RobotVisionSuite

拆垛案例

RobotVisionSuite

图漾科技

2023.02

关于本手册

本手册主要介绍如何利用图漾 RobotVisionSuite 平台软件（下简称 RVS）进行简单的拆垛项目。文档结构如下:

章节	标题	内容
第一章	RVS 视觉平台-拆垛案例	包含 RVS 简介，自动拆垛案例背景，以及离线文件的使用
第二章	手眼标定	初步了解 RVS 完成工程项目，并利用软件完成手眼标定
第三章	拆垛主流程	学习如何使用 RVS 坐标系转换、识别定位等
第四章	AI 训练	针对密集物体，学习如何采集图像、标注、训练。
附录 A	支持资源	介绍软件工具和相关的技术与文档支持。

发布说明

日期	版本	发布说明
2022.05	V1.0	第一次发布。
2022.10	V1.1	更新了交互面板的显示和 AI 推理模块的 MasKRCNN 算子
2023.02	V2.0	更新 XML。

免责和版权声明

目录

第一章 RVS 视觉平台-拆垛案例

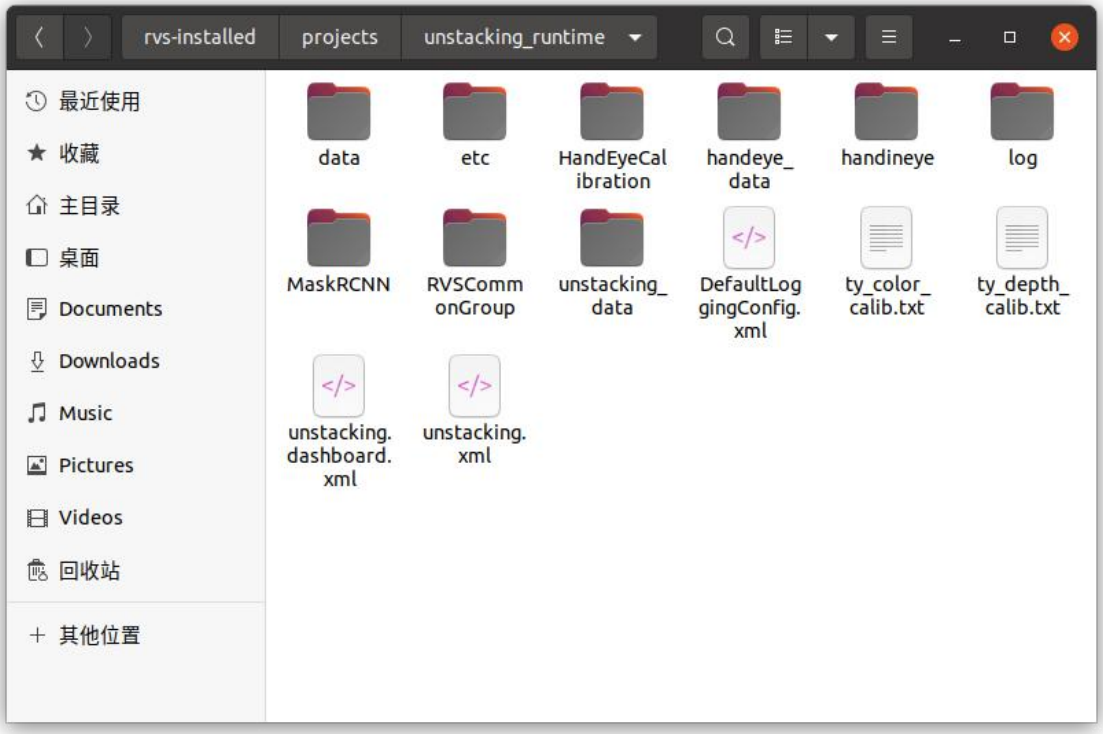
RobotVisionSuite（以下简称 RVS）是图漾科技有限公司（以下简称图漾）自行研发的一款 3D 机器视觉开发平台。RVS 中可以将 3D 视觉算子写成节点（Node）以供后续开发使用，各个算子节点可以在软件平台中相互连接以及嵌套成组，形成完整的项目程序流程。可以利用已有的算子快速搭建项目、验证项目可行性、开发出原型程序以供客户测试。为了方便用户的快速入手，图漾目前已经储备了大量常用节点，包括图漾 3D 相机的 SDK 调用节点，常用的 2D&3D 图形图像处理节点，常用工业机器人接口节点，视觉标定与手眼标定节点，通讯节点，图像深度学习训练与推理节点（目前支持 MaskRCnn）等等。节点的运算结果（字符串、图像、点云、位置姿态等）都可以在 RVS 中进行可视化。

RVS 的高效不仅仅体现在软件的优势，更主要的是与工业项目的实际结合。本手册的目的不仅仅是向用户演示如何利用 RVS 实现自动拆垛的项目落地，还旨在向用户展示软件在实际操作使用中的实用性和操作性。

自动拆垛案例背景：

- 功能要求：客户现场使用图漾相机，配合工业机器人将箱型从垛盘中逐一取下，搬运至流水线进行运输
- 作业场景：多种箱型，长宽高均不同，箱型在垛盘中摆放多层，同时会出现箱型倾斜位置不固定的情况
- 项目要求：1.找到箱型几何中点 2.根据一定顺序进行抓取

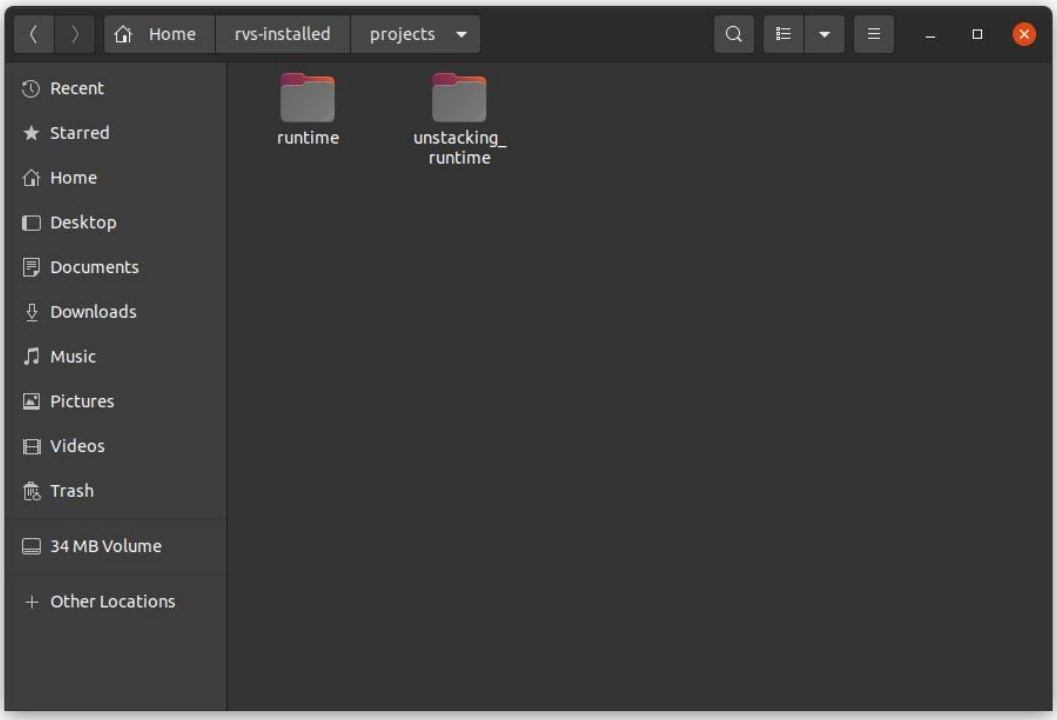
当您获得本文档时，您也应该同时获得 unstacking_runtime.zip 文件。离线文件的作用包含，当您拥有相机和机器人时，可以参考本文件中的 xml 进行检查；当您不具备相机或机器人时，加载离线本地文件，也可以获得完整软件操作体验。解压后其内容应包括：



其中：

- data 包含了本案例中所使用的机器人模型（部分），如果您需要其他模型请前往 RVS 软件发布模型网站下载。
- HandEyeCalibration 适用于进行手眼标定，其中包含了 TCP_test.xml 用于通讯测试；Camera_save.xml 用于录制简单的一些数据，其中请注意 SaveData 算法组的数据保存路径；HandEyeCalibration.xml 用于手眼标定的数据录制和矩阵计算，其中请注意交互面板中的 numeber_rows/cols 正确填写标定板的行列数、square_size_in_millimeter 正确填写标定板的方格尺寸、handeye_datadir 中数据保存和加载的路径是否一致，最后请根据您模式选择 eyetohand 还是 eyeinhand。
- handeye_data 是手眼标定中标准数据录制的案例，用户可直接加载相应路径进行计算。
- MaskRCNN 适用于 AI，其中 train_data 中包含了 AI 训练所录制的图像和已经训练好的 pth 和 annotations.json 文件，用户直接加载 AI 所需文件时注意路径；ty_ai_savedata.xml 适用于 AI 图像录制；ty_ai_train.xml 适用于 AI 训练。
- RVSCommonGroup 中提供了常用的一些算法组，包含保存加载和矩阵转换，根据需要加载即可。
- test_data 是手眼标定中简易数据录制的案例，仅作参考。
- unstacking_data 适用于拆垛程序离线模式数据的加载，使用时请注意路径正确。
- ty_color/depth_calib.txt 适用于 AI 拆垛程序的 ProjectMask 参数加载，使用不同相机时会更新这个文件内容。
- unstacking.xml 适用于拆垛程序，可做参考，当加载离线数据时，请先更新 Local 中离线数据的路径，点击 Restart 后再点击 LocalCapture 迭代离线数据。

拆垛的整个流程将由手眼标定和拆垛主流程两大模块进行。案例所有使用到的文件均保存在 unstacking_runtime 下，请放在 RVS 安装目录下的 projects 内运行。如：



注：本文档所使用 RVS 软件版本为 1.5.0，若在使用中出现问题或版本不兼容问题，请及时与我们反馈，发送邮件 rvs-support@percipio.xyz！

第二章 手眼标定

一、 手眼标定介绍

手眼标定是指，在相机安装完成后，计算出相机同机器人的坐标转换矩阵，目的是根据这个坐标转换矩阵将相机坐标系下拍摄的点云转换到机器人坐标系下，根据相机安装方式，手眼标定分为两种情形：

- 相机安装在机械臂上 HandInEye 标定板固定位置不动，手眼组合体变换姿态拍摄图片
- 相机（吊顶）固定安装 HandToEye 相机固定位置不动，机械臂抓取标定板变换姿态拍摄图片

本文将以 HandInEye 为例，过程总共分为四大步骤：通讯测试、数据录制、计算标定结果、使用标定结果，接下来将按照顺序分别进行介绍。

二、 通讯测试

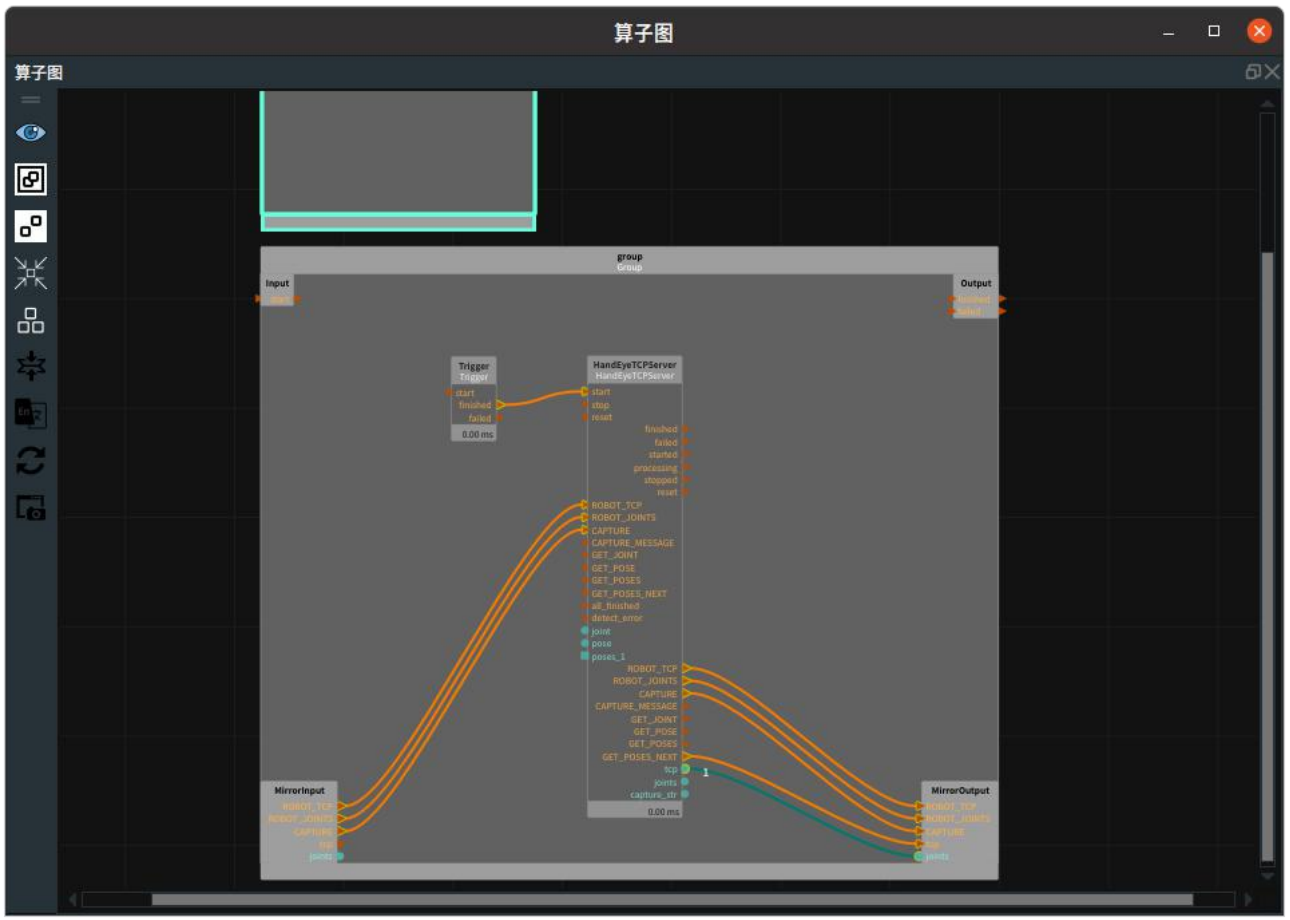
TCPCommad 类算子的作用是在 RVS 客户端建立 tcp_server 服务端 ,等待机器人建立连接并进行 tcp 通讯。通讯测试的目的是确保 RVS 软件能正常启动，并和机器人正常建立通讯，为接下来的数据录制做准备。

具体参数请参阅 RVS 软件使用手册_基础版。

操作流程：

1) 首先打开 RVS 软件，在 unstacking_runtime 路径下新建一个工程，另存路径为 unstacking_runtime/HandEyeCalibration/,并重新命名为 TCP_test.xml
2) 在算子浏览器中，搜索 HandEyeTCPServer，拖动到算子图中，无需修改任何参数
3) 在算子浏览器中，搜索 Trigger，拖动到算子图中，type 属性选择“InitTrigger”。
4) 连接两个算子之间的信号流 ,将 InitTrigger 中的 finished 信号和 HandEyeTCPServer 中的 start 信号连接
5) 同时选择 InitTrigger 和 HandEyeTCPServer，将他们合成一个 Group
6) 先将 HandEyeTCPServer 算子右侧的 ROBOT_TCP、ROBOT_JOINTS、CAPTURE 分别连接到 MirrorOutput 上，再从 MirrorInput 中和 HandEyeTCPServer 算子左侧的相应词条连接。这样形成一个循环是为了接收到指令时有信息可以返回给客户端。

最后的项目流程应如图所示：



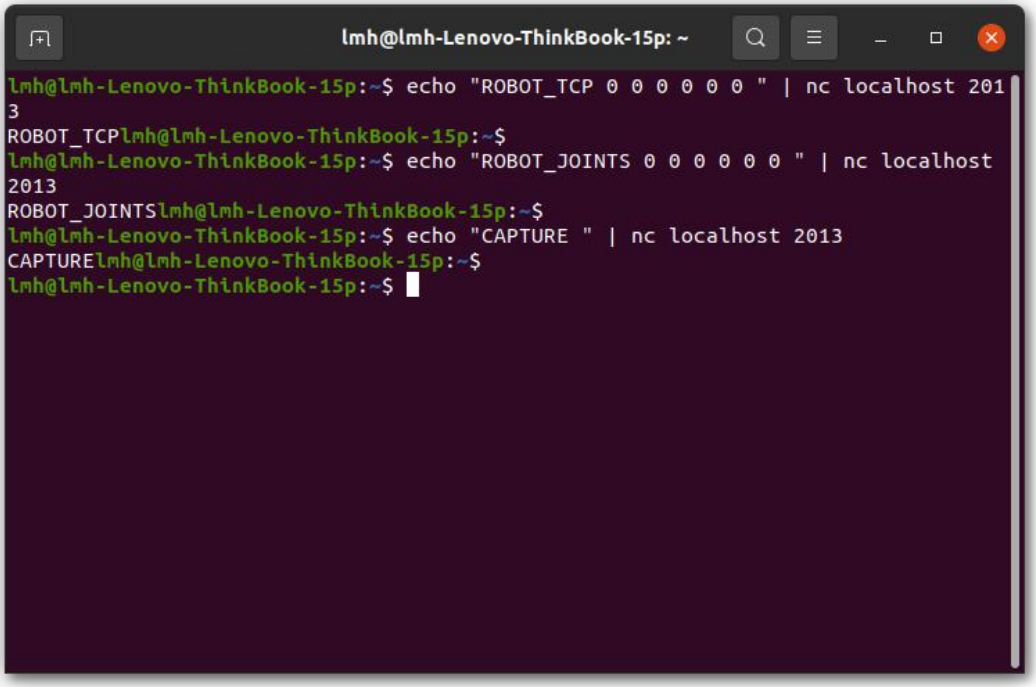
完整的 xml 已提供在 unstacking_runtime/HandEyeCalibration/TCP_test.xml 路径下。

离线测试：

在项目启动后，可以在本地进行测试，打开终端分别输入（注意结尾空格以及结尾符，算子中设置 RT，则无需更改，算子设置；等结尾，需在结尾空格后添加相应结尾符）：

- echo "ROBOT_TCP 0 0 0 0 0 0 " | nc localhost 2013
- echo "ROBOT_JOINTS 0 0 0 0 0 0 " | nc localhost 2013
- echo "CAPTURE " | nc localhost 2013

如果成功运行，则会在终端中分别返回 ROBOT_TCP、ROBOT_JOINTS、CAPTURE 字符，如图所示：



在线测试：

请机器人工程师协助：

1.机器人获得当前 TCP 值 ,通过连接向 RVS 服务端发送 ROBOT_TCP x y z rx ry rz \n 字符串(注意结尾空格以及结尾符，算子中设置 RT，则无需更改，算子设置；等结尾，需在结尾空格后添加相应结尾符，下同），x y z 以米为单位，rx ry rz 以弧度为单位

2.机器人获得当前位置的 joints 关节坐标值 ,取得前六个关节的弧度值 ,发送 ROBOT_JOINTS J1 J2 J3 J4 J5 J6 \n 给 RVS 服务端

3.机器人端发送 CAPTURE \n 字符串给 RVS 服务端

三、 数据录制

当我们已经完成通讯测试后，代表机器人可以正常与 RVS 软件通讯。那么接下来我们就可以进行数据录制的环节，数据录制过程中，我们需要记录多组、同一时刻下相机拍摄的 RGB 图、点云图，以及机器人当前的 TCP、JOINTS 数据，并保存在本地文件夹下，对于一组数据，父路径命名规范为 unstacking_runtime/test_data/时间戳，如 unstacking_runtime/test_data/20220301152509156。

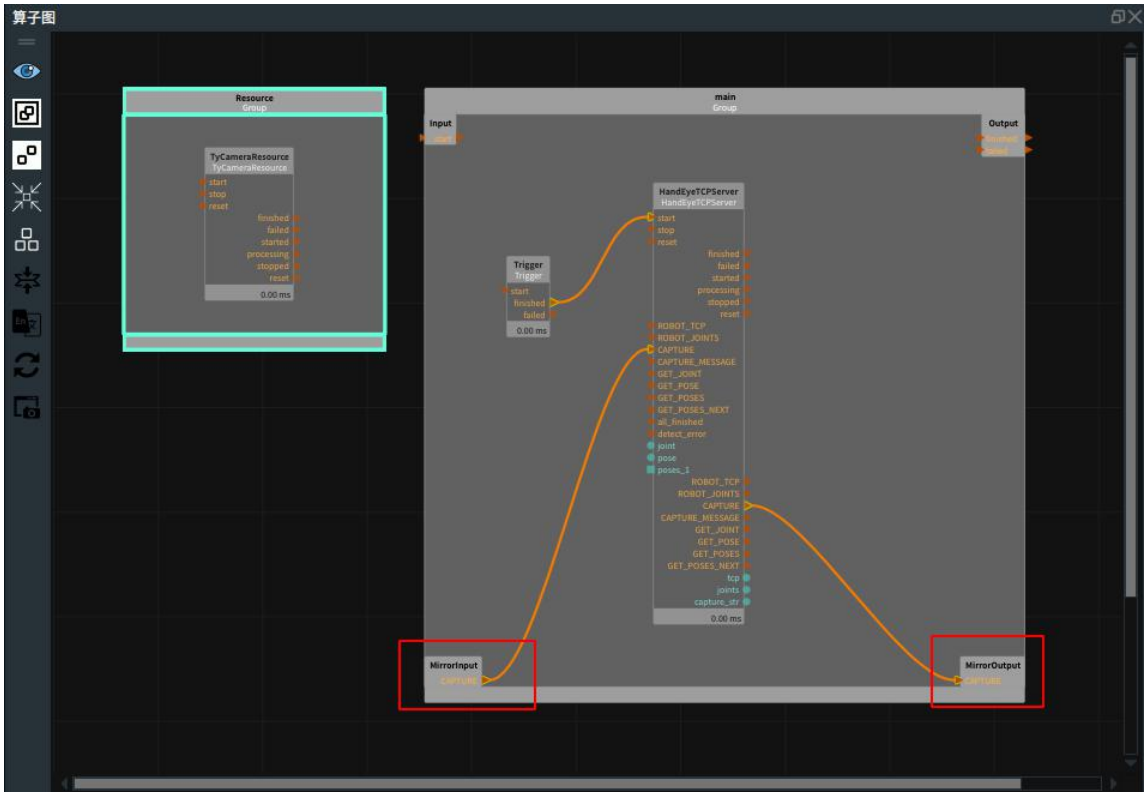
接下来，将通过一个简单的 xml 展示如何在 RVS 中使用图漾的 3D 相机并保存 RGB 图片和点云图，如若这部分您已经了解，可以跳往本章节标准手眼标定模块。

操作流程：

1) 首先打开 RVS 软件，在 unstacking_runtime 路径下新建一个工程，另存路径为 unstacking_runtime/HandEyeCalibration/,并重新命名为 Camera_save.xml

2) 与通讯测试内容相同，我们需要 HandEyeTCPServer 和 Trigger(type 属性选择

InitTrigger）两个算子并形成 Group，并重命名 Group 为 main，在将这两个算子连接后，我们仅需将 HandEyeTCPServer 算子右侧的 CAPTURE 连接到 MirrorOutput 上，再从 MirrorInput 中和 HandEyeTCPServer 算子左侧的 CAPTUR 连接。



3） 在工具栏中最上方点击资源，出现下拉栏，选择 TyCameraResource，在算子图中的 Resource Group 中就可以看到。

4） 点击 TyCameraResource，在属性面板中勾选 auto_start 和 output_pointcloud，其余参数

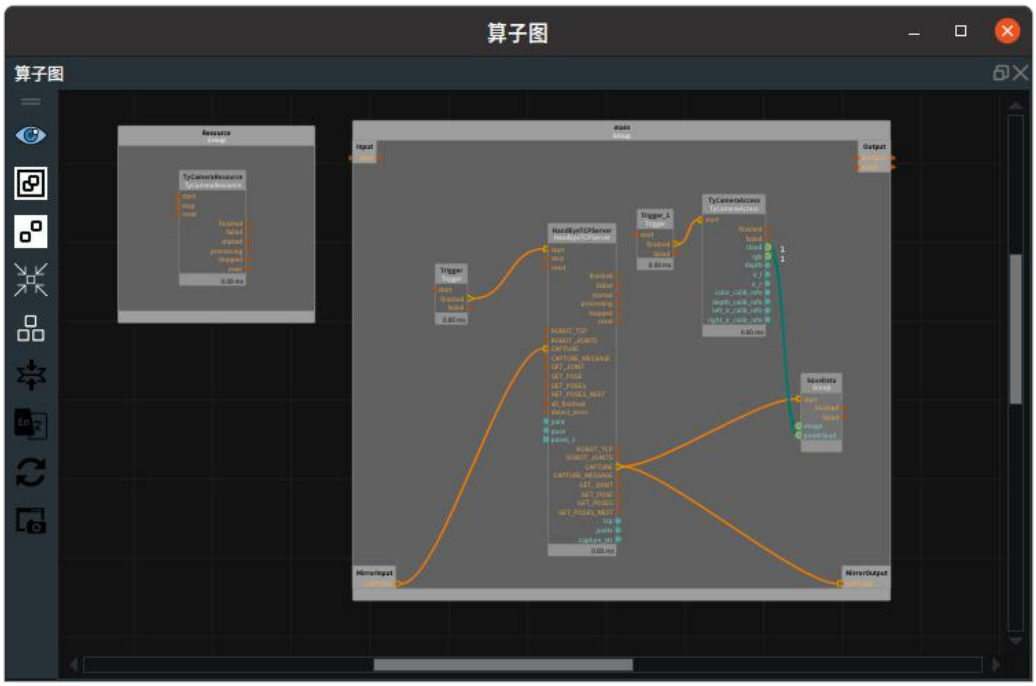
会在 RVS 软件第一次连接到相机时自动补全。

5） 在算子浏览器中搜索 TyCameraAcess 算子拖动到 main Group 中，在属性面板中打开 cloud 和 rgb 可视化属性，将 cloud_color 设置为-2，代表真实颜色，并给予一个 Trigger 节点连接 TyCameraAcess 去触发，Trigger 的属性面板中可以勾选 loop，这样在 3D 和 2D 视图中就可以看到不断刷新的相机画面。

6） 在 main Group 中空白处右键，在弹出的菜单栏中选择在此处导入 Group XML，选择 unstacking_runtime/RVSCommonGroup/SaveData.group.xml ,这个 group 的功能是保存 RGB 图片和点云图，点击这个 group，在属性面板中找到 savedata_parent_dir 词条，调整文件目录为 test_data/。

7） 将 HandEyeTCPServer 右侧 CAPTURE 信号和 SaveData 的 start 信号连接，这意味着当 TCP 中 CAPTURE 被触发时，不仅仅会回复 CAPTURE\n 字符串，还会触发保存数据的功能。同时将 TyCameraAccess 中的 cloud、rgb 数据流连接到 SaveData 相应的 image 和 pointcloud 作为输入。

最后的项目流程应如图所示：



完整的 xml 已提供在 unstacking_runtime/HandEyeCalibration/Camera_save.xml 路径下。

请先连接相机再启动此工程项目，启动加载需要几秒的时间，即在搜索局域网内空闲的图漾相机，当成功连接时，会在 log 中提示类似如下信息（本手册测试相机为 FS820-E1）：

成功后可以在本地进行测试，打开终端输入：

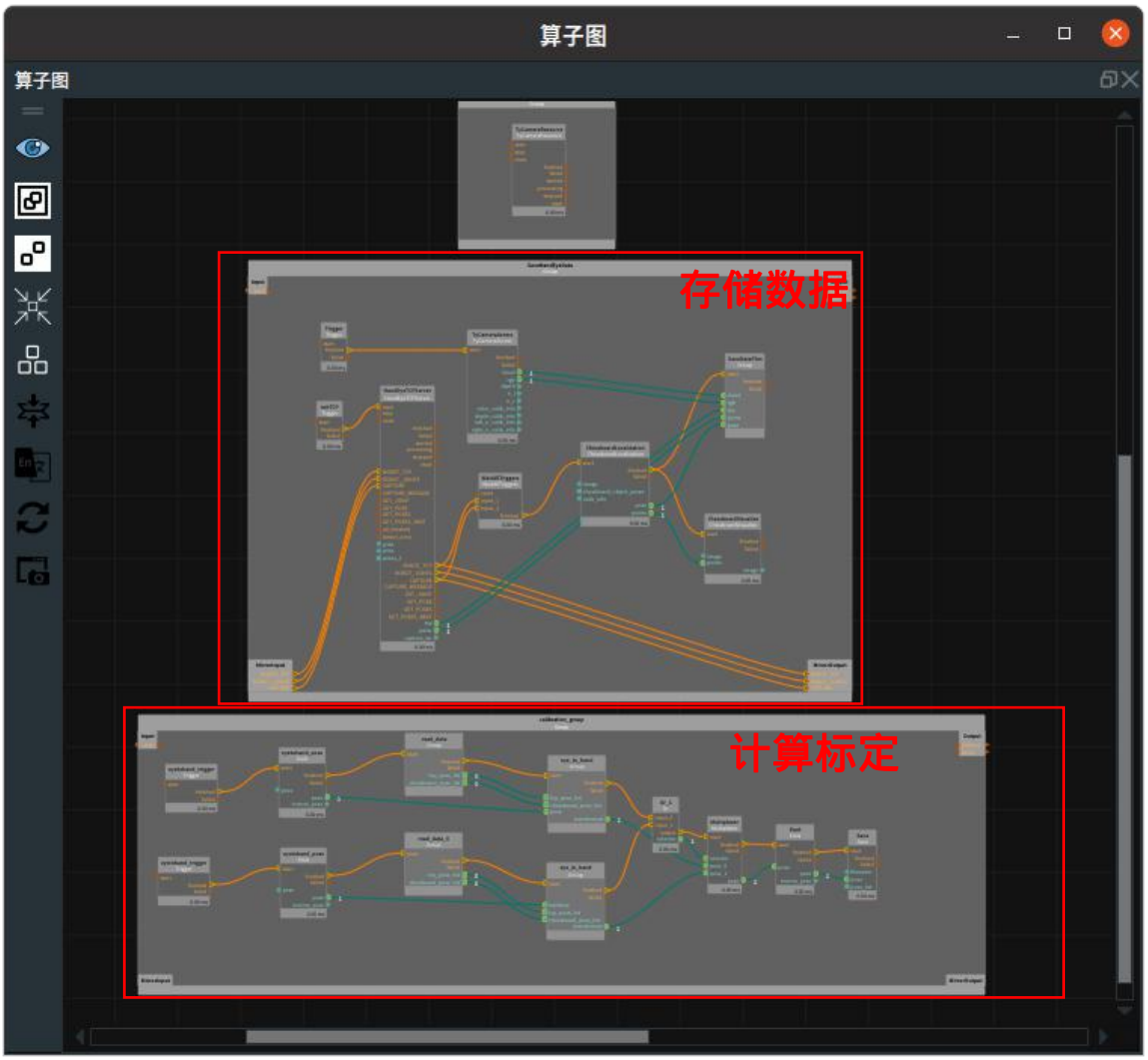
- echo "CAPTURE "｜nc localhost 2013

或使用机器人发送 CAPTURE \n 字符串给 RVS 服务端

若正常运行，则每次触发都会在 unstacking_runtime/test_data/下不断新建文件夹，内容包含 rgb.png 和 cloud.pcd。（test_data 文件夹如果之前不存在，则第一次运行会自动创建）

当您已经熟悉了如何使用 RVS 软件使用、连接相机后，将正式开始数据录制的环节。在 RVS 中提供了标准化的手眼标定模块。

请打 unstacking_runtime/HandEyeCalibration/HandEyeCalibration.xml



在这个 xml 中，整体上分为存储数据模块和计算标定结果，其中交互面板中：

操作流程：

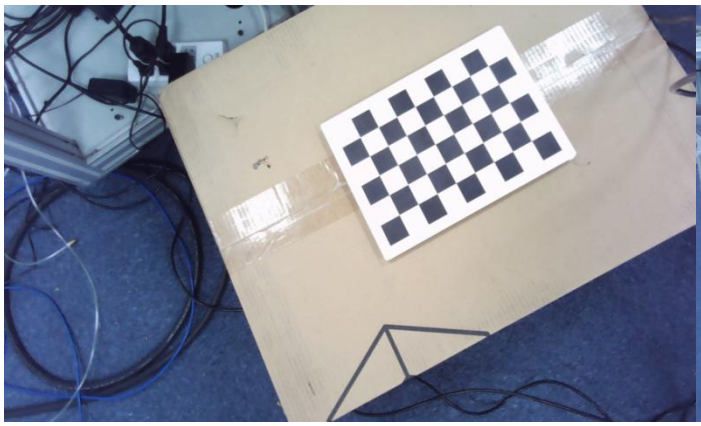
1) 首先打开 RVS 软件，加载 unstacking_runtime/HandEyeCalibration 路径下的 HandEyeCalibration.xml 工程文件

2) 正确填写标定板的行列数，和标定板格子的单位长度,和数据保存的文件路径

3) 启动工程后（如遇到红色报错，通道为 rvs_tycamera，消息为 no TyCamera avaliable，这意味着您上次关闭没有将循环显示功能关闭就保存退出，请点击 loopshow 先关闭循环显示，再打开 open_camera 后重新点击 loopshow 循环显示），首先点击 open_camera，（如遇到红色报错，通道为 rvs_tycamera，消息为 Cant open XXX.txt to export depth/colorcalib!,则请点击 TyCameraResource 将 depth_calib_file 和 color_calib_file 路径更换为 unstacking_runtime/HandEyeCalibration，选择相应的 txt 文件），再点击 loopshow 刷新图像；

4) 在保证标定板完整的出现在 2D 视图的视野中后，控制机器人连续发送 ROBOT_TCP、ROBOT_JOINTS、CAPTURE 字符触发保存数据功能（此处需机器人工程师协助完成），ChessboardLocalization 算子会判断视野内的角点数是否符合要求，若符合则正常保存，如不符合则会剔除无效数据，ChessboardVisualize 算子会在 2D 视图中同步显示已经成功保存的图片及角点。额外请注意：

- 确认标定板的角点清晰，不能有重叠或者间隙，行列一奇一偶
- 标定板在标定过程中位置始终固定，不能有任何的旋转移动，并且保障它可以完成的出现在画面内
- 注意灯光对于标定板的影响，角点处不能有过曝的情况，也不需要特别暗
- 位置姿态多进行变动，姿态一定要丰富，高度有多层次的变化，可以使标定板出现在图幅的 4 个边角处，有多中姿态的旋转
- 尽可能多地核对每一组文件夹下的 tcp、joints 数据，是否与发送数据时机器人控制面板上的数字一致
- 组数一般在 18 组以上，20-25 组为宜




四、 计算标定结果

当我们已经完成数据录制之后，请再次检查文件夹下的数据是否完整，图像及角点是否清晰，抽样检查 tcp 和 joints 是否和真实情况一致。随后我们开始计算标定结果：

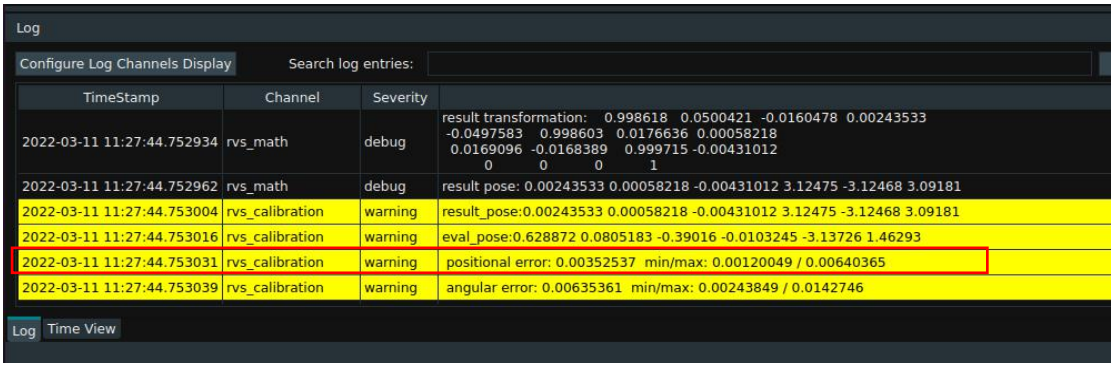
操作流程：

1) 仍然在 unstacking_runtime/HandEyeCalibration/HandEyeCalibration.xml 工程下

2) 确认读取路径是否与保存路径是否一致，在交互面板中正确选择路径后，点击相应的模式开始计算，这里以 eyeinhand_start 为例

3) 点击交互面板上的 eyeinhand_start 按键，即开始自动计算结果，在程序的 Log 窗口会通过黄色警告的方式显示出误差是多少，如下图

4) 如何判断误差是否正常？不同机器人配合不同的相机类型会有不同的精度表现，一般来说 positional error 在 0.005（5 毫米）以内，则比较理想；如果误差很大，请检查前述步骤是否正确执行



五、 使用标定结果

当我们成功计算出手眼标定的转换矩阵，且精度满足误差范围内的波动允许，我们就可以将转换矩阵应用于点云转换过程。

操作流程：

1) 首先额外打开 RVS 新窗口，在 unstacking_runtime 路径下新建一个工程（只做演示无需保存）
2) 在算子图空白处右键，选择在此处导入 Group Xml
3) 对于不同模式有不同的使用方法

- HandInEye 模式，请选择 unstacking_runtime/RVSCommonGroup/HandInEye_Depth2Robot.group.xml,将这个 group 展开后，找到 rgb2tcp 算子，将之前标定程序中的交互面板上的 resultpose 结果复制，粘贴到此处；在标定程序中点击 TyCameraResource,在属性面板中找到 extrinsic，将相机外参复制填入到 rgb2depth 算子
- HandToEye 模式，请选择 unstacking_runtime/RVSCommonGroup/HandToEye_Depth2Robot.group.xml,将这个 group 展开后，找到 rgb2robot 算子，将之前标定程序中的交互面板上的 resultpose 结果复制，粘贴到此处；在标定程序中点击 TyCameraResource,在属性面板中找到 extrinsic ,将相机外参复制填入到 rgb2depth 算子

第三章 拆垛主流程

一、 拆垛流程

经历过手眼标定过后，您一定对于 RVS 和机器人通讯及软件的使用有一定整体上的了解。在拆垛流程中，我们的最终需求是：1.找到箱型几何中点　2.根据一定顺序进行抓取，因此我们将拆垛流程主要分为以下 4 个过程：

- 加载本地数据：本案例中使用离线数据完成拆垛项目。
- 坐标系转换：将相机坐标系下的点云转到机器人坐标系下
- AI 推理：箱子摆放紧密时使用 AI 推理。
- 识别定位：识别到目标点云平面的中心点，并返回给机器人进行移动
- 机器人抓取：使用机器人模拟按照顺序进拆垛。

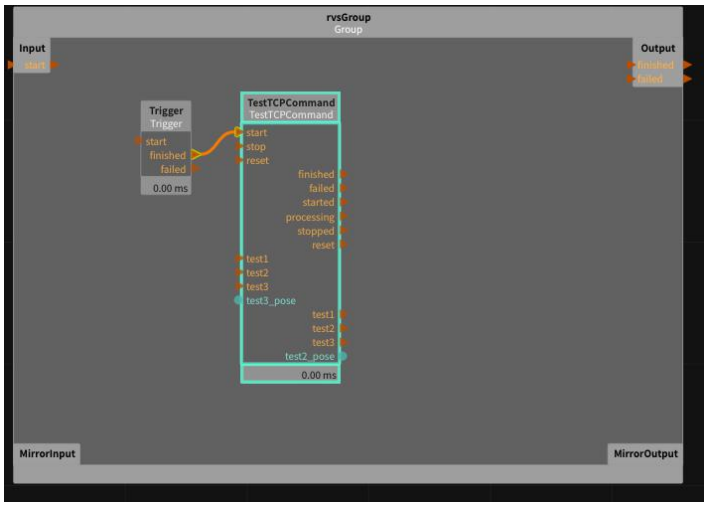
二、 加载本地数据

在此案例中，unstacking_data 文件夹下保存了 6 组数据，我们需要加载这些数据并遍历取完成拆垛项目。

操作流程：

1）新建工程项目 Unstacking.xml；在算子图中添加一个 TestTCPCommand，作为与模拟机器人通信的算子 ,获取机器人当前的 TCP。将 TestTCPCommand 算子的属性 server_mode 设为 Once 模式，为单次与机器人通信模式；

2）拖入 Trigger 算子，type 属性选择“InitTrigger”，并重新命名为“InitTrigger”，用于第一次运行 Xml 时，自动触发后续算子。

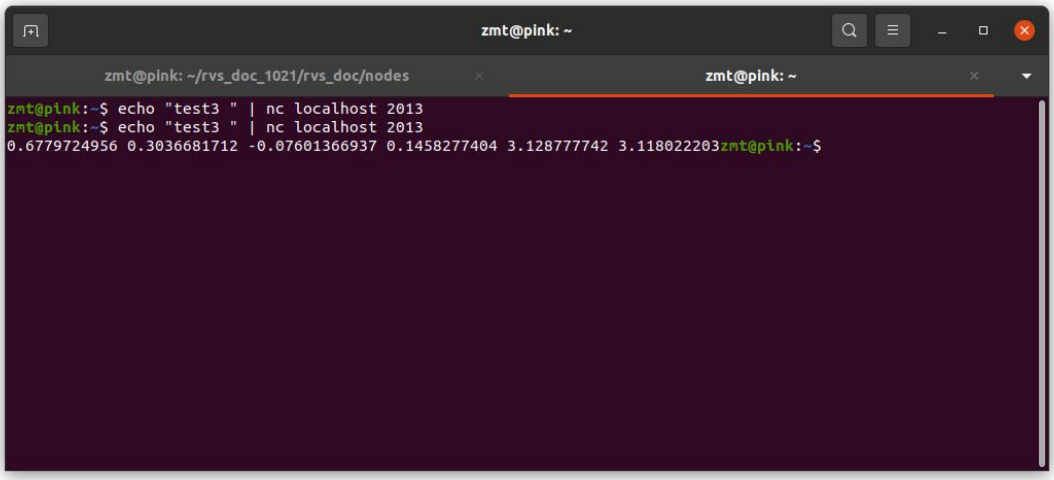


3）拖入 Trigger 至算子图中。将 Trigger 重命名为“LocalCapture”,当您希望获取模拟机器人 TCP 时也可通过手动触发 Trigger 来获取。将该算子的 Trigger 属性曝光 ,与交互面板中的输入工具——“按钮”进行绑定。将该按钮重命名为 LocalCapture。将 trigger 属性曝光 ,并与交互面板中输入工具——“按

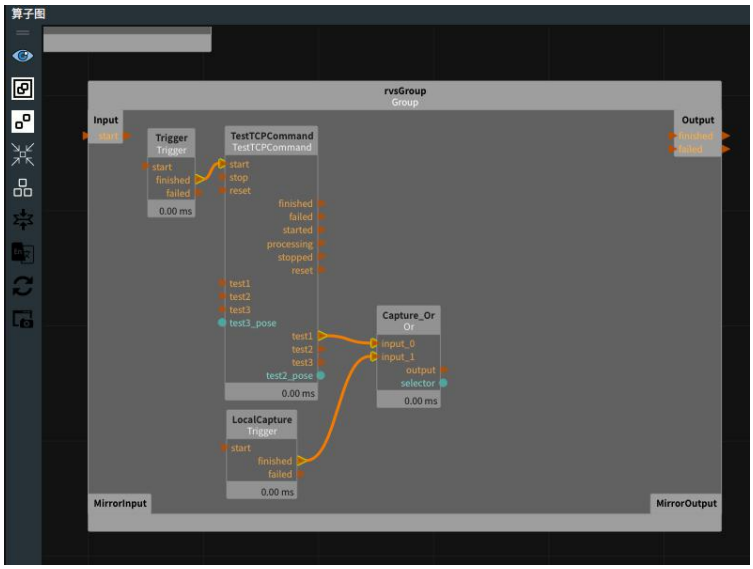
钮”进行绑定，将按钮重命名“LocalCapture”。

4）拖入 Or 算子至算子图中，并重命名为“Capture_Or”,可以使用 TCP 或者本地两种方式触发后续算子。

5）将 TestTCPCommand 算子的 test3 输出端口，连接到“Capture_Or”，当终端按照一定报文发送“echo "test3 " | nc localhost 2013”触发此算子。完成整个工程后，输入和结果返回如下：



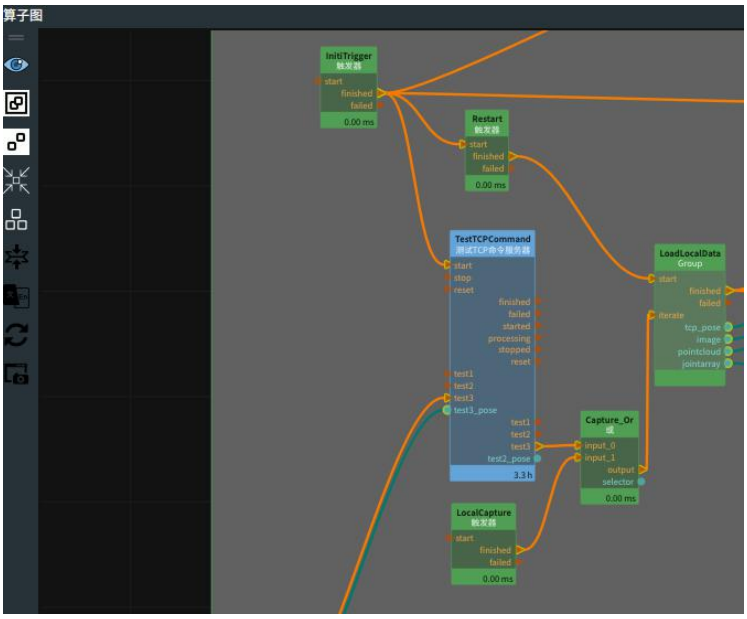
6）具体连接如下图所示：



7）在上步的算子组中导入 unstacking_runtime/RVSCommonGroup 中的 LoadLocalData 组，将 Or 算子的 output 端口连接到该算子组的 iterate 端口。

8）拖入 Trigger 重命名为“Restart”，用于重新加载本地数据。输入输出端口分别连接 InitTrigger 和 LoadLocalData 算子。可以在 2D 视图中看到离线的图片。将 trigger 属性曝光，并与交互面板中输入工具——“按钮”进行绑定，将按钮重命名“Restert”。

9）双击 LoadLocalData 算子组展开该组。找到名为“LoadImage”的算子，打开 image 可视化属性；打开“LoadCloud”算子的 cloud 属性。可以在 3D 视图中看到离线的点云。



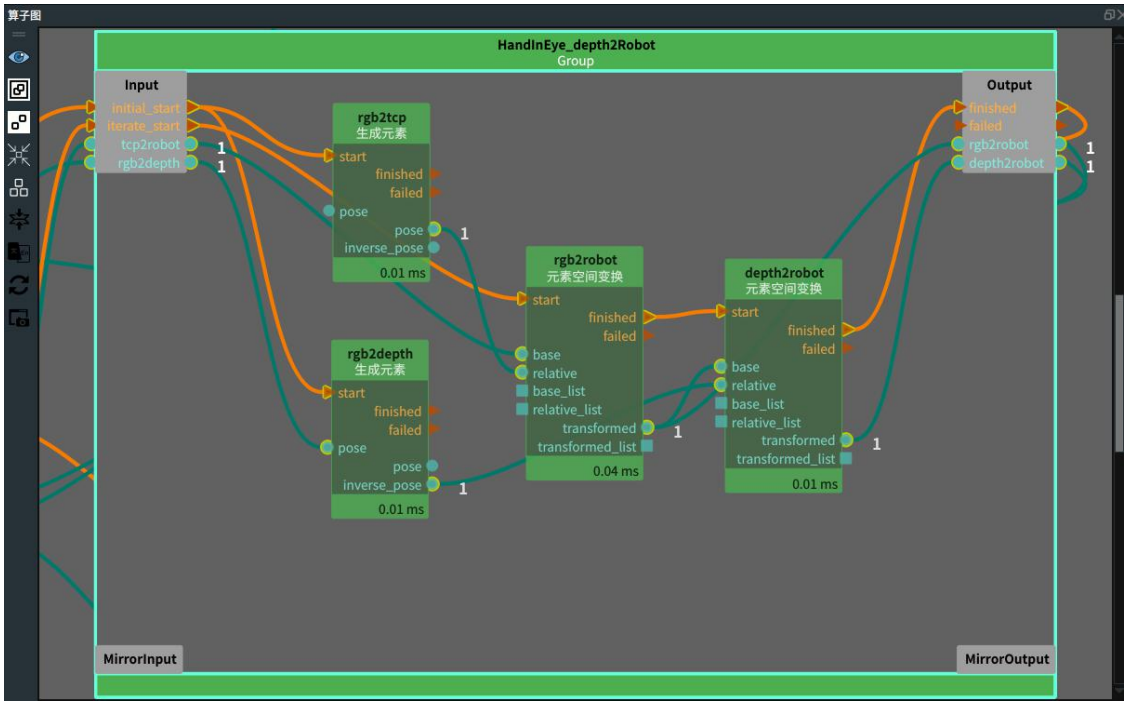
10）最后，直接导入 LoadLocalData 时，请注意调整 DirectoryOperation 算子中 parent_directory 属性的文件路径。

三、 坐标系转换

当您成功的使点云在 3D 视图区域显示出来后，我们开始进行下一步操作：坐标系转换。此操作旨在将点云所处的坐标系——相机 rgb 镜头坐标系转换至机器人坐标系，这一转换涉及相机外参及手眼标定结果。

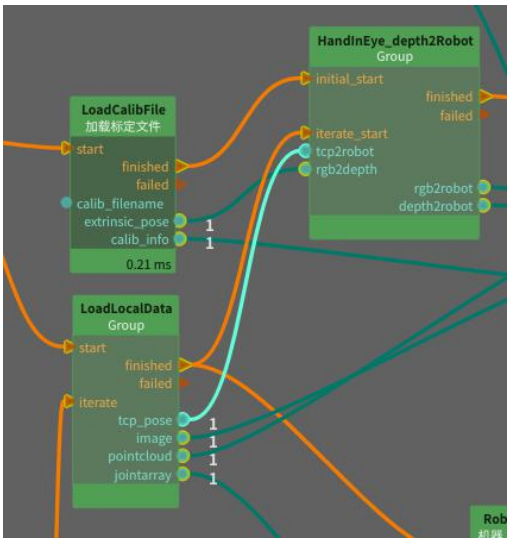
操作流程：

1）在算子图中右键选择在此处导入 Group XML，导入 RVSCommonGroup 中的 HandInEye_Depth2Robot.group.xml。需要注意的是，除了该文件之外，还有 HandToEye_Depth2Robot.group.xml，在实际的项目中，请根据您实际的相机安装方式选择对应的算子组。

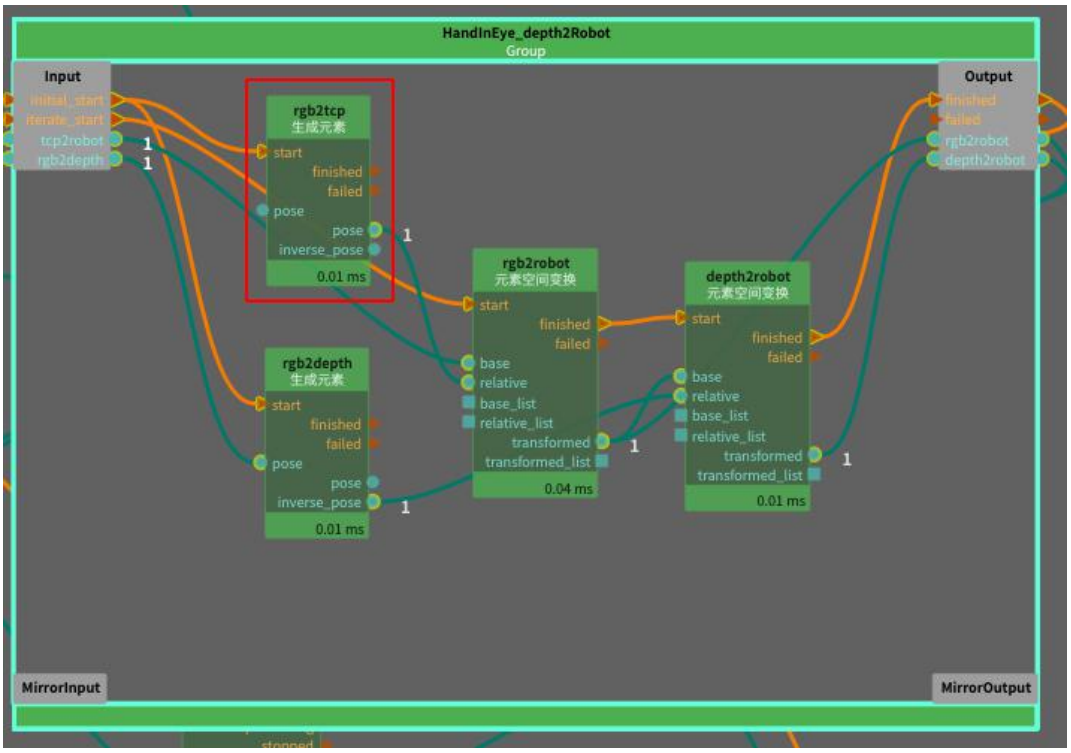


2）LoadLocalData 组的 tcp_pose 端口与 HandInEye_depth2Robot 组的 tcp2robot 端口连接。

3）拖入 LoadCalibFile 算子，用于加载标定文件，finshed 端口连接至 HandInEye_depth2Robot 组的 initial_start 端口；extrinsic_pose 端口与 rgb2depth 端口连接；start 端口与 InitTrigger 端口 finished 端口连接。具体连接如下：

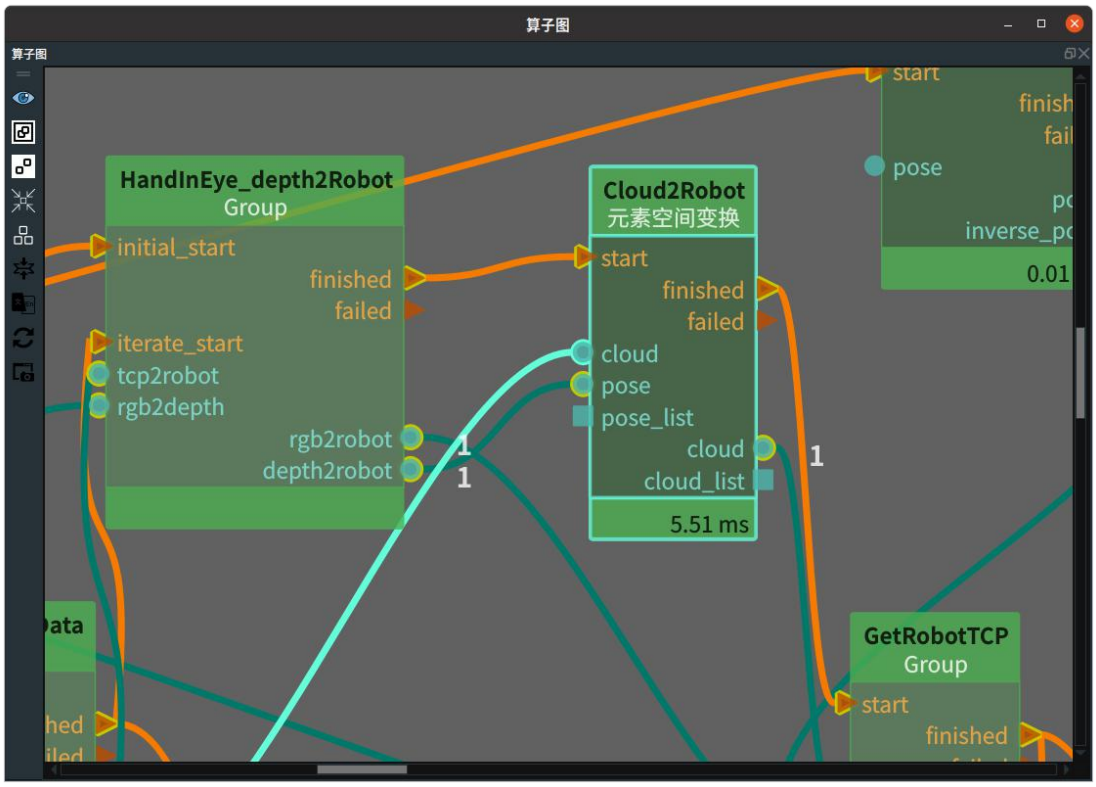


4）双击 Group，找到 rgb2tcp 算子，在属性面板的 pose 属性处，粘贴手眼标定的结果。



5）通过前述步骤，我们已经获取了相机 rgb 镜头转机器人坐标系的矩阵 rgb2robot 和相机深度镜头转机器人坐标系的矩阵 depth2robot ,此处我们将相机深度镜头坐标系下点云转换至机器人坐标系下。

6）首先拖入 Transform 算子，type 属性选择“PointCloud”，将 depth2robot 端口连接至该算子的 pose 输入端口，将 LoadLocalData 算子组的 pointcloud 端口连接到本算子的同名输入端口。至此，您已可以从输出端口获取机器人坐标系下的点云。



四、 AI 推理

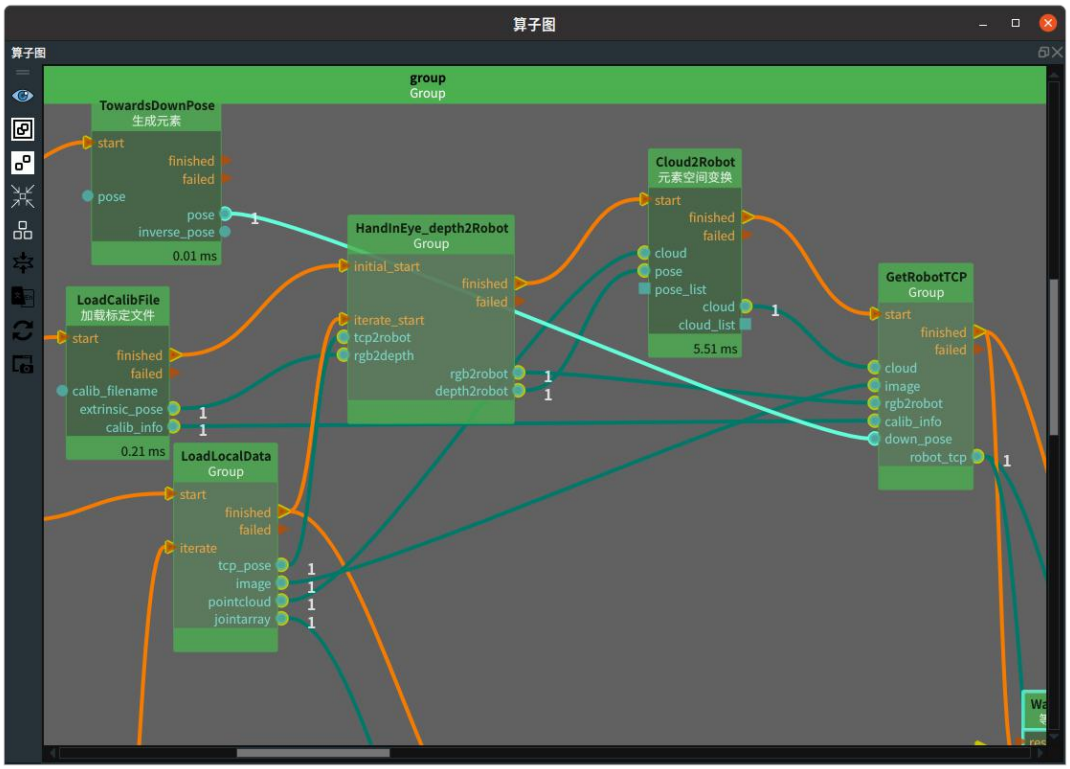
当我们在实际工作中，箱型摆放是很紧密时，多个箱子的上表面点云会连在一起变成不易分割的一块点云，针对这个问题我们可以为我们的工程项目添加一些包含 AI 部分的功能模块，由于 AI 训练需要一定配置（详情参考 RVS 安装教程），如果未能满足配置要求，可以根据官方提供的 pth 文件直接进行推理。总的流程将分为录制、标注、训练、推理。

采集训练图像、标注训练图像、训练 AI 模型教程放在下一章节中。

1）加载 unstacking_runtime/RVSCommonGroup/GetRobotTCP 组，将“Cloud2Robot”中 cloud 端口与该组的同名端口连接，“LoadLocalDate”组中 image 端口与该组同名端口连接，“HandInEye_depth2Robot”组中 rgb2Robot 端口与该组同名端口连接，“LoadCalibfile”算子中 calib_info 端口与该组同名端口连接。

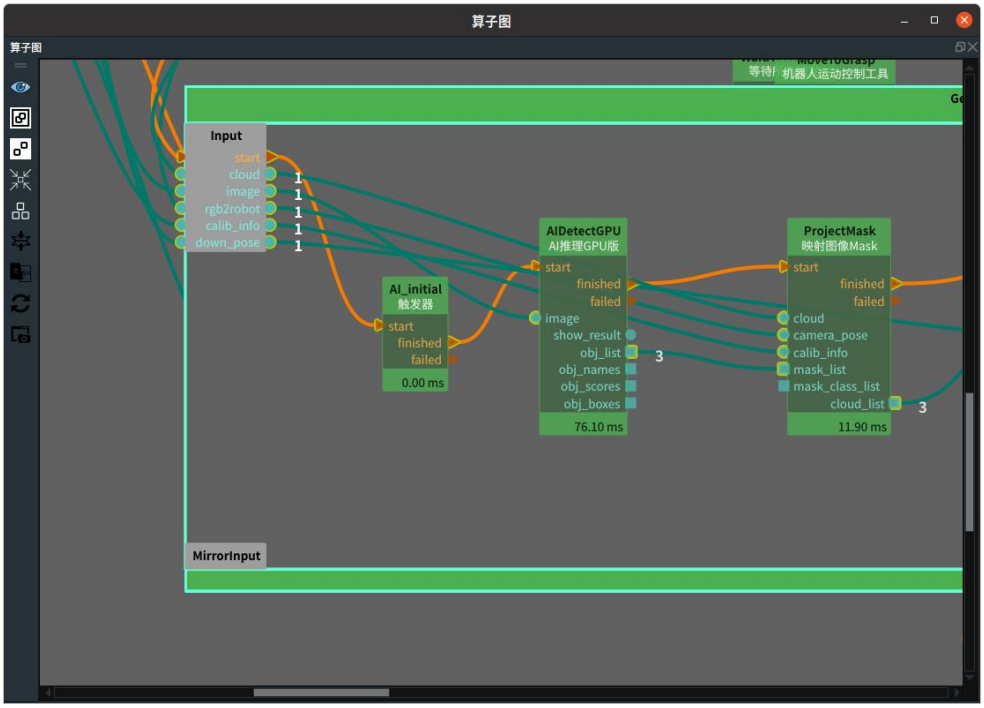
2）拖入一个 Emit 算子，type 属性选择“pose”，重命名为“TowardsDownPose”，将 pose_roll 属性输入“3.141592654”。这个算子在后续的算子中使用。将该算子中 pose 端口与 GetRobotTCP 组 down_pose 端口连接。

3）具体连接如下图所示：



4）双击展开 GetRobotTCP 组，我们需要预先使用 MaskRCNN 网络对数据进行训练，将其中的 AIDetectGPU 算子的 type 更改为 MaskRCNN 并对应修改其余配置文件参数。由于 AI 推理算子在正式运行前需要初始化运行一次，所以我们需要在算子前额外添加一个 Trigger（type 为 InitTrigger）。

5）AI 推理算子会获得目标在 2D 图像中的位置区域（即掩码图，对应的是 obj_list 端口），之后我们需要将这些位置区域转换到 3D 点云中，这一环节对应的是 GetRobotTCP 组中的 ProjectMask 算子。对于 ProjectMask 算子，我们不仅需要给入 AI 推理算子获得的 obj_list,还需要给入 2D 图对应的点云、2D 图采图时所用的 rgb 镜头坐标系同点云坐标系的转换矩阵、相机 rgb 镜头的内参。这里由于我们已经将点云转换到了机器人坐标系，所以我们需要输入 rgb 镜头到机器人坐标系的转换矩阵。相机的 rgb 镜头内参可以直接从相机参数文件中读取。算子运行完成后，我们会获得所有检测目标的点云列表。

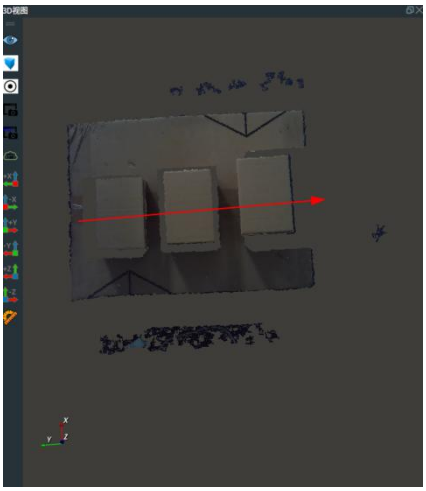


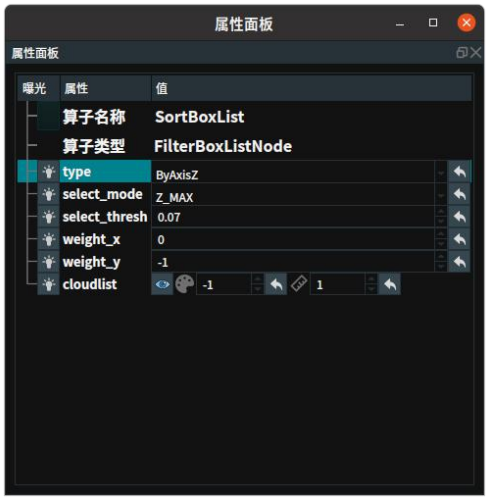
五、 识别定位

根据 AI 推理后的流程，我们已经获得了在机器人坐标系下所有检测目标的点云列表。接下来我们要获得他的点云中心坐标。

操作流程：

1）双击展开 GetRobotTCP 组中 GetObj 组。我们需要先筛选箱子，并按照箱子列表的 Z 轴坐标值进行筛选，筛选出最上层的箱子，并对上层箱子进行排序。因此这里使用 FilterBoxList 算子，重命名为“SortBoxLIst”，该算子的属性值调整如下：

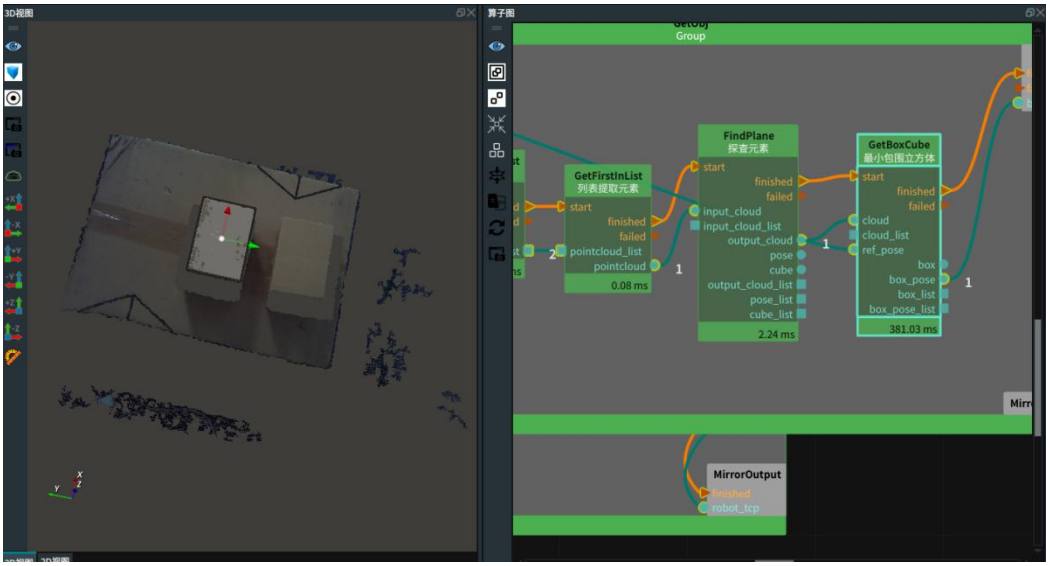




2）提取上层箱子点云列表中的第一个箱子的点云。使用 AtLitstNode,重命名为“GetFirstInList”；type 选择“PointCloud”，index 输入“0”。

3）获取平面，使用 FindElement，type 选择“Plane”，获得点云中适合抓取的平面。调整算子属性 distance_threshold 来调整所选取的平面。打开 cloud 可视化属性来查看选取的平面。

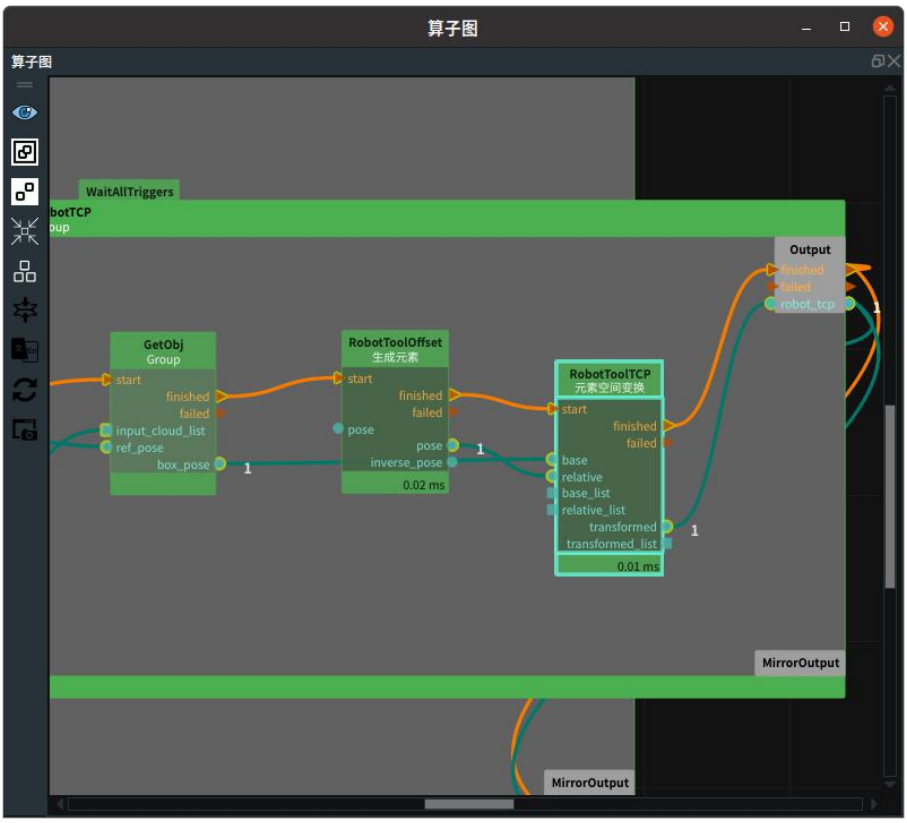
4）获取平面中心点，使用 MInimumBoundingBox 算子，重命名为“GetBoxCube”，type 属性选择“ApproxMVBB”获得一个方便机器人抓取的坐标中心点。这里需要给该算子一个 ref_pose，这里连接在第四章节第 2 小节中提到的“TowardsDownPose”，表示绕着 X 轴旋转 180°，使 Z 轴朝下，便于机器人抓取。打开“GetBoxCube”属性面板 box 和 box_pose 可视化属性即可显示计算出的平面中心点。



六、 机器人抓取

在完成上述操作后，我们已经获得了目标点坐标，我们需要通过加载机器人模型，模拟抓取。

1）首先我们需要测量机器人的吸盘工具的长度 length，由于吸盘在安装时同机器人的法兰盘是中心对齐的，则此时我们只需要沿着目标上表面中心姿态的 Z 轴向上调整 length，即可作为最终抓取时的机器人法兰盘位置。这里使用 Emit 算子，type 属性选择“Pose”，属性 z 输入“-0.192”。之后将调整好的姿态发送给机器人模型进行模拟移动，使用 Transform 算子，type 属性选择“Pose”。算子连接如下：



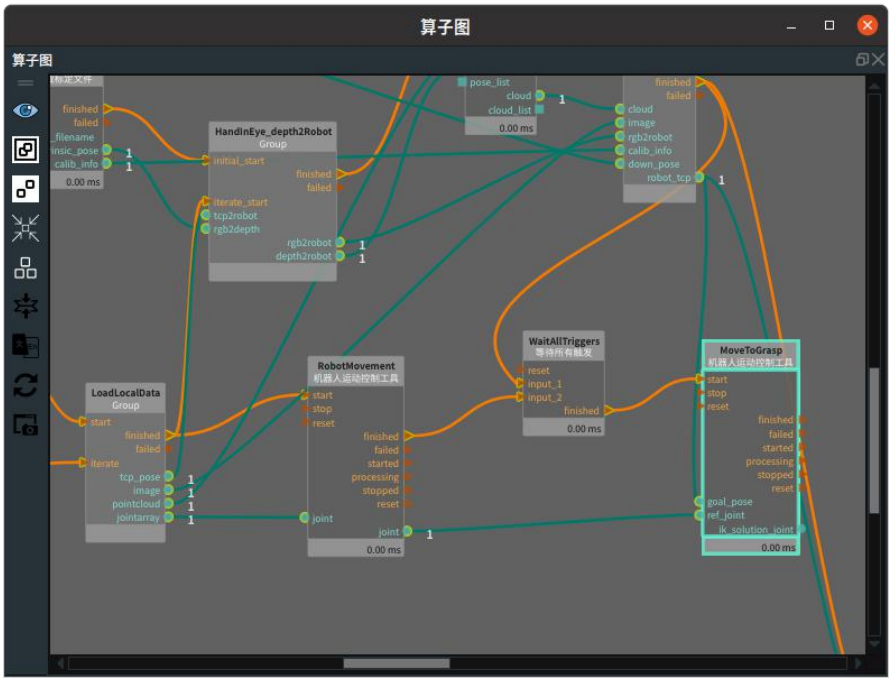
2）移动机器人。在 Resource 算子组中拖入 SimulatedRobotResource，设置属性 robot_filename 加载您的机器人文件，如 data 下的 EC66/EC66.rob。设置属性 tool_filename 加载机器人的吸盘工具，如 data 下的 Tool/EliteRobotSucker1.tool.xml。

3）在算子图中拖入 RobotMovement，type 属性选择“MoveJoint”，属性 robot_resource_name 属性选择”SimulateRobotResource”.设置 velocity、acceleration 来调整机器人的移动速度、加速度。左侧 Joint 端口与 LoadLocalData 组中的右侧 Jointarray 端口进行连接。start 端口与其 finished 端口连接。

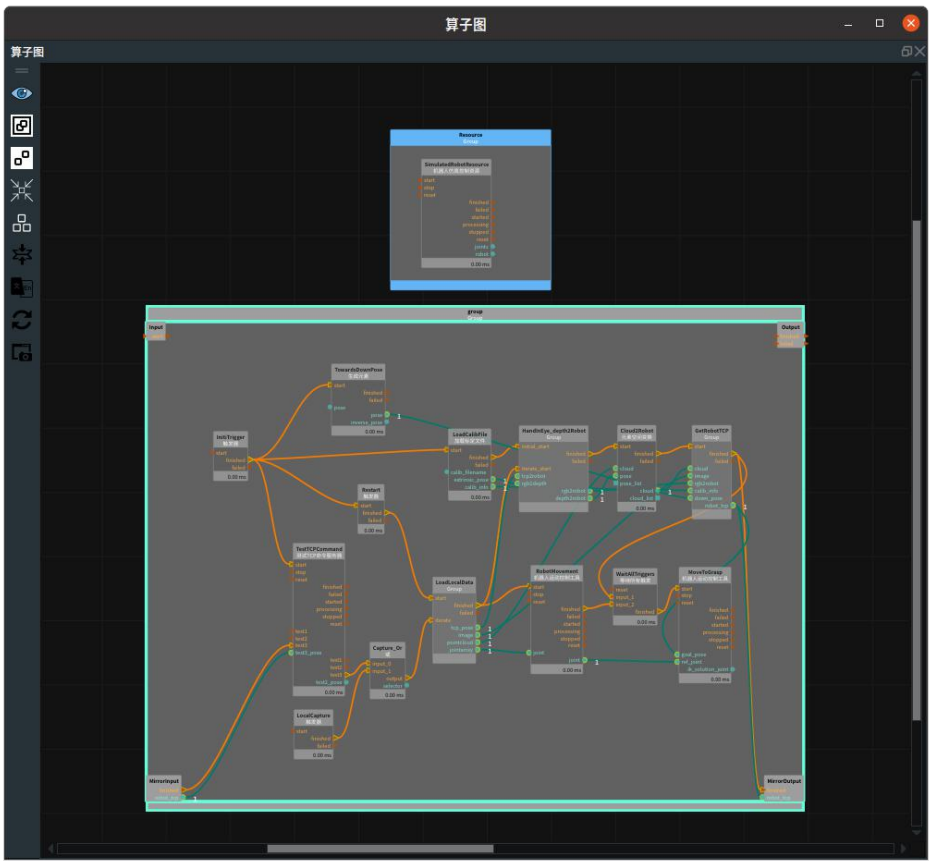
4）拖入 WaitAllTrigger 算子，将 GetRobotTCP 组的 finished 端口连接至其 input_1 端口。RobotMovement 算子的 finished 端口连接至其 input_2 端口。

5）在算子图中拖入 RobotMovement，type 属性选择“MoveTCP”，，重命名为“MoveToGrasp”属性 robot_resource_name 属性选择”SimulateRobotResource”。设置 velocity、acceleration 来调整机器人的移动速度、加速度。start 端口与 WaitAllTrigger 算子的 finished 端口进行连接。goal_pose 与 GetRobotTCP 组的 robot_tcp 端口连接，ref_pose 端口与 RobotMovement 的 joint 端口连接。

6）算子图具体连接如下：



7）最后，将 GetRobotTCP 的 finished 和 robot_tcp 端口连接到最外层算子组的 MirrorOutput 端口，并连接回 TestTCPCommand。至此，项目文件的编辑已经完成。



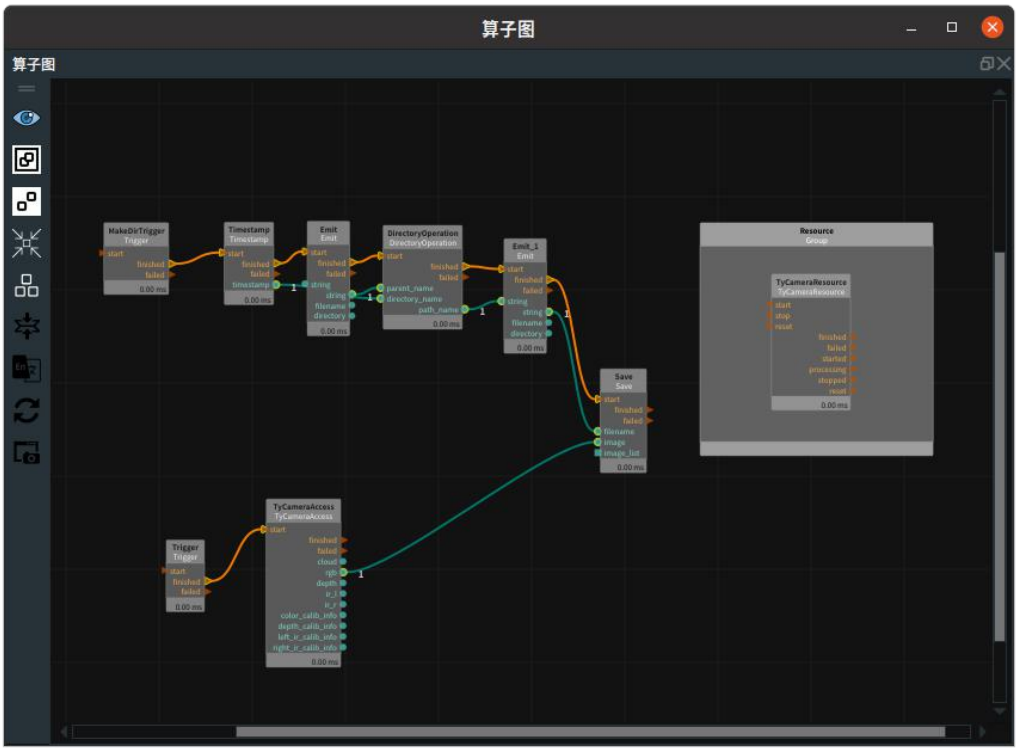
完整的 xml 已提供在 unstacking_runtime/unstacking.xml 路径下。

第四章 AI 训练

本章节主要目标为讲解采集训练图像、标注训练图像、训练 AI 模型。总的流程将分为采集、标注、训练。

一、 采集训练图像

打开 unstacking_runtime/MaskRCNN/ty_ai_savedata.xml，内容基本与录制 RGB 图像一致，在这里我们只需要调整 EmitString 中的 string 参数，设置为我们想要的路径即可。点击 Capture 录制图像。录制图像数量保证在 10 张以上，越多越好。



在图像采集时，我们应注意以下几点：

1) **背景光照**

(1) 单一稳定光源，亮暗适宜且没有过多反光

(2) 户外光照情况变化过大，不建议

2) **复杂工况考虑：尽量使得采样样本对实际运行的全局样本有足够的代表性而不仅仅是全局样本的一种特例**

(1) 目标物件多样化，尽量不选用固定单一目标物体。如果当前可供使用的样本数目比较少，可以考虑将样本正反两个面（两个面一样）都使用来倍增样本。

(2) 工况多样化，对于目标物体，尽可能多的考虑其在工况中出现的情形，多情况拍摄：

1：如果项目中垛车大小、垛车姿态、目标在垛车中的姿态、目标外形外观等有多种情形，则我们采集的训练数据应该把这些多样化的情形都囊括到。

2：要充分考虑训练样本同实际运行样本的偏差余量。比如垛车只有水平摆放一种姿态，我们在训练数据的时候也要考虑到给垛车增加小角度的倾斜，因为实际中不可能完美水平。再比如实际运行的都是机器摆放，目标两两比较贴近，在我们在录制训练集数据的时候，如果用的是人为摆放，则一定要保证目标两两比较贴近，而不是随意摆放导致目标两两之间间隙松散。

3：对于少部分的可能出现的目标破损褶皱等情况，虽然很少出现，但是如果项目实际有识别的需要，则要专门寻找较多的破损目标进行数据采集，虽然最后在训练样本中破损目标占的比例大于破损目标在实际样本中占有的比例，但是这也很有意义。

3) 图像质量

人眼需清晰可见目标边缘，尤其是距离相机最远的那层目标，否则考虑更换相机

具体图像录制可以参考 unstacking_runtime/MaskRCNN/train_data 中的 rgb 图像

二、 为训练图像进行标注

目前为已录制好的 RGB 标注 ,我们推荐使用 labelme 这款软件 ,本文档提供一种 labelme 的安装方法。

1) 安装 pip

sudo apt install python3-pip

2) 安装 pyqt5

sudo apt install python3-pyqt5

3) 在 labelme-4.5.7.tar.gz 路径下执行下述命令（网络自行查询）

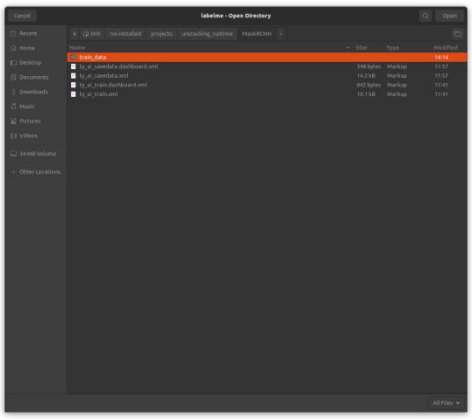
python3 -m pip install labelme-4.5.7.tar.gz -i https://pypi.tuna.tsinghua.edu.cn/simple

4) 安装完成，运行

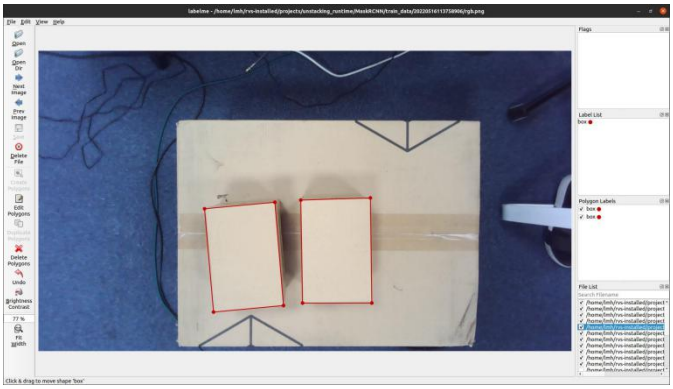
python3 -m labelme

操作流程：

1) 打开 labelme 软件，点击 OpenDir，选择我们标注的路径

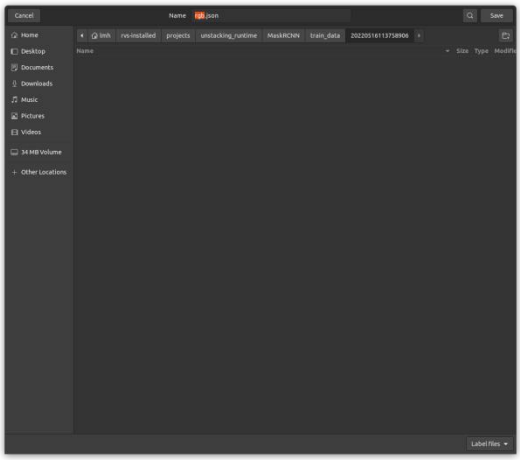


2) 点击 Create Polygons，为箱子绘制红色的边框



3) 完成后会弹出命名框，第一次请命名 box，后续同类直接选择即可

4) 当图像内所有箱子标注完成后，点击 Save 进行保存，默认当前文件夹，默认名称即可，随后选择 Next Image 切换到下一个图像



在标注过程中我们应当注意以下几点：

标注前准备

(1) 首先确定任务目标，明确在检测过程中什么物体需要被检测，什么物体不需要被检测，从而有针对性的进行标注

(2) 给定的标注条件无需过分苛刻，不要按照人的思维去考虑，而是按照自己主观设定的标注思路是否便于落实代码

标注过程

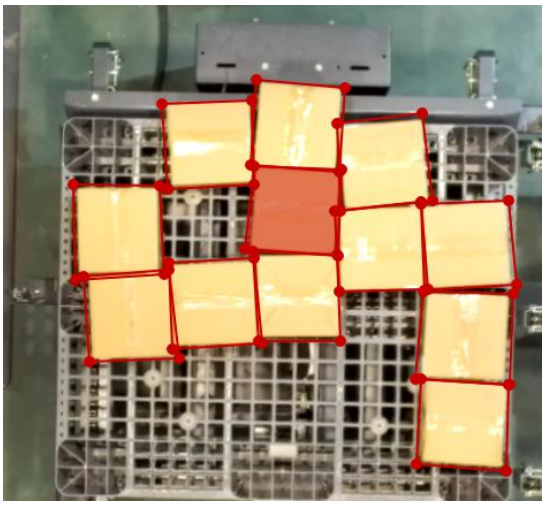
(3) 标记物体可见区域的边界框（不是估计的物体总范围）

(4) 标注画框结束，注意做边界检查，比如确保框坐标不在图像边界上

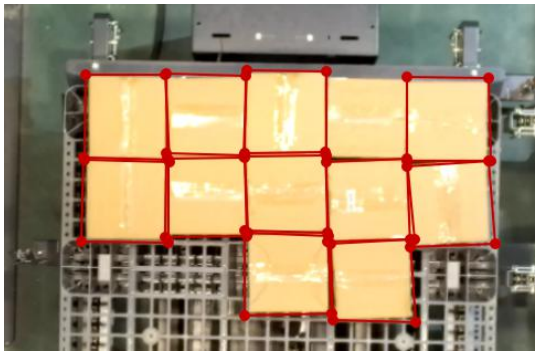
(5) 质量较差的图像（例如过度运动模糊），部分遮挡的目标，结合项目目标需求选择标注或舍弃

我们应尽量避免以下问题：

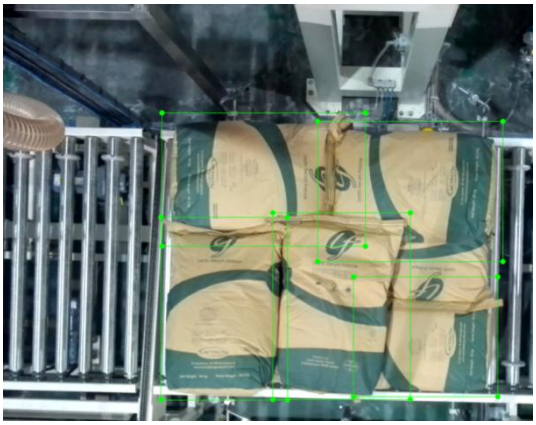
- 边界标注不准



- 遗漏



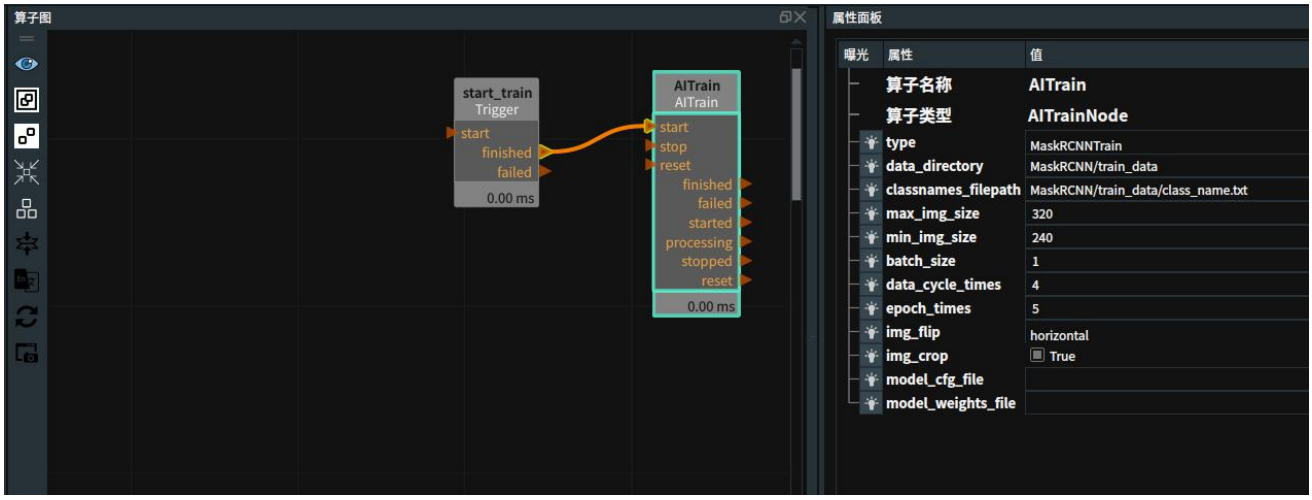
- 被遮挡部分不应该标注



三、 训练 AI 模型

打开 unstacking_runtime/MaskRCNN/ty_ai_train.xml ，这里我们只需要调整 data_directory 和 classnames_filepath 路径。点击 start_train 即开始训练。

最终，我们会生成一个 train_output 文件夹，在这个文件夹中有命名为 model_final.pth（默认可更改），是我们所需要的权重文件。



至此您已经完成了本文档的所有内容，感谢您的耐心查阅，相信您一定对于 RVS 软件已有所了解，可以自主创造属于您的独特项目，若在使用中出现问题，请及时与我们反馈，发送邮件 rvs-support@percipio.xyz！

附录 A 支持资源

本章描述图漾为用户提供的支持资源，包括相关的技术与文档支持.

A.1. 技术支持

我们对所有购买产品的用户提供以下两种官方技术支持方式：

1. 公众号支持：请关注官方微信公众号-图漾科技，并发表您的看法和意见
2. 邮件支持：如需其他帮助，请发邮件至 rvs-support@percipio.xyz。

A.2. 文档支持

您可能需要了解以下文档内容：

- 图漾产品选型指南

  本文档介绍图漾所有产品型号及其技术指标。

- Percipio SDK 入门指南

  本文档介绍 SDK 的安装、编译和使用。

A.3. 各版本下载链接

我们对不同版本提供了以下下载链接：

- Full 版本下载链接（包含全部功能模块）
- CPU 版本下载链接（不带 GPU 相关功能）

> 说明:
>
> 如需下载其他文档，请访问如下链接：downloadcenter — 图漾科技 | 3D 相机（percipio.xyz）。