山东大学 计算机科学与技术 学院

计算机视觉 课程实验报告

学号：201800130086	姓名： 徐鹏博	
实验题目：几何变换与变形		

实验过程中遇到和解决的问题：

1.图像变形

先进行坐标归一化:

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

再求出 r 和θ:

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

根据 f 的逆映射:

$$[x_{new},y_{new}]\begin{cases}[x',y'] & r\geq 1\\ [cos(\theta)x'-sin(\theta)y',sin(\theta)x'+cos(\theta)y'] & otherwise\end{cases}$$

再还原坐标:

$$X=(x_{new}+1)*0.5H$$
$$Y=(y_{new}+1)*0.5H$$

关键代码:

```
for(int i=0;i<H;i++){
    for(int j=0;j<W;j++){
        p=(double)(i-0.5*H)/(0.5*H);
        q=(double)(j-0.5*W)/(0.5*W);
        r=sqrt(p*p+q*q); theta=(1-r)*(1-r);
        if(r>=1) {
            op=p; oq=q;
        }
        else{
            op=cos(theta)*p-sin(theta)*q;
            oq=cos(theta)*q+sin(theta)*p;
        }
        I=int ((op+1)*0.5*H);
        J=int ((oq+1)*0.5*W);
        Out.at<Vec3b>(i,j)[0]=img.at<Vec3b>(I,J)[0];
        Out.at<Vec3b>(i,j)[1]=img.at<Vec3b>(I,J)[1];
        Out.at<Vec3b>(i,j)[2]=img.at<Vec3b>(I,J)[2];
    }
}
```

效果:

双线性重采样：

$$I_a = (1-\alpha)I_1 + \alpha I_2$$
$$I_b = (1-\alpha)I_3 + \alpha I_4$$
$$I = (1-\beta)I_a + \beta I_b$$
$$I = (1-\beta)((1-\alpha)I_1 + \alpha I_2) + \beta((1-\alpha)I_3 + \alpha I_4)$$
$$I = (1-\alpha)(1-\beta)I_1 + \alpha(1-\beta)I_2 + (1-\alpha)\beta I_3 + \alpha\beta I_4$$

双线性重采样需要先根据变化比例(newH=a H,newW=b W,，原图和新图长宽不变时 a,b=1)

换算出新图像中像素点(x,y)在原图中的位置，采用(x,y)所在最近的四个整数格点进行插值,另外需要注意图像 i,j 取值范围不能溢出。

(i,j),　　(i,j+1)

　　(x,y)

(i+1,j)　　(i+1,j+1)

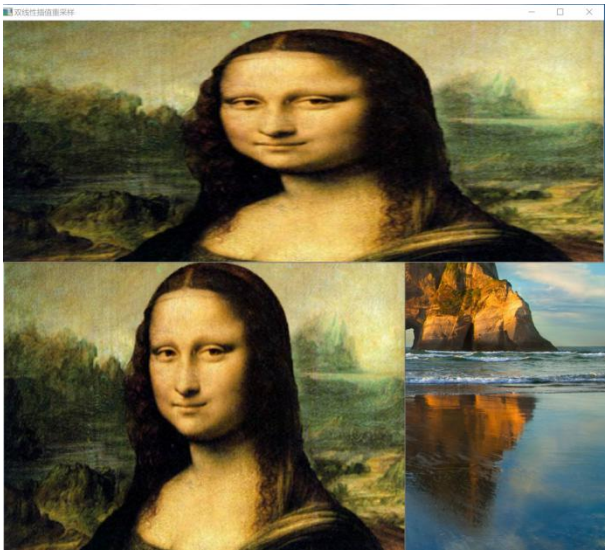
有:

$$I_1 = (i,j) = (floor(x), floor(y))$$
$$\alpha = x - floor(x), \beta = y - floor(y)$$
$$I_1 = (i,j); I_2 = (i,j+1); I_3 = (i+1,j); I_4 = (i+1,j+1)$$

```
for(int i=0;i<nH;i++){
    for(int j=0;j<nW;j++){
        x=(double)i/a; y=(double)j/b;
        if(x>=H-1) x=H-1; if(x<=0) x=0;
        if(y>=W-1) y=W-1; if(y<=0) y=0;
        I=floor(x);J=floor(y);
        if((x==0&&y==0)||(x==H-1&&y==W-1)||(x==0&&y==W-1)||(x==H-1&&y==0)){
            Out.at<Vec3b>(i,j)[0]=img.at<Vec3b>(I,J)[0];
            Out.at<Vec3b>(i,j)[1]=img.at<Vec3b>(I,J)[1];
            Out.at<Vec3b>(i,j)[2]=img.at<Vec3b>(I,J)[2];
        }
        else if(x==0||x==H-1){
            Out.at<Vec3b>(i,j)[0]=img.at<Vec3b>(I,J)[0]+(img.at<Vec3b>(I,J)[0]-img.at<Vec3b>(I,J+1)[0])*(y-J);
            Out.at<Vec3b>(i,j)[1]=img.at<Vec3b>(I,J)[1]+(img.at<Vec3b>(I,J)[1]-img.at<Vec3b>(I,J+1)[1])*(y-J);
            Out.at<Vec3b>(i,j)[2]=img.at<Vec3b>(I,J)[2]+(img.at<Vec3b>(I,J)[2]-img.at<Vec3b>(I,J+1)[2])*(y-J);
        }
```

```
            else if(y==0||y==W-1){
                Out.at<Vec3b>(i,j)[0]=img.at<Vec3b>(I,J)[0]+(img.at<Vec3b>(I,J)[0]-img.at<Vec3b>(I+1,J)[0])*(x-I);
                Out.at<Vec3b>(i,j)[1]=img.at<Vec3b>(I,J)[1]+(img.at<Vec3b>(I,J)[1]-img.at<Vec3b>(I+1,J)[1])*(x-I);
                Out.at<Vec3b>(i,j)[2]=img.at<Vec3b>(I,J)[2]+(img.at<Vec3b>(I,J)[2]-img.at<Vec3b>(I+1,J)[2])*(x-I);
            }
            else{
                Out.at<Vec3b>(i,j)[0]=   img.at<Vec3b>(I,J)[0]*(I+1-x)*(J+1-y)
                                       + img.at<Vec3b>(I,J+1)[0]*(I+1-x)*(y-J)
                                       + img.at<Vec3b>(I+1,J)[0]*(x-I)*(J+1-y)
                                       + img.at<Vec3b>(I+1,J+1)[0]*(x-I)*(y-J);
                Out.at<Vec3b>(i,j)[1]=   img.at<Vec3b>(I,J)[1]*(I+1-x)*(J+1-y)
                                       + img.at<Vec3b>(I,J+1)[1]*(I+1-x)*(y-J)
                                       + img.at<Vec3b>(I+1,J)[1]*(x-I)*(J+1-y)
                                       + img.at<Vec3b>(I+1,J+1)[1]*(x-I)*(y-J);
                Out.at<Vec3b>(i,j)[2]=   img.at<Vec3b>(I,J)[2]*(I+1-x)*(J+1-y)
                                       + img.at<Vec3b>(I,J+1)[2]*(I+1-x)*(y-J)
                                       + img.at<Vec3b>(I+1,J)[2]*(x-I)*(J+1-y)
                                       + img.at<Vec3b>(I+1,J+1)[2]*(x-I)*(y-J);
            }
        }
}
```

效果:



2.电子哈哈镜

设计函数:

先对坐标归一化:

$$p = \frac{x-0.5W}{0.5W} \quad q = \frac{y-0.5H}{0.5H}$$

再设定变换范围(double)R,变换强度 t,求出归一化坐标(p,q)距离原点距离 r。

$$r = \sqrt{p^2 + q^2}$$

变换函数 f:

$$[x_{new}, y_{new}] \begin{cases} [p, q] & r \geq R \\ [p \cdot (\frac{r}{R})^t - 1, q \cdot (\frac{r}{R})^t - 1] & otherwise \end{cases}$$

再还原坐标:

$$X = (x_{new} + 1) * 0.5H$$
$$Y = (y_{new} + 1) * 0.5H$$

```
for(int i=0;i<H;i++){
    for(int j=0;j<W;j++){
        p=(double)(i-0.5*H)/(0.5*H);
        q=(double)(j-0.5*W)/(0.5*W);
        r=sqrt(p*p+q*q);
        if(r>=R) {
            op=p; oq=q;
        }
        else{
            op=p*powf(r/R,t-1);   oq=q*powf(r/R,t-1);
        }
        I=int ((op+1)*0.5*H);
        J=int ((oq+1)*0.5*W);
        Out.at<Vec3b>(i,j)[0]=img.at<Vec3b>(I,J)[0];
        Out.at<Vec3b>(i,j)[1]=img.at<Vec3b>(I,J)[1];
        Out.at<Vec3b>(i,j)[2]=img.at<Vec3b>(I,J)[2];
    }
}
```

图片效果:

视频变换和保存:

```
VideoCapture cap;
cap.open("D:\\Codes\\CV\\CV_works\\CV-E3\\video.MP4");
//cap.open(0);
double rate = cap.get(CAP_PROP_FPS);
```

```
Size capsize=Size(cap.get(CAP_PROP_FRAME_WIDTH),cap.get(CAP_PROP_FRAME_HEIGHT));
printf("fps=%lf",rate);
double fps=(rate>0)?rate:25.0;
VideoWriter outputVideo;
outputVideo.open("D:\\Codes\\CV\\CV_works\\CV-E3\\out.mp4",VideoWriter::fourcc('m','p','4','v'),fps,capsize,true);
if(!outputVideo.isOpened())
{
    cout<< "Error : fail to open video writer\n"<<endl;
    return -1;
}
while (1)
{
    Mat frame,Out;
    cap >> frame;
    if (frame.empty())
    {
        cout << "Finish" << endl; break;
    }
    Out=Fun3(frame,1.9);
    imshow("Out video",Out);
    outputVideo.write(Out);
    if (cv::waitKey(20)=='0') break;
}
cap.release();
outputVideo.release();
```

结果分析与体会：

第一道题目的变换其实是图片中以四条边中点为长短半轴顶点的椭圆区域的围绕中心的旋转变换，并且由于 r 不同导致θ不同，越靠近中心的地方旋转幅度越大。

另外在进行重采样的时候需要注意四条边上的像素变换。

哈哈镜的变换其实有很多种方式，我采取的是使用 y=x^t 在 (0, 1) 之间的函数效果进行局部变换。