

4.2 标定原理

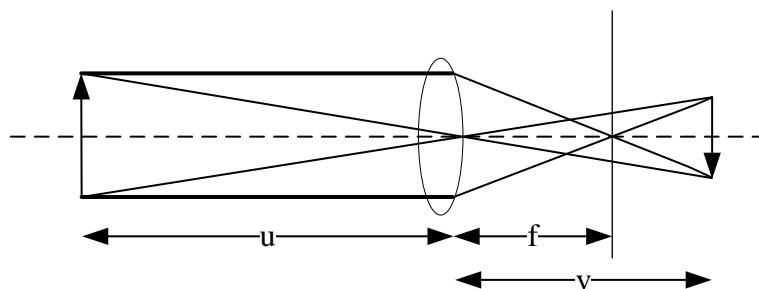
相机标定概念：图像测量过程以及计算机视觉中，为确定空间物体某点的三维几何关系位置与其在图像中对应点之间的相互关系，必须建立相机成像的几何模型，模型的参数就是相机的参数。求解参数的过程称为相机标定。

1) 相机模型：

数码相机图像拍摄的过程实际上是一个光学成像的过程。相机的成像过程涉及到四个坐标系：世界坐标系、相机坐标系、图像坐标系、像素坐标系以及这四个坐标系的转换。

理想透视模型——针孔成像模型

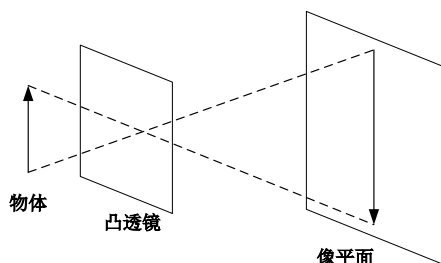
相机模型是光学成像模型的简化，目前有线性模型和非线性模型两种。实际的成像系统是透镜成像的非线性模型。最基本的透镜成像原理如图所示：



其中 u 为物距， f 为焦距， v 为相距。三者满足关系式

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

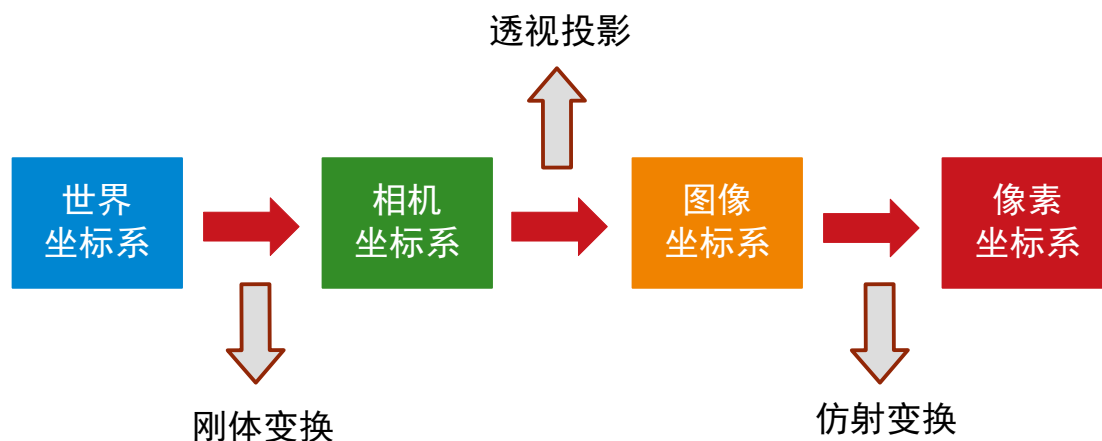
相机的镜头是一组透镜，当平行于主光轴的光线穿过透镜时，会聚到一点上，这个点叫做焦点，焦点到透镜中心的距离叫做焦距 f 。数码相机的镜头相当于一个凸透镜，感光元件就处在这个凸透镜的焦点附近，将焦距近似为凸透镜中心到感光元件的距离时就成为小孔成像模型。小孔成像模型如图所示。



小孔成像模型是相机成像采用最多的模型。在此模型下，物体的空间坐标和图像坐标之间是线性的关系，因而对相机参数的求解就归结到求解线性方程组上。

2) 坐标系定义

相机成像系统中，共包含四个坐标系：世界坐标系、相机坐标系、图像坐标系、像素坐标系。



这四个坐标系之间的转化关系为：

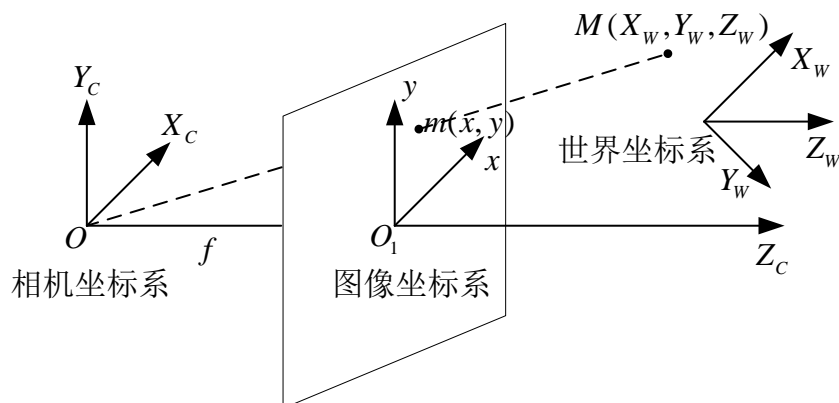
$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{dX} & -\frac{\cot \theta}{dX} & u_0 \\ 0 & \frac{1}{dY \sin \theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

仿射变换
透视投影
刚体变换

内参矩阵
外参矩阵

其中：(U,V,W)为在世界坐标系下一点的物理坐标，(u,v)为该点对应的在像素坐标系下的像素坐标，Z为尺度因子。

四个坐标系的关系图如下图所示，其中M为三维空间点，m为M在图像平面投影成的像点。



①世界坐标系：是客观三维世界的绝对坐标系，也称客观坐标系。因为数码相机安放在三维空间中，我们需要世界坐标系这个基准坐标系来描述数码相机的位置，并且用它来描述安放在此三维环境中的其它任何物体的位置，用 (X_w, Y_w, Z_w) 表示其坐标值。

②相机坐标系（光心坐标系）：以相机的光心为坐标原点， X 轴和 Y 轴分别平行于图像坐标系的 X 轴和 Y 轴，相机的光轴为 Z 轴，用 (X_c, Y_c, Z_c) 表示其坐标值。

③图像坐标系：以 CCD 图像平面的中心为坐标原点， X 轴和 Y 轴分别平行于图像平面的两条垂直边，用 (x, y) 表示其坐标值。图像坐标系是用物理单位（例如毫米）表示像素在图像中的位置。

④像素坐标系：以 CCD 图像平面的左上角顶点为原点， X 轴和 Y 轴分别平行于图像坐标系的 X 轴和 Y 轴，用 (u, v) 表示其坐标值。数码相机采集的图像首先是形成标准电信号的形式，然后再通过模数转换变换为数字图像。每幅图像的存储形式是 $M \times N$ 的数组， M 行 N 列的图像中的每一个元素的数值代表的是图像点的灰度。这样的每个元素叫像素，像素坐标系就是以像素为单位的图像坐标系。

3) 标定原理

为什么要进行相机标定呢？举个例子，当我们拿到一张图片，进行识别之后，得到的两部分之间的距离为 1 像素，但是这 1 像素究竟对应实际世界中的多少米呢？这就需要利用相机标定的结果来将像素坐标转换到物理坐标来计算距离。

我们要想对一个成像系统建模，进而进行相应的计算，所必须的参数就是相机的内参矩

阵：
$$\begin{pmatrix} \frac{f}{dX} & -\frac{f \cot \theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin \theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$
和相机的外参矩阵， $\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}$ ，因此，相机标定的第一个目的就是

获得相机的内参矩阵和外参矩阵。

4) 内参矩阵和外参矩阵

我们将矩阵：

$$\begin{pmatrix} \frac{1}{dX} & -\frac{\cot \theta}{dX} & u_0 \\ 0 & \frac{1}{dY \sin \theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{f}{dX} & -\frac{f \cot \theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin \theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

称为相机的内参矩阵，内参矩阵取决于相机的内部参数。其中， f 为像距， dX, dY 分别表示 X, Y 方向上的一个像素在相机感光板上的物理长度（即一个像素在感光板上是多少毫米）， u_0, v_0 分别表示相机感光板中心在像素坐标系下的坐标。

在相机为理想相机的情况下，焦距、分辨率、视场角关系为：

$$f = \frac{\omega}{2 \tan \frac{\theta}{2}}$$

可推出简化的内参矩阵

$$\begin{pmatrix} f & 0 & u_0 & 0 \\ 0 & f & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

其中 f 为焦距， ω 为分辨率的宽， θ 为视场角。

例如，当视场角为 90° 时，分辨率为 (640×480) 时，可得内参矩阵为：

$$\begin{pmatrix} 320 & 0 & 320 \\ 0 & 320 & 240 \\ 0 & 0 & 1 \end{pmatrix}$$

对于同一个相机，相机的内参矩阵取决于相机的内部参数，无论标定板和相机的位置关系是怎么样的，相机的内参矩阵不变。这也正是在求解内参矩阵中，我们可以利用不同的图片（标定板和相机位置关系不同）获取的矩阵 \mathbf{H} ，共同求解相机内参矩阵 \mathbf{A} 的原因。但是，外参矩阵反映的是标定板和相机的位置关系。对于不同的图片，标定板和相机的位置关系已经改变，此时每一张图片对应的外参矩阵都是不同的。

我们将矩阵： $\begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & \mathbf{1} \end{pmatrix}$ 称为相机的外参矩阵，外参矩阵取决于相机坐标系和世界坐标系的

的相对位置， \mathbf{R} 表示旋转矩阵， \mathbf{T} 表示平移矢量。

即单点无畸变的相机成像模型如下：

$$Z \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{dX} & -\frac{f \cot \theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY \sin \theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & \mathbf{1} \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

\mathbf{R} 的每一个列向量表示世界坐标系的每一个坐标轴在相机坐标系下的指向；而 \mathbf{t} 是世界坐标系原点在相机坐标系下的表示。所以我们需要进行坐标转换，将相机内部坐标转换为世界坐标系。

此模型的旋转矩阵如下：

$$\mathbf{R}_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & \sin \gamma \\ 0 & -\sin \gamma & \cos \gamma \end{bmatrix} \mathbf{R}_y(\beta) = \begin{bmatrix} \cos \beta & 0 & -\sin \beta \\ 0 & 1 & 0 \\ \sin \beta & 0 & \cos \beta \end{bmatrix} \mathbf{R}_z(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

由于我们已知的条件是相机的欧拉角，因此用相机坐标系推出世界坐标系的做法更为简便。因此我们使用另外一个模型：

假设点 P 是一个三维空间中的点，其在相机坐标系下的位置为 P_c ，在世界坐标系下的位置为 P_w 。 P_w 和 P_c 可以通过一个变换矩阵相互转换，该变换矩阵可以细分为旋转矩阵 \mathbf{R}' 和平移矩阵 \mathbf{t} 。其数学表达形式为：

$$P_w = \mathbf{R}' P_c + \mathbf{t}$$

旋转方向与对应矩阵变换的关系：

左乘右乘

左乘——相对于固定坐标系进行变换。（比如世界坐标系，例如绕世界坐标系旋转，先Z后Y再X，就可以得到旋转矩阵 $\mathbf{R} = \mathbf{R}_x \mathbf{R}_y \mathbf{R}_z$ 按照顺序往左边乘。）

右乘——相对于自身坐标系进行变换，每变一次下一次需要以新坐标系为标准进行变换。比如第一次变换后，原x轴的位置变为y轴，那么下一次绕y轴的变换，就会绕之前的x轴变换。比如按照机体坐标系进行旋转，例如绕机体坐标系先Z后Y再X，就有 $\mathbf{R} = \mathbf{R}_z \mathbf{R}_y \mathbf{R}_x$ ，按照顺序往右乘

按机体坐标轴旋转的旋转矩阵表示：

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$$

$$\mathbf{R}_y(\gamma) = \begin{bmatrix} \cos \gamma & 0 & \sin \gamma \\ 0 & 1 & 0 \\ -\sin \gamma & 0 & \cos \gamma \end{bmatrix}$$

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}' \text{——} 3 \times 3 \text{ 的旋转矩阵} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \gamma & 0 & \sin \gamma \\ 0 & 1 & 0 \\ -\sin \gamma & 0 & \cos \gamma \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}, \text{ 其中}$$

(θ, γ, α) 为标定板围绕相机坐标系3个轴的转角，称为旋转向量；

\mathbf{t} —— (t_x, t_y, t_z) 为平移向量。 \mathbf{R}' 与 \mathbf{t} 合并称为相机外参。完整的外参矩阵为 $\begin{pmatrix} \mathbf{R}' & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix}$ 。

5) Matlab 标定板标定

一、单目相机图像存储：

将 OneCameraCal.bat 中的路径改为使用电脑中的路径，双击运行。

OneCameraCal.bat 内容如下（标黄部分与路径相关）：

```
@ECHO OFF
```

```
REM Set the path of the RflySim tools
```

```
SET PSP_PATH=C:\PX4PSP
```

```
C:
```

```
REM kill all applications when press a key
```

```
tasklist|find /i "CopterSim.exe" && taskkill /im "CopterSim.exe"
```

```
tasklist|find /i "QGroundControl.exe" && taskkill /f /im "QGroundControl.exe"
```

```
tasklist|find /i "RflySim3D.exe" && taskkill /f /im "RflySim3D.exe"
```

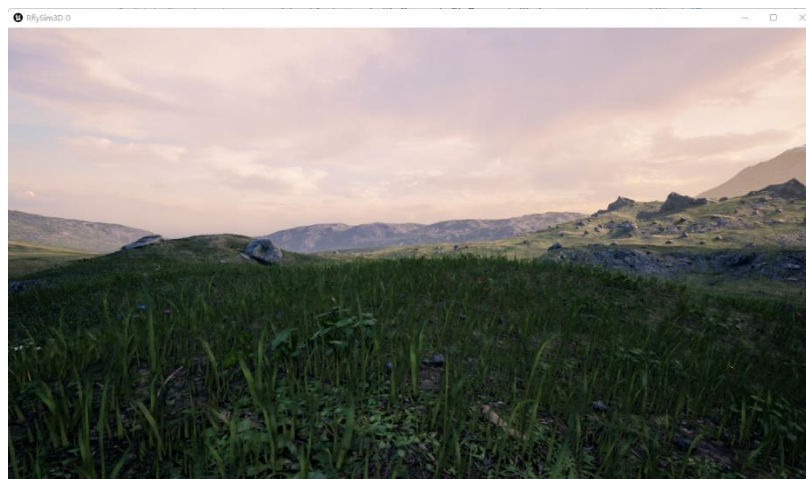
```
REM UE4Path
```

```
start %PSP_PATH%\RflySim3D\RflySim3D.exe
```

```
pause
```

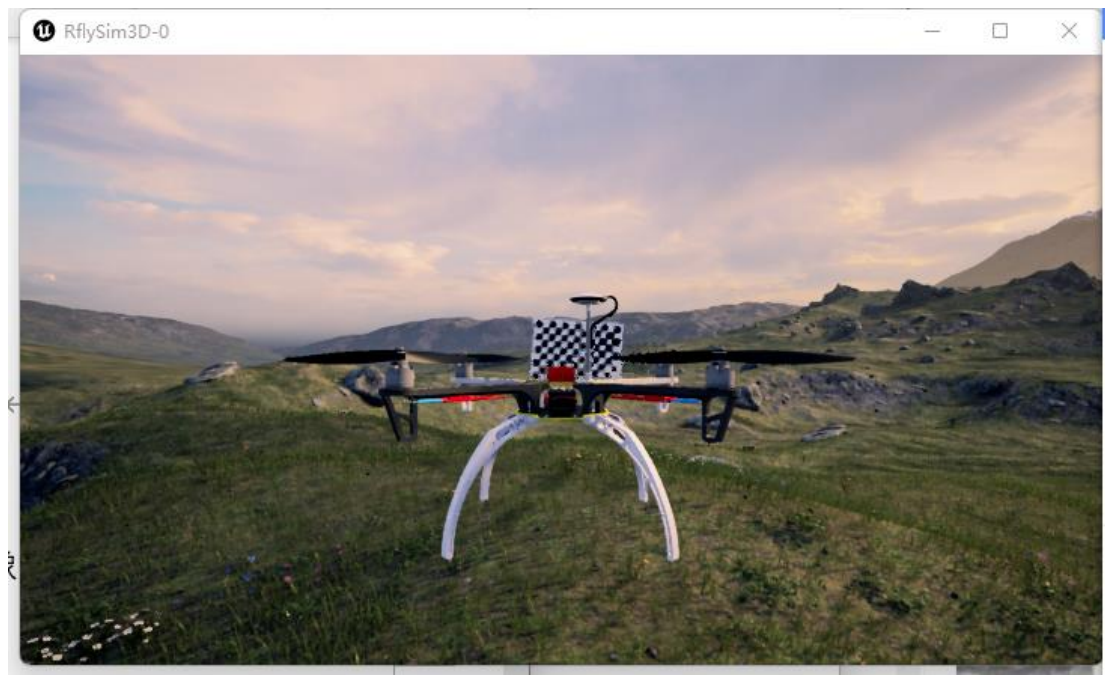
```
tasklist|find /i "RflySim3D.exe" && taskkill /f /im "RflySim3D.exe"
```

会得到如下图像：

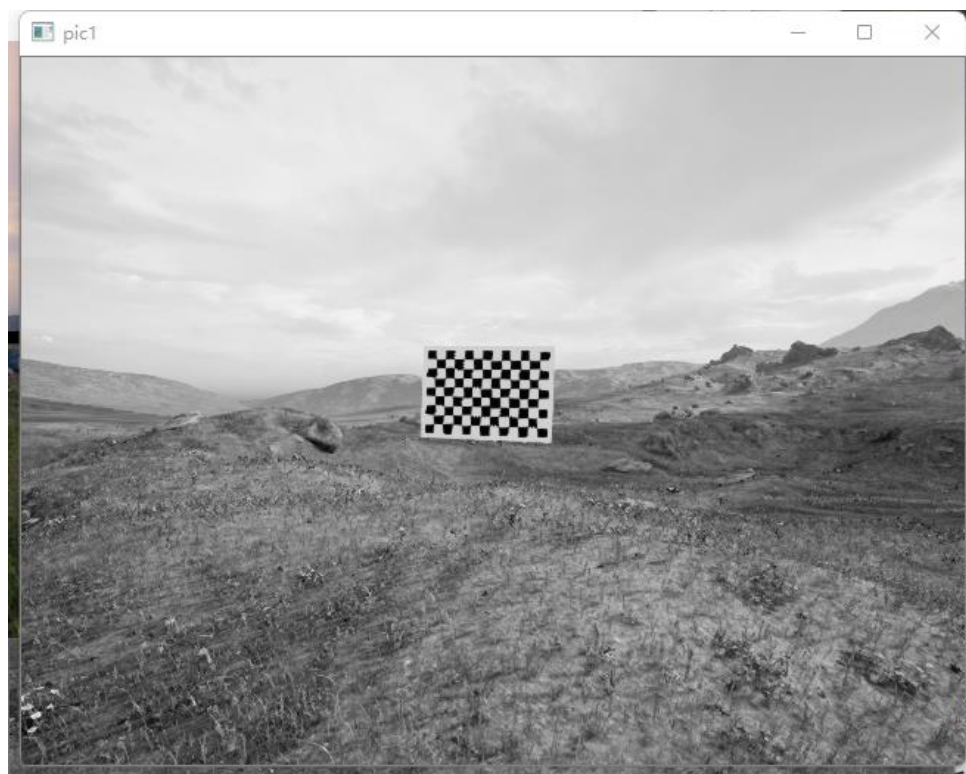


此时 Rflysim3D 已启动

在 VS Code 中运行 OneCameraCal.py，界面中会出现标定板，图像如下：




和相机拍摄的标定板图像:



捕获的图片保存在工作路径下的一个文件夹，例如

C:\PX4PSP\RflySimAPIs\PythonVisionAPI\2-CameraCalcDemo\20220331_172852

如图:

 20220331_172852

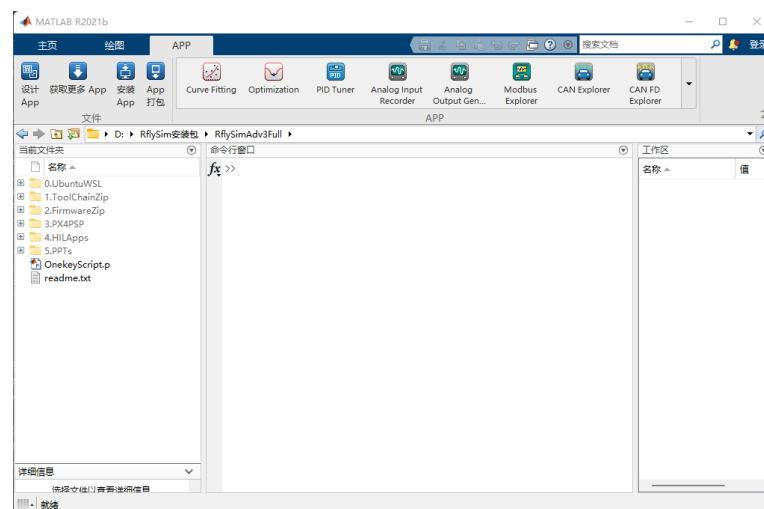
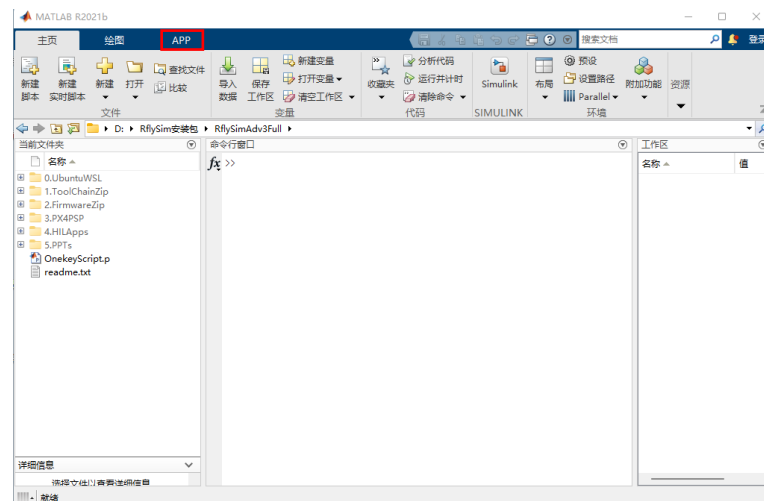
若标定板与相机的距离太远或太近，会影响检测结果，可以修改 `InitTargePos` 中的第一个数值来改变距离。

获得三十张图片左右后就可将程序停止。

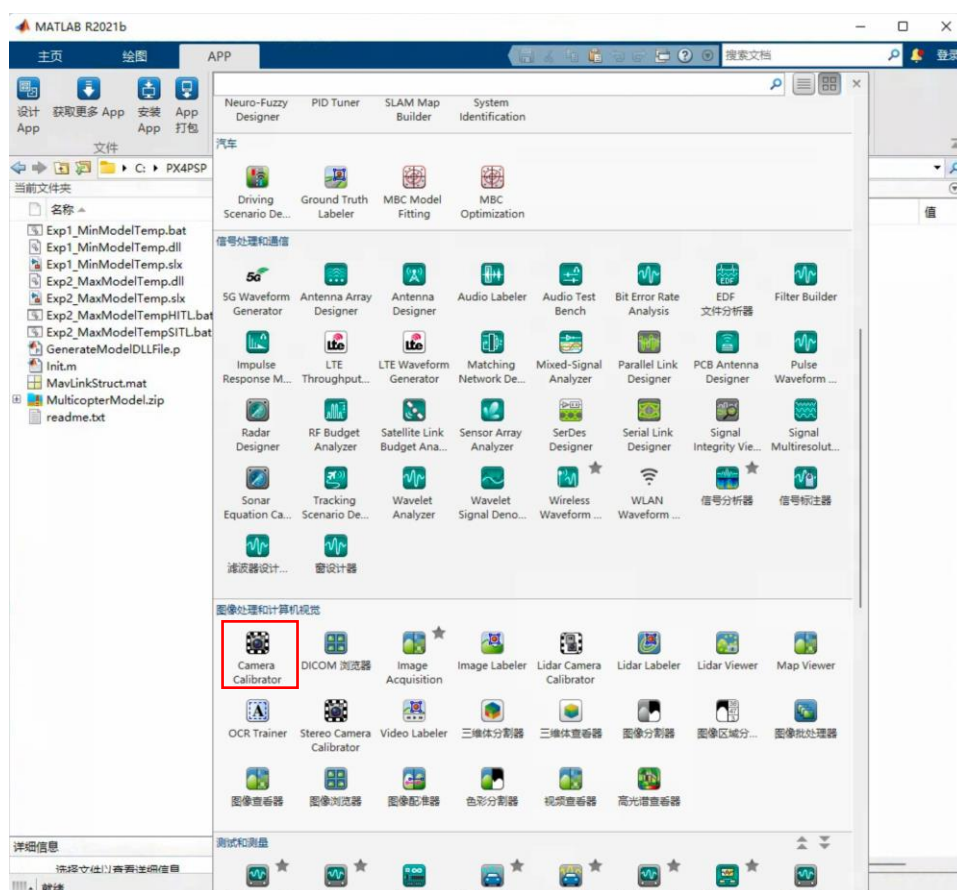
6) Matlab 标定

要求：将存储下来的单目相机图像，在 MATLAB 中校准，得到相机内参。

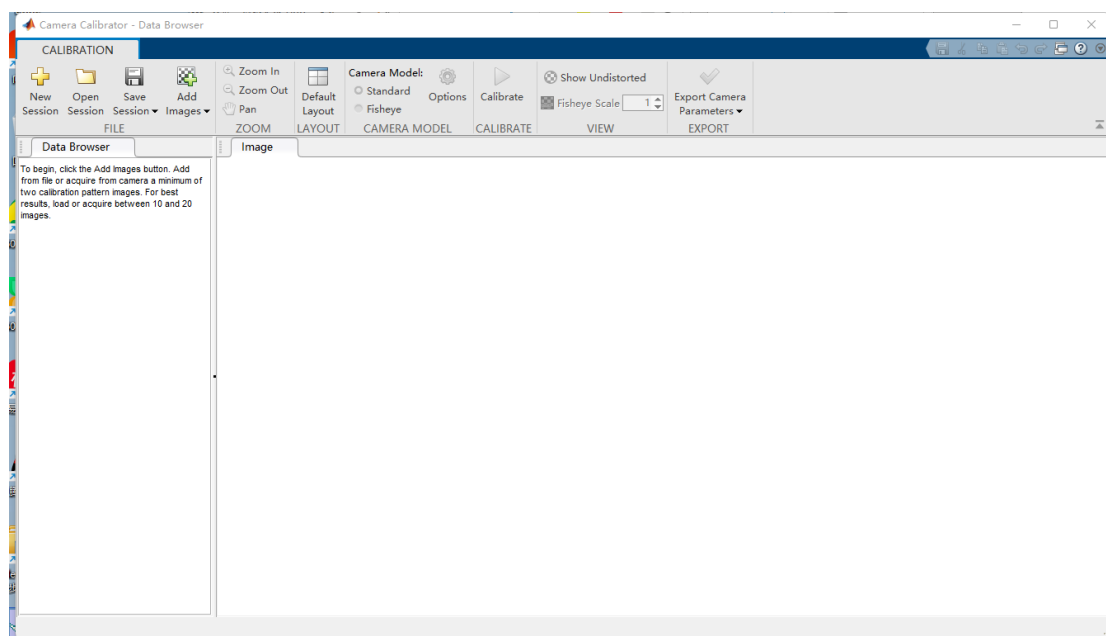
1.打开 matlab，点击上方的 APP 栏



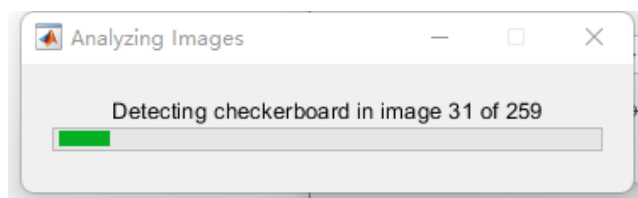
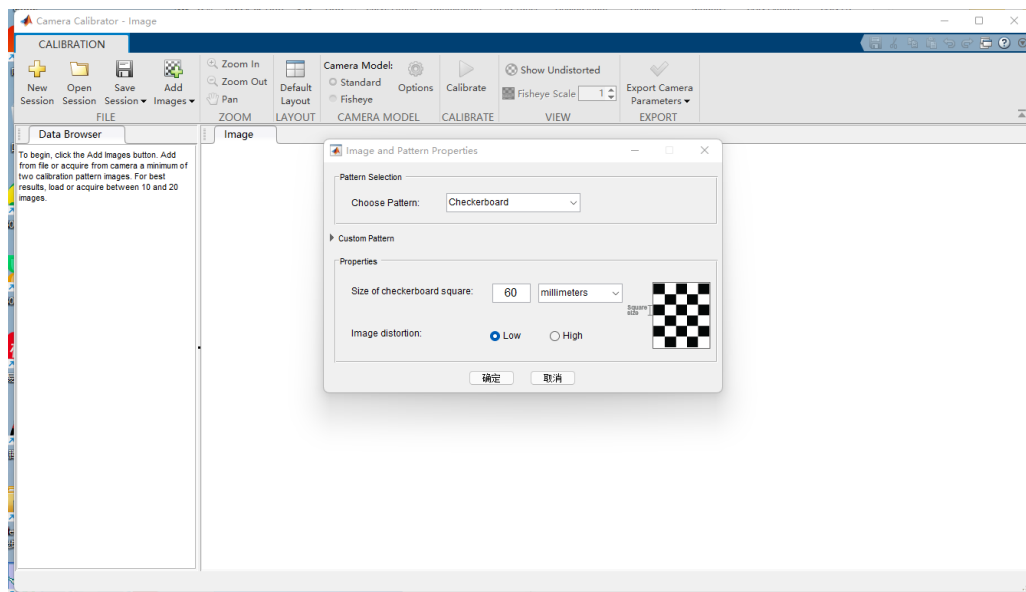
2.下拉工具栏，选择 Camera Calibrator 工具箱



3. 打开工具箱后，点击 **Add Images**，打开到我们抓取图片的位置，选取图片。选取完成后点击打开按钮。



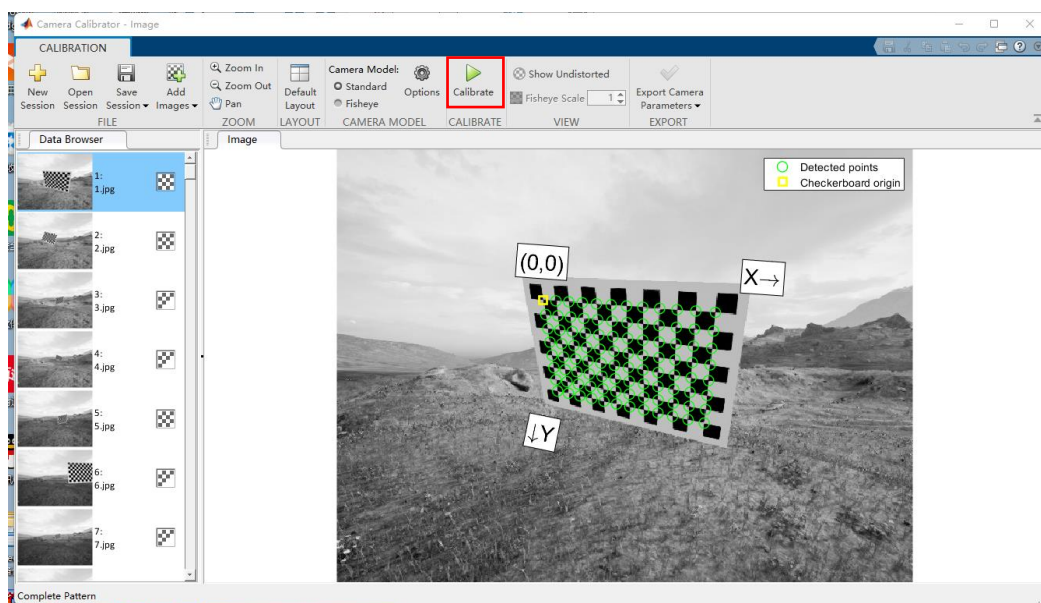
4. 弹出下面窗口，输入棋盘格的大小。场景中为大家提供的是 60mm，点击确定。



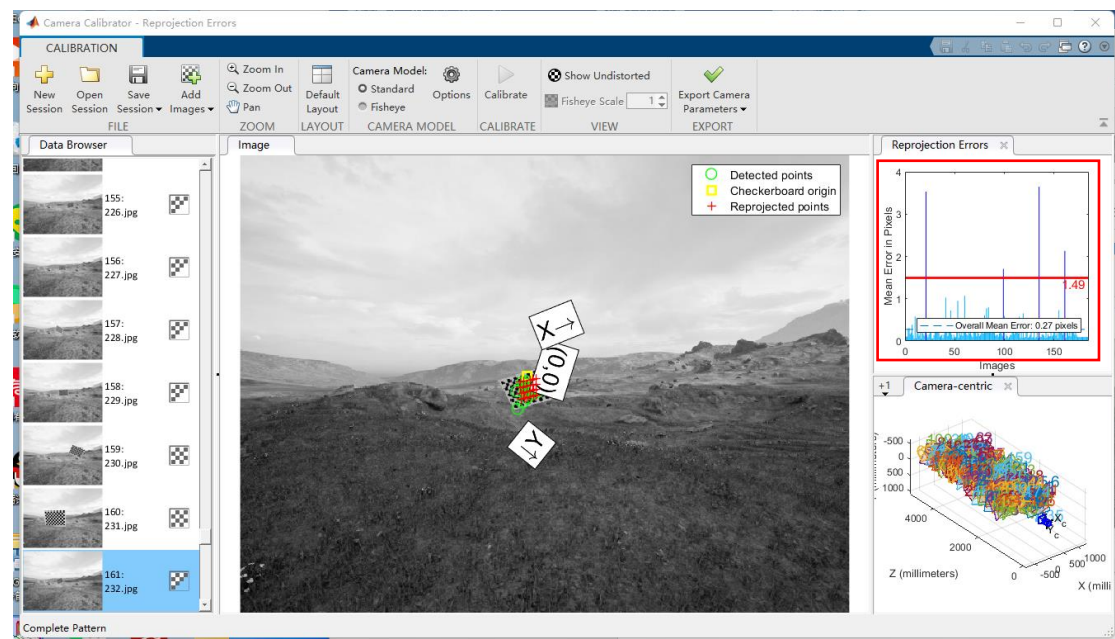
5.几秒钟后，工具箱会告诉你接受的图片，直接点确定。



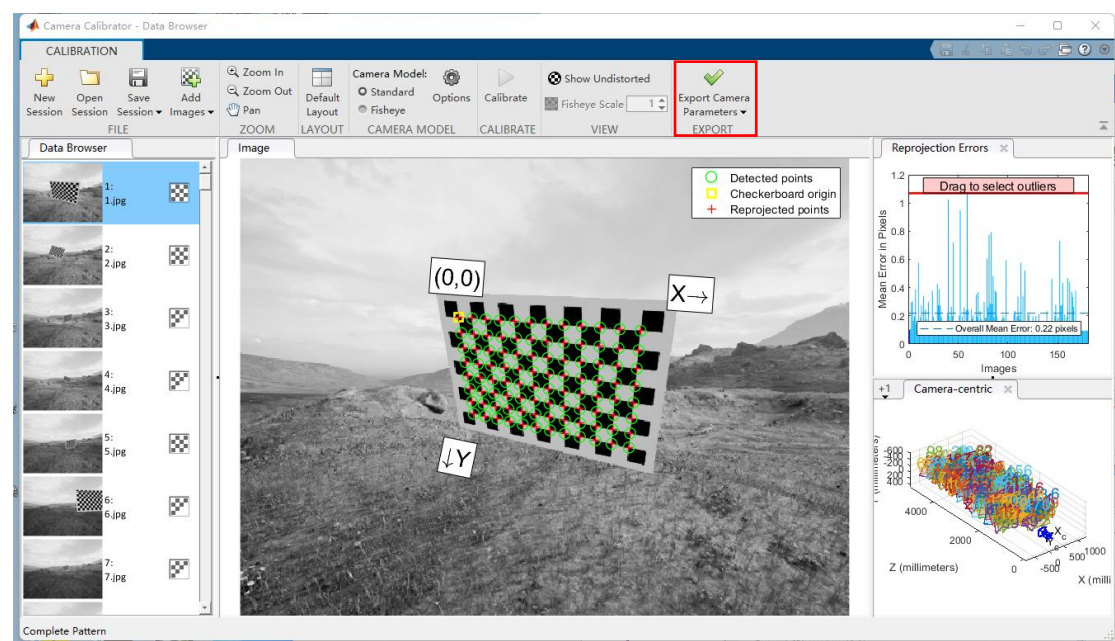
6.可以大概浏览一下棋盘格提取质量，应该是都在图像上。点击 Calibrate 按钮开始标定。



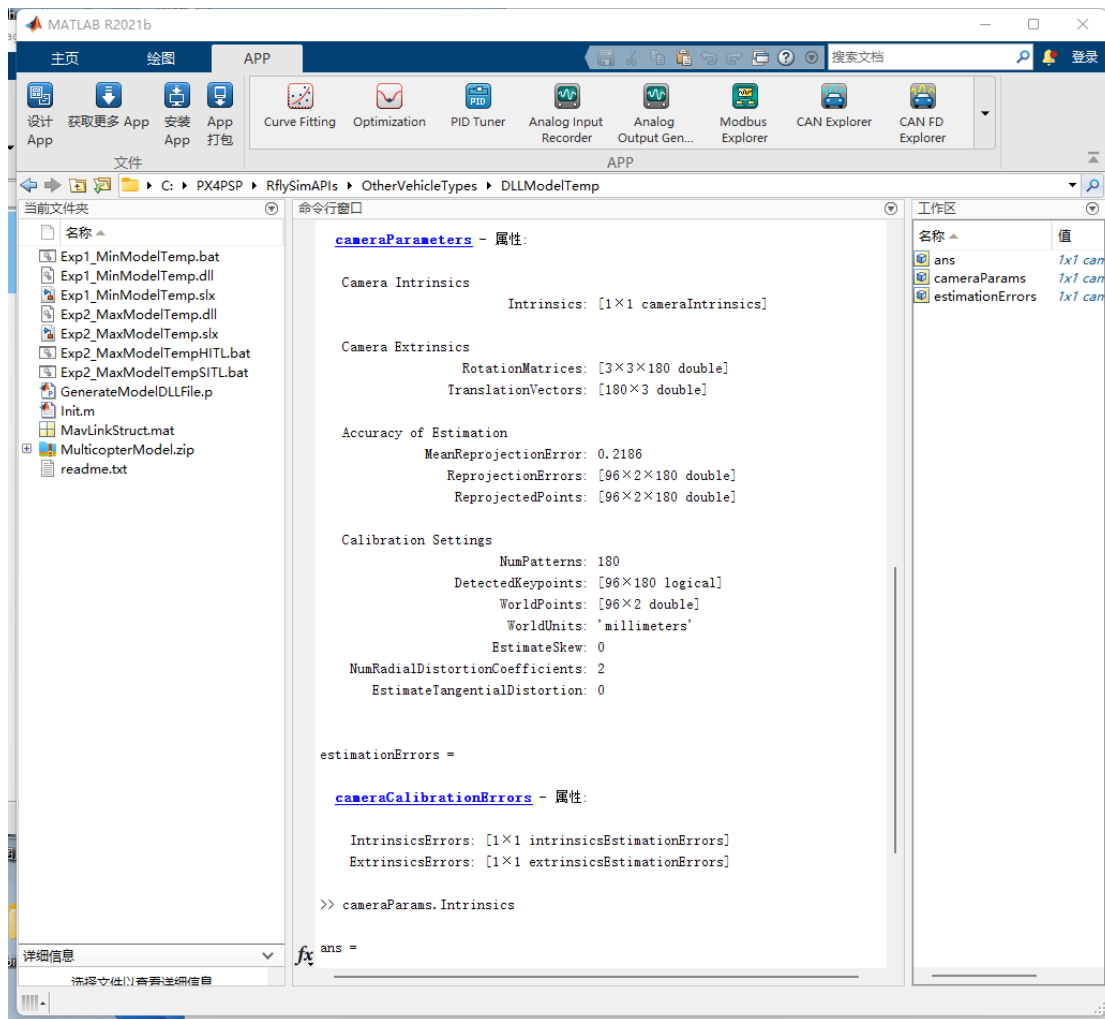
7.可以拖动红线选择反投影误差较大的图片，按 Delete 删除这些图片并重新标定。



8.结果满意之后点击 Export 导出标定的结果。



9.回到 Matlab 工作区，会看到一个名为 cameraParams 的结构体，双击可以看到各成员变量。也可以在下方输入 cameraParams.Intrinsics 查看相机信息，输入 cameraParams.IntrinsicMatrix 查看内参矩阵。



cameraIntrinsics - 属性:

```
FocalLength: [324.5972 321.5801]
PrincipalPoint: [319.6610 239.6478]
ImageSize: [480 640]
RadialDistortion: [-0.0012 -6.9343e-04]
TangentialDistortion: [0 0]
Skew: 0
IntrinsicMatrix: [3×3 double]
```

7) Python 标定

一、单目相机图像存储:

将 OneCameraCal.bat 中的路径改为使用电脑中的路径，双击运行。

OneCameraCal.bat 内容如下（标黄部分与路径相关）：

@ECHO OFF

REM Set the path of the RflySim tools

SET PSP_PATH=C:\PX4PSP

C:

REM kill all applications when press a key

tasklist|find /i "CopterSim.exe" && taskkill /im "CopterSim.exe"

tasklist|find /i "QGroundControl.exe" && taskkill /f /im "QGroundControl.exe"

tasklist|find /i "RflySim3D.exe" && taskkill /f /im "RflySim3D.exe"

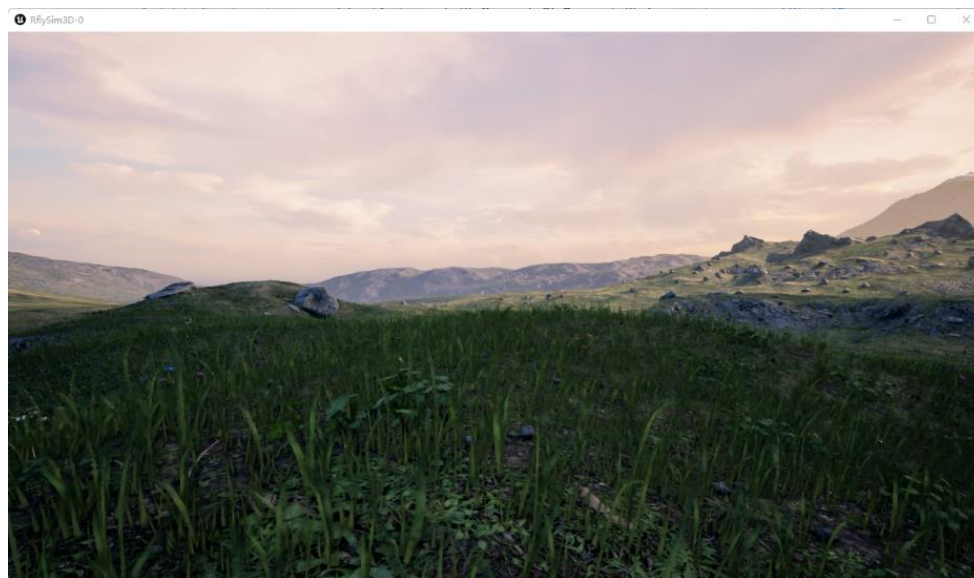
REM UE4Path

start %PSP_PATH%\RflySim3D\RflySim3D.exe

pause

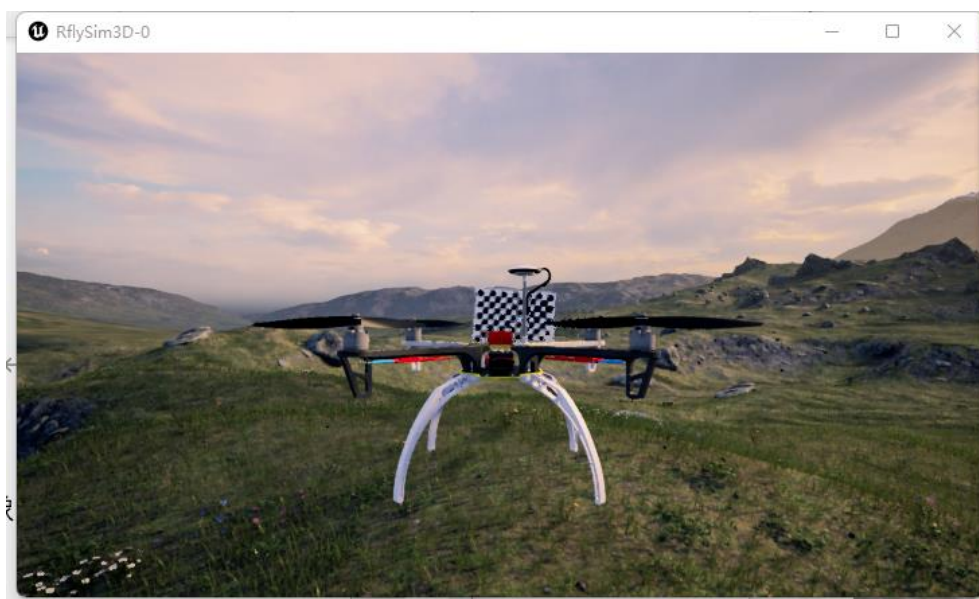
tasklist|find /i "RflySim3D.exe" && taskkill /f /im "RflySim3D.exe"

会得到如下图像：

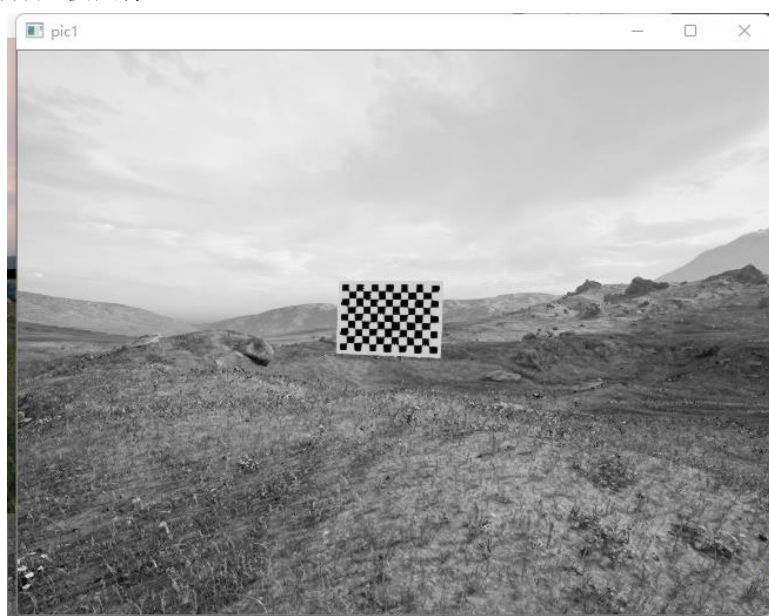


此时 Rflysim3D 已启动

在 VS Code 中运行 OneCameraCal.py，界面中会出现标定板，图像如下：




和相机拍摄的标定板图像：



捕获的图片保存在工作路径下的一个文件夹，例如

C:\PX4PSP\RflySimAPIs\PythonVisionAPI\2-CameraCalcDemo\20220331_172852

如图：

 20220331_172852

若标定板与相机的距离太远或太近，会影响检测结果，可以修改 `InitTargePos` 中的第一个数值来改变距离。

获得三十张图片左右后就可将程序停止。

二、Python 标定：

在 `bord-Calibration.py` 文件中将第 19 行的路径改为要标定的照片路径并运行。

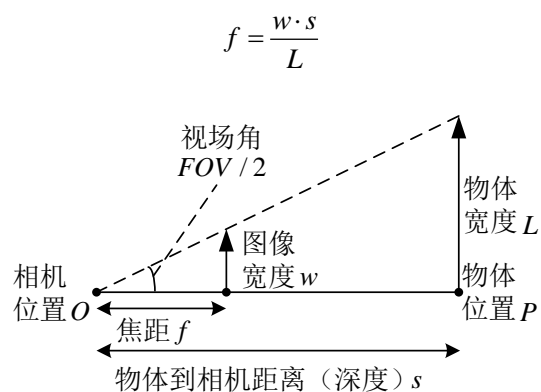
```
images = glob.glob("..\20220331_172852\img1\*.jpg")
```

得到的内参矩阵如下

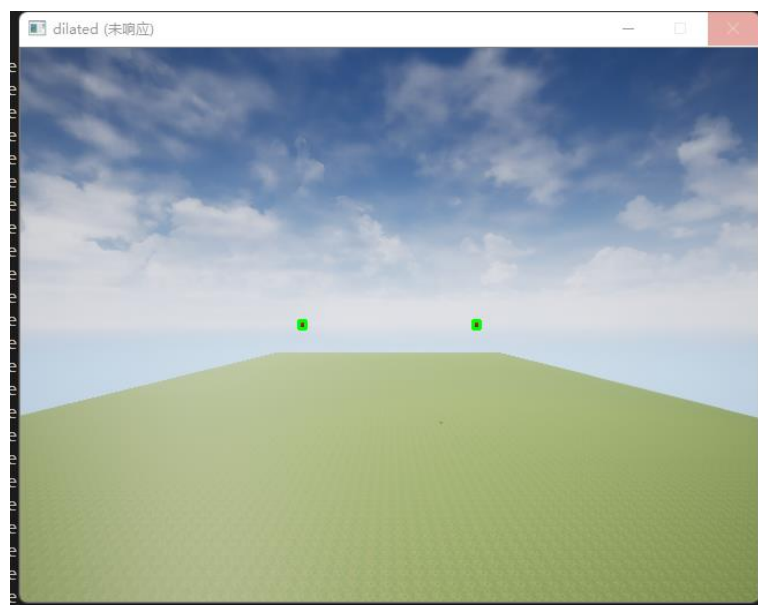
```
[[322.36795044    0.        318.71025401]
 [    0.        319.37265015  235.49389293]
 [    0.         0.         1.         ]]
```

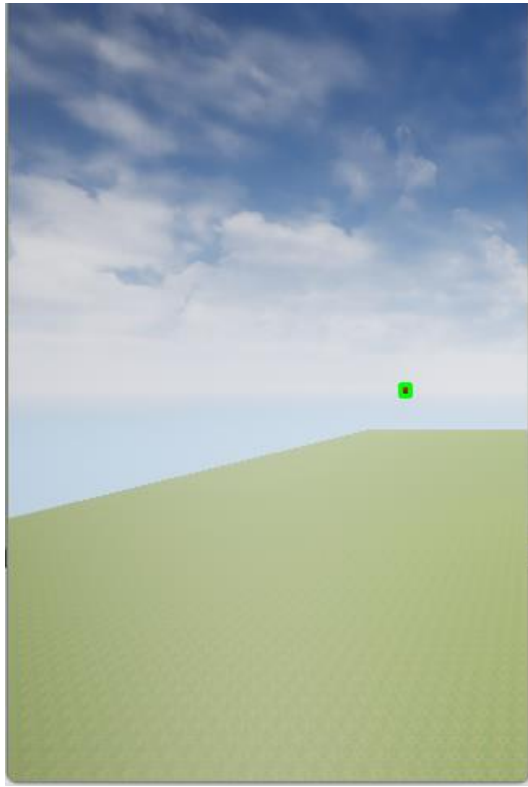
8) Python 双小球的标定流程

标定原理：生成两个小球，其距离固定为世界坐标下一米。根据两个小球在图像上的间距，来判断世界坐标系和图像坐标系的比例关系。再通过小球连线中点到相机的距离来量度坐标原点的平移。由两小球距离 L 和小球在图像上的宽度 w 和小球连线中点到相机的距离 s 可得出焦距 f 。



在 VS Code 中运行 `ball2-Calibration.py`,可以得到结果如下图：





```
ball2-Calibration.py X
C: > Users > RFLYSIM > Desktop > biaoding > OneCameraCal-gather > ball2-Calibration.py > {} random
1 # import required libraries
2 import time
3 import sys
4 import cv2 as cv
5 import random
6 import numpy as np
7
```

问题 输出 调试控制台 终端

```
The current focallength_x is 325.0000 ; The distance is 3.5500
The current focallength_x is 322.8750 ; The distance is 1.8750
The current focallength_x is 320.4500 ; The distance is 1.7500
The current focallength_x is 324.0750 ; The distance is 4.0250
The current focallength_x is 322.5000 ; The distance is 1.8000
The current focallength_x is 323.8500 ; The distance is 3.4750
The current focallength_x is 321.6000 ; The distance is 2.7000
The current focallength_x is 324.3000 ; The distance is 3.8250
The current focallength_x is 322.4000 ; The distance is 5.5000
The current focallength_x is 321.9500 ; The distance is 3.7250
The current focallength_x is 322.5000 ; The distance is 3.5250
The current focallength_x is 324.9000 ; The distance is 4.5750
The current focallength_x is 320.8500 ; The distance is 2.6250
The current focallength_x is 324.9000 ; The distance is 3.1500
The current focallength_x is 326.0250 ; The distance is 5.0250
The current focallength_x is 320.9500 ; The distance is 3.5750
The current focallength_x is 324.9000 ; The distance is 3.1500
The current focallength_x is 322.0000 ; The distance is 4.3250
The current focallength_x is 322.0500 ; The distance is 3.1250
The current focallength_x is 326.4000 ; The distance is 5.4000
The current focallength_x is 322.5000 ; The distance is 3.5250
The current focallength_x is 320.8500 ; The distance is 2.6250
The current focallength_x is 325.6000 ; The distance is 4.7000
The current focallength_x is 322.5000 ; The distance is 1.8000
The current focallength_x is 326.2500 ; The distance is 3.9250
The current focallength_x is 324.0000 ; The distance is 3.9000
The current focallength_x is 322.5250 ; The distance is 2.7250
The current focallength_x is 324.7000 ; The distance is 5.0750
The current focallength_x is 321.3000 ; The distance is 1.6500
The current focallength_x is 323.7000 ; The distance is 4.4500
The current focallength_x is 325.0500 ; The distance is 5.2250
The current focallength_x is 326.0250 ; The distance is 5.4750
The current focallength_x is 323.0000 ; The distance is 2.4250
The mean focal length of the camera is: 323.3808 100
```

powershell
Python
Python
Python

0 6.43 KiB 行 5, 列 14 空格: 4 UTF-8 CRLF Python