

2019 Spring Introduction to Computer Networks

Final Project Report - Team 28

Face Dance Machine Web Application

Author: Yu-Shan Huang (黃郁珊) Szu-Yu Mo (莫絲羽) Chun-Hung Yu (游雋閔)
Student ID: b04901091 b04901164 b04901089
Email: qazwsx860809@gmail.com evamo.tw1@gmail.com davidyu52401@gmail.com
Department: Electrical Engineering Department, National Taiwan University
Advisor: Professor Wan-jiun Liao
Github repo: <https://github.com/evamo0508/Face-Dance-Machine-python-webcam-flask>
Demo website: <http://stark-badlands-83896.herokuapp.com>
Project date: June 2019

Content

- Abstract
- Game Workflow & How To Play
- Achievement
- Implementation
- Challenge
- Future Work
- Reference

Abstract

We deployed a Heroku website for users to play with face-dance-machine, meaning that users will have to make similar facial expressions with the cartoon faces shown on screen to win. In this project, we utilized several networking knowledge such as video streaming, sound streaming, rate control, network security, python flask web application, javascript and html. On the application side, we applied facial landmarks detection to the video captured, and came up with an algorithm to recognize facial expressions through those landmarks.

Game Workflow

1. The web client sends video stream data (from the user's webcam) to a flask server using socketio.
2. The server detects the facial expressions and sound in the video, and overlays task images on frames.
3. The client receives the processed video stream and re-displays the results in a different frame.

How To Play

1. Open any browser (Chrome preferred) and access the demo website mentioned down below.
2. Allow webcam and microphone access during the first time visit. (might need to refresh the page after allowing)
3. Look at the video on the right. Try to make your face as similar as any cartoon faces in the video. The cartoon icons would disappear if the similarity is high enough.
4. If the task image is not a cartoon icon (i.e. a real person's face...), you should shout at the microphone. The face would explode only if the sound is loud enough.
5. The game does not end, nor does it record the score. This is just for fun, so please have fun!



Fig. 1 Game playing screenshot

Achievement

Our achievements can be easily illustrated by accessing the demo website. However, we can break down our results to several parts, including:

1. basic streaming features
 - a. video streaming
 - b. sound streaming
2. advanced network features
 - a. deploying to Heroku
 - b. rate control
 - c. network security
3. machine learning application
 - a. facial expression recognition

Some screenshots demonstrating the above features would be shown in the following paragraphs.

I. Basic Streaming Features

A. Video Streaming

We detect the facial expression at the server, so we need to send the video to the server after compression. And then the client will request the processed video from the server. Both client and server send and receive the video at the same time.

B. Sound Streaming

The audio source comes from microphone. We want to design a real time sound streaming and take the volume as an input of our game. Sound streaming operates almost the same as video streaming. The minor difference is that we need to create an audioContext to get the volume of audio, which is exactly what we want.

II. Advanced Network Features

A. Deploying to Heroku

We have deployed our game to <http://stark-badlands-83896.herokuapp.com> and users can access the url no matter which network domain he or she is in. Typically, one should buy a domain to deploy a website, otherwise you can only provide your service to people in the same local area network. Heroku is a company that offers you a free domain as well as some memory space on their server. Thus, we can own a domain by signing up a Heroku account.

B. Rate Control

It's a trade-off between video quality and network latency. Since lag is insufferable in video games, we need to decrease the video bitrate when network is busy. But once the latency is low enough, we would increase the video bitrate to get a better image quality.

C. Network Security

SSLify is implemented to the web application so that any HTTP request would be redirected to HTTPS. In this way, our application is provided with an HSTS (HTTP Strict Transport Security) policy. For any HTTPS request, a TLS handshake is required, in which the server and client side will transfer their encryption protocol as well as authentication keys. During data transfer, all messages would be encrypted by specific protocols and keys. Therefore, any intruders who intercept messages cannot decrypt them.

III. Machine Learning Application

A. Facial Expression Recognition

We select seven facial expressions as galleries. The photos captured by camera are first transformed into grayscale images, then we detect faces in images, transform them into landmarks, and eventually compare those landmarks (target) with galleries and compute their similarity.

1. Facial Landmark

- Face Detection

We first transform original images into grayscale. Then, we employ a pretrained detector provided by Dlib to detect human faces in grayscale images. The first detected face is chosen as an input image and passed to next process.

- Landmark Detection

The first detected face images are then marked with 68 facial landmarks after passed into another pretrained detector "shape_predictor_68_face_landmarks." The target landmarks are shown in fig. 7. The numbers shown in fig. 7 are used as fixed index. For example, we can always locate mouth positions by accessing landmarks 49~68.



Fig. 2 Facial landmarks

left eyebrow	landmark[18:22]
right eyebrow	landmark[13:27]
left eye	landmark[37:42]
right eye	landmark[43:48]
nose	landmark[28:36]
mouth	landmark[49:68]
face contour	landmark[1:17]

Table 1 Facial landmarks

The following computation methods are all based on position of those landmarks.

2. Facial Expressions Comparison

Local facial expression

As shown in below, there are several member function related to local facial expression. Those are conditions we used to distinguish if a target belongs to any gallery and compute their similarity. Their computation algorithms will be illustrated in this part. For further explanation, our computing algorithm only use ratio or relative size since the target size won't be the same. [Landmark are replaced by L]

- Frown

We first compute the ratio of distance between eyebrows($L[43,X]-L[40,X]$) over distance between eyes(cantheses)($L[23,X]-L[22,X]$). If all the conditions that (1)the ratio <0.7 , (2)the vertical distance between left/right canthus and left/right inner eyebrows is smallest among left/right eyes and left/right eyebrows landmarks, (3)the one between left/right middle eye and left/right middle eyebrow is largest, and (4) the left/right inner eyebrows($L[22,Y]/L[23,Y]$) is the lowest point among left/right eyebrows, we determine the target is frowning.
[similarity= $(1-\text{eyebrow_dis}/\text{eye_dis}) * 1.5$]

- MouthOpen

If the vertical distance between upper mouth and lower mouth ($L[67,Y]-L[63,Y]$) is bigger than thickness of mouth($L[63,Y]-L[52,Y]$), we determine the target is opening his/her mouth. [similarity=1]

- MouthClosed

If the vertical difference between mean of upper mouth points ($L[66:68,Y]$) and mean of lower mouth point ($L[62:64,Y]$) is smaller than thickness of mouth($L[63,Y]-L[52,Y]$), we determine the target is closing his/her mouth. [similarity=1]

- MouthLeft

We set middle point of nose($L[28,X]$) as point of reference. After calculating the difference between mouth landmarks and the point of reference. If the amount of point whose difference is positive(viewed as right point) is less than 8/10, we determine the target's mouth is in left side. [similarity= $1-\text{positive_num}/20 * 0.5$]

- MouthRight

We set middle point of nose($L[28,X]$) as point of reference. After calculating the difference between mouth landmarks and the point of reference. If the amount of point whose difference is negative(viewed as left point) is less than 8/10, we determine the target's mouth is in right side. [similarity= $1-\text{negative_num}/20 * 0.5$]

- Smile

We aim to detect if the mouth corners are rising. If mean of upper mouth point($L[61:68, Y] + L[49, Y] + L[55, Y]$) is lower than left/right mouth corners($L[49, Y] / L[55, Y]$), we determine the target is smiling. [similarity= $0.8 + 0.2 * (\text{height} - L[49, Y] + L[55, Y]) / 2 / \text{height}$]

- PointDown

We aim to detect if the mouth corners are pointing down. If mean of upper mouth point($L[65:68, Y] + L[60, Y]$) is higher than left/right mouth corners($L[49, Y] / L[55, Y]$), we determine the corner of target's mouth are pointing down.

[similarity= $0.8 + 0.2 * (\text{height} - L[49, Y] + L[55, Y]) / 2 / \text{height}$]

- MouthOval

We first calculate the width($L[55, X] - L[49, X]$) and the height($L[58, Y] - L[52, Y]$) of mouth. If the ratio of height over width is larger than 1, we determine the contour of target's mouth is oval.

[similarity=1]

- MouthCircle

We first calculate the width($L[55, X] - L[49, X]$) and the height($L[58, Y] - L[52, Y]$) of mouth. If $(\text{height} / \text{width} - 1)$ is smaller than 0.5, we determine the contour of target's mouth is circle.

[similarity= $1 - \text{abs}(\text{height} / \text{width} - 1) * 0.5$]

- FaceLeft

We set middle point of nose($L[28, X]$) as point of reference. We calculate the horizontal distance between left/right face($L[1, X] / L[17, X]$) and point of reference. If $\text{left_dis} / \text{right_dis} > 1.5$, we determine the target is turning to left. [similarity=1]

- FaceRight

We set middle point of nose($L[28, X]$) as point of reference. We calculate the horizontal distance between left/right face($L[1, X] / L[17, X]$) and point of reference. If $\text{right_dis} / \text{left_dis} > 1.5$, we determine the target is turning to right. [similarity=1]

Galleries








facial expression		
	conditions	MouthClosed, Not Smile, Not PointDown Not Frown
	similarity	1
	conditions	MouthOpen, Smile
	similarity	$\text{smile_score} * 0.7 + \text{mouth_open} * 0.3$
	conditions	Not Frown MouthLeft, MouthCircle FaceLeft
	similarity	$\text{mouth_left_score} * 0.5 + \text{mouth_circle_score} * 0.5$
	conditions	Not Frown MouthRight, MouthCircle FaceRight
	similarity	$\text{mouth_right_score} * 0.5 + \text{mouth_circle_score} * 0.5$
	conditions	MouthLeft, MouthClosed, Not smile
	similarity	$\text{mouth_left_score} * 0.5 + \text{mouth_closed_score} * 0.5$
	conditions	PointDown
	similarity	$\text{mouth_pointDown_score}$
	conditions	MouthOpen, Not MouthLeft, MouthCircle or MouthOval Not Frown
	similarity	$\text{mouth_open_score} * 0.3 +$ $(\max(\text{mouth_oval_score}, \text{mouth_circle_score})) * 0.7$

Table 2 Facial expression detection algorithms

Implementation

We will now then explain the code and package used with each section just mentioned, along with screenshots for each part of code.

I. Basic Streaming Features

A. Video Streaming

There are two parts in video streaming:

- a. upload to server:
 1. get a snapshot from camera
 2. compress the image to jpeg
 3. use socketio to upload the jpeg to the server

```
ctx.drawImage(video, 0, 0, video.videoWidth, video.videoHeight, 0, 0, 320, 180);  
  
let dataURL = canvas.toDataURL('image/jpeg', quality);  
socket.emit('input image', dataURL);
```

- b. download processed video from server:
 1. use http to request the image

```
<div id="container">  
  <video autoplay="true" id="videoElement">  
  </video>  
    
  <canvas id="canvasElement"></canvas>  
</div>
```

B. Sound Streaming

- a. upload to server:
 1. get audio from microphone(use getUserMedia)
 2. create as AudioContext
 3. connect Audiocontent with audio stream and store in a buffer
 4. select the largest value in every buffer and upload it to server

```
navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {  
  video.srcObject = stream;  
  localMediaStream = stream;  
  var context = new AudioContext();  
  var liveSource = context.createMediaStreamSource(stream);  
  var levelChecker = context.createScriptProcessor(1024,1,1);  
  liveSource.connect(levelChecker);  
  levelChecker.connect(context.destination);  
  levelChecker.onaudioprocess = function(e) {  
    var buffer = e.inputBuffer.getChannelData(0);  
    var maxVal = 0;  
    for (var i = 0; i < buffer.length; i++) {  
      if (maxVal < buffer[i]) {  
        maxVal = buffer[i];  
      }  
    }  
    detectaudio = maxVal;  
  }  
});
```

- b. reation comes from server
 1. server use socketIO to get the audio

```
@socketio.on('input audio', namespace='/test')
def test_audio(input):
    #input = input.split(",")[1]
    camera.audio(input)
```

2. Server analyzes audio volume. If the volume is bigger than 80, the audio related icon (we use the Kaohsiung Mayor's photo this time) will disappear.

II. Advanced Network Features

A. Deploying to Heroku

This requires several steps using Heroku Cli command line tool as shown.

1. sign up for a Heroku account on Heroku official website
2. install heroku


```
$ brew install heroku
```
3. install heroku cli


```
$ brew tap heroku/brew && brew install heroku
```
4. \$ git clone

<https://github.com/evamo0508/Face-Dance-Machine-python-webcam-flask.git>
5. \$ cd Face-Dance-Machine-python-webcam-flask
6. download landmarks.dat at <https://goo.gl/Z2JCch> and put it in current directory
7. \$ heroku login
8. \$ heroku create
9. Deploy to heroku


```
$ git push heroku master
```

 It will remotely install dependencies in requirements.txt automatically.
10. \$ heroku open
 It redirects to your default browser and open the demo website.

B. Rate Control

We estimate the network traffic condition by the length of the sending queue in the server. Every second, client would send a request for the length of the queue. If the queue is longer than 2, we would decrease the bitrate. Else if the queue is empty we would increase the bitrate.

```
update_quality = function(output_length, process_length){
  if (output_length >= 2 || process_length >= 2)
    worse_quality();
  else if (output_length == 0 && process_length == 0)
    better_quality();
  console.log('Current quality: ' + quality);
};
```

We also control the bitrate by adjusting the compress quality rate of the jpeg to be sent. The quality rate is in the range from 0.1 to 0.8.

```
function better_quality() {
  if (quality < 0.7) quality = quality + 0.1;
}

function worse_quality() {
  if (quality > 0.2) quality = quality - 0.1;
}
```

C. Network Security

We are using an extension of python Flask as the security package, which is flask-SSLify. The usage is shown in the following screenshot.

```
from sys import stdout
from makeup_artist import Makeup_artist
import logging
from flask import Flask, render_template, Response
from flask_socketio import SocketIO
from flask_sslify import SSLify
from camera import Camera
from utils import base64_to_pil_image, pil_image_to_base64

app = Flask(__name__)
sslify = SSLify(app)
```

It is so simple that you simply have to from flask_sslify import SSLify, and then use SSLify to wrap the flask app object up.

III. Machine Learning Application

A. Facial Expression Recognition

1. flip the image so the output can be intuitive.

```
img=img.transpose(Image.FLIP_LEFT_RIGHT)
```

2. use pretrained detector to find out facial landmarks

```
self.detector = dlib.get_frontal_face_detector()
self.predictor = dlib.shape_predictor("landmarks.dat")
self.fa = face_utils.FaceAligner(self.predictor, desiredFaceWidth=256)

gray=cv2.cvtColor(np.array(target.convert('RGB')), cv2.COLOR_RGB2GRAY)
start = time.time()
rects = self.detector(gray, 0)
#print("face_dance takes: ", 1000*(time.time()-start))
try:
  shape = self.predictor(gray, rects[0])
  shape = face_utils.shape_to_np(shape)
```

3. Use the rules mentioned in Achievement section III to distinguish the expressions.(which are written in similarity.py)

Challenge

I. javascript reloading cache problem

At first, we modified our .js code and push it to Heroku, and soon found out there are not any differences in the browser. Thanks to Byron Hsu, who told us some tips while he was presenting his video in class, we were able to figure out that we had to check the box “Disable cache” in the console log. In that way, the browser would load the latest javascript file.

II. App only visible under internal network

At first, we utilized our laptop as the server instead of Heroku cloud server. However, the web app is visible only under the same local area network. In other words, the client side has to connect to the same wifi as the server to play the game. We are not satisfied with this restriction. Therefore, we managed to deploy our project to Heroku, providing clients with an url to access under any networks.

III. dlib installation

Dlib is the package used to predict facial landmarks. As for the installation, to “pip install dlib” is easy as we can simply put this line in “requirements.txt”. Nevertheless, dlib requires some dependencies including cmake and boost, which are recommended to install via sudo apt-get. It is impossible to sudo apt-get while pushing the project to Heroku server. The most solutions we find on the Internet is that we have to first build a docker image, and import it to the project, but we fail to do so.

At last, we found a perfect solution, which is to add “boost==0.1”, “cmake==0.7.1”, and “boost-py” in “requirements.txt” before the “dlib” line. It works like charm!

IV. Network condition estimation (for rate control)

At first, we wanted to estimate the network traffic by image packet loss rate. However, we found that all the images were sent and received without loss even when the latencies were huge. Therefore, we need to change the indicator. We noticed that when the network was busy, the sending list at the server would start to stack. Therefore, we choose the length of the list as our new indicator. Finally it worked quite well.

Future Work

Currently, the game is aimed for a single person. We would like to make it a two-person-online game so that people can play against each other, enhancing the value of this app. To fulfill this target, the server should have different port for different clients to transfer the video messages.

Moreover, we should try to solve the delay problem. Although some rate control has been done, there is still some delay. To meet the need of competition, the transfer rate should be increased to the standard that people cannot tell.

Finally, we can design a model to classify the facial expression instead of using rule-based knowledge. This way can be more robust for recognizing various shape of face features.

Reference

- [1] python-web-flask github repository, https://github.com/log0/video_streaming_with_flask_example/blob/master/main.py
- [2] miguelgrinberg's blog about authentication, <https://blog.miguelgrinberg.com/category/Authentication>
- [3] 為你自已學 Ruby on Rails 系列30章, 第 29 章 - 網站部署(使用 Heroku), <https://ithelp.ithome.com.tw/articles/10189021>
- [4] Flask-SSLify github repository, <https://github.com/kennethreitz/flask-sslify>
- [5] Flask-SocketIO, <https://flask-socketio.readthedocs.io/en/latest/>
- [6] HTML with SocketIO, <https://ithelp.ithome.com.tw/articles/10102886>
- [7] Web Audio API, <https://zhuanlan.zhihu.com/p/26536898?fbclid=IwAR14Rz-QEfKlCogzBQnOBDES0zthrKG8n2ynh0RBv1yWrvLk0rUy4fs1PpI>