ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ПРЕПОДАВАТЕЛЬ

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

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1 |
| ИССЛЕДОВАНИЕ ПОЛУПРОВОДНИКОВОГО ДИОДА |
| по курсу: ЭЛЕКТРОНИКА И СХЕМОТЕХНИКА |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4329 |  |  |  | Д.С. Шаповалова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

Содержание

[1. Цель работы: 3](#_Toc183989581)

[2. Задание: 3](#_Toc183989582)

[3. Теоретические сведения: 3](#_Toc183989583)

[3.1 Аффинные преобразования 3](#_Toc183989584)

[3.2. Композиция аффинных преобразований на плоскости 5](#_Toc183989585)

[3.3. Методы улучшения растровых изображений при реализации аффинных преобразований 5](#_Toc183989586)

[3.3 Однородные координаты 6](#_Toc183989587)

[4. Описание алгоритма решения поставленной задачи 8](#_Toc183989588)

[5. Выбор языка программирования и библиотек: 9](#_Toc183989589)

[6. Скриншоты, иллюстрирующие результаты работы программы: 10](#_Toc183989590)

[7. Вывод: 17](#_Toc183989591)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 19](#_Toc183989592)

[ПРИЛОЖЕНИЕ А 20](#_Toc183989593)

# 1. Цель работы:

Изучение и практическое исследование работы и характеристик полупроводникового диода.

# 2. Задание:

ТЕКСТ

# 3. Теоретические сведения:

## 3.1 Аффинные преобразования

ТЕКСТ

## 3.2. Композиция аффинных преобразований на плоскости

ТЕКСТ

## 3.3. Методы улучшения растровых изображений при реализации аффинных преобразований

ТЕКСТ

ЧТО ЗА ФОРМУЛА:

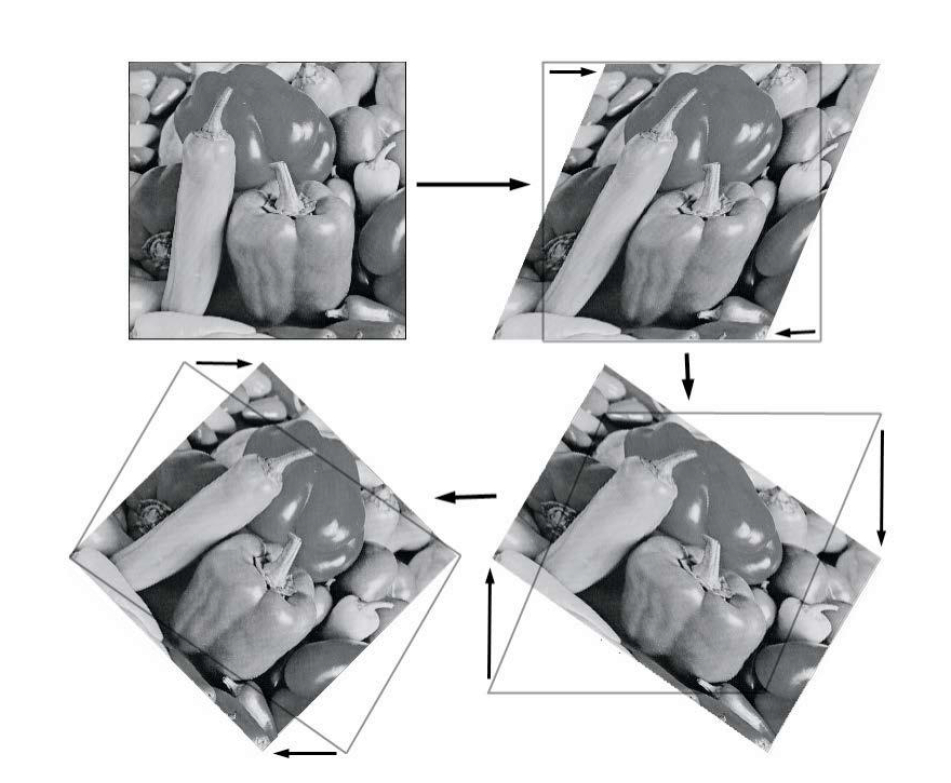


Рисунок 2 – ТЕКСТ

## 3.3 Однородные координаты

ТЕКСТ

# 4. Описание алгоритма решения поставленной задачи

ТЕКСТ

# 5. Выбор языка программирования и библиотек:

ТЕКСТ

# 6. Скриншоты, иллюстрирующие результаты работы программы:



Рисунок 3.1 – Исходное изображение

# 7. Вывод:

В данной работе мы подробно рассмотрели поворот изображения с помощью аффинных преобразований, а также артефакты, которые появляются в процессе. В ходе выполнения лабораторной мы изучили алгоритм Оуэна и Македона, который путём трёх последовательных сдвигов строк, столбцов и затем снова строк картинки позволяет повернуть изображение без артефактов.

В результате выполнения лабораторной работы удалось достигнуть поставленной цели – изучить артефакты, возникающие при аффинных преобразованиях.

В процессе работы с изображениями, преобразованными с использованием матрицы вращения, становится заметным появление пробелов — пикселей, цвет которых совпадает с цветом фона. Появление этих артефактов связано с тем, что после преобразования координаты точек изображения могут стать нецелыми, что не соответствует целочисленным координатам пикселей в сетке.

Из результатов работы программы становится очевидным, что с увеличением угла поворота количество неопределённых пикселей также увеличивается. Это связано с тем, что при увеличении угла увеличивается смещение пикселей, что повышает вероятность появления зазоров.

Один из способов решения этой проблемы — заполнение пробелов цветом, рассчитанным на основе соседних пикселей. Минус подхода в том, что это может привести к погрешностям или эффекту зазубренности вдоль краёв изображения, а мы ведь максимально стараемся избегать всяких артефактов.

В данной работе рассматривается эффективный метод, позволяющий устранить подобные искажения — алгоритм Оуэна и Македона, основанный на разбиении матрицы вращения на произведение трёх матриц сдвигов.

Последовательное умножение на эти матрицы приводит к искажению вдоль осей: сначала вдоль оси Ox, затем вдоль оси Oy, и снова вдоль Ox. Такой подход позволяет выполнять все преобразования в рамках сетки пикселей, что значительно снижает вероятность возникновения неопределённости в координатах.

По результатам работы программы можно заметить, что метод правда эффективен. Кроме того, угол поворота изображения оказался идентичен углу поворота изображения при использовании стандартной матрицы вращения. Свойства аффинных преобразований дают понять: последовательность нескольких аффинных операций равна одному аффинному преобразованию.

Алгоритм был изучен на практике, благодаря написанию программы, которая использует статические массивы в качестве матриц аффинного преобразования.

В ходе лабораторной были получены практические навыки работы с библиотеками языка Python – Pillow, Math и Numpy. Основные функции, с которыми происходила работа:

Image.open() – открывает изображение по заданному пути;

.save() – сохраняет изображение по заданному пути;

.putpixel() – позволяет изменять цвет пикселя. Для этого нужно указать координаты пикселя (в виде кортежа из двух элементов) и цвет;

math.radians() – конвертирует заданный угол из градусов в радианы.

В ходе работы возникли проблемы. Одна из основных – неконтролируемый поворот изображения, то есть не относительно центра, из-за чего картинка оказывалась непонятно где и была странно обрезана. Решением стало перемещение центра изображения в начало координат, которое было упущено по невнимательности.

Также присутствует проблема некоторой медлительности программы из-за использования метода Image.putpixel(), так что для более масштабных изображений лучше использовать другие функции, например ImageDraw. В ходе лабораторной работы скорость полученной программы меня вполне устраивает.

Таким образом, в процессе лабораторной работы был изучен и применён на практике алгоритм предотвращения артефактов, который может пригодится в дальнейшем для решения других задач в сфере информационных технологий, науке о данных и других областях, где используется деформация изображений или вычислительное моделирование.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Что за зверь — аффинные преобразования? – URL:  <https://habr.com/ru/articles/539420/> (дата обращения 28.09.2024)
2. Д. Роджерс – Математические основы машинной графики / Д. Роджерс, Дж. Адамс: Пер. с англ. – П.А. Монахова Г.В. Олохтоновой, Д.В. Волкова – М.: Мир, 2001. – 604с., ил.
3. Библиотека Pillow в Python – URL: <https://docs-python.ru/packages/biblioteka-pillow-python/> (дата обращения 01.12.2024)
4. Morgan D. Цифровая обработка сигналов и изображений – URL: <https://www.bsu.by/upload/page/353593.pdf> (дата обращения 27.09.2024)
5. Выжимка из документации Numpy: функции, методы и примеры. – URL: <https://pythonru.com/biblioteki/rukovodstvo-po-ispolzovaniju-python-biblioteki-numpy> (дата обращения 01.12.2024)
6. Owen Ch.B., Makedon F. High quality alias free image rotation // Proceeding of 30th Asilomar Conference on Signals, Systems, and Computers Pacific Grove, California, November 2—6, 1996. URL: <https://digitalcommons.dartmouth.edu/cgi/viewcontent.cgi?article=5030&context=facoa> (дата посещения 01.12.2024).

# ПРИЛОЖЕНИЕ А

Листинг Программы

import numpy as np  
from PIL import Image  
import math  
  
def rotate\_image\_with\_artifacts(input\_path, output\_path, angle):  
 # Открываем исходное изображение  
 input\_image = Image.open(input\_path)  
 width, height = input\_image.size  
  
 # Конвертируем угол в радианы  
 angle\_rad = math.radians(angle)  
  
 # Создаем новое пустое изображение (начальное состояние пикселей - None)  
 output\_image = Image.new("RGB", (width, height))  
  
 # Центр изображения  
 cx, cy = width // 2, height // 2  
  
 for y in range(height):  
 for x in range(width):  
 # Перемещаем координаты так, чтобы центр изображения был в (0,0)  
 x\_shifted = x - cx  
 y\_shifted = y - cy  
  
 # Применяем матрицу поворота  
 new\_x = round(x\_shifted \* math.cos(angle\_rad) - y\_shifted \* math.sin(angle\_rad))  
 new\_y = round(x\_shifted \* math.sin(angle\_rad) + y\_shifted \* math.cos(angle\_rad))  
  
 # Перемещаем координаты обратно  
 new\_x += cx  
 new\_y += cy  
  
 # Копируем пиксель, если он попадает в границы изображения  
 if 0 <= new\_x < width and 0 <= new\_y < height:  
 output\_image.putpixel((new\_x, new\_y), input\_image.getpixel((x, y)))  
  
 # Сохраняем изображение с артефактами  
 output\_image.save(output\_path)  
 print(f"Изображение сохранено: {output\_path}")  
  
def apply\_transformation(input\_image, width, height, matrix, center):  
 size = input\_image.size  
 new\_size = (width, height)  
  
 # Создаем черное изображение для перемещения  
 output = Image.new('RGB', new\_size, (0, 0, 0))

# Перемещение изображения в центр области  
 for y in range(size[1]):  
 for x in range(size[0]):  
 output.putpixel((x + new\_size[0] // 2 - size[0] // 2, y + new\_size[1] // 2 - size[1] // 2), input\_image.getpixel((x, y)))  
  
 transformed = Image.new('RGB', new\_size, (0, 0, 0))  
  
 for y in range(new\_size[1]):  
 for x in range(new\_size[0]):  
 # Преобразование координат в систему с центром  
 pos = np.array([x - new\_size[0] / 2.0, y - new\_size[1] / 2.0, 1])  
 result = np.zeros(3)  
  
 # Применение матрицы преобразования  
 for i in range(3):  
 result[i] = sum(matrix[i][j] \* pos[j] for j in range(3))  
  
 # Возвращение в исходную систему координат  
 new\_x = int(result[0] + new\_size[0] / 2)  
 new\_y = int(result[1] + new\_size[1] / 2)  
  
 # Проверка границ изображения  
 if 0 <= new\_x < new\_size[0] and 0 <= new\_y < new\_size[1]:  
 transformed.putpixel((new\_x, new\_y), output.getpixel((x, y)))  
  
 return transformed  
  
def crop\_image(input\_image, x, y, width, height):  
 *""" Обрезает изображение до заданных размеров """* return input\_image.crop((x, y, x + width, y + height))  
  
def rotate\_image\_without\_artifacts(input\_image, angle, base\_filename):  
 size = input\_image.size  
 # Размеры увеличенной области для преобразований  
 width = size[0] \* 2  
 height = size[1] \* 2  
  
 temp1 = Image.new('RGB', (width, height), (0, 0, 0))  
 temp2 = Image.new('RGB', (width, height), (0, 0, 0))  
 output = Image.new('RGB', (width, height), (0, 0, 0))  
  
 rad = np.deg2rad(angle)  
 center = (size[0] / 2.0, size[1] / 2.0)  
  
 # Первая матрица  
 shift\_matrix1 = np.array([  
 [1, -np.tan(rad) / 2, 0],  
 [0, 1, 0],  
 [0, 0, 1]  
 ])  
  
 # Вторая матрица  
 shift\_matrix2 = np.array([  
 [1, 0, 0],  
 [np.sin(rad), 1, 0],  
 [0, 0, 1]  
 ])  
  
 # Применение первой матрицы  
 temp1 = apply\_transformation(input\_image, width, height, shift\_matrix1, center)  
 temp1\_cropped = crop\_image(temp1, size[0] / 2.0, size[1] / 2.0, size[0], size[1])  
 temp1\_cropped.save(base\_filename + "\_crop\_step1.bmp")  
  
 # Применение второй матрицы  
 temp2 = apply\_transformation(temp1, width, height, shift\_matrix2, center)  
 temp2\_cropped = crop\_image(temp2, size[0] / 2.0, size[1] / 2.0, size[0], size[1])  
 temp2\_cropped.save(base\_filename + "\_crop\_step2.bmp")  
  
 # Применение первой матрицы  
 output = apply\_transformation(temp2, width, height, shift\_matrix1, center)  
 output\_cropped = crop\_image(output, size[0] / 2.0, size[1] / 2.0, size[0], size[1])  
 output\_cropped.save(base\_filename + "\_crop\_step3.bmp")  
  
 # Обрезка окончательного изображения  
 #output\_cropped = crop\_image(output, size[0] / 2.0, size[1] / 2.0, size[0], size[1])  
 #output\_cropped.save(base\_filename + "\_cropped.bmp")  
  
 return output\_cropped  
  
  
# Задаём параметры:  
a, a3, a5=8, 8\*3, 8\*5  
x=a  
# Поворот через аффинные, с артефактами

rotate\_image\_with\_artifacts("розовый-танк.bmp", f"output\_affin\_{x}.bmp", x)

# Поворот через Оуэна-Македона, без артефактов  
input\_image = Image.open("розовый-танк.bmp")  
rotated\_image = rotate\_image\_without\_artifacts(input\_image, x, f"rotated\_{x}")