



日期：2023 年 3 月 24 日

课程：图像处理

实验：图像变换

周次：第 4-5 周

专业：智能科学与技术

姓名：周德峰

学号：21312210

1 实验内容及目的

1.1 实验一：坐标变换

- 给定一张图片，求图片中所有圆形物体对应的在世界坐标系之间的距离

1.2 实验二：仿射变换

- 自行选择图片，分别做出缩放平移，旋转，反射，错切的效果，并写出对应效果的矩阵
- 构建一个合适的仿射矩阵，将任务图矫正，写出构建的过程矩阵

1.3 选做：实验三：图像融合

- 利用仿射变换做出图像融合的效果
- 写出过程矩阵

2 实验相关原理描述

2.1 实验一

坐标变换是指将图像中的像素点在平面坐标系中进行变换，从而实现对图像的几何变换。常见的坐标变换包括平移、旋转、缩放、镜像等设 (x, y) 是原始坐标系下的一个像素点， (x', y') 是目标坐标系下对应的像素点，则坐标变换可以表示为以下矩阵乘法形式：

1, 平移变换

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (1)$$

2, 旋转变换

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2)$$

我们的目标是求解 $[X_w, Y_w, Z_w]$, 通过求解下式可得

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{1}{dX} & 0 & u_0 \\ 0 & \frac{1}{dY} & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{相机内参}} \underbrace{\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}}_{\text{相机外参}} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M_1 M_2 \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = R * \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T$$

2.2 实验二

仿射变换变化包括缩放 (Scale)、平移 (transform)、旋转 (rotate)、反射 (reflection)、错切 (shear mapping), 原来的直线仿射变换后还是直线, 原来的平行线经过仿射变换之后还是平行线, 这就是仿射. 仿射变换的公式为

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3)$$

其中, 矩阵 $\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$ 称为变换矩阵, 它描述了原始坐标系和目标坐标系之间的关系。具体来说,

矩阵中的参数 a, b, c, d, e, f 可以表示出对应的缩放、旋转、平移等变换操作。

例如,当矩阵为 $\begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 时,表示对图像进行等比例缩放,缩放比例为 s ;当矩阵为 $\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$

时,表示对图像进行旋转,旋转角度为 θ ;当矩阵为 $\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$ 时,表示对图像进行平移,平移距离

分别为 t_x 和 t_y 。在实际应用中,可以通过设置变换矩阵的参数,来实现对图像的不同变换操作。需要注意的是,变换后的像素坐标可能不是整数,需要进行插值处理,才能得到最终的图像像素值。

2.3 实验三

此处介绍下图像融合的原理图像融合的原理是将不同图像中的有用信息进行融合,以得到一个更具信息量和可视化效果的图像。

3 实验过程

3.1 实验一

3.1.1 求解外参矩阵

外参矩阵由旋转矩阵 R 和平移矩阵 T 组成,判断世界坐标系相对相机坐标系的位置,计算外参矩阵。根据题目信息可知,平移矩阵只有在 Z 轴方向上有偏移量, R 矩阵则分别绕着 x 轴和 z 轴旋转 90°

```
1 T=[0;0;-2.9];  
R=[0 0 1; 1 0 0; 0 1 0];
```

3.1.2 读入图片及 Z_c 和相机内参

Z_c 为原点距小球中心的水平距离

```
I=[650.1821 0 315.8990; 0 650.5969 240.3104; 0 0 1.0000];  
2 img=imread("image\1.jpg");  
zc1=[37.5 64];
```

3.1.3 彩色图转灰度图

判断目标小球相对背景的是明是暗，利用函数`im2gray`对图片进行灰度化处理

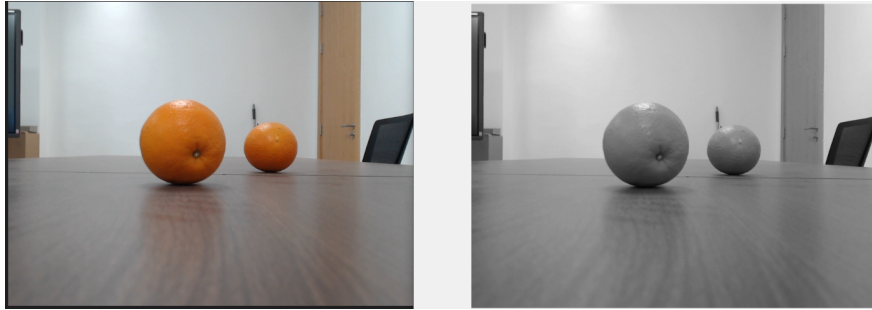


图 1: 原始图片与灰度化图片

3.1.4 识别小球的中心位置

利用`imfindcircles`函数即可在图像找到圆形物体的位置，并且可以通过调整敏感度来调节识别精度利用`viscircles`函数可以画图像上圆的轮廓，需要输入通过上步得到的两个参数 `centers`, `radii`，另外可以设置轮廓的颜色和线条样式，例如：'`Color`','`b`', '`LineStyle`' 等。

```
[circle, radius]=imfindcircles(img_g, [40 200], 'ObjectPolarity','dark','Sensitivity',0.97);  
2 viscircles(circle, radius, 'Color','b');
```

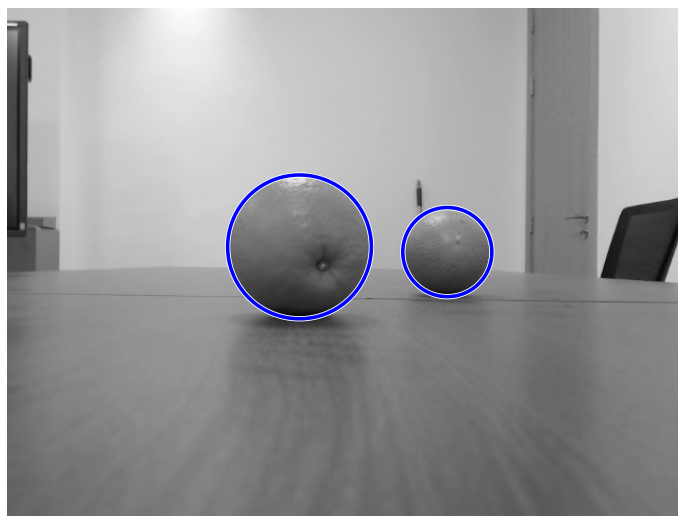


图 2: 识别小球中心位置及轮廓

3.1.5 转齐次坐标并求取小球中心在真实世界中的坐标

由于 $M1$ 是 3×4 的矩阵, 不能求逆, 所以更换求解思路我们是已知像素坐标 (u, v) , 反向求解 (X_w, Y_w, Z_w) , 世界坐标就不需要转换为齐次坐标, 上式变为

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{dX} & 0 & u_0 \\ 0 & \frac{f}{dY} & v_0 \\ 0 & 0 & 0 \end{bmatrix} * \left(R * \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T \right) \quad (4)$$

因此, 只需求解等式, 即可得到坐标 $[X_w, Y_w, Z_w]$.

```
[u1,v1]=deal(circle(1,1),circle(1,2)); %赋值
2 [u2,v2]=deal(circle(2,1),circle(2,2));
    pos1=R\((inv(I)*zc1(1)*[u1;v1;1]-T);
4 pos2=R\((inv(I)*zc1(2)*[u2;v2;1]-T);
```

3.1.6 求解距离

利用 `dis=norm(pos1-pos2);` 即可求出两者间的距离

3.2 实验二

3.2.1 任务一

1, 自定义一个仿射矩阵

```
M=[0 1 0; -1 0 0; 0 0 1];    %旋转变换
2  M2=[1 0 0; .5 1 0; 0 0 1]; %剪切变换
```

2, 利用 maketform() 函数将自定义的矩阵转换为仿射变换用的矩阵 T

```
T=maketform('affine', M);
2  tform = affine2d([1 0 0; .5 1 0; 0 0 1]); %剪切变换
```

3, 利用 imwrap() 函数将原图转换为经过矩阵 T 仿射后

```
img_t=imtransform(img,T);
2  J = imwarp(I,tform);
```

3.2.2 任务二

要求利用仿射变换对图像进行矫正观察图片可知，需首先对图片进行向左旋转，再进行整体向右剪切，最后再稍加放大，利用一个仿射矩阵达到以上所说的效果矫正矩阵如下

```
tform3 = affine2d([2 0 0; 4 2 0; 0 0 1]);
2  img_s=imwarp(img,tform3);
```

3.3 实验三

首先读取了两张图像，然后使用 fitgeotrans 函数计算出了一个仿射变换矩阵。接下来，我们使用 imwarp 函数将第一张图像进行仿射变换，使其与第二张图像对齐。最后，我们使用 imfuse 函数将两张

图像进行融合，这里选择了'blend' 模式，即叠加融合。

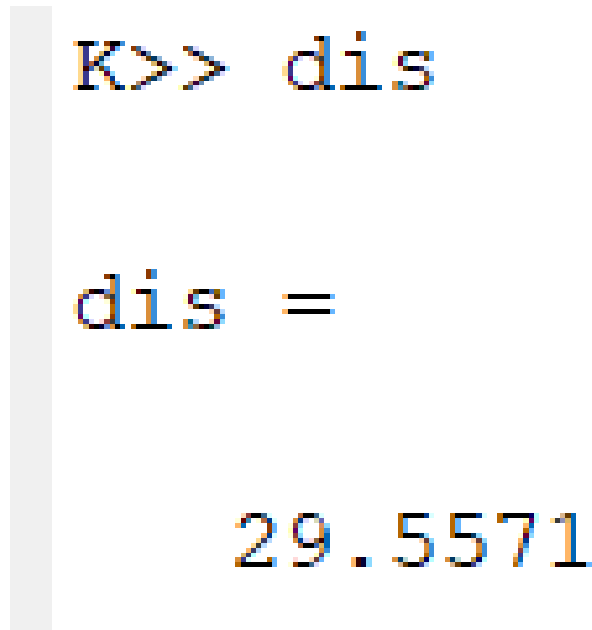
```
% 读取图像
2 image1 = imread('image1.jpg');
  image2 = imread('image2.jpg');
4
% 计算仿射变换矩阵
6 tform = fitgeotrans([1 1; size(image1, 2) 1; 1 size(image1, 1); size(image1, 2) size(image1, 1)], ...
                      [1 1; size(image2, 2) 1; 50 size(image2, 1)-50; size(image2, 2) size(image2, 1)], 'affine');
8
% 对 image1 进行仿射变换
10 image1_transformed = imwarp(image1, tform, 'OutputView', imref2d(size(image2)));

12 % 将两张图像进行融合
    blended_image = imfuse(image1_transformed, image2, 'blend');
14
% 显示结果
16 imshow(blended_image);
```

4 实验结果

4.1 实验一

先对第一张图片处理，得到结果如下



```
K>> dis

dis =

29.5571
```

图 3: 两个橘子的距离

对后续图片处理操作相同，为提高代码简洁性与复用性，将上述操作封装成函数，直接调用即可得到每个圆形物体在世界坐标系中的坐标，再两两计算其之间的距离即可！

```
1 function dis=cal_dis(file_path,zc,range,light,sen)
    img= imread(file_path);
2 img_g=im2gray(img);
    imshow(img_g)
3 I=[650.1821 0 315.8990; 0 650.5969 240.3104; 0 0 1.0000];
    T=[0;0;-2.9];
4 R=[0 0 1; 1 0 0; 0 1 0];
    while 1
5     [circle , radius]=imfindcircles(img_g, range, 'ObjectPolarity',light,'Sensitivity',sen);
        viscircles ( circle , radius, 'Color','b');
6     if size(zc,2)> size(circle,1) %调整敏感度，且要控制在【0.1】间
            sen=sen*1.05;
7     elseif size(zc,2) <size(circle,1)
            sen=sen*0.97;
8     else
            break
9     end
10    end
11 row=size(circle,1);
    dis_up=[];
12 for i=1:row
        [u,v]=deal(circle(i,1), circle(i,2));
```



```
23 pos=R\((inv(I)*zc(i)*[u;v;1]-T);  
    dis_up=[dis_up,pos];  
25 end  
    dis=dis_up;  
27 end
```

```
>> transform  
第1张图从左往右数1和2物体的距离为29.113781cm  
第2张图从左往右数1和2物体的距离为44.689025cm  
第3张图从左往右数1和2物体的距离为25.819651cm  
第4张图从左往右数1和2物体的距离为13.161442cm  
第5张图从左往右数1和2物体的距离为9.042900cm  
第6张图从左往右数1和2物体的距离为14.709895cm  
第7张图从左往右数1和2物体的距离为11.582404cm  
第7张图从左往右数1和3物体的距离为23.635954cm  
第7张图从左往右数2和3物体的距离为13.999921cm  
第8张图从左往右数1和2物体的距离为12.831753cm  
第8张图从左往右数1和3物体的距离为17.055638cm  
第8张图从左往右数2和3物体的距离为16.470653cm  
第9张图从左往右数1和2物体的距离为12.740994cm  
第9张图从左往右数1和3物体的距离为16.197342cm  
第9张图从左往右数1和4物体的距离为26.251942cm  
第9张图从左往右数2和3物体的距离为21.459385cm  
第9张图从左往右数2和4物体的距离为25.213059cm  
第9张图从左往右数3和4物体的距离为13.172016cm  
>> |
```

图 4: 实验结果

补充：由于后续球体数量增多，用`imfindcircles`函数误差已较大，于是采用`ginput`手动进行球心位置标注

4.2 实验二

任务一：熟悉仿射变换



图 5: 仿射变换

任务二：主要对示例图和实验图进行了矫正



图 6: 示例图的仿射变换



图 7: 任务图的仿射变换

4.3 实验三



图 8: 图像融合

可以看出，直接利用仿射变换来实现图像融合效果并不好分析之后原因有如下几点：

- 仿射变换只能处理线性的变换，对于非线性的变换效果不佳。在实际应用中，很多图像的变换是

非线性的，如扭曲、弯曲、拉伸等，这些变换无法用仿射变换来描述。

- 仿射变换不能保证融合后的图像形状的完整性。在进行仿射变换时，图像的形状会发生扭曲或拉伸，这可能导致一些重要的信息丢失或图像的形状发生变化，影响图像的可视化效果
- 在利用仿射变换进行图像融合时，需要手动确定变换参数。这样容易出现误差，导致融合后的图像质量不佳。
- 仿射变换在图像融合中缺乏灵活性。在进行图像融合时，通常需要将多个图像中的不同部分进行融合，而这些部分的形状和位置可能会发生变化。由于仿射变换只能对图像进行固定的线性变换，因此无法灵活地适应这种情况。

5 总结

经过此次实验，我对图像处理技术有着更进一步的了解，对具体如何实现各种图像变换操作技术也更加熟练，同时，除了正常完成此次实验外，经过思考，本人实验也有着自己的创新之处
在实验一中：

- 将图像变换操作封装成一个函数，极大的提高了代码的复用性与便捷性
- 使用了 `ginput` 函数，手动标注小球位置，避免了较大误差的出现
- 创新的设计了敏感度调节结构，减少了调参的次数，提高了代码的效率

在实验二, 三中：

- 使用了 `imtransform`, `imwrap`, `affine2d`, `fitgeotrans` 等多种函数进行图像变换
- 使用 `regionprops` 函数来确定实验三中电视的位置