

实验三：EP 视觉伺服矿石抓取 manual

Author:  曹展翔

Email: caozx1110@163.com

一、实验目的

1. 熟悉使用 ROS 框架
2. 熟悉调用 RoboMaster EP SDK 接口
3. 学习使用相机标定工具
4. 将之前所编写工具包进行组织并完成矿石抓取的任务

二、实验要求

1. 对相机进行参数标定
2. 部署 ROS 工程
3. 调整视觉伺服参数
4. 有能力的同学可自行编写策略

三、实验环境

1. 克隆仓库

在任一文件夹下：

```
1 git clone https://github.com/caozx1110/course_experiment2.git
```

2. 安装依赖库

```
1 cd ./course_experiment2
2 pip install -r ./requirements.txt
```

3. 编译工作空间 & source

```
1 catkin_make
2 echo "source $(pwd)/devel/setup.bash" >> ~/.bashrc
3 source ~/.bashrc
```

四、实验内容

(1) EP 数据传输节点

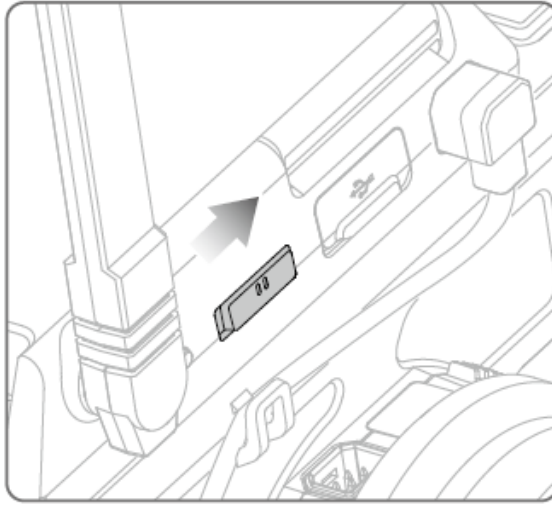
1. 更改 config 配置文件

打开 `src/config/config.yaml` 文件，更改其中的 WIFI 路由器名称和密码：

```
1 wifi:
2   ssid: '路由器名称'
3   password: '路由器密码'
```

2. EP 准备

- 开机
- 切换智能中控连接方式为组网连接



3. 运行 EP 数据传输节点

- 启动 launch

```
1 roslaunch ep_bringup bringup.launch
```

- 如是第一次连接，将弹出二维码，将 EP 摄像头对准二维码，点按智能中控上的圆形按钮，如下图：



- 根据语音提示进行连接，输出提示信息及 SN 码表示连接成功！

```
/home/czx/ws/repo/course_experiment2/src/ep_bringup/launch/bringup.launch http... - □ ×
/home/czx/ws/repo/cours... × rviz × rostopic echo /ep/servo/a... ×
/home/czx/ws/repo/course_experiment2/src/ep_bringup/launch/bringup.launch http://localhost:11311 80...
=====
PARAMETERS
* /rostdistro: noetic
* /rosversion: 1.15.14

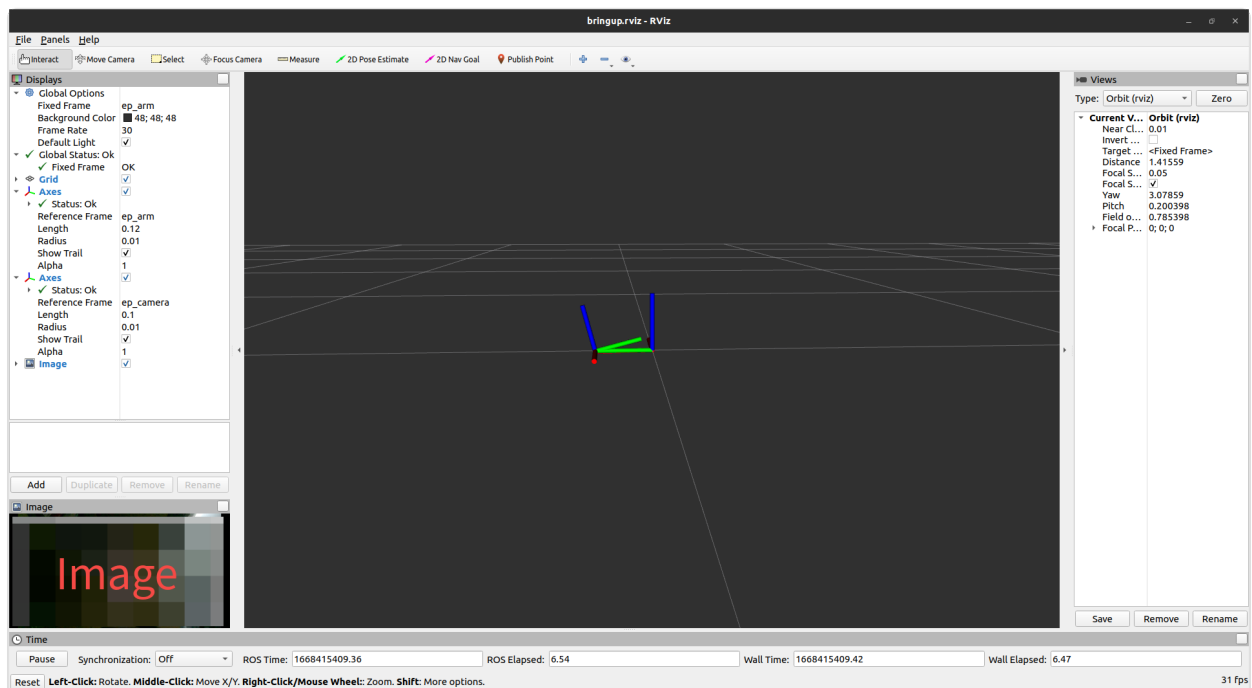
NODES
/
  ep_bringup_node (ep_bringup/ep_bringup.py)

auto-starting new master
process[master]: started with pid [25160]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 9e04dbcc-63ee-11ed-9fb4-7baa4dc7f70d
process[rosout-1]: started with pid [25190]
started core service [/rosout]
process[ep_bringup_node-2]: started with pid [25193]
[INFO] [1668411203.609264]: EP Robot connected!
[INFO] [1668411203.649639]: Robot SN: 3JKDH3B0016P59
```

4. 检查节点是否正常输出

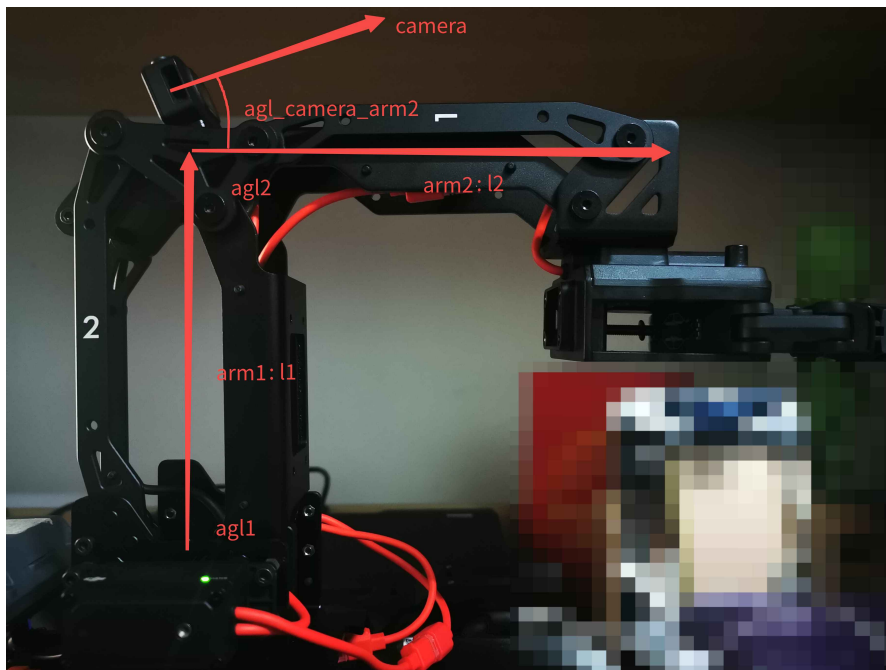
```
1 rviz -d `rospack find ep_bringup`/rviz/bringup.rviz
```



(2) 机械臂标定（粗略）

1. 基准值标定

- 将机械臂移动至下图所示姿态（臂横平竖直，称为 **base_pose**）：



- 查看当前舵机角度值

```
1 rostopic echo /ep/servo/angles
```

```
rostopic echo /ep/servo/angles
/home/czx/ws/re... x rviz-d`rospack fi... x rostopic echo /ep... x rostopic echo tf x
rostopic echo /ep/servo/angles 80x21
layout:
  dim: []
  data_offset: 0
data: [2.2235493659973145, 0.0, 3.3772120475769043]
---
layout:
  dim: []
  data_offset: 0
data: [2.2235493659973145, 0.0, 3.3807027339935303]
---
layout:
  dim: []
  data_offset: 0
data: [2.2235493659973145, 0.0, 3.3772120475769043]
---
layout:
  dim: []
  data_offset: 0
data: [2.2235493659973145, 0.0, 3.3772120475769043]
---
```

- 确定角度对应关系

通过旋转两个角度，确定 agl1、agl2 分别对应的 data 中的下标，例如：只旋转 agl1，发现只有 data[2] 变化，则 agl1 对应的 index 则为 2。对应修改 config.yaml 配置文件（以下简称配置）中的参数值：

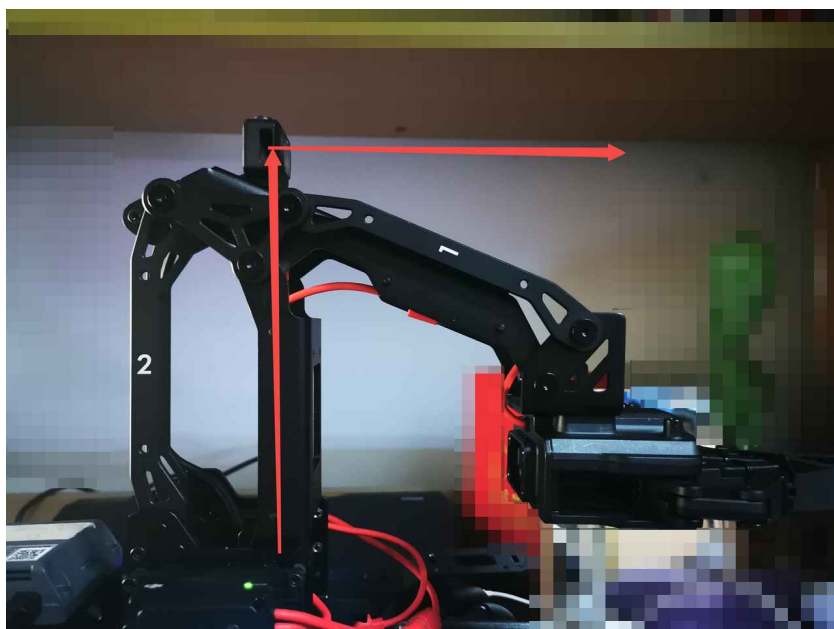
```
1 # the agl1 & agl2 in the servo.angle_topic_name msg's index
2 agl1_idx: 2
3 agl2_idx: 0
```

- 记录 base_pose 下的 agl1、agl2 数值，写入配置：

```
1  # in this pose, the base agl1, agl2 in rad
2  agl1_base: 3.377
3  agl2_base: 2.224
```

2. 相机偏角标定

- 旋转 agl2，使得相机保持垂直，相机面朝正前方，如下图所示：



- 当前 agl2 数值，减去 base_pose 下的 agl2 数值（即 agl2_base），结果记录为 agl_camera_arm2，写入配置：

```
rostopic echo /ep/servo/angles
/home/czx/ws/re... x rviz-d`rospack fi... x rostopic echo /ep... x rostopic echo tf x
rostopic echo /ep/servo/angles 80x21
layout:
  dim: []
  data_offset: 0
data: [2.616248607635498, 0.0, 3.3772120475769043]
---
layout:
  dim: []
  data_offset: 0
data: [2.616248607635498, 0.0, 3.3772120475769043]
---
layout:
  dim: []
  data_offset: 0
data: [2.616248607635498, 0.0, 3.3807027339935303]
---
layout:
  dim: []
  data_offset: 0
data: [2.616248607635498, 0.0, 3.3807027339935303]
---
```

```
1 # the angle between the front view of the camera and the arm2
2 agl_camera_arm2: 0.399
```

3. Rviz 可视化

```
1 rviz -d `rospack find ep_bringup`/rviz/bringup.rviz
```

移动机械臂，可看到图中 frame 也随之变化。

(3) 相机标定

参考链接：

[camera_calibration/Tutorials/MonocularCalibration - ROS Wiki](#)

[摄像头标定](#)

[解决ROS系统 rosdep update 超时问题的新方法](#)

1. 安装工具包

```
1 sudo apt-get install ros-noetic-camera-calibration
```

或者：

```
1 rosdep install camera_calibration
```

Note: 如果出现 rosdep 安装错误，可以参考 [解决ROS系统 rosdep update 超时问题的新方法](#) 解决源问题

2. 启动 EP 数据传输节点

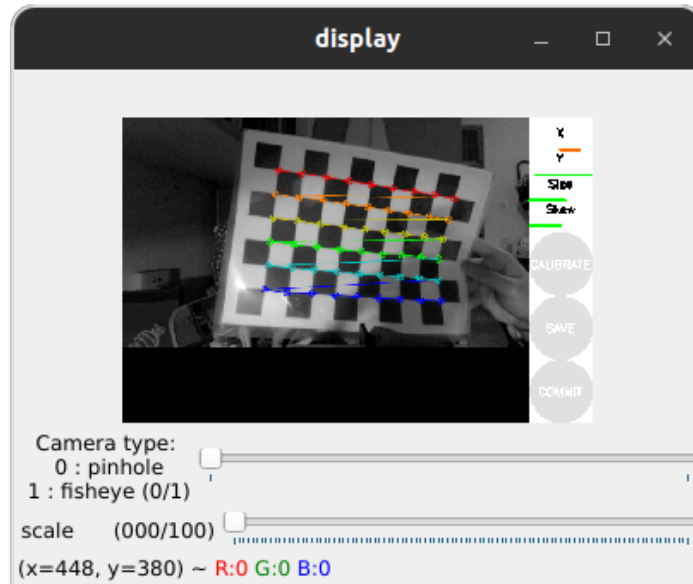
```
1 roslaunch ep_bringup bringup.launch
```

3. 运行工具包

- 启动工具节点


```
1 rosrn camera_calibration cameracalibrator.py --size 9x6 --square 0.027 image
```

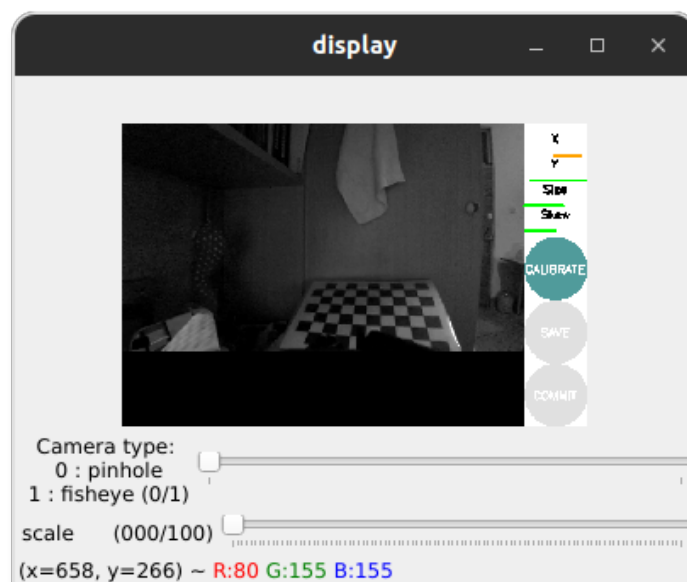
- `--size 9x6` 用于指明标定板的内角点数量
- `--square 0.027` 表示方格实际边长，单位：m
- `image:=/ep/camera/color/image_raw` 指明相机图像的 topic



- 移动标定板

分别在左右移动 (X)，直到标定界面上X下方的精度条为绿色，然后依次在上下 (Y)，远近 (Size)，倾斜 (Skew) 方向移动，移动过程尽量缓慢，避免大幅度的移动使相机成像产生拖影。持续移动直到所有的进度条都为绿色。

- 输出标定参数



CALIBRATE按钮由灰色变成深绿色，点击CALIBRATE，等待一会终端会输出标定结果


```
rosrun camera_calibration cameracalibrator.py --size 9x6 --square 0.027
/home/czx/ws/re... x rviz-d`rospack fi... x roslaunch camera_c... x rostopic echo tf x
rosrun camera_calibration cameracalibrator.py --size 9x6 --square 0.027 80x21
**** Calibrating ****
mono pinhole calibration...
D = [-0.0390559230545487, -0.010411321181939404, -0.001110832899114294, 0.001646
5888060415025, 0.0]
K = [314.9005209119623, 0.0, 329.98747930215467, 0.0, 314.10651285309314, 180.84
805102799896, 0.0, 0.0, 1.0]
R = [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
P = [295.2228698730469, 0.0, 333.49847880152083, 0.0, 0.0, 308.6782531738281, 18
0.00840015165704, 0.0, 0.0, 0.0, 1.0, 0.0]
None
# oST version 5.0 parameters

[image]

width
640

height
360

[narrow_stereo]
```

- 写入配置

将上图中结果写入配置：

```
1 # info 标定生成
2 info:
3     D: [-0.0738882155841218, -0.005760327982189524, -0.00086482682466968, 0.0036
4     K: [309.2180664557364, 0.0, 316.9312826802545, 0.0, 308.864233481454, 175.91
5     R: [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]
6     P: [274.54193115234375, 0.0, 321.5302651052116, 0.0, 0.0, 299.58294677734375
```

(4) 运行抓取任务

配合 (5) 参数调整运行

```
1 roslaunch control grab_task.launch
```

(5) 参数调整



约定：x 轴正方向朝左；y 轴正方向朝上；z 轴正方向朝前；yaw 角为小车姿态绕 y 轴偏转角

策略：two-stage

- 1. 先底盘控制使得机械臂在 x, z 方向有一个合适的值，并控制 yaw 角，对正物块
- 2. 再控制机械臂使得机械臂在 y, z 方向上达到目标点
 - 控制为有差的比例（P）调节，即通过当前值和目标值的差值，反馈给执行器，使得当前值向目标值偏移，当误差达到所能容忍的范围内，停止反馈校正


1. 底盘，x, z, yaw

- 配置文件：

```
1 chassis:
2   # x - leftward; y - upward; z - forward
3   x_target: -0.008
4   z_target: 0.25
5   yaw_target: 0.05
6   # The tolerance of error
7   x_tolerance: 0.005
8   z_tolerance: 0.02
9   yaw_tolerance: 0.05
10  # gain of the P control
11  x_gain: 4
12  z_gain: 1
13  yaw_gain: 1
```

- 参数含义及调整方法

参数	单位	含义	备注
x_target / z_target	m	x, z 方向上期望抓取时物块相对机械臂坐标系的位置	手动将底盘移动到合适的抓取位置，通过此时输出的目标位置修正
yaw_target	rad	期望抓取时物块相对机械臂坐标系的 yaw 角	通过摆正底盘，观察此时输出的目标 yaw 角修正
x_tolerance / z_tolerance	m	x, z 方向上可以容忍的误差	自己估计
yaw_tolerance	rad	yaw 角可容忍的误差	自己估计
x_gain / z_gain / yaw_gain	/	控制律： $u(k) = gain * [x(k - 1) - y(k - 1)]$ u(k)输出，x(k - 1) - y(k - 1) 偏差	不宜太大(< 10)

 在 `src/control/script/grab_task.py` 中 `ctrl + F` 搜索 `TODO` 可以看到调试参数所需的改动及提示，先在不加控制的情况下将目标值设定好，再逐渐增加 gain 到合适的值

2. 机械臂

- 配置文件

```
1 arm:
2   # the pose y, z offset
3   y_offset: -0.07
4   z_offset: 0.07
```

- 参数含义及标定方法

- 可以理解为机械臂末端到执行器末端（夹爪）的偏差值。
- 标定方法：通过给机械臂发布位置控制（相对于机械臂坐标系）命令，并将夹爪末端与目标 marker 对齐，根据输出的目标相对于机械臂坐标系的位置进行标定。

（6）策略调整（选做）

有能力的同学可以尝试修改 `src/control/script/grab_task.py` 的 `update()` 函数，对抓取策略进行更改。