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, 04.02.2024

Статья 1 «Отслеживание лица в реальном времени в браузере с помощью библиотеки TensorFlow.js»

Статья 1 Отслеживание лица в реальном времени в браузере с помощью библиотеки TensorFlow.js(Real-Time Face Tracking in the Browser with TensorFlow.js); https://www.codeproject.com/Articles/5293491/Real-Time-Face-Tracking-in-the-Browser-with-Tensor) является статьей из серии статей Фильтры искусственного интеллекта лица в браузере(AI Face Filters in the Browser).

2 февраля 2021

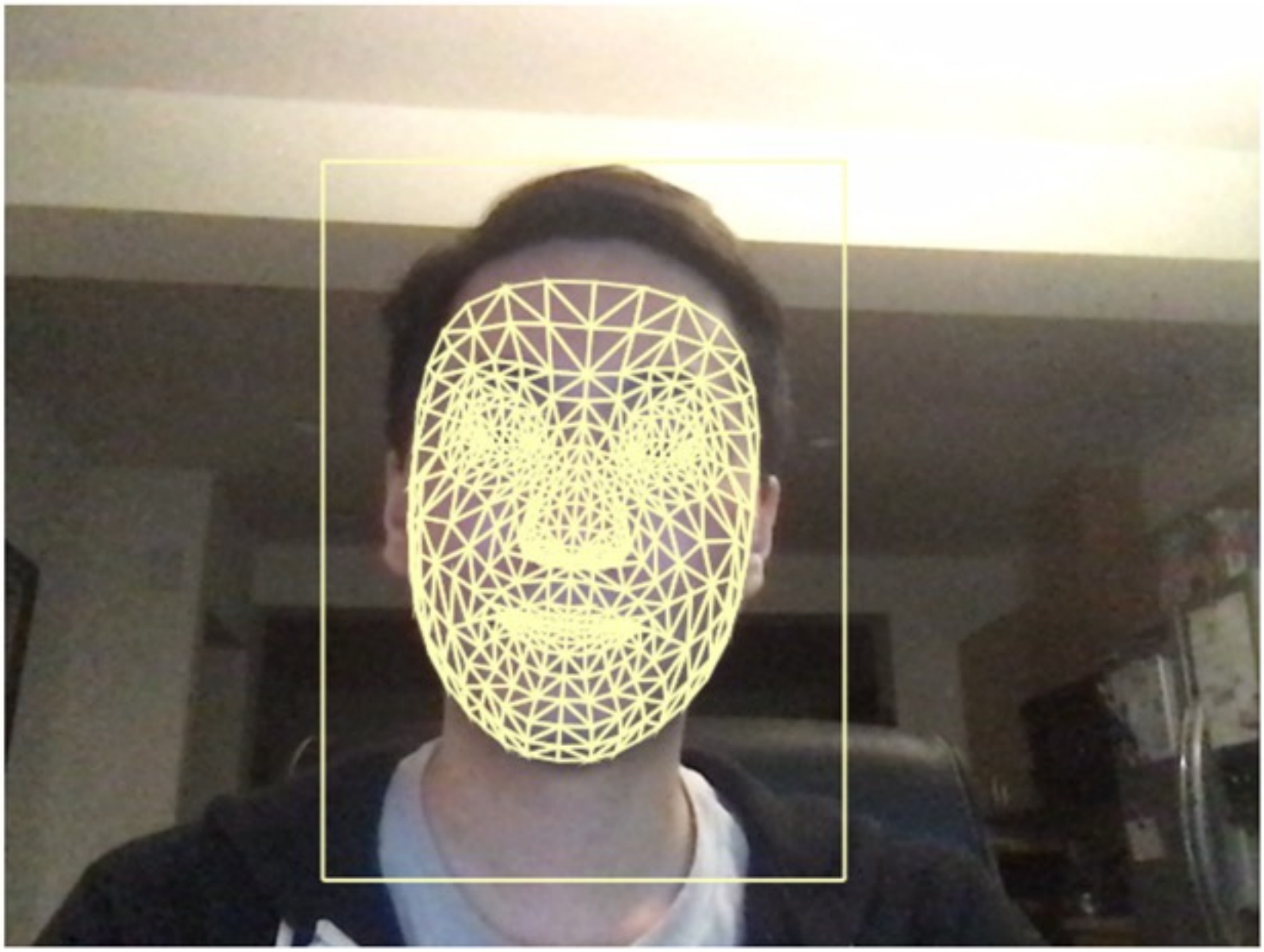
В этой серии статей мы собираемся показать, как в браузере создать фильтры в стиле Snapchat, используя отслеживание лица и библиотеку Tensorflow.js.

Здесь мы изучим, как использовать модели искусственного интеллекта(AI models), чтобы обнаружить форму лиц(shape of faces).

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, используя отслеживание лица и библиотеку Tensorflow.js.



(надпись под снимком экрана: **Уверенность отслеживания лица(Face Tracking Confidence): 1.000**)

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 the код и файлы для этой серии статей.

Предполагается, что вы знакомы с языками **JavaScript** и **HTML** и имеете, по крайней мере, базовое понимание нейронных сетей. 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 Browser Using TensorFlow.js»**;

https://www.codeproject.com/Articles/5272760/Getting-Started-With-Deep-Learning-in-Your-Browser), которая является статьей из серии статей Обнаружение касания лица с помощью Tensorflow.js(Face Touch Detection with Tensorflow.js)

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 TensorFlow.js»; 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.)

Первый шаг к созданию фильтра лица с нуля должен обнаружить лицо и определить местоположение лиц на изображениях и поэтому мы можем начать с него.

Отслеживание лица может быть сделано с помощью фреймворка TensorFlow.js и модели «Обнаружение признаков лица»(Face Landmarks Detection), которая за несколько миллисекунд может дать нам 486 различных признаков лица, в 3D-пространстве, для каждого лица на изображении или в видео-кадре. Особенно значимо то, что модель может выполняться внутри веб-страницы и поэтому вы можете также отследить лица на мобильных устройствах, используя один и тот же код.

Давайте установим и настроим проект по загрузке модели и выполнению отслеживанию лиц в видео-потоке с веб-камеры.

Начальная точка

Вот первоначальный шаблон веб-страницы, которую мы будем использовать для отслеживания лиц.

Этот шаблон включает:

- Библиотеки TensorFlow.js, необходимые для этого проекта

- Набор индексов меша лица(маски лица) в файле **triangles.js**(включен в код проекта)
- **HTML**-элемент **canvas(холст)** для отрендеренного вывода
- Скрытый **HTML**-элемент видео для веб-камеры
- **HTML**-элемент **text(текст)** статуса и вспомогательная функция **setText**
- Вспомогательные функции **drawLine** и **drawTriangle** для **HTML**-элемента **canvas(холст)**

HTML

```
<html>
    <head>
        <title>Отслеживание лица в реальном времени в браузере с помощью библиотеки TensorFlow.js(Real-Time Face Tracking in the Browser with TensorFlow.js)</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-detection@0.0.1/dist/face-landmarks-detection.js"></script>
        <script src="web/triangles.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;
        }

        function drawLine( ctx, x1, y1, x2, y2 ) {
            ctx.beginPath();
            ctx.moveTo( x1, y1 );
            ctx.lineTo( x2, y2 );
            ctx.stroke();
        }
```

```
        function drawTriangle( ctx, x1, y1, x2, y2, x3, y3 ) {
            ctx.beginPath();
            ctx.moveTo( x1, y1 );
            ctx.lineTo( x2, y2 );
            ctx.lineTo( x3, y3 );
            ctx.lineTo( x1, y1 );
            ctx.stroke();
        }

        (async () => {
            // TODO: Добавьте код здесь
        })();
        </script>
    </body>
</html>
```

Использование программного API-интерфейса камеры для разработки HTML5 Webcam API с помощью TensorFlow.js

Запуск веб-камеры в **JavaScript**-коде довольно прост, если у вас есть отрывок кода для этого. Вот служебная функция для установки и настройки веб-камеры и запроса доступа от пользователя:

JavaScript

```
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();
        }
```

```
    });
}
```

Мы можем вызвать эту функцию установки и настройки веб-камеры **setupWebcam** в **асинхронном блоке(async block)** в конце нашего кода и заставить его воспроизводить видео веб-камеры после того, как она загрузится.

JavaScript

```
(async () => {
    await setupWebcam();
    const video = document.getElementById( "webcam" );
    video.play();
})();
```

Затем, давайте установим и настроим **HTML**-элемент **canvas(холст)** для отрендеренного вывода и подготовимся чертить линии и треугольники для ограничивающего прямоугольника и каркас на лице.

Контекст холста будет использован для вывода результатов отслеживания лица и поэтому мы сможем сохранить это глобально за пределами асинхронного блока. Обратите внимание на то, что мы зеркально отразили веб-камеру горизонтально, чтобы вести себя более естественно, подобно реальному зеркалу.

JavaScript

```
let output = null;

(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 камеру
    output.fillStyle = "#fdffb6";
    output.strokeStyle = "#fdffb6";
    output.lineWidth = 2;
})();
```

Давайте отслеживать некоторые лица

Теперь мы готовы! Все, нам осталось загрузить **TensorFlow**-модель **обнаружения признаков лиц(Face Landmarks Detection)** и выполнить ее на кадрах нашей веб-камеры, чтобы показать результаты.

Сначала мы определим глобальную переменную модели **model**, для сохранения загруженной модели:

JavaScript

```
let model = null;
```

Затем в конце асинхронного блока мы можем загрузить модель и установить текст состояния, указывающий то, что наше приложение отслеживания лица готово:

JavaScript

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

setText( "Загружено!/Loaded!" );
```

Теперь давайте создадим функцию отслеживания лица **trackFace**, берущую видеокадры веб-камеры, выполняющую модель отслеживания лица, копирующую изображение веб-камеры в выходной холст, и затем рисующую ограничивающий прямоугольник вокруг лица и рисующую треугольники меша каркаса поверх лица.

JavaScript

```
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
    );

    faces.forEach( face => {
setText( `Face Tracking Confidence:
         ${face.faceInViewConfidence.toFixed( 3 )}` );

// Рисуем ограничивающий прямоугольник вокруг лица
// Draw the bounding box
 const x1 = face.boundingBox.topLeft[ 0 ];
const y1 = face.boundingBox.topLeft[ 1 ];
const x2 = face.boundingBox.bottomRight[ 0 ];
const y2 = face.boundingBox.bottomRight[ 1 ];
const bWidth = x2 - x1;
const bHeight = y2 - y1;
drawLine( output, x1, y1, x2, y1 );
drawLine( output, x2, y1, x2, y2 );
drawLine( output, x1, y2, x2, y2 );
drawLine( output, x1, y1, x1, y2 );

// Рисуем  треугольники меша каркаса поверх лица
// Draw the face mesh
const keypoints = face.scaledMesh;
for( let i = 0; i < FaceTriangles.length / 3; i++ ) {
    let pointA = keypoints[ FaceTriangles[ i * 3 ] ];
    let pointB = keypoints[ FaceTriangles[ i * 3 + 1 ] ];
    let pointC = keypoints[ FaceTriangles[ i * 3 + 2 ] ];
    drawTriangle( output, pointA[ 0 ], pointA[ 1 ], pointB[ 0 ],
                  pointB[ 1 ], pointC[ 0 ], pointC[ 1 ] );
```

```
        }
    });

    requestAnimationFrame( trackFace );
}
```

Наконец, мы можем выбрать первый кадр для отслеживания, вызывая эту функцию в конце нашего асинхронного блока:

JavaScript

```
(async () => {
    ...

    trackFace();
})();
```

Финишная черта

Полный код должен быть похожим на этот код:

HTML

```
<html>
    <head>
        <title>Отслеживание лица в реальном времени в браузере с помощью
библиотеки TensorFlow.js(Real-Time Face Tracking in the Browser with
TensorFlow.js)</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="web/triangles.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;
}

function drawLine( ctx, x1, y1, x2, y2 ) {
    ctx.beginPath();
    ctx.moveTo( x1, y1 );
    ctx.lineTo( x2, y2 );
    ctx.stroke();
}

function drawTriangle( ctx, x1, y1, x2, y2, x3, y3 ) {
    ctx.beginPath();
    ctx.moveTo( x1, y1 );
    ctx.lineTo( x2, y2 );
    ctx.lineTo( x3, y3 );
    ctx.lineTo( x1, y1 );
    ctx.stroke();
}

let output = null;
let model = null;

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();
        }
    });
}

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
        );
         //(Уверенность отслеживания лица)
        faces.forEach( face => {
            setText( `Уверенность отслеживания лица(Face Tracking
Confidence)  ${face.faceInViewConfidence.toFixed( 3 )}` );

            // Рисуем ограничивающий прямоугольник вокруг лица
            // Draw the bounding box
            const x1 = face.boundingBox.topLeft[ 0 ];
            const y1 = face.boundingBox.topLeft[ 1 ];
            const x2 = face.boundingBox.bottomRight[ 0 ];
            const y2 = face.boundingBox.bottomRight[ 1 ];
            const bWidth = x2 - x1;
            const bHeight = y2 - y1;
            drawLine( output, x1, y1, x2, y1 );
            drawLine( output, x2, y1, x2, y2 );
            drawLine( output, x1, y2, x2, y2 );
            drawLine( output, x1, y1, x1, y2 );

            // Рисуем  треугольники меша каркаса поверх лица
            // Draw the face mesh
            const keypoints = face.scaledMesh;
            for( let i = 0; i < FaceTriangles.length / 3; i++ ) {
                let pointA = keypoints[ FaceTriangles[ i * 3 ] ];
                let pointB = keypoints[ FaceTriangles[ i * 3 + 1 ] ];
                let pointC = keypoints[ FaceTriangles[ i * 3 + 2 ] ];
                drawTriangle( output, pointA[ 0 ], pointA[ 1 ],
                              pointB[ 0 ], pointB[ 1 ],
                              pointC[ 0 ], pointC[ 1 ] );
            }
        });

        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 ); // Mirror cam
            output.fillStyle = "#fdffb6";
            output.strokeStyle = "#fdffb6";
            output.lineWidth = 2;

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

            setText( "Загружено!/Loaded!" );

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

Что далее? Можно ли сделать большее при отслеживании лица?

Комбинируя **TensorFlow**-модель **обнаружения признаков лиц(Face Landmarks Detection)** с видео веб-камеры, мы смогли отследить лица в режиме реального времени прямо в браузере. Наш код отслеживания лица также работает на изображениях, и признаки(ключевые точки) могли бы сказать нам больше, чем мы могли бы сначала ожидать. Возможно мы должны попробовать его на наборе данных лиц, таком как **распознавание выражения лица(FER+ Facial Expression Recognition)**?

В следующей статье(next article) этой серии мы будем использовать **«Глубокое обучение на отслеженных лицах у FER + набор данных»(«Deep Learning on the tracked faces of the FER+ dataset»)** и попытаемся точно предсказать эмоцию человека на основе **лицевых точек(facial points)** в браузере с фреймворком **TensorFlow.js**.

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