RobotVisionSuite

模板匹配案例

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）进行简单的模板匹配。文档结构如下:

章节	标题	内容
第一章	RVS 视觉平台—模板匹配	包含 RVS 介绍，模板匹配案例背景，以及离线文件的使用。
第二章	模板点云生成	使用 RVS 将模型文件生成模板点云并进行点云预处理及保存。
第三章	模板匹配	使用 RVS 进行模板匹配。
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
2022.11	V1.0	第一次发布。
2023.02	V2.0	RVS 发布新的版本，文档内 XML 重新连接。

免责和版权声明

本手册为图漾产品的使用说明，其受版权保护，未经图漾事先书面同意，任何人不得以任何形式复制、修改本手册的内容。图漾对任何人使用被篡改过产品使用说明所造成的损失或伤害，不承担任何责任。本文档未以禁止反言或其他方式授予任何知识产权的许可，无论是明示的还是暗示的。

在现行法律许可的情况下：（1）本使用说明仅基于产品目前的现状，对产品将来是否适销、品质是否良好、是否侵犯他人产品的权益、是否适用等问题不做任何形式的声明与保证；（2）在将来任何情况下，对使用本手册所造成的任何损失和伤害（包括但不限于直接损失、间接损失、特别损失、附随损失、间接损失或惩罚性赔偿），图漾将不承担责任，即使这些损失和损害是可以预见的，或图漾曾被告知将有可能造成这些损失。

这个文档本身可能包含印刷错误和产品技术说明方面的错误。图漾有权在不通知用户的情况下，对产品的使用说明做更改。客户在购买产品的时候，须向当地经销商索取最新的产品使用说明。

图漾保证本产品符合注明的质量标准，并在质保期内承担产品的质量保责任。但本产品只能用作指定用途，将产品挪作它用而造成的损失，图漾不承担任何责任。

版权 ©2022 图漾科技。保留所有权利。

目录

第一章 RVS 视觉平台-模板匹配

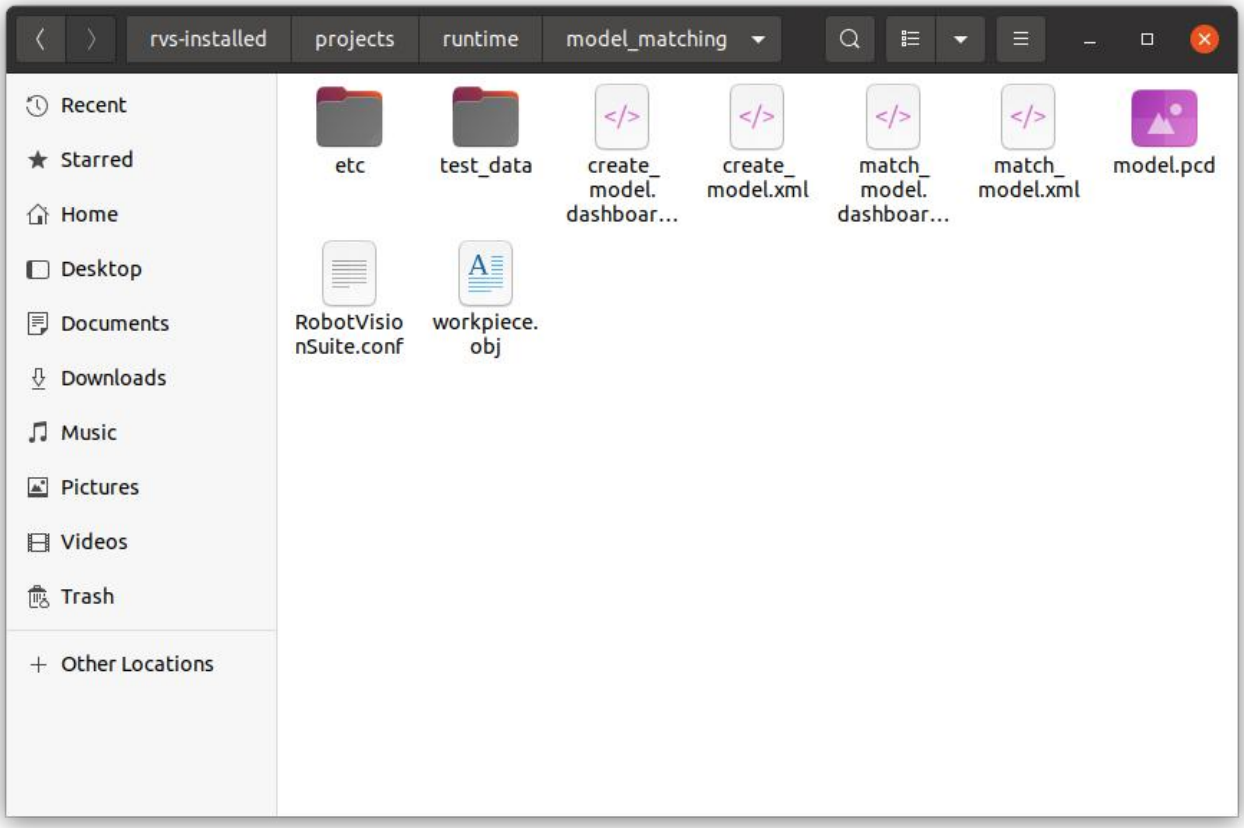
RobotVisionSuite（以下简称 RVS）是图漾科技有限公司（以下简称图漾）自行研发的一款 3D 机器视觉开发平台。RVS 中可以将 3D 视觉算子写成节点（Node）以供后续开发使用，各个算子节点可以在软件平台中相互连接以及嵌套成组，形成完整的项目程序流程。可以利用已有的算子快速搭建项目、验证项目可行性、开发出原型程序以供客户测试。为了方便用户的快速入手，图漾目前已经储备了大量常用节点，包括图漾 3D 相机的 SDK 调用节点，常用的 2D&3D 图形图像处理节点，常用工业机器人接口节点，视觉标定与手眼标定节点，通讯节点，点云配准节点（ICP）、图像深度学习训练与推理节点等等。节点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

模板匹配案例背景：

- 功能要求：通过已有模型文件生成模型点云，在作业场景中与目标点云进行匹配。
- 作业场景：模型匹配是 3D 手眼协作项目中的重要前置环节。通过模型匹配，我们可以将提前设计好的基于模型的位姿转换为实际目标处的位姿，进而达到视觉引导机械手的作业目的，具体的包含无序抓取、喷绘、打标、切割、焊接等应用。
- 项目要求：1.生成模型文件的点云；
  2.计算模型点云与目标点云的最优变换，使得模型点云与目标点云重合程度尽可能高。

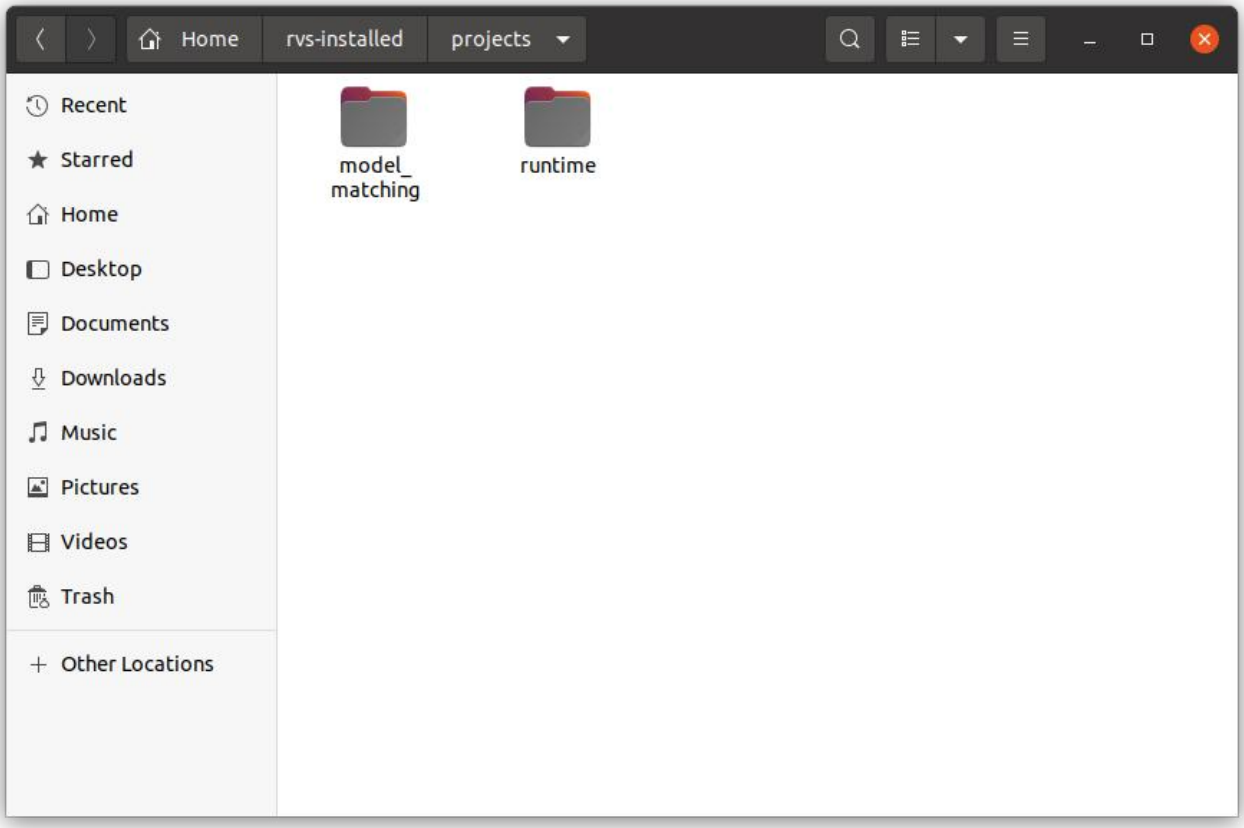
当您获得本文档时，您也应该同时获得 model_matching.zip 文件。加载离线本地文件，可以获得完整软件操作体验。解压后其内容应包括：



其中本案例中需要用到的有：

- create_model.xml：将模型文件生成模板点云并进行点云预处理及保存。
- model.pcd：该文件为 create_model.xml 运行后保存的点云文件。
- match_model.xml：用于离线模板匹配，使用时需要更新 ReadDirectoryNames 算子中离线数据的路径和 LoadModelCloud 算子中的模板点云文件。使用时先点击交互面板中 start 按钮启动工程后，再点击 iterate 按钮迭代 test_data 中的离线数据。
- test_data：用于模板匹配数据的加载，使用时请注意路径正确。
- workpiece.obj：用于 create_model.xml 中 LoadPolydata 算子加载的 3D 模型文件。（RVS 支持的模型文件格式有：*.ply *.stl *.obj *.gltf *.glb；如果您当前拥有的模型文件格式不符，需要提前完成转换。）

案例所有使用到的文件均保存在 model_matching 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 模板点云生成

一、 生成模板点云流程

在模板匹配流程中，需要用到模板点云。在拥有 3D 模型文件的前提下，我们需要根据 3D 模型生成模板点云，并对点云进行预处理、调整并保存。本章节我们主要分为以下 4 个过程：

- 网格采样：根据给定的 3D 模型创建点云。
- 点云预处理：划分点云中合适的工作区域。
- 调整点云：调整模板点云物体面与坐标系。
- 保存点云：保存模板点云。

二、 网格采样

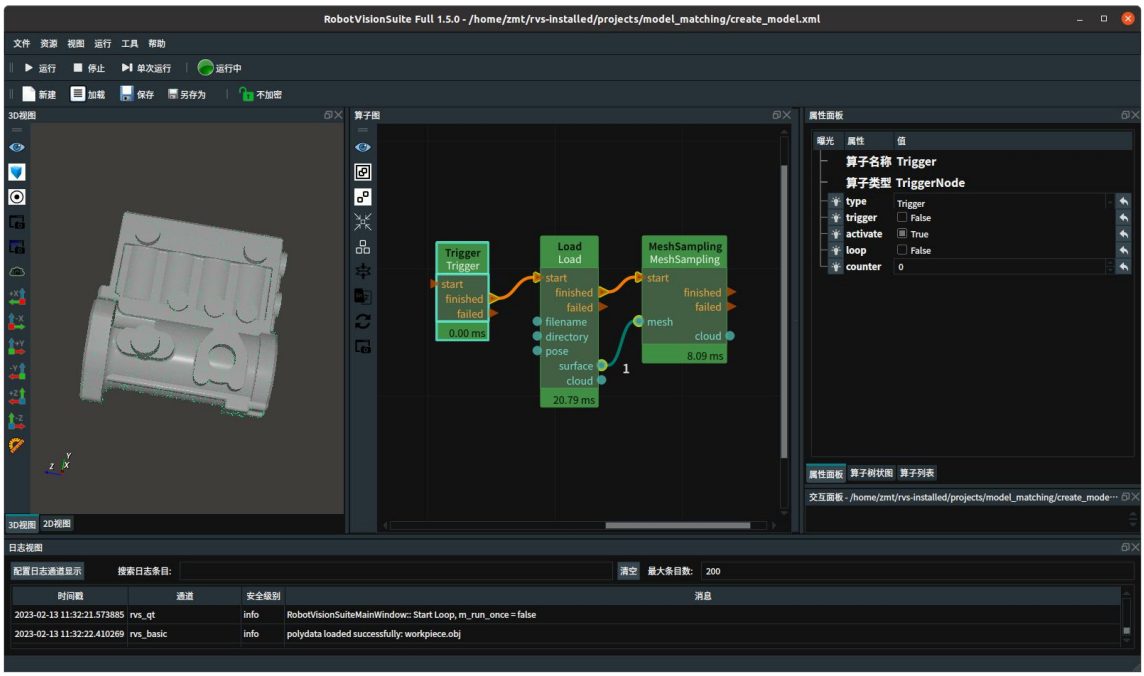
加载 3D 模型文件，使用 MeshSampling 算子将其转化为点云。

操作流程：

1）新建工程项目 create_model.xml;在算子图中添加一个 Load 算子，type 属性选择“PolyData”，用于加载 3D 模型，将其 filename 属性选择“workpiece.obj”文件名（如果您是直接加载已有的 create_model.xml，需要注意路径）；将该算子的 surface_visibility 属性勾选为 True，可以在 3D 视图中看到该模型。

2）添加 MeshSampling 算子，连接 LoadPolyDate 算子的 finished 至 MeshSampling 算子的 start 端口，LoadPolyDate 算子 surface 端口连接至 MeshSampling 算子的 mesh 端口；将该算子的 cloud_visibility 属性勾选为 True，触发后，在 3D 视图中看到点云。

3）下图中算子图为算子连接；3D 视图中，显示原始颜色的为模型，绿色为转化后的点云。



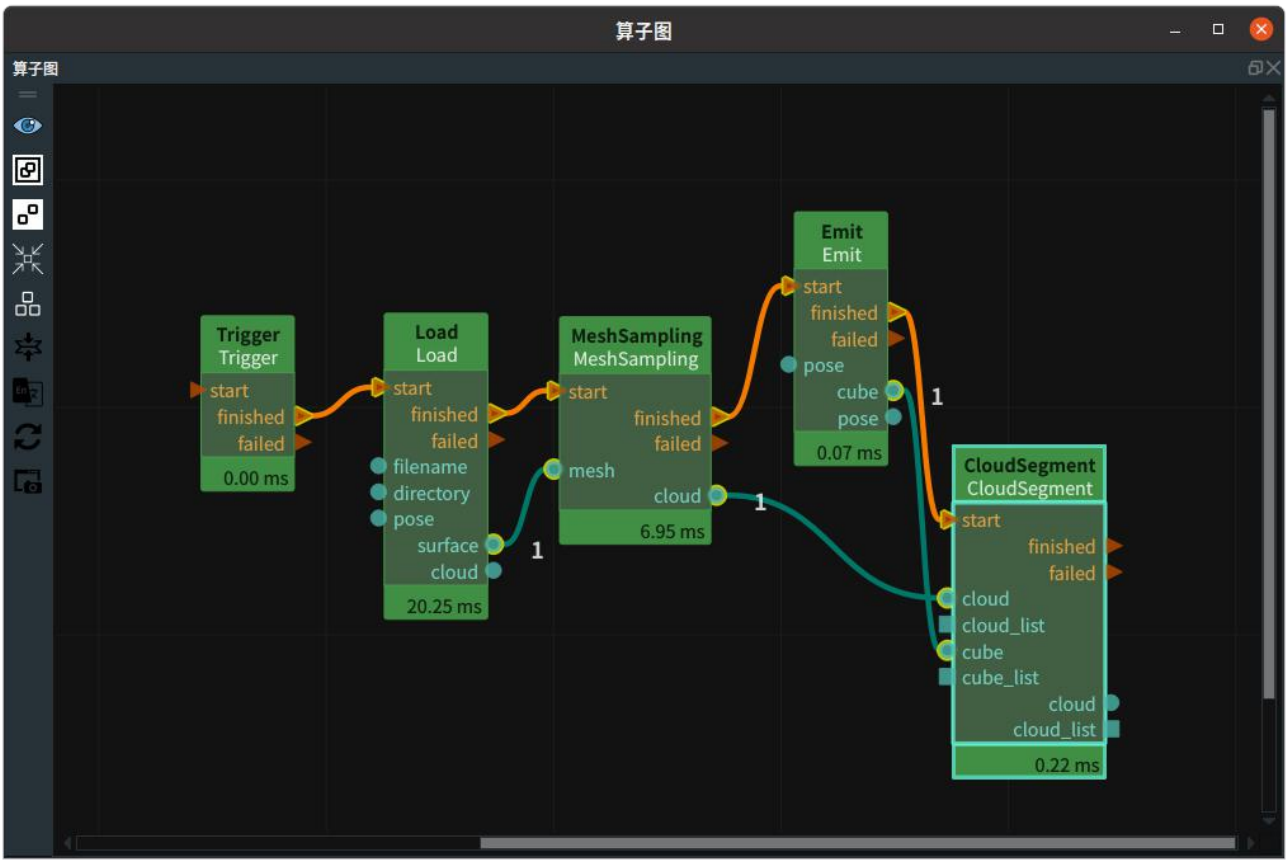
三、 点云预处理

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：点云预处理。进行模板匹配时，只需要用到模型点云的上表面；因此，我们需要对点云进行切割。

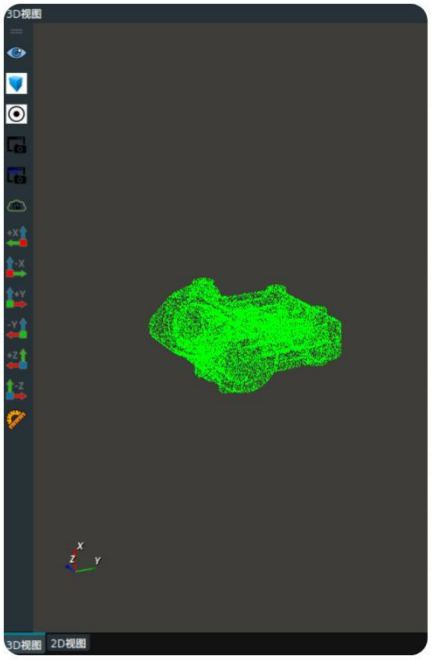
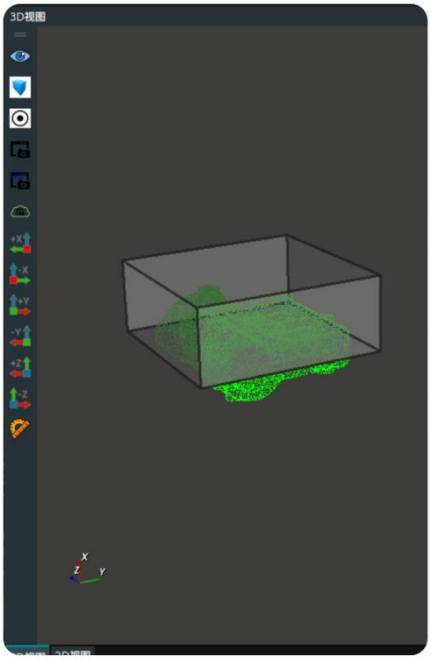
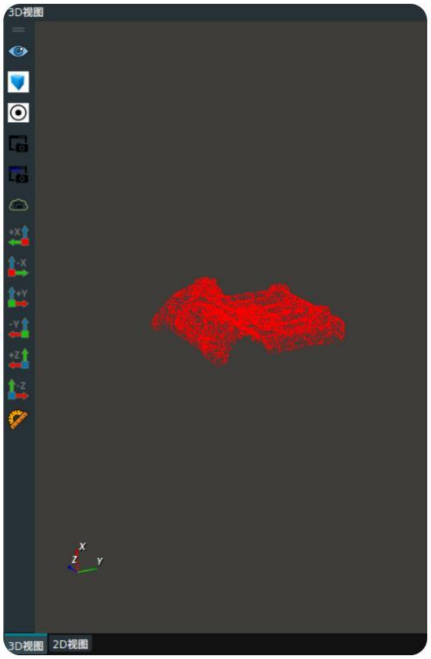
操作流程：

1）在算子图中添加 Emit 算子，type 属性选择“Cube”，其作用是在 3D 视图中生成一个立方体。将 MeshSampling 算子的 finished 端口连接至 Emit 算子的 start 端口；可以修改 Emit 算子的属性 width、height、depth 来改变立方体的长宽高，打开 cube 的可视化属性，可在 3D 区查看其在点云图中的具体位置和大小，需要注意的是，每次修改立方体大小都要重新触发来查看；移动 Cube 包裹住您所需要的点云区域，Emit 将会自动更新对应 pose。

2）添加 CloudSegment 算子，type 属性选择“CropboxSegmen”，该算子可以用给定的立方体切割出所需要需要的点云，连接 Emit 的 finished 端口至本算子的 start 端口，cube 端口连接至本算子同名输入端口；将 MeshSampling 算子的 cloud 端口连接至 CloudSegment 算子的 cloud 端口；打开 cloud 可视化属性；触发后，可以在 3D 视图中看到切割后的点云。



3）下图中，分别展示了点云切割前后 3D 视图中显示的点云：

原始点云　　生成立方体　　切割后点云

四、 调整点云

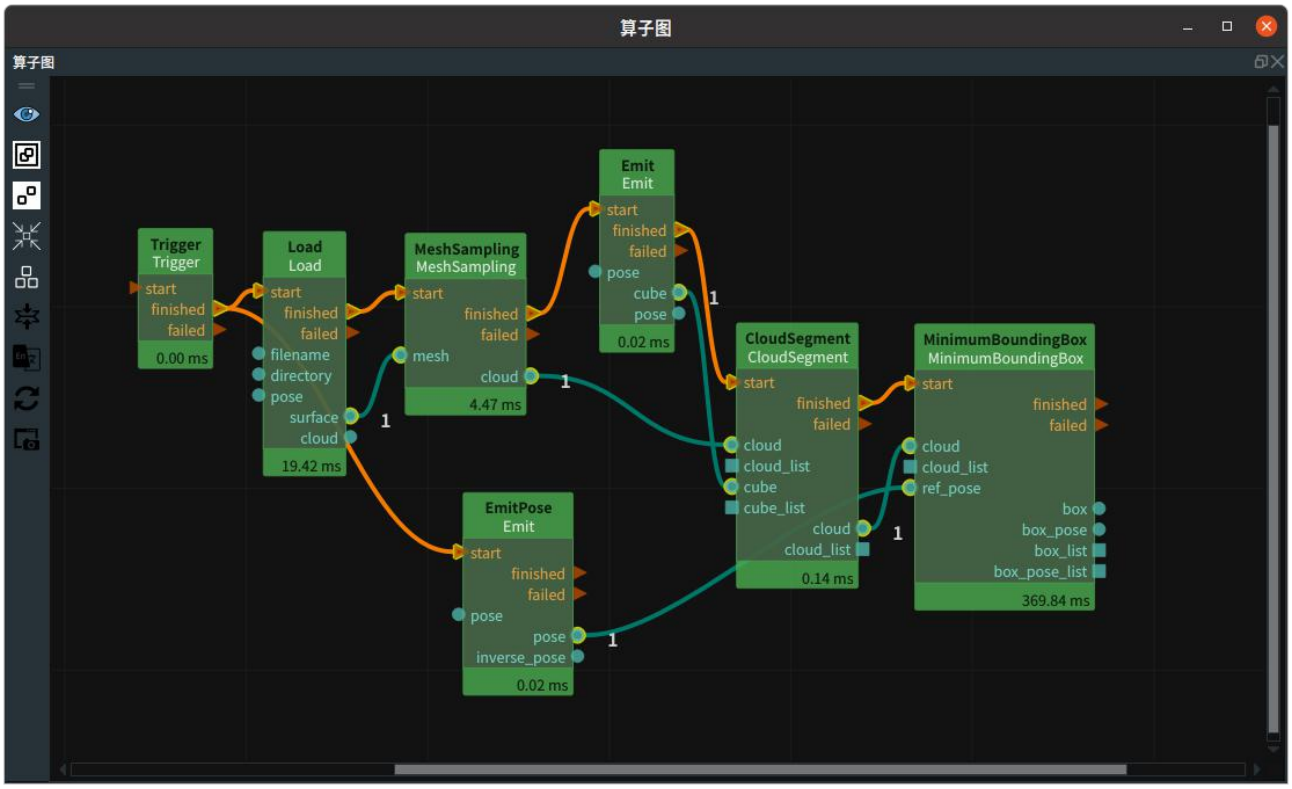
获得了剪裁好的模板点云后，为了方便后文模板匹配算子（ICP 算子）的调用，需要将当前模板姿态的 0xy 平面与实际来料物体姿态的 0xy 平面保持平行，同时需要将模板点云的中心移动到 RVS 中的 3D 世界坐标系（坐标系方向即为 RVS 中的 3D 视图窗口左下角对应的三个轴）的原点。

我们默认目标来料时的 0xy 平面平行于机器人坐标系下的 0xy 平面，又由于机器人坐标系重合于 RVS 中的 3D 世界坐标系，则我们需要将模板点云进行空间变换，使得其姿态的 0xy 平面平行于 RVS 的 3D 世界坐标系的 0xy 平面。

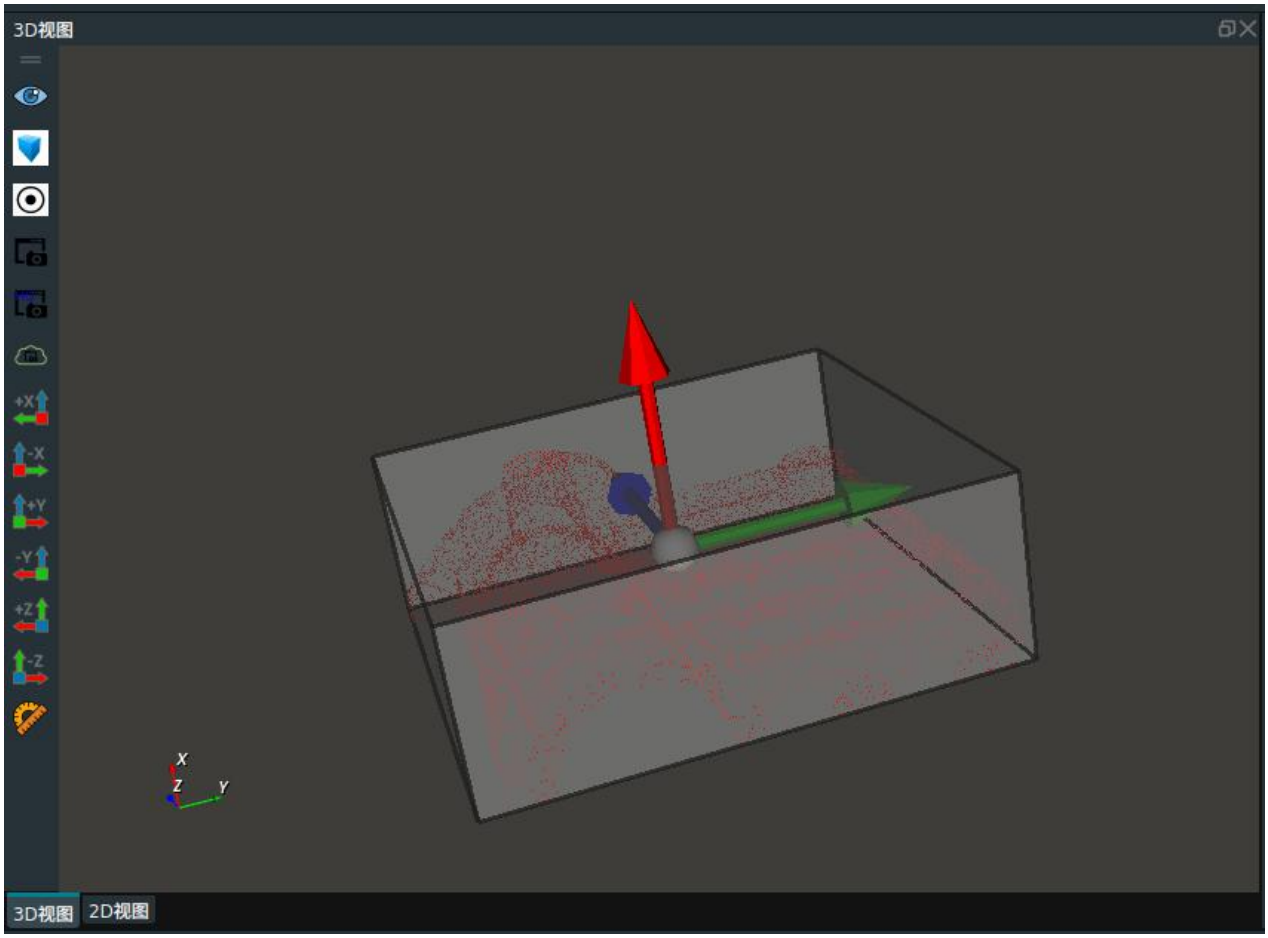
操作流程：

1）获取点云中心点，拖入 MinimumBoundingBox 算子，type 属性选择“ApproxMVBB”，连接前置 CloudSegment 算子的 finished 至本算子的 start 端口，同时连接 cloud 至本算子 cloud 输入端口。

2）拖入 Emit 算子，type 属性选择“Pose”，并且重命名为 EmitPOse，生成 Pose(0,0,0,0,0,0)，连接 MinimunBoundingBox 算子左侧的 ref_pose 端口，这里目的是将输入的 pose 姿态进行 Hight-Width-Ddpth 同 XYZ 的匹配。

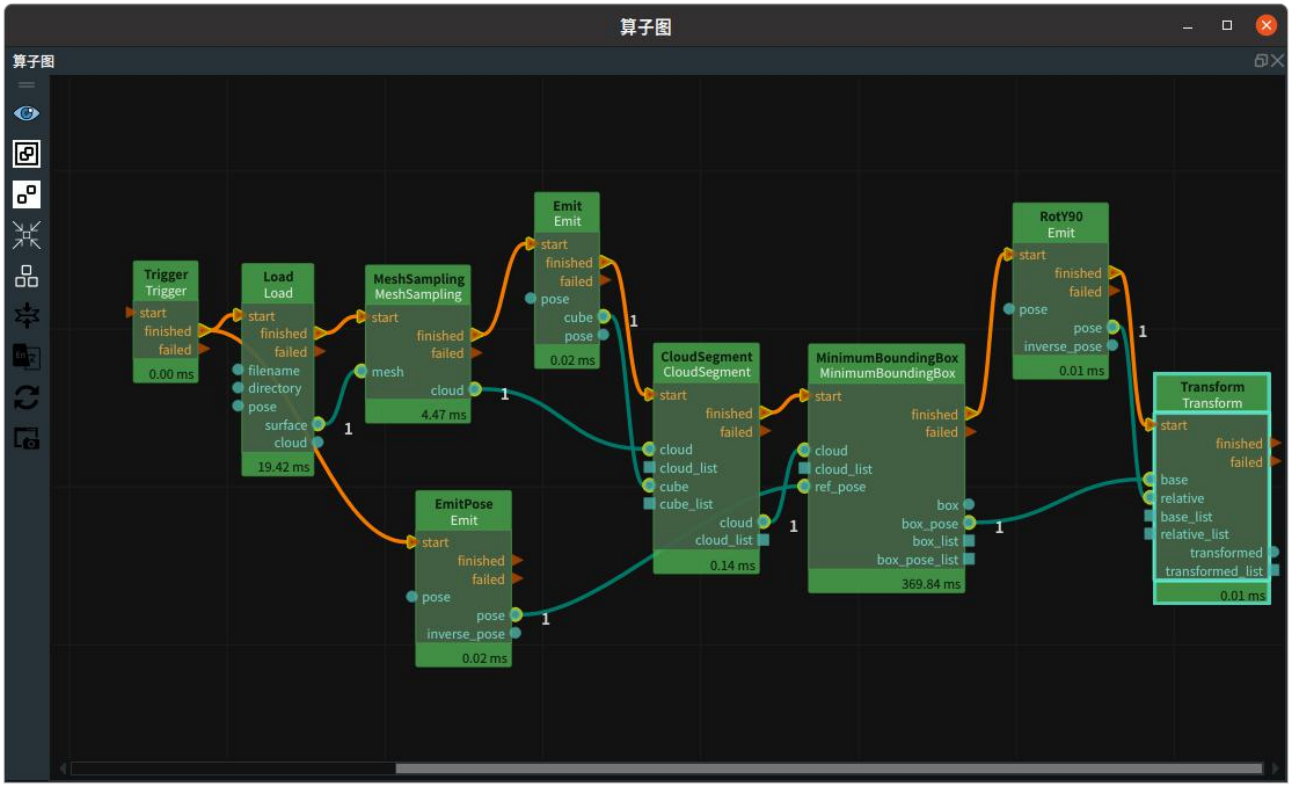


3）打开 MinimunBoundingBox 算子的 box 和 box_pose 可视化属性，即可在 3D 视图中显示计算出的最小包围立方体以及该立方体的中心点位姿。

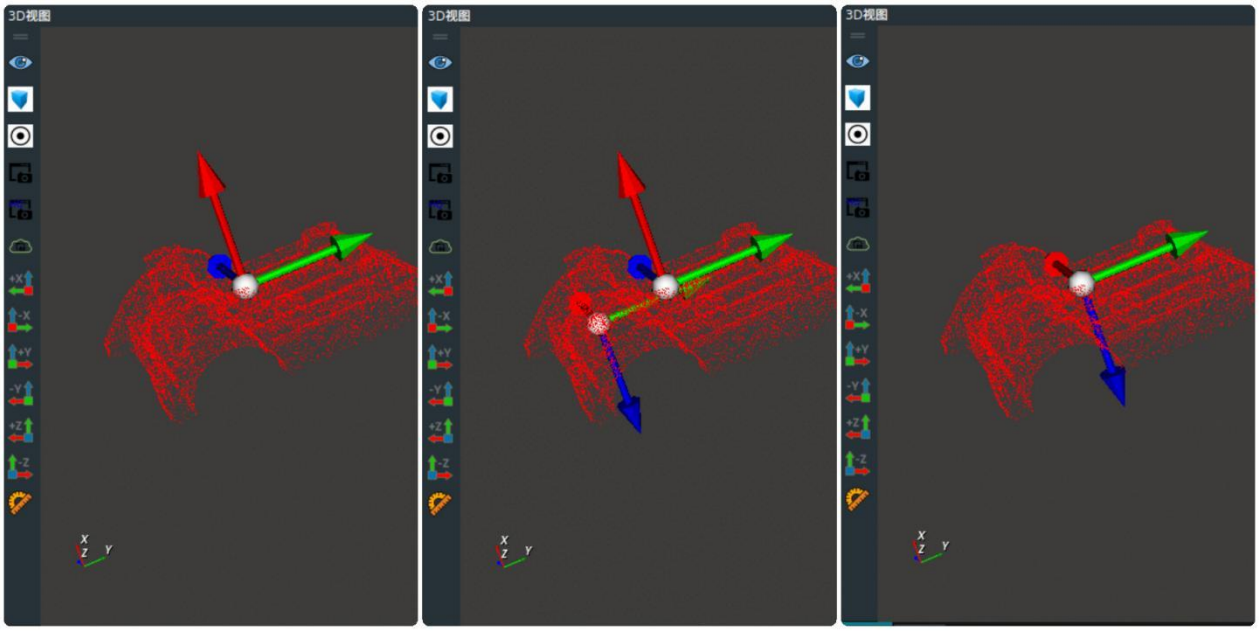


4）上图 Pose 的中心点可以作为模板点云的中心点，但是该 Pose 的 0xy 平面并未与 RVS 3D 坐标系

的 Oxy 平面平行，所以需要将当前的 pose 绕着其 Y 轴旋转-90°。拖入 Emit 算子，type 属性选择“Pose”，重命名为“RotY90”，将该算子的 pose_pitch 属性修改为“-1.5708”，将 MinimunBoundingBox 算子的 finished 端口连接至“RotY90”算子的 start 端口，再拖入 Transform 算子，type 属性选择“Pose”，将“RotY90”算子 finished 端口连接至 TransformPose 算子的 start 端口，将“RotY90”算子的 pose 端口连接至 TransformPose 算子的 relative 端口；将 ApproxMVBB 算子的 obb_base 端口接入 TransformPose 算子的 base 端口。



5）下图中，展示了模板姿态 Oxy 转换的过程：

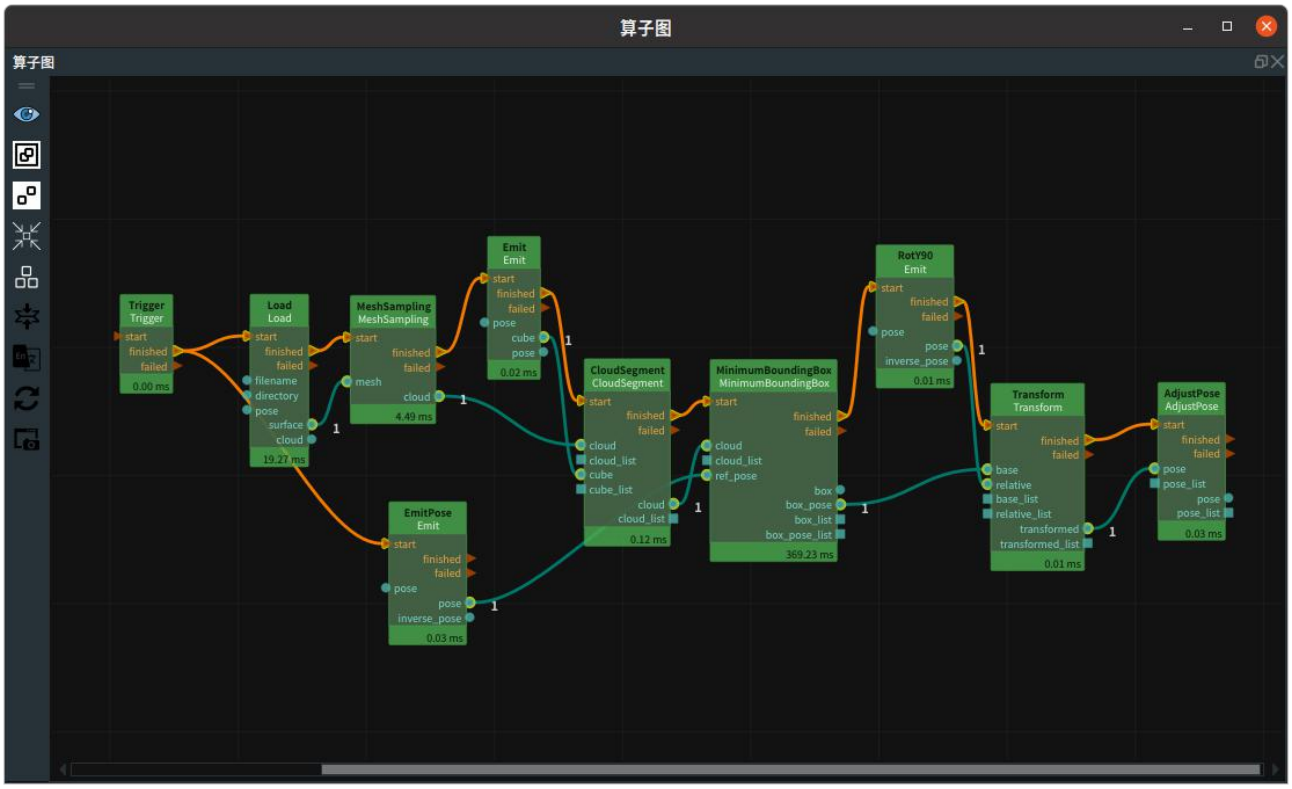


原始 pose　　生成的 pose　　旋转后 pose

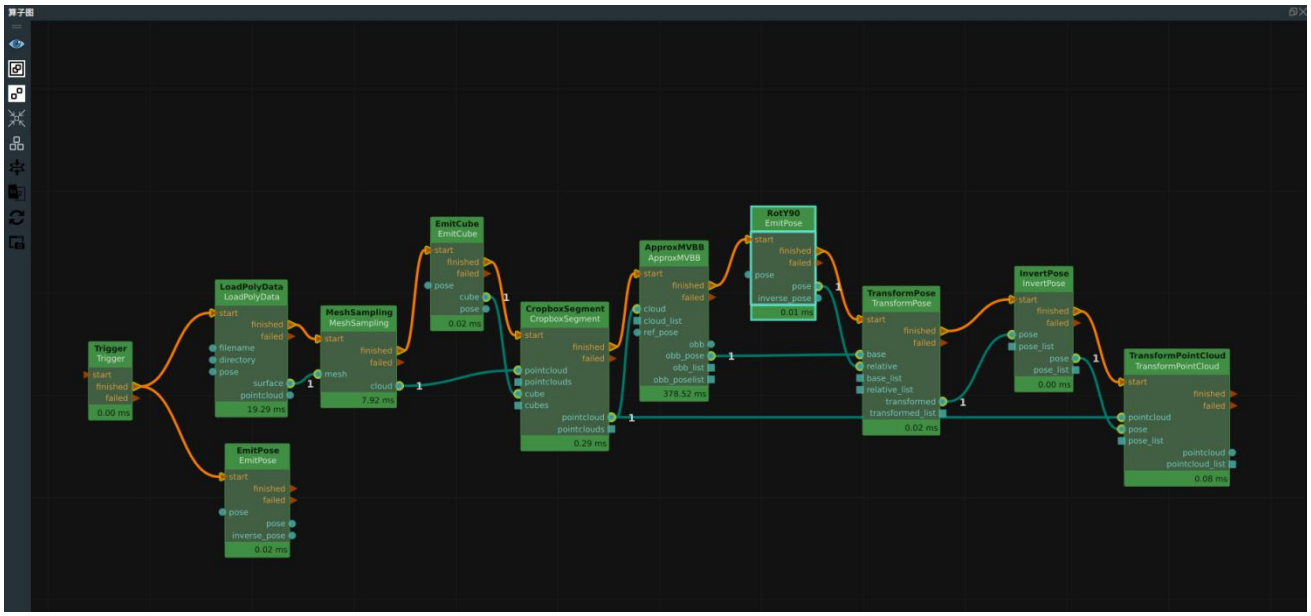
6）我们已经将当前模板姿态的 Oxy 平面与实际来料物体姿态的 Oxy 平面保持平行。之后需要将模板

点云的中心移动到 RVS 中的 3D 世界坐标系的原点。拖入 Adjust 算子，type 属性选择“InvertPose”，连接如下图所示。其作用是对原 Pose 取逆。

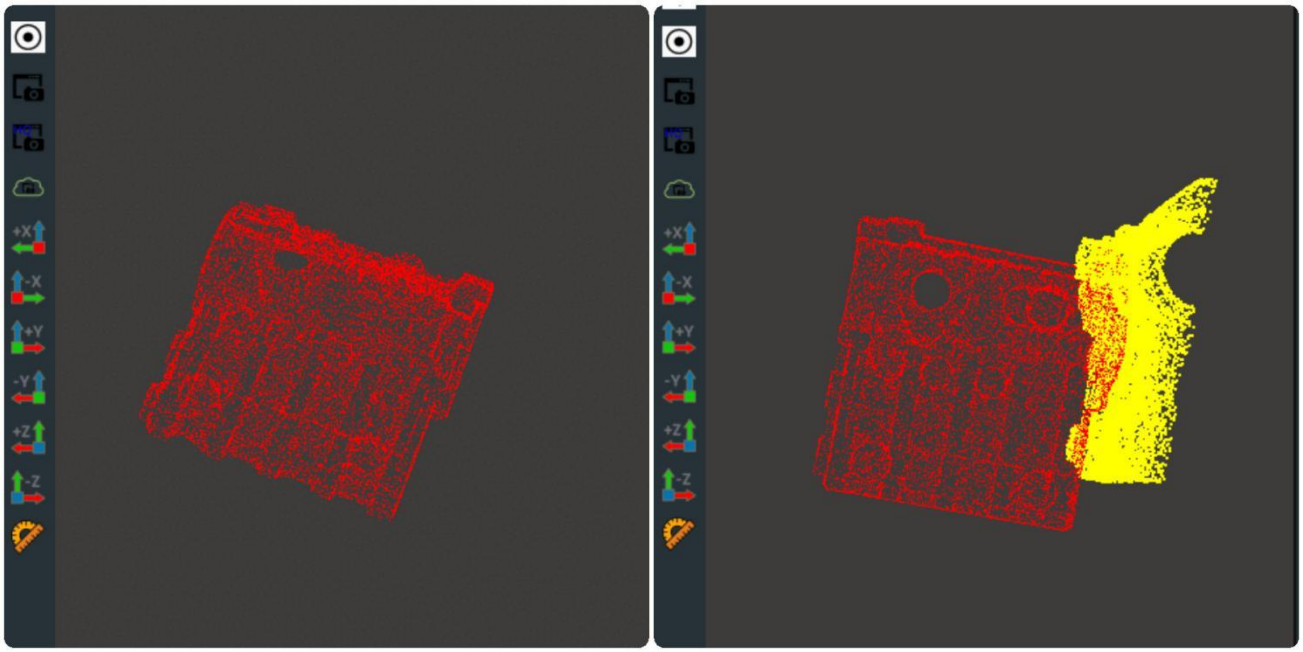
7）为了更直观的参考原始坐标，可以将 EmitPose 的可视化属性打开，生成的就是基于 RVS 3D 世界坐标原点 pose（0，0，0，0，0，0）。连接如下图所示：



8）之后，基于取逆后的 Pose 对裁剪后的模板点云进行空间变换，可以使得变换后的新点云的中心点位于坐标系原点，同时使得新点云的在原 pose 姿态下的 x/y/z 轴分别转换为了 RVS 3D 坐标系的 x/y/z 轴。原 pose 的算子图中拖入 Transform 算子，type 属性选择“PointCloud”，并重命名为“TransformPointCloud”，将 AdjustPose 算子的 finished 端口与本算子的 start 端口连接，AdjustPose 算子的 pose 端口与 TransformPointCloud 算子的 pose 端口连接；CloudSegment 算子的 cloud 端口与 TransformPointCloud 算子的 cloud 端口连接；连接如下图所示：



9）将 TransformpointCloud 算子的 pointCloud_visibility 属性勾选为 True 后，3D 视图中黄色为转换后的点云。



点云转换前　　点云转换后

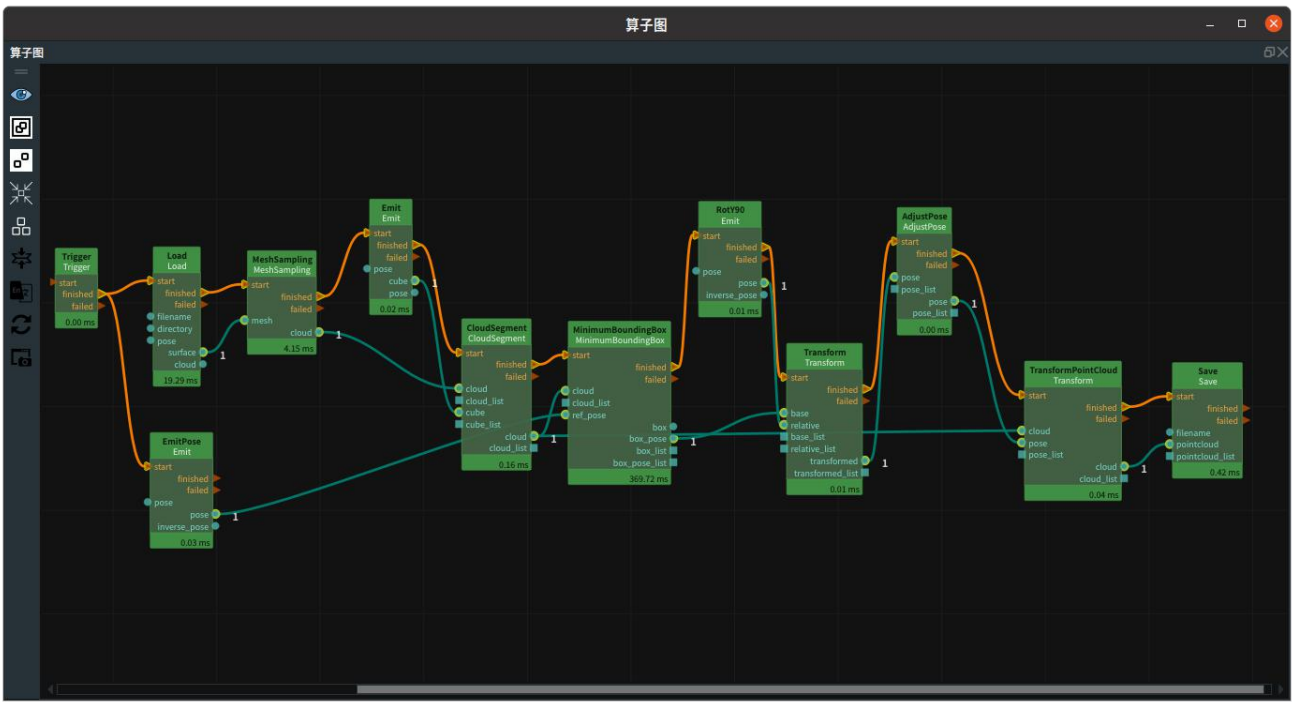
五、 保存点云

根据调整点云的流程后，我们获得了在原始坐标系下的模板点云，接下来我们要保存该模板点云，以便我们在模板匹配中使用。

操作流程：

1）保存点云，拖入 Save 算子，type 属性选择“PointCloud”，将该算子的 filename 属性输入点云文件名，例如：“model.pcd”。如下图：

2）将 TransformPointCloud 算子的 finished 端口连接至 SavePointCloud 算子的 start 端口；将 TransformPointCloud 算子的 cloud 端口连接至 SavePointCloud 算子的 pointcloud 输入端口。至此，生成模板的 create_model.xml 已经编辑完成。如下图：



3）运行成功后，可以在 runtime 目录下看到保存的“model.pcd”。

至此 create_model.xml 已经全部完成;完整的 xml 已提供在 model_matching/create_model.xml 路径下。

第三章 模板匹配

在创建好模板点云后，我们可以进行模板匹配工作。本案例中主要依赖于 ICP 算子进行模板匹配，ICP 算子需要准备模板点云（source_cloud）、目标点云（target_cloud）、两个点云之间的转换关系的初始值 initial_pose。 我们需要确保 initial_pose 基本准确，然后 ICP 算子就是在这个 initial_pose 的基础上进一步的微调从而获得最终的 result_pose，使得模板点云根据 result_pose 进行空间变换以后能够和真实点云达到重合效果。

因此本章节主要分为以下 5 步：

- 模板点云准备：加载模板点云；对点云进行稀疏化处理，提高算子执行效率。
- 目标点云准备：本案例有多组离线测试数据，用 ForeachString 算子遍历读取目标点云名称。
- 点云预处理：去除一定范围之外的点，去除地面；对点云进行稀疏化处理，提高算子执行效率。
- 初值推测值准备：给定多组初始推测值，得到最优解。
- 模板匹配：ICP 算子获得 result_pose，将模板点云根据该 pose 进行空间变换后与目标点云重合。

一、 模板点云准备

加载模板点云；对点云进行稀疏化处理，提高算子执行效率。

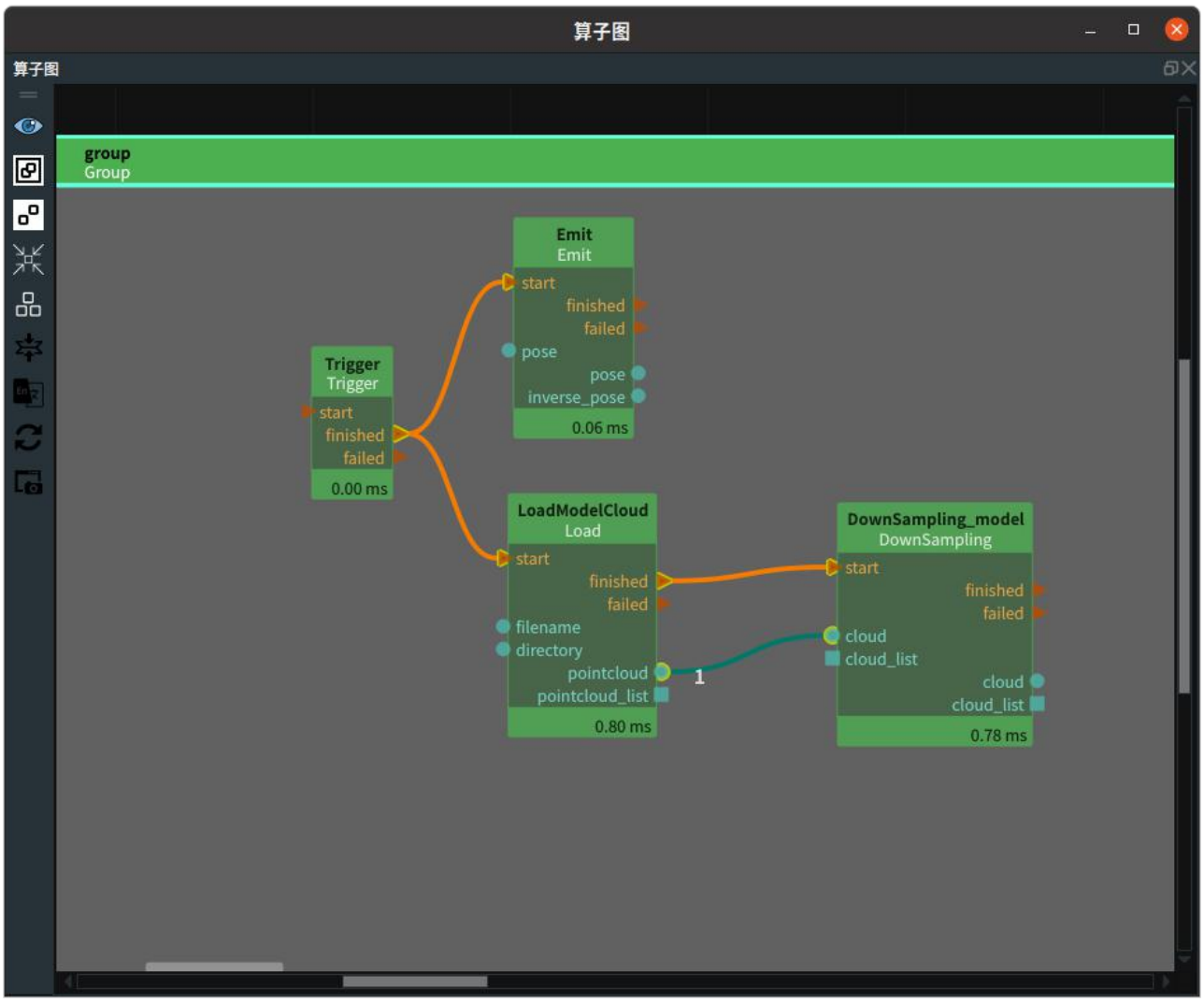
操作流程：

1）新建工程项目 match_model.xml;拖入 Load 算子，type 属性选择“PointCloud”，将其重命名为“LoadModelCloud”，将其 filename 属性选择保存的 model.pcd 文件路径——“model.pcd”。

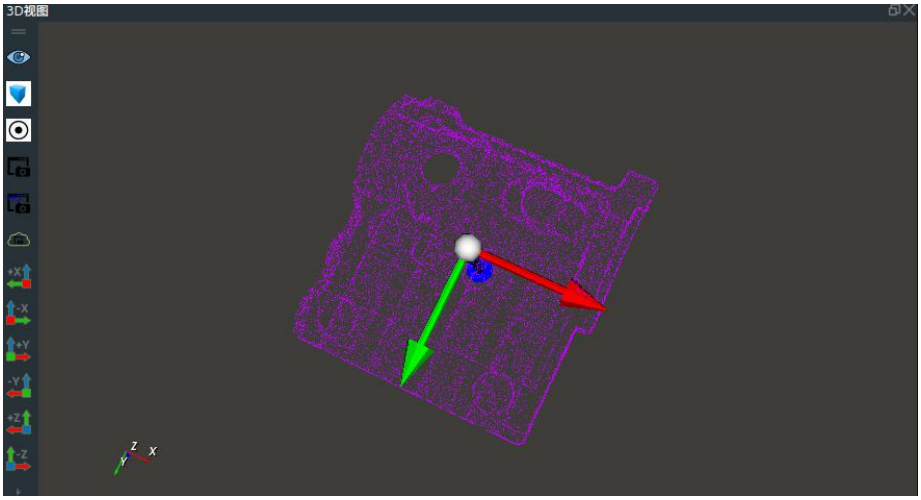
2）生成坐标原点。拖入 Emit 算子，type 属性选择“Pose”，生成基于 RVS 3D 世界坐标原点 Pose(0,0,0,0,0,0)。

3）拖入 Trigger 算子，type 属性选择“InitTrigger”，将其 finished 输出端口连接至 LoadModelCloud 算子及 Emit 算子的右侧 start 输入端口，该步操作可以在运行 xml 时自动触发 LoadModelCloud 算子和 Emit 算子。

4）将模板点云降采样，提高算子执行效率。拖入 DownSampling 算子重命名为“DownSampling_model”，将 LoadModelCloud 算子的 finished 端口连接至该算子的 start 端口；右侧 pointcloud 输出端口连接至该算子的左侧 cloud 输入端口。将 DownSampling 算子的 leaf_x、leaf_y、leaf_z 指定 xyz 轴方向点云重采样间距，此处设置为 0.002m，表示在 0.002m*0.002m*0.002m 的空间尺度内仅取一个点。算子图连接如下图所示：



5）将 DownSamling_model 算子的 cloud 可视化属性打开；将 Emit 算子的 pose 可视化属性打开，在 3D 视图中看到降采样后的模板点云和生成的坐标原点 pose。



二、 目标点云准备

加载测试数据，本案例有多组离线数据，因此需要使用 ForeachString 算子进行遍历加载目标点云文件。

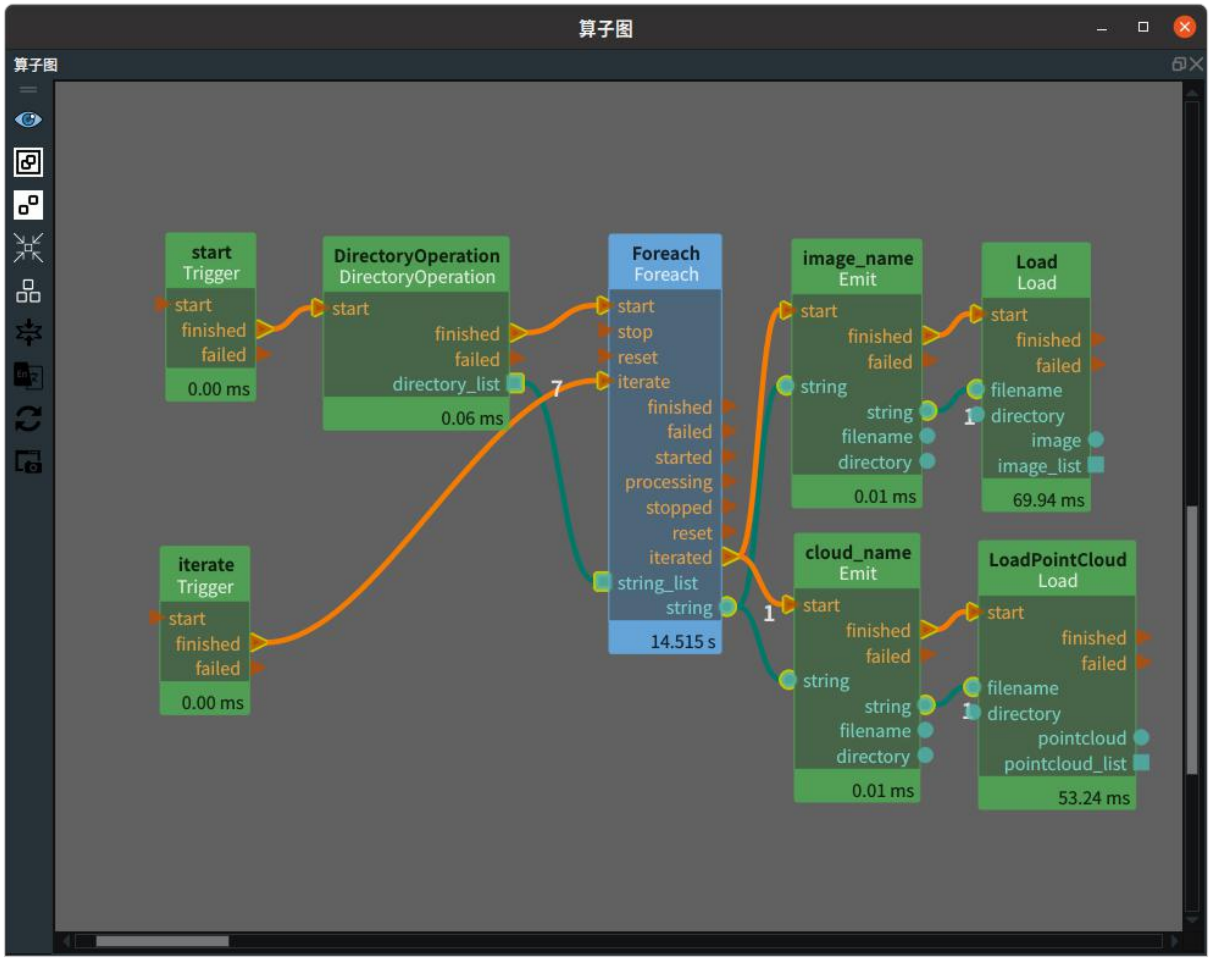
操作流程：

1）加载离线数据文件夹。在算子图中拖入一个 Trigger 算子重命名为”start”并将 trigger 属性曝光；拖入 DirectoryOperation 算子，type 属性选择“ReadDirectory”，将其 parent_directory 属性选择“test_data”（如果您是直接加载已有的 match_model.xml，需要注意路径）。将 start 算子的 finished 端口连接至 DirectoryOperation 算子的 start 端口。

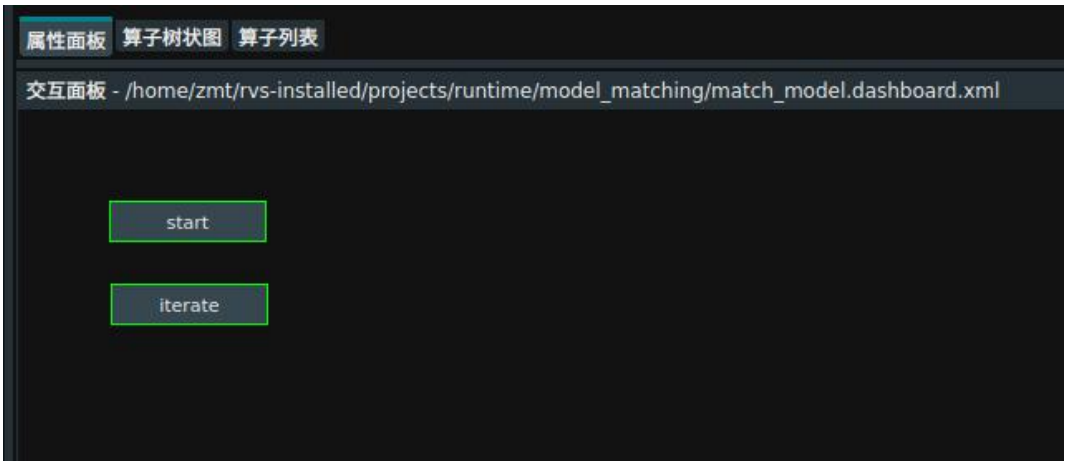
2）遍历 test_data 文件夹下多组测试数据。拖入 Foreach ，type 属性选择 String“” ，将 DirectoryOperation 算子的 finished 端口连接至 Foreach 的 start 端口，directory_list 端口连接至 string_list 端口。拖入 Trigger 算子重命名为”iterate”并将 trigger 属性曝光。将其 finished 端口连接至 Foreach 算子的 iterate 端口。

3）加载遍历文件夹中图像。拖入 Emit 算子，type 属性选择“String”，重命名为“image_name”，将其 string 属性输入“/rgb.png”；连接 Foreach 算子的 iterated 端口至 Emit 算子的 start 端口，Foreach 算子的 string 端口连接至 image_name 算子左侧同名输入端口。拖入 Load 算子，type 属性选择“Image”，重新命名为“LoadImage”，将 image_name 算子的 finished 连接至 LoadImage 算子的 start 端口，image_name 算子 string 端口连接至 LoadImage 算子的 filename 端口。

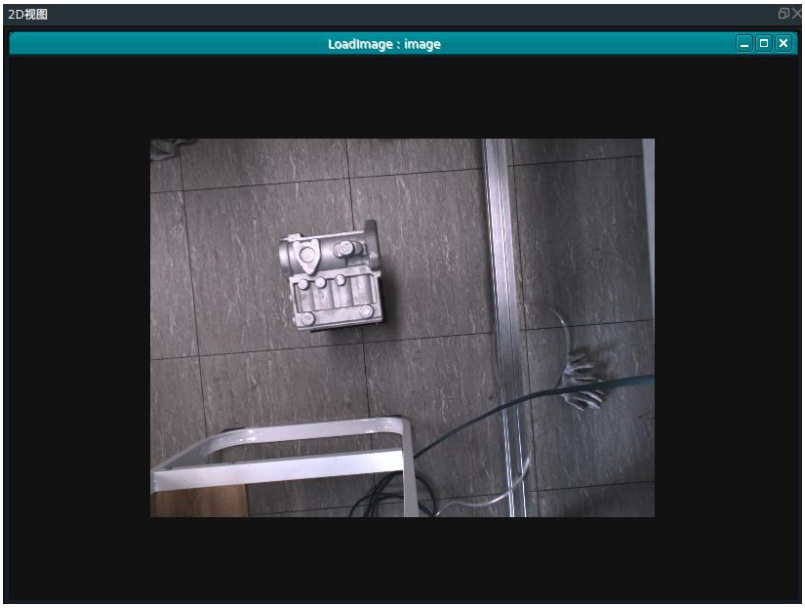
4）加载遍历文件夹中点云。拖入 Emit 算子重命名为“cloud_name”，将其 string 属性输入“/cloud.pcd”；连接 Foreach 算子的 iterated 端口至 cloud_name 算子的 start 端口，Foreach 算子的 string 端口连接至 cloud_name 算子的 string 端口。拖入 Load 算子，type 属性选择“PointCloud”，重新命名为“LoadPointCloud”，将 cloud_name 算子的 finished 连接至该算子的 start 端口，cloud_name 算子的 string 端口连接至该算子的 filename 端口；如下图所示。



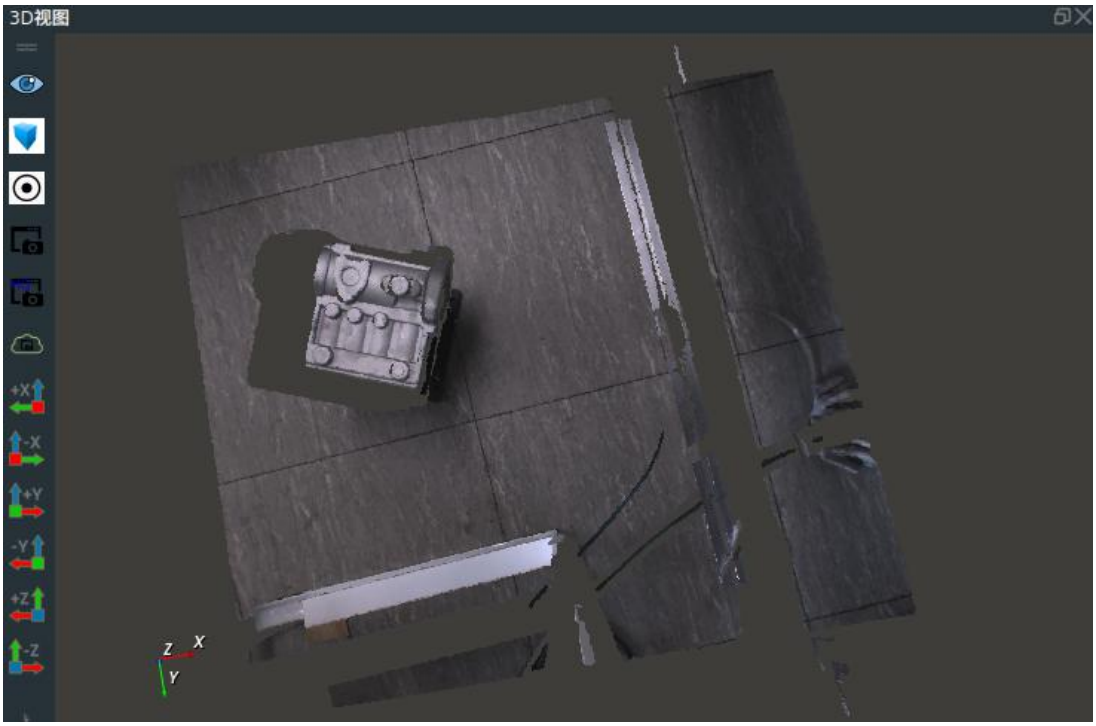
5）交互面板设置触发按钮。我们在上述的步骤中，曝光了两个 Trigger 的属性，此时，可以在交互面板中拖入两个“按钮”，分别取名为“start”和“iterate”，鼠标滚轮键点击按钮，选择同名的曝光属性。右击锁定。当点击“start”按钮时，触发后置算子；当点击“iterate”按钮时，遍历 test_data 文件夹。



6）将 LoadImage 算子的 image_visibility 属性勾选为 True，在 2D 视图中看到加载的图像。



7）将 LoadPointCloud 算子的 pointcloud_visibility 属性勾选为 True，在 3D 视图中看到加载的点云。



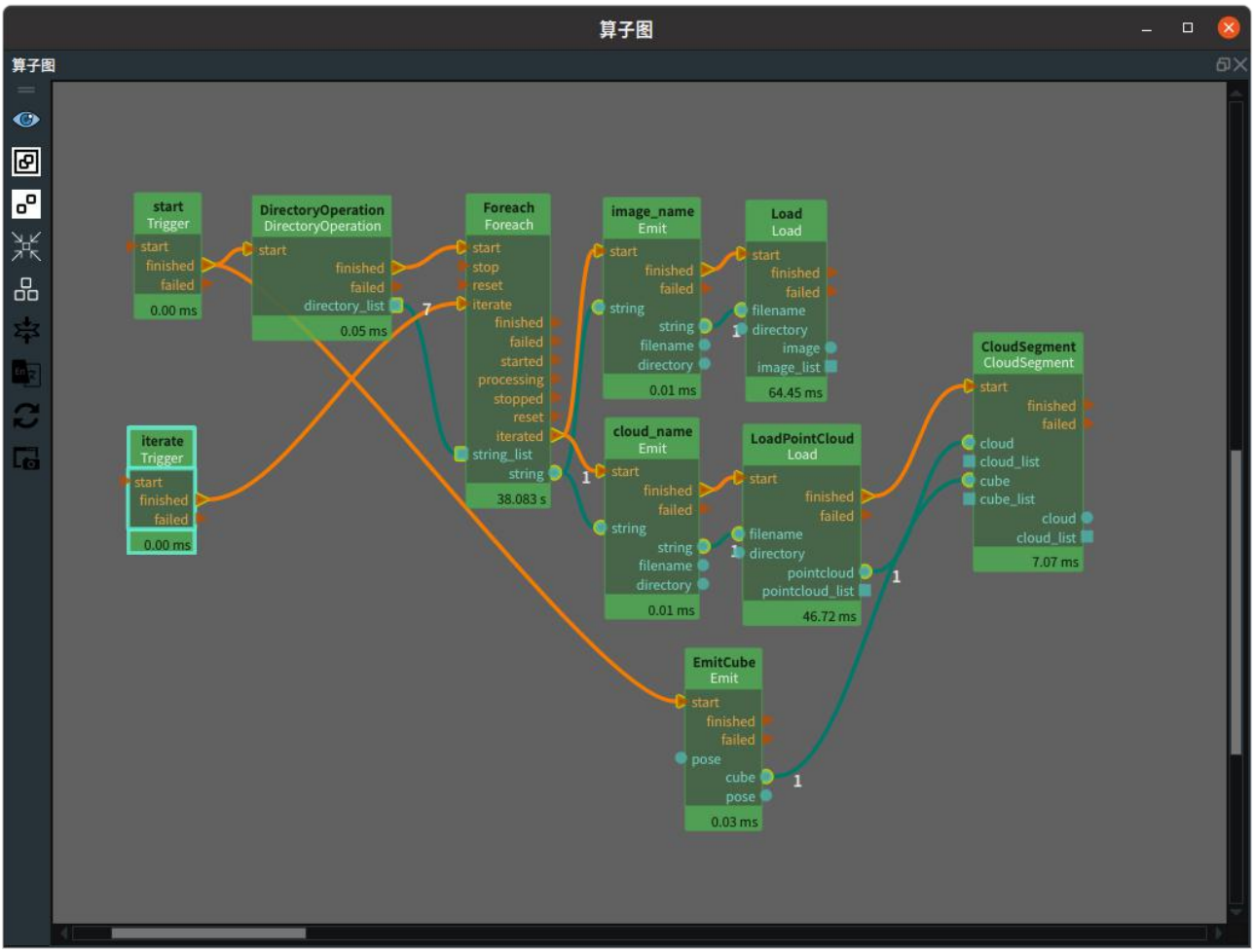
三、 点云预处理

加载测试点云数据后，需要剔除背景点云，筛选出工作区域；对点云进行稀疏化处理，提高算子执行效率。

操作流程：

1）针对背景点云，我们可以人为筛选出工作区域。首先在算子图中添加 Emit 算子，type 属性选择“Cube”，将“start”算子的 finished 端口连接至本算子的 start 端口，这一步的目的是在 3D 区域生成一个立方体。

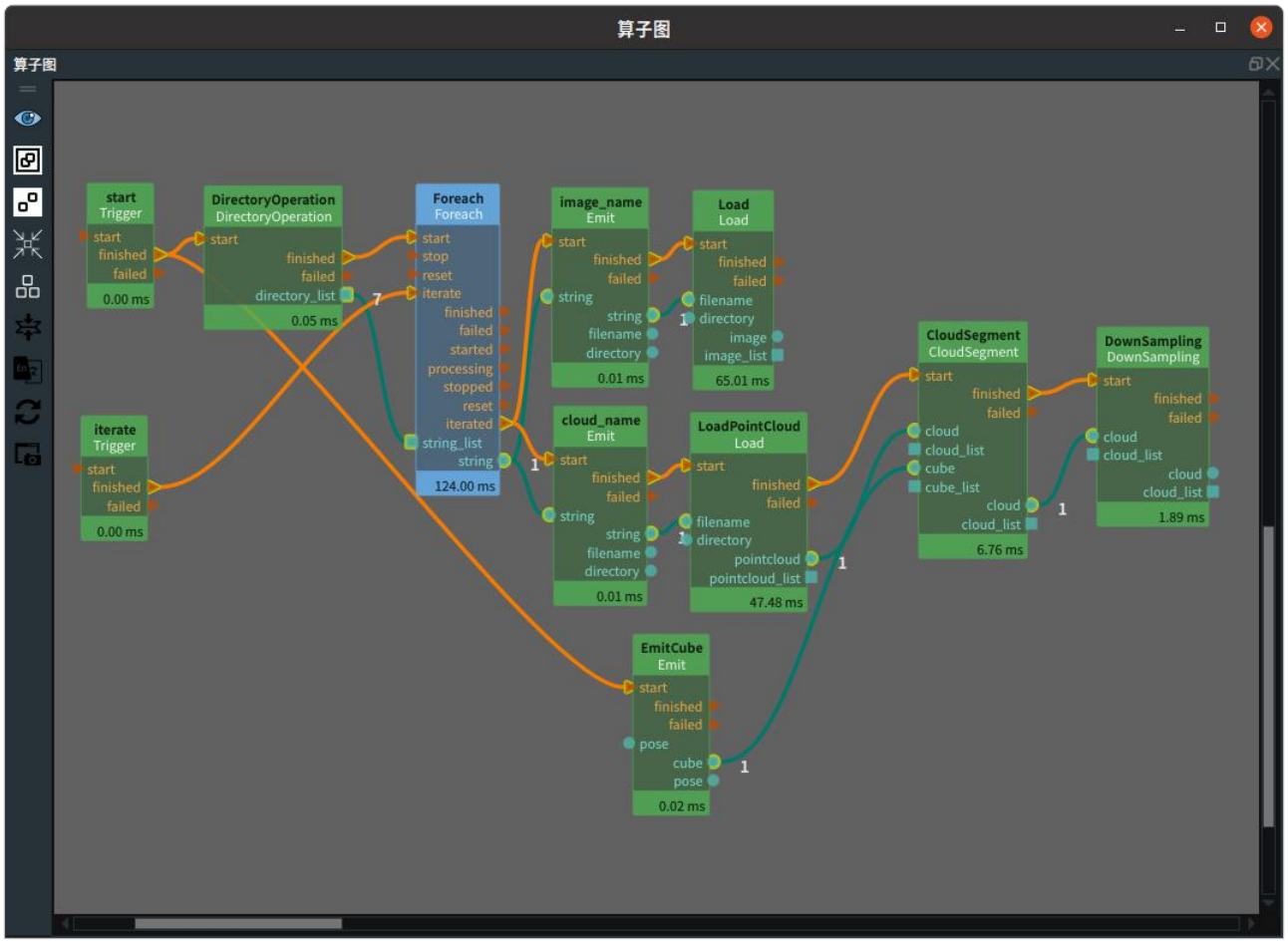
2）接下来，添加 CloudSegment 算子，type 属性选择“CropboxSegment”，该算子可以用给定的立方体切割出所需要需要的点云，连接 LoadPointCloud 的 finished 端口至本算子的 start 端口，pointcloud 输出端口连接至本算子的 cloud 输入端口；同时 EmitCube 算子的 cube 端口连接至本算子同名输入端口，您可以修改 EmitCube 算子的属性 width、height、depth 来改变立方体的长宽高，打开 cube 可视化属性，可在 3D 区查看其在点云图中的具体位置和大小，需要注意的是，每次修改立方体大小都要重新触发来查看；移动 Cube 包裹住您所需要的点云区域，EmitCube 将会自动更新对应 pose。算子图连接如下图所示：



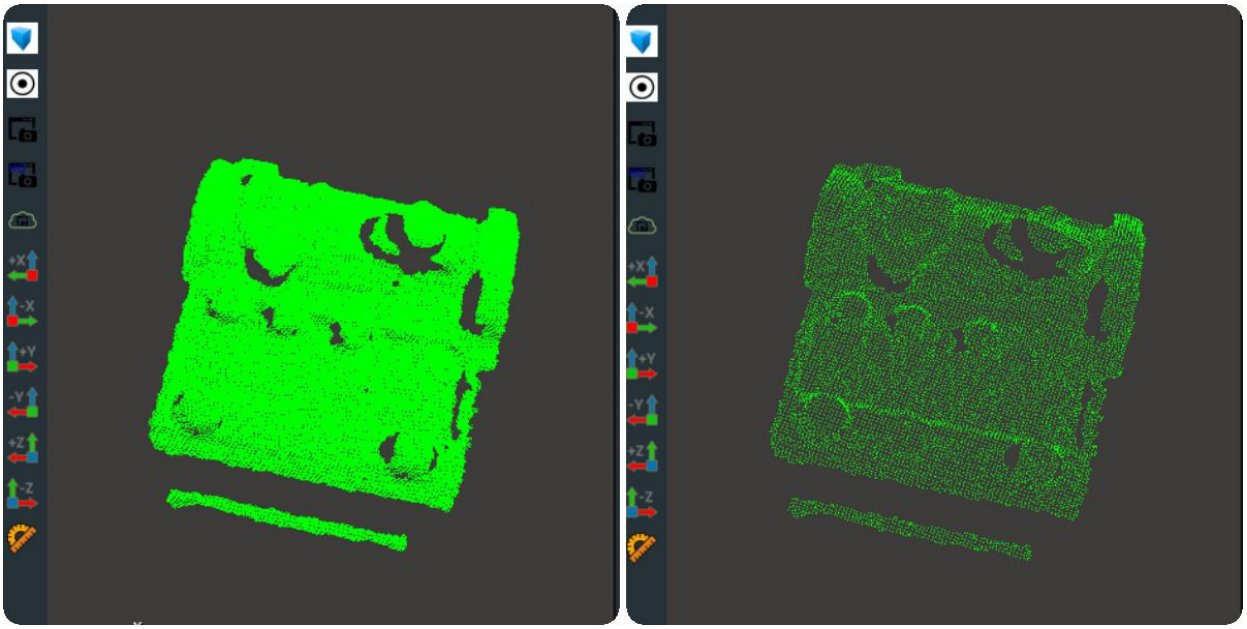
3）下图为裁剪框前后的点云：

切割前点云　　生成 cube　　切割后点云（绿色部分）

4）将切割后点云降采样，提高算子执行效率。拖入 DownSampling 算子，将 CloudSegment 算子的 finished 端口连接至 start 端口；cloud 端口连接与同名端口连接。将 DownSampling 算子的 leaf_x、leaf_y、leaf_z 指定 xyz 轴方向点云重采样间距，此处设置为 0.002m，表示在 0.002m*0.002m*0.002m 的空间尺度内仅取一个点。算子图连接如下图所示：



5）下图为降采样前后点云对比：



降采样前　　降采样后

四、 初始推测值准备

模板匹配计算的是模板点云到实际点云的转换 Pose，在计算其真实值之前，我们需要给定一个较为准确的初始评估值。评估值同真实值越接近，则越容易获得稳定的匹配结果。

获得目标的实际点云后，可以通过最小包围框算子（MVBB）来获得目标点云的中心位姿 Pose 并作为初始推测值。由于此时 Pose 的 xyz 值即为点云中心点，所以此时如果根据该 Pose 将模板点云进行转换，则转换后的模板点云中心也位于实际点云中心附近。又由于目标实际点云的长短边分别沿着 Pose 的 x 轴或 y 轴方向，所以模板点云转换后的长短边也同实际点云的长短边彼此平行。

但是此时存在“转换后的模板点云的 x 轴变成了实际点云的 y 轴”或者“虽然 x 轴对应了 x 轴但是正方向定义不一样”的情形，所以，我们需要基于该 MVBB 的 Pose，沿着其 z 轴依次旋转 90 度总共获得 4 个 pose，再全部作为初始推测值，即可保证四个 pose 中必定有一个对应了真实情形。

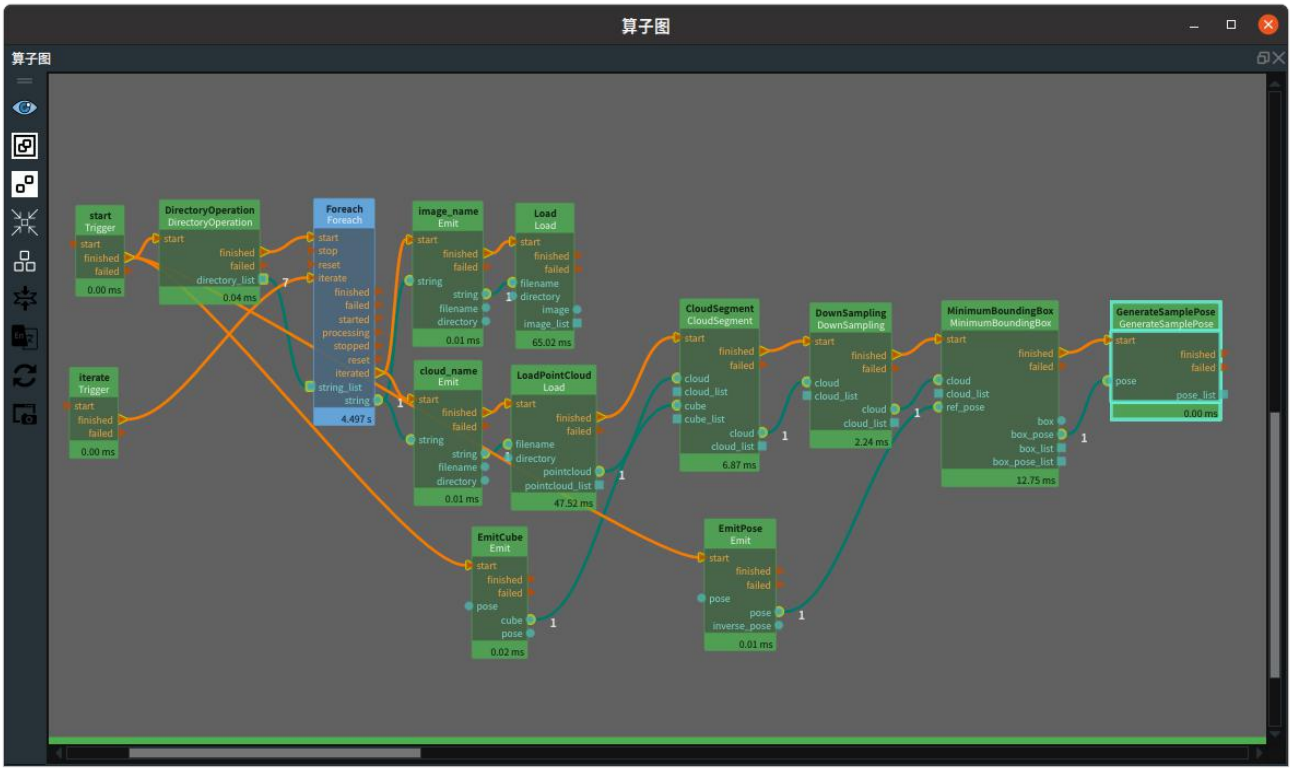
操作流程：

1）获取点云最小包围立方体以及该立方体的中心点位姿。拖入 MinimumBounding 算子，type 属性选择“ApproxMVBB”，连接 DownSampling 算子的 finished 至该算子的 start 端口，同时连接 cloud 端口至本算子同名输入端口。打开属性 obb 可视化属性，即可显示计算出的最小立方体包围框；打开 obb_pose 可视化属性，即可显示计算出包围框的中心点。

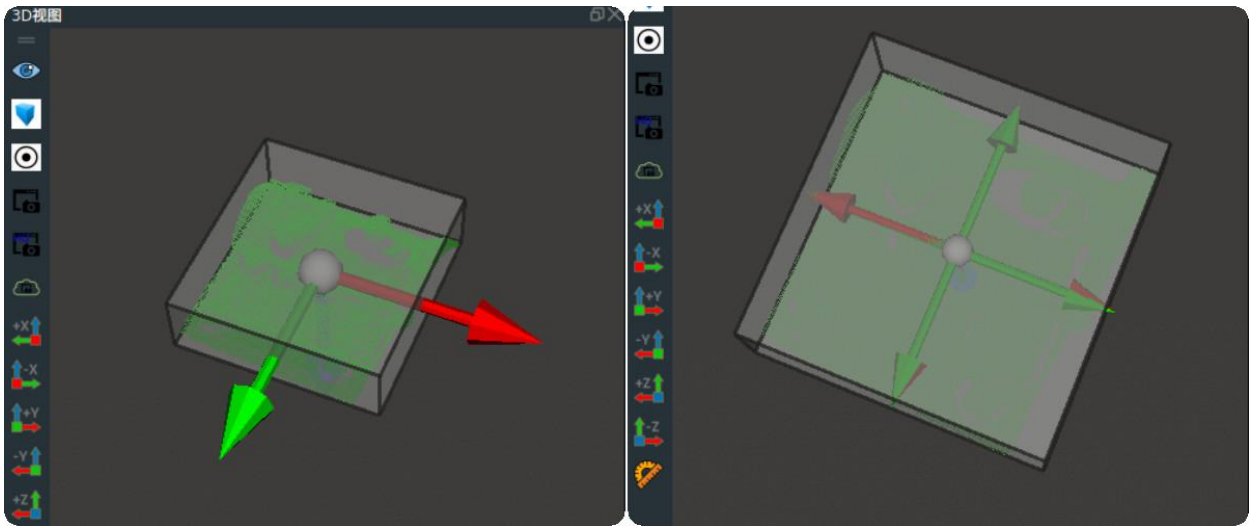
2）拖入 Emit 算子，type 属性选择“Pose”，重新命名为“EmitPose”，将 statrt 端口的 finished 端口与其 start 端口进行连接，将其右侧 pose 端口与 MinimunBoundingBox 左侧 ref_pose 连接。

3）生成四个方向的坐标作为初始推测值。拖入 GenerateSamplePose，type 属性选择“BoxGridSample”，将 MinimunBoundingBox 算子的 finished 连接至本算子的 start 端口，obb_pose 输出端口连接至本算子的 box_pose 输入端口。将 GenerateSamplePose 算子的 min_position 属性中 min_poseition_yaw 输入“-3.1415926 52”;max_position 属性中 max_poseition_yaw 输入“1.5708”（XYZ 都置 0），samples_yaw 输入“4”；设置属性面板 pose_list_visibility 为 True，显示生成 4 个 pose。

4）算子图连接如下：



5）下图为 MinimumBoundingBox 和 GenerateSamplePose 在 3D 视图中的显示：



MinimumBoundingBox 算子　　GenerateSamplePose 算子

五、 模板匹配

经过上述四个步骤后，我们已经准备好了模板点云（source_cloud）、目标点云（target_cloud）、4 组初始值 initial_pose，此时，我们需要使用 ICP 算子获得最优变换 result_pose，将模板点云根据该 pose 进行空间变换后与目标点云重合。

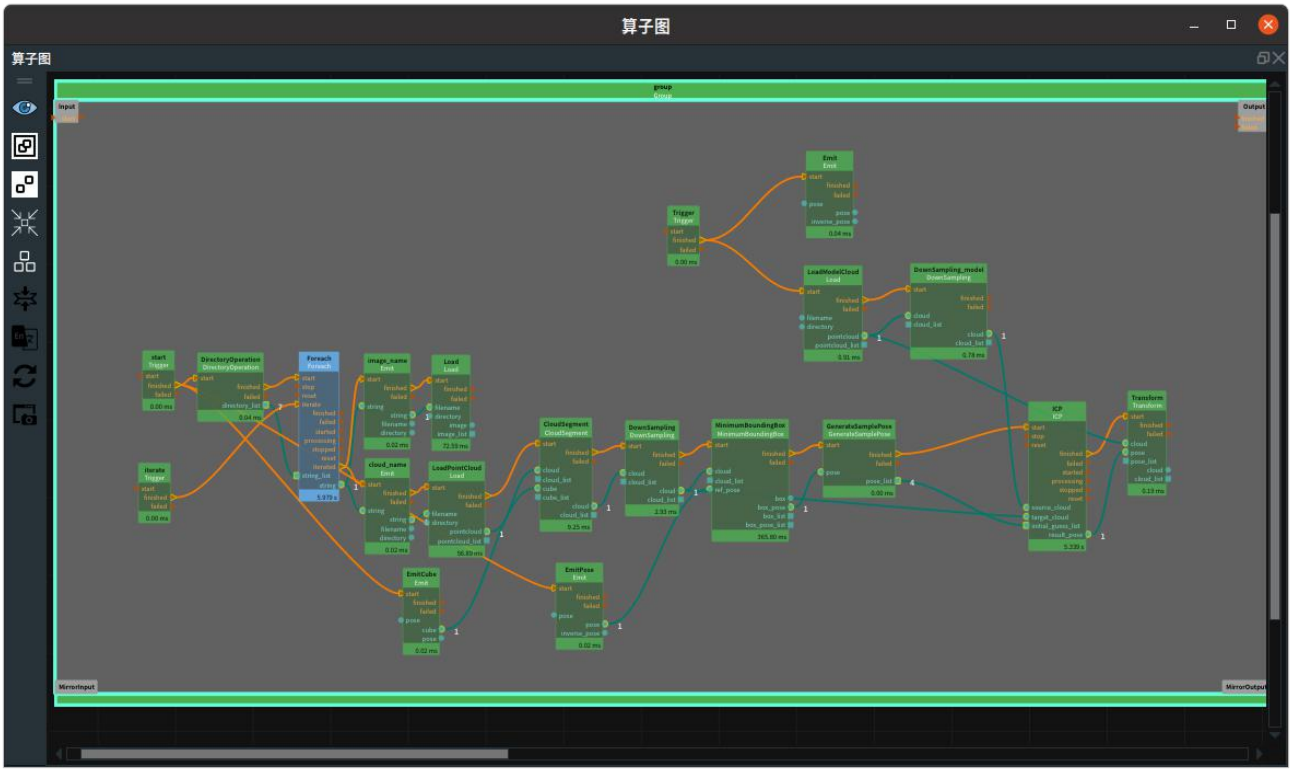
操作流程：

1）拖入 ICP 算子，将 GenerateSamplePose 算子 finished 端口连接至该算子的 start 端口，同时连接 pose_list 至该算子的 initial_guess_list 端口；“DownSampling_model”算子的 cloud 端口连接至 ICP 算子的

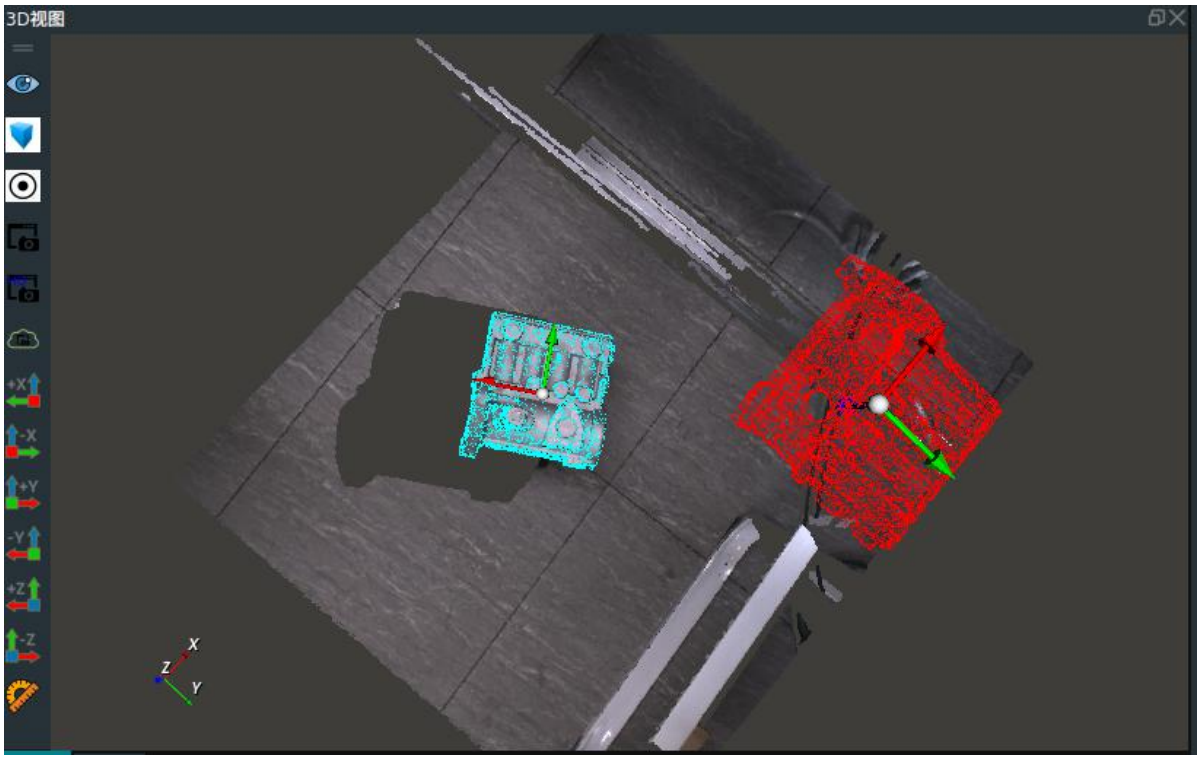
source_cloud 端口；DownSampling 算子的 cloud 端口连接至 ICP 算子的 target_cloud 端口。

2）调整 ICP 算子的属性参数，使得计算出的结果更准确：

- max_correspondence_distance：0.005；该值表示目标点云与模板点云匹配时两个匹配点之间的最大空间距离。超过该距离认为该组匹配点无效。过多的无效匹配意味当前的匹配效果很差。
- ransac_outlier_rejection_threshold：0.15；该值表示去除目标点云 0.15m 外的杂点。
- max_iterations :2000 ;ICP 算子会在所有的初始评估值的基础上 ,分别进行不断的迭代优化，每一个初始评估值的最大优化迭代次数是 max_iterations 对应的数值。
- transformation_epsilon：0.00000001；该值表示两次相邻迭代之间的平移变换矩阵的最大调整值。
- rot_epsilon：0.00000001；该值表示两次相邻迭代之间的旋转变换矩阵的最大调整值。（通常 transformation_epsilon 与 rot_epsilon 的值一致。）
- score_threshold：-1；该属性表示分数阈值，得分越低，匹配效果越好。如果最终的最优匹配结果得分超过该阈值，算子触发 failed 信号；该数值默认为-1，表示在计算结果中取最优值。
- 输出得分最高的推测值（result_pose）后，可以将点云模板根据该 pose 进行空间变换。拖入 Transform 算子，type 属性选择“PointCloud”，将 ICP 算子的 finished 连接至该算子的 start 端口，result_pose 输出端口连接至该算子的 pose 端口；LoadModelCloud 算子（1.3 中加载模板点云的算子）pointcloud 输出端口连接至 Transform 算子的 cloud 输入端口。算子连接如下图所示：



3）将 LoadPointCloud 算子 pointcloud 属性打开 ;ICP 算子 result_pose 属性打开 ;Transform 算子 cloud 属性打开；LoadModelCloud 算子（1.3 中加载模板点云的算子）pointcloud 属性打开；在 3D 视图中最终显示如下图所示：（图中原始色彩点云为目标点云，红色为匹配前模板点云，蓝色为匹配后点云。）



至此 match_model.xml 已经全部完成;完整的 xml 已提供在 model_matching/match_model.xml 路径下。

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）。