**Project Requirement and Specification**

**On**

**IMAGE PROCESSING**

**2 -Week Winter Programming Bootcamp**

**Submitted to: Submitted by:**

Ms. Meenakshi Mandola NANDINI GUPTA

UNIVERSITY ROLL.NO:-2014421

**Guided by:** SEMESTER:-III

Ms. Meenakshi Mandola Session: 2020-2021

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GRAPHIC ERA DEEMED TO BE UNVERSITY, DEHRADUN**

* **ABOUT PROJECT**:

The project statement is “Input an image, input an angle and rotate the image by the given angle”. So,this project is about rotating an image to the given angle with the help of open source computer vision with C++ . Open CV is a powerful tool in the area of image processing because of its speed and intuitive API. Open CV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. Open CV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products.

* **TECHNOLOGIES USED IN THE PROJECT**:

1.Visual Studio Community

2.Open CV

Before, I get into the use of Open CV; Let’s get familiar with the same. As, by now you must have understood that it is a 'library' , so to use the functions available in the library you would need a compiler.

To start off, you need to install 'Open CV' and a 'compiler' and then establish a linkage between the two (i.e. Compiler is able to use the functions available with the library).

* **Steps to install and set open CV.**

Step 1 - Download the latest binary from open CV’s GitHub repository .

Step 2 – Extract the pre-built library.

Step 3 - Add open CV’s bin directory to path.

* **Steps to install visual studio community.**

Step 1 - Make sure your computer is ready for **Visual Studio**.

Step 2 - Download latest version of **Visual Studio**.

Step 3 - **Install** the **Visual Studio** installer.

Step 4 - Choose workloads.

Step 5 - Choose individual components. (Optional)

Step 6 - Select the Installation location.(Optional)

Step 7 - Create a project in visual studio.

Step 8 - Configure your new project.

**Set platform target to x64** - Pre-built binaries are built for x64 Windows platform .

**Add to Include Directories**  - Tell the compiler how the Open CV library  *looks*.

This is done by providing a path to the header files

(build/include).

**Add to Library Directories** - Tell the linker where it can find the lib files for

different modules.

**Add Additional Dependencies**- List .lib files for different modules. Note that we’re

only going to list a single all-in-one file

open CV\_ world.

* **Functions /Methods used in this project are:**

1. **Headerfile**

#include<opencv2/opencv.hpp>

#include <iostream

**2. imread()**

This function has to be first since it is essential to starting your project with an image. Use the function **imread()** to read an image. The image should be in the working directory or a full path of image should be given..

**3.imshow()**

Use the function **imshow()** to display an image in a window. The window automatically fits to the image size.

First argument is a window name which is a string. second argument is our image. You can create as many windows as you wish, but with different window names.

**4.Point2f ()**

This function gives the point from where to rotate the image.

**5.getrotateMatrix2D()**

This function is used to get rotation matrix for rotating the image around its center in pixel coordinates.

The function has three input parameters:

Center: rotation center of the picture

Angle: rotation angle

Scale: the scale of the rotated image compared to the original

This function is mainly used to obtain the rotation matrix of the image around a certain point. This function requires three parameters, the center of rotation, the angle of rotation, and the zoom ratio of the rotated image. Returns a 2\*3 matrix, mainly used for warpAffine() Affine transformation.

6. **wrapAffine()**

This function mainly uses Transformation matrix M .For image transformation such as rotation, affine, translation, etc., we only need to provide a 2\*3 transformation matrix M to transform the image. It is generally andcv2.getRotationMatrix2Dwithcv.GetAffineTransformThe two functions are used together. These two functions are used to obtain the transformation matrix M, so that we don't need to set M ourselves.

wrapAffine takes 4 parameters . First is source image file , second is destination image file , third parameter is mat object that is output after rotating with some point taking into reference and last is taking the size.

warpAffine(src, M, dsize, dst, flags, borderMode, borderValue)

**7.rotatedRect()**

The class represents rotated (i.e. not up-right) rectangles on a plane.

Each rectangle is specified by the center point (mass center), length of each side (represented by [**Size2f**](https://docs.opencv.org/3.4/dc/d84/group__core__basic.html#gab34496d2466b5f69930ab74c70f117d4) structure) and the rotation angle in degrees.

Input parameters are:

|  |  |
| --- | --- |
| **center** | : The rectangle mass center. |
| **Size:** | Width and height of the rectangle. |
| **Angle:** | The rotation angle in a clockwise direction. When the angle is 0, 90, 180, 270 etc., the rectangle becomes an up-right rectangle. |

**8.boundingRect2f()**

returns the minimal (exact) floating point rectangle containing the rotated rectangle, not intended for use with images.