ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

По дисциплине «Web-программирование»

ВАРИАНТ 4

Выполнили: ст. гр. ТКИ-541

Поткин Дмитрий

Галыба Леонид

Проверил: к.т.н., доц. Сафронов А.И.

Москва 2024

1. Цель работы

Средствами *SVG* или *Canvas* под управлением фреймворка *Vue.js* построить программное, браузерное *web*-обеспечение, реализующее анализ подгруженного изображения (*\*.jpeg*) по варианту.

1. Формулировка задачи

**Реализовать:**

Спроецировать на *Canvas* цветовую инверсию изображения в *RGB*.

1. Технология локального подключения Vue.js

В работе была использована vue.js 3 версии, была подключена сборка vue.js, установленная с источника <https://unpkg.com/vue@3/dist/vue.global.js>.

1. Таблица соотвествия переменных и методов, используемых в web-приложении

|  |  |
| --- | --- |
| Переменная/метод | Описание |
| originalCanvas | Оригинальное изображение |
| invertedCanvas | Инвертированное изображение |
| histogram | Гистограмма инвертированного изображения |
| histogramMode | Режим гистограммы |
| histogramData | Массив, содержащий данные для отрисовки гистограммы (высота столбца, положение столбца, цвет столбца) |
| loadImage | Загрузка изображения, получение его инверсии и гистограммы |
| calculateHistogram | Вычисление данных для построения гистограммы |
| valueData | Массив из 256 элементов для гистограммы яркости (градации серого) |
| colorData.r, colorData.g, colorData.b | Объект с тремя массивами по 256 элементов для каналов R, G, B. |

1. Содержательная часть
   1. Код web-приложения

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Цветовая инверсия и гистограмма</title>

<script src="vue.global.js"></script>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

display: flex;

flex-direction: column;

align-items: center;

background-color: #f5f5f5;

}

.app-container {

border: 2px solid black;

padding: 20px;

background: white;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

max-width: 800px;

}

h2 {

text-align: center;

margin-bottom: 20px;

}

.controls {

margin-bottom: 20px;

}

canvas {

border: 1px solid black;

display: block;

margin: 10px auto;

}

.histogram-container {

margin-top: 20px;

text-align: center;

}

.histogram {

width: 256px; /\* Ширина гистограммы \*/

height: 200px; /\* Высота гистограммы \*/

margin: 10px auto;

border: 1px solid black;

position: relative;

overflow: hidden; /\* Обеспечиваем обрезку графиков, выходящих за рамки \*/

}

.bar {

position: absolute;

bottom: 0;

width: 1px; /\* Фиксированная ширина каждой полоски \*/

}

.histogram-controls {

margin-bottom: 10px;

}

label {

margin-right: 10px;

}

</style>

</head>

<body>

<div id="app" class="app-container">

<h2>Анализ изображения</h2>

<div class="controls">

<input type="file" accept="image/jpeg" @change="loadImage" />

</div>

<canvas ref="originalCanvas" width="600" height="400"></canvas>

<canvas ref="invertedCanvas" width="600" height="400"></canvas>

<div class="histogram-container">

<p>Гистограмма</p>

<div class="histogram-controls">

<label>

<input type="radio" value="value" v-model="histogramMode" /> Значение

</label>

<label>

<input type="radio" value="color" v-model="histogramMode" /> Цвет

</label>

</div>

<div class="histogram" ref="histogram">

<div

v-for="(bar, index) in histogramData"

:key="index"

class="bar"

:style="{

height: bar.height + 'px',

left: bar.left + 'px',

backgroundColor: bar.color,

}"

></div>

</div>

</div>

</div>

<script>

const { createApp, ref, reactive, watch } = Vue;

createApp({

setup() {

const originalCanvas = ref(null);

const invertedCanvas = ref(null);

const histogram = ref(null);

const histogramMode = ref('value'); // Режим гистограммы

const histogramData = reactive([]);

const loadImage = (event) => {

const file = event.target.files[0];

if (!file) return;

const img = new Image();

const reader = new FileReader();

reader.onload = (e) => {

img.src = e.target.result;

};

img.onload = () => {

const ctxOriginal = originalCanvas.value.getContext('2d');

const ctxInverted = invertedCanvas.value.getContext('2d');

// Отрисовка оригинала

ctxOriginal.clearRect(0, 0, originalCanvas.value.width, originalCanvas.value.height);

ctxOriginal.drawImage(img, 0, 0, originalCanvas.value.width, originalCanvas.value.height);

// Извлечение пикселей

const imageData = ctxOriginal.getImageData(0, 0, originalCanvas.value.width, originalCanvas.value.height);

const pixels = imageData.data;

// Применение цветовой инверсии

for (let i = 0; i < pixels.length; i += 4) {

pixels[i] = 255 - pixels[i]; // R

pixels[i + 1] = 255 - pixels[i + 1]; // G

pixels[i + 2] = 255 - pixels[i + 2]; // B

}

// Отрисовка инверсии

ctxInverted.clearRect(0, 0, invertedCanvas.value.width, invertedCanvas.value.height);

ctxInverted.putImageData(imageData, 0, 0);

// Обновление данных для гистограммы

calculateHistogram(imageData.data);

};

reader.readAsDataURL(file);

};

const calculateHistogram = (pixels) => {

const valueData = Array(256).fill(0);

const colorData = { r: Array(256).fill(0), g: Array(256).fill(0), b: Array(256).fill(0) };

for (let i = 0; i < pixels.length; i += 4) {

const r = pixels[i];

const g = pixels[i + 1];

const b = pixels[i + 2];

const gray = Math.round((r + g + b) / 3);

valueData[gray]++;

colorData.r[r]++;

colorData.g[g]++;

colorData.b[b]++;

}

const maxCount = Math.max(

...valueData,

...colorData.r,

...colorData.g,

...colorData.b

);

histogramData.splice(0, histogramData.length);

if (histogramMode.value === 'value') {

valueData.forEach((v, index) => {

histogramData.push({

height: Math.min((v / maxCount) \* 200, 200), // Нормализуем высоту

left: index,

color: 'gray',

});

});

} else if (histogramMode.value === 'color') {

colorData.r.forEach((v, index) => {

histogramData.push({

height: Math.min((v / maxCount) \* 200, 200),

left: index,

color: 'red',

});

});

colorData.g.forEach((v, index) => {

histogramData.push({

height: Math.min((v / maxCount) \* 200, 200),

left: index,

color: 'green',

});

});

colorData.b.forEach((v, index) => {

histogramData.push({

height: Math.min((v / maxCount) \* 200, 200),

left: index,

color: 'blue',

});

});

}

};

watch(histogramMode, () => {

const ctxOriginal = originalCanvas.value.getContext('2d');

const imageData = ctxOriginal.getImageData(0, 0, originalCanvas.value.width, originalCanvas.value.height);

calculateHistogram(imageData.data);

});

return {

originalCanvas,

invertedCanvas,

histogram,

histogramMode,

histogramData,

loadImage,

};

}

}).mount('#app');

</script>

</body>

</html>

* 1. Отображение в браузере

|  |
| --- |
|  |
| 1. – Отображение в браузере |

|  |
| --- |
|  |
| 1. – Отображение в браузере |

* 1. Сеть Петри

|  |
| --- |
|  |
| 1. – Полная сеть Петри |

1. Вывод

В результате проделанной работы было разработано программное, браузерное *web*-обеспечение, реализующее анализ подгруженного изображения средством *Canvas* под управлением фреймворка *Vue.js*.