МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«Санкт–Петербургский государственный университет  
аэрокосмического приборостроения»

Кафедра №43 «Компьютерных технологий и программной инженерии»

ОТЧЁТ ПО ПРАКТИКЕ

ЗАЩИЩЁН С ОЦЕНКОЙ

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ст. преподаватель |  |  |  | С.А. Рогачев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| вид практики | производственная | |
| тип практики | технологическая (проектно-технологическая) | |
| на тему индивидуального задания | | Алгоритм поворота изображения на произвольный угол |
|  | | | |
|  | | | |

|  |  |
| --- | --- |
| выполнен |  |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 09.03.04 |  | Программная инженерия |
|  | код |  | наименование направления |
|  | | | |
| наименование направления | | | |
| направленности | 01 |  | Проектирование программных систем |
|  | код |  | наименование направленности |
| систем | | | |
| наименование направленности | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | Z9431 |  | 20.04.2021 |  | Пайвин А.А. |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2021

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc74181463)

[2 Исходные данные 4](#_Toc74181464)

[3 Теоретическая часть 5](#_Toc74181465)

[4 Практическая часть 8](#_Toc74181466)

[4.1 Реализация программы 8](#_Toc74181467)

[4.2 Тестирование 9](#_Toc74181468)

[5 Выводы 11](#_Toc74181469)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 12](#_Toc74181470)

[ПРИЛОЖЕНИЕ 1 13](#_Toc74181471)

# **Цель работы**

Целью работы является изучение и реализация алгоритма поворота изображения на произвольный угол.

Для достижения заданной цели необходимо решить следующие задачи:

1) Изучить общие и частные теоретические положения;

2) Выбрать исходные данные;

3) Разработать алгоритм решения задачи;

4) Написать код программы на заданном языке программирования;

5) Провести тестирование программы;

6) Сделать выводы.

# **Исходные данные**

В качестве исходных данных будем использовать изображение cat.jpg, показанное на рисунке 1, и угол поворота.



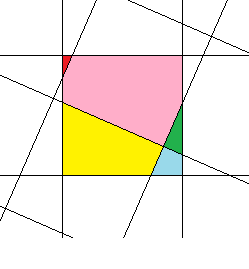
Рисунок 1 – Исходное изображение

# **Теоретическая часть**

Поворот растрового изображения на углы, кратные 90°, относительно геометрического центра изображения – задача тривиальная и решается без потери качества простым преобразованием координат каждого пикселя.

Для поворота растрового изображения на произвольный угол разработаны быстрые алгоритмы, дающие приемлемую для практических целей аппроксимацию с потерей качества

Алгоритм



Из вышеприведённого рисунка видно, что после поворота растрового изображения, каждый пиксель разбивается на несколько «осколков». Поэтому для идеально точного поворота необходимо высчитывать площади осколков каждого пикселя, складывать цвета пропорционально площади осколков. На эти операции уходит много времени и много вычислительной мощности. Поэтому в общем виде решение нашей задачи будет состоять в том, чтобы для центра каждого исходного пикселя найти новые координаты, и записать цвет, то есть, куда попал центр пикселя, таким цветом и закрашиваем весь пиксель. В процессе поворота будут возникать ситуации, когда центры двух пикселей попадут в один пиксель. Также будут возникать ситуации, что пиксель результирующего изображения останется не закрашенным. Чтобы избавиться от пустых пикселей, необходимо закрасить их с помощью интерполяции, найти среднее значение среди окружающих пикселей и записать его. Данный алгоритм быстро поворачивает изображение. Недостаток в незначительном изменении изображении, некоторые пиксели навсегда утеряны, другие незначительно сдвигаются. Но с учетом немаленького разрешения изображения, разница незаметна невооруженным глазом

Моделью итогового изображения будет сетка из параллельных горизонтальных и вертикальных линий с расстоянием между линиями = 1.

Поворот изображения будет осуществляться относительно центра, но при повороте увеличится разрешение изображения, поэтому необходимо сначала их вычислить и, поделив размеры нового изображения на 2, вычислить центр

Вычисление размеров нового изображения:

int w1 = abs(w \* cos(α) + h \* sin(α) + 1);

int h1 = abs(w \* sin(α) + h \* cos(α) + 1);

Где w1 - ширина, h1 – высота, w и h – ширина и высота исходного изображения соответственно

Преобразование точки исходного изображения в точку повернутого изображения:

double x1 = cx1 + (0.5 - cx) \* cos(f) - (0.5 - cy) \* sin(f); // x1 - y4 - координаты углов картинки после поворота

double y1 = cy1 + (0.5 - cy) \* cos(f) + (0.5 - cx) \* sin(f);

Где double cx1 и double cy1 – координаты центра повернутого изображения, double cx и double cy – координаты центра исходного изображения, double f - угол поворота

Для быстрого расчета, сначала найдем новые координаты центров углов исходного изображения, затем, зная разрешение исходного изображения, вычисляем новые координаты для центра каждого пикселя и записываем в массив, размер которого равен разрешению исходного изображения.

Затем создаем двумерный массив bool размером с результирующее изображения для поиска и записи незаполненных пикселей, проходимся по изображению построчно, и заполняем массив.

Далее закрашиваем пустые пиксели средним цветом среди окружающих пикселей

# **Практическая часть**

## **4.1 Реализация программы**

1. Загружаем в память ЭВМ исходное изображение.

2. Рассчитываем размеры итогового изображения в пикселях.

3. Если угол поворота кратен 90 градусам, тогда создаем новое изображение и записываем пиксели по порядку, на этом задача закончена

4. Иначе создаём промежуточный трехмерный массив, который содержит новые координаты для каждого пикселя. Размер массива равен размерам исходного изображения.

5. Находим новые координаты для центров крайних по углам пикселей исходного изображения

6. Создаем массив для новых координат центров пикселей всего исходного изображения и заполняем его

7. Заполняем новое изображение пикселями, используя их новые координаты

8. Ищем незаполненные пиксели и заполняем их средним значением среди окружающих пикселей

9. Сохраняем повернутое изображение

Программа реализована на языке программирования С++ стандарта 17 года. Кодовая база использует библиотеку OpenCV.

Программный код можно увидеть в приложении 1.

## **4.2 Тестирование**

Тестирование программы производится в Visual Studio 2019

Пример ввода файла и угла поворота показан на рисунке 2.

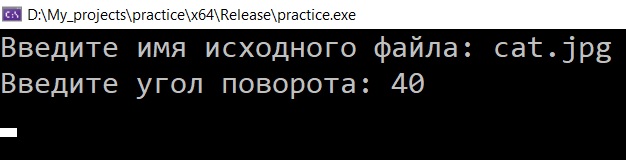


Рисунок 2 – Пример ввода данных

На рисунке 3 можно увидеть результат поворота.



Рисунок 3, результат поворота

Введем угол поворота 220 градусов, на рисунке 4 показан ввод данных

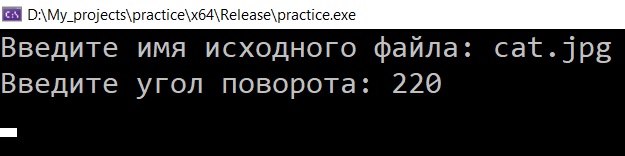


Рисунок 4, пример ввода данных

На рисунке 5 можно увидеть результат поворота



Рисунок 5, результат поворота

Исходя из тестов делаем вывод, что задача на реализацию выполнена.

# **Выводы**

В данной работе была разработана программа, которая принимает на вход изображение, угол поворота, и осуществляет поворот относительно центра на заданный угол, был изучен алгоритм поворота

Для реализации программы использовался язык программирования C++ 17.

К достоинствам программы можно отнести:

Программа выполняет поставленную задачу без ошибок;

Программа декомпозирована на функции, что позволяет использовать её в других проектах;

Существенных недостатков, кроме незначительного искажения изображения, программа не имеет.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1 Вейцман, В.М. Проектирование информационных систем : учебное пособие / В.М. Вейцман. — Санкт-Петербург : Лань, 2019. — 316 с. Текст : электронный https://e.lanbook.com/book/122172

2 Ключарев, А. А. Структуры и алгоритмы обработки данных [Текст] : учебное пособие / А. А. Ключарев, В. А. Матьяш, С. В. Щекин ; С.-Петерб. гос. ун-т аэрокосм. приборостроения. - СПб. : Изд-во ГУАП, 2004. - 47 с.

3 Обработка изображений с помощью OpenCV / Глория Буэно Гарсия; пер. с англ. А.А. Слинкина. - Москва : ДМК Пресс, 2016. - 210 с. Текст : электронный. - URL: https://znanium.com/catalog/document?id=341197

4 Страуструп, Б. Язык программирования C++ [Текст] = The C++ Programming Language : специальное издание / Б. Страуструп ; пер.: С. Анисимов, М. Кононов ; ред.: Ф. Андреев, А. Ушаков. - [Б. м.] : Бином-Пресс, 2008. - 1098 с.

5. Анисимов Б.В., Курганов В.Д., Злобин В.К. Распознавание и цифровая обработка изображений. - СПб.: БХВ – Санкт-Петербург, 2010. – 528 с.

6. Кашкин В.Б., Сухинин А.И. Цифровая обработка изображений. - М.: СК Пресс.- 2011.

# **ПРИЛОЖЕНИЕ 1**

#include <opencv2/opencv.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <iostream>

#include <string>

#include <cmath>

using namespace cv;

using namespace std;

const double PI = 3.141592653589793238462;

void set\_resolution(const int w, const int h, int& w1, int& h1, double& f\_gr, double& f\_rad, double& f) {

// преобразуем угол поворота, чтобы он был от 0 до 360 градусов

while (f\_gr < 0) {

f\_gr += 360;

}

int int\_f = int(f\_gr);

f\_gr -= int\_f;

int\_f = int\_f % 360;

f\_gr += int\_f;

f = PI \* f\_gr / 180.0; // перевод в радианы

if (f\_gr == 0 || f\_gr == 180) {

w1 = w;

h1 = h;

}

else if (f\_gr == 90 || f\_gr == 270) {

w1 = h;

h1 = w;

}

else {

if (f\_gr < 90) {

w1 = abs(w \* cos(f) + h \* sin(f) + 1);// размеры нового изображения

h1 = abs(w \* sin(f) + h \* cos(f) + 1);

}

else if (f\_gr < 180) {

f\_gr = 90 - (f\_gr - 90);

f\_rad = PI \* f\_gr / 180.0; // перевод в радианы

w1 = abs(w \* cos(f\_rad) + h \* sin(f\_rad) + 1);// размеры нового изображения

h1 = abs(w \* sin(f\_rad) + h \* cos(f\_rad) + 1);

}

else if (f\_gr < 270) {

f\_gr = f\_gr - 180;

f\_rad = PI \* f\_gr / 180.0; // перевод в радианы

w1 = abs(w \* cos(f\_rad) + h \* sin(f\_rad) + 1);// размеры нового изображения

h1 = abs(w \* sin(f\_rad) + h \* cos(f\_rad) + 1);

}

else {

f\_gr = 360 - f\_gr;

f\_rad = PI \* f\_gr / 180.0; // перевод в радианы

w1 = abs(w \* cos(f\_rad) + h \* sin(f\_rad) + 1);// размеры нового изображения

h1 = abs(w \* sin(f\_rad) + h \* cos(f\_rad) + 1);

}

}

}

void rotation\_180(const int h, const int w, Mat rotation, Mat image) {

for (int i = 0; i < h; i++) { // вывод

for (int j = 0; j < w; j++) {

rotation.at<Vec3b>(h - i - 1, w - j - 1)[0] = image.at<Vec3b>(i, j)[0];

rotation.at<Vec3b>(h - i - 1, w - j - 1)[1] = image.at<Vec3b>(i, j)[1];

rotation.at<Vec3b>(h - i - 1, w - j - 1)[2] = image.at<Vec3b>(i, j)[2];

}

}

}

void rotation\_90(const int h, const int w, Mat rotation, Mat image) {

for (int i = 0; i < h; i++) { // вывод

for (int j = 0; j < w; j++) {

rotation.at<Vec3b>(j, h - i - 1)[0] = image.at<Vec3b>(i, j)[0];

rotation.at<Vec3b>(j, h - i - 1)[1] = image.at<Vec3b>(i, j)[1];

rotation.at<Vec3b>(j, h - i - 1)[2] = image.at<Vec3b>(i, j)[2];

}

}

}

void rotation\_270(const int h, const int w, Mat rotation, Mat image) {

for (int i = 0; i < h; i++) { // вывод

for (int j = 0; j < w; j++) {

rotation.at<Vec3b>(w - j - 1, i)[0] = image.at<Vec3b>(i, j)[0];

rotation.at<Vec3b>(w - j - 1, i)[1] = image.at<Vec3b>(i, j)[1];

rotation.at<Vec3b>(w - j - 1, i)[2] = image.at<Vec3b>(i, j)[2];

}

}

}

void count\_pixel\_centers(double\*\*\* centre, const int h, const int w, const double h1\_w2, const double w1\_h2, const double x1, const double y1) {

double x\_, x\_\_, y\_, y\_\_; // координаты центра пикселя из первого столбца для создания таблицы центров пикселей

for (int i = 0; i < h; i++) {

centre[i] = new double\* [w];

for (int j = 0; j < w; j++) {

centre[i][j] = new double[2];

}

}

x\_ = x1;

x\_\_ = x1;

y\_ = y1;

y\_\_ = y1;

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

centre[i][j][0] = x\_;

centre[i][j][1] = y\_;

x\_ = x\_ + w1\_h2;

y\_ = y\_ + h1\_w2;

}

y\_\_ += w1\_h2;

x\_\_ -= h1\_w2;

x\_ = x\_\_;

y\_ = y\_\_;

}

}

void calculate\_angles(double f, int h, int w, int h1, int w1, double& cx, double& cy, double& cx1, double& cy1,

double& x1, double& y1, double& x2, double& y2, double& x3, double& y3, double& x4, double& y4, double& h1\_w2, double& w1\_h2) {

cx1 = w1 / 2;

cy1 = h1 / 2;// центр повернутого изображения

cx = w / 2;

cy = h / 2; // центр исходного изображения

x1 = cx1 + (0.5 - cx) \* cos(f) - (0.5 - cy) \* sin(f); // x1 - y4 - координаты углов картинки после поворота

y1 = cy1 + (0.5 - cy) \* cos(f) + (0.5 - cx) \* sin(f);

x2 = cx1 + (w - 0.5 - cx) \* cos(f) - (0.5 - cy) \* sin(f);

y2 = cy1 + (0.5 - cy) \* cos(f) + (w - 0.5 - cx) \* sin(f);

x3 = cx1 + (0.5 - cx) \* cos(f) - (h - 0.5 - cy) \* sin(f);

y3 = cy1 + (h - 0.5 - cy) \* cos(f) + (0.5 - cx) \* sin(f);

x4 = cx1 + (w - 0.5 - cx) \* cos(f) - (h - 0.5 - cy) \* sin(f);

y4 = cy1 + (h - 0.5 - cy) \* cos(f) + (w - 0.5 - cx) \* sin(f);

h1\_w2 = (y2 - y1) / (w - 1); // разница между центрами пикселей

w1\_h2 = (y3 - y1) / (h - 1); // разница между центрами пикселей

}

void write\_matrix(double\*\*\* centre, const int h, const int w, Mat rotation, Mat image) {

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

rotation.at<Vec3b>(int(centre[i][j][1]), int(centre[i][j][0]))[0] = image.at<Vec3b>(i, j)[0];

rotation.at<Vec3b>(int(centre[i][j][1]), int(centre[i][j][0]))[1] = image.at<Vec3b>(i, j)[1];

rotation.at<Vec3b>(int(centre[i][j][1]), int(centre[i][j][0]))[2] = image.at<Vec3b>(i, j)[2];

}

}

}

void remove\_black\_dots(int w1, int h1, Mat rotation) {

bool\*\* black\_p = new bool\* [h1];// инициализация массива черных точек

for (int i = 0; i < h1; i++) {

black\_p[i] = new bool[w1];

}

for (int i = 0; i < h1; i++) { // обнуление массива

for (int j = 0; j < w1; j++) {

black\_p[i][j] = false;

}

}

//поиск черных точек

for (int i = 1; i < h1 - 1; i++) {

for (int j = 1; j < w1 - 1; j++) {

if (rotation.at<Vec3b>(i, j)[0] == 0 && rotation.at<Vec3b>(i, j)[1] == 0 && rotation.at<Vec3b>(i, j)[2] == 0) {

if ((rotation.at<Vec3b>(i, j - 1)[0] != 0 || rotation.at<Vec3b>(i, j - 1)[1] != 0 || rotation.at<Vec3b>(i, j - 1)[2] != 0) &&

(rotation.at<Vec3b>(i, j + 1)[0] != 0 || rotation.at<Vec3b>(i, j + 1)[1] != 0 || rotation.at<Vec3b>(i, j + 1)[2] != 0)) {

black\_p[i][j] = true;

}

}

}

}

//изменение черных точек интерполяцией

int num\_p = 0, sum\_r = 0, sum\_g = 0, sum\_b = 0;

for (int i = 1; i < h1 - 1; i++) {

for (int j = 1; j < w1 - 1; j++) {

if (black\_p[i][j]) {

if (!black\_p[i - 1][j - 1]) {

sum\_r += rotation.at<Vec3b>(i - 1, j - 1)[0];

sum\_g += rotation.at<Vec3b>(i - 1, j - 1)[1];

sum\_b += rotation.at<Vec3b>(i - 1, j - 1)[2];

num\_p += 1;

}

if (!black\_p[i - 1][j]) {

sum\_r += rotation.at<Vec3b>(i - 1, j)[0];

sum\_g += rotation.at<Vec3b>(i - 1, j)[1];

sum\_b += rotation.at<Vec3b>(i - 1, j)[2];

num\_p += 1;

}

if (!black\_p[i - 1][j + 1]) {

sum\_r += rotation.at<Vec3b>(i - 1, j + 1)[0];

sum\_g += rotation.at<Vec3b>(i - 1, j + 1)[1];

sum\_b += rotation.at<Vec3b>(i - 1, j + 1)[2];

num\_p += 1;

}

if (!black\_p[i][j - 1]) {

sum\_r += rotation.at<Vec3b>(i, j - 1)[0];

sum\_g += rotation.at<Vec3b>(i, j - 1)[1];

sum\_b += rotation.at<Vec3b>(i, j - 1)[2];

num\_p += 1;

}

if (!black\_p[i][j + 1]) {

sum\_r += rotation.at<Vec3b>(i, j + 1)[0];

sum\_g += rotation.at<Vec3b>(i, j + 1)[1];

sum\_b += rotation.at<Vec3b>(i, j + 1)[2];

num\_p += 1;

}

if (!black\_p[i + 1][j - 1]) {

sum\_r += rotation.at<Vec3b>(i + 1, j - 1)[0];

sum\_g += rotation.at<Vec3b>(i + 1, j - 1)[1];

sum\_b += rotation.at<Vec3b>(i + 1, j - 1)[2];

num\_p += 1;

}

if (!black\_p[i + 1][j]) {

sum\_r += rotation.at<Vec3b>(i + 1, j)[0];

sum\_g += rotation.at<Vec3b>(i + 1, j)[1];

sum\_b += rotation.at<Vec3b>(i + 1, j)[2];

num\_p += 1;

}

if (!black\_p[i + 1][j + 1]) {

sum\_r += rotation.at<Vec3b>(i + 1, j + 1)[0];

sum\_g += rotation.at<Vec3b>(i + 1, j + 1)[1];

sum\_b += rotation.at<Vec3b>(i + 1, j + 1)[2];

num\_p += 1;

}

rotation.at<Vec3b>(i, j)[0] = sum\_r / num\_p;

rotation.at<Vec3b>(i, j)[1] = sum\_g / num\_p;

rotation.at<Vec3b>(i, j)[2] = sum\_b / num\_p;

num\_p = 0;

sum\_r = 0;

sum\_g = 0;

sum\_b = 0;

}

}

}

for (int i = 0; i < h1; i++) {

delete[] black\_p[i];

}

delete[] black\_p;

}

void del\_centre(double\*\*\* centre, int h, int w) {

for (int i = 0; i < h; i++) {

for (int j = 0; j < w; j++) {

delete[] centre[i][j];

}

delete[] centre[i];

}

delete[] centre;

}

int main()

{

setlocale(LC\_ALL, "Russian");

string imageName;

Mat image;

//imageName = "cat.jpg";

cout << "Введите имя исходного файла: ";

cin >> imageName;

image = imread(imageName.c\_str(), IMREAD\_COLOR);

if (image.empty())

{

cout << "Ошибка открытия файла " << endl;

return -1;

}

const int w = image.size().width;

const int h = image.size().height;

int h1, w1;

double f\_gr; // угол поворота в градусах

double f, f\_rad; // угол поворота в радианах

cout << "Введите угол поворота: ";

cin >> f\_gr;

set\_resolution(w, h, w1, h1, f\_gr, f\_rad, f);

Mat rotation(h1, w1, CV\_8UC3, Scalar(0, 0, 0));

if (f\_gr == 0) {

rotation = image;

}

else if (f\_gr == 180) {

rotation\_180(h, w, rotation, image);

}

else if (f\_gr == 90) {

rotation\_90(h, w, rotation, image);

}

else if (f\_gr == 270) {

rotation\_270(h, w, rotation, image);

}

else {

double cx, cy; // центр исходного изображения

double cx1, cy1;// центр повернутого изображения

double x1, y1, x2, y2, x3, y3, x4, y4; // x1 - y4 - координаты углов картинки после поворота

double h1\_w2, w1\_h2; // разница между центрами пикселей

//расчет координат углов повернутого изображения

calculate\_angles(f, h, w, h1, w1, cx, cy, cx1, cy1, x1, y1, x2, y2, x3, y3, x4, y4, h1\_w2, w1\_h2);

double\*\*\* centre = new double\*\* [h];// инициализация массива центров пикселей

// вычисляем координаты центров пикселей

count\_pixel\_centers(centre, h, w, h1\_w2, w1\_h2, x1, y1);

// запись в матрицу изображения

write\_matrix(centre, h, w, rotation, image);

// удаляем черные точки

remove\_black\_dots(w1, h1, rotation);

//очистка памяти

del\_centre(centre, h, w);

}

imshow("rotation", rotation);

imshow("image", image);

imwrite("./output.jpg", rotation);

waitKey(0);

return 0;

}