DIMPLER DETECTION USING FACIAL LANDMAKRS IN VIDEOS

An Undergraduate Research Scholars Thesis

by

SHUAIFANG WANG

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Anxiao Jiang

May 2020

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

Dimpler Detection Using Facial Landmarks in Videos

Shuaifang Wang
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Anxiao Jiang
Department of Computer Science and Engineering
Texas A&M University

Dimpler is one of the body languages that contributes to the emotion contempt when the action appears unilaterally, and to boredom. It is one of the subtle expressions that people did in everyday life. Although the universal seven microexpressions are clear signs of concealed emotions, subtle expressions such as dimple probably occur much more frequently than universal expressions. The dimpler muscle pulls the lip corners to the side and creates a dimple in the cheek. This study developed a dimpler detection model using 2D facial landmarks. 3 videos of totally 6 minutes were recorded while each video involved dimple and non-dimple expressions. Cheek and lip landmark were detected from each frame of the video using the Face-Alignment facial landmark detector. Features such as horizontal lip distance, vertical lip distance and lip ratio served as inputs to a linear Support Vector Machine (SVM) model. The SVM approach achieved a performance of accuracy 82.37%, sensitivity 86.58% and specificity of 84.29%.The results suggest that horizontal lip distance, vertical lip distance and lip ratio are useful features for the detection of dimpler in videos.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Anxiao Jiang, for his support of my research and dedication. It is my honor to study under Dr. Jiang and do research with him. Additionally, I want to thank other people in our great team for their assistances.

Thanks also to my friends within the Department of Computer Science and Engineering for their help and support.

LIST OF FIGURES

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

Dimpler refers to lip corners tightened and pulled inwards as shown from figure 1. It is action 14 in the Facial Action Coding System (FACS).[1] In this paper, we detect dimpler in the following way. For each image frame in the video, we first obtain the facial landmarks for the face in the frame. (There are totally 68 facial landmarks as shown in Figure 1. The method to extract facial landmarks from a video is shown in the Appendix.) We then define three features based on the facial landmarks: horizontal lip distance, vertical lip distance as well as lip ratio.

Figure 1: An illustration of dimple.

Horizontal Lip Distance

Consider the 68 facial landmarks as illustrated in Figure 1. For i = 1,2…68, let(x_i, y_i) denote the coordinates of the ith facial landmark in the image. For facial landmarks corresponding to lips, we define the *horizontal lip distance* as the below formula (x_l for the left most point and x_r for the right most point).

$$D_h = ||l_{1l} - l_{1r}|| = \sqrt{(x_{64} - x_{48})^2 + (y_{64} - y_{48})^2}$$

(point 64 is the right most point on the lip and point 48 is the left most point on the lip).

Vertical Lip Distance

We define the *vertical lip distance* D_v as

$$D_v = ||p_{1l} - p_{1r}|| = \sqrt{(x_{57} - x_{50})^2 + (y_{57} - y_{50})^2}$$

Lip Ratio

We define the lip ratio as the ratio of horizontal lip distance and vertical lip distance.

$$R = \frac{horizontal\ lip\ distance}{vertical\ lip\ distance}$$

Figure 2: An illustration of the 68 facial landmarks for a human face.

We then train the model that detect dimple for a frame in the video. For the SVM model, we let M_i be their input, and let l_i be their target output. To simplify the training process for the model, we impose the constraint that the labels l_1, l_2, … have the same value. A classification accuracy of 82.37%, sensitivity of 86.58% and specificity of 84.29% were achieved.

There has been a number of related works on detecting facial actions in the Facial Action Coding System. The action units and related references are given in Table 1.

The lip landmarks are detected for every frame in a video. The degree of dimpling differs every time, and the time of dimpling lasted also varies every time a subject dimples. Dimpler involves lip corners tighten and pull inwards. During pulling of lip corners, the upper and lower lips come close to each other, while the distance between the left most and right most corner of the lips get larger. It was observed that other portion of the lips do not move much so the position of those points don't change significantly during dimpling. Therefore, it was decided to use D_h and D_v as features along with ratio R to detect dimpler.

Table 1: Facial Action Units

Action unit (AU)	AU name
AU1	Inner Brow Raiser
AU2	Outer Brow Raiser
AU4	Brow Lowerer
AU5	Upper lid Raiser
AU6	Cheek Raiser
AU10	Upper Lid Raiser
AU13	Cheek Puffer
AU14	Dimpler
AU15	Lip Corner Depressor
AU16	Lower Lip Depressor
AU17	Chin Raiser
AU18	Lip Puckerer
AU20	Lip Stretcher

CHAPTER II

METHODS

Classification

During the data collection, it was noted that whenever the subject dimples, its presence is seen in at least three frames of the video. Also, dimpler occurs while the subject had been dimpling for a certain amount of time. [3] had experimentally found that for each frame, using a 13-dimensional feature obtained by concatenating *O*s of its ±6 neighboring frames have notable impact in dimpler detection. This concept has been adopted in this study. Here for each frame, a feature was computed, considering ±6 neighboring frames for each of the three features. In this study, support vector machines(SVM) algorithm were used to implement the dimple detection and their performance were evaluated. Similarly, ground-truth dimples with "1" labels were considered as positive samples and frames with "no dimple" label whose ±6 neighboring frames also had no dimple "0" labels were considered as negative samples during training. [4]&[5] It was experimentally observed that the selection of negative samples per the above method helps the classifier perform better while testing.

Classification Metrics

In this study, we have used metrics such as accuracy, specificity and sensitivity to evaluate the classification models. We define the terms used in estimating the metrics below:

- True Positives(TP) : The total number of accurate predictions that were "positive" or "1". In our study, this is the total number of samples correctly predicted as dimples.

- False Positives(FP): The total number of inaccurate predictions that were "positive" or "1". In our study, this is the total number of samples incorrectly predicted as dimples.
- True Negatives (TN): The total number of accurate predictions that were "negative" or "0". In our study, this is the total number of samples correctly predicted as "not dimple".
- False Positives(FN): The total number of inaccurate predictions that were "negative" or "0". In our study, this is the total number of samples incorrectly predicted as "not dimple".

Now we are able to define the classification metrics such as accuracy, sensitivity and specificity and present their mathematical expressions.

- Accuracy: It is a measure of all the instances that are correctly classified. It is the ratio of the number of correct classifications over the total number of predictions.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

- Sensitivity: It is the proportion of instances that were predicted as positive with respect to all the instances that are actually positive.

$$Sensitivity = \frac{TP}{TP+FN}$$

- Specificity: It is a measure of the fraction of all instances that are correctly classified. It is the ratio of the number of correct negative classifications to the total number of negative instances.

$$Specificity = \frac{TN}{TN+FP}$$

	Label	feature	xdis	ydis
0	0	2.7317	112	41
1	0	2.4583	118	48
2	0	2.36	118	50
3	0	2.38	119	50
4	0	2.3333	119	51
5	0	2.3529	120	51
6	0	2.3077	120	52
7	1	2.5192	131	52
8	1	2.4727	136	55
9	1	2.5818	142	55
10	1	2.5818	142	55
11	1	2.4483	142	58
12	1	2.4833	149	60
13	1	2.4833	149	60
14	1	2.4833	149	60
15	1	2.459	150	61
16	1	2.4918	152	61
17	1	2.4918	152	61
18	1	2.5082	153	61
19	1	2.5082	153	61
20	1	2.5082	153	61
21	1	2.5082	153	61
22	1	2.5082	153	61
23	1	2.5082	153	61
24	1	2.5082	153	61
25	1	2.3906	153	64
26	0	2.3182	153	66
27	0	2.3182	153	66
28	0	2.3182	153	66
29	0	2.3182	153	66
30	0	2.3182	153	66
31	0	2.3182	153	66
32	0	2.3182	153	66
33	0	2.3182	153	66
34	0	2.3182	153	66
35	0	2.3182	153	66
36	0	2.3182	153	66
37	0	2.3182	153	66
38	0	2.3182	153	66
39	0	2.3182	153	66
40	0	2.3182	153	66
41	0	2.3182	153	66
42	0	2.3182	153	66
43	0	2.3182	153	66
44	0	2.3182	153	66
45	1	2.3333	154	66
46	1	2.3333	154	66
47	1	2.3333	154	66
48	1	2.3333	154	66
49	1	2.3333	154	66
50	1	2.3333	154	66
51	1	2.3333	154	66
52	1	2.3333	154	66
53	1	2.3333	154	66
54	1	2.3333	154	66
55	1	2.3333	154	66
56	1	2.3333	154	66

Figure 3: Calculated three features for every frame in the training set. (xdis stands for horizontal lip distance and y stands for vertical lip distance).

CHAPTER III

RESULTS

The training dataset contains 3367 frames taken from two videos. The testing data-set contains 2256 frames taken from one video. Facial landmarks for the lip were estimated using face-alignment model.[2] A threshold of 0.80 gave the best classification metrics for the SVM model. The classification metrics is shown in Table 2. The output video was successfully generated as expected, where each frame contains text overlapped on the video to indicate the predicted results.(“1” stands for dimple and “0” stands for non-dimple)

Table 2: Classification Metrics

Accuracy	82.37%
Sensitivity	86.58%
Specificity	84.29%

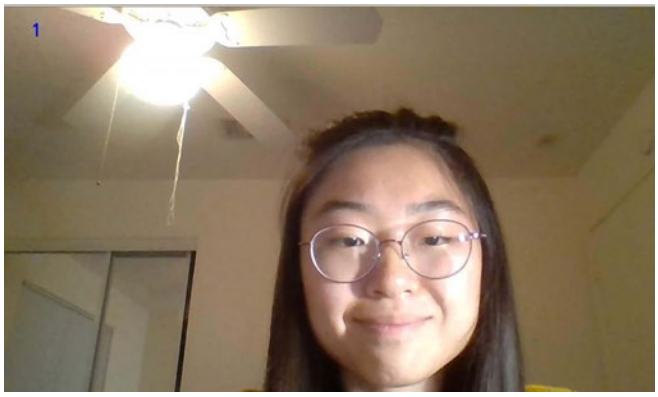

Figure 4: frame where the model predict dimpler accurately.

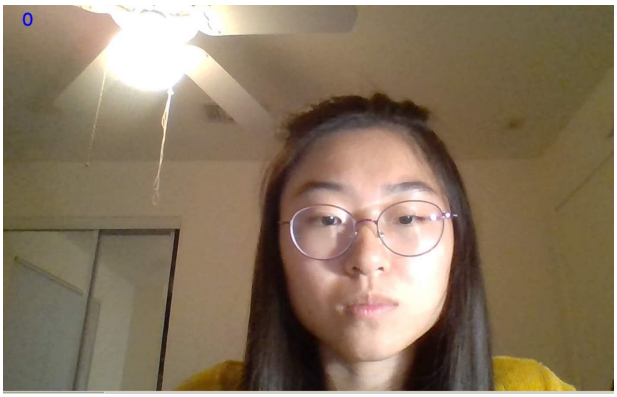

Figure 5: Frame where the model predict non-dimpler accurately.

Figure 6: Frame where the model failed to predict dimple accurately.

The model failed to predict dimple when there is subtle widening of horizontal lip distance and no significant narrowing of vertical lip distance. There were also fluctuations in predicting several dimples which may be due to the poor resolution of the images. The example of the situation where the model did not perform well is shown in figure 7.

CHAPTER IV

CONCLUSION

A synchronous dimpler detection algorithm has been developed in this study. The dimpler detection model using 2D facial landmark detectors have proven to be competent in terms of performance on unseen data. This algorithm runs in real-time and the model proposed in this study can be improved by using 3D landmarks and replace shallow network (SVM) with Deep Neural Network (DNN).[8] The biggest limitation in this study is data collection involves a single subject only. A future progress will be collecting data from multiple subjects and using deep neural network for detection. Another future work may include further analysis of features of the expression. Additionally, the study could be improved by detecting multiple expressions at the same time, with a likelihood of the percentage of each expression overlapped on top of the video.[11] These findings will be able to help people read the emotions of the subject while watching videos. Therefore, if users are curious about the emotion of those people giving TED talks, they could apply this algorithm on those videos and read out their minds.

REFERENCES

[1] Paul Ekman and Wallace V Friesen. Manual for the facial action coding system. Consulting Psychologists Press, 1978.

[2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In International Conference on Computer Vision, 2017.

[3] Tereza Soukupova and Jan Cech. Eye blink detection using facial landmarks. In 21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia, 2016.

[4] Cheng-Hao Tu, Chih-Yuan Yang, and Jane Yung-jen Hsu. Idennet: Identity-aware facial action unit detection. In 2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019), pages 1–8. IEEE, 2019.

[5] Kaili Zhao, Wen-Sheng Chu, and Honggang Zhang. Deep region and multi-label learning for facial action unit detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3391–3399, 2016.

[6] Yong Li, Jiabei Zeng, Shiguang Shan, and Xilin Chen. Self-supervised representation learning from videos for facial action unit detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 10924–10933, 2019.

[7] Zhiwen Shao, Jianfei Cai, Tat-Jen Cham, Xuequan Lu and Lizhuang Ma. Weakly-supervised unconstrained action unit detection via feature disentanglement. arXiv preprint arXiv:1903.10143, 2019.

[8] Bilal Taha, Munawar Hayat, Stefano Berretti, and Naoufel Werghi. Learned 3d shape representations using fused geometrically augmented images: Application to facial expression and action unit detection. arXiv preprint arXiv:1904.04297, 2019.

[9] Le Yang, Itir Onal Ertugrul, Jeffrey F Cohn, Zakia Hammal, Dongmei Jiang, and Hichem Sahli. Facs3d-net: 3d convolution based spatiotemporal representation for action unit detection.

[10] Ying-li Tian, Takeo Kanade, and Jeffrey F Cohn. Evaluation of gabor-wavelet-based facial action unit recognition image sequences of increasing complexity. In Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition, pages 229–234. IEEE, 2002.

[11] Stefanos Eleftheriadis, Ognjen Rudovic, and Maja Pantic. Multi-conditional latent variable model for joint facial action unit detection. In Proceedings of the IEEE International Conference on Computer Vision, pages 3792–3800, 2015.

[12] Chu Wang, Jiabei Zeng, Shiguang Shan, and Xilin Chen. Multi-task learning of emotion recognition and facial action unit detection with adaptively weights sharing network. In 2019 IEEE International Conference on Image Processing (ICIP), pages 56–60. IEEE, 2019.

APPENDIX

Detect_dimpler.py Part 1

```
preds = []
xlist = []
ylist = []
xdistance = 0
ydistance = 0
test_xdistance = 0
test_ydistance = 0
ratio = 0

#pred_types =  {}
#cross_validation
#f1/f-measure. Evaluation. sklearn has function that you can call

def get_feature(path):
  #try:
  input_img = io.imread(path) #aflw-test
  #except FileNotFoundError:
      #input_img = io.imread(path)

  preds = fa.get_landmarks(input_img)[-1]

  # 2D-Plot
  plot_style = dict(marker='o',
                    markersize=4,
                    linestyle='-',
                    lw=2)

  pred_type = collections.namedtuple('prediction_type', ['slice', 'color'])
  pred_types = {'face': pred_type(slice(0, 17), (0.682, 0.780, 0.909, 0.5)),
                'eyebrow1': pred_type(slice(17, 22), (1.0, 0.498, 0.055, 0.4)),
                'eyebrow2': pred_type(slice(22, 27), (1.0, 0.498, 0.055, 0.4)),
                'nose': pred_type(slice(27, 31), (0.345, 0.239, 0.443, 0.4)),
                'nostril': pred_type(slice(31, 36), (0.345, 0.239, 0.443, 0.4)),
                'eye1': pred_type(slice(36, 42), (0.596, 0.875, 0.541, 0.3)),
                'eye2': pred_type(slice(42, 48), (0.596, 0.875, 0.541, 0.3)),
                'lips': pred_type(slice(48, 60), (0.596, 0.875, 0.541, 0.3)),
                'teeth': pred_type(slice(60, 68), (0.596, 0.875, 0.541, 0.4))
                }

  fig = plt.figure(figsize=plt.figaspect(.5))
  ax = fig.add_subplot(1, 2, 1)
  ax.imshow(input_img)
```

Detect_dimpler.py Part 2

```
for pred_type in pred_types.values():
    #print preds[pred_type.slice,0];
    #print preds[pred_type.slice,1];
    ax.plot(preds[pred_type.slice, 0],
            preds[pred_type.slice, 1],
            color=pred_type.color, **plot_style)

ax.axis('off')

# 3D-Plot
ax = fig.add_subplot(1, 2, 2, projection='3d')
surf = ax.scatter(preds[:, 0] * 1.2,
                  preds[:, 1],
                  preds[:, 2],
                  c='cyan',
                  alpha=1.0,
                  edgecolor='b')

for pred_type in pred_types.values():
    ax.plot3D(preds[pred_type.slice, 0] * 1.2,
              preds[pred_type.slice, 1],
              preds[pred_type.slice, 2], color='blue')
'''
ax.view_init(elev=90., azim=90.)
ax.set_xlim(ax.get_xlim()[::-1])
plt.show()
'''
a=preds[pred_types['lips'].slice]
#print a
aa=[]
for point in a:
  info = ''
  for axis in point:
    info = info + str(axis) +','
  info = info[:-1]
  aa.append(info)
```

Detect_dimpler.py Part 3

```
  for i in aa:
   temp = i.split(',')
   xlist.append(float(temp[0]))
   ylist.append(float(temp[1]))
  #xlist.append( data[:,0]);
  #ylist.append data[:,1];
  xdistance = max(xlist) - min(xlist)
  ydistance = max(ylist) - min(ylist)
  ratio = xdistance / ydistance

  return xdistance, ydistance, ratio
'''
  for item in preds[pred_types['lips'].slice]:
    lips_list.append(item)  #3D
    lips_list.append(item[:2])  #2D

  #print lips_list
'''
pic_fea_list=[]


root='/home/wangshuaifang/face-alignment/test/assets/frame/pic'

xdistance_all = []
ydistance_all = []
ratio_all = []
Label = []
file1 = open("/home/wangshuaifang/face-alignment/test/assets/label.txt","r")
a=file1.readlines()[0]
file1.close()
a=a.split(',')
for i in a:
  Label.append(int(i))
```

Detect_dimpler.py Part 4

```
for i in range(1,3366):
  path = root+str(i)+'.jpg'
  print('picture ', i)
  xdistance, ydistance, ratio = get_feature(path)
  xdistance_all.append(xdistance)
  ydistance_all.append(ydistance)
  ratio_all.append(ratio)
  pic_fea_list.append([xdistance, ydistance, ratio])

df = pd.DataFrame({'xdis':xdistance_all, 'ydis':ydistance_all, 'feature': ratio_all,'Label':Label})
df.to_csv('data_1.csv', mode='w', index=True)

  #print pic_fea_list

my_csv = pd.read_csv('data_1.csv')
feature_all = my_csv.feature
print feature_all

# train and test
print("train and test")
from sklearn import svm
clf = svm.SVC()
pic_fea_list_test =[]
feature_all = np.array(feature_all)
feature_all= feature_all.reshape(-1,1)
#arg2 = np.reshape(Label, (-1, 1))
clf.fit(feature_all, Label)

root_test='/home/wangshuaifang/face-alignment/test/assets/testfile/pic'
for i in range(1,1000):
  path_test = root_test+str(i)+'.jpg'
  print('test picture ', i)
  test_xdistance, test_ydistance, fea_test = get_feature(path_test)
  pic_fea_list_test.append(fea_test)# get the feature for each frame
  print ('feature for picture', i, 'is ', fea_test)
  #feature_test = np.array(pic_fea_list_test)

arg1= np.reshape(feature_all, (-1, 1))
arg2 = np.reshape(Label, (-1, 1))

feature_all = np.array(feature_all)
feature_all = feature_all.reshape(-1,1)
```

Detect_dimpler.py Part 5

```
file1 = open("/home/wangshuaifang/face-alignment/test/assets/truelabel.txt","r")
a=file1.readlines()[0]
file1.close()
a=a.split(',')
for i in a:
  true.append(int(i))

print('predict', predict)
print accuracy_score(true, predict)


cap = cv2.VideoCapture(r"/home/wangshuaifang/face-alignment/examples/myvideo.avi")
length = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
currentFrame = 0
image = ""
# videowriter object
codc = int(cap.get(cv2.CAP_PROP_FOURCC))
fps = int(cap.get(cv2.CAP_PROP_FPS))
w = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 15, (300,200))

predictions =  [classes_names[i] for i in clf.predict(get_feature(image))]
arr = np.array([])

j = 0
for prediction in predictions:
    pt = (0, 3 * image.shape[0] // 4)
    cv2.putText(image, prediction, pt ,cv2.FONT_HERSHEY_SIMPLEX, 2, [0, 255,
    0], 2)
    np.append(arr,cv2.imwrite(str(j)+'.jpg',image))

out.write(image)
currentFrame += 1
cap.release()
out.release()
cv2.destroyAllWindows()
```