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：201900130151	姓名： 莫甫龙	
实验题目：几何变换与变形		
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。） 实验 2.1：图像变形 在做这个实验的时候，因为习惯了用坐标系中的 oxy 坐标系，而在这个里面 x、y 表示的量是相反的，用的不是很习惯，所以在后面的公式中导致写出来的很多都是相反的，导致最后画出来的图像只有人物的轮廓是对的，其它地方全部都是雪花纹（就是老电视机没有信号的那种）。后来我将它就按照 x 表示行，y 表示列来写，就很好地解决了这个问题，并且在二次线性插值的时候，我刚开始用的是最后得到的那一个公式，我想也有可能是中间的某一个地方写错了，所以我在后面的更改中按照步骤一步步来写。 而在解决完上面那个问题以后，又产生了一个问题，就是在函数所表示的范围内，图像的颜色是不对的，就是那种偏向于白色的色调，有点像雪花纹。后面我又仔细检查了自己的代码，又和同学的比较了一下，发现这个问题是因为没有将归一化的点还原所导致的。我只保留了逆变换后的像素的左上角的那一个还原归一化的点，却没有将逆变换后的那个点还原归一化以后的点保留下来，导致后面的计算出现了问题，后面我将这些点保留下来就将问题解决了。		

```cpp
void fun1(const Mat &image) {
    Mat image2;
    image2 = Mat::zeros(image.size(), image.type());
    int H = image.rows, W = image.cols;
    for (int x = 0; x < H; x++) {
        for (int y = 0; y < W; y++) {
            double x_, y_;
            double theta, r;
            double x1, y1;
            x_ = (double) (x - 0.5 * H) / (0.5 * H);    //归一化
            y_ = (double) (y - 0.5 * W) / (0.5 * W);
            r = sqrt( x: x_ * x_ + y_ * y_);
            theta = (1 - r) * (1 - r);
            if (r >= 1) {
                x1 = x_;
                y1 = y_;
            } else {
                x1 = cos(theta) * x_ - sin(theta) * y_;
                y1 = cos(theta) * y_ + sin(theta) * x_;
            }
            x1 = (x1 + 1) * H *0.5;      //还原归一化的点
            y1 = (y1 + 1) * W *0.5;
            int X = (int) x1;           //左上角的点
            int Y = (int) y1;
            double a, b;
            a = x1 - X;                 //求出a和b
            b = y1 - Y;
            for (int c = 0; c < 3; c++) {
                int p1, p2, p3, p4;
                p1 = image.at<Vec3b>(X, Y)[c];
                p2 = image.at<Vec3b>( i0: X + 1, Y)[c];
                p3 = image.at<Vec3b>(X,  i1: Y + 1)[c];
                p4 = image.at<Vec3b>( i0: X + 1,  i1: Y + 1)[c];
                double pa, pb;
                pa = (1 - a) * p1 + a * p2;
                pb = (1 - a) * p3 + a * p4;
                image2.at<Vec3b>(x, y)[c] = saturate_cast<uchar>( v: (1 - b) * pa + b * pb);
            }
        }
    }
    imshow( winname: "E2",  mat: image2);
    waitKey();
```

实验 2.2：电子哈哈镜

这个实验比较简单，只需要套用上面的那个代码就行，就是因为要调用摄像头和保存视频，所以我上网找了一个模板，而因为我想设计一个凸透镜的效果，所以我设计了一个函数来实现这个效果：

```
r = sqrt( x_ * x_ + y_ * y_)*1.001;
if (r >= 1) {
    x1 = x_;
    y1 = y_;
} else {
    x1 = x_*r*r*r;
    y1 = y_*r*r*r;
}
```

这个就是那个函数。

代码如下：

```cpp
Mat fun2(const Mat &image) {
    Mat image2;
    image2 = Mat::zeros(image.size(), image.type());
    int H = image.rows, W = image.cols;
    for (int x = 0; x < H; x++) {
        for (int y = 0; y < W; y++) {
            double x_, y_;
            double theta, r;
            double x1, y1;
            x_ = (double) (x - 0.5 * H) / (0.5 * H);    //归一化
            y_ = (double) (y - 0.5 * W) / (0.5 * W);
            r = sqrt( X: x_ * x_ + y_ * y_)*1.001;
            if (r >= 1) {
                x1 = x_;
                y1 = y_;
            } else {
                x1 = x_*r*r*r;
                y1 = y_*r*r*r;
            }
            x1 = (x1 + 1) * H *0.5;      //还原归一化的点
            y1 = (y1 + 1) * W *0.5;
            int X = (int) x1;           //左上角的点
            int Y = (int) y1;
            double a, b;
            a = x1 - X;                 //求出a和b
            b = y1 - Y;
            for (int c = 0; c < 3; c++) {
                int p1, p2, p3, p4;
                p1 = image.at<Vec3b>(X, Y)[c];
                p2 = image.at<Vec3b>( i0: X + 1, Y)[c];
                p3 = image.at<Vec3b>(X,  i1: Y + 1)[c];
                p4 = image.at<Vec3b>( i0: X + 1,  i1: Y + 1)[c];
                double pa, pb;
                pa = (1 - a) * p1 + a * p2;
                pb = (1 - a) * p3 + a * p4;
                image2.at<Vec3b>(x, y)[c] = saturate_cast<uchar>( v: (1 - b) * pa + b * pb);
            }
        }
    }
    return image2;
```

这个是调用摄像头和保存视频的代码：

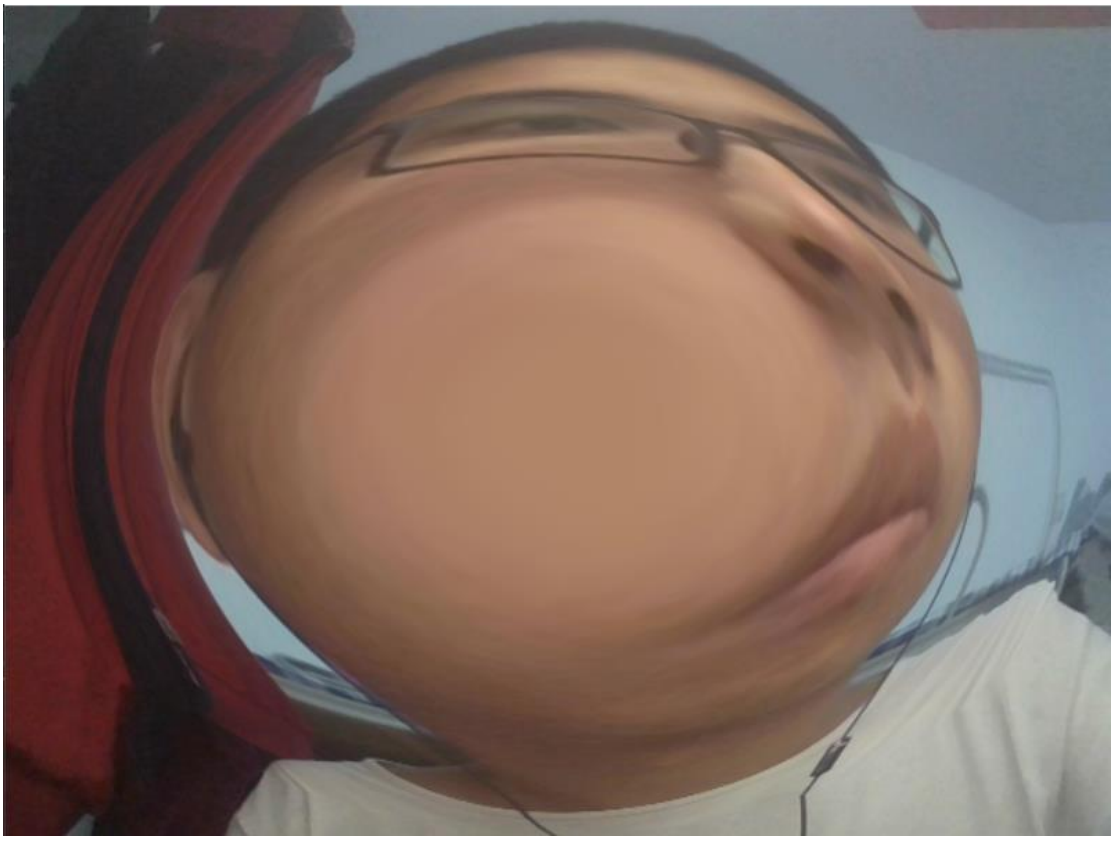
```cpp
void fun3() {
    VideoCapture capture;
    capture.open( index: 0);
    VideoWriter writer( filename: "C:\\Users\\15339\\Desktop\\VideoTest.avi",  fource: CV_FOURCC( c1: 'M',  c2: 'J',  c3: 'P',  c4: 'G'),  fps: 25.0,  frameSize: Size( _width: 640,  _height: 480));
    if (!capture.isOpened()) {
        cout << "Read video Failed !" << endl;
        exit( _Code: 0);
    }
    Mat frame;
    namedWindow( winname: "video test");

    int frame_num = 800;

    for (int i = 0; i < frame_num - 1; ++i) {
        capture >> frame;
        frame = fun2(frame);
        writer << frame;
        imshow( winname: "video test",  mat: frame);
        if (waitKey( delay: 30) == 'q') {
            break;
        }
    }
    destroyWindow( winname: "video test");

    capture.release();
}
```

结果如图：



结果分析与体会：

该实验真的十分有趣，可以将上课的内容转化为代码，真的很有成就感，特别是自己设计了一个凸透镜效果的哈哈镜。