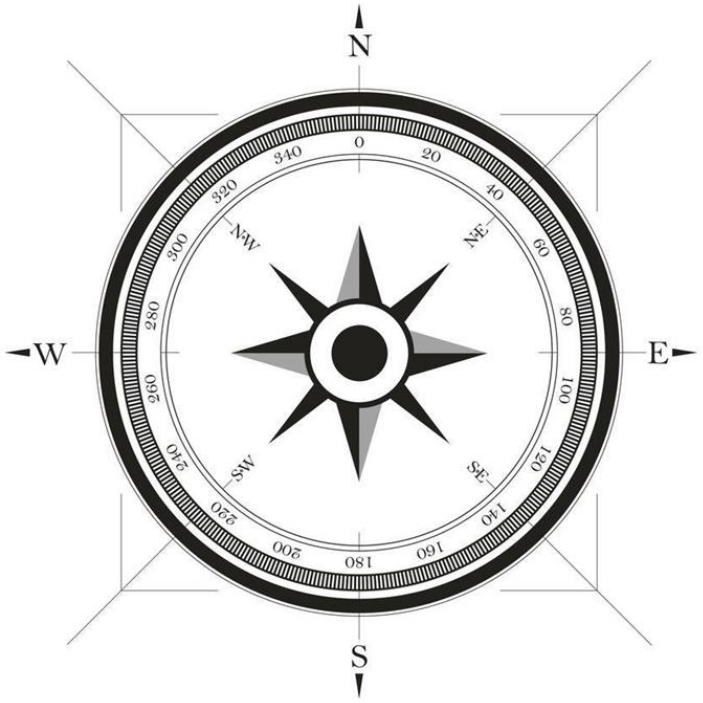


大华 AGV 导航相关施工和问题排查手册

大华机器人导航组

2020.9.8

版本修改记录

修改时间	修改内容	修改人
2020.9.8	更新二维码码带文件	项宇
2020.8.23	添加日志描述、脱轨判断、重定位失败	项宇
2020.8.22	更新二维码最新标定工具操作描述 添加激光二维码切换点标定方法	刘瑞利
2020.8.20	合并生产文档	陈亦彪、项宇
2020.7.27	初稿	项宇、吕品

目录

一、基础

1.1 Windows 基本工具集

Xshell

Smplayer，这个脱轨分析需要，售中电脑建议默认安装。

1.2 Linux 基本命令

rz 和 sz，sz 当前设备上可能叫 ssz

sudo，cd，ls，cat

1.3 Vim 编辑器基本操作

大部分人在接触 AGV 前就没有听说过 vim 更别说操作了，但是被空投到现场的你，如果不会 vim 的基本操作，除了耗费青春折磨自己，还会在联系研发后收到大大的鄙视。

按照 xxx 步骤(找个网上视频)，学习下基本的操作吧。

二、导航问题分析基础

导航使用的工具都在/home/dahua/agv_workspace/install/install_nav/lib/navtools/script 下，最常使用的是日志解析和传感器标定两个功能。

2.1 日志解析命令 nav-parse

这个命令是导航最重要的分析命令，它会将传入的导航日志按研发需要解析成多个文件，放到新建的 debug 文件夹下。

nav-parse 命令默认解析/var/robot/log/nav/nav.log 日志，也可以指定日志，比如希望解析/var/robot/log/nav.log.1，命令为：

nav-parse /var/robot/log/nav.log.1

为了节约空间，系统会将日志压缩成 nav.log.x.tar.gz 的形式，这种情况工具会先解压日志再解析，命令为：

nav-parse /var/robot/log/nav/nav.log.1.tar.gz

输入 nav-parse 后打印如下：

```
[10.35.191.15]@/home/dahua#    nav-parse
生成二维码调试辅助文件...
dm    dm_adjust    exception    map    monitor    move    odom    sec    task    temp_nav.log
```

中间生成二维码调试辅助文件的提示表明设备支持二维码功能，会额外产生二维码特有的日志。

最下面一行的内容是解析命令生成的文件，这些文件内容描述如下：

dm	扫到的所有二维码详细信息
dm_adjust	扫到二维码后对定位的调整
exception	严重问题记录，这个内容最佳情况应该是空，实际由于人为操作或代码 bug、待优化功能等会出现些问题
map	地图模块的信息
monitor	任务管理模块的信息
move	运动模块的信息
odom	定位模块的信息
sec	安全模块的信息
task	所有行走任务的信息，这个一般用于通过 taskId 查找对应任务信息
temp_nav.log	所有日志集合

现场分析最常用的是 exception 和 task 两个文件。

如果现场没有请求外部协助的条件，请通过打开 task 文件，通过大致时间或者明确的 TaskId 获取准确的出问题时间，然后找到对应时间的 exception 文件内容，通过上下文基本可以大致判断出问题的原因。比如任务是碰撞了、二维码没扫到、坐标跳变等。

如果能发手机信息，可将出问题时的 exception 内容拍图片发给研发。

如果熟悉了，和地图相关的问题就查看 map，如果不行走查看 monitor，如果避障有问题查看 sec。查看使用 cat 命令，大部分打印含义应该是清楚的。

下面是 exception 和 task 文件的详细介绍。

2.1.1 exception 文件描述

exception 文件是导航问题分析的起点，一般现场能感知的导航问题都能在 exception 中看到原因，所以推荐优先查看这个文件。常见的内容如下：

```
|T:08/22 17:08:31.848|L:Error|M:[loc]|m: fail: wait map
|T:08/22 17:19:59.569|L:Error|M:[loc]|m: fail: wait map
|T:08/22 17:22:50.996|L:Error|M:[loc]|m: fail: wait map
|T:08/22 17:29:21.238|L:Error|M:[loc]|m: fail: wait map
|T:08/22 17:32:15.801|L:Info|M:[sec]|m: recv back emergency press
|T:08/22 17:32:16.098|L:Info|M:[sec]|m: recv back emergency unpress
|T:08/22 17:36:22.017|L:Error|M:[loc]|m: fail: wait map
|T:08/22 17:36:22.604|L:Error|M:[mon]|m: fail: pose is not valid
```

开始表示这个打印的时间，然后看 m：后面的实际内容，比如上面的 wait map 表示当前没有地图，等待设置地图，最后的 pose is not valid 表示当前坐标无效，说明定位丢失了。

有些错误连续时间出现的，看第一个明确的错误就好了，后面的错误打印大概率是第一个问题触发的，比如下图短时间出现了五条打印：

```
|T:08/22 11:24:28.207|L:Info|M:[sec]|m: recv back emergency press
|T:08/22 11:24:28.215|L:Error|M:[sec]|m: fail: clear task fail
|T:08/22 11:24:28.288|L:Error|M:[loc]|m: fail: encoder disappear, stop!
|T:08/22 11:24:28.288|L:Error|M:[loc]|m: fail: clearTask with code 0x300d
|T:08/22 11:24:28.290|L:Error|M:[loc]|m: fail: clear task fail
```

第一句是后急停被按下。

第二句是按下后应用进入人工干预取消任务，但是安全模块已经取消导致取消任务失败。

第三句是定位模块检测到编码器丢失取消任务，
第四句是因为编码器丢失（0x300d）取消任务，
第五句是定位模块取消任务失败，因为当前任务已经被取消了没有任务。
可见连续报错一般看第一条明确错误即可。

常见错误字段和含义列表如下：

字段内容	含义
wait map	激光设备当前没有地图，偶尔一次没关系，连续出现说明设置地图不成功或者从来没有下载过地图
wait relocation	激光设备等待重定位，打印这个一般表示迟迟未调用重定位，如果反复出现一定是 bug，偶尔一次可能是系统卡了下
recv front emergency press	前面的急停被按下，这里的 front 字段也可能是 back，press 也可能是 unpress 表示拔出
pose is not valid	当前定位丢失，如果是二维码导航说明连续 5 米以上没扫到码，如果是激光，说明定位丢失了
get shelfLoadModel failed	获取货架模型失败，如果是仓储设备，确定是否是谁手动或者遥控升起了举升导致，否则联系研发
not match, error	激光行走过程明显匹配失败导致定位丢失，确认是否场地变化过大，或者重定位后坐标明显偏差了
encoder disappear, stop	编码器丢失停车，大部分是急停导致，如果没有按急停就出现，说明系统卡顿明显或者底层电机或者 can 异常了
clearTask with code 0xNNNN	因为某种原因外部取消任务，错误码具体见附录 1、导航相关错误码
internal cancel task reason 0x3xxx	因为某种原因内部停止行走，错误码见附录 1、导航相关错误码
send front collision event	设备碰撞
clear task fail	取消任务失败，一大半不是问题，小部分是 bug，这个研发分析即可
stop sync movement	提示表示同步导航任务被提前终止，大部分不是问题，最常见原因是栈板识别提前碰到挡板了导致任务提前结束
movebaseControl ret xx	movebase 检测到某些错误，大部分是 bug，少量正常
alg feedback when movebase	不正常但是不影响结果，现场可以忽略
relocation fail as low score	当前重定位位置错误导致重定位匹配分数很低，这时候需要调整位置或者调整重定位坐标
no map when relocation	没有地图的时候调用了重定位
set lidarx areayy failed with 100	说明避障激光 x 不存在
not support dm map but only has dm json	设备不支持二维码导航，但是地图只有二维码导航地图，新设备说明设备能力集配置错误，老设备说明地图错误
out of rail	脱轨了
refuse relocation as no scan	激光数据异常导致拒绝重定位
diff time is zero	说明连续收到时间戳一样的数据，大部分是系统有些卡或者刚启动，偶尔出现不用在乎
plan freePath (-7.660, 8.116, 89.22)->(-7.500, 8.185, 89.22) fail	到目标点（绿色）的自由上轨失败，这个现场重新遥控规避，问题反馈研发处理

move base start fail	极少见，大概率是算法 license 问题
selftest 10 x y	驱动器异常，x 为 1 表示左轮，2 表示右轮
call set_move_params fail	极少见，大概率是算法 license 问题
add slice fail, new start 00000050 but previous end 00000036	路径拼接失败，之前任务下发到绿色点了，但是本次却从红色点开始拼接，平台和设备问题概率一半一半，直接找研发处理
imu Va change 27.000d/s, try to use encoder 23.314d/s	表示设备 imu 角度方向突变，偶尔一次属于正常。有两种可能，一种是过坎过坑了，另一种是单片机时间戳不对。
resume movebase fail, will try next time	遇障恢复失败，大概率不是问题，可能是恢复任务时任务已经判断完成了
curve size 0 pack path failure	这两个都是同时出现表示尝试类似地图上的弧线轨迹上轨失败，大部分非问题
stop vehicle timeout	设备停止时间超过 2 秒，偶现不算问题
vector angle is 1.045 (05000008->01000006)	表示如果绿色码无误差，则红色码贴歪了蓝色角度，这里表示如果 06 码正常，那么 08 码贴歪了 1.04 度，超过两度的尽量处理下
ignore dm 62.8ms 05110618	表示绿色码识别完用于定位时太晚了，可能是系统卡顿，也可能是识别耗时太长，如果持续出现需要分析码是否有问题
dm adjust all 0.0316 0.0154 -0.943 0.15 05160072->05160073 1 0.032	表示扫到码时检测到坐标或者角度波动较大，这个待研发提供工具分析，除非问题严重现场不用处理
wait better 05110051	表示这个码可能不是太好，可能是脏了或者被损坏了，反复出现时建议查看，一般忽略
updatePath ret 0x3202	Movebase 添加路径失败，大部分是时序不一致，是个待优化项，设备会停止 2 秒，然后可以恢复
alg …. err with xxxx	算法返回错误，错误码见附录 10、算法错误码列表

2.1.2 task 文件描述

背景

行走任务表示从起点到业务终点，行走任务有唯一的 ID 成为 taskId。

每个任务拆分为多段任务下发，每段 4~6 个点，每段都有唯一的表示叫 sliceId。

最后一个段的任务完成后，任务完成。

下面看一个完整任务在 task 文件中的打印如下：

```
08/16 16:20:08.812 taskId:R_6E05972PA4000051:051105061597566193_RT_05110506, sliceId:algo1_856_93953
08/16 16:20:08.812   detail: new append road from 05220065, callback
08/16 16:20:08.814   waypoint: type 1 dm:05110688 (29.325 0.800 90.00 1.00 50.00) syncRotate:0
08/16 16:20:08.814   waypoint: type 1 dm:05100062 (29.325 1.500 90.00 1.00 50.00) syncRotate:0
08/16 16:20:08.926   waypoint: type 1 dm:05110507 (29.325 3.060 90.00 1.00 50.00) syncRotate:0
08/16 16:20:08.926   waypoint: type 1 dm:05110508 (29.325 4.620 90.00 1.00 50.00) syncRotate:0
08/16 16:20:08.927   waypoint: type 3 dm:05110508 (29.325 4.620 180.00 1.00 50.00) syncRotate:0

08/16 16:20:11.296 taskId:R_6E05972PA4000051:051105061597566193_RT_05110506, sliceId:algo1_856_93960
08/16 16:20:11.296   detail: old append end from 05110508, callback
08/16 16:20:11.297   waypoint: type 1 dm:05110506 (28.175 4.620 180.00 1.00 50.00) syncRotate:0
08/16 16:20:11.297   waypoint: type 4 dm:05110506 (28.175 4.620 0.00 1.00 50.00) syncRotate:0
08/16 16:20:26.698   result: 0 taskId:R_6E05972PA4000051:051105061597566193_RT_05110506, sliceId:algo1_856_93960
```

第一行 `08/16 16:20:08.812 taskId:R_6E05972PA4000051:051105061597566193_RT_05110506, sliceId:algo1_856_93953`

前面两列表示收到这条任务的时间，注意是收到的时间不是执行时间。

第三列 taskId：后的 R_6E05972PA4000051:051105061597566193_RT_05110506 是 taskId，

一般反馈问题时都会说哪条任务，就是这个 taskID。

第四列 sliceId：后的是 sliceId，这个反馈问题时一般用不到，因为平台上没有显示，是研发内部使用的。

第二行 `detail: new append road from 05220065, callback`

new 表示这是任务的第一段，是 old 表示不是第一段；

append 表示是拼接任务，是 cover 表示是覆盖任务；

road 表示不是最后一段，是 end 表示是最后一段；

from xxxx，这里的 xxxx 表示本段拼接任务起点，只有第一段可以是 NULL，不是 NULL 则必须和上一段的终点相同，否则导致路径拼接失败，拼接失败详细描述见。

第三行 `waypoint: type 1 dm:05110688 (29.325 0.800 90.00 1.00 50.00) syncRotate:0`

type 1，这里的 1 表示前进，也可以其它值，完整列表和对应含义如下：

1	前进
2	后退
3	左转
4	右转
5	等待
6	弧线
10	自由上轨请求
11	自由上轨许可

dm:xxxx，表示本次路径目标点。Xxx 很少为 NULL，为 NULL 表示这个点不在地图中，或者充电点、识别后的点、货架点、货架调整等特殊情况，如果成片出现 NULL 必是 bug，具体见 4.2 平台下发的路径都是 NULL。

(x，y，angle，vl，va)，目标点的坐标和到目标点的线速度和角速度，线速度和角速度为 0 表示设备按照自己最大能力来，但不能是负数。

sycnRotate:x，x 为 0 转弯后负载和设备相对关系不变，1 表示设备转后对外负载朝下不变，也就是我们平时说的转盘同步。

最后一行 `result: 0 taskId:R_6E05972PA4000051:051105061597566193_RT_05110506, sliceId:algo1_856_93960`

result:x，x 为 0 表示正常完成，其余表示失败，错误码见附录 1、导航相关错误码。

taskId：哪条任务的结果

sliceId：哪条段任务的结果，由于一个任务有很多段任务，一般只有最后一段才会有 result 返回，除非中途出错结束、下发取消、断网导致下发任务完成了还没收到后续任务等情况，这种时候 sliceId 提示最后路径接收收到的那段。

2.2 导航错误码

AGV 的的错误码都是四位，导航相关的错误码是 0x30xx、0x32xx、0x34xx，导航相关的所有错误码列表见附录 1、导航相关错误码，很多异常平台都会有提示，这里介绍一些最常见的几个，应该能够涵盖一半以上问题。

脱轨错误码 0x3207

所有现场都应该见过平台提示导航异常，大部分情况下都是由于设备上报 3207 错误码导致，我们内部称 3207 导致的导航异常为脱轨，对于脱轨问题的详细描述见 3.1 脱轨。

路径拒绝错误码 0x3202

现场经常遇到下发了任务（绿灯）但是设备不走，其有一半概率是平台下发但设备拒绝行走，这中间大概率是导航路径拒绝，对于路径拒绝的详细描述见 3.2 路径拒绝。

遇障错误码 0x3001

这个是现场遇到问题的一大类。遇障本身不一定是问题，但是经常遇到没有障碍但是报遇障，或者虽然有障碍但是不应该停这些问题，部分是实施现场可以规避的，有些却需要研发更细致的分析，对于遇障详细描述见 3.3 遇障。

驱动器不响应错误码 0x3206

这个不常见但每个月都有。大部分情况下驱动器异常会上报错误码，这样平台和设备都不会在故障期间收到行走任务，但是有概率设备本身未检测出异常，导致下发行走但是设备不走， 如果发现这个错误码，在排除是按下急停这种真的不能动的，可直接联系研发。

自由上轨失败错误码 0x3214

这个在平台上提示也是导航异常，特点是进入了设备认为的死路无法自己完成回到轨道的情况，一半出现在墙角、转弯处、货架区、充电桩附近这种行走受限的情况，遇到这种情况，人工移动回轨道上，然后上报研发分析，大概率是问题。

二维码施工后试运行常见错误码

二维码施工后，正式运行前可以尝试全场行走，行走过程设备可能检测到施工的角度或者距离误差过大的情况，比较常见的是提示角度误差过大错误码 0x300c、二维码间距误差过大错误码 0x3011 两种，如果发现施工后设备突然停了，错误码是这两者之一，说明是施工问题，关于二维码施工，相关内容比较复杂，详细描述见五、二维码施工问题。

2.3 发送足够的日志即可

现场经常会发送整个日志文件给研发，实际大部分情况下，我们只需要其中的 10%即可，现场最好在发送日志前，删除明显不需要的日志，比如当前出的问题，前几天的一半日志是多余的，发送前先去掉前几天的日志，当然如果传输手段方便或者和网速可以满足时之前发整个即可。

三、行走类问题

3.1 脱轨

脱轨的判定标准非常明确，出现如下任意一个情况即为脱轨：

1、 连续 N 个二维码没扫到，
2、 终点二维码没扫到
3、 转弯点二维码没扫到
4、 激光二维码切换点没扫到

5、 行走过程角度偏差过大（默认值 0.2 弧度，也就是 11.5 度）
6、 行走过程左右偏差过大（默认值是 0.2m）
7、 转弯过程坐标偏差过大（默认 0.2m）
前三点是二维码导航特有，第四点时激光二维码融合设备特有，后两点是通用的。
第一点中的 N，可在设备执行如下命令查看：

rosparam get /nav_monitor_node/yaw_code_2d_num

没有就是默认值，激光二维码融合设备默认为 2，纯二维码默认为 3，少数场地会临时修改参数。
第五、六、七点的过大说明有意味着有阈值，分别在设备执行：

```
rosparam get /nav_monitor_node/straight_theta_err        # 行走左右角度偏差
rosparam get /nav_monitor_node/straight_position_err     # 行走左右坐标偏差
rosparam get /nav_monitor_node/turn_err                  # 旋转坐标偏差
```

没有就是默认值，当前分别为 0.2 弧度（11.5 度），0.2 米，0.2 米。
当前现场很多时候为了临时规避问题会放大判断阈值，减少异常上报，具体操作方法见 3.1.3 修改脱轨阈值。

3.1.1 脱轨原因确认

现场通过平台提示导航异常作为出现脱轨的标记，实际导航异常不一定是脱轨。所以建议出现导航异常后，nav-parse 查看 task 确定任务失败的错误码，提示 result: 3207 才能确认是脱轨。如下就是一个真实的脱轨：

```
08/23 12:46:45.227 taskId:1@37245@179343_05110480, sliceId:algo1_562_17152
08/23 12:46:45.227   detail: old append road from 05110439, callback
08/23 12:46:45.233   waypoint: type 1 dm:05110440 (29.325 29.580 90.00 1.00 90.00) syncRotate:1
08/23 12:46:45.233   waypoint: type 1 dm:05110441 (29.325 31.140 90.00 1.00 90.00) syncRotate:1
08/23 12:46:45.233   waypoint: type 1 dm:05220009 (29.325 32.700 90.00 1.00 90.00) syncRotate:1
08/23 12:46:45.233   waypoint: type 1 dm:05220029 (29.325 34.260 90.00 1.00 90.00) syncRotate:1
08/23 12:46:45.234   waypoint: type 4 dm:05220029 (29.325 34.260 0.00 1.00 90.00) syncRotate:1

08/23 12:46:51.623 taskId:1@37245@179343_05110480, sliceId:algo1_562_17156
08/23 12:46:51.623   detail: old append road from 05220029, callback
08/23 12:46:51.624   waypoint: type 1 dm:05220030 (30.475 34.260 0.00 1.00 90.00) syncRotate:1
08/23 12:46:51.625   waypoint: type 1 dm:05220031 (31.625 34.260 0.00 1.00 90.00) syncRotate:1
08/23 12:46:51.625   waypoint: type 1 dm:05220032 (32.775 34.260 0.00 1.00 90.00) syncRotate:1
08/23 12:46:56.995 cancel 0x3207
08/23 12:46:57.305   result: 3207 taskId:1@37245@179343_05110480, sliceId:algo1_562_17156
```

那么如何确认脱轨原因呢？
很遗憾，当前没有精准的方法，需要一些经验，原因见下面的解释。

> 由于坐标波动和系统卡顿，设备并不能精确知道当前在执行下发的哪一段。
> 比如 A->B->C 的任务，当程序认为执行 B->C 时，突然扫到 B 码，如果坐标往跳回 20cm（贴码误差），那么程序认为在 B->C 扫到码后发现实际 A->B 还没完成，人知道当前路径实际还是在 A->B，但考虑到系统的卡顿，这个在程序上处理非常复杂。同理也存在认为在 A->B，实际在 B->C 的情况。
> 另外当定位更新后，程序中不同的模块收到更新有先后，部分逻辑会认为自己在 A->B，部分认为在 B->C，因此当前判断脱轨是要确认当前坐标在当前认为的那段及其前后一段都算脱轨，才当做脱轨。

脱轨原因分析要结合定位和任务看，直接看 temp_nav.log 文件这次脱轨前相关打印为：

```
|T:08/23 12:46:56.965|L:Info|M:[mov]|m: pv: 0.200 -0.171
|T:08/23 12:46:56.975|L:Info|M:[loc]|m: 29.293 33.448 101.4 0.20 9.0 -9.2E
|T:08/23 12:46:56.995|L:Info|M:[mon]|m: straight angle diff > 11.51, dest:90.00, cur:101.51
|T:08/23 12:46:56.995|L:Info|M:[mon]|m: turn out 0.810m, exp(29.325, 34.260), cur(29.293, 33.450)
|T:08/23 12:46:56.995|L:Info|M:[mon]|m: crosswise 0.75m away from the road36
|T:08/23 12:46:56.995|L:Error|M:[mon]|m: fail: out of rail
|T:08/23 12:46:56.995|L:Error|M:[mon]|m: fail: internal cancel task reason 0x3207
|T:08/23 12:46:56.995|L:Info|M:[mon]|m: movebaseControl: cmd 2
```

从[loc]|m: 29.293 33.448 101.4 0.20 9.0 -9.2E 可以知道当前坐标和角度为（29.293，22.448,101.4），这个坐标在 task 上往上找，大致在这如下 09->29 的中间：

```
08/23 12:46:45.233   waypoint: type 1 dm:05110441 (29.325 31.140 90.00 1.00 90.00) syncRotate:1
08/23 12:46:45.233   waypoint: type 1 dm:05220009 (29.325 32.700 90.00 1.00 90.00) syncRotate:1
08/23 12:46:45.233   waypoint: type 1 dm:05220029 (29.325 34.260 90.00 1.00 90.00) syncRotate:1
08/23 12:46:45.234   waypoint: type 4 dm:05220029 (29.325 34.260 0.00 1.00 90.00) syncRotate:1
```

然后看到 straight angle diff > 11.51, dest:90.00, cur:101.51，意思是前进的过程中，角度偏差超过 11.5 度，的确这个路径在 90 度方向走，当前 101.5 度超过预期 11.5 度，算脱轨。

当前段会当做脱轨，前面提到的程序还会一次判断后一段和前一段（第一段没有前一段，最后一段没有后一段）。

后一段是转弯（type 4），判断时提示：

turn out 0.810m, exp(29.325, 34.260), cur(29.293, 33.450)，

意思是按照当前是转弯那段判断，当前坐标和预期距离 0.81 米，算脱轨。

前一段是直线，crosswise 0.75m away from the road36 表示当前坐标和这线段最近距离有 0.75 米（离 05220009 点），算脱轨

至此，不管按照当前段、后一段、前一段都满足脱轨条件，那么当前设备脱轨，人很容易知道这个设备是在走 05220009 到 05220029 过程脱轨了。

通过上面的例子，应该可以看出脱轨判断是联合判断，最后由人得出脱轨的具体原因，不同情况的 temp_nav.log 中的打印关键字为：

连续 N 个没扫到	line derailed
终点二维码没扫到	movebase ret: 0x3207
转弯点二维码没扫到	turn derailed
激光二维码切换点没扫到	laser2dm switch code miss derail
行走过程角度偏差过大	straight angle diff > 11.51, dest:90.00, cur:101.51
行走过程左右偏差过大	crosswise 0.75m away from the road36
转弯过程坐标偏差过大	turn out 0.810m, exp(29.325, 34.260), cur(29.293, 33.450)

区分原因后，如果是二维码导致的脱轨，由于二维码脱轨会记录过程的录像，因此查录像大部分情况下就知道原因了，预计 9 月后会提供可以查看在导航脱轨最后码的功能，除此之外都让研发处理。

3.1.2 查看脱轨录像

二维码设备脱轨建议现场人员直接查看二维码脱轨录像。脱轨前我们会存放 10 秒以上的录像。录像存放地址为/var/robot/log/nav/bag/xx-yy/derail/derail_aa.bb.cc.avi，其中的 xx-yy 是月和日期，aa.bb.cc 是时间，比如/var/robot/log/nav/bag/08-16/derail/derail_14.43.05.avi，就是 8 月 16 日 14 点 43 分 05 秒上报的脱轨的录像。

设备上使用 ssz /var/robot/log/nav/bag/08-16/derail/derail_14.43.05.avi 命令将录像拷贝到电脑，然后用视频播放器观看。播放器建议使用 smplayer，因为可以使用<和>逐帧查看，

防止播放太快了一闪而过。

如果不会看导航日志，只要看最后的那个码是否有明显的遮挡、污损、反光、模糊，如果有就解决然后重试，如果没有直接求助研发。一些实际场景遇到的不识别情况如下：

图 3.1 严重反光不识别

图 3.2 遮挡导致不识别

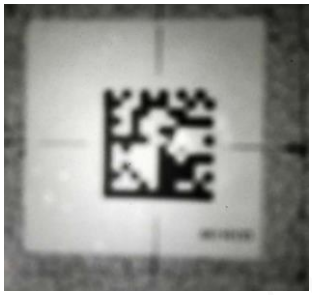


图 3.3 模糊导致不识别

3.1.3 修改脱轨阈值

3.2 路径拒绝

当前经常有反馈下发任务设备不走问题，首先要区分导航是否收到了任务，这个通过 task 文件查看，当前一大半的不走是电机未初始化的问题导致的，剩下的才是路径拒绝。

路径拒绝的原因多样，表现为下发了任务设备不动，从 nav-parse 后的 task 搜 3202，有一个说明拒绝了一次，类似如下：

```
08/22 21:01:59.626 taskId:1@37106@178649_05110092, sliceId:algo1_562_10028
08/22 21:01:59.626   detail: old append road from 05000390, callback
08/22 21:01:59.626   task result: 3202
08/22 21:02:05.663   result: 3203 taskId:1@37106@178649_05110092, sliceId:algo1_562_10027
```

路径拒绝的原因很多，部分导致设备不走，部分只是当前不走，下次就正常了。现场排查的肯定都是一致不走的问题，大部分都是拼接失败和不符合规则。

3.2.1 路径拼接失败

拼接失败是现场不走的常见原因。本质是本次的起点和上次的终点连不上，在 exception 中会有类似如下错误提示：

add slice fail, new start 00000050 but previous end 00000036

意思是本次从 00000050 开始拼接，但是上一段结尾是 00000036，拼接失败。出现这个问题的原因有两个，一个是平台下发错误，一个是设备自己收到任务不报错但就是不走。这个大部分是导航 bug，出现了找研发分析。

由于这个问题出现不容易被观察到，实际每周都有出现。

3.2.2 路径不符合规则

详细的路径规范见路径协议文档，比较复杂，现场出现大部分是自行修改默认参数后引入，待补充。

对策 1：修改本地判断阈值
放大脱轨阈值

3.3 遇障

哪些是预期外的遇障，
判断是什么传感器导致的遇障。

3.3.1 激光没遇障但是提示一直遇障排查方法

3.3.2 TOF 一直没遇障但是提示一直遇障排查方法

3.4 如何分析二维码施工误差

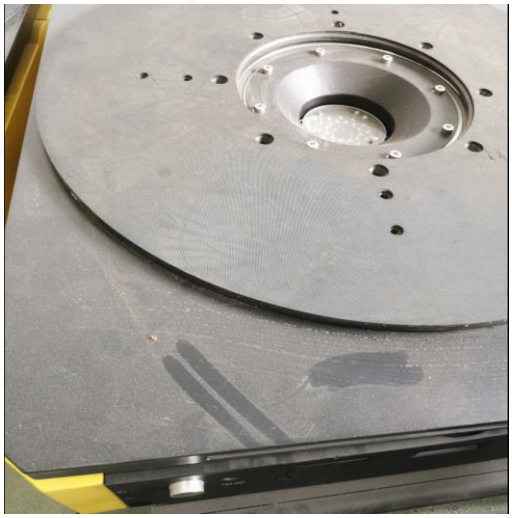
待补充。

四、其他类问题

4.1 激光导航脏

在 20 年上半年的所有版本在检测到导航激光脏了都会上报提示，提示内容是导航激光故障，这个本身定义时是作为提示，并不算故障，出现这个后，需要进行激光的清洁。

回顾我每次到现场，我都看到设备比较脏，脏本身有碍观瞻，同时也会影响相机和激光，所以建议到每个现场，如果发现设备较脏，都可以先进行外观的清理。北阳导航激光相对娇贵，需要在吹落大的颗粒，然后使用眼镜布擦拭。



导航激光明显脏了会影响定位效果，导致行走偏离轨道遇障。当前尚未出现过脏直接导

致遇障的问题。

4.2 平台下发的路径都是 NULL

二维码设备到现场后，初期行走有时会遇到走一段路必定位丢失的问题，或者明显走过头问题，本质是一样的，就是地图配置点类型错误导致。排查方法如下：

查看 nav-parse 解析的 task，如果出问题时间前后大量路径码值为 NULL 说明是这个问题，比如如下：

```
08/13 17:01:00.237 taskId:R_D100-7512:000010441597309326_RT_00001044, sliceId:algo12_885_9
08/13 17:01:00.237   detail: new append end from NULL, callback
08/13 17:01:00.237   waypoint: type 1 dm:NULL (11.420 4.150 90.00 0.50 60.00) syncRotate:0
08/13 17:01:00.237   waypoint: type 1 dm:NULL (11.420 5.700 90.00 0.50 60.00) syncRotate:0
08/13 17:01:00.237   waypoint: type 1 dm:NULL (11.420 6.000 90.00 0.50 60.00) syncRotate:0
08/13 17:01:00.237   waypoint: type 6 dm:NULL (10.500 6.650 180.00 0.30 60.00) syncRotate:0
08/13 17:01:14.815   result: 0 taskId:R_D100-7512:000010441597309326_RT_00001044, sliceId:algo12_885_9
```

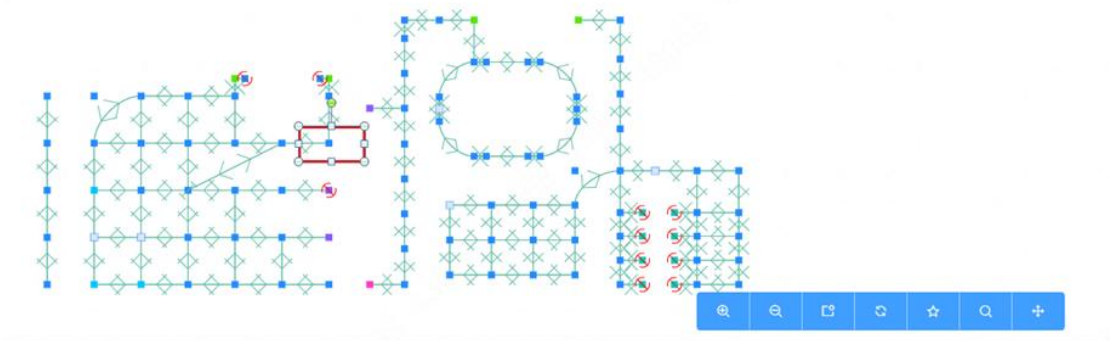
可以看到 type 1 dm:NULL 很多，由于平台下发的路径不可能连续是 NULL，这种情况就是配置错误问题。

此时要确定上面这些任务点（11.42，4.15），（11.42，5.7），（11.42，6.0）是否在地图上，如果在说明没有配为导航标记点，配置步骤如下：

步骤一、使用地图编辑工具，导入地图

步骤二、点击预览地图

3、点击待确认点



4、找到待确认点的导航标识点配置

5、确认导航标识点配置，如果是二维码点，这个一定要选择是

4.3 地图加载失败问题

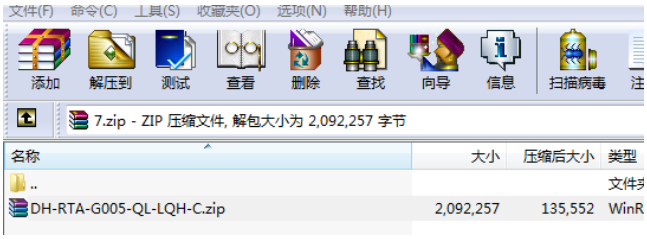
这个都是出现在进场第一天，原因大分为如下几种：

1、 设备无法下载地图

这个确认所在场地是否开启了相应端口，待补充查看方法

2、 地图名称错误

最新的（20 年 7 月后）地图格式为压缩包里有一个或者多个“设备型号.zip”，一个场地有几种设备就几个，大部分是一个，如下：



表示这个地图支持只设备型号 DH-RTA-G005_QL_LQH-C，设备型号查看方法见附录 9、查看设备型号。“设备型号.zip”里最新（20 年 7 月后）格式如下：

其中比原先多了 x_reloc.pgm 和 x_reloc.yaml 两个文件，这个是用于重定位优化的，有这个问题后，重定位正常不超过 3 秒完成，没这个的可能 10~30 秒。

3、 地图内容错误

内容错误大部分都是上述中的 topo.json 不对，当前似乎地图编辑工具没有正式的版本，现场的版本可能早可能晚，导致一些兼容性问题，这个一般通过 nav-parse 后看 cat exception 可以看出来，不确定的发加载地图失败时的 exception 给研发。

4.4 重定位失败或成功后定位又丢失问题

现场重定位失败也很常见，原因有如下几个：

1、 没有地图或者地图不对

查看 nav-parse 后的 exception，是否有大量的 wait map 打印，当设备没有地图时这个打印会没 20 秒打印因此，如果重定位失败，且有大量这个打印就是没有地图或者地图不对，回到4.3 地图加载失败问题处理。

2、 地图变化大导致定位后失败

如果给的位置误差很大，容易导致重定位失败，重定位操作见附录 8、激光重定位操作，如果大致知道位置就手动重定位，看是否成功，凡是重定位后 o 命令看到坐标为 true 了，那么重定位就是成功的。

有时重定位成功后，坐标是错误的，导致走一小段定位又丢失。

此时需要重新定位，给的目标坐标要尽量接近真实的坐标，所以点击错误的位置大概率导致重定位成功但是坐标明显不对问题。

五、二维码施工规范

二维码导航根据识别地上的码确定车体本身的位置调整行走，所以施工的好坏直接影响设备行走精度。但二维码施工码是人一个个贴出来的，费力不说还一定有误差，特别是高精度场景和全场地二维码导航的场景，施工相当困难。

需要强调的是，二维码施工是体力活更是技术活，高精度场景的施工更是非常高难度的

技术活，它需要足够的经验、智慧、耐性和协作，当前没有任何人可以轻松完成真实场景中的施工，在成熟流程和方法出现前，任何人也没有指手画脚指责施工误差的权利。

这里我试图结合个人经验和二维码导航实现原理，提供一个复杂但可复制的二维码施工流程，希望能给现场一些参考和帮助。

相比当前，这个完整流程的二维码施工步骤势必增加施工的工作量，所以具体操作现场人员自己把握，但是建议先理解这个复杂流程的背后原因，毕竟返工我是见得多了。

货架施工由于是相互独立的，现场自行操作，个人建议操作见附录 4、货架二维码施工。

二维码施工的完整工具列表：

激光水平仪、码和膜、墨斗（墨汁）、细笔（铅笔）、5m（3m）卷尺、直尺。

5.1 二维码施工基础

之前大部分场景的施工精度要求并不高，因为客户场景的要求不高，所以虽然也抱怨施工过程的繁琐，好在调整调整总是能满足的（是的，高精度场景你可能永远调整不好）。

在描述复杂的施工前，我们先介绍一些最基础的操作。

5.1.1 二维码贴纸

二维码是一种通过行列表达信息的一种方式，有非常多的类型，我们最常见的手机支付的二维码叫 QR 码，当前我们 AGV 使用的二维码叫 DM 码。当前大部分 AGV 厂家使用 DM 码做二维码导航，原因是 DM 码的识别速度更快。

图 5.1 QR 码

图 5.2 DM 码

尺寸和颜色

现场使用的二维码贴纸默认长宽 7.5cm，二维码大小 3.5cm，现场可以根据客户要求减小二维码白边的大小，特殊情况可以减少二维码大小到 3cm（需告知研发调整配置）。

如果是较为清洁的场地，也可以选择合适底色的材料，不一定非要白色，（问实施要下材料的链接），后续建议使用默认 5.5cm，不同颜色的纸张，以适配客户的地面颜色。

打印要求

打印后，要保证边上四条基准线居中贯穿二维码，部分场地出现四条线和码明显不在码中间的情况，这个会导致定位出现问题，对于高精度场景是致命的。

打印机多少有些误差，但对整体偏移的（比如整体上移）即使误差较大也不影响。比如下图，从辅助线和边缘的情况可以看出整体朝上 2mm，往右 1mm 左右，由于线还是贯穿二维码中心所以不影响。影响的是如右图的这种，仔细看会发现打印总体偏了一格，施工时人是对着辅助线贴码，这样就贴偏了，实际也能看出码整体在路径的左边。

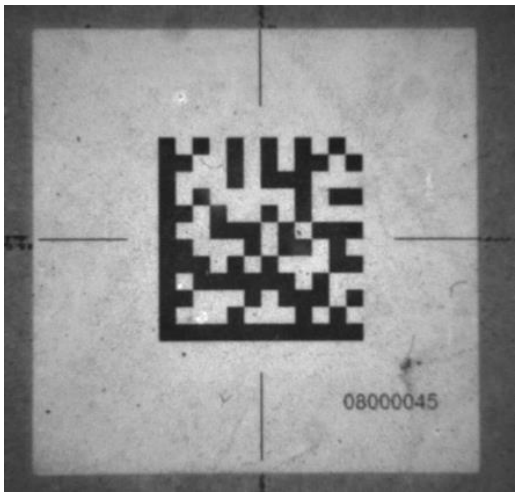


图 5.3

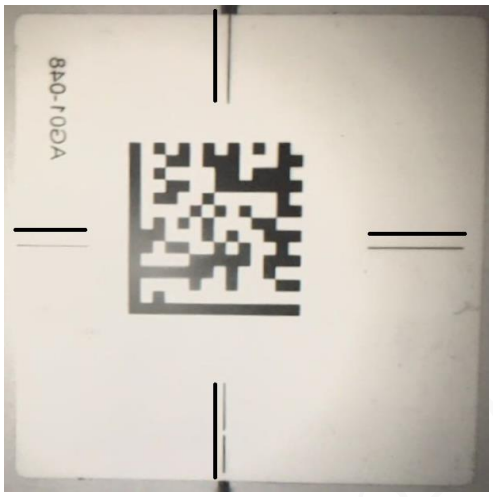


图 5.4

码的内容

细心比较会发现上面两个二维码的密集程度不同，前者是 12x12 格，后者是 14x14 格，理论上格数越少识别耗时越小，精度也会更高。10x10 的码能已经表达 6 位数字，对于我们的使用场景肯定够了，所以如果可以，后续可以考虑全部改为用 6 位数字的二维码。最早定 8 位是预期前两位表示场地号，后 6 位表示数字，这样扫到码就知道当前在哪个区域，当前不是这么实现的，所以 6 位就够了。

5.1.2 二维码间距

当前很多场地施工的二维码间距是 1.35 米，这 1.35 的来源真的是个意外，那具体间距多少合适呢？

实际上，间距越大施工要求越高误差越大，操作上最好不超过 1.5 米。纯粹为了锻炼提高施工技巧的，可以尝试间距 3 米的施工，大部分情况是可以做到的。但你会发现过大的间距反而会花费更多的时间且显著降低整体运行的稳定性。

至于贴得密集是否会提高精度，当然是，理论上相同的施工误差下，间距越小定位精度越高。

所以对于间距，研发的建议是（0~1.5 米），现场间距在 1~1.5 米内都是合适的，具体看现场情况和施工量大小，精度要求高的贴密一些。

5.1.3 在地上画出直线

初中数学课上，我们用直尺和笔在纸上划直线，但是二位施工遇到的情况是线几米甚至上百米长，去哪找这么长的直尺呢？实际操作上，我们采用墨斗这种古老的工具。

公司当前使用的墨斗是 15 米或者 30 米，只要确定了两端，两个人配合就可以在地面弹出直线。

墨斗操作建议多试验，掌握好墨水、拉线力度和两端位置，尽量保证目标路径上是清晰的黑线。

实际操作中，为了后续贴码的方便，场地在弹墨线前要清扫一次，因为很多灰尘的地面码贴不牢。

5.1.4 贴一条 10 米的直线

假定地面上已经有了 10 米长的墨线痕迹，且二维码间距 1 米，那么如何贴码呢？

贴码本身的误差有前后、左右、旋转三个维度，由于只有一条线，所以还需要用尺子量出每个点中心的位置，然后画出一条垂直方向的线（注意不要使用很粗的那种白板笔）。相对而言，具体贴码步骤建议拿着两个对角，对准后下手贴住两角，如果有少许偏差，调整后重试，确认无问题后铺平压实，然后贴膜，正常前后左右偏差能贴到 1mm 左右，角度误差 1 度以内。

施工误差的直观感受

这里以 7.5cm 的码为例，描述误差 0.5 度、1 度、2 度是什么样子。

Xxxxx

角度误差 0.5 度影响 0.65mm，1 度的影响 1.3mm，2 度的影响 2.6mm，所以如果线本身较细，超过 1 度的仔细看能看出来，0.5 度的误差基本肉眼不可区分了。

所以，我们前面提到墨斗的使用、笔的使用都是为了让这个线尽可能不超过 1mm，使得施工便于区分角度误差。

误差的影响

假定间距 1 米，角度施工误差的影响如何？

角度误差	左右偏差
0.5 度	8.7mm
1 度	17.4mm
2 度	34.9mm

可见角度误差对于行走的影响是非常大的，所以如果全场精度要求 10mm，即使间距 1 米，这个施工实际也是无法完成的。所以设备行走时只有转弯后或者直线第一段开始，程序才会完全受施工角度误差影响，所以施工尽量将转弯点或者起止点角度贴准，其余行走区的施工要求可以放宽。

从上面理论上施工误差 1 度导致到下个二维码时左右偏差 17.4mm，而下个码施工的左右前后误差一般也都小于 2mm，所以说大部分情况下，角度误差是影响最大的因素，至于部分场地坐标也差几 cm 的，这个就是另外的问题了。

动手测试

这里 10 米，间隔 1 米，一共 11 个码，起点和终点角度误差尽量小，中间点可以尝试故意贴歪 2 度和贴直，可以看到实际效果是没有差别的。

如果把起点和终点贴歪 2 度，可以看到开始行走很歪，但是到第二个码后，调整完也就直了。同时我们可以尝试起点顺时针贴歪 2 度和逆时针贴歪 2 度的区别，实验会发现逆时针歪了，设备就往右偏，也就是设备会顺着你贴的方向走，这个明白了就很好理解。

通过这个实验，如果平时有单行线，中间点的角度不用特别高要求，但是如果是转弯点或者起止点，就需要关注角度误差。

5.1.5 调整角度施工误差

假定当前已经完成施工，而起点的误差 2 度，人也可以看出走得歪了，这时如何调整呢。

可以选择重新贴码然后重试，这个是当前比较常见的操作。为了减少实施的重贴，最新地图工具加入了设置施工角度误差的功能，以这里起点码往逆时针贴歪 2 度为例，可以在地图工具进行设置施工误差角度：

图 5.5

用过地图工具的应该都知道这个页面，页面最下面有个“二维码贴纸角度”就是标识逆时针贴歪的多少度，比如我现在假定认为逆时针贴歪了 1.5 度，那么选中这个框输入 1.5 **然后回车**（开始多实验几次，很容易忘了回车导致没设置），接着点击确定完成设置。

另外，这离不支持负值，所以比如顺时针偏了 1.5 度，这里要输入 358.5 度。

设置完成后保存地图然后导入设备，这个建议学习基本命令，平台导入再到设备实测比较慢，而且据说要 1 分钟以上才生效，这个要确认下，建议直接导入设备验证这个差值。

地图工具生成的地图直接拷贝到设备验证的方法见附录 5、地图工具生成的地图直接拷贝到设备使用步骤。

实际上不看日志谁都不知道角度误差到底多少，所以需要尝试多次。比如开始发现设备前进明显往左偏，这里就尝试设置 1 度看是否好了，如果还明显往左偏，那么尝试 2 度，类似这样，多试几次大致值了就行了。有经验后，可以 0.5 度为单位尝试。

这个实验告诉我们，在线基本贴直的前提下，角度可以通过脚本调整。如果嫌这个过程慢可以直接重新贴码，但还是建议所有实施人员去现场前都熟悉下这个操作。

5.2 实际遇到的二维码施工问题

这里介绍精度要求比较低的场地施工实际遇到的一些问题。

5.2.1 贴一条 100 米的直线

贴 10 米和贴 100 米的差别是什么？

激光水平仪照不了那么远。

现场操作是，一个 100 米的线，经过 4,5 次拼接的，导致线是歪歪扭扭的，所以对于长直线的问题，建议先使用棉线确认长直线的两端，然后在中间比如每 10 米画个点，这样画出 11 个点，然后用激光水平仪照，照的时候尽量能多照到 3、4 个点，那么这个线整体就应该是比较直的。

5.2.2 大场地无法闭合问题

这个问题是一个长方形的房间，贴到最后一个角发现不是直角了，示意图如下：

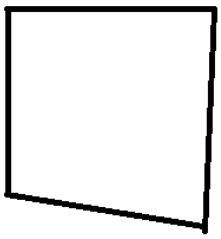


图 5.6

当然实际没有这么夸张，假定每边都是 50 米，你会发现两条平行的边差个 50cm 简直是太容易了，只要一条边偏差 1 度，50 米就会差出 0.87 米，如果要保持每边 50 米，这个四边形是无法闭合的，这时候你怎么调码都是调不好的。

所以对于大场地，一定记得要先确定边界顶点然后再贴码。

比如上图，在大致确定（自己观察确定）四个角点后，要测量每条边的长度，如果超过 10cm 一定继续调整，这个过程需要找多人配合，如果现场只有一两个人是很难操作的，大场地二维码施工最重要的是确认相关区域的边界角点，现场需要 4 个 50 米甚至百米的皮尺。

确定大框架后再进行关键通道的处理，最后在进行细节的填充，千万不要上来就 1.5 米一个贴码。这里由于一定有误差了，误差建议平均掉，不要到结尾了突然就多个 10cm。

大场地的误差如何均匀掉，建议做个图进行计划，否则容易忘记，导致到最后才考虑这个误差。

5.2.3 大场地相交线无法垂直问题

回到上面的问题，假定还是个正方形，通过皮迟我们已经基本确定了四个边的长度，那么如何保证相交线是垂直的呢，实际场地也遇到过相交 89 度，88 度这样情况。

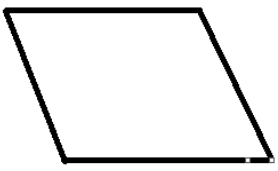


图 5.7

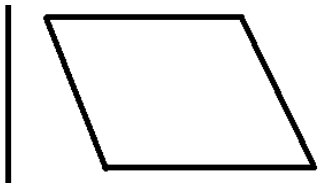


图 5.8

似乎很难是吧，不考虑周围的确很难。但是周围一般总有参考物，比如右图四边形左侧有墙，那么只要测量贴近墙这条边两个点离墙的距离，基本能保证误差不太大。

绝对垂直虽然不可能，但通过参考周围的墙基本都可以做到垂直从而不影响使用。话说回来，我还没有遇到四面都无法参考的情况呢。

有个有意思的问题，如果确定角度是 89 度了，这个码怎么贴？

得分情况，如果是双向路径，当然是一边一半，这样每边都差 0.5 度，这都是最好的选择。如果是单行线，保证作为起点时尽量走直即可，也就是贴码倾向让后续路径平直。

5.2.4 不同设备行走效果不同问题

这个情况一定是研发的问题，说设备的一致性不够。

遇到这种情况，首先确定设备是否经过了标定，具体参考附录 2、朝下相机标定步骤中判断是否完成标定。理论上标定后的设备之间，安装误差一般不超过 2mm，角度误差不超过 0.3 度，应该没有特别明显的差别。这时候可以尝试在现场重新标定。如果依然存在此问题，联系研发解决，很可能是结构问题。

如果是没有经过标定的，那么现场进行重新标定，如果相机内参没有标定的，放弃这台好了，让公司重新发标定好的相机，更换后然后现场重新标定外参。

这个问题无法解决的，联系研发解决。

5.3 二维码施工标准流程

标准的二维码施工规范流程分为确定施工点位和施工精度要求、点位方案验证、关键点位施工、全场施工和验证这几个步骤。

5.3.1 确定点位和施工精度要求

使用二维码的场地分为全部二维码和部分二维码两种，不论哪种情况，都建议先在图纸上做详细计划，确定大体的点位安排。

施工精度的要求在现场应该很快就能确定不同位置设备到点要求，比如对接点位本身要求就是 5mm，那么施工的要求就非常苛刻，有的点位是运行区，但是要过个窄门，那么要求就很高，对于精度要求高的点位，都需要进行细致测量，然后确定点位的排列。

下面以某客户现场图片为例描述如何确定较高精度场景的点位。

图 5.8

目标是一排上下料口，完成施工后如右上角，业务中的场景如右下角。

在开始施工前，已经确定了每个上下料口会并排 3 个货架，每个货架宽 1.00m 长 0.8m，铝合金框架净宽度 3.28m，所以测量后我确定 3.28m 划分为：5cm、1m、9cm、1m、9cm、1m、5cm。这个划分的几个考虑如下：

1、 货架间空白要比边上的大，因为货架前间隔越大，货架碰撞的概率越小

2、 货架和边缘不能太近，那多近有碰撞的风险

假定极限差时设备最大误差 2cm、角度误差 2 度，货架和设备偏差 1cm 和 1 度，那么货架和边缘的距离是 3cm+一半货架长*tan3 度=5cm，所以 5cm 是极限差的情况，如果要确保极限情况不相撞，那么留空应该是 5cm，中间留空 10cm，这个和当前相差 2cm，所以只能是中间的做些牺牲。

至此我们确定了三排码的纵向间隔和位置，接着确定横向位置和间隔。这个过程需要实地验证，所以我们把设备和货架都搬到期望的地方，然后让设备背起，确定设备最后合适停止的位置。

现场要求货架必须在下面这个框架内，框架深度 85cm，货架的宽度是 80cm，实测考虑一些余量，货架的位置是基本确定的，基本就一两 cm 自由度。

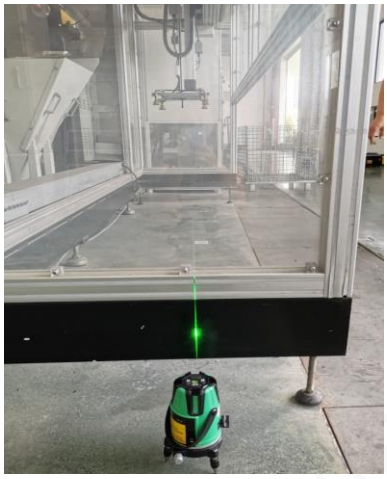


图 5.9

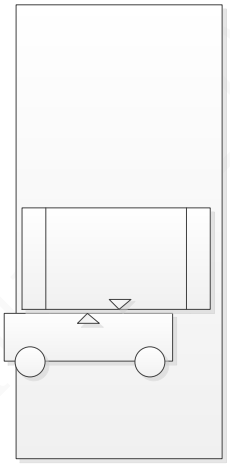


图 5.10

车长（不含触边）84cm，货架边离边缘 3cm，车停在货架中间，必然会撞到黑色挡板，所以车必不举在货架中心，如上面右图所示意。

货架中心确定好离边缘 40+3=43cm，经过实测摸索车和货架中心那个偏差合适，如下是偏差 3cm 时（下左图情况），设备离急停按钮留 1cm（下右图）。所以再加 3cm，确定偏移 6cm，那么设备最后离黑色挡板距离为 43+6=49cm。

图 5.11

图 5.12

这样就确定了第一条横线的位置，也就是下图的最上面一条。

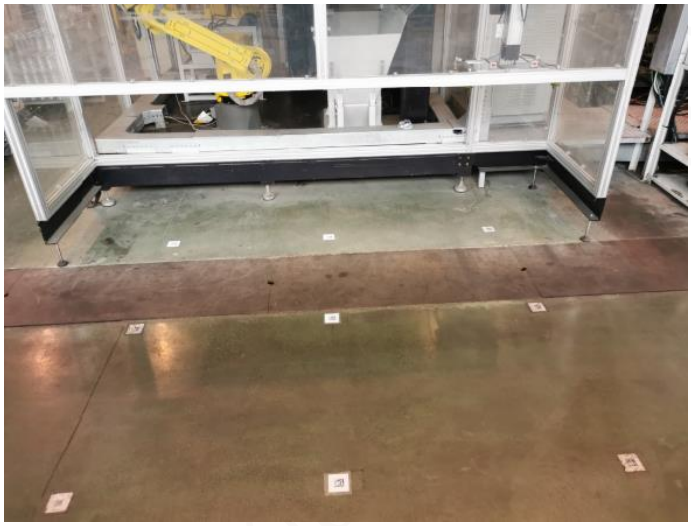


图 5.13

至此，最重要的三个对接点确定，和客户对接结构的相对关系也确定了，接着要确定最外面那排，也就是上图的最下面一排。

这个具体现场确定，确定即可，无特殊考虑。

然后确定中间那排，大体上我们习惯选中间点，这里刚好中间点在铁板上，所以我们选择贴在接近铁板的地面上，距离关系不大。

支持我们确定了所有 9 个点的相对关系，确定点位方案的步骤完成。

5.3.2 点位方案施工方法验证

施工过程基本原则：在关键点位完成地面标记并确认无误前，不要贴码。

对于精度要求低的区域，比如长的运行道路，可以直接施工，但对于类似上面描述的高精度、经过计算的点位方案，在大批量施工前要先确认点位方案的正确性和可行性。一般在首次验证点位方案过程中，操作者会发现一些限制、优化点、遗漏或者错误，然后调整方案，经过多次调整后通过验证，最终输出本场地的点位施工流程，往往对于新情况，点位施工方案要进行多次反复后才会趋于完善。

这里依然以上面场地的例子描述这个点位方案施工方法验证的过程。

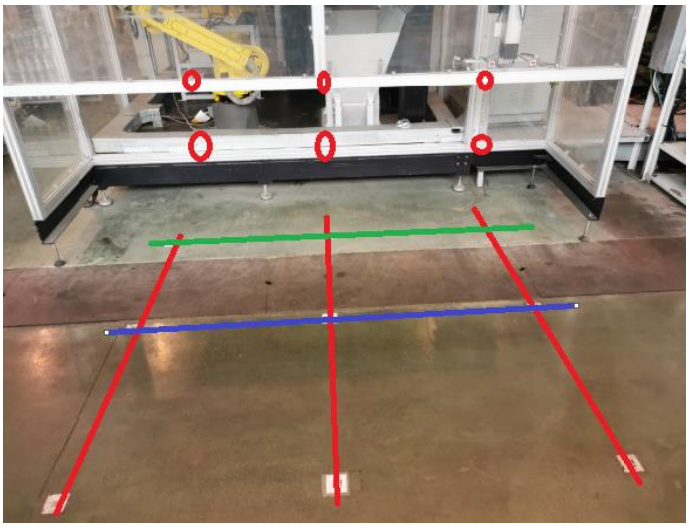


图 5.14

点位方案确定了要施工 9 个码和相对关系，但施工顺序还是很有讲究的。主要原则是可以直接确认的点位先施工，由已知点位推到、测量出的后施工。

由于竖直方向是设备的行走方向且能直接测量，所以先施工图 5.14 中的三条红色线。如何确定红色线的位置呢，图上我们用圆圈标了 6 个点，这些就是我们借助周围环境辅助标点的基准。这个过程要使用激光水平仪对准两个点，然后使用墨斗弹线，如此完成三条红线的绘制。

找标点辅助需要动脑子，不好找时，比如一面墙，可以把设备推到预期的位置，将车尾中心作为辅助标注的基准，然后激光打线经过这个基准点并且保持投影在墙上的线基本竖直即可。（这里要补个图）。

然后施工图上的绿线，这个要求和三条红线垂直且焦点和黑边距离基本相同，这个过程没有特别好的方法，需要人去感受，大体要标准焦点离黑边距离差别不大，且连线大致和原先垂直即可，否则到时候会停的前后不一。

然后施工图上紫色线，可以从绿线和红线的焦点量出来，用激光大致对齐即可。

最外面的线需要事先标注位置，然后将交点作为点位。

注意，在具体实施时会遇到夹角不是直角的情况，这时候要明白最需要保证的是哪个方向的精度，比如上图中，如果绿线和红线不垂直，贴码优先保证和红线一致，因为这个是行走方向。

完成点位标点后，建议使用尺子和笔进行交点的描摹，因为施工过程会有调整，导致地上线可能有好几条，我们要将最终的明确一些，同时也是为了保证下次重新施工时，撕了码能看到清晰的基准线。

描线之后完成贴码。施工时坐标尽量贴准，角度出现两难时优先保证可作为起点的角度精度。

施工完成后需要实车进行测试验证误差，这个需要平台下发任务，然后分析日志，这个过程当前没有图形化的工具。

设置至少一对导航标定点

设备运行一段时间后，轮胎大小会明显变化，这时候导航标定点的作用开始体现。

导航标定点只有空载、匀速通过一段首尾都是导航标定点时才会分析当前预测和实际的差值，从而估计轮子的真实大小，使得后续行走更准。所以要求标定点：

1、 相连的两个

2、 距离准，之间的距离和地图距离一致，间距误差尽量保证 2mm 以内

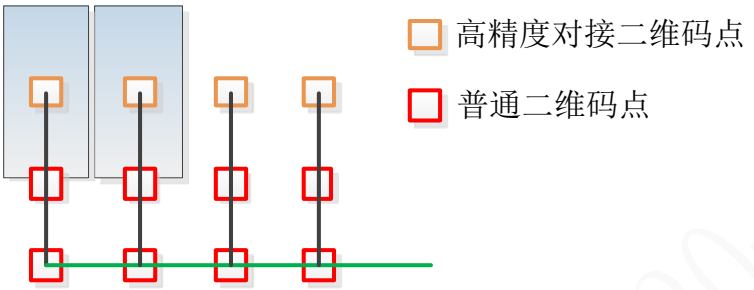
3、 尽量放在长直路的中间

这个标定点作用非常明显，但是也不要设置过多，因为这个要求精准施工，很差的施工设置标定点是个灾难。

高精度施工方法

这个问题的确非常复杂，对于每个场景还是需要多观察，察觉这个场地施工难度的关键。

基本思路还是最高精度的点出发，尽量让高精度点之间少关联。如果必须关联的，需要作为整体思考，举一个现场的例子，一排 N 个相邻的点位都要 5mm 的高精度。



由于现场一直无法达到要求的精度，于是我到现场去分析原因，看到上面四个黄色高精度点，停车后间距很小，所以它们是互相影响的。但是施工时只保证了四个黑线本身是直的，互相不平行，而且当时黑线和绿线不垂直，导致最后车间距非常近。

这种场地重新施工的话，首先标定图上四个黄色高精度对接点的位置，保证它们是一排。

然后使用设备停到目标位置，使用水平确定上述的黑线，弹墨线后再次确认黄色点之间的间距和路线上四个红色点的间距，没问题再施工就好了。

5.3.3 关键点位施工

这里的关键单位是指一些交接点，前面 5.3.2 是为了验证方案可行，可行后就要从细节中抽出来，再看看整体点位的布置。我见过很多现场施工时从一条线一条线施工的，然后到后来就积累了很多的误差，所以正确的方式肯定是先确定框架然后才是补充细节。

这个过程需要多人配合，尝试不同的点位进行测量。比如施工一个长方形，关键定位就是要确定四个顶点，如果这四个顶点的间距很长，一定需要反复调整，保证两边的距离基本一样长，然后角度又大致垂直。

这个过程只要测量，不用真车验证。

5.3.4 全场施工和验证

这个过程就是补充细节，施工过程建议也是折半确定点位，比如一个希望 10 米长的路当前只能是 10.1 米了，那么把中间点当做 5 米的地方，然后补充剩余点位。

完成施工后，完成地图编辑导入设备，进行整个场地的施工验证，具体见 3.6 如何分析二维码施工误差。

5.3.5 二维码施工的一些建议

施工前把地扫干净

相信我，绝对是划算的。

使用笔进行二次描线

现场经常有出现码损坏需要重新施工的情况，所以尽量在码下保留完整的十字，这样下次重新贴时有个明显的参考。这个十字建议使用铅笔或者较细的黑笔，前者用于不容易用水笔化的场景。视频连接。

5.4 激光二维码切换点坐标标定

找到完成相机标定和激光标定的设备，不确定是否标定的查看附录 2、朝下相机标定步骤和附录 3、激光标定步骤确认，具体工具操作见附录 6、激光二维码切换点坐标标定方法。

六、生产类问题

6.1 北阳导航激光

6.1.1 北阳导航激光无法烧入

1、 检查激光配置文件是否可以正常打开，无法正常打开询问相关人员获取相应文件
2、 断电重启设备并且重新打开激光客户端
3、 登陆设备输入命令 rosservice call /close_scan "data: true"
4、 连接数据线、进行烧录处理

6.1.2 各种错误码

错误码 57

1、检查激光板是否烧录固件
2、激光线是否连接准确

错误码 68

开机后静置 1 分钟后断电重启，如不行再进行重试，重试 3 次无改善，联系研发人员

错误码 84-85、B1-C1

使用眼镜布擦拭激光表面，并且观察激光表面是否存在刮痕，如存在明显刮痕则需要更换激光

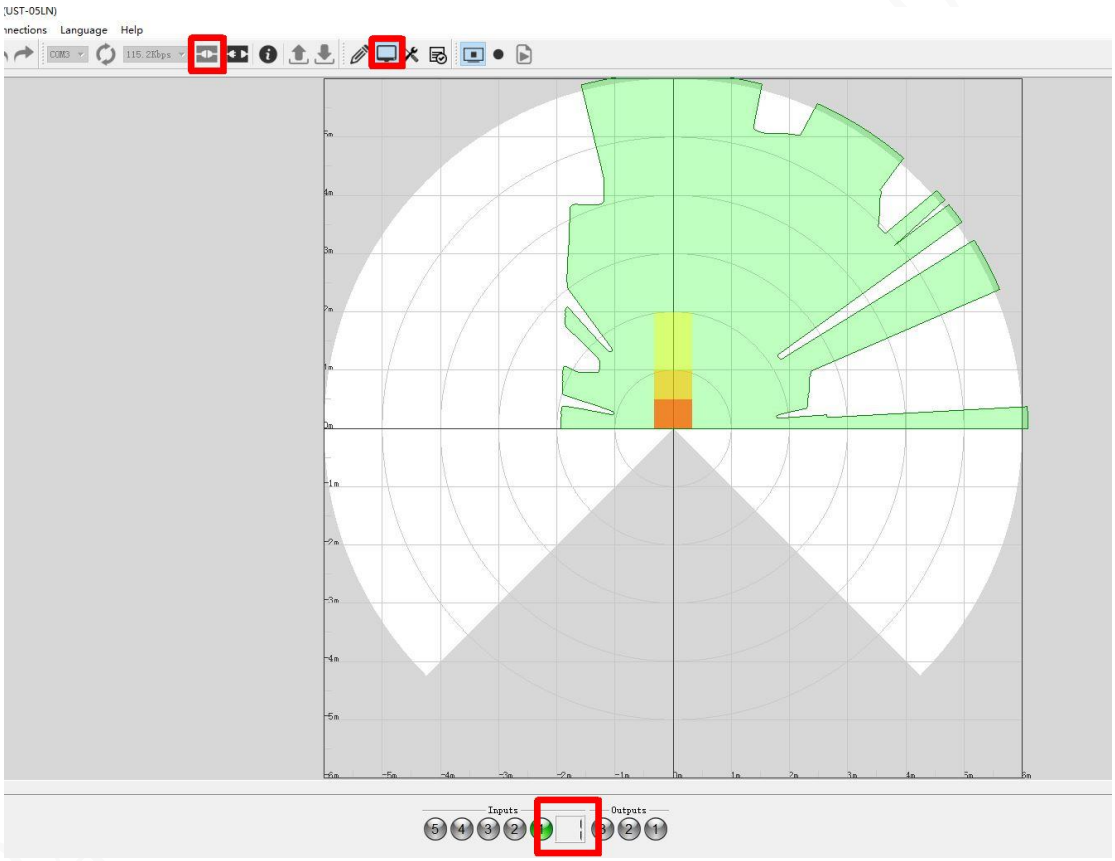
6.2 北阳避障激光

6.2.1 测试避障距离与预期不对应

登陆设备输入命令

```
[10.35.191.132]@/home/dahua# rosservice call /agv_security/safety_id "type: 0
fromInternal: false
id:
- index: 0
  id: 0"
```

连接客户端，点击连接和监控按钮，查看图片下方是否为 1。不为 1，需检查接线和激光板



检查配置文件是否为对应型号文件，不对应则进行重新烧写对应配置

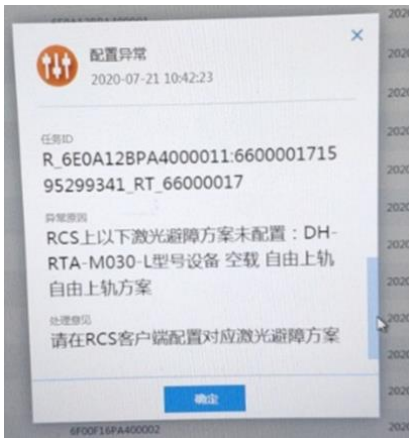
6.3、运行问题

6.3.1 设备运行超出二维码，未报脱轨

详细见 4.2 平台下发的路径都是 NULL。

6.3.2 平台提示激光避障方案未配置

确认下平台避障配置是否正确导入，不确定找平台人员确认



6.3.3 运行过程中经常遇障

联系研发查看

6.4、标定问题

朝下相机标定见附录 2、朝下相机标定步骤，激光标定见附录 3、激光标定步骤。

附录

附录 1、导航相关错误码

```
//0x30 表示安全模块
stopAsUsAbnormal = 0x3000,          //超声波模块异常
stopAsObstacle = 0x3001,            //因为障碍距离过近停止
stopAsCollision = 0x3002,           //因为碰撞停止
decDistanceError = 0x3003,          //设置的减速触发距离小于急停触发距离
stopDistanceError = 0x3004,         //设置的急停触发距离大于减速触发距离
emergencyStop = 0x3005,             //因按下紧急制动按钮而停止
stopAsMotorFail = 0x3006,           //因底层电机相关异常停止
stopAsIMUFail = 0x3007,             //因 IMU 异常停止
stopAsLidarFail = 0x3008,           //因激光异常停止
stopAsIMUAbnormal = 0x3009,         //因 IMU 跳变停止
stopAsEncoderAbnormal = 0x300a,     //因编码器跳变停止
stopAsFallingCheck = 0x300b,        //因检测到潜在跌落可能停止
```

```
    stopAsAngleAbnormal = 0x300c,          //因检测到新二维码和当前预测角度偏差过大停止，防止行走方向错误
    stopAsEncoderDisappear = 0x300d,       //因编码器消失停止
    stopAsIMUDisappear = 0x300e,           //因 IMU 消失停止
    stopAsReLocation = 0x300f,             //因触发重定位导致任务停止
    stopAsTofFail = 0x3010,                //因 TOF 异常停止
    stopAsDmJump = 0x3011,                 //因检测到新二维码导致坐标跳变较多停止
    stopAsSetObstacleAreaFail = 0x3012, //因设置避障失败停止
    stopAsNoSecurityConfig = 0x3013,       //因设备没有自主避障配置停止

    //0x32 表示导航模块
    navYawException = 0x3200,          //偏航
    navTaskExecuting = 0x3201,         //先前任务还在执行中
    navTaskWrong = 0x3202,             //路径不合法
    navTaskCanceled = 0x3203,          //任务被取消
    navTaskUnKownPosition = 0x3204, //导航模块不知道当前位置
    navClearNullTask = 0x3205,         //没任务时下发清除任务
    navRunCmdNoResponse = 0x3206,      //运动命令不响应
    navDerailment = 0x3207,            //脱轨
    navCarNoStop = 0x3208,             //下发任务时小车没有停止
    callCameraStateErr = 0x3209,       //调用二维码相机模块失败
    navUnknowAddType = 0x3210,         //未知任务添加类型
    navLaserNearRange = 0x3211,        //激光数据过近，例如用小盒子将激光罩住
    navNoLaserData = 0x3212,           //无激光数据
    navLaserFarRange = 0x3213,         //激光数据过远
    navUntrack = 0x3214,               //自由上轨失败
    navTaskArcWrong = 0x3215,          //平台下发让设备自己算弧线，设备计算失败
    navTaskSlip = 0x3216,              //设备因为严重打滑停止行走
    navLicenseAbnormal = 0x3217,       //设备 license 异常导致无法运行

    // 0x34 运动执行模块
    callNavServicesErr = 0x3401,           //调用导航模块返回错误
    callPathPlanningErr = 0x3402,          //调用路径规划模块失败
    pathAnomalous = 0x3403,                //规划处错误的路径
    roadblock = 0x3404,                    //有障碍物
    callInitElevatorServiceErr = 0x3405, //调用初始化机械臂模块返回错误
    callInitPtzServiceErr = 0x3406,        //调用初始化相机云台模块返回错误
    robotMotorUninit = 0x3407,             //电机未初始化
    sportsAddTaskFailed = 0x3408,          //添加拼接任务失败
    sportsPathReplanFailed = 0x3409,       //遇障恢复重新规划路径失败
    sportsArcNotReach = 0x340a,            //弧形转弯不可到达
    stopAsModeSwitchFail = 0x3411,         //导航模式自动切换失败

    // 0x35  充电模块
```

dockStartPointWrong = 0x3501, //起点扫不到二维码
dockConnectFailed = 0x3502, //对接失败
dockChargeFailed = 0x3503, //充电失败
dockSrcDstAngleNotEqual = 0x3504, //开始对接前，设备当前角度和目标点角度不一致

// 0x36 货架位姿检测模块
shelfOverShifting = 0x3600, //货架偏移过大
shelfLableNotFound = 0x3601, //货架二维码没找到
shelfCameraNotFound = 0x3602, //货架相机没找到
shelfAngleOverShifting = 0x3603, //货架角度偏移过大
shelfIdentifyFailed = 0x3604, //货架识别算法返回失败
shelfModelUpdateFailed = 0x3605, //货架识别模型更新失败
shelfCurvePathPlanFailed = 0x3606, //货架识别弧形路径规划失败
shelfCurveNavMoveFailed = 0x3607, //货架识别弧形导航移动失败
shelfLaserUrdfParamLoadFailed = 0x3608, //< 货架识别激光外参加载失败
shelfModelPalletAngleParamAbnormal = 0x3609, //< 货架模型托盘角度参数异常

附录 2、朝下相机标定步骤

标定前的准备

标定需要专用的二维码纸带，如果现场没有，可以去打印店打印下面压缩文件的图形内容。

二维码标定码带.rar

打印按照真实大小打印，也就是二维码大小 3.5cm，间距 60cm，一条长 1.3 米。
打印后记得最好覆一层亚光膜，这样可以用的更久一些，否则可能标定多了码就碾断了。
为了减少浪费，一次打印 11 条，多的备用，售中建议常备几条。
。

有标定码带后，找一块平整地面，3 个码朝上贴到地上，贴的时候注意要拉紧铺平，纸带不能有褶皱，然后用透明胶再将纸带贴到地面，注意透明胶不要盖住二维码。

执行标定程序

1、 **确认 license 有效**

使用命令 rosservice call /platform_access/getLicenseInfo "{}"，类似如下显示则为正常

```
algTypesNum: 3
allType: 0
algTypeSupport: []
begintime: "2020-06-18 11:02:20"
days: 99999
state: 1
```

2、 把车推到第一个二维码 **44000001** 上，调用命令 d 确定车中心应该在二维码中心[2cm,2cm]范围内，例如 d 命令输出

details:

-

data: "44000001"

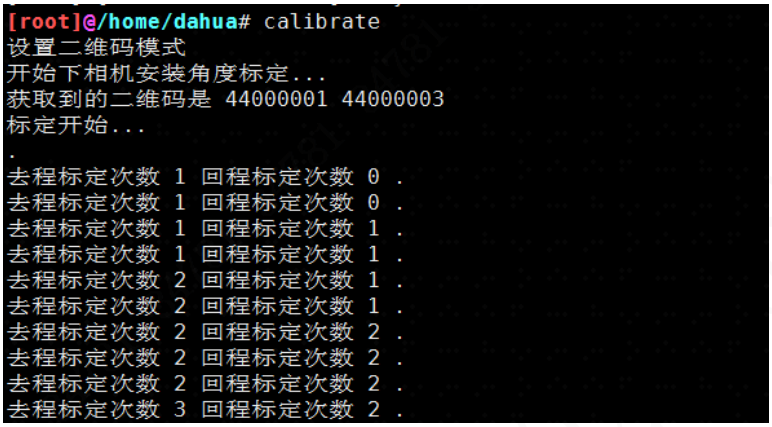
angle: 359.80670995

centerX: -0.000614855925374

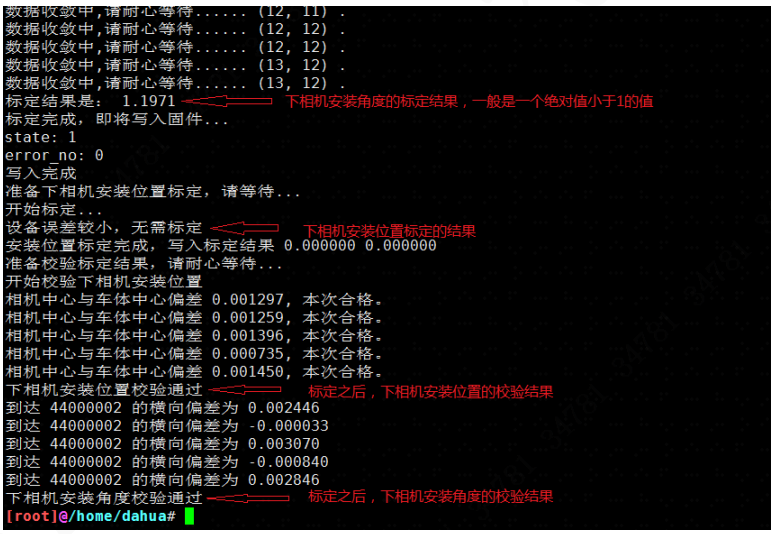
centerY: 0.00117767423808

angleErrorEstimate: 0.325436702371

3、 在任意目录下执行 calibrate 标定程序，车开始在二维码之间来回走，正常 10 分钟内标定程序会执行完成并提示标定是否成功，执行过程提示如下：



标定完成的提示如下：



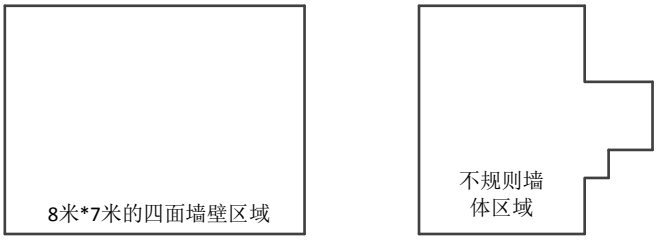
如果标定失败或校验不通过请尝试重新标定，如果重新标定还是不行联系研发人员。

附录 3、激光标定步骤

在设备完成组装、更换激光或者轮子后需要重新标定。

环境要求

1、 环境特征明显（墙壁、柱子）



默认标定的行走路线是 8 字形，大约需要占地 4 米*5 米左右，区域的大小最好在 6 米*6 米以上。稍微大一些有助于提高标定效果，太大也没有必要，甚至需要弄个挡板遮挡下。

2、 标定时环境尽量少变化且激光水平面上极少杂乱无章的物体

设备激光外参标定时需保证环境没有明显变化，比如人不能站在 AGV 前方区域乱走，地牛标定时中人减少走动，标定过程也不允许路线上有障碍。

部分现场难以满足这些要求，比如下面这个墙角东西很乱，标定时就需要临时找了个挡板遮挡下。

3、 标定区域尽量保证地面水平不打滑

4、 标定时如果设备明显翘起则需在小车上放承重物，保证在运行中激光不会因小车加减速导致激光上下晃动。

如下场景就很理想，四周都是白墙，且没有明显的激光高度上的物体。

标定路线

标定路线是设备直行 2m 左弧线掉头，前进 2m 右弧线掉头，前进两 2m 右弧线掉头，前进两米左弧线掉头的“8”字形路径，重复两次完成标定。

标定操作流程

xshell 登录设备，sudo 后，执行如下命令：

/home/dahua/agv_workspace/install/install_nav/lib/navtools/script/laserOffset/laserCalibration.sh

顺利的话，5 分钟左右后在终端会提示“标定成功”。

小场地标定特殊操作

为了适应不同环境，提供小场地的标定路线，具体路线是设备在起点向左向右各旋转一圈，接着前进 2m，然后掉头再前进 2m 回到起点，然后掉头回到初始状态，重复两次完成标定。

xshell 登录设备，sudo 后，执行如下命令：

/home/dahua/agv_workspace/install/install_nav/lib/navtools/script/laserOffset/SmallGroundCalibration.sh

顺利的话，5 分钟左右后在终端会提示“标定成功”。

检查是否完成标定

xshell 登录设备，sudo 后，执行如下命令：

/home/dahua/agv_workspace/install/install_nav/lib/navtools/script/laserOffset/check.py

会提示“设备还未标定激光外参”或者“设备已经完成激光标定”。

附录 4、货架二维码施工

货架码施工分为中心施工和偏移一定位置施工两种。

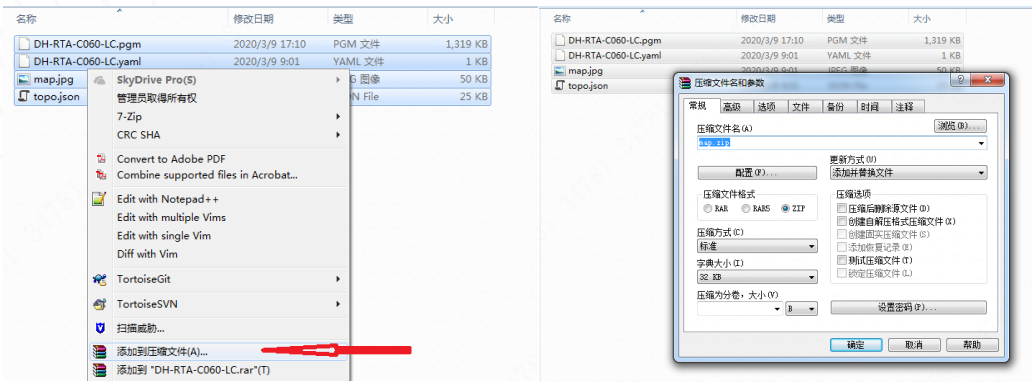
附录 5、地图工具生成的地图直接拷贝到设备使用步骤

1、 地图编辑工具 -> 保存地图，会保存一个 zip 压缩格式的地图文件。这个是平台需要的地图格式，设备使用需要改一下地图的文件打包方式。

1.1、 如果是二维码地图，只需要将 zip 包解压，获取到 json 文件。通过 xshell 将 json 文件拷贝到设备中

1.2、 如果是激光地图或者激光和二维码混合地图，需要将地图编辑工具生成的 zip 解压，在将 pmg、yaml、json 文件直接打成一个 zip 包，不要包含目录，并将新的 zip 包通过 xshell 拷贝的设备中

选中所有地图文件，直接右键打包压缩：



2、 xshell 登录设备，sudo su 后，执行如下命令加载地图

rosservice call /mapsvc/loadMap "mapPath:

- '/home/dahua/map.zip'

autoSetLocation: false"

注：标红部分是地图的全路径，需要根据实际情况填写

3、 对设备进行重定位，具体见附录 8、激光重定位操作

附录 6、激光二维码切换点坐标标定方法

1、 先确保设备在激光定位模式且定位正常，定位模式查看方法见附录 7、确定设备处于什么定位模式

2、 xshell 登录设备，sudo su 后执行：

/home/dahua/agv_workspace/install/install_nav/lib/navtools/script/calibrateDmCoordnate/calibrateDMCodeCoords.py

将设备遥控到激光**二维码切换点**，**注意遥控**尽量模拟实际运行方向，遥控距离要至少 3 米，中间不要有明显的角度调整，当设备扫到二维码后会持续打印切换点在激光地图中的坐标，坐标字段是 x，y，角度

```
[root]@/home/dahua#
[root]@/home/dahua#
[root]@/home/dahua# /home/dahua/agv_workspace/install/install_nav/lib/navtools/script/calibrateDmCoordnate/calibrateDMCodeCoords.py
请将设备推至二维码切换点上...
二维码 44000001 在地图中的坐标: (0.247, -0.191, 1.8)
二维码 44000001 在地图中的坐标: (0.247, -0.191, 1.8)
二维码 44000001 在地图中的坐标: (0.247, -0.191, 1.7)
二维码 44000001 在地图中的坐标: (0.247, -0.191, 1.7)
二维码 44000001 在地图中的坐标: (0.247, -0.191, 1.7)
二维码 44000001 在地图中的坐标: (0.247, -0.191, 1.7)
二维码 44000001 在地图中的坐标: (0.247, -0.191, 1.7)
```

当前从不同方向到达激光二维码切换点坐标会有少许差别，所以标定时请按照实际使用的方向到达该切换点。如果多种方向都有，那么要尝试不同的方向。

不同方向标定结果是有明显差别，此时坐标点需要计算不同方向坐标的均值。

3、 打开地图编辑工具 -> 编辑点位，将**二维码坐标值**填入对应点位的物理坐标和二维码贴纸角度中。

附录 7、确定设备处于什么定位模式

xshell 登录设备，sudo su 后执行：

rosservice call /nav/get_navlocalization_mode "{}"

```
[root]@/home/dahua#
[root]@/home/dahua# rosservice call /nav/get_navlocalization_mode "{}"
mode: 2
success: True
[root]@/home/dahua#
```

mode 含义：

1：里程计模式

2：激光模式

3：二维码模式

附录 8、激光重定位操作

重定位是指我们希望设备在某个范围内匹配它认为最正确的位置，一般在首次进入场地、设备明显坐标错误后操作。

操作建议在客户端进行重定位操作，这里补充下操作说明。

在某些情况下，平台有问题或者没有平台时，可以进行手动重定位，命令为：

rosservice call /nav_monitor/set_init_pose "{pose_x: 18000, pose_y: -3200, pose_theta: 0, width: 0, height: 0, isFromNav: false}"

参数介绍：

pose_x: 设备所在位置 x 坐标的大概值，单位 mm

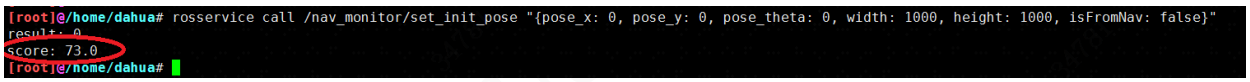
pose_y: 设备所在位置 y 坐标的大概值，单位 mm

pose_theta：不要填写

width、height：在 pose_x、pose_y 为中心的 width * height 的区域匹配位置

所以上面的命令表示在坐标（18.0 米，-3.2 米）附近搜索匹配的坐标。大部分情况下修改 pose_x 和 pose_y 即可，如果坐标不是特别确定，那么设置 width 和 height 字段为 10000，表示在以这个目标点为长宽为 10 米的长方形的框内搜索最佳的匹配坐标，不行加大长宽。

返回结果：下面 score 字段得分低于 30 大概率坐标不行，要调整坐标（pose_x、pose_y）和范围（width、height）重新定位。



附录 9、查看设备型号

客户端的设备管理页面中有设备型号一项，正常不需要修改，做地图压缩包时请务必按照这个型号名称命名。

这个型号来自设备的上报，在设备的型号可以通过如下命令查看设备型号：

roscat agv_config ProductDefinition |grep "DeviceType.*DH"

附录 10、算法错误码列表

```
MOVE_CONTROL_FAILED_IDLE = 0x1001,              // 当前为空闲状态，不允许执行
MOVE_CONTROL_FAILED_EXECUTING = 0x1002,         // 当前为执行中，不允许执行
MOVE_CONTROL_FAILED_NO_2DCODE = 0x1003,         // 执行结束时没有扫到二维码
MOVE_CONTROL_FAILED_UNINIT_MAP = 0x1004,        // 执行任务时未设置地图
```

MOVE_CONTROL_FAILED_ABNORMAL_PARAMS = 0x1005, // 接口入参存在问题

任务相关具体错误码

MOVE_CONTROL_ABNORMAL_CONTROL_POINT = 0x2001, // 控制点异常
MOVE_CONTROL_PATH_UNCONNECTED = 0x2002, // 路径没有连通性
MOVE_CONTROL_NO_PATH_POINT = 0x2003, // 没有路径点
MOVE_CONTROL_NO_TARGET_THETA = 0x2004, // 没有路径点角度
MOVE_CONTROL_NO_TASK_POINT = 0x2005, // 任务中没有路径点
MOVE_CONTROL_OTHER_TASK_ERROR = 0x2006, // 其他任务异常
MOVE_CONTROL_PATH_LATERAL_HIGH = 0x2007, // 小车偏离路径过远
MOVE_CONTROL_PATH_NOT_REACHED = 0x2008, // 路径不可达
MOVE_CONTROL_PATH_TOO_LONG = 0x2009, // 路径过长
MOVE_CONTROL_PATH_ROTATE_ERROR = 0x200a, // 旋转路径异常
MOVE_CONTROL_PATH_CHECKAPPEND_ERROR = 0x200b, // 路径拼接异常

license 相关错误码

MOVE_CONTROL_LICENSE_ERR_BASE = 0x10000, //0x10000,
MOVE_CONTROL_LICENSE_ERR_OUT_OF_MEMORY = 0x10001, //内存错误
MOVE_CONTROL_LICENSE_ERR_INVALID_PARAM = 0x10002, //参数错误
MOVE_CONTROL_LICENSE_ERR_SYSTEM = 0x10003, //系统错误
MOVE_CONTROL_LICENSE_ERR_DATA = 0x10004, //license 数据错误
MOVE_CONTROL_LICENSE_ERR_BIND = 0x10005, //绑定信息错误
MOVE_CONTROL_LICENSE_ERR_TIME = 0x10006, //license 过期
MOVE_CONTROL_LICENSE_ERR_VERSION = 0x10007, //license 版本错误
MOVE_CONTROL_LICENSE_ERR_TYPE = 0x10008, //算法大类不匹配
MOVE_CONTROL_LICENSE_ERR_JSON_LIC = 0x1000a, //json 相关错误
MOVE_CONTROL_LICENSE_ERR_HTTP_GET = 0x1000b, //http get 请求错误
MOVE_CONTROL_LICENSE_ERR_NET_PROTOCOL = 0x1000c, //netProtocol 错误
MOVE_CONTROL_LICENSE_ERR_JSON_CONFIG = 0x1000d, //json 配置错误
MOVE_CONTROL_LICENSE_ERR_NOT_COMPLEMENT = 0x1000e, //没有实现
MOVE_CONTROL_LICENSE_ERR_NO_LICENSE = 0x1000f, //没有 license
MOVE_CONTROL_LICENSE_ERR_LICMODE = 0x10010, //加密模式错误
MOVE_CONTROL_LICENSE_ERR_NOT_FOUND = 0x10011, //没找到
MOVE_CONTROL_LICENSE_ERR_OTHER = 0x10012, //其他加密错误
MOVE_CONTROL_LICENSE_ERR_NUM = 0x10013, //其他加密错误