Shot detection (by Omar Hiweish)

I decided to use the optical flow to detect continuous movement between the frames as it was discussed by Tapaswi et. al in the paper *Knock! Knock! Who is it" Probabilistic Person Identification in TV Series*.

Optical flow looks for the continuous movement of an object from one frame to another by comparing the points in the previous frame to the points in the current frame. If the points in the previous frame exist in the current frame, then the object has a continuous motion and we do not have a shot. If the points of the previous frame don't exist in the current frame, then we have a shot.

I used cv2's goodFeaturesToTrack function to find important features that can be tracked. For goodFeaturesToTrack, I used the following parameters to fine-tune the function:

```
maxCorners = 10000,   // This is maximum number of corners to detect

qualityLevel = 0.1,   // This controls the quality of the corners, 0.1 means
accept corners with 90% quality

minDistance = 10,     // This sets the maximum distance between the corners

blockSize = 5         // This is the block size of the neighborhood
```

I used the cv2's calcOpticalFlowPyrLK function to calculate the points of the current frame. I used the following parameters to fine-tune the function:

```
winSize  = (9,9),  // This is the window size to use when looking for
detecting motion

maxLevel = 5,      // This determines the levels of pyramids to do.

// The criteria has two options, TERM_CRITERIA_EPS is epsilon, and
TERM_CRITERIA_COUNT is the number of iterations

criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 4, 0.2)
```

To decide if a point exists in the next frame or not, I get the difference between the points in the old frame and the new frame. Then, I check if the percentage of points that have a difference of more than 4 pixels is more than 70% or not. If it is then I consider it a new shot.

For the shot performance, I calculate the score by adding a point for each detected shot. Then, I subtract 0.5 point for each false positive.

My results for running shot detection on the 3 clips:

```
clip 1
Correct hits:  1 / 1
False hits:  0
Score:  100.0 / 100
----------------
clip 2
Correct hits:  7 / 8
False hits:  1
```

```
Score:  81.25 / 100
----------------
clip 3
Correct hits:  17 / 26
False hits:  6
Score:  53.84615384615385 / 100
```

Detecting the logo and Face (by Ali Salem)

I decided to use Template matching using this link as a reference(https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html)

Template matching finds the location of the logo picture in a bigger image. It moves the ogo image over the bigger image and compares the logo picture and patch the given image under the logo image Then returns a grayscale image.

I used cv2's matchTemplate function to match the logo picture (given as a parameter in the main function) to detect the logo in the clip if it matches we draw a green rectangle around it, I used the following parameters inside the matchTemplate function:

Image: gray image

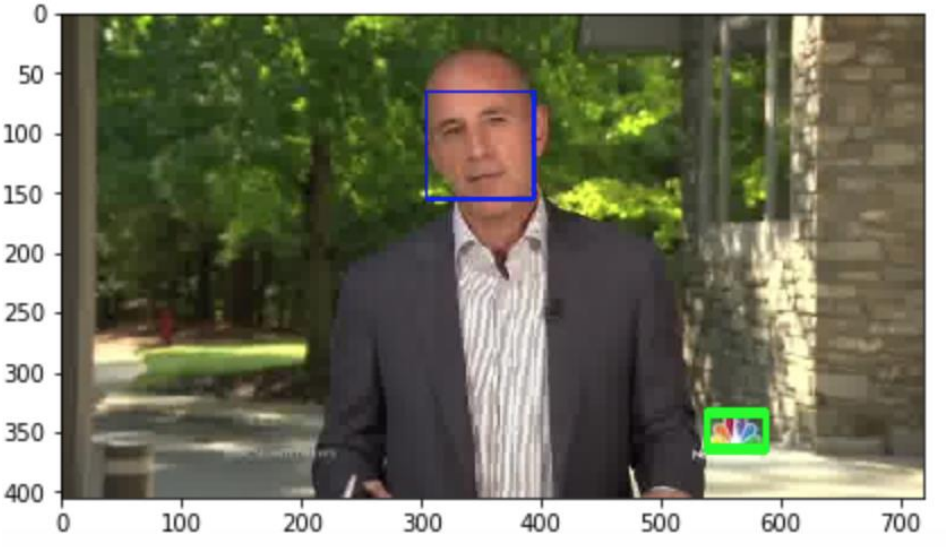
Template: logo picture

Method: specifying the comparison method

I decided to use face_recogntion library after import dlib. using this link as a reference (https://pypi.org/project/face_recognition/)

Face location finds the faces in the picture. So to detect the faces, it finds and manipulates facial features in the images. It uses CNN face detector. It returns an array listing the coordinates of found faces in the image (top, left , bottom,right).

The parameter for face_location function is image we are trying to detect faces in it.



Gender detection (by Omar Hiweish and Ali Salem)

For detecting genders in the video, we're using the face_recognition library which uses the dlib library. We use the face_recognition.face_locations to find the location of the faces in each frame. Then, we use face_recognition.face_landmarks to find face features such as left eye, right eye, etc. Here are some examples:

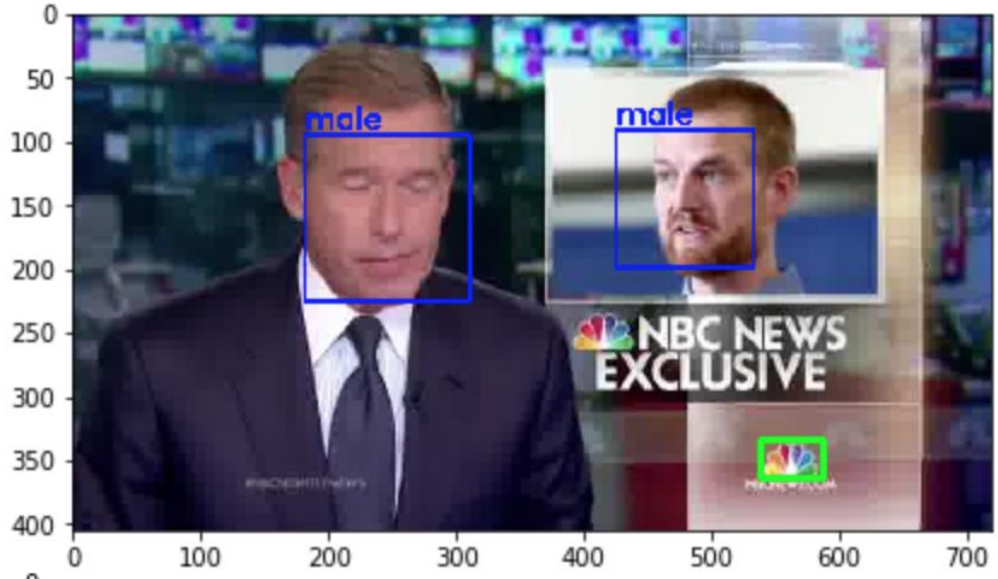
We use these features as test data for the next step which is predicting whether each face is male or female.

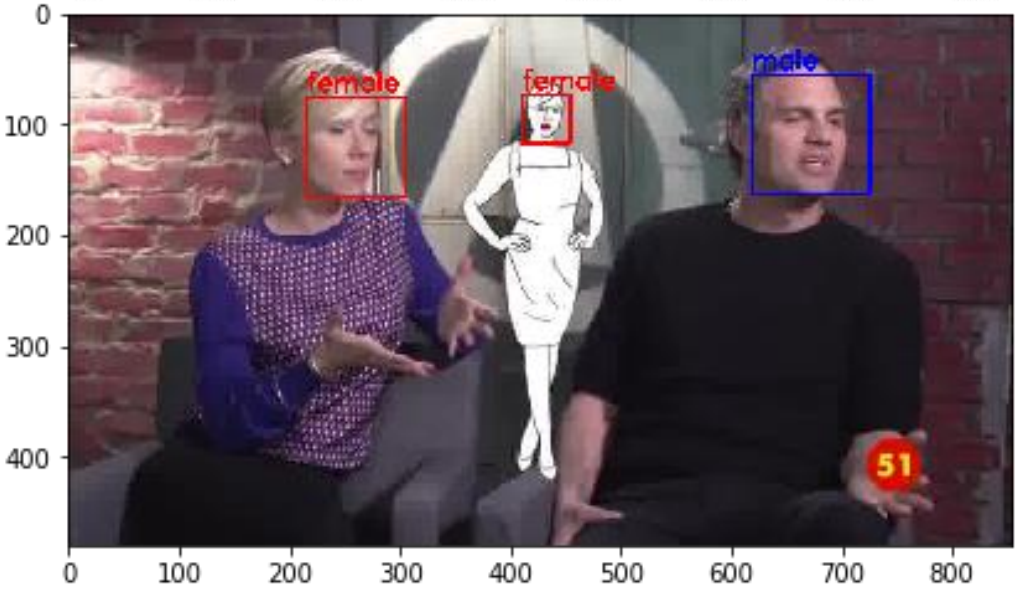
We used the Quadratic Discriminant Analysis classifier and trained it with the given mat files in the train_data folder.

After we trained the classifier, we ran it on our test data that we got from get_feat function to predict if each face in the clips is male or female.

Then, we used the list of gender prediction outputted by the classifier to add the gender above each face in the visualization part.

Here are examples of our gender detection:


male
male
NBC NEWS
EXCLUSIVE


female
female
male
51


female
male
male
female
female
the Voice
THE VOICE NAMES WINNER

Resources:

https://www.researchgate.net/publication/269114790_Video_Shot_Boundary_Detection_A_Review

https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

Makarand Tapaswi, Martin Baeuml and Rainer Stiefelhagen, "*Knock! Knock! Who is it*" *Probabilistic Person Identification in TV Series*, CVPR 2012, https://cvhci.anthropomatik.kit.edu/~mtapaswi/papers/CVPR2012.pdf

Gary B. Huang, Marwan Mattar, Honglak Lee, and Erik Learned-Miller. Learning to Align from Scratch. Advances in Neural Information Processing Systems (NIPS), 2012.

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_template_matching/py_template_matching.html

https://pypi.org/project/face_recognition/

https://machinelearningmastery.com/compare-machine-learning-algorithms-python-scikit-learn/

https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

https://www.pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jaw-dlib-opencv-python/

https://justinshenk.github.io/posts/2018/04/optical-flow/

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html

https://docs.opencv.org/2.4.13.7/modules/imgproc/doc/object_detection.html

https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769