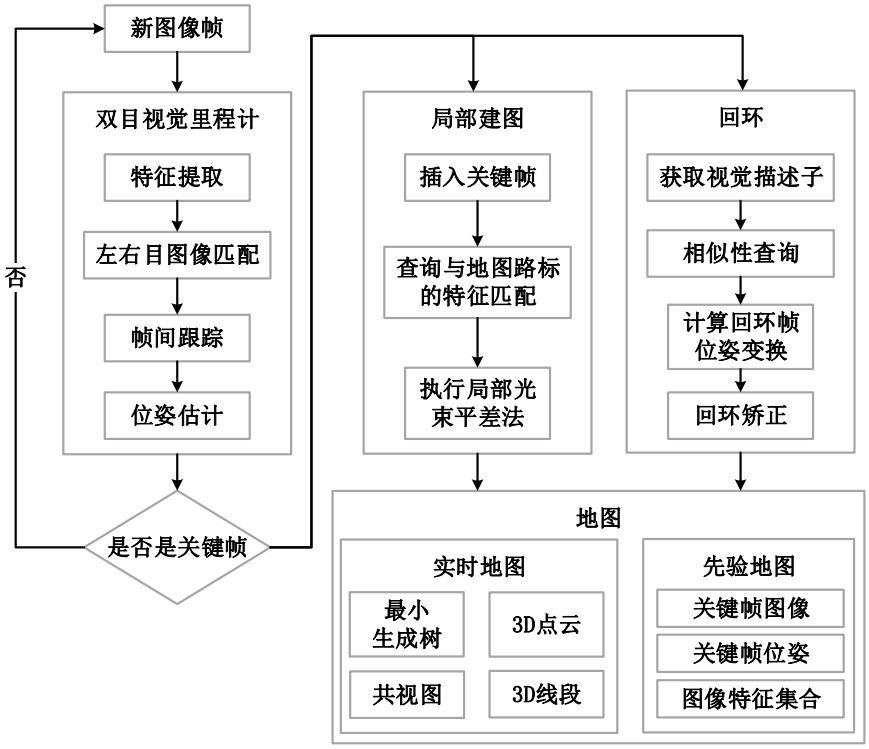
双目视觉 SLAM 方案技术开发阶段性验收报告

1. 系统运行

通过运行 catkin_ws/src/launch/sweeper.launch 运行整个系统。Demo 视频保存在 catkin_ws/src/args/all_dataset 下。下面分模块进行介绍。

2. 定位与建图

2.1 点线 SLAM 方法



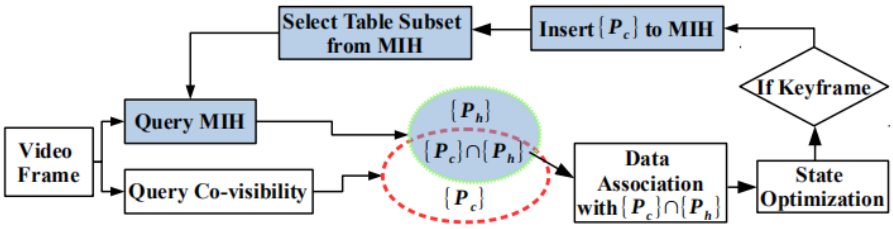
代码实现：

- 线特征提取与描述子提取
- 基于栅格的线特征匹配
- 线特征重投影误差构建
- 代码参考：ORB-SLAM2, PL-SLAM

结果：

Translation	1		2		3	
	mean	median	mean	median	mean	median
ORB-SLAM	1.357	1.2358	1.2214	1.1612	1.2255	1.1294
PL-SLAM	2.0619	2.0680	2.2388	2.4932	1.4777	1.5083
Rotation	1		2		3	
	mean	median	mean	median	mean	median
ORB-SLAM	0.0153	0.0032	0.0122	0.0021	0.0041	0.0021
PL-SLAM	0.2032	0.0765	0.2188	0.0813	0.1290	0.0458

2.2 去除局部地图中的冗余匹配



思路解析：

- 利用图像中具有共视关系的特征点的描述子构建多个哈希树
- 关键帧插入时利用之前构建的哈希树对关键帧的特征点进行索引
- 依照索引结果获得局部地图中关键帧序列的公共特征点
- 取哈希树得到的特征点与共视关系得到的特征点的交集，达到减小优化特征点数量的目的

结果：

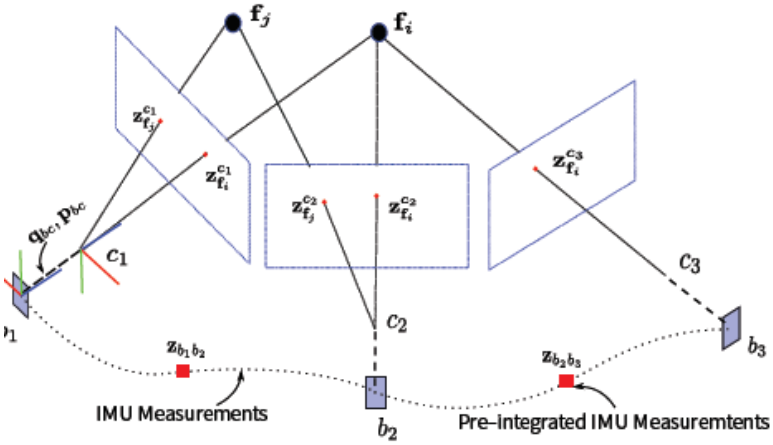
- 耗时：统计了不同数据集上的最大耗时和平均耗时，无明显差别（可理解为构建哈希树和索引的时间与该方法节省的优化时间抵消）
- 精度：在一半数据集上精度得到提升，另一半下降，不好确定该方法的有效性。

2.3 双目立体匹配加速

思路：去掉原有的对整幅图片进行矫正，转而只对特征点进行矫正（去畸变）。匹配范围也从极线搜索转为指搜索极线附近的特征点。

结果：每帧的运行时间减少 10ms。

2.4 IMU 偏置估计



$$arg\min_{\delta \mathbf{b}^g} \sum_{k \in B} \left\| 2 \left[\mathbf{q}_{c_0 b_{k+1}}^{-1} \otimes \mathbf{q}_{c_0 b_k} \otimes \mathbf{q}_{b_k b_{k+1}} \right]_{xyz} \right\|^2$$

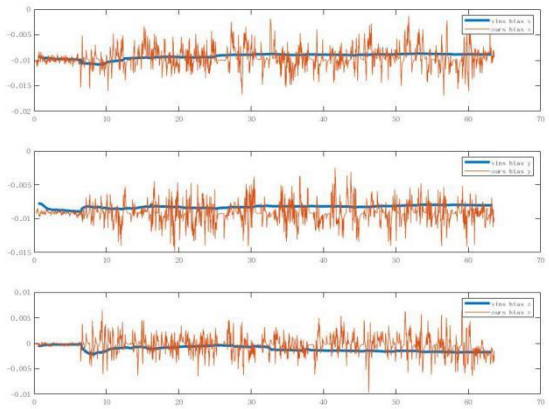
$$\mathbf{q}_{b_k b_{k+1}} \approx \hat{\mathbf{q}}_{b_k b_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \mathbf{J}_{b^g}^{\mathbf{q}} \delta \mathbf{b}^g \end{bmatrix}$$

思路：因为基础框架 ORB-SLAM 有自己的视觉特征优化框架，因此集成 VINS 系统的紧耦合方法难度较大，但考虑到相机并不会跟着时间漂移，且 ORB-SLAM 通过建立局部地图，连续几帧的位姿变化误差较小，因此采用如上松耦合方案：假想连续几帧的位姿变化误差较小，以此通过最小二乘估计 IMU 偏置。

算法流程：

- IMU 预积分，得到关于偏置的雅克比矩阵
- 建立滑动窗口，得到 ORB-SLAM 局部建图优化过后的窗口位姿变化量
- 利用最小二乘进行优化，得到偏置估计
- 设定偏置变化阈值，并基于一阶滤波器对偏置进行平滑
- 采用 ZUPT（零速度更新），检测静止的情况，对 IMU 偏置进行更新，并将此结果与依靠相机得到的 IMU 偏置结果进行加权融合。

结果：

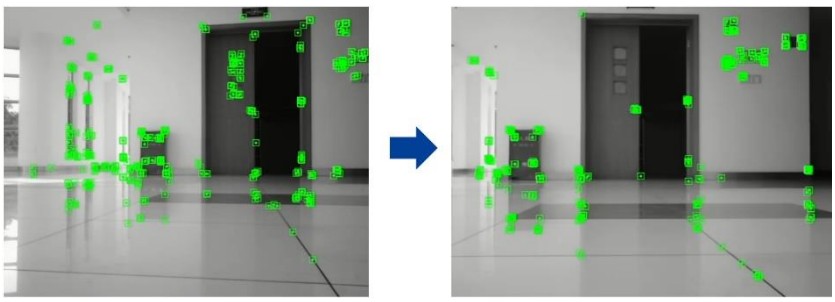


蓝色是 VINS 系统估计的结果，橘红色是我们方案的结果，最大估计误差在 0.005° /s，耗时 1.7ms，可以使用。

2.5 多子图连接与平滑本质图构建

思路：

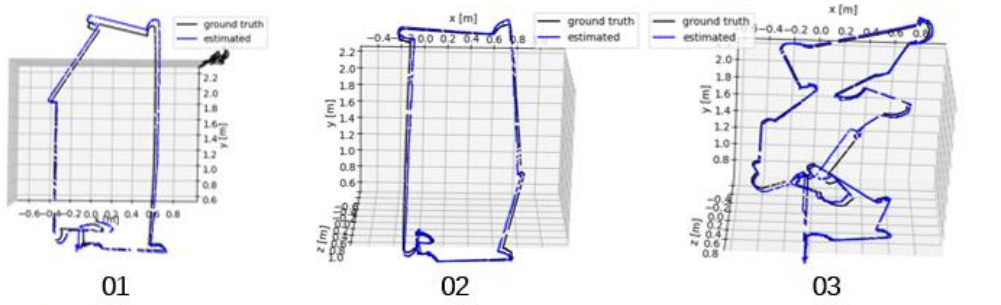
- 检测一个条件发生（例如机器人被抱起，轮子空转），则新建一个地图
- 新地图的关键帧放入数据库，寻找与旧地图的重合点，若发现则拼接地图。
- 在一个地图中，检测“丢失条件”发生（例如局部地图中没有足够多的特征点），则根据里程计距离和角度变化，插入不需要维护特征点的关键帧，并将此关键帧的父节点设定为上一帧能正常跟踪的关键帧。



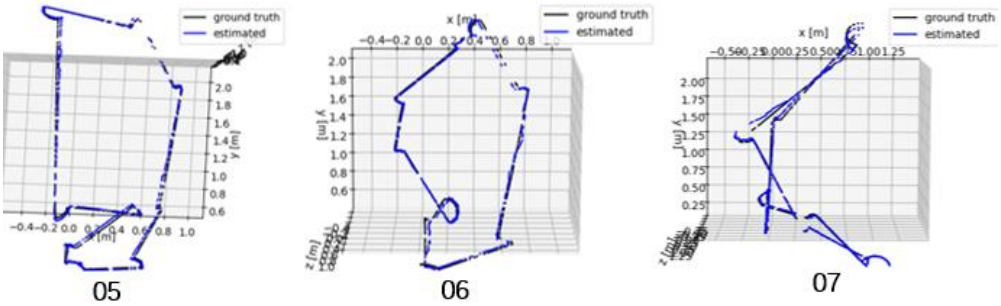
2.6 算法评测

平移误差百分比 = 估计值与真实值的欧氏距离 / 已走路程

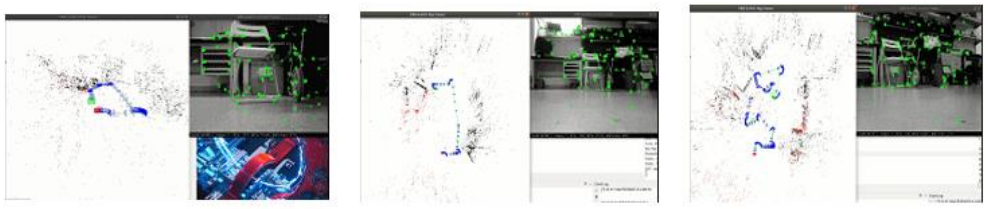
旋转误差百分比 = 估计值与真实值相对旋转的欧拉角的模长 / 已旋转的角度



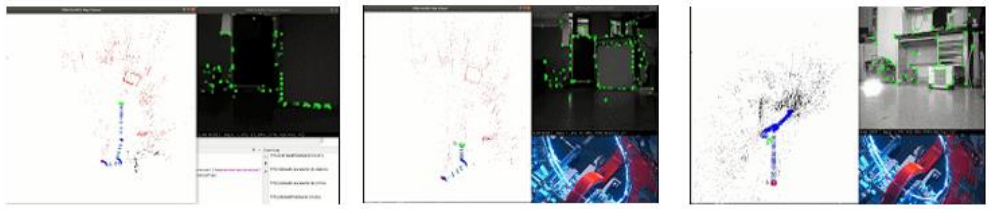
01 02 03



05 06 07



01 02 03



05 06 07

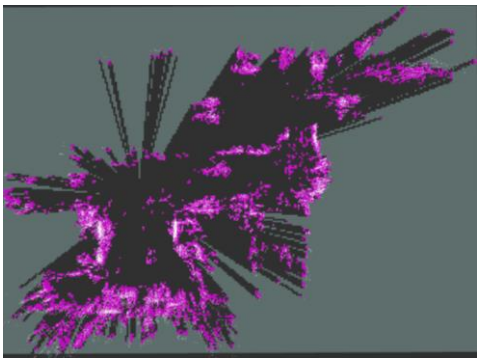
数据集介绍：01,02,03：正常光照；05,06：昏暗环境，包含直接图片漆黑；07：直视台灯，运动物体。

结果：

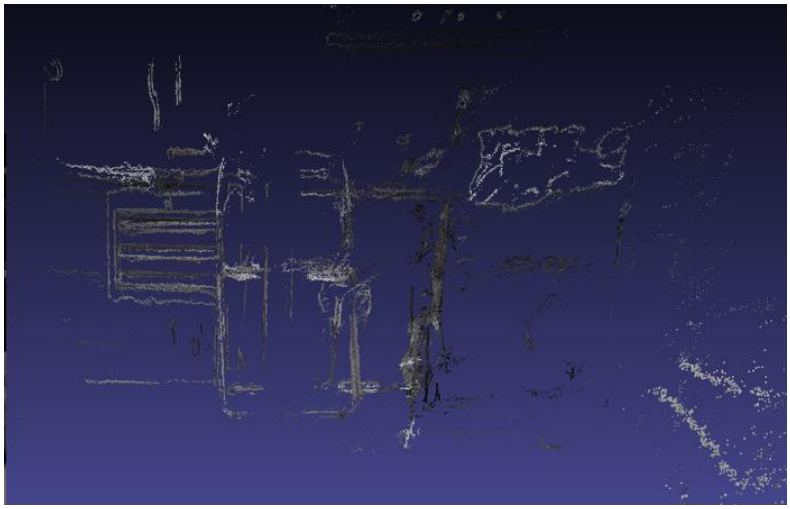
Translation Error (百分比)	1（11.3m）		2（11.1m）		3（11.7m）	
	mean	median	mean	median	mean	median
	1.654	1.203	0.841	0.727	0.629	0.492
	5（11.8m）		6（9.3m）		7（12.7m）	
	mean	median	mean	median	mean	median
	0.544	0.497	0.63	0.574	0.922	0.479
Rotation Error (百分比)	1（11.3m）		2（11.1m）		3（11.7m）	
	mean	median	mean	median	mean	median
	0.009	0.001	0.009	0.001	0.002	0.0009
	5（11.8m）		6（9.3m）		7（12.7m）	
	mean	median	mean	median	mean	median
	0.011	0.001	0.019	0.006	0.008	0.002

2.7 地图可视化

- 2D 栅格地图构建



- 依照线特征与深度图进行半稠密地图构建



展示了效果较好的一个结果，但由于双目深度估计的准确度仍然不是很理想，导致线段拟合有时会出现较多的错误，因此这个建图方法并不鲁棒。

3. 重定位

3.1 点线词典构建

思路：

- 由 ORB 特征提取器获得点特征描述子集合
- 由 LSD 检测线特征，LBD 提取线特征描述子集合
- 基于点线特征集合，实现小规模词典构建

评价指标：

训练与测试数据集为同一场景的不同序列

Recall_N： 对于一张图片，找到最相似的 N 张，若该 N 张找对了 M 张，则计算真值占比 M/N，统计平均召回率

真值：VICON 获得的轨迹位移小于 20cm，角度小于 10° 认为是属于同一场景。

结果：

	ORB	ORB+LBD
Recall_1	99.6516783	99.6516783
Recall_20	99.3263141	99.3587714
Recall_40	98.4456143	98.5366529
Recall_60	95.356502	95.6449229

3.2 词袋法加速

- 已实现接口：FBoW，DBoW2，DBoW3
- 以实现功能：词典读取，词典保存，词典训练，词袋向量转换，词袋向量评分
- 结果：

K=10, L=3

	加载	保存	训练	词典向量转换
DBoW2	4.9ms	4.2ms	114.0ms	2.7ms
DBoW3	870.9us	300.4us	152.1ms	**1.6ms**
FBow	**24.3us**	**217.2us**	**116.8ms**	2.0ms

4. 主方向识别

4.1 方案

主方向识别基于房间的物体大都垂直于墙体摆放的假设，通过检测图像上的灭点得到房间主方向相对于相机的旋转角。主方向识别的过程如图 4-1 所示。对于每一帧新的双目图像，分别对左右两张图像进行 LSD 直线检测、灭点检测，并利用相机投影公式分别得到主方向相对于两个相机的旋转角。然后对双目的检测结果进行双目验证：如果两个相机的旋转角结果的差小于阈值则认为通过双目验证。如果通过了双目验证，则通过轮式里程计得到的偏航角将当前帧的主方向结果转换到第一帧下，并将结果存入容器中，并且等待新一帧图像。如果距离第一帧间隔超过了 N（我们取 50）帧，则利用容器内的所有第一帧检测结果，利用 RANSAC 算法得到最终第一帧主方向检测结果和置信度。如果置信度小于 0.5，那么重新进行主方向识别，直到得到置信度大于 0.5 的结果，并且将此时的主方向识别结果利用里程计的值转换到全局初始帧下。

需要说明的是，主方向识别仅在扫地机器人系统运行初进行，一般 10s 左右可以完成。主方向成功识别的前提是基本假设的满足，即要求房间中的物体都垂直于墙体摆放。为了得到更好的主方向结果，要求主方向识别过程中扫地机器人缓慢平稳的运动。

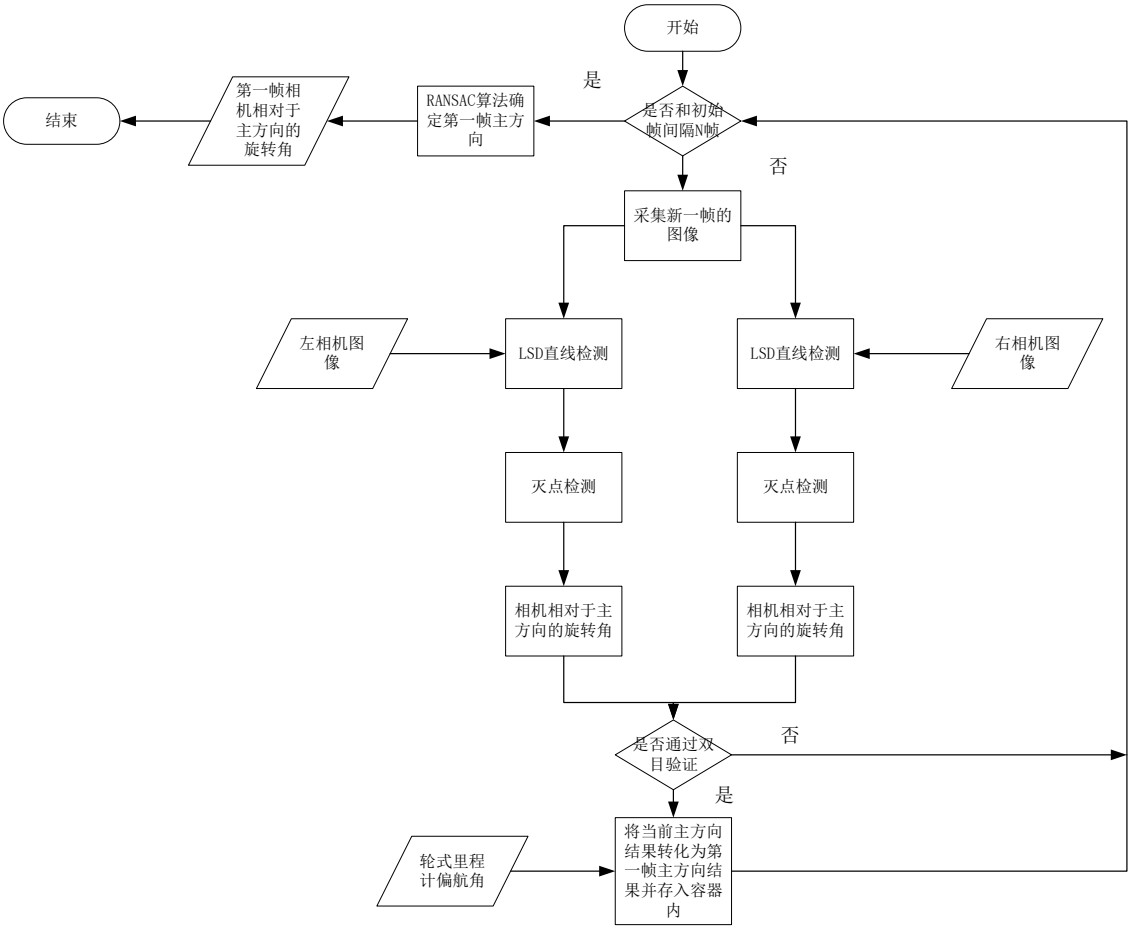


图 4-1 主方向识别流程图

4.2 程序运行

运行(注意将-d 后面的地址修改为自己的保存相机参数的地址)

rosrun main_direction_detection main_direction_detection_node -d=/home/qzj/code/catkin_ws/src/args/all_dataset/

运行主方向识别节点

（1）订阅的消息包括：

"/cam0/image_raw"

"/cam1/image_raw"

"/imu0"

（2）发布的消息包括：

"/main_direction_yaw"：主方向相对于相机旋转角

4.3 精度检测

4.3.1 检测方法

在 dataset/10/提供的 2020-09-25-13-30-56.bag 数据集初始的 0-15s 内随机开始进行主方向识别，识别结果保存在 catkin_ws/src/args/all_dataset/main_direction_result/main_direction_gt.txt 中，各列表示：

(1) 用于主方向识别的 50 帧中第一帧的时间戳；

(2) 房间主方向相对于相机的角度值；

(3) 置信度；

(4) 程序开始运行的全局第一帧时间戳；

(5) 全局第一帧房间主方向相对于相机的角度

真实值通过 VICON 测量得到，保存在 catkin_ws/src/args/all_dataset/main_direction_result/main_direction_gt.txt 文件中，各列表示：

(1) 时间戳；

(2) 房间主方向相对于相机的角度值；

通过时间戳将主方向检测结果和真实值进行对应和比较。

4.3.2 精度检测结果

角度误差的平均值为 1.02 度。认为角度误差小于阈值 T 度即为识别成功，T 取不同值时的识别成功率为：

T = 2:　　92%

T =2.5：　98%

5. 障碍物检测和目标识别

5.1 方案

障碍物检测和目标识别模块既需要对障碍物进行检测以实现避障，也需要对羽毛球、麻将、线和狗屎这四类物体的位置、大小和类别进行目标识别，为更上层的任务服务。整体过程如图 4-1 所示。

障碍物检测在相机距离地面高度已知的前提下，通过将同一像素纵坐标的视差和地面视差进行比较得到障碍物掩模，然后将障碍物掩模作用与视差图上，利用双目模型和相机投影模型从视差图得到障碍物点云。

物体识别部分首先通过 yolo 网络得到二维检测结果，然后对每一个二维检测框内的区域的像素点根据 3D 图映射到三维空间，并对这部分点云进行聚类，对各个聚类按照距离相机光心距离、聚类点云点数和点云投影到像素平面后的二维框和二维检测框交并比进行打分，得到物体点云聚类，并将二维检测框的类别赋予该点云聚类。

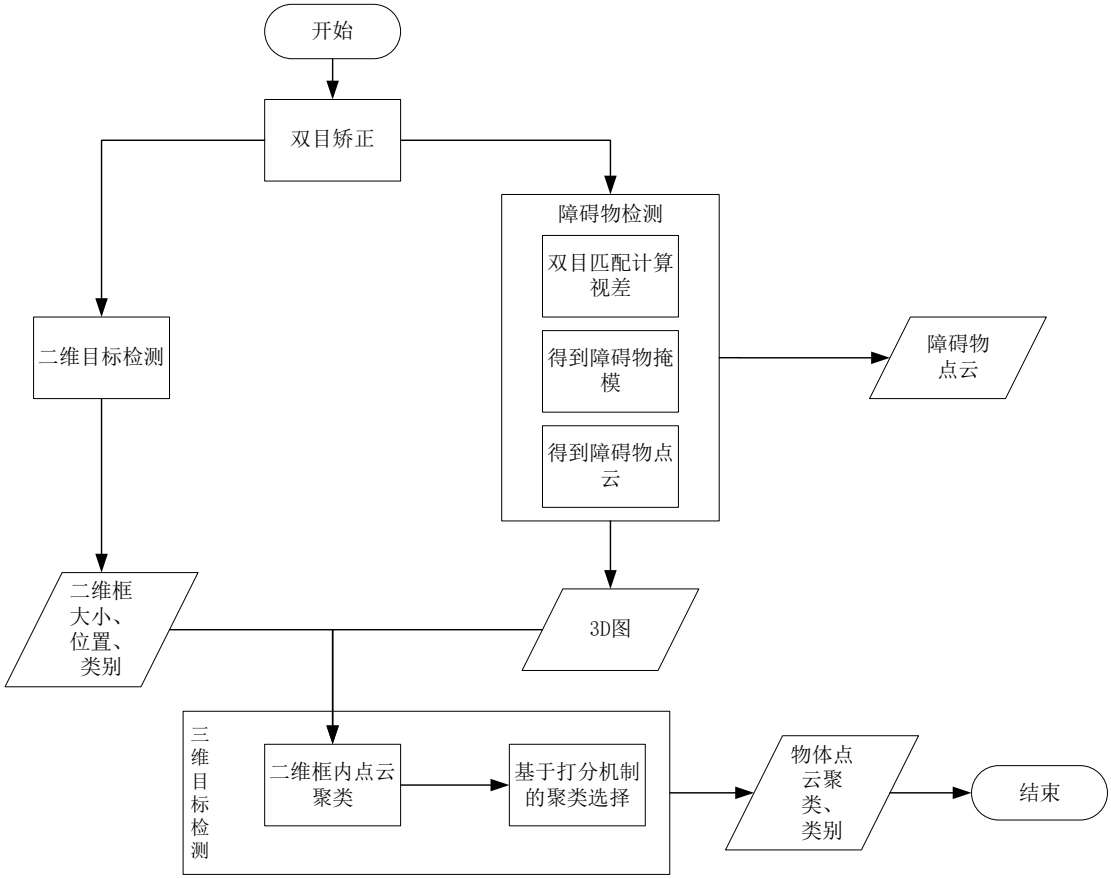


图 5-1 障碍物检测和物体识别流程图

5.2 程序运行

运行(注意将-d 后面的地址修改为自己的保存相机参数的地址)

rosrun objection_detection object_detection_v2_node -d=/home/qzj/code/catkin_ws/src/args/all_dataset/

运行障碍物检测和物体识别节点

（1）订阅的消息包括：

"/cam0/image_raw"

"/cam1/image_raw"

（2）发布的消息包括：

"/rectified_left_image" : 双目矫正后左图像

"/rectified_right_image"：双目矫正后右图像

"/obstacle_mask"：障碍物掩模

"/points_obstacle"：障碍物点云

"/points_object"：物体点云

"/2d_obj_det"：带有二位检测结果的图像

"/yolo_mask"：物体掩模

5.3 精度检测

5.3.1 检测方法

为了检测精度，单独录制了四类物体距离相机深度 0.5m 到 1.8m，横向距离-0.15m,0m，0.15m，包含 240 张图片的数据集，保存为 dataset/10/exp0702.bag。

通过运行 catkin_ws/src/launch/object_detection.launch 进行精度检测。

结果保存在 catkin_ws/src/args/obj_det_dataset/object_det_matlab_sgbm.txt 中。各列分别表示：

(1) 帧序号

(2) 该帧 yolo 检测到的物体数量

(3) 该物体类别序号（0：羽毛球；1：线；2：麻将；3：狗屎）

(4) 该物体世界坐标系下 x、y、z 坐标(m)

真实值通过尺子测量得到，保存在

catkin_ws/src/args/obj_det_dataset/object_det_result/gt.txt 中，各列表示：

(1) 类别序号

(2) 世界坐标系下 x、y、z 坐标(m)

5.3.2 检测结果

运行

python catkin_ws/src/object_detection/scripts/eval/eval.py data_to_catkin_ws/src/args/obj_det_dataset/ object_det_matlab_sgbm.txt

位置估计结果输出：

////////////////obsatcle result//////////////////

e_z_0: 0.014215000000000021

e_z_1: 0.024795425000000003

e_z_2: 0.012935264705882354

e_z_3: 0.01760174999999999

e_x_0: 0.032323202894736835

e_x_1: 0.035623471250000004

e_x_2: 0.03481014411764706

e_x_3: 0.031825130000000014

其中，其中 z 表示 z 轴方向(深度方向)的误差,x 表示 x 轴方向(横向)的误差,序号 0-3 为类别序号,仅对 0.5m-1.4m 范围内的物体进行精度评估。

物体识别率输出如下

count0: 38 38 1.0

count1: 40 40 1.0

count2: 34 40 0.85

count3: 40 41 0.975609756097561

all:0.9559748427672956

其中，前四行分别输出四种物体各自的识别成功个数,总个数,识别成功率,最后一行输出总的识别成功率。

6. 毛毯识别

6.1 方案

基于障碍物检测和目标识别的结果,可从感兴趣区域的图像中剔除大部分的干扰物体，从而保留地面区域。通过在地面上寻找连通区域，以面积的大小作为判断依据，最终可得到毛毯所在区域。毛毯识别方案的流程图如图 6-1 所示。

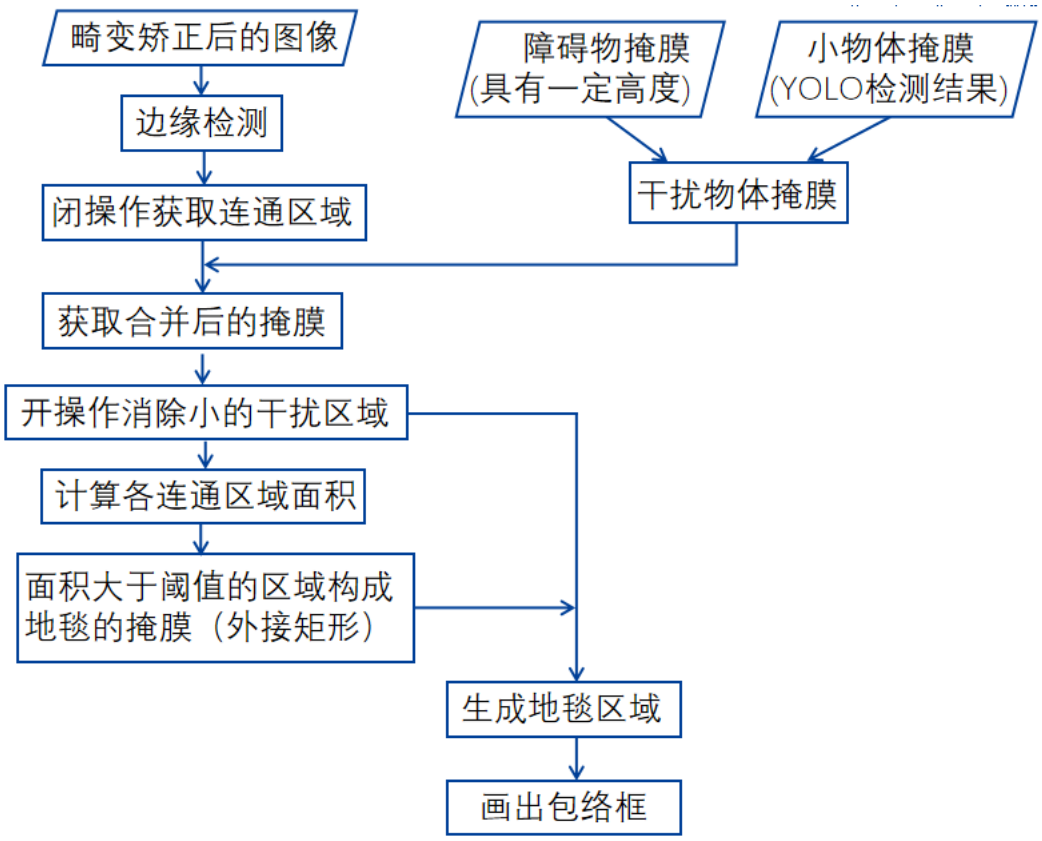


图 6-1 毛毯识别方案流程图

毛毯识别的结果主要由两个掩膜经过基本运算组合而成。

（1）障碍物掩膜与 YOLO 检测得到的小物体掩膜共同构成干扰物体掩膜；

（2）对边缘检测的图像进行闭操作（基于同一尺寸的核，先膨胀再腐蚀）处理得到所有连通区域的掩膜。

将上述两个掩膜进行与运算后，可得到含有多个连通区域的掩膜。为了消除部分因掩膜合并后遗留下的小区域，同时不影响其他连通区域，还需对图像进行开操作处理，即利用相同尺寸的核，对图像先进行腐蚀处理再进行膨胀处理。对于图像中所有的连通区域，可通过检测闭合轮廓的方式找到，再进一步计算各轮廓相应的面积，设置关于面积的阈值，便可区分出毛毯与其他诸如地贴，胶带等位于地面上的干扰区域。最终，只需将地毯区域掩膜与感兴趣区域的图像做与运算就能生成毛毯区域的图像，包络框则为不规则毛毯区域对应的外接矩形框。

优化：由于纯色毛毯的纹理特征不明显，难以检测到毛毯中间部分的边缘，无法直接得出毛毯的准确区域，但仍然能从图像中识别到地毯的边缘部分。为此，考虑当所识

别到的毛毯最远处位于图像中间以下的区域时，可补全图像的上半部分，扩大包络框的范围。原因为此时图像中显示的区域可以被视为毛毯的边缘，并且毛毯与机器人的距离较近，即感兴趣区域基本被毛毯部分所占满。在执行此操作后，算法对于纯色毛毯的识别成功率以及可识别距离具有一个较大地提升。

6.2 通讯架构

目前算法的通讯主要基于 ROS 来实现，毛毯识别与目标检测的通讯架构如图 6-2 所示。

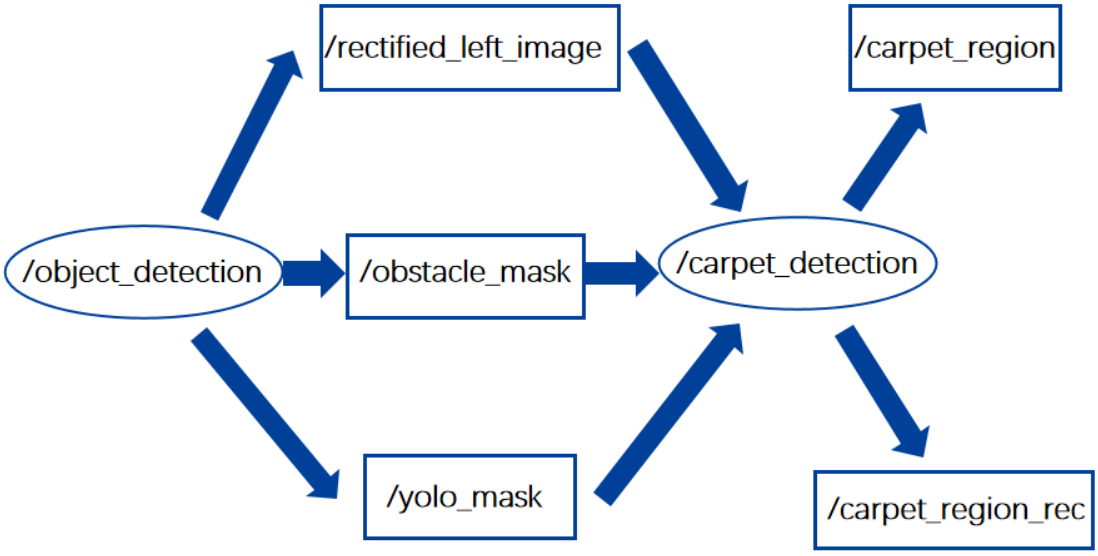


图 6-2 ROS 通讯架构示意图（毛毯识别与目标检测）

毛毯识别节点(“carpet_detection”)订阅与发布的所有消息如下：

（1）订阅的消息包括：

″/rectified_left_image″：双目矫正后左图像

″/obstacle_mask″：障碍物掩模

″/yolo_mask″：物体掩模

（2）发布的消息包括：

″/carpet_region″：不规则毛毯区域及其包络框

″/carpet_region_rec″：包络框中的毛毯区域

6.3 评价指标

6.3.1 成功识别毛毯的标准

通过肉眼判断，能够识别出图像中毛毯的有效面积至少达50%以上即判定为识别成功。

6.3.2 毛毯识别成功率

在0.2-0.6m的范围内，毛毯的识别成功率为95%。