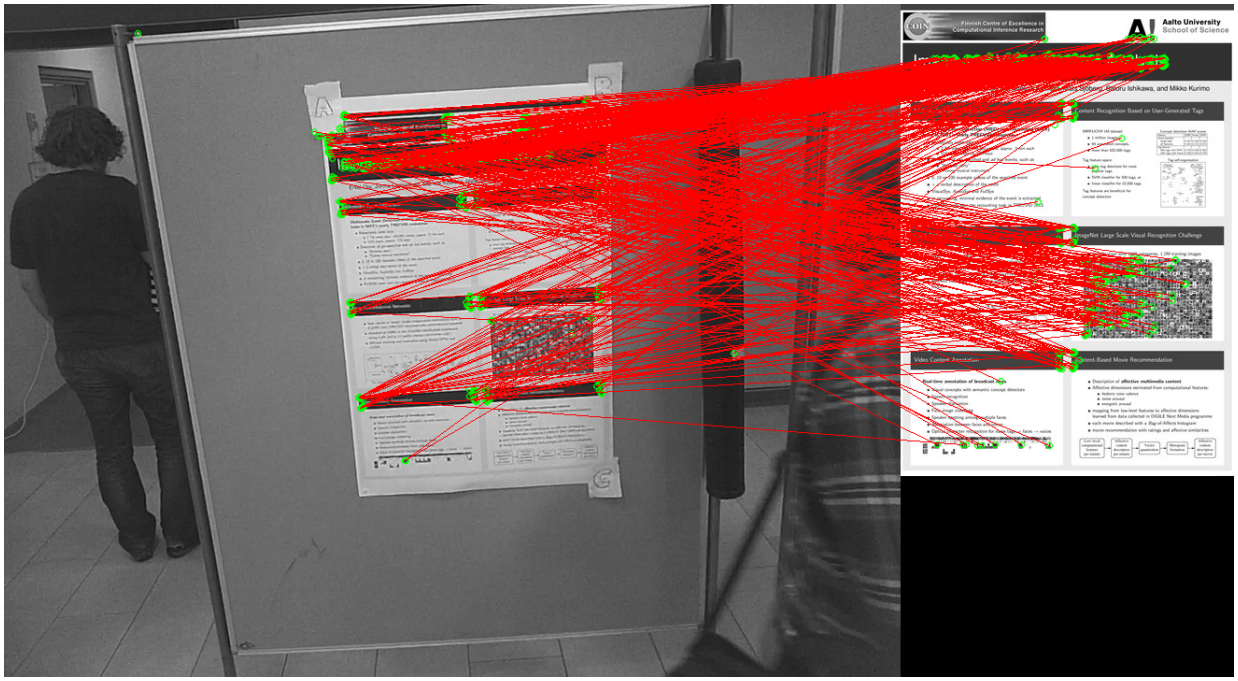
May 20, 2018

1 Day 4 hands-on

1.1 Keypoints for both images combined

```
poster = cv2.imread('poster.jpeg', cv2.IMREAD_GRAYSCALE)
frame = cv2.imread('frame.jpeg', cv2.IMREAD_GRAYSCALE)

#Use OpenCV's cv2.ORB_create() to create the detector and descriptor object orb
#Apply orb.detectAndCompute() to get the keypoints and their corresponding descriptors
orb = cv2.ORB_create()
poster_kp, poster_des = orb.detectAndCompute(poster, None)
frame_kp, frame_des = orb.detectAndCompute(frame, None)

#ORB returns by default 500 keypoints, which might be a good number, could be changed

#Visualize the detections using cv2.drawKeypoints()
poster2 = cv2.drawKeypoints(poster, poster_kp, None, color=(0, 255, 0), flags=0)
frame2 = cv2.drawKeypoints(frame, frame_kp, None, color=(0, 255, 0), flags=0)

poster2_resize = cv2.copyMakeBorder(poster2, 0, frame2.shape[0] - poster2.shape[0], 0,
combod = np.concatenate((frame2, poster2_resize), axis=1)
```

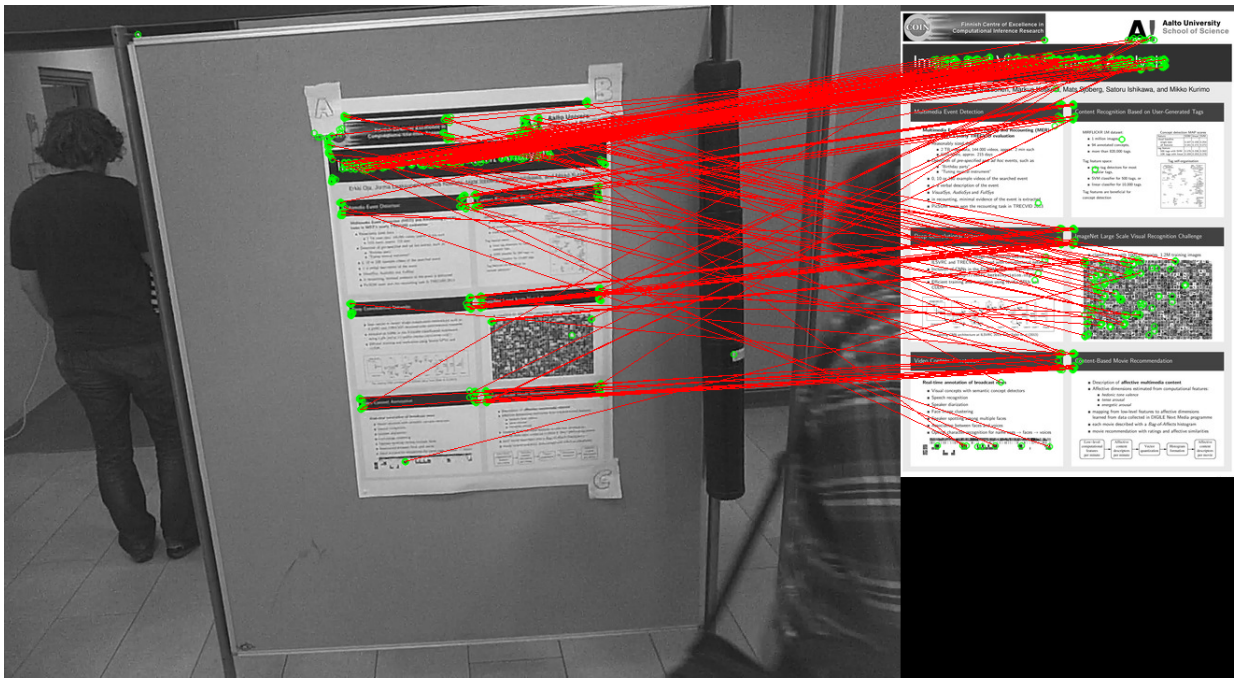
1.2 One way connect



```
knn = cv2.ml.KNearest_create()
poster_des_f32 = np.asarray(poster_des, dtype=np.float32)
frame_des_f32 = np.asarray(frame_des, dtype=np.float32)
poster_kp_vec = cv2.KeyPoint_convert(poster_kp)
frame_kp_vec = cv2.KeyPoint_convert(frame_kp)

labels = np.asarray(np.arange(0, len(poster_kp)), dtype=np.float32).reshape(500, 1)
knn.train(poster_des_f32, cv2.ml.ROW_SAMPLE, labels)
ret, results, neighbours, dist = knn.findNearest(frame_des_f32, k=1)

### Connect the matching keypoint pairs across the images
poster_offset = (frame2.shape[1], frame2.shape[0])
red = (0, 0, 255)
combod = np.concatenate((frame2, poster2_resize), axis=1)
poster_resize = cv2.copyMakeBorder(poster2, 0, frame2.shape[0] - poster2.shape[0], 0,

poster_idx = 0
for point in results:
    frame_idx = int(point[0])
    poster_kp_pos = poster_kp_vec[poster_idx]
    poster_img_pos = (int(poster_kp_pos[0] + poster_offset[0]), int(poster_kp_pos[1]))
```

```
    frame_img_pos = (int(frame_kp_vec[frame_idx][0]), int(frame_kp_vec[frame_idx][1]))
    print(poster_img_pos)
    print(frame_img_pos)
    cv2.line(combod, poster_img_pos, frame_img_pos, red, 1)
    poster_idx += 1

#cv2.imwrite('one_way_match_knn.png', combod)
```

1.3 Two way connect



```
Change to use a 1-NN matching also in the opposite direction by creating a k-NN model
Find the nearest descriptors in the first image for those in the second image, and vic

If a pair of descriptors, one in the first image ant the other in the second image, ar

Again connect the matching pairs and show results
knn_ftop = cv2.ml.KNearest_create()
knn_ftop.train(frame_des_f32, cv2.ml.ROW_SAMPLE, labels)
ret, results_ftop, neighbours, dist = knn_ftop.findNearest(poster_des_f32, k=1)

for frame_idx in range(len(results_ftop)):
    poster_idx = int(results_ftop[frame_idx][0])
```

```
            back_ref = int(results[poster_idx][0])
            print(frame_idx, poster_idx, back_ref)
            if frame_idx == back_ref:
                poster_kp_pos = poster_kp_vec[back_ref]
                frame_kp_pos = frame_kp_vec[poster_idx]
                poster_img_pos = (int(poster_kp_pos[0] + poster_offset[0]), int(poster_kp_pos
                frame_img_pos = (int(frame_kp_pos[0]), int(frame_kp_pos[1]))
                cv2.line(combod, poster_img_pos, frame_img_pos, red, 1)

    cv2.imwrite('two_way_match_knn.png', combod)
```

1.4 KNN from video

Example frame times:

```
frame processing time: 0.027774810791015625 seconds.
frame processing time: 0.02882528305053711 seconds.
frame processing time: 0.030350208282470703 seconds.
frame processing time: 0.02928614616394043 seconds.
frame processing time: 0.029371023178100586 seconds.
frame processing time: 0.029389619827270508 seconds.
```

1.5 Flann from video

Example frame times:

```
frame processing time: 0.03621077537536621 seconds.
frame processing time: 0.03606915473937988 seconds.
frame processing time: 0.03673076629638672 seconds.
frame processing time: 0.03754067420959473 seconds.
frame processing time: 0.03644680976867676 seconds.
frame processing time: 0.03664064407348633 seconds.
frame processing time: 0.036463022232055664 seconds.
```

Seems to be a bit slower.

1.6 Code for video flann and knn

```
def get_neighbours_knn(train_data, test_data):
    labels = np.asarray(np.arange(0, len(train_data)), dtype=np.float32).reshape(len(t
    knn = cv2.ml.KNearest_create()
```

```
    knn.train(train_data, cv2.ml.ROW_SAMPLE, labels)
    ret, results, neighbours, dist = knn.findNearest(test_data, k=1)
    return results

def get_neighbours_flann(train_data, test_data):
    FLANN_INDEX_KDTREE = 0
    index_params = dict(algorithm = FLANN_INDEX_KDTREE, trees = 5)
    search_params = dict(checks=50)   # or pass empty dictionary
    flann = cv2.FlannBasedMatcher(index_params,search_params)
    matches = flann.knnMatch(train_data, test_data, k=1)
    return matches

def process_frame_knn(frame, poster):
    poster_offset = (frame.shape[1], frame.shape[0])
    red = (0, 0, 255)
    poster_resize = cv2.copyMakeBorder(poster, 0, frame.shape[0] - poster.shape[0], 0,

    poster_kp, poster_des = orb.detectAndCompute(poster, None)
    frame_kp, frame_des = orb.detectAndCompute(frame, None)
    poster_kp_vec = cv2.KeyPoint_convert(poster_kp)
    frame_kp_vec = cv2.KeyPoint_convert(frame_kp)

    results1 = get_neighbours_knn(np.float32(poster_des), np.float32(frame_des))
    results2 = get_neighbours_knn(np.float32(frame_des), np.float32(poster_des))

    combod = np.concatenate((frame, poster_resize), axis=1)
    for frame_idx in range(len(results1)):
        poster_idx = int(results1[frame_idx][0])
        back_ref = int(results2[poster_idx][0])
        if frame_idx == back_ref:
            poster_kp_pos = poster_kp_vec[back_ref]
            frame_kp_pos = frame_kp_vec[poster_idx]
            poster_img_pos = (int(poster_kp_pos[0] + poster_offset[0]), int(poster_kp_
            frame_img_pos = (int(frame_kp_pos[0]), int(frame_kp_pos[1]))
            cv2.line(combod, poster_img_pos, frame_img_pos, red, 1)

    return combod


def process_frame_flann(frame, poster):
    poster_offset = (frame.shape[1], frame.shape[0])
    red = (0, 0, 255)
    poster_resize = cv2.copyMakeBorder(poster, 0, frame.shape[0] - poster.shape[0], 0,

    poster_kp, poster_des = orb.detectAndCompute(poster, None)
    frame_kp, frame_des = orb.detectAndCompute(frame, None)
```

```
    poster_kp_vec = cv2.KeyPoint_convert(poster_kp)
    frame_kp_vec = cv2.KeyPoint_convert(frame_kp)

    results1 = get_neighbours_flann(np.float32(poster_des), np.float32(frame_des))
    results2 = get_neighbours_flann(np.float32(frame_des), np.float32(poster_des))

    combod = np.concatenate((frame, poster_resize), axis=1)
    for frame_idx in range(len(results1)):
        poster_idx = results1[frame_idx][0].trainIdx
        back_ref = results2[poster_idx][0].trainIdx
        if frame_idx == back_ref:
            poster_kp_pos = poster_kp_vec[back_ref]
            frame_kp_pos = frame_kp_vec[poster_idx]
            poster_img_pos = (int(poster_kp_pos[0] + poster_offset[0]), int(poster_kp_
            frame_img_pos = (int(frame_kp_pos[0]), int(frame_kp_pos[1]))
            cv2.line(combod, poster_img_pos, frame_img_pos, red, 1)

    return combod

cap = cv2.VideoCapture('video.avi')

ret, frame = cap.read()
while ret:
    start_time = time.time()
    combo_pic = process_frame_flann(frame, poster2_resize)
    #combo_pic = process_frame_knn(frame, poster2_resize)
    print('frame processing time: %s seconds.' % (time.time() - start_time))
    cv2.imwrite('knn_video.png', combo_pic)
    cv2.waitKey(1)
    ret, frame = cap.read()
```

1.7 How long it took

Maybe 8 hours of not very intensive or focused working

2 Day 4 Homework

2.1 Face detect parameter experimentation

With the following code/parameters the program found 6 eyes (and a mouth it shouldn't have)

```
face_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_eye.xml')
img = cv2.imread('people.jpeg')
```

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.01, 250)
for (x,y,w,h) in faces:
    cv2.rectangle(img,(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]
    eyes = eye_cascade.detectMultiScale(roi_gray, 1.001, 45)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

2.2 Parameter explanation

From `https://stackoverflow.com/questions/20801015/recommended-values-for-opencv-detectmulti`
`minNeighbors` - Parameter specifying how many neighbors each candidate rectangle should have to retain it. This parameter will affect the quality of the detected faces. Higher value results in less detections but with higher quality.

`scaleFactor` - The face detector features are applied as differently sized "windows" in a cascading fashion to save time, `scaleFactor` sets how much we decrease the size of the window on each step. F.ex with 1.05 the size is decreased by 5%.

2.3 OpenCV CascadeClassifier source

is in `cascadedetect.cpp`, for example can be found from `https://github.com/opencv/opencv/blob/10ba6a93a6fee952fb7812b28989eb209d4f49a1/modules/objdetect/src/cascadedetect.cpp`

2.4 Other classifier files

`haarcascadeshaarcascade_eye_tree_eyeglasses.xml` was good at finding more eyes from the image

2.5 How long it took

1-2 hours

2.6 Homework source

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from scipy import ndimage as nd
from scipy.spatial import distance

print(cv2.__version__)

face_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascades/haarcascade_eye_tree_eyeglasses.xml')
img = cv2.imread('people.jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.01, 250)
for (x,y,w,h) in faces:
    cv2.rectangle(img,(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]
    eyes = eye_cascade.detectMultiScale(roi_gray, 1.001, 5)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)
cv2.imwrite('eyeglasses_detect.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```