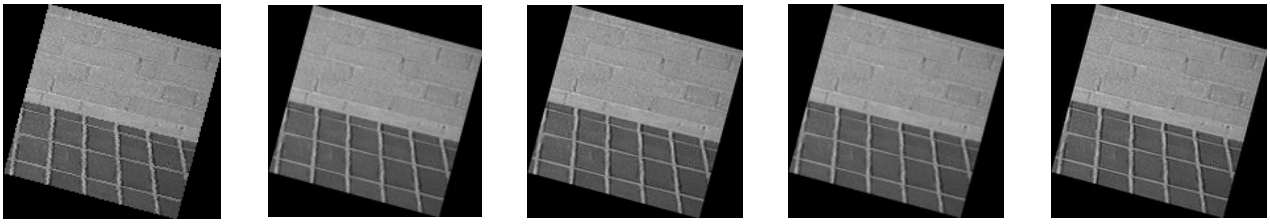
DIP第二章作业_苏智龙

一、图像旋转

图像旋转用OpenCV，使用了5种插值方法：最近邻插值法、双线性插值法 、双三次插值 、基于区域的插值和兰索思插值。代码见文末。

不同插值方法旋转后输出的图片见图1：



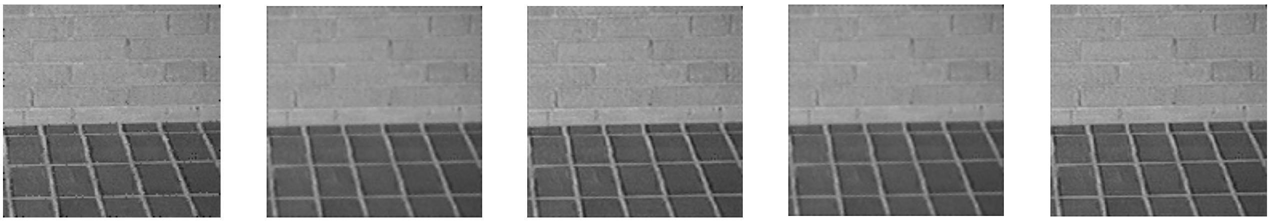
最近邻插值法　双线性插值法　双三次插值　基于区域的插值　兰索思插值

图1 不同插值方法旋转后输出的图片

二、插值法的比较

用旋转后又旋转回来图片和原图片的相似度来说明插值法的效果，比较用了余弦相似度和基于直方图的方法，图片相似度的比较代码见文末(python)。

不同插值方法旋转回来的图片见图2：



最近邻插值法　双线性插值法　双三次插值　基于区域的插值　兰索思插值

图2 不同插值方法旋转回来的图片

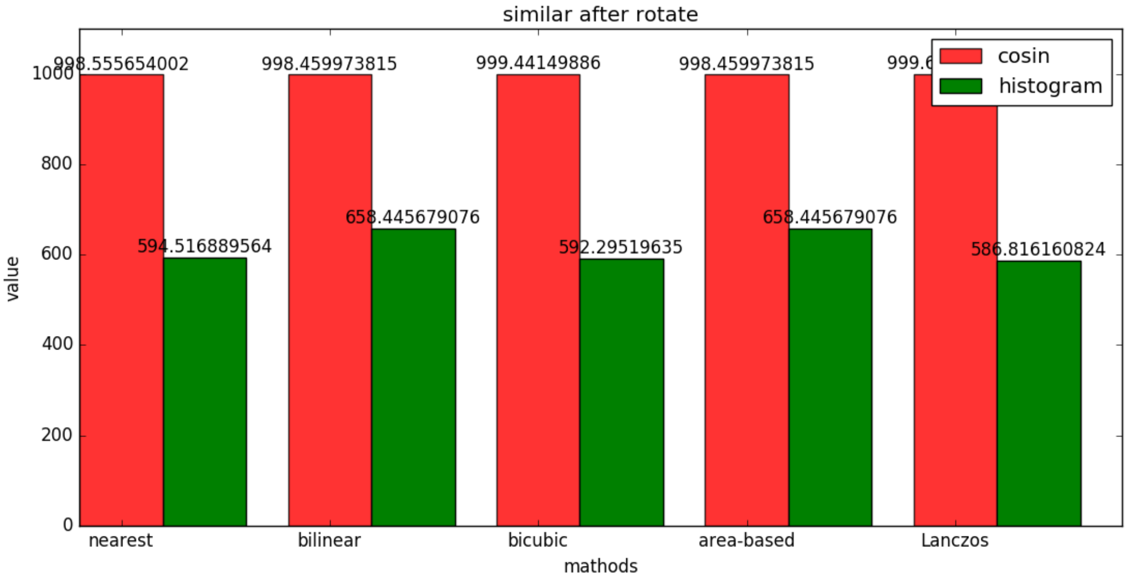


图3 不同插值方法旋转后的相似度比较

三、实验结果

由图3可知，当使用余弦相似度来比较图片的相似性时，兰索思插值效果最好，双三次插值次之。当使用基于直方图的比较方法时，双线插值法效果最好，基于区域的插值效果次之。

四、代码

c++代码：

```
#include<opencv2/core/core.hpp>
#include<opencv2/imgproc/imgproc.hpp>
#include<opencv2/highgui/highgui.hpp>
#include<iostream>
#include<math.h>
#include<fstream>
#include<string>

using namespace cv;

#define SCALE 1   //缩放比例

Mat imgRotate(Mat srcMat,int angle,int interpolation)
{
    double pi_angle = abs(angle * CV_PI / 180);//cos sin 函数需要用弧度值
    double w = 0., h = 0., w_r = 0., h_r = 0.;

    h = srcMat.rows;
    w = srcMat.cols;
    w_r = w*cos(pi_angle) + h*sin(pi_angle);//输出图像的宽
    h_r = h*cos(pi_angle) + w*sin(pi_angle);//输出图像的高

    Mat tempImg(h_r, w_r, srcMat.type(), Scalar(0));
    int roi_x = w_r / 2 - srcMat.cols / 2;//roi左上角的x坐标
    int roi_y = h_r / 2 - srcMat.rows / 2;//roi左上角的y坐标
    Rect roiRect(roi_x, roi_y, srcMat.cols, srcMat.rows);//roi矩形
    Mat tempImgRoi(tempImg, roiRect);//tempImg的中间部分，与原Mat关联，并不会创建一个新Mat
    srcMat.copyTo(tempImgRoi);//将原图复制到tempImg的中心

    Point2f center(w_r / 2, h_r / 2);//旋转中心
    Mat trans = getRotationMatrix2D(center, angle, SCALE);//计算旋转的仿射变换矩阵

    std::cout << "变换矩阵：" << std::endl;
    std::cout << trans.at<double>(0, 0) << ","
               << trans.at<double>(0, 1) << ","
               << trans.at<double>(0, 2) << ","
               << std::endl;
    std::cout << trans.at<double>(1, 0) << ","
               << trans.at<double>(1, 1) << ","
               << trans.at<double>(1, 2) << ","
               << std::endl;

    Mat dstMat;//旋转后的图像
    warpAffine(tempImg, dstMat, trans, Size(w_r, h_r), interpolation);//仿射变换

    return dstMat;

}

Mat imgRotateBack(Mat origin ,Mat src, int angle, int interpolation)
{
    double w = 0., h = 0.;

    h = src.rows;
    w = src.cols;

    Point2f center(w / 2, h / 2);//旋转中心
```

```
      Mat trans = getRotationMatrix2D(center, angle, SCALE);//计算旋转的仿射变换矩阵

      std::cout << "变换矩阵：" << std::endl;
      std::cout << trans.at<double>(0, 0) << ","
                << trans.at<double>(0, 1) << ","
                << trans.at<double>(0, 2) << ","
                << std::endl;
      std::cout << trans.at<double>(1, 0) << ","
                << trans.at<double>(1, 1) << ","
                << trans.at<double>(1, 2) << ","
                << std::endl;

      Mat dst_2;//旋转后的图像
      warpAffine(src, dst_2, trans, Size(w, h), interpolation);//仿射变换

      int roi_x = w / 2 - origin.cols / 2;//roi左上角的x坐标
      int roi_y = h / 2 - origin.rows / 2;//roi左上角的y坐标
      Rect roiRect(roi_x, roi_y, origin.cols, origin.rows);//roi矩形
      Mat tempImgRoi(dst_2, roiRect);//dst_2的中间部分
      std::cout << "rotate back img_columns: " << tempImgRoi.cols << std::endl;
      std::cout << "rotate back img_rows: " << tempImgRoi.rows << std::endl;
      return tempImgRoi;
}


int main()
{
    std::string path = "/home/su/code/DIP/Hw2/";
    std::string savePath = "opt/";
    Mat origin = imread(path+"Chapter2_1.pgm", 0);

    for(int interpolation = 0; interpolation < 5; interpolation++)
    {
        //旋转角度(正值表示逆时针旋转)
        int angle_1 = -15;
        int angle_2 = 15;

        Mat rotatedImg = imgRotate(origin,angle_1,interpolation);
        Mat rotateBack = imgRotateBack(origin, rotatedImg, angle_2, interpolation);
        // 保存图像
        imwrite(path + savePath + "rotated_"+std::to_string(interpolation)+".jpg",
rotatedImg);
        imwrite(path + savePath + "rotatedBack_"+std::to_string(interpolation)+".jpg",
rotateBack);
    }

      waitKey(0);
      return 0;
}
```

python代码：

```
# -*- coding: utf-8 -*-

import math
from PIL import Image
from scipy.misc import imread
import numpy as np

import matplotlib.pyplot as plt
import matplotlib

#cos
def get_thumbnail(image, size=(132, 135), greyscale=False):
    image = image.resize(size, Image.ANTIALIAS)
    if greyscale:
        image = image.convert('L')
    return image
```

```
def image_similarity_vectors_via_numpy(image1, image2):
    image1 = get_thumbnail(image1)
    image2 = get_thumbnail(image2)
    images = [image1, image2]
    vectors = []
    norms = []
    for image in images:
        vector = []
        for pixel_tuple in image.getdata():
            vector.append(np.average(pixel_tuple))
        vectors.append(vector)
        norms.append(np.linalg.norm(vector, 2))
    a, b = vectors
    a_norm, b_norm = norms
    res = np.dot(a / a_norm, b / b_norm)
    return res
#直方图
def make_regalur_image(img, size = (132, 135)):
    return img.resize(size).convert(‘RGB')

def hist_similar(lh, rh):
    assert len(lh) == len(rh)
    return sum(1 - (0 if l == r else float(abs(l - r))/max(l, r)) for l, r in zip(lh, 
rh))/len(lh)

def calc_similar(li, ri):
    return hist_similar(li.histogram(), ri.histogram())

if __name__ == '__main__':
     image1 = Image.open('/home/su/code/DIP/Hw2/Chapter2_1.pgm')
     cmp_list = {}
     for x in xrange(5):
           image2 = Image.open('/home/su/code/DIP/Hw2/opt/rotatedBack_%s.jpg'%x)
           cosin = image_similarity_vectors_via_numpy(image1, image2)
           histogram = calc_similar(image1, image2)
           cmp_list['%scosin'%x] = cosin
           cmp_list['%shistogram'%x] = histogram
     #设置中文字体和负号正常显示
     matplotlib.rcParams['font.sans-serif'] = ['SimHei']
     matplotlib.rcParams['axes.unicode_minus'] = False
     #横坐标刻度显示值
     label_list = ['nearest', 'bilinear', 'bicubic', 'area-based', 'Lanczos']
     #纵坐标值1
     num_list1 = [1000*cmp_list[‘0cosin’],1000*cmp_list[‘1cosin'], 
1000*cmp_list['2cosin'],1000*cmp_list['3cosin'],1000*cmp_list['4cosin']]
     num_list2 = [1000*cmp_list['0histogram'],1000*cmp_list['1histogram'], 
1000*cmp_list['2histogram'],1000*cmp_list['3histogram'],1000*cmp_list['4histogram']]
     #纵坐标值2
     x = range(len(num_list1))
     #绘制条形图
     rects1 = plt.bar(left=x, height=num_list1, width=0.4, alpha=0.8, color='red', 
label="cosin")
     rects2 = plt.bar(left=[i + 0.4 for i in x], height=num_list2, width=0.4, 
color='green', label="histogram")
     plt.ylim(0, 1100)     #y轴取值范围
     plt.ylabel("value")
     plt.xticks([index + 0.2 for index in x], label_list)
     plt.xlabel("mathods")
     plt.title("similar after rotate")
     plt.legend()     # 设置题注
     # 编辑文本
     for rect in rects1:
         height = rect.get_height()
         plt.text(rect.get_x() + rect.get_width() / 2, height+1, str(height), 
ha="center", va="bottom")
     for rect in rects2:
         height = rect.get_height()
         plt.text(rect.get_x() + rect.get_width() / 2, height+1, str(height), 
ha="center", va="bottom")
     plt.show()
```