**A**

**Project Report**

On

**Rotating an Image via OpenCV**

(CSE III Semester Mini project)

2019-2023

****

**Submitted To: Submitted By:**

Mrs. Meenakshi Maindola Sona verma

Prof. of Department (CSE) Roll no: 1918733

CSE-F-III-Sem

Session:2019-23

**Department of Computer Science & Engineering**

**Acknowledgement**

I am profoundly grateful to Mrs. Meenakshi Maindola for her expert guidance and continuous encouragement throughout to see that this project rights its target.

At last I must express my sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped us directly or indirectly during this course of work.

Sona Verma

**INTRODUCTION**

# **OpenCV Image Rotation**

The image can be rotated in various angles (90,180,270 and 360). OpenCV calculates the affine matrix that performs affine transformation, which means it does not preserve the angle between the lines or distances between the points, although it preserves the ratio of distances between points lying on the lines.

The syntax of the rotate image is the following:

1. cv2.getRotationMatrix2D(center, angle, scale rotated = cv2.warpAfifne(img,M,(w,h))

### Parameters:

* **center:** It represents the center of the image.
* **angle:** It represents the angle by which a particular image to be rotated in the anti-clockwise direction.
* **rotated:** ndarray that holds the rotated image data.
* **scale:** The value 1.0 is denoted that the shape is preserved. Scale the image according to the provided value.

**Rotate Image**

Rotating images by a given angle is a common image processing task. Although it seems little bit complicated, OpenCV provides some built-in functions making it easy to do it. Here is a simple OpenCV C++ example code to rotate an image.

**Pseudo Code**

#include<opencv2/opencv.hpp>

#include<opencv2/core/utility.hpp>

#include<opencv2/core/core.hpp>

#include<opencv2/highgui/highgui.hpp>

#include<opencv2/imgproc/imgproc.hpp>

#include<iostream>

using namespace std;

using namespace cv;

int main()

{

Mat image1, final\_im;

float angle;

image1 = imread("lena.jpg");

cout << "Dimensions are" << image1.rows << "x" << image1.cols << endl;

if (image1.empty())

{

cout << "Not Successfully Loaded!" << endl;

}

else {

cout << "Enter the angle by which the image should be rotated" << endl;

cin >> angle;

Point2f pt(image1.cols / 2., image1.rows / 2.);

Mat v = getRotationMatrix2D(pt, angle, 1.0);

warpAffine(image1, final\_im, v, Size(image1.cols, image1.rows));

imshow("Image before rotation", image1);

imshow("Final Image", final\_im);

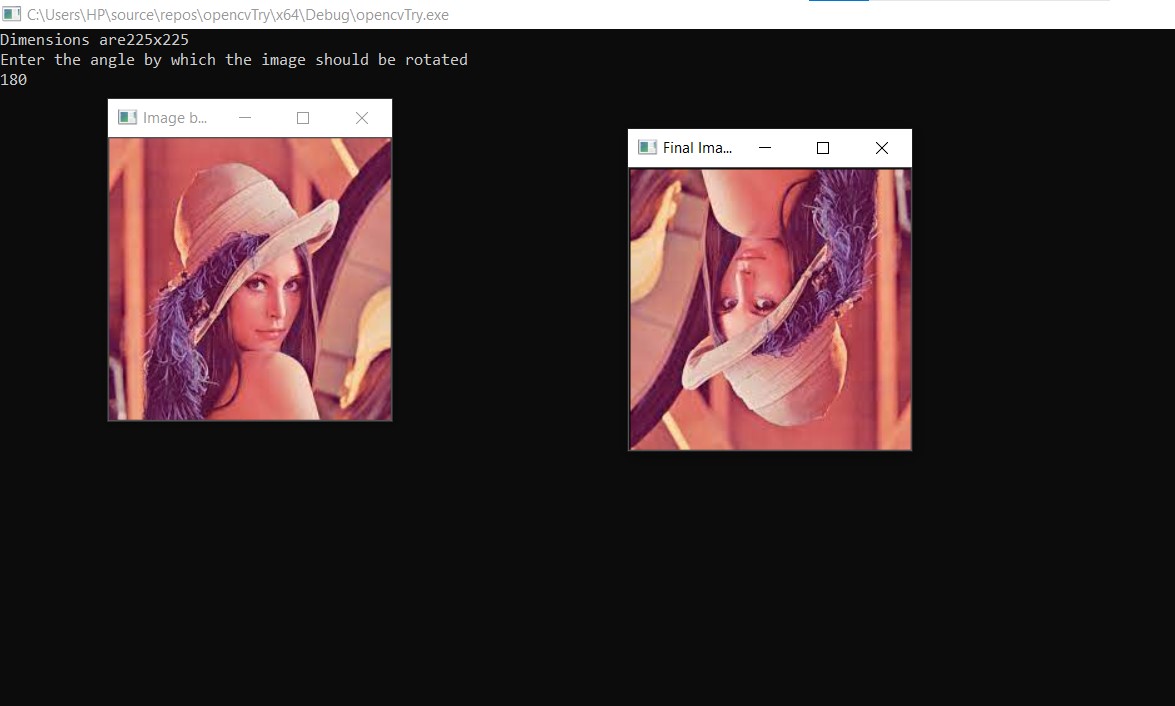
waitKey(0);

}

return 0;

}

**Snapshot of Project**

****

**Explanation of Functions Used In Code**

**Mat getRotationMatrix2D( Point2f center, double angle, double scale )**

This function returns 2x3 affine transformation matrix for the 2D rotation.

**Arguments -**

* + **center**- The center of the rotation of the source image.
  + **angle**- Angle of rotation in degrees (Positive values for counter-clockwise direction and negative **values**for clockwise rotation)
  + **scale**- The scaling factor of the image. (Scaling factor of 1 means its original size)

Try different values for **center**, **angle**and **scale**and observe the output image.

**void warpAffine( InputArray src, OutputArray dst, InputArray M, Size dsize, int flags = INTER\_LINEAR, int bordreMode=BORDER\_CONSTANT, const Scalar& borderValue=Scalar() )**

This OpenCV function applies [affine transformation](http://en.wikipedia.org/wiki/Affine_transformation) to an image.

**Arguments** -

* + **src**- Source Image
  + **dst**- Destination image which should have the same type as the source image(The transformed image is stored in this location)
  + **M**- 2x3 affine transformation matrix
  + **dsize**- Size of the destination image
  + **flags**- Interpolation methods
  + **borderMode**- pixel extrapolation method.
  + **borderValue**- If you use BORDER\_CONSTANT for **borderMode**, this argument define the value used for the border

**References**

1. javatpoint.com
2. opencv tutorial