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：201600301304 | 姓名：贾乘兴 | 班级：人工智能16 |
| 实验题目：几何变换与变形 | | |
| 实验内容：   1. 图像仿射变换 2. 仿射变换为非奇异的线性变换，可以用一个2\*3矩阵表示该变换，第三个和第六个参数表示平移，在齐次坐标下可以表示为3\*3矩阵，共六个参数 3. 本次实验即设计一个仿射变换函数，可实现任意的仿射变换，本实验设计了warpAffine1()函数用以实现 4. 该函数首先使用t变换将四个端点进行变换，获得了四个坐标共八个参数，获取行列的最大值最小值做差从而得到一个新的图片大小 5. 利用最小值对新图像的坐标进行逆映射，映射得到在原图的位置，在原图进行插值，本实验采用了较为简单的双线性插值 6. 随机设置了变换矩阵的到了结果如下   a31a.png   1. 利用设计好的warpAffine1()函数，多次调用以实现旋转变换 2. 得到旋转中心（cx，cy），以及旋转角度theta，需要调用三次函数，每次调用函数的矩阵为平移矩阵T1，旋转矩阵T2，平移矩阵T3     实验选用了图像中心，角度选取了arctan（4/3），结果如下：  a31.png  旋转操作实验代码如下：  #include **<iostream>** #include **<opencv2/opencv.hpp>** #include **<cmath>  using namespace** std; **using namespace** cv;  Mat in\_image;  **void** WarpAffine1(**const float** t[2][3]){  **if** (t[0][0]\*t[1][1]-t[0][1]\*t[1][0]==0.0){  **return**;  }  *//定位并确定新图像大小* **int** bor\_y[4];  **int** bor\_x[4];  **int** point[4][2] = {{0,0},{0,in\_image.cols},{in\_image.rows,0},{in\_image.rows,in\_image.cols}};  **for**(**int** k=0;k<4;k++){  bor\_x[k] = point[k][1]\*t[0][0]+point[k][0]\*t[0][1]+t[0][2];*//x* bor\_y[k] = point[k][1]\*t[1][0]+point[k][0]\*t[1][1]+t[1][2];*//y* }  **int** maxy=bor\_x[0];  **int** maxx=bor\_x[0];  **int** miny=bor\_y[0];  **int** minx=bor\_y[0];  **for**(**int** k=1;k<4;k++){  **if** (bor\_y[k]>maxy){  maxy=bor\_y[k];  }  **if** (bor\_y[k]<miny){  miny=bor\_y[k];  }  **if** (bor\_x[k]>maxx){  maxx=bor\_x[k];  }  **if** (bor\_x[k]<minx){  minx=bor\_x[k];  }  }   Mat out\_image = Mat::zeros(Size((maxx-minx+1),(maxy-miny+1)),in\_image.type());   *//双线性插值* **for**(**int** y=0;y<out\_image.rows;y++){  **for**(**int** x=0;x<out\_image.cols;x++){  **for**(**int** k=0;k<3;k++){  **float** y1 = y + miny + 1;  **float** x1 = x + minx + 1;  **float** real\_y = (t[1][0]\*(x1-t[0][2])-t[0][0]\*(y1-t[1][2]))/(t[0][1]\*t[1][0]-t[0][0]\*t[1][1]);  **float** real\_x = (t[0][1]\*(y1-t[1][2])-t[1][1]\*(x1-t[0][2]))/(t[0][1]\*t[1][0]-t[0][0]\*t[1][1]);  **int** sy = (**int**)(real\_y);  **int** sx = (**int**)(real\_x);  **if** ((sx>=0)&&(sy>=0)&&(sx<in\_image.cols-1)&&(sy<in\_image.rows-1)){  **float** h1 = (real\_x-sx)\*in\_image.at<Vec3b>(sy,sx+1)[k]+(sx+1-real\_x)\*in\_image.at<Vec3b>(sy,sx)[k];  **float** h2 = (real\_x-sx)\*in\_image.at<Vec3b>(sy+1,sx+1)[k]+(sx+1-real\_x)\*in\_image.at<Vec3b>(sy+1,sx)[k];  **float** h = (real\_y-sy)\*h2+(sy+1-real\_y)\*h1;  out\_image.at<Vec3b>(y,x)[k] = (**int**)(h+0.5);  }**else**{  out\_image.at<Vec3b>(y,x)[k] = 0;  }  }  }  }  in\_image = out\_image; }  **int** main(){  in\_image = imread(**"/Users/apple/Desktop/a.png"**);  **float** cy = (**float**)in\_image.rows/2;  **float** cx = (**float**)in\_image.cols/2;   **float** t0[2][3] = {{1,0,-cx},{0,1,-cy}};  WarpAffine1(t0);  **float** t1[2][3] = {{0.8,0.6,0},{-0.6,0.8,0}};  WarpAffine1(t1);  **float** t2[2][3] = {{1,0,cx},{0,1,cy}};  WarpAffine1(t2);   imshow(**"out"**,in\_image);  waitKey(0);  imwrite(**"/Users/apple/Desktop/a31.png"**,in\_image);  }   1. 图像变形 2. 通过分析我们可知，原图大小与变换后的图像大小一致，所以可以不选用定位的方法确定图像大小 3. 由归一化公式可知，本实验坐标变换有三步，将新图像坐标归一化，后映射为原图归一化坐标，再逆向反归一化得到原图坐标，进行双线性插值即可得到原图   由与可得       1. 最终得到结果如下：   a32.png  对该效果进行分析，即越靠近图像中心的部分，r越小，theta越大，旋转角度越大，越边缘的位置则旋转效果不明显，可处理一些人物或者目标在中心的图像  实验代码如下：  #include **<iostream>** #include **<opencv2/opencv.hpp>** #include **<cmath>  using namespace** std; **using namespace** cv;  Mat in\_image1; Mat out\_image1;  **void** WarpAffine1(){  out\_image1 = Mat::zeros(in\_image1.size(),in\_image1.type());  *//双线性插值* **for**(**int** y=0;y<out\_image1.rows;y++){  **for**(**int** x=0;x<out\_image1.cols;x++){  **for**(**int** k=0;k<3;k++){  **float** y0 = (**float**)(y - out\_image1.rows/2)/(**float**)(out\_image1.rows/2);  **float** x0 = (**float**)(x - out\_image1.cols/2)/(**float**)(out\_image1.cols/2);  **float** r = sqrt(pow(y0,2)+pow(x0,2));  **if** (r>=1.00){  *//cout<<1<<endl;* **float** yp = y0;  **float** xp = x0;  **float** real\_y = (yp \* in\_image1.rows/2) + in\_image1.rows/2;  **float** real\_x = (xp \* in\_image1.cols/2) + in\_image1.cols/2;   **int** sy = (**int**)(real\_y);  **int** sx = (**int**)(real\_x);  **if** ((sx>=0)&&(sy>=0)&&(sx<in\_image1.cols-1)&&(sy<in\_image1.rows-1)){  **float** h1 = (real\_x-sx)\*in\_image1.at<Vec3b>(sy,sx+1)[k]+(sx+1-real\_x)\*in\_image1.at<Vec3b>(sy,sx)[k];  **float** h2 = (real\_x-sx)\*in\_image1.at<Vec3b>(sy+1,sx+1)[k]+(sx+1-real\_x)\*in\_image1.at<Vec3b>(sy+1,sx)[k];  **float** h = (real\_y-sy)\*h2+(sy+1-real\_y)\*h1;  out\_image1.at<Vec3b>(y,x)[k] = (**int**)(h+0.5);  *//cout<<h<<endl;* }**else**{  out\_image1.at<Vec3b>(y,x)[k] = 0;  *//cout<<0.0<<endl;* }  }**else**{  *//cout<<0<<endl;* **float** theta = pow(1-r,2);  **float** yp = sin(theta)\*x0+cos(theta)\*y0;  **float** xp = cos(theta)\*x0-sin(theta)\*y0;  **float** real\_y = (yp \* in\_image1.rows/2) + in\_image1.rows/2;  **float** real\_x = (xp \* in\_image1.cols/2) + in\_image1.cols/2;   **int** sy = (**int**)(real\_y);  **int** sx = (**int**)(real\_x);  **if** ((sx>=0)&&(sy>=0)&&(sx<in\_image1.cols-1)&&(sy<in\_image1.rows-1)){  **float** h1 = (real\_x-sx)\*in\_image1.at<Vec3b>(sy,sx+1)[k]+(sx+1-real\_x)\*in\_image1.at<Vec3b>(sy,sx)[k];  **float** h2 = (real\_x-sx)\*in\_image1.at<Vec3b>(sy+1,sx+1)[k]+(sx+1-real\_x)\*in\_image1.at<Vec3b>(sy+1,sx)[k];  **float** h = (real\_y-sy)\*h2+(sy+1-real\_y)\*h1;  out\_image1.at<Vec3b>(y,x)[k] = (**int**)(h+0.5);  *//cout<<h<<endl;* }**else**{  out\_image1.at<Vec3b>(y,x)[k] = 0;  *//cout<<0<<endl;* }  }  }  }  } }  **int** main(){  in\_image1 = imread(**"/Users/apple/Desktop/a.png"**);  WarpAffine1();  imshow(**"out"**,out\_image1);  waitKey(0);  imwrite(**"/Users/apple/Desktop/a32.png"**,out\_image1); } | | |
| 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。）   1. 一开始没有搞清楚仿射变换一定是非奇异的，在原坐标的确定上思考了一段时间，认识到可以使用逆映射 2. 对于新图像大小定位问题，最后采用了八个参数的确定的方法 3. 没有及时转float导致图像出现问题 | | |
| 结论分析与体会： 本次实验进一步底层操作了一些图像的变换，认识到很多较为神奇的图像变形往往并不是一个复杂的变换，同时对底层操作的一些算法思考也是非常有意义的 | | |