ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент, канд. техн. наук		А. В. Аграновский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

Вариант №1

по курсу: Компьютерная Графика

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4321		К. А. Лебедев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1. Цель работы

Изучение артефактов, возникающих при аффинных преобразованиях.

2. Формулировка задания

1. Прочитать методические указания.

2. Перевести представленную формулу Оуэна и Македона в однородные координаты.

3. Выбрать для последующего преобразования изображение, использование которого не имеет ограничений, связанных с авторским правом. Рекомендуется использовать изображения, позволяющие увидеть артефакты, возникающие при осуществлении поворота.

4. Выбрать вариант задания в соответствии со своим порядковым номером в списке группы: осуществить поворот изображения на угол в градусах, равный сумме цифр порядкового номера; затем на данный угол, умноженный на три и исходный угол, умноженный на пять. Если артефакты аффинного поворота не проявляются, измените углы поворота, умножая исходный угол не на 3 и 5, а на 0,3 и 0,5 соответственно.

5. Написать программу на любом языке высокого уровня, реализующую преобразование изображения в соответствии со своим вариантом задания с использованием матрицы аффинного преобразования (поворота) и комбинации матриц Оуэна и Македона.

6. Ввод исходных и вывод преобразованных изображений производить в одинаковом формате (предпочтительно, BMP).

3. Теоретические сведения

Аффинные преобразования представляют собой набор операций, позволяющих изменять геометрическую форму изображения с помощью линейных преобразований координат, таких как масштабирование, сдвиг, поворот, отражение и наклон. Эти преобразования находят широкое применение в различных областях, включая компьютерную графику, обработку изображений, медицинскую визуализацию и робототехнику.

Тем не менее, применение аффинных преобразований к растровым изображениям сопряжено с рядом сложностей, связанных с дискретной природой пиксельных данных. Одной из основных проблем является неопределенность пикселей. При выполнении аффинного преобразования новые координаты пикселей вычисляются с использованием математических операций, часто приводящих к дробным значениям. Эти координаты округляются до ближайших целых чисел, соответствующих дискретной сетке пикселей. Округление может привести к образованию "дыр" — пустых областей в результирующем изображении, которые не получают цветового значения из-за отсутствия соответствующих пикселей в исходном изображении. Обычно такие "дырки" заполняются фоновым цветом (чаще всего белым или черным), что может визуально проявляться в виде дефектов на преобразованном изображении, особенно на светлых или однотонных участках.

Кроме того, проблема потери данных также является значительной. При обратных преобразованиях исходный пиксель может не быть восстановленным, так как его координаты в результирующем изображении могут не совпадать с координатами, полученными после округления. Это приводит к искажению изображения и снижению его точности. Например, в приложениях, связанных с медицинской визуализацией, такие искажения могут затруднить диагностику, так как детали, имеющие критическое значение, могут быть потеряны.

Дополнительно возникает эффект "лесенки", особенно при вращении или наклоне изображений. Он проявляется в виде ступенчатых контуров, что обусловлено ограниченным разрешением растровых изображений и резкими переходами между пикселями. Этот эффект особенно заметен при работе с изображениями, содержащими высококонтрастные границы, например, в архитектурной визуализации или при обработке текстур.

Для решения этих проблем разработаны различные методы улучшения качества растровых изображений при выполнении аффинных преобразований. Одним из наиболее эффективных подходов является алгоритм Оуэна и Македона, который был предложен в 1987 году. Этот метод основан на разложении матрицы поворота на три последовательных этапа, каждый из которых представляет собой сдвиг: сначала сдвигаются строки изображения, затем выполняется основной поворот, и в завершение производится корректирующий сдвиг строк. Благодаря такому подходу преобразование становится более точным и не требует интерполяции для заполнения "дыр". Алгоритм минимизирует артефакты, поскольку каждая операция сдвига сохраняет целостность структуры пикселей и не создает неопределенных областей.

Метод Оуэна и Македона зарекомендовал себя как надежный инструмент для обработки изображений с минимальными потерями качества. Его применение позволяет значительно улучшить визуальные результаты аффинных преобразований, особенно в тех случаях, когда требуется высокая точность или исходное изображение содержит тонкие детали, где артефакты становятся особенно заметными. В результате, использование данного алгоритма открывает новые возможности для повышения качества обработки изображений в таких областях, как компьютерная графика, медицинская визуализация и цифровая фотография.

Дополнительно, современные методы интерполяции, такие как билинейная и бикубическая интерполяция, также используются для улучшения качества аффинных преобразований. Билинейная интерполяция

учитывает значения соседних пикселей, что позволяет сгладить переходы и уменьшить эффект "лесенки". Бикубическая интерполяция, в свою очередь, использует 16 соседних пикселей для вычисления нового значения, что обеспечивает еще более высокое качество изображения. Однако эти методы могут быть более ресурсоемкими и требуют дополнительных вычислительных затрат.

Таким образом, аффинные преобразования остаются важным инструментом в цифровой обработке изображений, и их эффективность продолжает улучшаться благодаря новым алгоритмам и методам интерполяции, что позволяет справляться с возникающими проблемами и повышать качество визуализации в различных приложениях.

4. Описание алгоритма решения поставленной задачи

Решение поставленной задачи включает несколько этапов, направленных на выполнение аффинного преобразования изображения, в частности, на поворот с использованием алгоритма Оуэна и Македона. Основная цель данного алгоритма заключается в минимизации артефактов, таких как неопределенные пиксели или "дырки", которые могут возникать в процессе поворота.

Для начала исходное изображение загружается в формате BMP или создается искусственно для демонстрации. Чтобы избежать обрезки изображения во время поворота, его предварительно увеличивают, добавляя рамку, что позволяет сохранить видимость всего содержимого.

В соответствии с заданием определяются три угла поворота: исходный угол, который рассчитывается как сумма цифр порядкового номера, а также два дополнительных угла, получаемых путем умножения исходного угла на три и на пять. В случае, если артефакты недостаточно выражены, углы могут быть уменьшены в 10 раз для более плавного поворота.

Алгоритм выполняет поворот через последовательность трех операций:

1. **Первый сдвиг строк**: на этом этапе используется матрица, которая сдвигает строки изображения на величину, пропорциональную половинному углу поворота. Это подготавливает изображение к следующему этапу.
2. **Поворот с использованием матрицы аффинного преобразования**: применяется стандартная матрица поворота, которая осуществляет поворот изображения на заданный угол.
3. **Обратный сдвиг строк**: этот шаг корректирует эффект первого сдвига, возвращая строки изображения в их исходное положение.

Для каждого пикселя исходного изображения вычисляются новые координаты путем умножения их на текущую матрицу преобразования. Если новые координаты выходят за пределы изображения, такие пиксели

игнорируются, что помогает избежать потерь информации и поддерживать целостность изображения.

Преобразованное изображение сохраняется в формате BMP. Итоговые результаты демонстрируют влияние поворота на углы 1°, 3° и 5°, а также визуализируют возможные артефакты, возникающие в процессе преобразования. Эти изображения могут быть полезны для анализа качества алгоритма и его способности минимизировать искажения, что особенно важно в приложениях, требующих высокой точности, таких как медицинская визуализация или графический дизайн.

5. Выбор языка программирования

Для реализации программы построения фигур и аффинных преобразований был использован язык программирования JavaScript, а также следующие API. Язык программирования JavaScript JavaScript был выбран в качестве основного языка программирования для реализации задачи ввиду его расширяемости, читабельности и мультипарадигменности его предметной области.

Браузерный Javascript поддерживает однопоточность и предоставляет API для работы с графикой в браузере (Canvas API), что делает его удобным инструментом для построения сложных математических структур, таких как аффинные преобразования.

Используемые API Canvas API — это API для двумерного рисования. Он позволяет рисовать линии, фигуры, изображения и текст прямо в браузере без использования плагинов, таких как Flash или Java. Canvas изначально был создан Apple для своих виджетов, но с тех пор был принят всеми разработчиками основных браузеров и теперь является частью спецификации HTML5. Browser API — это интерфейс прикладного программирования для веб-сервера или веб-браузера. Это концепция веб-разработки, обычно ограниченная клиентской стороной веб-приложения (включая любые используемые веб-фреймворки), и поэтому обычно не включает детали реализации веб-сервера или браузера, такие как SAPI или API, если они не доступны для общего доступа через удаленное веб-приложение.

Используемые интерфейсы document: используется для манипуляций с DOM (древовидная структура элементов веб страницы).

6. Описание разработанной программы.

Разработанная программа предназначена для отображения поворота изображения на заданные углы с использованием Canvas API и алгоритма Оуэна и Македона в сравнении. Алгоритм минимизирует артефакты, возникающие при стандартных аффинных преобразованиях, таких как "дырки" (пустые пиксели), которые появляются из-за округления координат при вычислении новых позиций пикселей.

Программа реализована на языке JavaScript с использованием Canvas API.

Программа начинает свою работу с загрузки файла стилизации и поиска root элемента для дальнейшей работы с ним. Затем происходит вычисление параметров варианта студента и создается функция для создания канваса. По загрузке картинки отрабатывает поворот с помощью Canvas API и предложенного алгоритма в задании, используя аффинные преобразования.

Алгоритм Оуэна и Македона включает три последовательных этапа преобразования. На первом этапе осуществляется сдвиг строк изображения на величины, пропорциональные половинному углу поворота. Этот шаг подготавливает изображение к следующему повороту. Второй этап включает стандартный поворот изображения с применением аффинной матрицы, рассчитанной на заданный угол. На третьем этапе выполняется обратный сдвиг строк, что позволяет компенсировать первый шаг и восстановить исходное положение строк относительно результирующего изображения. Эти три шага помогают избежать образования "дырок", которые часто возникают при традиционных аффинных преобразованиях.

Процесс преобразования основывается на пересчете координат каждого пикселя изображения. Для каждой точки исходного изображения вычисляются новые координаты с использованием матрицы преобразования. Если новые координаты выходят за пределы результирующего изображения, такие пиксели игнорируются. Это обеспечивает сохранение целостности изображения и предотвращает ненужные вычисления.

В результате программа выполняет поворот изображения на три заданных угла (1°, 3° и 5°). Итоговые изображения сохраняются в формате BMP, что обеспечивает их сохранение без потерь. Код является расширяемым и достаточным, что особенно актуально в разработке точных вычислений.

7. Результат работы программы

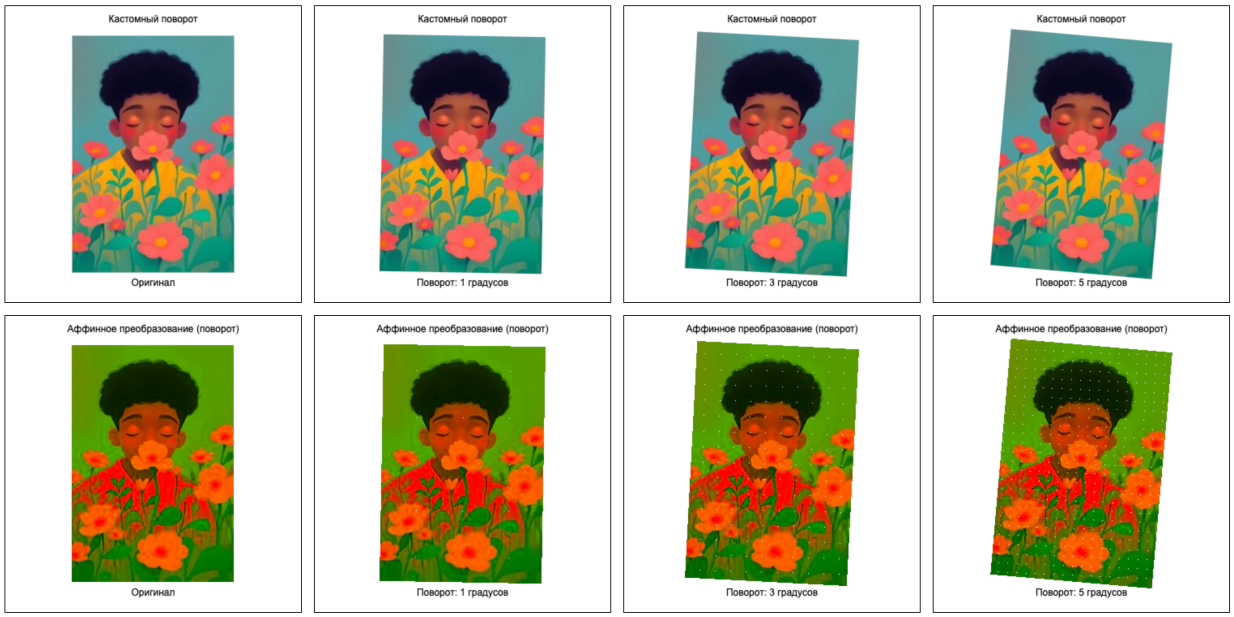

Рисунок 1 – скриншот работы программы

8. Вывод

В результате выполнения алгоритма аффинного преобразования с использованием поворота изображения и алгоритмов Оуэна и Македона, реализованного в предоставленном коде, достигается эффективное визуальное представление различных углов поворота (1, 3, 5), что позволяет продемонстрировать влияние аффинных преобразований на исходное изображение. Код создает несколько канвасов, на которых отображаются как оригинальное изображение, так и его версии, повернутые на заданные углы, основанные на сумме цифр порядкового номера задания. Версии алгоритмов предполагают сравнение решений объектной задачи разными способами и инструментами. Используя Canvas API в языке JavaScript, получилось добиться лучшего результата при повороте изображения.

В лабораторной работе было выяснено как использовать манипуляции с пикселями для создания новых изображений с учетом изменений в их координатах и рассмотрены различные методы и подходу к решению задачи. В результате, итоговые изображения позволяют визуализировать результаты аффинных преобразований и оценить их влияние на качество и целостность изображения.

Таким образом, реализованный алгоритм показывает свою эффективность в минимизации артефактов и качественном отображении повернутых изображений, но стоит заметить, что поиск альтернативных и готовых решений может облегчить выполнение задачи и увеличить эффективность результата, что может сэкономить средства компании заказчика, что приветствуется в продуктовой разработке.

Список использованных источников

1. Owen Ch.B., Makedon F. High quality alias free image rotation // Proceedings of the 30th Asilomar Conference on Signals, Systems, and Computers. Pacific Grove, California, November 2–6, 1996. С. 1523–1526. URL: https://digitalcommons.dartmouth.edu/cgi/viewcontent.cgi?article=5030&context=facoa (дата обращения: 29.11.2024).

2. Аграновский А.В. Исследование артефактов аффинных преобразований: методические указания. Санкт-Петербург: СПбГУАП, 2024. (дата обращения: 30.11.2024).

3. Левин В.Л., Фомин С.В. Аффинные преобразования и их применение в обработке изображений. // Прикладная информатика. 2021. Т. 27. № 3. С. 54–67. (дата обращения: 1.12.2024).

4. Современный учебник JavaScript, последнее обновление: 25 ноября 2024 г. URL: https://learn.javascript.ru (дата обращения: 29.11.2024).

Листинг программы

```
import "./style.css";

const app = document.querySelector("#app");

app.classList.add("container");

const IMG_WIDTH = 200;
const IMG_HEIGHT = 292;

const TASK_VARIANT = 10;

const variantRatio = new String(TASK_VARIANT)
  .split("")
  .reduce((a, b) => Number(a) + Number(b));

const variantArray = [
  variantRatio * 0,
  variantRatio * 1,
  variantRatio * 3,
  variantRatio * 5,
];

const createCanvas = () => {
  const canvas = document.createElement("canvas");
  canvas.width = IMG_HEIGHT * 1.25;
  canvas.height = IMG_HEIGHT * 1.25;
  canvas.classList.add("canvas");

  return canvas;
```

```
};

const image = document.createElement("img");
image.src = "image.bmp";
image.width = IMG_WIDTH;
image.height = IMG_HEIGHT;

image.onload = function () {
  variantArray.forEach((ratio) => {
    const canvas = createCanvas();
    const ctx = canvas.getContext("2d");

    app.insertAdjacentElement("beforeend", canvas);

    const angle = (ratio * Math.PI) / 180;

    ctx.save();
    ctx.translate(canvas.width / 2, canvas.height / 2);
    ctx.rotate(angle);
    ctx.drawImage(image, -image.width / 2, -image.height / 2);
    ctx.restore();

    ctx.fillStyle = "black";
    ctx.font = "12px Arial";
    ctx.textAlign = "center";
    ctx.fillText(
      ratio ? `Поворот: ${ratio} градусов` : "Оригинал",
      canvas.width / 2,
      canvas.height - 20
    );
```

```
  ctx.fillText("Кастомный поворот", canvas.width / 2, 20);
});

variantArray.forEach((ratio) => {
  const canvas = createCanvas();
  const ctx = canvas.getContext("2d");

  app.insertAdjacentElement("beforeend", canvas);

  const angle = (ratio * Math.PI) / 180;
  const width = image.width;
  const height = image.height;

  ctx.drawImage(image, 0, 0);

  const imageData = ctx.getImageData(0, 0, width, height);
  const data = imageData.data;

  const newWidth = Math.round(
    Math.abs(width * Math.cos(angle)) + Math.abs(height * Math.sin(angle))
  );
  const newHeight = Math.round(
    Math.abs(width * Math.sin(angle)) + Math.abs(height * Math.cos(angle))
  );
  const newImageData = ctx.createImageData(newWidth, newHeight);
  const newData = newImageData.data;

  const centerX = width / 2;
  const centerY = height / 2;
```

```
        const newCenterX = newWidth / 2;
        const newCenterY = newHeight / 2;

        for (let y = 0; y < height; y++) {
          for (let x = 0; x < width; x++) {
            const pixelIndex = (y * width + x) * 4;
            const r = data[pixelIndex];
            const g = data[pixelIndex + 1];
            const b = data[pixelIndex + 2];
            const a = data[pixelIndex + 3];

            const newX = Math.round(
              (x - centerX) * Math.cos(angle) -
                (y - centerY) * Math.sin(angle) +
                newCenterX
            );
            const newY = Math.round(
              (x - centerX) * Math.sin(angle) +
                (y - centerY) * Math.cos(angle) +
                newCenterY
            );

            if (newX >= 0 && newX < newWidth && newY >= 0 && newY <
newHeight) {
              const newPixelIndex = (newY * newWidth + newX) * 4;
              newData[newPixelIndex] = r;
              newData[newPixelIndex + 1] = g;
              newData[newPixelIndex + 1] = b;
              newData[newPixelIndex + 3] = a;
```

```
        }
      }
    }

    ctx.clearRect(0, 0, canvas.width, canvas.height);

    ctx.putImageData(
      newImageData,
      (canvas.width - newWidth) / 2,
      (canvas.height - newHeight) / 2
    );

    ctx.fillStyle = "black";
    ctx.font = "12px Arial";
    ctx.textAlign = "center";
    ctx.fillText(
      ratio ? `Поворот: ${ratio} градусов` : "Оригинал",
      canvas.width / 2,
      canvas.height - 20
    );

    ctx.fillText("Аффинное преобразование (поворот)", canvas.width / 2,
20);
  });
};
```