Broadcast Video Feature Detection

Hirotaka Ishihara Yichun Zhang

December 2019

1 Introduction

In this project, we are going to perform analysis on videos, in particular, news broadcast. There are various tasks associated with the video analysis, such as shot detection, feature detection and recognition, and feature tracking between frames. Video processing has been a popular area in computer vision for a long time. In the project, we are going to employ novel ideas from the past few years to complete our analysis.

2 Shot Detection

Shot detection is crucial in video processing, it is a useful helper for other computer vision operations like face detection in consecutive frames[10]. I will perform three shot detection algorithms in 2.1 on the given video clips [5].

2.1 Method

The first two algorithms: Average Intensity Measurement 2.1.2 and Color Histogram Comparison 2.1.3, are very widely used in shot detection. They in general compares the color distribution of two images, which is robust to camera motion. The third method 2.1.4 is a relatively novel measurement, instead of collecting pixel color information, it focuses on object features.

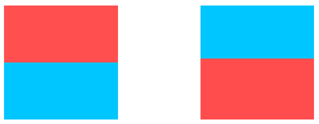
Figure 1: case where traditional color histogram doesn't work in shot detection

2.1.1 Adaptive Thresholding

Before describing the details of the methods, I would talk about out threshold policy. In this project, the threshold policy we choose is called Adaptive Thresholding [14]. There are two hyperparameters in this thresholding, W denotes the window size and by default is an odd integer. T_c is a constant. For each dissimilarity score at time slot t, the threshold is

$$m_t = max(\mu_{left} + T_c\sqrt{\sigma_{left}}, \mu_{right} + T_c\sqrt{\sigma_{right}})$$

$\mu_{left}, \sigma_{left}$ is calculated from the dissimilarity values within interval $[d - W/2, d)$, and the similar manner for $\mu_{right}, \sigma_{right}$.

2.1.2 Mean Intensity Measurement (MIM)

This method is described briefly in [13], which computes the average intensity of each image channel, and sum the difference over all channels with the values of the next frame.

2.1.3 Color Histogram Comparison (CHC)

The traditional color histogram described in [13] takes intensity values at gray-scale, and calcutes the entire image intensities into one histogram. While we think this may not be ro-

Figure 2: the red box moves between frames

bust, Figure 1 is the case. If we take color histogram of these two images, the difference would zero. Our method slightly changes the traditional one. We first take color histogram of all three channels. Instead of taking the entire image size as the input, we make a small patch with size 16 to gather color information.

2.1.4 Edge Change Ratio (ECR)

The edge change ratio [11] is a method focuses more on the feature information within the image. Figure 2 shows a case that a box moves between frames but within a certain boundary. ECR tracks how edge features move between frames. This needs pre-processing on two frames.

Implementation

- We first calculate edge features (our implement applies canny edge with $\sigma = 5$) of both images, denoting both the image I_t, I_{t+1}, and processed canny edges C_t, C_{t+1}. Here we used OpenCV Canny Edge Detection functons.[2]

- Then apply dilation on both edge feature D_t^-, D_{t+1}^-, then we calculate the inverse

$$D_t = 1 - D_t^-$$

$$D_{t+1} = 1 - D_{t+1}^-$$

- Take binary operations between C_t, D_{t+1} and C_{t+1}, Dt. A binary operation is defined as $binary(A, B) = A * B$ when A, B only have values zero and one.

$$EC_{out} = \frac{binary(C_t, D_{t+1})}{Ct}$$

$$EC_{in} = \frac{binary(C_{t+1}, D_t)}{Ct+1}$$

- The final edge change ratio is

$$ECR = max(EC_{in}, EC_{out})$$

2.2 Evaluation

[R:Recall, P:Precision, F1: F Score]
[C:Correct, M:Missed, F:False]

Clip	C	M	F	R	P	F1
#clip 1	1	0	2	1.00	0.33	0.50
#clip 2	8	0	2	1.00	0.80	0.89
#clip 3	4	2	9	0.67	0.40	0.50

Table 1: metrics of *Mean Intensity Measurement* over three clips

Clip	C	M	F	R	P	F1
#clip 1	1	0	2	1.00	0.33	0.50
#clip 2	8	0	7	1.00	0.53	0.70
#clip 3	4	2	8	0.67	0.33	0.44

Table 2: metrics of *Color Histogram Comparison* over three clips

Clip	C	M	F	R	P	F1
#clip 1	1	0	0	1.00	1.00	1.00
#clip 2	6	2	0	0.75	1.00	0.86
#clip 3	5	1	3	0.83	0.62	0.71

Table 3: metrics of *Edge Change Ratio* over three clips

The graph in Figure 3 4 5 shows the performance of three measurements respectively. The yellow vertical spans are the true video shot detections, where those thin lines indicate the hard cuts and wide spans indicate dissolve transitions. We also have the following tables 1 2 3 show the metrics of the three algorithm. The mathematical expressions for Recall, Precision and F measure (F1) [9] are

$$Recall = \frac{Correct}{Correct + Missed}$$

$$Precision = \frac{Correct}{Correct + False}$$

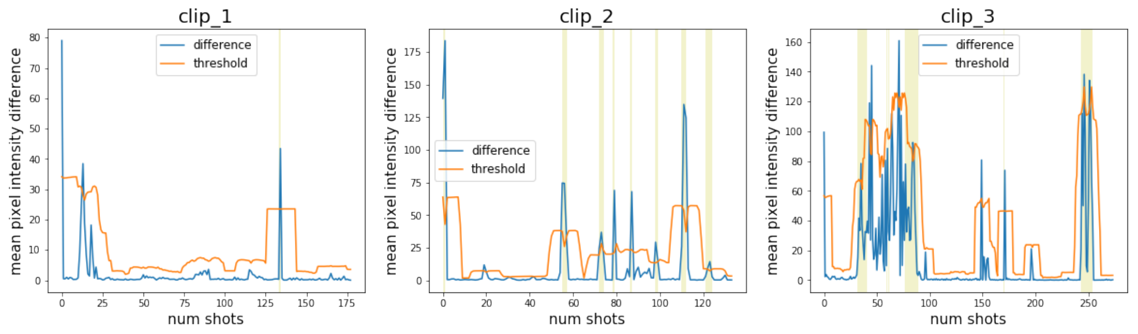

Figure 3: performance of Mean Intensity Measurement

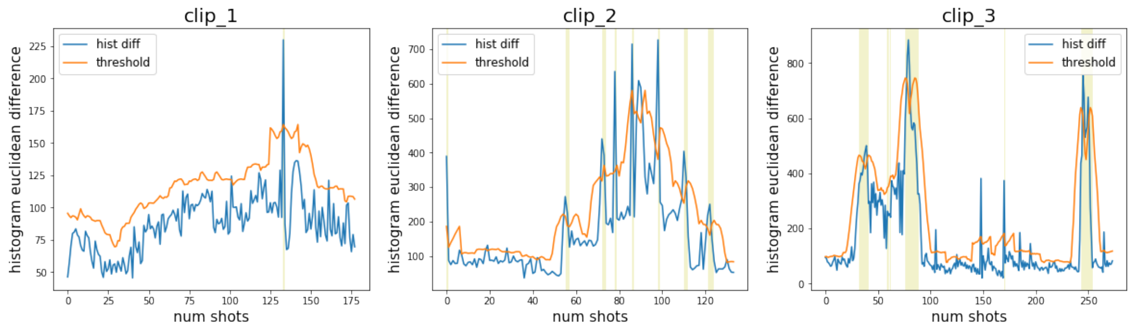

Figure 4: performance of Color Histogram Comparison

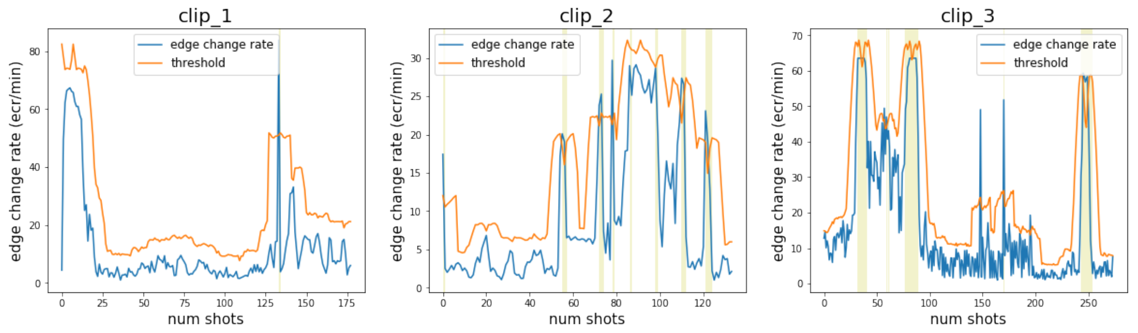

Figure 5: performance of Edge Change Ratio

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

Intensity based algorithms tend to have a higher Recall and low Precision, since the given video clips[5] generally have frequently intensity changes, the algorithms capture both correct shot boundaries and intensity noise. On the other hand, the feature based algorithm (ECR) generally out performs the other two algorithms. Observe that the worst performance of the three clips[5] is the clip 3. This is because a rapid changing comic scenes (Marvel comic), the ECR is not robust to video clips that have low sampling rate, since in this case the object has a larget movement gap which causes the False detections.

3 Feature Detection

Feature detection field has been rapidly improved as the Deep Learning concept came out. Some deep learning algorithms now have extraordinary performances. However, there are still some widely used techniques that are non deep learning based which also have good performances. For face detection, we are going to use Haar Cascade Classifier[12], and for logo detection, we adopt a method that learnt from

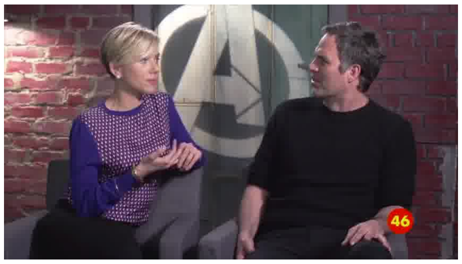

Figure 6: missed face detection

lecture materials.

3.1 Face Detection

We uses Haar Cascade Classifier[12] as our face detector. The algorithm first collects a large number of features, we need to figure out which features among all of these best determine a face. And this is done by using Adaboost, a collection of weak learner. All the features are stacked together with different amount, from less to more, grouped into the classifier. The algorithm is also fast at real time because of the Cascade concept, it immediately discard a sliding window in detecting when it fails at any position inside the classifier. Though in terms of the performance, CNN is better, but it is more efficient in training and has lighter model. So for simple task such as face detection, Haar Cascade has always been a good choice.

3.1.1 Face Detection Results

From Table 4, we can see that the Haar Cascade Classifier has better Precison than Recall. This is due to the Cascade property discussed previously. In contrast, the performance in the first two clips is better than the last one, especially for Recall measurement. This is because the angle that people facing at in clips 3 is not towards the camera like in Fig 6. So the Haar feature filters are not able to apply on these case variations. Even though in general people has a common front face feature, the style on each left or right may vary. Unlike convolutional neural networks, which is of high freedom to perform optimal kernels (as weights to train), Haar features are manually determined, thus with a tighter limitation.

[R:Recall, P:Precision, F1: F Score]
[C:Correct, M:Missed, F:False]

Clip	C	M	F	R	P	F1
#clip 1	253	45	10	0.85	0.96	0.90
#clip 2	373	73	4	0.84	0.99	0.91
#clip 3	329	165	12	0.67	0.96	0.78

Table 4: metrics of *Haar Cascade Classifier* over three clips

3.2 Logo Detection

The method for Logo detection we used is applying the knowledge we learnt from class: Scale Invariant Feature Transform (SIFT) [7], match keypoints with RANSAC [4], and apply homography, mapping the Logo to the location in the given video clips [5].

The advantage of these sequence of operations over machine learning is that it only requires the minimum amount of data (1 is enough), and fast to apply, without the training process. While the drawback is also inevitable, which the SIFT [7] algorithm is very sensitive to noise. In addition, the algorithm is not prepared for multiple appearances of the same Logo. As shown in Fig 7, the Logo of NBC news is half-transparent, so the detection result is not optimal. While in a relatively stable, less noise environment Fig 8, the detection is more accurate.

By using this method, though we are unable to improve Recall, the it is able to improve Precision. This is done by pruning bad mapping points, such us negative width and height, or points that are mapped out of the image, we abandon the boundary box in these case.

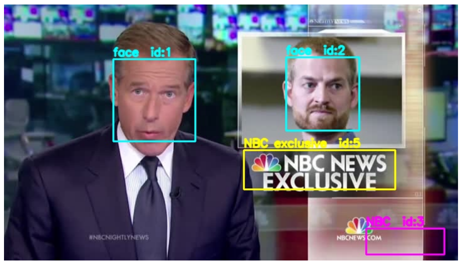

Figure 7: false Logo detection & positive face detection

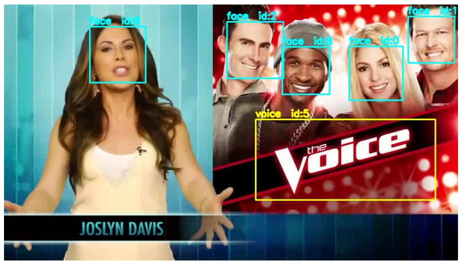

Figure 8: positive Logo detection & face detection

3.2.1 Logo Detection Evaluation

[R:Recall, P:Precision, F1: F Score]
[C:Correct, M:Missed, F:False]

Logo	C	M	F	R	P	F1
nbc news	92	76	10	0.55	0.90	0.68
nbc excl	96	10	9	0.91	0.91	0.91
the voice	57	0	0	1.0	1.0	1.0
clevver	77	36	0	0.68	1.0	0.81
flick	22	8	0	0.73	1.0	0.85

Table 5: metrics of *Logo Detection* over all Logos

The outcome varies between Logo types. Like the one showed in Fig 8, the type of Logo with solid background (i.e Logo has high opacity) tend to be robust in detection. In this method, the transparency of the Logo introduces noice in position detection.

4 Face Tracking

Perform face tracking on video or real time camera helps gain both spatial and temporal information. The most difficult part lies on the assignment problem. Most algorithm perform assignment on consecutive frames. We have come up a simple method that performs tracking even between non-continues frames and shots.

4.1 Algorithm

Algorithm 1 Greedy Detection Tracking

```
initiate a MAX_HEAP
for object in 1st frame do
    MAX_SCORE = 1
    SECOND_MAX = 0
    for object in 2nd frame do
        calculate similarity score
        if score > MAX_SCORE then
            SECOND_MAX = MAX_SCORE
            MAX_SCORE = score
        end if
    end for
    if RATIO > THRESHOLD then
        prune the pair
    else
        push pair to heap
    end if
end for
while MAX_HEAP not empty do
    obj1, obj2 = MAX_HEAP.pop
    if obj2 already belongs to a tracking then
        continue
    end if
    if neither obj1 nor obj2 belongs to a tracking then
        initiate new tracking with obj1 and obj2
    end if
    if obj1 belongs to a track then
        add obj2 to the same track
    end if
end while
LEFT_OVER = [object in 1st frame not get paired]
carry LEFT_OVER to the next round
```

The algorithm starts from the very first two consecutive frames, the process it similar to find the matching in SIFT matching[7].

4.2 Similarity Function for tracking

The essential parts of performing an accurate object tracking are detection and similarity measurement. In this subsection, we talk about how we set up the similarity function. In order to determine if two objects are very similar, we focus on both color distribution and feature information. To tackle color problem, we chose the color histogram difference measurement, over all three channels, this give a detailed color difference of two images. For feature comparison, we chose Histograms of Oriented Gradients (HOG).

To compute HOG, we first calculate both the magnitude and the direction of the gradients:

$$|\nabla I(x,y)| = \sqrt{I_x^2 + I_y^2}$$

$$\theta(x,y) = \arctan(I_y/I_x)$$

We take a 2×2 cell with each cell size of 8×8 pixel blocks sliding over the image, quantizing the directions into 9 bins with 20 degree increment. The gradient with larger magnitude votes into the bin. Then normalize 4 of such blocks together (2×2). So, in total, our similarity function S with Color Histogram difference CH_D and Histogram of Oriented Gradients difference HOG_D is:

$$S(I_1, I_2) = \frac{1}{HOG_D(I_1, I_2) + CH_D(I_1, I_2) + 1}$$

4.3 Performance

The algorithm works better in situation where number of people in the same scenes is small. Our algorithm can tackle discontinuous shot tracking (e.g identify and track people in 1st shot scene and 3rd scene). For frames that involves many people, it usually swaps the tacking suquence, like Fig 9 10

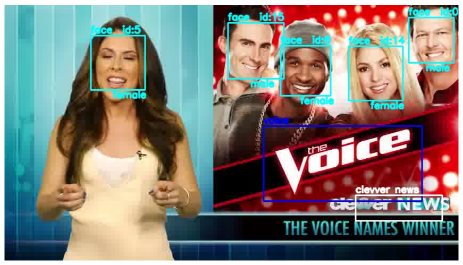

Figure 9: false Logo detection & positive face detection

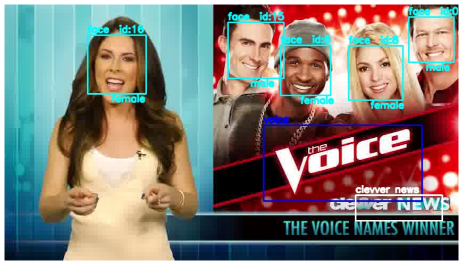

Figure 10: positive Logo detection & face detection

5 Gender Classification

Automated gender recognition is important in many applications such as biometric and human computer interaction. As a result, classifying gender from face images becomes an important research area. We tried two methods to perform this task.

5.1 ResNet18

Deeper neural networks turn out to be more powerful theoretically, but in practice they suffer from the degradation problem and are harder to train due to vanishing gradient. Residual blocks are proposed to solve this problem by using shortcut connections.

We originally trained a residual network–Resnet18. This is an end-to-end approach, that we feed the model with an image and would directly obtain the class of it. Data are separated into training data and validation data in order to see the performance. Our neural net was trained with an Adam optimizer [6] and cross entropy loss. To fasten the training process, the model was trained with a minibatch of size 32. However, after 25 epochs, the training accuracy approached 100%, but validation accuracy is only around 80%, even after adding one dropout layer and applying weight decay. This implies that our model faces the problem of overfitting.

We thought this problem can be alleviated by employing more regularization techniques. However, training the neural net took a large amount of time. In my experiment, training 30 epochs requires more than 2 hours. Due to the time concerns, we didn't do lots of experiments

on it.
For further improvement, experiments such like cropping part of the image before feeding it to the neural net are considered. Also, representing the image in lower dimensional space may help since it decreases the number of parameters in the model.

5.2 Histogram of Oriented Gradient Feature

Another method we tried is simply extracting features from images and then feeding the features into a classifier. In real world, people may find that outline of faces contributes a lot to gender recognition from facial image. In order to represent the edge information, we choose the Histogram of Oriented Gradient[3] as the descriptor.

According to the paper [3], classic HOG features are obtained by normalizing the image, computing the gradient of each pixel, creating a histogram of edge directions and doing block normalization. One of the main contributions of the paper is that they use dense grid of cells, allowing overlapping both horizontally and vertically when sliding the windows. Also, for better accuracy, they apply local contrast normalization to each block before sending them to the classifier.
In my implementation, each individual histogram has 20 signed bins, with an interval length of 18.
Since people could tell the gender of a person from a black-white face image, I transformed the image to gray scale before extracting the HOG features. The only preprocessing we perform is image resizing. Observe that the resolution of image can affect the extracted edge feature. Images with low resolution contain mainly the information of position and coarse edges while large scale images include more of fine edges. Once a face is detected, it is cropped and resized to four scales—$16 \times 16, 32 \times 32, 64 \times 64, 128 \times 128$. Then the feature is extracted using a window of size $2 \times 2, 4 \times 4, 8 \times 8, 16 \times 16$ respectively, either with or without overlapping. After obtaining the input data, we fit a svm.SVC classifier to it and perform classification. [8].

5.2.1 Result

After comparing the accuracy scores on test data, I chose the classifier trained by 64×64 images with overlapped windows. Most faces in first two clips are classified correctly. Good results are in Fig 11, Fig 12. There are a few

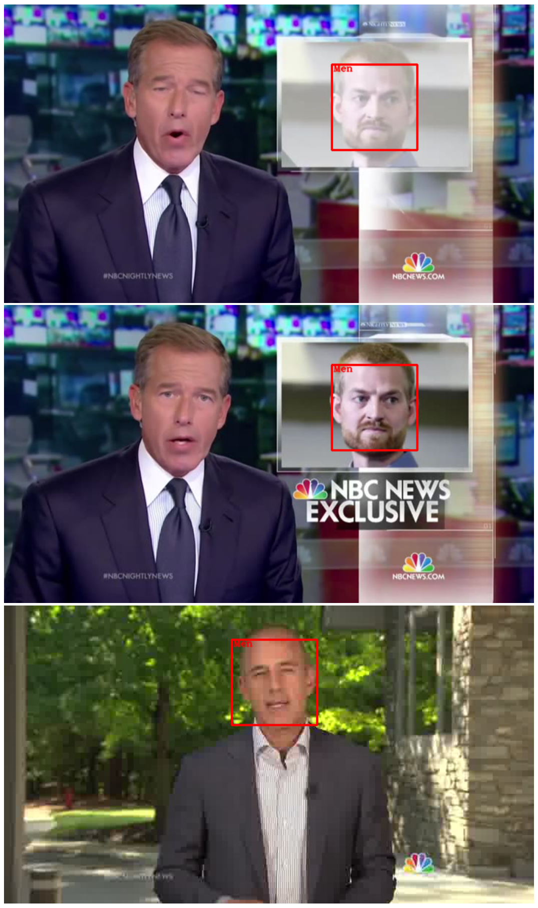

Figure 11: Correct classifications from clip1.

failures shown in Fig13, Fig 14 The top example in 13 could be caused by the change of il-

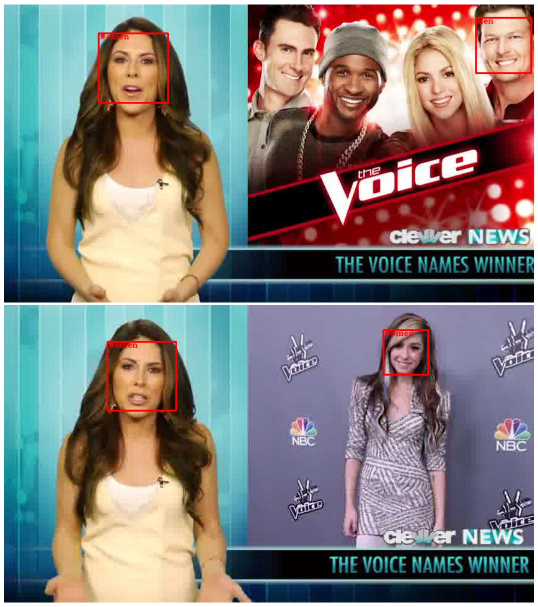

Figure 12: Correct classifications from clip2.

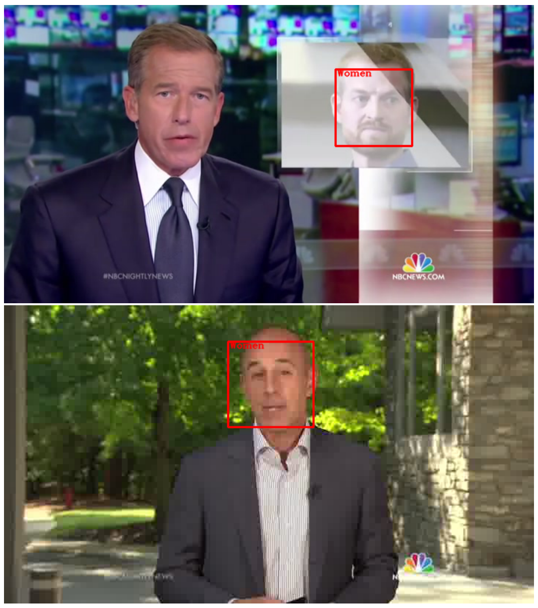

Figure 13: Bad results from clip1.

lumination within the face. This may give us a wrong edge which would influence the result. The potential reason for the other two failures could be the vague outlines.
But when applied the trained classifier to clip3, we obtained low precision with all classifiers. I found the outlines of faces in clip3 are not as clear as those in clip1 and clip2. This could be one possible reason for the failure, since the classifier is trained on edge features. Another possible reason is that faces in the training data set are mostly front faces, but in clip3, the man in the last frame bows his head. In order to improve the generalization of the classifier, we could include more images with various head poses in the training set.

5.2.2 Future Work

- As previously mentioned, the classifier may fail when faces are not perfectly front or edges are not clear. It could also fail when there exists a partial mask in the face. One improvement I could make is to gather training images with different head poses and augment the input by smoothing the faces or applying masks on randomly chosen areas. Rotating or translating the training data may also help.

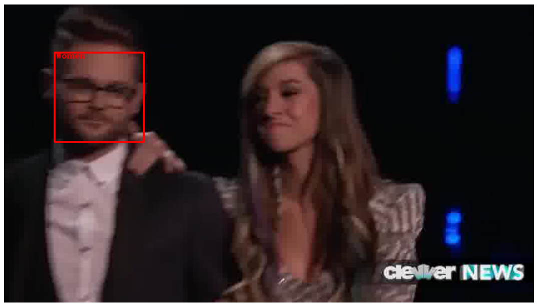

Figure 14: Bad result from clip2.

- The method has not yet been tested on other datasets. So, future work includes testing it on larger test sets.

- Alexandre and Lu (2010) [1] shows that decision fusion improves the result significantly. Instead of training a single classifier, this paper proposed to train various classifiers using different image scales or different features, predict the gender separately and make final decision via majority vote. I think above approach can be included for the future improvement since given a specific image, some features may be easy to obtain while others are not. We're interested of extracting geometric face features such as the relative position of eyes, nose and lip, as well as texture features shown in the paper.

- Feature extraction process consumes a lot of time. This could be shorten by code optimization

6 Conclusion

In this project, we found that the traditional SVM classifier with facial feature performs well when the face edges are clear but with a large possibility to fail when the illumination varies. Also, the resolution of the face image would affect the classification results.

7 Contribution

Hirotaka Ishihara

- 3 shot detection
- face detection
- logo detection
- face tracking and similarity function
- all code above

Yichun Zhang

- gender classification

References

[1] Luıs A Alexandre. "Gender recognition: A multiscale decision fusion approach". In: *Pattern recognition letters* 31.11 (2010), pp. 1422–1427.

[2] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[3] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: 2005.

[4] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: http://doi.acm.org/10.1145/358669.358692.

[5] Gary B. Huang et al. "Learning to Align from Scratch". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 764–772. URL: http://dl.acm.org/citation.cfm?id=2999134.2999220.

[6] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[7] David G Lowe. "Object Recognition from Local Scale-Invariant Features". In: ().

[8] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] C. J. Van Rijsbergen. *Information Retrieval*. 2nd. Newton, MA, USA: Butterworth-Heinemann, 1979. ISBN: 0408709294.

[10] Makarand Tapaswi, Martin Bauml, and Rainer Stiefelhagen. "Knock! Knock! Who is it? probabilistic person identification in TV-series". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* i (2012), pp. 2658–2665. ISSN: 10636919. DOI: 10 . 1109 / CVPR . 2012.6247986.

[11] Aidan Totterdell. "An Algorithm for Detecting and Classifying Scene Breaks in MPEG Video Bit Streams “”". In: *Dublin City University* September (1998). DOI: 10.1.1.74.5363.

[12] P. Viola and M. Jones. "Haar-like". In: *Cvpr* 1 (2001), pp. I-511-I–518. ISSN: 1063-6919. DOI: 10 . 1109 / CVPR . 2001 . 990517. URL: http : / / ieeexplore . ieee . org / document/990517/.

[13] Y. Yusoff, W. Christmas, and J. Kittler. "A study on automatic shot change detection". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 1425 (1998), pp. 177–189. ISSN: 16113349. DOI: 10 . 1007 / 3 - 540 - 64594 - 2{_}94.

[14] Y Yusoff, W Christmas, and J Kittler. "Video Shot Cut Detection Using Adaptive Thresholding". In: January 2000 (2014).