基于立体视觉的斑马鱼三维轨迹重建

1913416 智能科学与技术 阎丝雨

基于立体视觉基本原理，本程序实现了斑马鱼，三位轨迹重建算法，在精度要求不高的前提下舍弃了对特征点进行跟踪的方法，而是检测出斑马鱼的轮廓，进而得到鱼的重心坐标，进而通过相机的参数计算得到斑马鱼的三维坐标。

程序实现步骤：

一、数据获取

1.1、运动图像

斑马鱼的三位轨迹重建需要基于斑马鱼的实地运动数据图像。在本例的解决过程中，得到两段斑马鱼的运动作为实验数据。对每段斑马鱼的运动数据采取左右两个摄像机成像，组成斑马鱼运动轨迹重建数据集。

1.2、标定图像

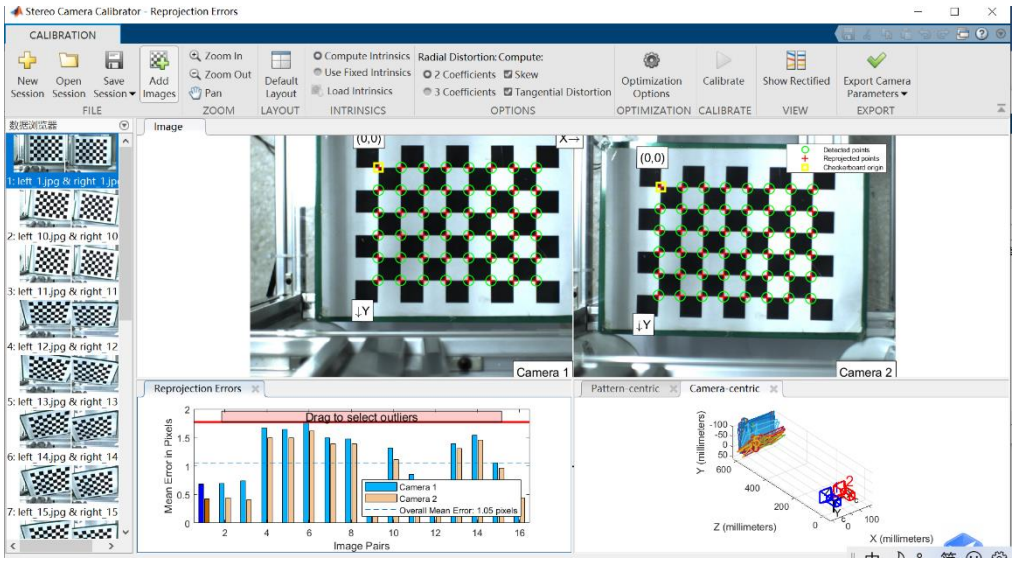
在摄像机的标定作业中，每个摄像机的标定给出十六张角度不同的棋盘格作为标定板，棋盘格的大小参数为格边长 25mm。

二、相机标定

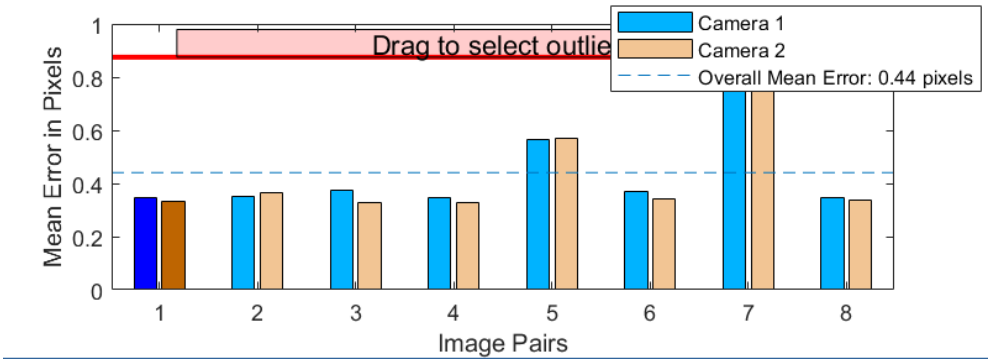
通过摄像机标定来建立有效的成像模型，求解出摄像机的内外参数，这样就可以结合图像的匹配结果得到空间中的三维点坐标，从而达到进行三维重建的目的。

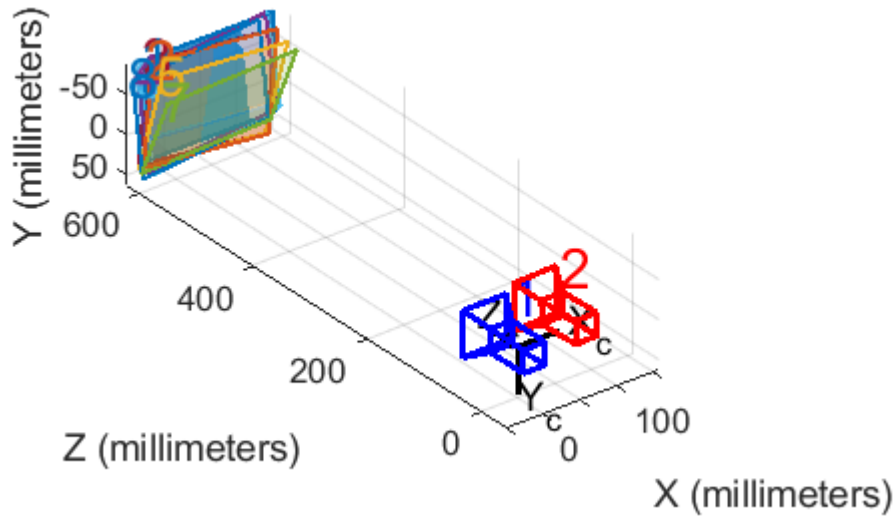
2.1 标定处理

使用 MATLAB 自带 stereoCameraCalibrator 进行相机标定，导入左右标定板数据集



去除误差过大的标定板留下八对标定板图片，得到结果如下，导出 stereoParams 里存储相机参数





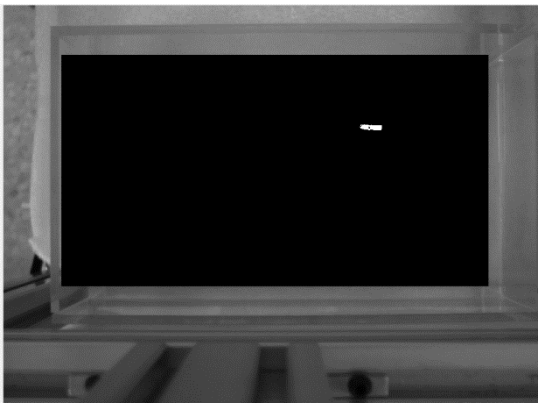
在 stereoParams 中直接获取 camera1 与 camera2 的内参数矩阵，将世界坐标系远在设在 camera1 的光学中心，则 camera2 相对 1 的旋转矩阵与平移矩阵无需处理即组成 camera2 的外参数，由下式可知，外参数矩阵和内参数矩阵共同得到摄像机参数矩阵。

摄像机参数矩阵：

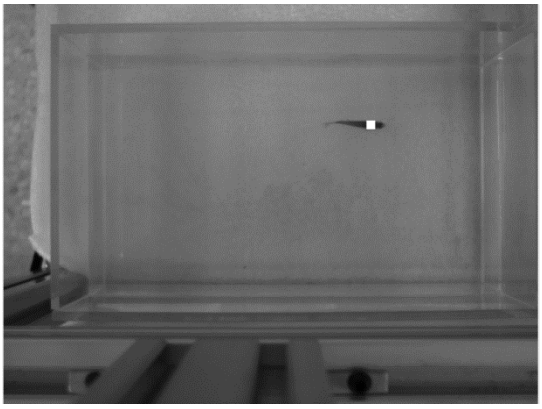
$$M_{(3*4)} = [M_{in} \ \mathbf{0}]\ {}^{c}M_{w}$$

三、定位斑马鱼

观察发现，斑马鱼自身与周围的背景有较大差异，可经过阈值化寻找到鱼的身体，但同时浴缸周围会结果产生较大影响，因此为了消除鱼缸周围物体对阈值化结果的影响，在遍历过程中舍弃了鱼缸周围的部分，只对鱼缸本身进行遍历处理。



图黑色部分为阈值化遍历的部分，白色部分为检测出的鱼的身体的一部分



图中的白色矩形的中心为最后的鱼的坐标，由图可以看出，用阈值化的方法很好地定位到了斑马鱼的位置

```
function [u,v] = FindFish(I)
%定位斑马鱼
I=rgb2gray(I);
[m,n]=size(I);
count=0;
u=0;
v=0;
%为了消除鱼缸周围对阈值化结果的影响
%在遍历中舍弃了鱼纲周围部分，只对鱼缸本身部分进行遍历
for i=120:m-290
    for j=140:n-120
        if I1(i,j)<50
            count=count+1;
            u=u+i;
            v=v+j;
        end
    end
end
u=u/count;
v=v/count;
end
```

四、获取三维坐标

得到了鱼的坐标，结合摄像机标定的内外参数，就可以恢复出三维场景信息。根据 ppt 中的等式，如下图。

- **摄像机成像方程：**

$$z_{ci}\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = M_i \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11}^i & m_{12}^i & m_{13}^i & m_{14}^i \\ m_{21}^i & m_{22}^i & m_{23}^i & m_{23}^i \\ m_{31}^i & m_{32}^i & m_{33}^i & m_{34}^i \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad i = 1,2$$

- **消去 z_{ci}**

$$\begin{cases} (u_i m_{31}^i - m_{11}^i)x + (u_i m_{32}^i - m_{12}^i)y + (u_i m_{33}^i - m_{13}^i)z = m_{14}^i - u_i m_{34}^i \\ (v_i m_{31}^i - m_{21}^i)x + (v_i m_{32}^i - m_{22}^i)y + (v_i m_{33}^i - m_{23}^i)z = m_{24}^i - v_i m_{34}^i \end{cases}$$

- **联立方程求解空间点坐标** $i = 1,2$

图中的矩阵为摄像机的参数矩阵 M，在第二步骤中得到。此方程组由四个式子联立，需要 M，left_fish 的坐标，right_fish 的坐标。

在函数中，A 为等式左面的系数矩阵，B 为等式右边的值，B*（A 的转置），最后得到[x,y,z]，即为鱼的三维坐标。

```
function  [x,y,z] = get_3D(m1,m2,u1,v1,u2)
A = zeros(3, 3);
B = zeros(3, 1);

A(1,1) = u1 * m1(3,1) - m1(1,1);
A(1,2) = u1 * m1(3,2) - m1(1,2);
A(1,3) = u1 * m1(3,3) - m1(1,3);

A(2,1) = v1 * m1(3,1) - m1(2,1);
A(2,2) = v1 * m1(3,2) - m1(2,2);
A(2,3) = v1 * m1(3,3) - m1(2,3);

A(3,1) = u2 * m2(3,1) - m2(1,1);
A(3,2) = u2 * m2(3,2) - m2(1,2);
A(3,3) = u2 * m2(3,3) - m2(1,3);

B(1,1) = m1(1, 4) - u1 * m1(3,4);
B(2,1) = m1(2, 4) - v1 * m1(3,4);
B(3,1) = m2(1, 4) - u2 * m2(3,4);

points3D = A\B;
x=points3D(1);
y=points3D(2);
z=points3D(3);
```

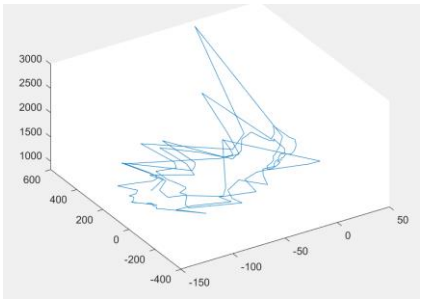
五、三维轨迹重建

得到了每张图片中鱼的三维位置后，接下来要做的就是把坐标显示在三维坐标中并连接以显示鱼的运行轨迹

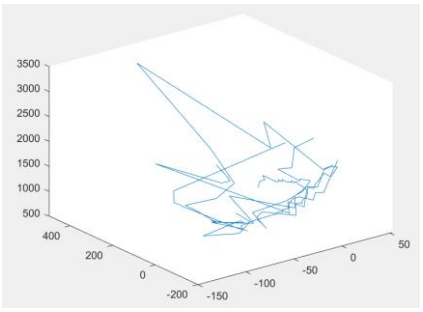
```
file_path1 ='D:\temp\homework_data\fish2\left2\';% 图像文件夹路径
img_path_list1 = dir(strcat(file_path1,'*.jpg'));
file_path2 ='D:\temp\homework_data\fish2\right2\';% 图像文件夹路径
img_path_list2 = dir(strcat(file_path2,'*.jpg'));

img_num = length(img_path_list1);%获取图像总数量
if img_num > 0 %有满足条件的图像
    for j = 1:img_num %逐一读取图像
        image_name1 = img_path_list1(j).name;% 图像名
        image_name2 = img_path_list2(j).name;% 图像名
        image1 =imread(strcat(file_path1,image_name1));
        image2 =imread(strcat(file_path2,image_name2));
        [u1,v1]=FindFish(image1);
        [u2,v2]=FindFish(image2);
        [x,y,z]=get_3D(a,b,u2,v2,u1);
        q(j)=x;
        w(j)=y;
        e(j)=z;
     end
end
plot3(q,w,e);
```

六、结果分析

由结果可知，此算法对两端鱼的运动轨迹都能做到较好的重建。



Fish1 的运动轨迹



Fish2 的运动轨迹

运行效率：

Fish1 运行时间：1.928309s

共处理 194*2 张图片，平均一秒处理 100.06*2 帧

Fish2 运行时间：2.403560s

共处理 187*2 张图片，平均一秒处理 77.80*2 帧