计算机科学与技术 学院

计算机视觉 课程实验报告

学号：	姓名：	班级： 智能

实验题目：图像仿射变换与图像变形

实验内容：

2.1 设计一个函数 WarpAffine，可以对图像进行任意的二维仿射变换（用 2*3 矩阵表示）：

采用双线性插值进行重采样；

可以只考虑输入图像为 3 通道，8 位深度的情况；

函数接口可以参考 OpenCV 的 warpAffine 函数

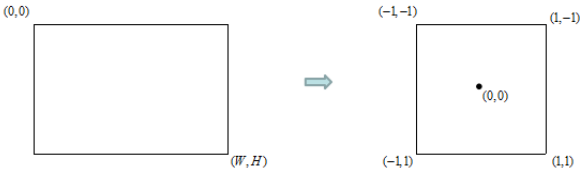
调用 WarpAffine，实现绕任意中心的旋转函数 Rotate

2.2 记[x’，y’]=f([x，y])为像素坐标的一个映射，实现 f 所表示的图像形变。f 的逆映射为：

$$[x,y]=f^{-1}([x',y'])=\begin{cases}[x',y'] & \text{if } r\geq 1\\ [\cos(\theta)x'-\sin(\theta)y',\sin(\theta)x'+\cos(\theta)y'] & \text{otherwise}\end{cases}$$

其中：$r=\sqrt{x'^2+y'^2}$ $\theta=(1-r)^2$

$[x,y],[x',y']$ 都是中心归一化坐标，请先进行转换；



$$x'=\frac{x-0.5W}{0.5W} \qquad y'=\frac{y-0.5H}{0.5H}$$

实验过程中遇到和解决的问题：

（记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）

2.1.1 参考 opencv 定义函数接口等，如下：

```
void my_warpAffine(const Mat &src,Mat &dst,double arr[][3]){ ... }

void CalcRotationMatrix(double arr[2][3],int x,int y,double angle){ ... }
```

my_warpAffine，可以对图像进行任意的二维仿射变换（用 2*3 数组表示矩阵），CalcRotationMatrix 函数得到进行指定变换的数组；

//注意二维数组应定义成 double 类型

2.1.2 对于绕任意中心进行旋转，需要求取正变换的逆矩阵，正变换矩阵如下：

$$\begin{pmatrix} \cos\theta & -\sin\theta & -c_x\cos\theta + c_y\sin\theta + c_x \\ \sin\theta & \cos\theta & -c_x\sin\theta - c_y\cos\theta + c_y \\ 0 & 0 & 1 \end{pmatrix}$$

求取逆矩阵如下：

其中令 $t1 = -c_x\cos\theta + c_y\sin\theta + c_x$

$t2 = -c_x\sin\theta - c_y\cos\theta + c_y$

逆矩阵为：

```
[cos(θ) ]   [sin(θ)]  [-sin(θ)*t2-cos(θ)*t1]
[-sin(θ)]   [cos(θ)]  [-cos(θ)*t2+sin(θ)*t1]
[   0   ]   [   0  ]  [          1          ]
```

2.1.3 而后设计双线性插值的实现：

```
float bilinear(float a, float b, float c, float d, float dx, float dy)
{
    float  h1=a+dx*(b-a);          // = (1-dx)*a + dx*b
    float  h2=c+dx*(d-c);
    return h1+dy*(h2-h1);
}
```

基于老师的课件，代码如下，由于映射辉源图坐标后可能会发生越界，需注意处理：异常如下：

```
template<typename _Tp> inline
const _Tp& Mat::at(int i0, int i1) const
{
    CV_DbgAssert(dims <= 2);
    CV_DbgAssert(data);
    CV_DbgAssert((unsigned)i0 < (unsigned)size.p[0]);
    CV_DbgAssert((unsigned)(i1 * DataType<_Tp>::channels < (unsigned)(size.p[1] * channels()));
    CV_DbgAssert(CV_ELEM_SIZE1(traits::Depth<_Tp>::val
    return ((const _Tp*)(data + step.p[0] * i0))[i1];
}

template<typename _Tp> inline
_Tp& Mat::at(Point pt)
{
    CV_DbgAssert(dims <= 2);
    CV_DbgAssert(data);
    CV_DbgAssert((unsigned)pt.y < (unsigned)size.p[0])
```

未经处理的异常

0x00007FFFA2C03C58 处(位于 cv03.exe 中)有未经处理的异常:
Microsoft C++ 异常: cv::Exception，位于内存位置
0x0000003BC38FE8E0 处。

复制详细信息

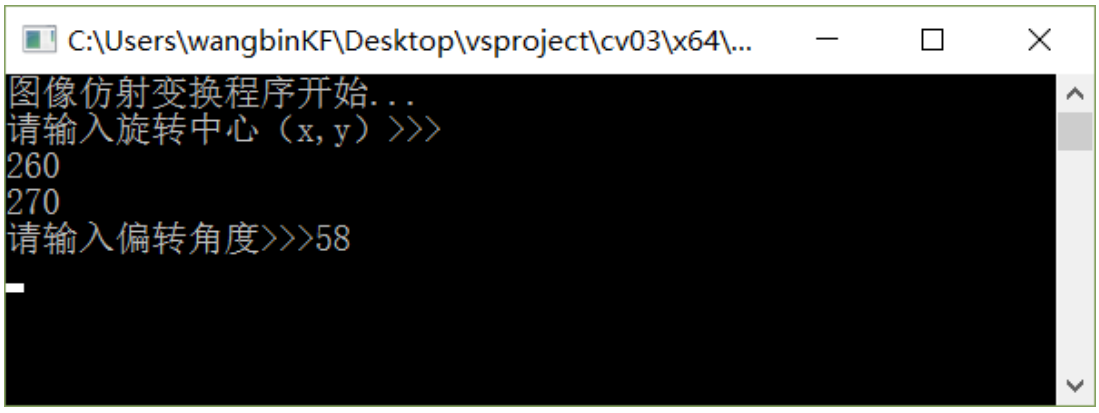
异常设置

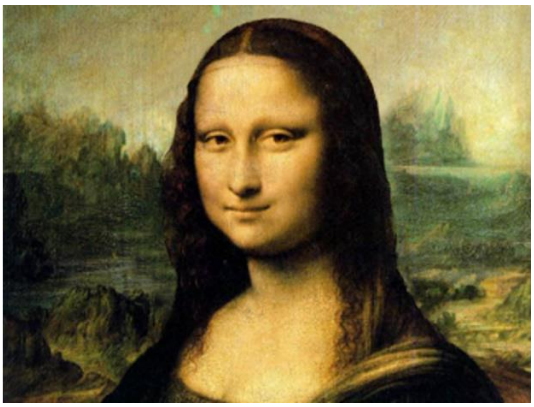
```
        if (x_ < 0 || y_ < 0||x_>=src.rows||y_>=src.cols) {
                    for (int c = 0; c < 3; c++) {
                        dst.at<Vec3b>(x, y)[c] = saturate_cast<uchar>(0);
                    }
                }
                else {
                    for (int c = 0; c < 3; c++) {
```

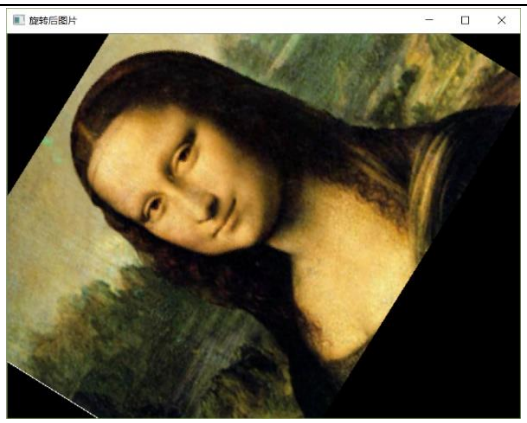
```
                    //计算双线性插值
                    //左上角坐标（X1，Y1）
                    int X1 = (int)x_;
                    int Y1 = (int)y_;
                    //四个顶点像素值
                    //注意访问越界
                    if (X1 == (src.rows - 1) || Y1 == (src.cols -
1)||X1==0||Y1==0) {
                        dst.at<Vec3b>(x, y)[c] =
saturate_cast<uchar>(src.at<Vec3b>(X1, Y1)[c]);
                    }
                    else {
                        int aa = src.at<Vec3b>(X1, Y1)[c];
                        int bb = src.at<Vec3b>(X1, Y1+1)[c];
                        int cc = src.at<Vec3b>(X1+1, Y1)[c];
                        int dd = src.at<Vec3b>(X1+1, Y1+1)[c];

                        double dx = x_ - X1;
                        double dy = y_ - Y1;
                        double h1 = aa + dx * (bb - aa);
                        double h2 = cc + dx * (dd - cc);
                        dst.at<Vec3b>(x, y)[c] = saturate_cast<uchar>(h1+dy*(h2-
h1));
                    }
                }
```

2.1.4 测试结果如下：







2.2.1 函数设计：

函数设计：

```
Mat changeShape(const Mat &src)
```

注意中心归一化后和还原：

```
x_ = (x_ + 1.0)*((row - 1) / 2.0);
y_ = (y_ + 1.0)*((col - 1) / 2.0);
```

结果测试如下：