МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Цифровая обработка изображений»

Тема: Геометрические преобразования изображений

Студент гр. 7381 _____ Минуллин М.А.

Преподаватель _____ Черниченко Д.А.

Санкт-Петербург

2021

Постановка задачи.

Реализовать на языке Python с использованием библиотеки OpenCV программу, выполняющее геометрические преобразования изображения

Входные данные.

1. Цветное изображение в формате bmp, jpg
1. Угол поворота α – целое число
2. Коэффициент k – рациональное число
3. Признак используемой интерполяционной схемы (0 – нулевого порядка, 1 – первого порядка, 2- третьего порядка).

Выходные данные.

Цветное изображение в формате bmp (result.bmp), являющиеся результатом следующих преобразований:

1. Исходное изображение поворачивается на α градусов против часовой стрелке
2. Из получившегося изображения вырезается прямоугольник максимальной площади, вписанный в повернутое изображение
3. Данное изображение увеличивается по оси Y в $\frac{1}{k}$ раз и уменьшается по оси X в k раз.
4. Результат с использованием интерполяции соответствующего порядка записывается в файл result.bmp

Выполнение работы.

Входные данные будем получать из параметров командной строки:

```
filename = sys.argv[1]
alpha = int(sys.argv[2])
k = float(sys.argv[3])
interpol = int(sys.argv[4])
```

В качестве примера будут использоваться параметры «image.jpg 30 1.5 2». Откроем исходное изображение и отобразим его на экране (результат на рис. 1):

```
original_image = cv.imread(filename)
cv.imshow('Original image', original_image)
```

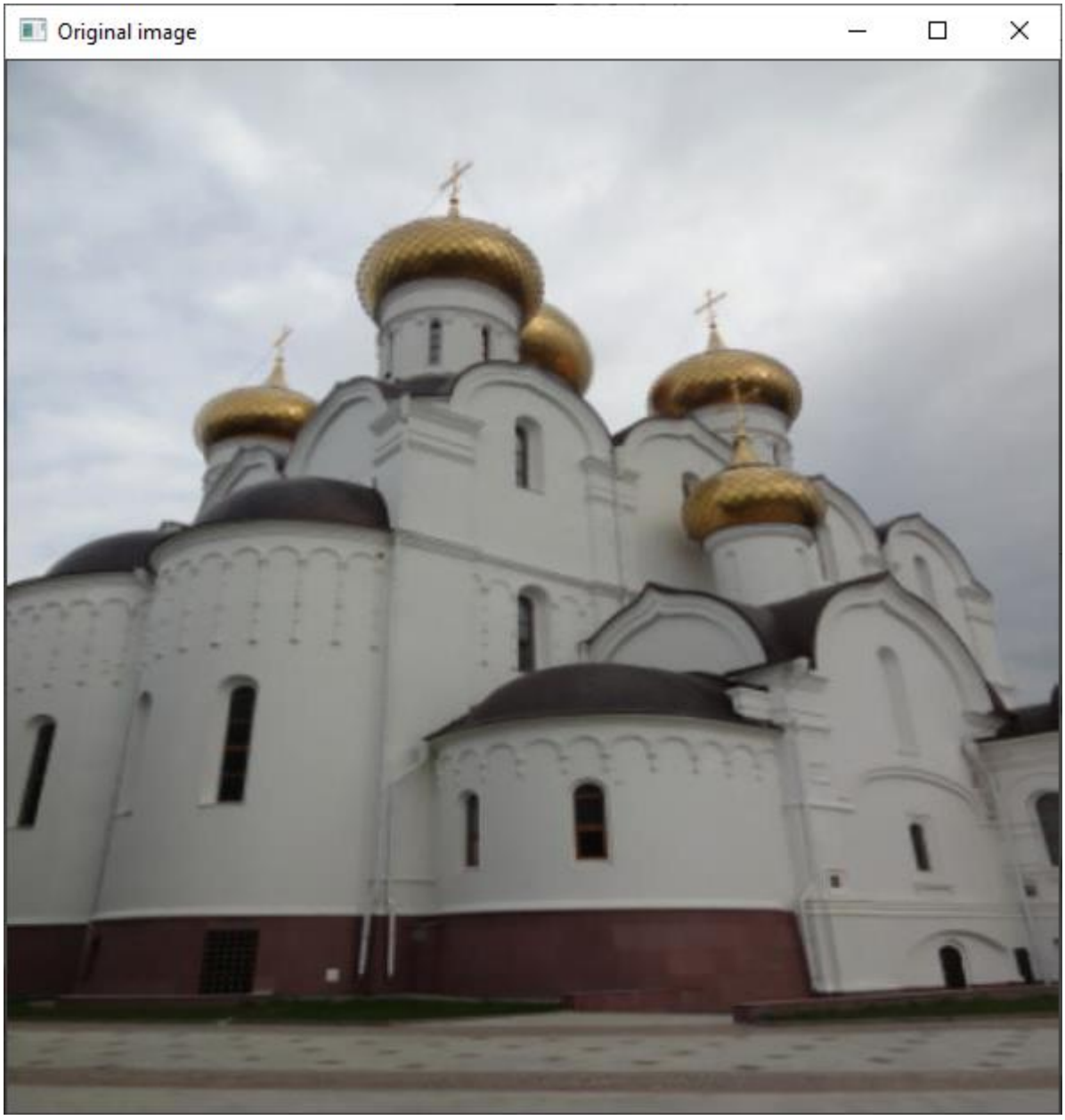

Рисунок 1 – Исходное изображение

Осуществим поворот изображения и выведем полученный результат на экран (результат представлен на рис. 2):

```
(rows, cols) = original_image.shape[:2]
M = cv.getRotationMatrix2D(((cols - 1) / 2.0, (rows - 1) / 2.0), alpha, 1)
rotated_image = cv.warpAffine(original_image, M, (cols, rows))
```

```
cv.imshow('Rotated image', rotated_image)
```

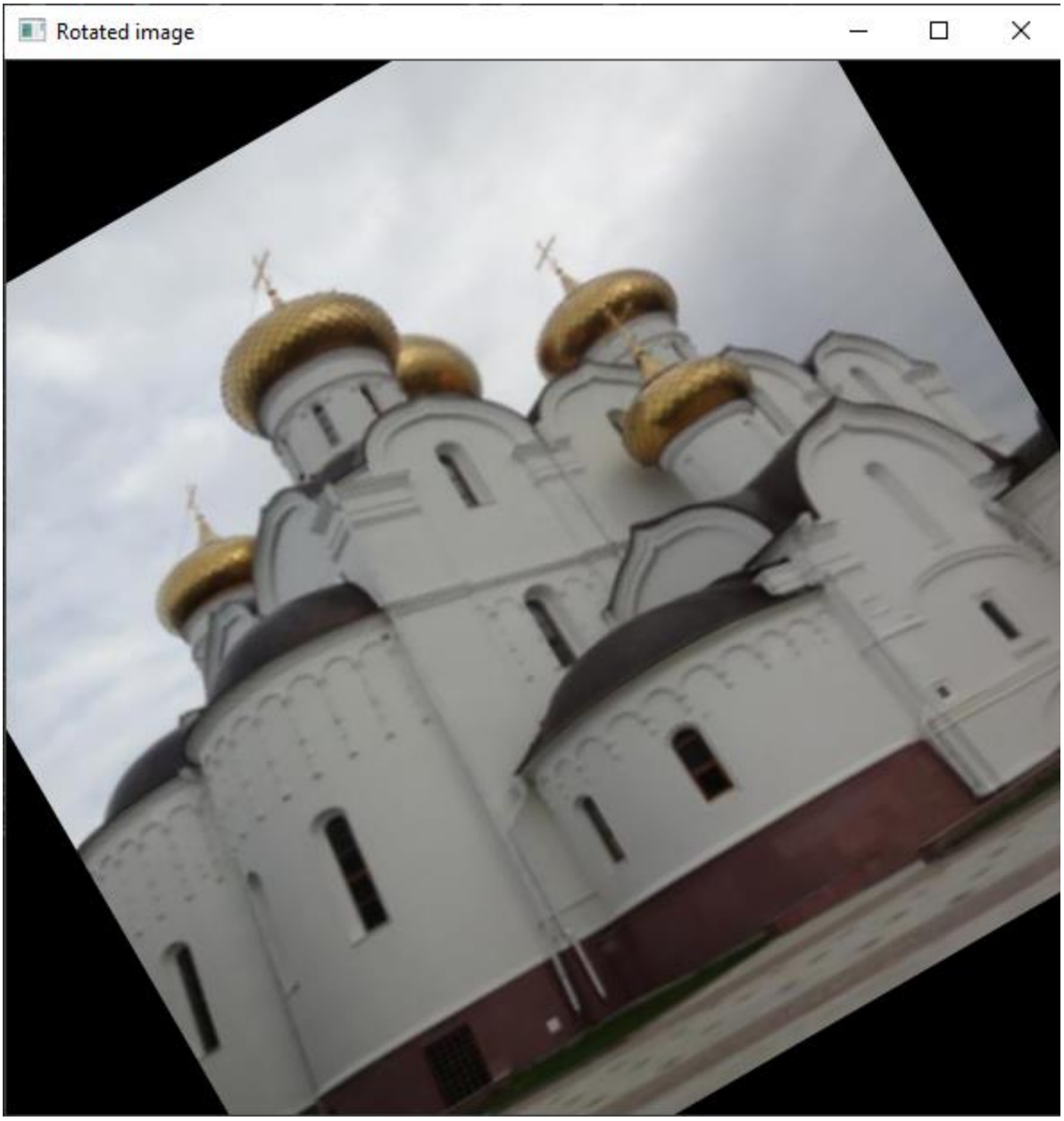

Рисунок 2 – Повёрнутое изображение

Из полученного изображения вырежем прямоугольник наибольшей площади, вписанный в повёрнутое изображение. Отобразим полученный результат на экране (результат представлен на рис. 3).

```
cropped_image = crop_around_center(rotated_image, *largest_rotated_rect(cols, rows, math.radians(alpha)))
cv.imshow('Cropped image', cropped_image)
```

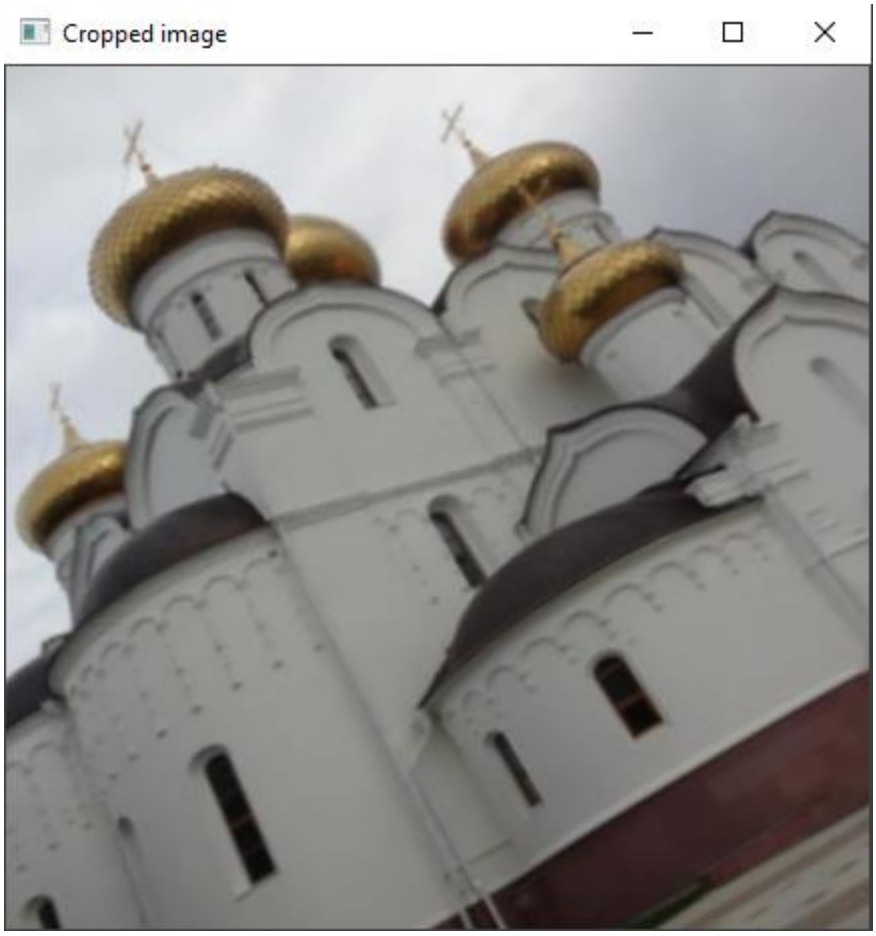

Рисунок 3 – Обрезанное изображение

Увеличим обрезанное изображение по оси Oy в $\frac{1}{k}$ раз и уменьшим по оси Ox в k раз. Используем при этом заданную схему интерполяции (результат представлен на рис. 4):

```
resized_image = cv.resize(cropped_image, None, fx=k, fy=1.0/k, interpolation=f(interpol))
cv.imshow('Resized image', resized_image)
```

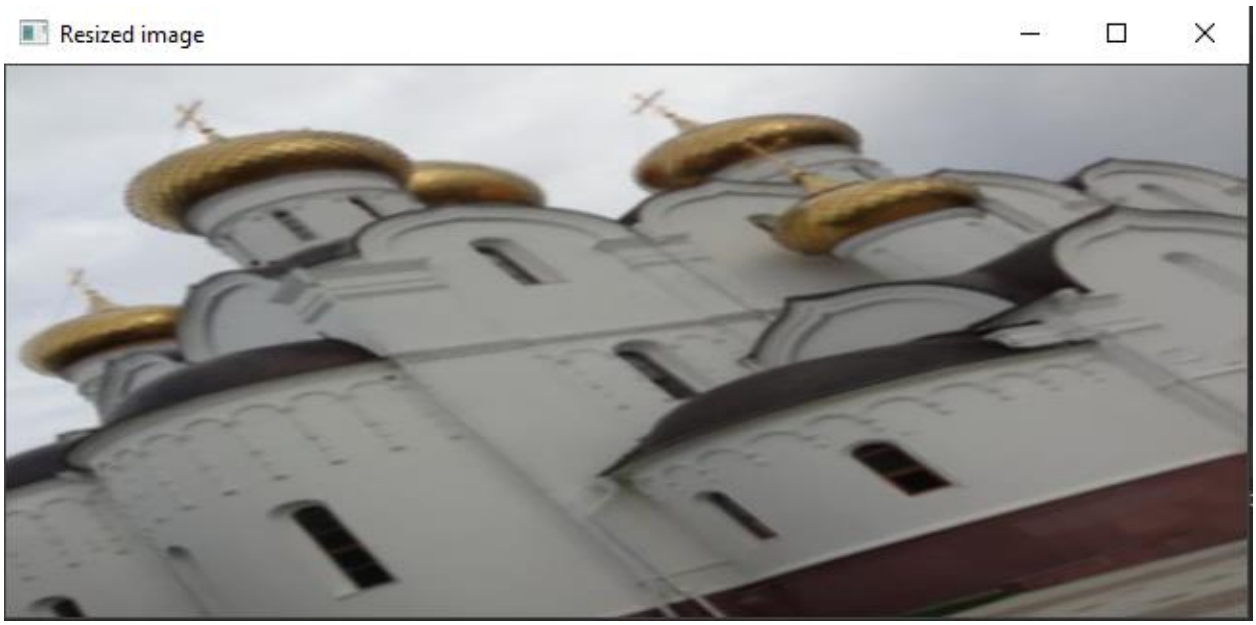

Рисунок 4 – Изображение после масштабирования

Сохраним полученный результат с именем «result.bmp»:

```
cv.imwrite('result.bmp', resized_image)
```

Полный код программы представлен в приложении А.

Выводы.

В ходе выполнения лабораторной работы были изучены различные геометрические преобразования изображений с помощью библиотеки OpenCV.

На языке Python была написана программа, выполняющая данные геометрические преобразования.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

```
import cv2 as cv
import math
import sys


def largest_rotated_rect(w, h, angle):
    quadrant = int(math.floor(angle / (math.pi / 2))) & 3
    sign_alpha = angle if ((quadrant & 1) == 0) else math.pi - angle
    alpha = (sign_alpha % math.pi + math.pi) % math.pi

    bb_w = w * math.cos(alpha) + h * math.sin(alpha)
    bb_h = w * math.sin(alpha) + h * math.cos(alpha)

    gamma = math.atan2(bb_w, bb_w) if (w < h) else math.atan2(bb_w,
bb_w)

    delta = math.pi - alpha - gamma

    length = h if (w < h) else w

    d = length * math.cos(alpha)
    a = d * math.sin(alpha) / math.sin(delta)

    y = a * math.cos(gamma)
    x = y * math.tan(gamma)

    return (
```

```
        bb_w - 2 * x,
        bb_h - 2 * y
    )


def crop_around_center(image, width, height):
    image_size = (image.shape[1], image.shape[0])
    image_center = (int(image_size[0] * 0.5), int(image_size[1] *
0.5))

    if width > image_size[0]:
        width = image_size[0]

    if height > image_size[1]:
        height = image_size[1]

    x1 = int(image_center[0] - width * 0.5)
    x2 = int(image_center[0] + width * 0.5)
    y1 = int(image_center[1] - height * 0.5)
    y2 = int(image_center[1] + height * 0.5)

    return image[y1:y2, x1:x2]


def f(interpol):
    return {
        0: cv.INTER_NEAREST,
        1: cv.INTER_LINEAR,
        2: cv.INTER_CUBIC
    }[interpol]
```

```
filename = sys.argv[1]

alpha = int(sys.argv[2])

k = float(sys.argv[3])

interpol = int(sys.argv[4])


original_image = cv.imread(filename)

cv.imshow('Original image', original_image)


(rows, cols) = original_image.shape[:2]

M = cv.getRotationMatrix2D(((cols - 1) / 2.0, (rows - 1) / 2.0),
alpha, 1)

rotated_image = cv.warpAffine(original_image, M, (cols, rows))

cv.imshow('Rotated image', rotated_image)


cropped_image = crop_around_center(rotated_image,
*largest_rotated_rect(cols, rows, math.radians(alpha)))

cv.imshow('Cropped image', cropped_image)


resized_image = cv.resize(cropped_image, None, fx=k, fy=1.0/k,
interpolation=f(interpol))

cv.imshow('Resized image', resized_image)

cv.imwrite('result.bmp', resized_image)

cv.waitKey(0)
```