

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ (МИИТ)**

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

**Отчет по лабораторной работе №6**

**по дисциплине**

«Web программирование»

Вариант 10

**Выполнил:** студенты

группы ТКИ-541

Плуталов Е.А, Мацко А.А.

**Проверил:** доцент кафедры УиЗИ, к.т.н. Сафронов А.И.

**Москва 2024 г.**

Оглавление

[Цель работы 3](#_Toc181981334)

[Формулировка задачи 3](#_Toc181981335)

[Таблица соответствия переменных и методов, используемых в web-приложении 3](#_Toc181981336)

[Содержательная часть по наблюдателям 5](#_Toc181981337)

[Вывод 17](#_Toc181981338)

# Цель работы

Создать веб-приложение для загрузки и анализа изображений в формате JPEG, используя HTML5 Canvas и RGB модель. Приложение обеспечит отображение изображения и возможность инвертирования цветов для анализа цветовых компонентов.

# Формулировка задачи

– локально (LSPWA), не прибегая к инструментарию Node.js и npm (Node Package Manager);

* **Задание :**

Спроецировать на SVG цветовую инверсию изображения в CMYK.

# Таблица соответствия переменных и методов, используемых в web-приложении

**Задание :**

|  |  |  |
| --- | --- | --- |
| **Переменная** | **Описание** | **Тип данных** |
| imageSrc | URL исходного изображения, загруженного пользователем. | string |
| invertedImageSrc | URL инвертированного изображения, созданного на основе imageSrc. | string |
| histogramData | Объект, содержащий массивы для хранения данных гистограммы для каждого цветового канала. | object |
| histogramData.Brightness | Массив, содержащий количество пикселей для каждого значения яркости. | array |
| histogramData.C | Массив, содержащий количество пикселей для канала Cyan (циан) в модели CMYK. | array |
| histogramData.M | Массив, содержащий количество пикселей для канала Magenta (маджента) в модели CMYK. | array |
| histogramData.Y | Массив, содержащий количество пикселей для канала Yellow (желтый) в модели CMYK. | array |
| histogramData.K | Массив, содержащий количество пикселей для канала Black (черный) в модели CMYK. | array |
| histogramHeight | Высота гистограммы, используемая для масштабирования значений (уменьшенная версия — 300 пикселей). | number |
| histogramScale | Масштаб для отображения гистограммы, рассчитанный на основе максимального значения данных. | number |
| histogramMode | Режим отображения гистограммы: яркость (brightness) или цветовые каналы CMYK (cmyk). | string |

# Содержательная часть по наблюдателям

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Уменьшенная гистограмма</title>  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>  <style>  svg {  border: solid 1px #999;  }  p {  margin: 5px 0 3px;  }  .card {  margin: 15px 5px;  }  .histogram {  margin-top: 20px;  border: 1px solid #999;  }  </style>  </head>  <body>  <div id="app"></div>  <script>  const { createApp, h } = Vue;  const App = {  data() {  return {  imageSrc: null,  processedImage: null,  imageWidth: 0,  imageHeight: 0,  histogramData: {  Brightness: [],  C: [],  M: [],  Y: [],  K: [],  },  histogramScale: 1,  histogramHeight: 300, // Уменьшенная высота гистограммы  histogramMode: "brightness",  };  },  methods: {  handleFileChange(event) {  const file = event.target.files[0];  if (file && file.type.startsWith("image/")) {  const reader = new FileReader();  reader.onload = (e) => {  this.imageSrc = e.target.result;  };  reader.readAsDataURL(file);  } else {  alert("Пожалуйста, выберите файл изображения.");  }  },  processImage(event) {  const img = event.target;  const canvas = document.createElement("canvas");  const ctx = canvas.getContext("2d");  canvas.width = img.width;  canvas.height = img.height;  this.imageWidth = img.width;  this.imageHeight = img.height;  ctx.drawImage(img, 0, 0);  const imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);  const data = imageData.data;  const histogramBrightness = new Array(256).fill(0);  const histogramC = new Array(256).fill(0);  const histogramM = new Array(256).fill(0);  const histogramY = new Array(256).fill(0);  const histogramK = new Array(256).fill(0);  for (let i = 0; i < data.length; i += 4) {  const r = data[i] / 255;  const g = data[i + 1] / 255;  const b = data[i + 2] / 255;  const k = 1 - Math.max(r, g, b);  const c = (1 - r - k) / (1 - k || 1);  const m = (1 - g - k) / (1 - k || 1);  const y = (1 - b - k) / (1 - k || 1);  const cInv = 1 - c;  const mInv = 1 - m;  const yInv = 1 - y;  const kInv = 1 - k;  data[i] = 255 \* (1 - cInv) \* (1 - kInv);  data[i + 1] = 255 \* (1 - mInv) \* (1 - kInv);  data[i + 2] = 255 \* (1 - yInv) \* (1 - kInv);  histogramC[Math.round(cInv \* 255)]++;  histogramM[Math.round(mInv \* 255)]++;  histogramY[Math.round(yInv \* 255)]++;  histogramK[Math.round(kInv \* 255)]++;  const brightness = Math.round((cInv + mInv + yInv + kInv) / 4 \* 255);  histogramBrightness[brightness]++;  }  ctx.putImageData(imageData, 0, 0);  this.processedImage = canvas.toDataURL();  const max = Math.max(  ...histogramBrightness,  ...histogramC,  ...histogramM,  ...histogramY,  ...histogramK  );  this.histogramData = {  Brightness: histogramBrightness,  C: histogramC,  M: histogramM,  Y: histogramY,  K: histogramK,  };  this.histogramScale = max > 0 ? this.histogramHeight / max : 1;  },  updateHistogram() {},  },  render() {  const fileInput = h("input", {  id: "file",  type: "file",  onChange: this.handleFileChange,  });  const imageElement =  this.imageSrc &&  h("img", {  src: this.imageSrc,  alt: "Загруженное изображение",  onLoad: this.processImage,  });  const processedImage =  this.processedImage &&  h(  "svg",  { width: this.imageWidth, height: this.imageHeight, style: "border: solid 1px #999;" },  [  h("image", {  href: this.processedImage,  width: this.imageWidth,  height: this.imageHeight,  }),  ]  );  const brightnessHistogram =  this.histogramMode === "brightness" &&  h(  "svg",  { class: "histogram", width: 512, height: this.histogramHeight },  this.histogramData.Brightness.map((value, index) =>  h("rect", {  key: `b-${index}`,  x: index \* 2,  y: this.histogramHeight - value \* this.histogramScale,  width: 2,  height: value \* this.histogramScale,  fill: "gray",  })  )  );  const cmykHistogram =  this.histogramMode === "cmyk" &&  h(  "svg",  { class: "histogram", width: 512, height: this.histogramHeight },  ["C", "M", "Y", "K"].flatMap((channel, i) =>  this.histogramData[channel].map((value, index) =>  h("rect", {  key: `${channel.toLowerCase()}-${index}`,  x: index \* 2,  y: this.histogramHeight - value \* this.histogramScale,  width: 2,  height: value \* this.histogramScale,  fill:  channel === "C"  ? "cyan"  : channel === "M"  ? "magenta"  : channel === "Y"  ? "yellow"  : "black",  opacity: channel === "K" ? 0.7 : 1,  })  )  )  );  const histogramControls = h("div", [  h("input", {  id: "brightness",  type: "radio",  value: "brightness",  checked: this.histogramMode === "brightness",  onChange: () => (this.histogramMode = "brightness"),  }),  h("label", { for: "brightness" }, "Яркость"),  h("input", {  id: "cmyk",  type: "radio",  value: "cmyk",  checked: this.histogramMode === "cmyk",  onChange: () => (this.histogramMode = "cmyk"),  }),  h("label", { for: "cmyk" }, "Цвет (CMYK)"),  ]);  return h("div", { id: "hist" }, [  h("div", { class: "card" }, [  h("p", {}, "Выберите изображение из файла"),  fileInput,  h("br"),  imageElement,  ]),  h("div", { class: "card" }, [  h("p", {}, "Результат инверсии (CMYK)"),  processedImage,  ]),  h("div", { class: "card" }, [  h("p", {}, "Режим отображения гистограммы:"),  histogramControls,  h("p", {}, "Гистограмма:"),  brightnessHistogram || cmykHistogram,  ]),  ]);  },  };  createApp(App).mount("#app");  </script>  </body>  </html> |
| Код программы |

|  |
| --- |
|  |
| Рисунок 1 - Отображение в браузере задание |
|  |
| Рисунок 2 - Отображение в браузере исходной картинки |
|  |
| Рисунок 3 - Отображение в браузере рисунка в изменённом состоянии |
| Рисунок 4 – Гистограмма яркости |
| Рисунок 6 - Гистограмма цвета |
| Рисунок 7 - Сеть Петри к заданию |

# Вывод

Веб-приложение для анализа изображений успешно обеспечивает загрузку, отображение и инверсию цветов в формате JPEG, используя технологии HTML5 SVG и Vue.js. Функционал приложения позволяет детально анализировать цветовые компоненты изображения, делая его полезным инструментом для визуального контента.