Camera Calibration

参考资料

- 主要资料
  - https://gist.github.com/hshi74/edabc1e9bed6ea988a2abd1308e1cc96
- 其他选项
  - https://github.com/IFL-CAMP/easy_handeye

流程

- 关于相机位置的摆放
  - 用多个物体表示出机械臂能到达的边界，检查相机是否能拍全这些物体
  - 将三个相机连到一个usb3.2 hub，再将另一个相机直接连到主机的usb3.0接口，注意不要将它们相邻连接，否则可能会出现usb的带宽问题
- 安装 `realsense-ros`
  - 必须使用 ROS1-legacy https://github.com/IntelRealSense/realsense-ros/tree/ros1-legacy
  - `sudo apt-get install ros-$ROS_DISTRO-realsense2-camera`
- 对于每一个相机，使用 `apritag_ros` 获得它们相对 tag 的 pose
  - 基本直接照着教程做就可以
  - 对各个相机运行 `roslaunch realsense2_camera rs_camera.launch serial_no:=???????????? camera:=cam? depth_width:=640 depth_height:=480 color_width:=640 color_height:=480 depth_fps:=15 color_fps:=15 align_depth:=true enable_pointcloud:=true`
  - 输出结果：`pose/cam1_tag.yml`，`pose/cam2_tag.yml`，`pose/cam3_tag.yml`，`pose/cam4_tag.yml`
- 用示教模式把 arm 的 fingertip center 和 tag center 贴在一起，通过 `polymetis` 获得此时的 robot state
  - 代码文件：`write_robot_state.py`

    - 输出结果：`pose/robot_state.yml`
- 利用 cam2tag 和 robot2tag，计算 cam2robot poses
    - 代码文件：`compute_transform.py`
    - 输入：`pose/cam?_tag.yml` *4 和 `pose/robot_state.yml`
    - 输出结果：`pose/camera_pose_robot.yml`
- 可视化，把各个视角的 RGBD 通过 cam2robot poses 拼在一起
    - 使用 `pyrealsense2` 获得相机的 depth_frame & color_frame
    - 使用的函数 `open3d.camera.PinholeCameraIntrinsic`, `open3d.geometry.RGBDImage.create_from_color_and_depth`, `open3d.geometry.PointCloud.create_from_rgbd_image`
    - TODO: 研究一下 extrinsic matrix 和 cam2robot pose 的关系
    - 代码文件：`visualize_pcd.py`
    - 输入：`pose/camera_pose_???.yml`
- 利用桌面作为共同的支撑平面对计算出 poses 做 finetune
    - 修改教程中的代码，直接使用 pyrealsense2 来获取点云
    - 代码文件：`align_plane.py`
    - 输入：`pose/camera_pose_robot.yml`
    - 输出结果：`pose/camera_pose_aligned_plane.yml`
- 利用 `rviz` 微调结果