Липецкий государственный технический университет

Кафедра АСУ

Лабораторная работа №4

по организации графических систем и систем мультимедиа

Обработка потокового виде/аудио

Студент Бровкин А.Н.

Группа М-АС-19

Руководитель Кургасов В.В.

Липецк

2020

**Задание кафедры**

Реализовать программный продукт, в котором есть возможность обработки потокового виде/аудио. Обработка – это внесение изменений, применение фильтров и т.п., а не только визуализация.

# Теоретические сведения

## JavaScript

Приложение было разработано с использованием языка JavaScript.

У браузера есть собственный движок, который иногда называют «виртуальная машина JavaScript», в которой скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся «движком» JavaScript.

Возможности JavaScript сильно зависят от окружения, в котором он работает. Например, Node.JS поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

* Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.
* Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
* Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии AJAX и COMET).
* Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.

Современный JavaScript – стал важным компонентом каждого веб-браузера. JavaScript поддерживает сложные вычисления и конструкции, включая замыкания, анонимные (лямда) функции и даже метапрограммирование[1].

## Оператор Кэнни

Оператор Кэнни (детектор границ Кэнни, алгоритм Кэнни) в дисциплине компьютерного зрения — оператор обнаружения границ изображения. Был разработан в 1986 году Джоном Кэнни (англ. John F. Canny) и использует многоступенчатый алгоритм для обнаружения широкого спектра границ в изображениях.

Кэнни изучил математическую проблему получения фильтра, оптимального по критериям выделения, локализации и минимизации нескольких откликов одного края. Он показал, что искомый фильтр является суммой четырёх экспонент. Он также показал, что этот фильтр может быть хорошо приближен первой производной Гауссианы. Кэнни ввёл понятие подавления не максимумов (англ. Non-Maximum Suppression), которое означает, что пикселями границ объявляются пиксели, в которых достигается локальный максимум градиента в направлении вектора градиента.

Хотя его работа была проведена на заре компьютерного зрения, детектор границ Кэнни до сих пор является одним из лучших детекторов. Кроме особенных частных случаев трудно найти детектор, который бы работал существенно лучше, чем детектор Кэнни [2].

Алгоритм состоит из пяти отдельных шагов:

1. Сглаживание. Размытие изображения для удаления шума.

2. Поиск градиентов. Границы отмечаются там, где градиент изображения приобретает максимальное значение.

3. Подавление не-максимумов. Только локальные максимумы отмечаются как границы.

4. Двойная пороговая фильтрация. Потенциальные границы определяются порогами.

5. Трассировка области неоднозначности. Итоговые границы

# Листинг кода

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <script src=canny.js></script>

        <link rel="stylesheet" href="style.css" type="text/css"/>

    </head>

    <body>

        <nav>

            <label class="but" for="btnFile" >Открыть видеофайл</label>

            <input style="display: none;" type="file" id="btnFile" accept="video/mp4,video/x-m4v,video/\*"/>

            <span class="but" id="wc">Камера</span>

            <span class="but" onclick="setFilter(TypeFilter.NOTHING)">Оригинал</span>

            <span class="but" onclick="setFilter(TypeFilter.CANNY)">Границы</span>

            <span class="but" onclick="setFilter(TypeFilter.NEGATIVE)">Негатив</span>

            <span class="but" onclick="setFilter(TypeFilter.GRAY)">Сделать серым</span>

        </nav>

        <section>

            <div id="ImageContainer">

                <video style="display: none;" id="video" autoplay></video>

                <canvas style="display: none;" id="canvasVideo">

                    Сори canvasVideo не поддерживается :(

                </canvas>

                <canvas style="display: none;" id="canvasCanny" width="0" height="0">

                    Сори canvasVideo не поддерживается :(

                </canvas>

                <canvas id="canvasResult" width="0" height="0">

                    Сори canvasVideo не поддерживается :(

                </canvas>

                <script>

                    const TypeFilter = {

                        GRAY: 1,

                        NEGATIVE: 2,

                        CANNY: 3,

                        NOTHING: 4

                    }

                    //холст для видео, для удобства преобразования

                    var canvasVideo = document.getElementById("canvasVideo");

                    //холст для детектора границ, для хранения всех этапов преобразования

                    var canvasCanny = document.getElementById("canvasCanny");

                    //результирующий холст который видит пользователь

                    var canvasResult = document.getElementById("canvasResult");

                    const ctxCanvasResult = canvasResult.getContext("2d")

                    const ctx = canvasVideo.getContext("2d");

                    var gl = canvasVideo.getContext("webgl", { preserveDrawingBuffer: true }) || canvasVideo.getContext("experimental-webgl", { preserveDrawingBuffer: true });

                    //обработка нажатия кнопки камеры

                    wc.addEventListener("click", function(){

                        //отправляем запрос на получение доступа к камере

                        navigator.mediaDevices.getUserMedia({ video: true }).then((stream) => {

                            //устанавливаем источник видеопотока с камеры

                            video.srcObject = stream;

                        });

                    });

                    var btnFile = document.getElementById("btnFile");

                    //обработка нажатия на кнопку загрузки видеофайла

                    btnFile.addEventListener("change", function () {

                        //если пользователь ничего не выбрал, то выход из функции

                        if (this.files[0] == undefined) return;

                        //убираем звук видео и ставим на циклическое воспроизведение

                        video.muted = video.loop = true;

                        //убираем источник для видео потока (если это была камера)

                        video.srcObject = null;

                        //устанавливаем источник в качестве видеофайла

                        video.src = URL.createObjectURL(this.files[0]);

                        //запускаем видео

                        video.play();

                    });

                    var loadedmetadata = false

                    //обработка события загрузки источника виедо

                    video.addEventListener("loadedmetadata", function (e) {

                        //устанавливаем размеры всех холстов

                        canvasVideo.width = this.videoWidth;

                        canvasVideo.height = this.videoHeight;

                        canvasResult.width = this.videoWidth;

                        canvasResult.height = this.videoHeight;

                        //инициализируем обработчик детектора границ

                        init(canvasCanny, this.videoWidth, this.videoHeight)

                        //устанавливаем флаг что данные загружены

                        loadedmetadata = true

                    }, false);

                    //как только загружена страница запустить функцию рендеринга

                    window.onload = function(){

                        render();

                    }

                    //фильтр для видео, изначально отображается оригинал

                    var typeFilter = TypeFilter.NOTHING

                    //функция рендеринга видео

                    function render(){

                        //если источник для видео загружен, то запускаем рендеринг

                        if (loadedmetadata){

                            //переносим кадр из видео на холст canvasVideo

                            ctx.drawImage(video, 0, 0, canvasVideo.width, canvasVideo.height);

                            //сохраняем данные с холста canvasVideo

                            var imageData = ctx.getImageData(0, 0, canvasVideo.width, canvasVideo.height)

                            switch (typeFilter){

                                //оригинал - просто устанваливаем на результирующий холст данные из кадра видео

                                case TypeFilter.NOTHING: setImageData(imageData); break

                                //границы - запускаем фильтр для отображения границ Canny

                                case TypeFilter.CANNY: setCanny(); break;

                                //негатив - преобразуем оригинальный кадр видео в негатив

                                case TypeFilter.NEGATIVE: setImageData(setNegativeImageData(imageData)); break;

                                //серый - преобразуем оригинальный кадр видео в оттенки серого

                                case TypeFilter.GRAY: setImageData(setGrayImageData(imageData)); break;

                            };

                        }

                        //зацикливаем рандеринг видео

                        requestAnimationFrame(render);

                    }

                    //функция для установки кадра видео в оттенки серого

                    function setGrayImageData(imageData){

                        for (i = 0; i < imageData.data.length; i += 4){

                            let rgb = [imageData.data[i], imageData.data[i+1], imageData.data[i+2]]

                            let avg = (rgb[0] + rgb[1] + rgb[2]) / 3

                            imageData.data[i] = imageData.data[i+1] = imageData.data[i+2] = avg

                        }

                        return imageData

                    }

                    //функция для установки кадра видео в негатив

                    function setNegativeImageData(imageData){

                        for (i = 0; i < imageData.data.length; i += 4){

                            let newRGB = [255 - imageData.data[i], 255 - imageData.data[i+1], 255 - imageData.data[i+2]]

                            imageData.data[i] = newRGB[0]

                            imageData.data[i+1] = newRGB[1]

                            imageData.data[i+2] = newRGB[2]

                        }

                        return imageData

                    }

                    //время в мс по UTC, когда был в последний запуск преобразования для определения границ

                    var lastTimeMils = 0

                    //функция для установки кадра видео в негатив

                    function setCanny(){

                        //полчаем текущее время в мс

                        var cur = new Date().getTime();

                        var delta = cur - lastTimeMils

                        //если прошло время когда предыдущий кадр уже был преобразован,

                        if (delta > timeToResult + 50){

                            //запускаем преобразование операторм Canny

                            canny(canvasVideo, ctx, canvasCanny)

                            //сохраняем время

                            lastTimeMils = cur

                        }

                    }

                    //устанваливаем выбранный фильтр

                    function setFilter(type){

                        typeFilter = type

                    }

                    //устанавливаем данные на реузльтирующий холст

                    function setImageData(imageData){

                        ctxCanvasResult.putImageData(imageData, 0, 0, 0, 0, imageData.width, imageData.height);

                    }

                </script>

            </div>

        </section>

    </body>

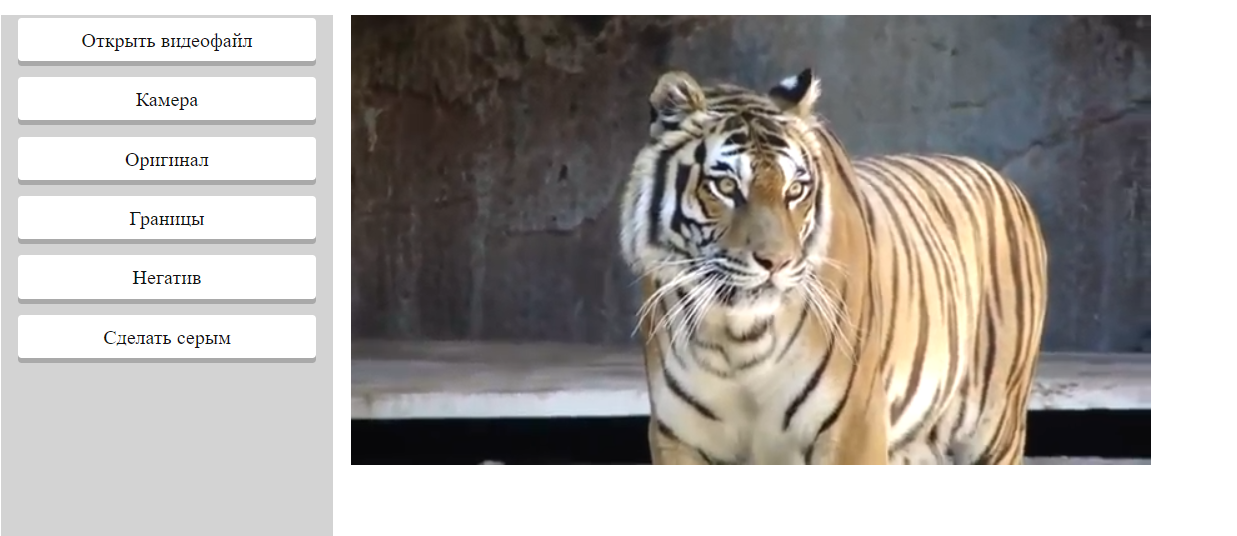
</html>

# Результаты выполнения работы

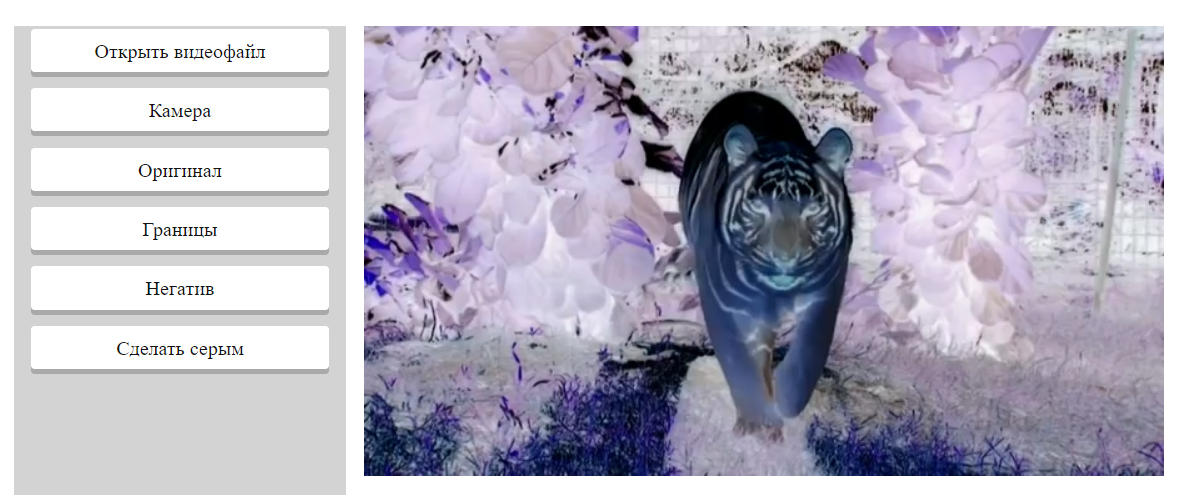
Вся работа в основном выполнена на основе предыдущих работ, где взят детектор границ Canny, а также преобразования изображений в оттенки серого и в негатив. Видеопоток разделялся на кадры каждый из которых преобразовался.

Все фильтры отзывчиво работают за исключением детектора границ Canny, так как обработка одного кадра видео с разрешением 480p занимает 300-600 мс (для ноутбука с процессором AMD Ryzen 5 2500U, видеокартой AMD Radeon 535 2 GB и ОЗУ 4 ГБ), то заметно что не все кадры успевают обрабатываться. Эту задержку можно объяснить сложностью вычислений.

На рисунках 1-2 показаны результаты выполнения работы.



1. Оригинальное видео



1. Видео в негативе



1. Видео с фильтром оттенки серого



1. Видео с фильтром детектора границ Кэнни

**Вывод**

В ходе выполнения данной лабораторной работы были получил практические навыки работы с видеопотоком с использованием JavaScript. Также были закреплены навыки работы с языком JavaScript в связке с языком HTML и языка разметки CSS.

**Список используемой литературы**

1. Закас Н. JavaScript для профессиональных веб разработчиков / [Пер. с англ. А. Лютича]. – СПб.: Питер, 2015. – 960 с.
2. Википедия – Оператор Кэнни [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Оператор\_Кэнни