РАФАЭЛЬ МУН(RAPHAEL MUN)

Серия статей «Фильтры искусственного интеллекта лица в браузере»

УЧЕБНЫЕ РУКОВОДСТВА

Перевод: С. Кузнецов, 2024 г.

Articles: AI Face Filters in the Browser

Raphael Mun

2021

https://www.codeproject.com/Articles/instafluff#Article

Серия статей «Фильтры искусственного интеллекта лица в браузере»

Рафаэль Мун

2021

https://www.codeproject.com/Articles/instafluff#Article

Перевод: С. Кузнецов, 01.03.2024

Статья 6 «Активация волшебства на экране с помощью лица, в браузере с использованием библиотеки TensorFlow.js»

Статья **6 «Активация волшебства на экране с помощью лица, в браузере с использованием библиотеки TensorFlow.js»(«Activating Screen Magic with Your Face in the Browser with TensorFlow.js»); https://www.codeproject.com/Articles/5293496/Activating-Screen-Magic-with-Your-Face-in-the-Brow)** является статьей из серии статей **Фильтры искусственного интеллекта лица в браузере(AI Face Filters in the Browser)**.

9 февраля 2021

В этой, заключительной статье этой серии статей мы реализуем **обнаружение моргания глазами и открытие рта(detect eye blinks and the mouth opens)**, чтобы сделать интерактивную сцену.

Разве не было бы крутым серию статей закончить статьей о событиях, когда мы обнаружим изменения на наших лицах? Здесь мы покажем, как использовать **ключевые лицевые точки(key facial points)**, чтобы проверить на событие, **когда мы открываем рот и моргаем глазами(when we open our mouths and blink our eyes)**, чтобы **активировать события на экране(activate on-screen events)**.

- **Загрузка кода и файлов - 565.6 KB**

Введение

Приложения, подобные приложению Snapchat, предлагают удивительное разнообразие фильтров лиц и линз, которые позволяют вам накладывать интересные эффекты на фотографии и видео. Если когда-либо вы «приделывали» себе виртуальные уши собаки или маскарадную шляпу, то знаете, это может быть забавно!

Задавались ли вы вопросом, как создать эти виды фильтров с нуля? Ну, теперь есть шанс научиться делать все в веб-браузере! В этой серии статей мы собираемся показать, как в браузере создать фильтры в стиле **Snapchat**, обучить **AI**-модель понимать выражения лица, используя отслеживание лица и библиотеку машинного обучения **Tensorflow.js**.

(надпись под снимком экрана: **Закрыты глаза:нет Открыт рот:да (Eyes: false Mouth: true)**

Вы можете загрузить демонстрационный пример этого проекта. Возможно, для обеспечения производительности, вы будете должны в своем веб-браузере включить поддержку **Web**-графики **WebGL**. Также можно загрузить код и файлы для этой серии статей.

Если вы плохо знакомы с фреймворком **TensorFlow.js**, то рекомендуем прочитать статью **«Начало работы с глубоким изучением в браузере с использованием фреймворка TensorFlow.js»(«Getting Started With Deep Learning in Your B r o w s e r U s i n g T e n s o r F l o w . j s » ; https://www.codeproject.com/Articles/5272760/Getting-Started-With-Deep-Learning-in-Your-Browser)**.

Если хотели бы увидеть больше того, что возможно в веб-браузере с помощью фреймворка **TensorFlow.js**, прочтите статьи из серии по **искусственному интеллекту(AI)**: «Собаки и пицца: машинное зрение в браузере с использованием TensorFlow.js»**(«Dogs and Pizza: Computer Vision in the Browser With T e n s o r F l o w . j s » ; https://www.codeproject.com/Articles/5272771/Dogs-and-Pizza-Computer-Vision-in-the-Browser-With)** и **«Роботы чатов с помощью фреймворка TensorFlow.js»(Chatbots using TensorFlow.js.)**.

Добро пожаловать к прочтению заключительной статьи этой серии статей по **искусственному интеллекту(ИИ; AI)**, статьи о виртуальной забаве с отслеживанием лица! Разве не было бы крутым серию статей закончить статьей о событиях, когда мы обнаружим изменения на наших лицах? Здесь мы покажем, как использовать **ключевые лицевые точки(key facial points)**, чтобы проверить на событие, **когда мы открываем рот и моргаем глазами(when we open our mouths and blink our eyes)**, чтобы **активировать события на экране(activate on-screen events)**.

Обнаружение моргания глазами и открытие рта

Мы собираемся использовать использовать **ключевые лицевые точки(key facial points)** из кода отслеживания лица, который мы разработали в первой статье этой серии статей, **«Отслеживание лица в реальном времени в браузере с помощью библиотеки TensorFlow.js»(«Real-Time Face Tracking in the Browser with TensorFlow.js»)**, чтобы обнаружить моргания глазами и открытие рта.

Аннотируемые точки лица дают достаточно информации, чтобы определить, когда глаза закрыты и когда рот открыт. Трюк в коде здесь должен масштабировать позиции с учетом относительного размера всего лица.

Для этого мы можем обратиться к удобному **расстоянию между глазами(eye-to-eye distance)**, чтобы аппроксимировать относительный масштаб лица **faceScale** в функции отслеживания лица **trackFace**:

JavaScript

```
async function trackFace() {
    ...

    faces.forEach( face => {
        const eyeDist = Math.sqrt(
            ( face.annotations.leftEyeUpper1[ 3 ][ 0 ] -
                    face.annotations.rightEyeUpper1[ 3 ][ 0 ] ) ** 2 +
            ( face.annotations.leftEyeUpper1[ 3 ][ 1 ] -
                    face.annotations.rightEyeUpper1[ 3 ][ 1 ] ) ** 2 +
```

```
            ( face.annotations.leftEyeUpper1[ 3 ][ 2 ] -
                    face.annotations.rightEyeUpper1[ 3 ][ 2 ] ) ** 2
        );
        const faceScale = eyeDist / 80;
    });

    requestAnimationFrame( trackFace );
}
```

Затем мы можем вычислить расстояние между верхней частью глаза и нижней частью глаза, для обоих глаз, и левого глаза и правого глаза, и использовать значение относительного масштаба лица `faceScale`, чтобы чтобы аппроксимировать, когда порог был пересечен. Подобное вычисление мы можем использовать для обнаружения открывания рта.

Посмотрите:

JavaScript

```
async function trackFace() {
    ...
    let areEyesClosed = false, isMouthOpen = false;
    // добавлено при переводе, русификация сообщения:
    let rus_areEyesClosed = `нет`, rus_isMouthOpen = `нет`;

    faces.forEach( face => {
        ...

        // Проверка на то, что глаза закрыты
        // Check for eyes closed
        const leftEyesDist = Math.sqrt(
            ( face.annotations.leftEyeLower1[ 4 ][ 0 ] -
                    face.annotations.leftEyeUpper1[ 4 ][ 0 ] ) ** 2 +
            ( face.annotations.leftEyeLower1[ 4 ][ 1 ] -
                    face.annotations.leftEyeUpper1[ 4 ][ 1 ] ) ** 2 +
            ( face.annotations.leftEyeLower1[ 4 ][ 2 ] -
                    face.annotations.leftEyeUpper1[ 4 ][ 2 ] ) ** 2
        );
        const rightEyesDist = Math.sqrt(
            ( face.annotations.rightEyeLower1[ 4 ][ 0 ] -
                    face.annotations.rightEyeUpper1[ 4 ][ 0 ] ) ** 2 +
            ( face.annotations.rightEyeLower1[ 4 ][ 1 ] -
                    face.annotations.rightEyeUpper1[ 4 ][ 1 ] ) ** 2 +
            ( face.annotations.rightEyeLower1[ 4 ][ 2 ] -
                    face.annotations.rightEyeUpper1[ 4 ][ 2 ] ) ** 2
```

```
);
if( leftEyesDist / faceScale < 23.5 ) {
    areEyesClosed = true;
     // добавлено при переводе, русификация сообщения:
     rus_areEyesClosed = `да`;
}

if( rightEyesDist / faceScale < 23.5 ) {
    areEyesClosed = true;
     // добавлено при переводе, русификация сообщения:
     rus_areEyesClosed = `да`;
}

// Проверка на то, что рот открыт
// Check for mouth open
const lipsDist = Math.sqrt(
    ( face.annotations.lipsLowerInner[ 5 ][ 0 ] -
     face.annotations.lipsUpperInner[ 5 ][ 0 ] ) ** 2 +
    ( face.annotations.lipsLowerInner[ 5 ][ 1 ] -
     face.annotations.lipsUpperInner[ 5 ][ 1 ] ) ** 2 +
    ( face.annotations.lipsLowerInner[ 5 ][ 2 ] -
     face.annotations.lipsUpperInner[ 5 ][ 2 ] ) ** 2
);
// Масштаб относительно размера лица
 // Scale to the relative face size
if( lipsDist / faceScale > 20 ) {
            isMouthOpen = true;
           // добавлено при переводе, русификация сообщения:
           rus_isMouthOpen = `да`;
        }
    });
        // добавлено при переводе, русификация сообщения:
        setText( `Закрыты глаза: ${rus_areEyesClosed} Открыт рот:
                                ${rus_isMouthOpen}` );
        // закомментировано при переводе, удалено сообщение на английском:
        // setText( `Eyes: ${areEyesClosed} Mouth: ${isMouthOpen}` );

    requestAnimationFrame( trackFace );
}
```

Теперь все установлено и настроено, чтобы обнаружить некоторые события на лице(события мимики).

Время для вечеринки с конфетти

Каждое празднование нуждается в конфетти, правильно? Мы собираемся соединить виртуальное конфетти с морганием глазами и открытием рта, чтобы сделать вечеринку с конфетти.

Для этого мы будем использовать `JavaScript`-библиотеку, с открытым исходным кодом, с названием Party-JS. Включите ее наверху своей страницы, как в этом коде:

JavaScript

```
<script
src="https://cdn.jsdelivr.net/npm/party-js@1.0.0/party.min.js"></script>
```

Давайте сохраним состояние в глобальной переменной, чтобы отследить, запустили ли мы уже конфетти или нет.

JavaScript

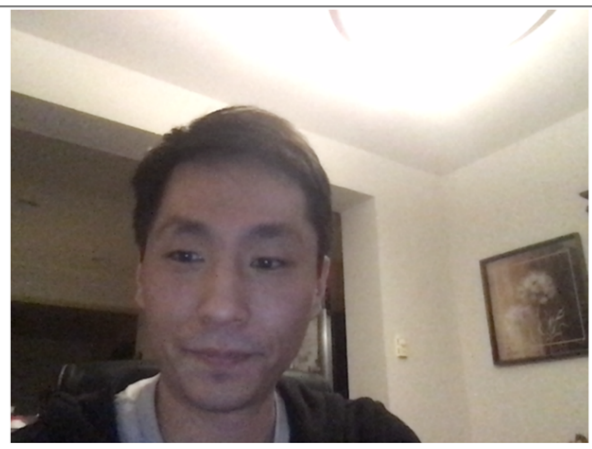
```
let didParty = false;
```

Наконец, что не менее важно, мы можем активировать анимацию вечеринки каждый раз, когда моргнули глазами или открыли рот.

JavaScript

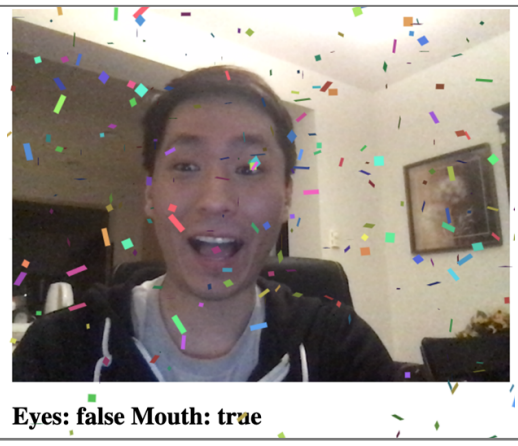
```
async function trackFace() {
    ...

    if( !didParty && ( areEyesClosed || isMouthOpen ) ) {
        party.screen();
    }
    didParty = areEyesClosed || isMouthOpen;

    requestAnimationFrame( trackFace );
}
```

И теперь - время вечеринки! Используя мощь отслеживания лица и конфетти, вы получили вечеринку на экране прямо на ваших губах.



Eyes: false Mouth: false

(надпись под снимком экрана: **Закрыты глаза:нет Открыт рот:нет (Eyes:false Mouth:false)**



Eyes: false Mouth: true

(надпись под снимком экрана: **Закрыты глаза:нет Открыт рот:да(Eyes: false Mouth: true)**

Финишная черта

Этот проект не был бы завершен без листинга полного кода, чтобы вы могли рассмотреть его, и поэтому, он здесь:

HTML

```
<html>
    <head>
    <meta charset="UTF-8">
        <title>Активация волшебства на экране с помощью лица</title>
        <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.4.0/dist/tf.min.js"><
/script>
        <script
src="https://cdn.jsdelivr.net/npm/@tensorflow-models/face-landmarks-detect
ion@0.0.1/dist/face-landmarks-detection.js"></script>
```

```
        <script
src="https://cdn.jsdelivr.net/npm/party-js@1.0.0/party.min.js"></script>
    </head>
    <body>
        <canvas id="output"></canvas>
        <video id="webcam" playsinline style="
            visibility: hidden;
            width: auto;
            height: auto;
            ">
        </video>
        <h1 id="status">Загрузка.../Loading...</h1>
        <script>
        function setText( text ) {
            document.getElementById( "status" ).innerText = text;
        }

        async function setupWebcam() {
            return new Promise( ( resolve, reject ) => {
                const webcamElement = document.getElementById( "webcam" );
                const navigatorAny = navigator;
                navigator.getUserMedia = navigator.getUserMedia ||
                navigatorAny.webkitGetUserMedia ||
                     navigatorAny.mozGetUserMedia ||
                navigatorAny.msGetUserMedia;
                if( navigator.getUserMedia ) {
                    navigator.getUserMedia( { video: true },
                        stream => {
                            webcamElement.srcObject = stream;
                            webcamElement.addEventListener( "loadeddata",
                              resolve, false );
                        },
                    error => reject());
            }
            else {
                reject();
            }
        });
    }

    let output = null;
    let model = null;
    let didParty = false;

    async function trackFace() {
        const video = document.getElementById( "webcam" );
        const faces = await model.estimateFaces( {
            input: video,
            returnTensors: false,
            flipHorizontal: false,
        });
```

```
output.drawImage(
    video,
    0, 0, video.width, video.height,
    0, 0, video.width, video.height
);

let areEyesClosed = false, isMouthOpen = false;
 // добавлено при переводе, русификация сообщения:
 let rus_areEyesClosed = `нет`, rus_isMouthOpen = `нет`;

faces.forEach( face => {
    const eyeDist = Math.sqrt(
        ( face.annotations.leftEyeUpper1[ 3 ][ 0 ] -
          face.annotations.rightEyeUpper1[ 3 ][ 0 ] ) ** 2 +
        ( face.annotations.leftEyeUpper1[ 3 ][ 1 ] -
           face.annotations.rightEyeUpper1[ 3 ][ 1 ] ) ** 2 +
        ( face.annotations.leftEyeUpper1[ 3 ][ 2 ] -
          face.annotations.rightEyeUpper1[ 3 ][ 2 ] ) ** 2
    );
    const faceScale = eyeDist / 80;

    // Проверка на то, что глаза закрыты
    // Check for eyes closed
    const leftEyesDist = Math.sqrt(
        ( face.annotations.leftEyeLower1[ 4 ][ 0 ] -
          face.annotations.leftEyeUpper1[ 4 ][ 0 ] ) ** 2 +
        ( face.annotations.leftEyeLower1[ 4 ][ 1 ] -
          face.annotations.leftEyeUpper1[ 4 ][ 1 ] ) ** 2 +
        ( face.annotations.leftEyeLower1[ 4 ][ 2 ] -
          face.annotations.leftEyeUpper1[ 4 ][ 2 ] ) ** 2
);
const rightEyesDist = Math.sqrt(
    ( face.annotations.rightEyeLower1[ 4 ][ 0 ] -
      face.annotations.rightEyeUpper1[ 4 ][ 0 ] ) ** 2 +
    ( face.annotations.rightEyeLower1[ 4 ][ 1 ] -
      face.annotations.rightEyeUpper1[ 4 ][ 1 ] ) ** 2 +
    ( face.annotations.rightEyeLower1[ 4 ][ 2 ] -
 face.annotations.rightEyeUpper1[ 4 ][ 2 ] ) ** 2
);
if( leftEyesDist / faceScale < 23.5 ) {
    areEyesClosed = true;
      // добавлено при переводе, русификация сообщения:
      rus_areEyesClosed = `да`;

}
if( rightEyesDist / faceScale < 23.5 ) {
    areEyesClosed = true;
      // добавлено при переводе, русификация сообщения:
      rus_areEyesClosed = `да`;
}
```

```
        // Проверка на то, что рот открыт
        // Check for mouth open
        const lipsDist = Math.sqrt(
            ( face.annotations.lipsLowerInner[ 5 ][ 0 ] -
              face.annotations.lipsUpperInner[ 5 ][ 0 ] ) ** 2 +
            ( face.annotations.lipsLowerInner[ 5 ][ 1 ] -
              face.annotations.lipsUpperInner[ 5 ][ 1 ] ) ** 2 +
            ( face.annotations.lipsLowerInner[ 5 ][ 2 ] -
              face.annotations.lipsUpperInner[ 5 ][ 2 ] ) ** 2
        );

        // Масштаб относительно размера лица
        // Scale to the relative face size
        if( lipsDist / faceScale > 20 ) {
            isMouthOpen = true;
         // добавлено при переводе, русификация сообщения:
             rus_isMouthOpen = `да`;
        }
    });

    if( !didParty && ( areEyesClosed || isMouthOpen ) ) {
        party.screen();
    }
    didParty = areEyesClosed || isMouthOpen;

     // добавлено при переводе, русификация сообщения:
      setText( `Закрыты глаза: ${rus_areEyesClosed} Открыт рот:
                        ${rus_isMouthOpen}` );
      // закомментировано при переводе, удалено сообщение на англ.:
      // setText( `Eyes: ${areEyesClosed} Mouth: ${isMouthOpen}` );

        requestAnimationFrame( trackFace );
    }

    (async () => {
        await setupWebcam();
        const video = document.getElementById( "webcam" );
        video.play();
        let videoWidth = video.videoWidth;
        let videoHeight = video.videoHeight;
        video.width = videoWidth;
        video.height = videoHeight;

        let canvas = document.getElementById( "output" );
        canvas.width = video.width;
        canvas.height = video.height;

        output = canvas.getContext( "2d" );
        output.translate( canvas.width, 0 );
        output.scale( -1, 1 ); // Зеркалируем
```

```
            output.fillStyle = "#fdffb6";
            output.strokeStyle = "#fdffb6";
            output.lineWidth = 2;

            // Загрузка модели обнаружения признаков лица
            // Load Face Landmarks Detection
            model = await faceLandmarksDetection.load(
                faceLandmarksDetection.SupportedPackages.mediapipeFacemesh
            );

            setText( "Загружено!/Loaded!" );

            trackFace();
        })();
        </script>
    </body>
</html>
```

Что далее?

Фактически, это - все на данный момент. В этой серии статей мы изучили, как использовать **искусственный интеллект(ИИ; AI)** с лицами, чтобы отследить их в режиме реального времени, обнаружить эмоции на лице и движения рта и глаз. Мы даже с нуля создали нашу собственную «забаву» с **добавленной реальностью(augmented reality)** с виртуальными очками, и все это работает в веб-браузере.

Несмотря на то, что мы приняли решение показать примеры с «забавами», для этой технологии также есть много бизнес-приложений. Вообразите магазин розничной торговли очками, который хочет позволить посетителям веб-сайта примерять очки, при просмотре веб-сайта. Не трудно предположить, как для создания такой функциональности, вы использовали бы знания, полученные при чтении этой серии статей. Хотелось бы надеяться, у вас теперь есть инструменты для создания более полезных решений с использованием **искусственного интеллекта(ИИ; AI)** и библиотеки машинного обучения **TensorFlow.js**.

Попробуйте код с конфетти поместить в проект виртуальных очков, посмотрите, можете ли вы использовать обнаружение эмоций в фотоальбоме. Если вы захотите изучить, как сделать большее с

использованием **искусственного интеллекта(ИИ; AI)** в браузере, прочтите связанные статьи из серии статей по **искусственному интеллекту(AI)**:«Собаки и пицца: машинное зрение в браузере с использованием TensorFlow.js»**(«Dogs and Pizza: Computer Vision in the Browser With T e n s o r F l o w . j s »; https://www.codeproject.com/Articles/5272771/Dogs-and-Pizza-Computer-Vision-in-the-Browser-With)** и **«Роботы чатов с помощью фреймворка TensorFlow.js»(Chatbots using TensorFlow.js.)**.

И если эти серии статей вдохновляют вас разрабатывать более крутые проекты, совместно используйте их с нами! Мы хотели бы услышать о ваших проектах.

Удачи, удовольствия и веселья от кодирования!

Эта статья является статьей из серии статей **Фильтры искусственного интеллекта лица в браузере(AI Face Filters in the Browser)**.