Face Tracking for Animated Video with Tracking.js

Computer Vision in Web Based Application

Fanda Yuliana Putri
Informatics/Computer Science, School of Electrical Engineering and Informatics
Institut Teknologi Bandung,
Jl. Ganesha No. 10, Bandung 40132, Indonesia
13514023@std.stei.itb.ac.id

***Abstract*—Computer vision is a scientific discipline that is concerned with the theory and technology to obtain information from image. Computer vision is being used in a wide variety of real-world applications, which include match move, medical imaging, auto safety, and others. The existed applications were limited in the form of native apps which need more complex environment. Meanwhile, Tracking.js is one of the modern approaches for bringing computer vision into the web. This open source JavaScript library contains different algorithms and techniques to be used on the browser environment. This library allows the developer(s) to create interactive yet effective web based application, including color tracking, face tracking, face detection, and much more. Tracking.js will interpret both predefined or real-time images and videos, and then the developer can manipulate the result to get desired output. All those processes happened on the client-side of the browser.**

Keywords—computer vision; face tracking; animation, web application

I. INTRODUCTION

Humans utilize their eyes and brain to sense and gain information from their surroundings. The eyes will receive subtle patterns of light and shading from the surface of any objects. From that point onward, human’s brain combines all those visions and then interprets it into some perceptions regarding the related information. Then, we can conclude whether the objects are moving or staying in place, have colors or not, being far or close to us, and many more. Moreover, from those shading and stuff, human’s brain can get the sense of the 3D model from the particular object. Humans do that kind of process on daily basis almost effortlessly. But few years ago, it was still impossible for the machine to do it.

In parallel with the mathematical techniques for analyzing pattern, shape, and appearance of objects in imagery, the researchers have been aiming to give a similar capability of interpreting pictures to a machine or a computer. The discipline field is called computer vision. In short, computer vision is a discipline that studies how to reconstruct, interpret and understand a 3D scene from its 2D images in terms of the properties of the structures present in the scene [3]. The researches in computer vision have been tremendously developing into something that may be even better than human’s vision at some cases. In an example is face recognition. In the early year of computer vision development, the computer hardly recognized or identified the structure of the human face. But now days, with a little bit of data training, the combination of face recognition algorithm and artificial intelligence, an application can recognize the face of a certain person with highly noticeable accuracy and precision.

There are a lot of frameworks or libraries from various languages that take up computer vision as their main object of development, but in this paper, the focus of the discussion is Tracking.js, one of the JavaScript library that bring computer vision into the browser environment. With the power of JavaScript as their core, Tracking.js has become one of the potential resources in computer vision. In parallel with the creativity of the developer, the client-side processing and its lightweight core will conduct a faster web development. For example, this library could support the movement-based web interface with its further-developed features such as color detection and face recognition [4]. Unfortunately, due to its early development phase, this library still doesn’t have as many algorithms as another platform-based library such as OpenCV [1].

Actually, in the Tracking.js official website, there are a lot of interesting examples of its usage in web application, such as “Tag Friend” that uses face recognition feature to recognize specific person, “Draw something” that extends color recognition features to draw colorful line on screen, and many more [2]. The discussion will traverse on the use of Tracking.js in simple and interactive web application.

II. RELATED WORKS

The main feature of web application to be discussed here is kind of animation (of gif) generator that would follow along the direction of our face on the screen. This kind of animated video is a common feature on social media such as Snapchat. For example in Figure 1, the user will show their face on the camera and the application will add animation to create funny or scary effect on the resulting video. Besides Snapchat, there are another mobile application that could provide the same effect on real-time videos, for example MSQRD, SNOW, BOO!, and Camera360. Those applications also provide animated video with diverse filters to create the expected effect.

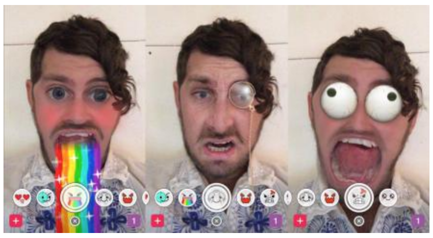
Figure 1 Snapchat Animated Filter

Source: https://tctechcrunch2011.files.wordpress.com/

As mentioned in the previous chapter, this kind of application was limited in native apps such as mobile apps or desktop apps. We barely find this kind of application on website because it needs slightly complicated computation and powerful backend to support both real-time and lightweight website.

III. METHODS

The steps to create animated video on the users' face are quite simple; those are recognizing the user's face structure and locate the desired animation on the right place. The animation should be well-placed to create flawless effect and increase the resemblance with the real thing. For that, the developer(s) need a well-build framework or library with good accuracy when analyzing the user's face structure. Well known library for face tracking are build with C/C++, Python, or Java.

In this project, Tracking.js will be utilized to locate the users' face with its predefined algorithm. Other things to do to make this project works are loading the frames for the animation, manage the timing of the frames to create realistic animation, and locate the animation to create an impactful effect, including resizing the frames and finding the right place for the animation.

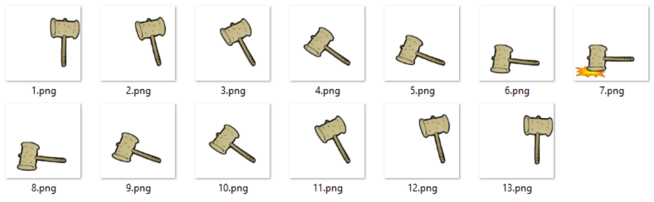

Figure 2 Hammer Frames

Frames for the animation are actually a bunch of images that contains related pictures of moving object as shown in Figure 2. These images will be saved as an array and will be processed alternately and sequentially to create a moving object, it's just the same like how other animation works. Here's a code snippet from the main JavaScript file to load the image.

```
function loadImages() {
   var promises = [];
   for (var i = 1; i < 14; i++) {
        var deferred = new $.Deferred();
        var img = new Image();
        img.onload = deferred.resolve;
        img.src = "img/" + i + ".png";
        hammerFrame.push(img);
        promises.push(deferred.promise());
   }
   return $.when.apply($, promises);
}
```

After the frames have been loaded, the next step is locate the user's face location with Tracking.js. Before that, we should make sure that the browser support Modernizr to get the user media such as camera or audio (if it's needed). After that, make sure that there's an available camera, it can be a front camera or embedded one. The code below is a snippet to find camera on a computer.

```
function searchForFrontCamera() {
   var deferred = new $.Deferred();

   if (MediaStreamTrack && MediaStreamTrack.getSources) {
      MediaStreamTrack.getSources(function (sources) {

        var rearCameraIds = sources.filter(function
         (source) {
             return (source.kind === 'video' &&
             source.facing === 'user');
        }).map(function (source) {
              return source.id;
        });
        if (rearCameraIds.length) {
             deferred.resolve(rearCameraIds[0]);
        } else {
             deferred.resolve(null);
        }
      });
   } else {
      deferred.resolve(null);
   }
   return deferred.promise();
}

var deferred = new $.Deferred();
if (!Modernizr.getusermedia) {
   deferred.reject('Your browser doesn\'t support
   getUserMedia (according to Modernizr).');
}
deferred.resolve();
```

After getting the permission to use the camera and set it into ready-to-use state, the next step is to locate the user's face location with Tracking.js. We just need to add some line of codes that are already available on its official website. Here's some code snippet for setting up the Tracking.js in our project.

```
var tracker = new tracking.ObjectTracker('face');
   tracker.setInitialScale(4);
   tracker.setStepSize(2);
   tracker.setEdgesDensity(0.1);
trackingTask = tracking.track('#step1 video', tracker);
```

We just need to state the name of the object that we want to track, in this example it's 'face'. Another thing to do is initialize the precision of the tracker when identifying the users' face. After that, start the tracking. While tracking the face, there's another thing we need to do at the same time, and that is locating the animation that has been prepared earlier. Here's the code snippet. We can also get the coordinate information from the face location on the screen. It can be used as the guide for animation positioning later on.

```
tracker.on('track', function (event) {
   ctx.clearRect(0, 0, canvas.width, canvas.height);
   hammer = [];

   event.data.forEach(function (rect) {
        frameCount++;
        var orgWidth = 240;
        var orgHeight = 240;
        var newWidth = (rect.width * 2);
        var newHeight = newWidth / orgWidth * orgHeight;
        var fixTop = rect.height * 0.2;
        var fixLeft = -rect.width / 2;
        var image = hammerFrame[frameCount %
         hammerFrame.length];

        hammer.push({
           image: image,
           x: (rect.x + fixLeft),
           y: (rect.y - newHeight + fixTop),
           width: newWidth,
           height: newHeight
        });
   ctx.drawImage(image,  (rect.x  +  fixLeft),  (rect.y  -
    newHeight + fixTop), newWidth, newHeight);
   });
 });
```

There's no exact value for the code above. It does all depend on the size of the video screen and the type of the animation. Some code may need another adjustment according to the requirements itself. Beside the main JavaScript file, we do need other files such as HTML files and CSS files to make our project works perfectly. But those files are really depends on the need of the project and it's mainly talks about the design. So it's basically depending on the hands of the developer to decide what kind of application we have.

IV. RESULTS AND DISCUSSION

After the main program has been completed, the next thing to do is testing our program to check whether it works perfectly or not. To run the main program locally, we need a basic HTTP server to serve out pages. We can use XAMPP, AMPP, or other related things. After that, make sure that the permission for the camera usage is on and then the program should running by now.

Here's the result of the writer's simple video animation web apps as shown in Figure 3. It shows the results from the camera on the computer.

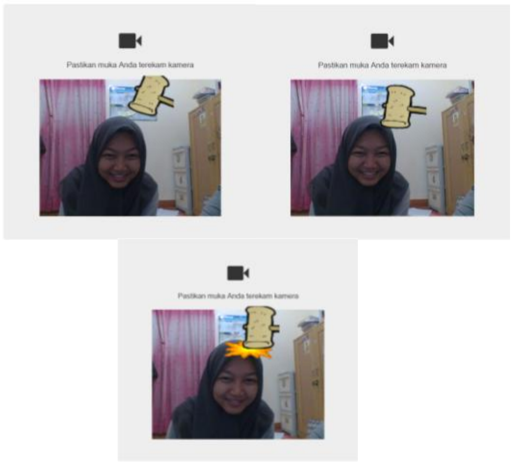
Pastikan muka Anda terekam kamera
Pastikan muka Anda terekam kamera
Pastikan muka Anda terekam kamera

Figure 3 Animated Video with Face Tracking Testing Result

The testing result shows that the program works perfectly fine. The desired animation is well-aligned with the face so it could produce a good effect on the video. The writer also tried this program on many browsers. It works fine on the latest Google Chrome, Mozilla Firefox, and Microsoft Edge. It doesn't work on Internet Explorer because it cannot detect getUserMedia. Even though the program works perfectly fine, in some cases, the writer found out that the face tracking algorithm is lost the moment the writer try the program when wearing hijab. The result is not as good as the time the writer didn't wear hijab. Probably it's because the writer's eyes didn't showing perfectly on screen. At some angle it couldn't detect the face at all. But besides that problem, overall the program works fine. Tracking.js really made this project easy to make and fun to use.

V. CONCLUSION

Based on the making process and program testing, the writer concludes that Tracking.js is a good JavaScript library and has so many potentials in it. The main program works perfectly fine with Tracking.js as the library to track users' face location. This library makes the process so much easier and fun. For the future development it'll be great if there's a feature to save the video with animation in it.

ACKNOWLEDGMENT

The writer would like to thank God for all His mercy so this paper could be done finely without any significant problem. The writer also feels grateful towards all the Socio-informatics and Professionalism Lecturers who supported my works so it could get a better result.

REFERENCES

[1] *Tracking.js & The Computer Vision Power of Javascript*. Retrieved May 4, 2017, from https://usersnap.com/blog/tracking-js-the-computer-vision-power-of-javascript/

[2] *Tracking.js - A Modern Approach For Computer Vision on The Web*. Retrieved May 4, 2017, from https://trackingjs.com/

[3] *What's Computer Vision?*. Retrieved May 3, 2017, from http://www.bmva.org/visionoverview

[4] *8 Web Development Trends To Look Out For In 2017*. Retrieved May 3, 2017, from https://careerfoundry.com/en/blog/web-development/8-web-development-trends-2017/

PRONOUNCEMENT

I hereby stating that the paper I wrote is my own writing, not an adaption, or translation from someone else's paper, and also not plagiarism.

Bandung, May 4, 2017

[illegible]

Fanda Yuliana Putri (13514023)