《MRPT 应用工具》

《MRPT AppTool》

MRPT_V1.0.2

汪若博※译

keyearth@gmail.com

Mobile Robot Programming Toolkit
移动 机器人 编程 工具箱

版本	历史
V1.0.2	建立文档

... 说明 ...

Warning:本文内容仅供参考，为方便 mrpt 开发者学习而整理撰写，不保证内容准确无误，以 mrpt 官网内容为准: www.mrpt.org

Tips:编程语言为 C++语言，适合具有一定 C++/STL 基础的开发者。

... 知行 ...

MRPT 版本 1.0.2

http://www.mrpt.org/list-of-mrpt-apps/

安装完 mrpt 后，在 mrpt/bin 目录下可以找到所有的 mrpt 应用软件，在 mrpt/app 目录下是应用软件的源代码。

需要将 mrpt/bin 目录添加到系统环境变量 PATH 中，才可以执行后面章节中的命令。

目　录

第一章 GridmapNavSimul 栅格地图导航模拟器

1.1 简介

GridmapNavSimul 栅格地图导航模拟器，是一个基于占据栅格地图的 GUI 程序，可以让用户模拟并移动一个装有激光扫描仪的虚拟机器人。这个机器人的运动可以手动由键盘控制，或者由游戏手柄控制，或者执行一套设定在文件中的姿态序列。

这个程序的目的是保存所有的传感器数据作为一个数据集文件（ dataset files ）：激光扫描和里程。这个程序将生成一个合成的机器人数据集。

这个应用已经包含在 MRPT 包中，（Ubuntu 用户需要安装 mrpt-apps），它的源代码在 MRPT/apps/.，这个程序相关的 C++类：

- mrpt::slam::COccupancyGridMap2D: 2D 栅格地图类
- mrpt::slam::CObservation2DRangeScan: 2D 激光扫描观测数据
- mrpt::utils::CRobotSimulator: 一个非常简单的 2D 机器人运动模拟器

1.2 应用

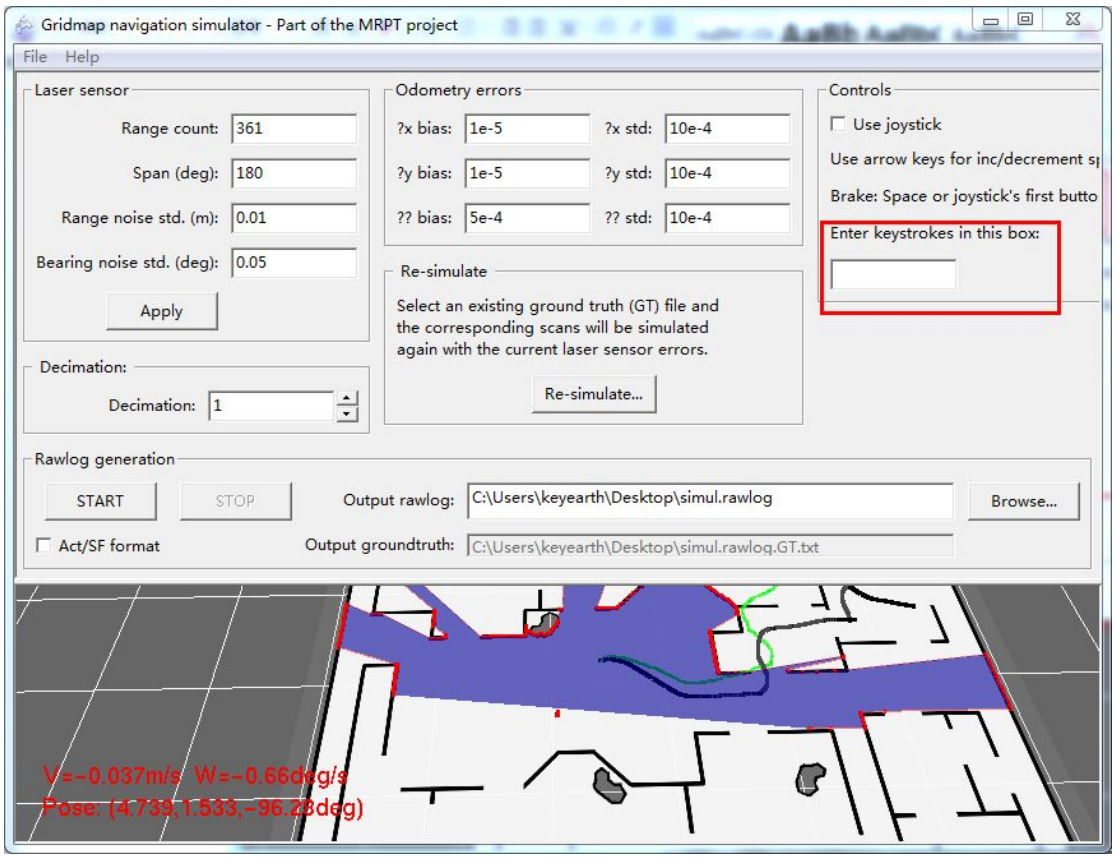
两个传感器（激光扫描仪和里程计）都有可配置的噪声级和不同类型的模拟随机误差。

用户可以配置虚拟激光扫描仪的参数，比如扫描角度、角度误差、扫描范围、范围误差。也可以配置虚拟里程计的误差参数。

【本章以下部分是由译者添加，不保证绝对准确】

键盘方向键控制：将鼠标焦点移到图上红框处的编辑框中，用键盘上的前后左右方向键控制加速、减速、左转、右转。长按空格可刹车。

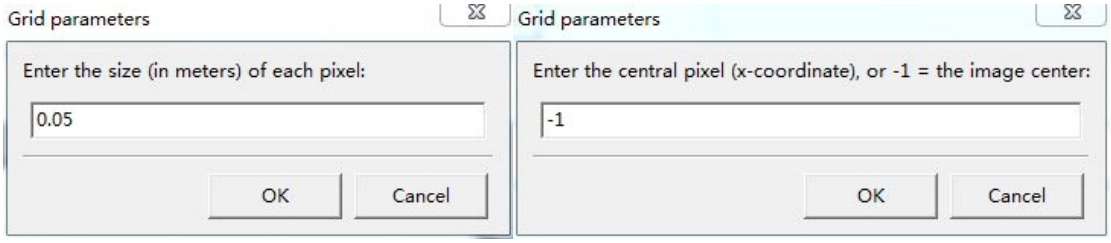
游戏手柄控制：普通游戏手柄基本都可以控制，只是键位可能不同。需要选中“use joystick”。用摇杆控制前后左右，用某个按键控制刹车。



用户也可以导入自己画的栅格地图，或者别的方式得到的栅格地图。必须是灰度图。地图必须是.png 格式。

- 白色代表可行区域。
- 黑色代笔禁行区域。
- 灰色代表障碍物。

菜单中，File-->Load gridmap 选择要载入的地图文件，然后会弹出小窗口要求设定参数，设定每个像素换算为多少米。以及坐标原点在图片上的位置（-1 表示中点）。

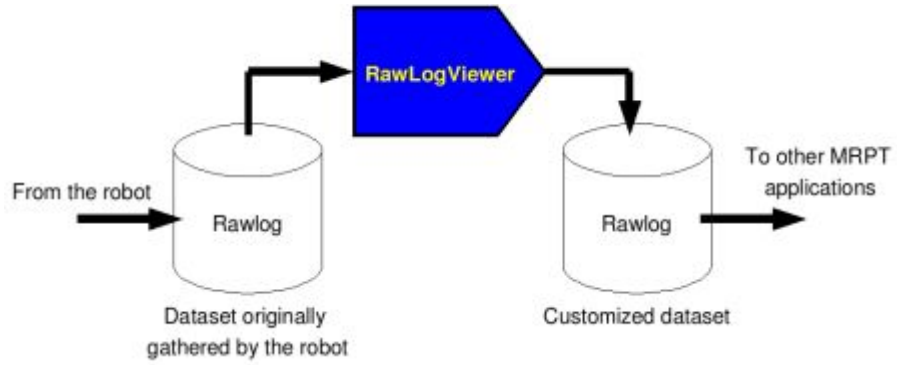


载入后，用户就可以控制虚拟机器人在这个地图下的运动，点击“START”可以开始将虚拟机器人的运动过程数据记录到 rawlog 文件中。Rawlog 文件可以被 RawlogViewer 软件（后面介绍）读取编辑，用于数据分析、算法仿真、创建地图等。

第二章 RawLogViewer 原始记录查看器

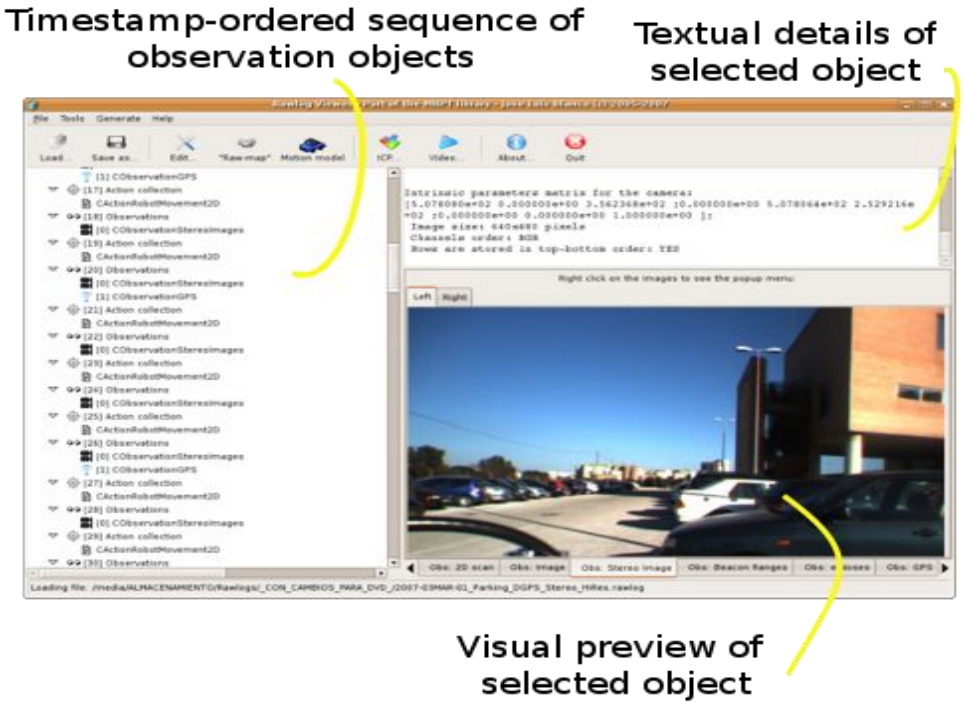
2.1 目的

RawLogViewer 是一个 GUI 应用程序，主要目的是允许用户对一个机器人采集的 Rawlog （数据集）进行快速检查、剪切、修改，以便将修改后的 rawlog 作为其它程序的输入，比如用于定位（pf-localization）或者建图（kf-slam，rbpf-slam）。



一些典型的应用示例，比如在庞大的数据集中仅提取一部分，比如从高密数据集中间隔抽取一部分，或者是修改机器人上的传感器位置。这个程序允许从使用 CARMEN 格式（也见:carmen2rawlog）或者 MOOS 异步记录格式（“*.alog”）的记录文件中导入数据集。另外，也实现了有限导出为纯文本文件（比如通过 MATLAB 处理它们)。请注意,这个应用程序未来的发展将集中在可视化方面，修改数据集时将会使用一个单独的程序：rawlog-edit。

2.2 界面描述



- **Left：**已载入数据集中存储的所有对象序列。单击选中的一个对象可以在右边的两

个窗口中预览它的内容。

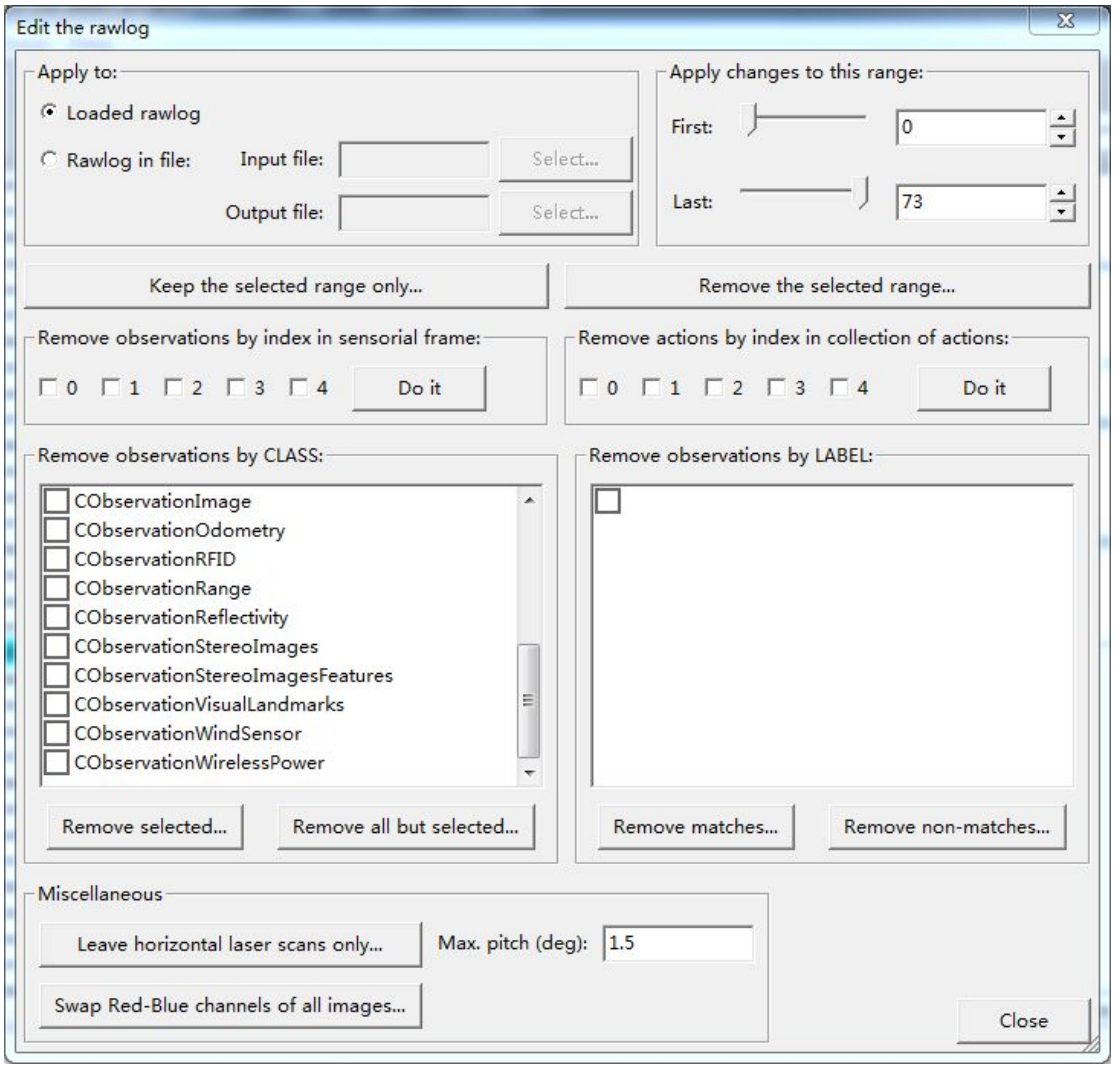
- **Top-right:**被选择对象的文本描述。
- **Bottom-right：**对于某些对象，可以可视化预览观测量内容。比如，激光扫描的二维视图，单目或立体摄像头的图像等等。当没有选择对象时，这个面板展示一个数据集的概要，包括每个传感器速率的信息等等。

2.3 编辑原始记录

2.3.1 剪切，选择性删除

菜单路径：“Edit ”—>“Edit rawlog.....”

描　　述：这个操作会打开一个对话框，其中有几个可配置的过滤器，可根据时间位置，传感器类型或者传感器标签去移除数据集中的指定部分。

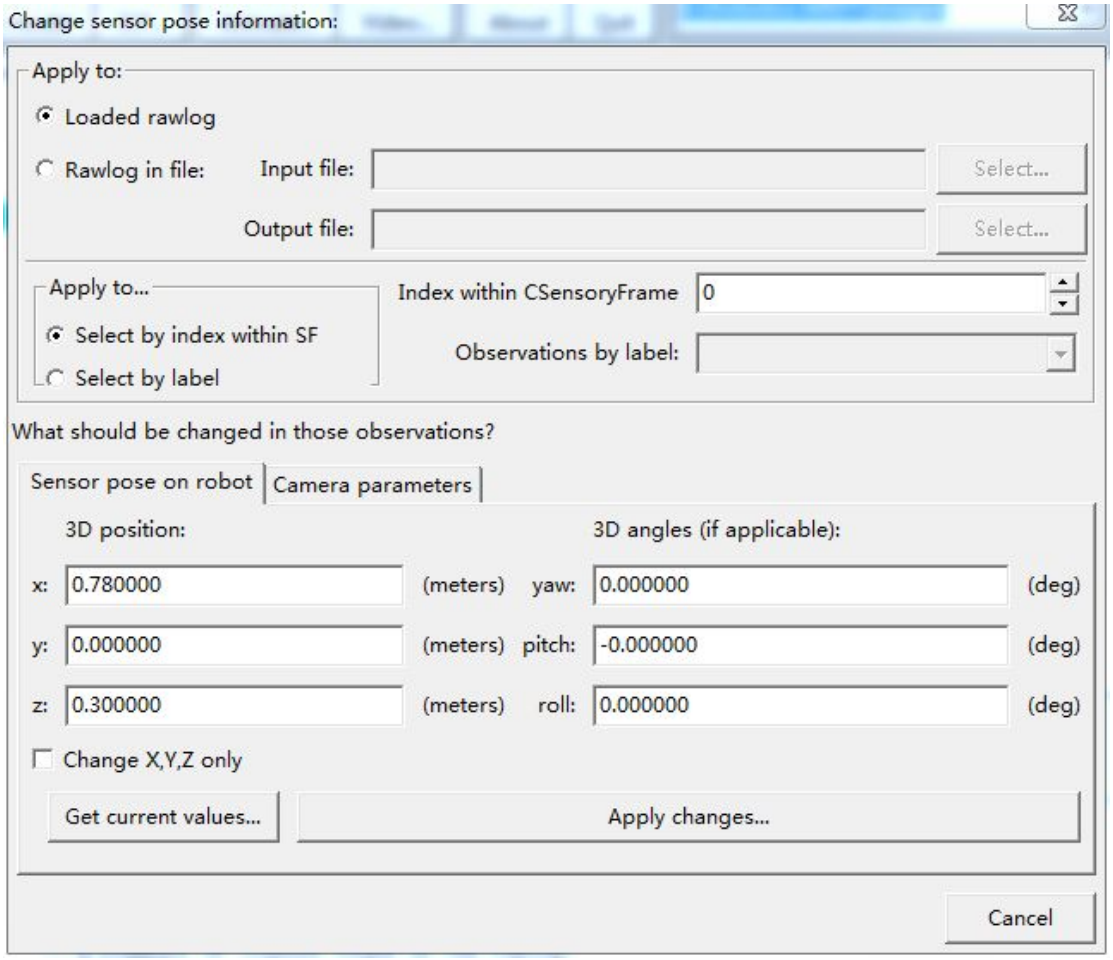


2.3.2 改变传感器位姿

菜单路径：“Edit”->“Change sensor/camera parameters....”

描　　述：数据集中的所有的传感器对象（观测 observation）都有一个相关的“pose on the robot”，描述了传感器在车架参考（例如，通常都是手持设备原点）上的 6D 坐标位置。

下面这个对话框允许用户一次完成获取和设置数据集中所有对象的位姿，通常与传感器标签一起使用：



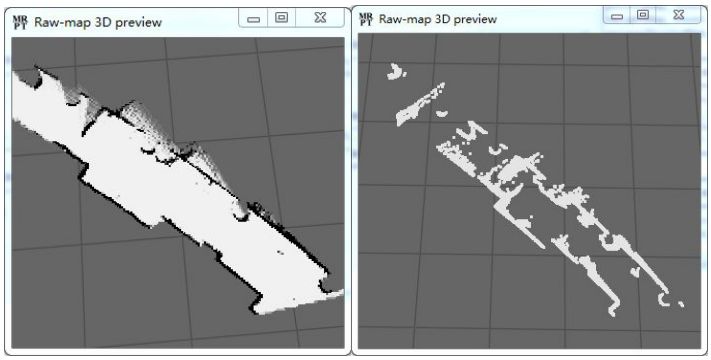
2.4 地图路径生成模块

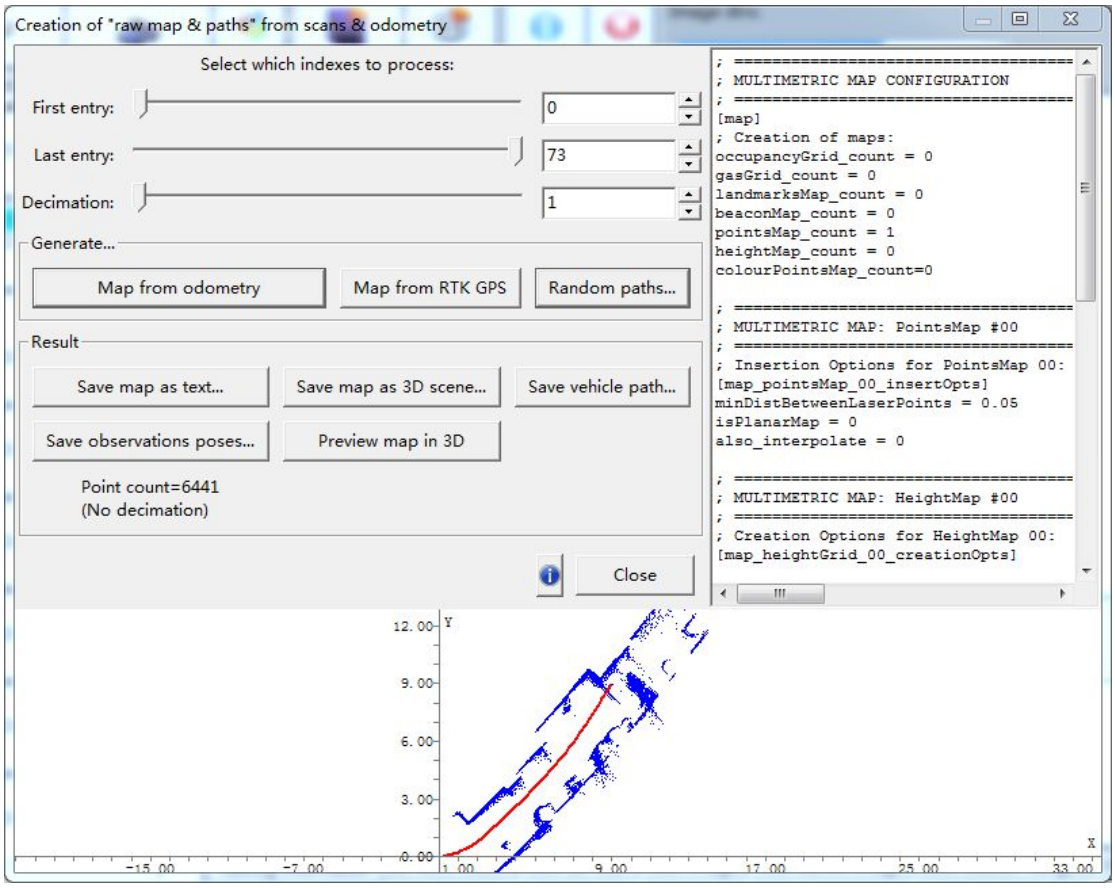
选项路径：“Tools”->“Open the maps & path generatiion module ...”

描　　述：这个对话框允许用户做 3 种事，主要用于 2D/3D 激光扫描数据。

用户可以使用 rawlog 文件中的激光扫描数据和里程计数据，或 GPS 数据，直接生成地图【译者注：这种方法并没有使用 slam 算法】。点击 preview 可以在 3D 窗口查看。

右上角窗口是生成地图的配置文件内容，里面的参数可以被手动修改，比如改为生成占据栅格地图。（目前仅支持生成点云地图和占据栅格地图 这两种地图）





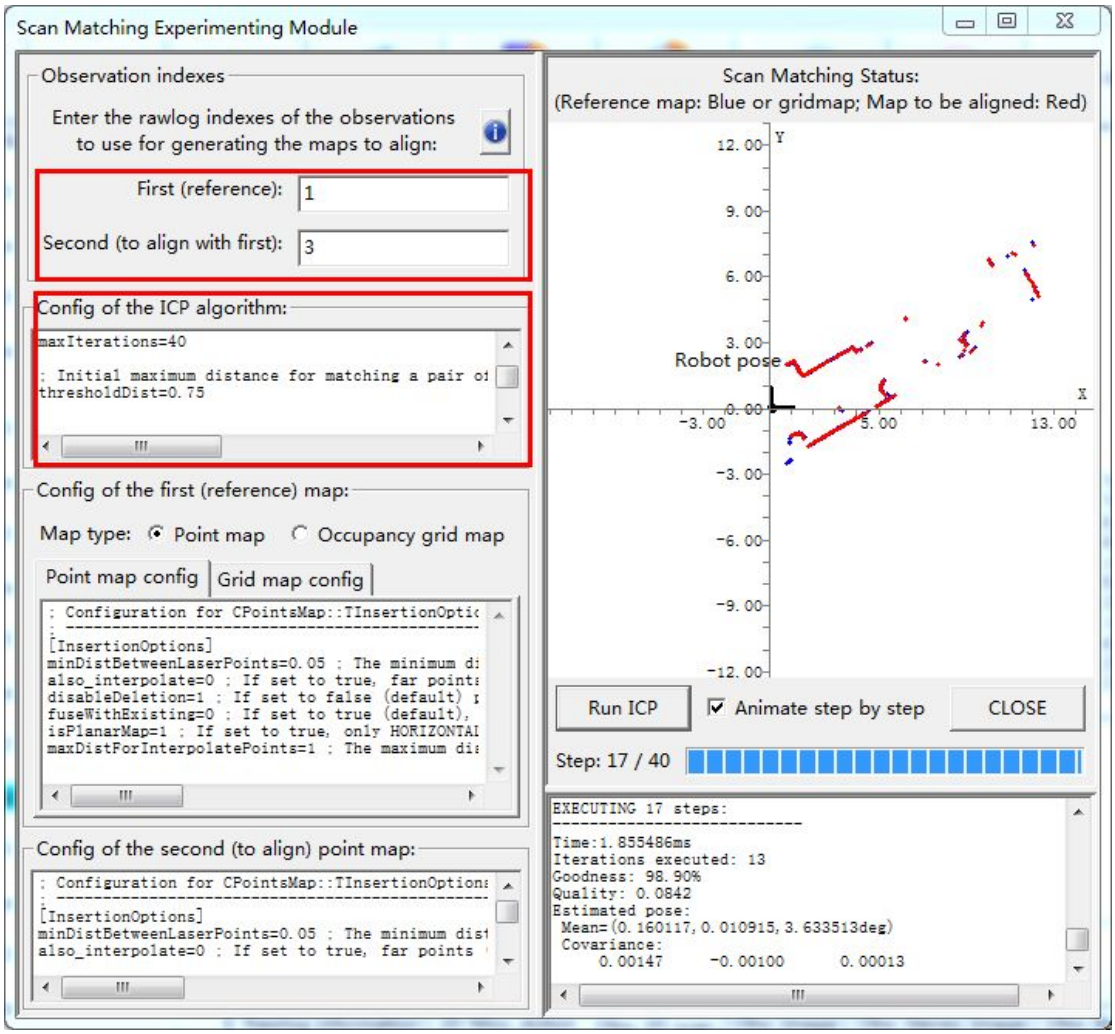
2.5 扫描匹配(ICP)模块

ICP 扫描匹配模块是 RawLogViewer 应用软件中的一个 ICP 算法图形化前端，基于 MRPT C++库实现。即当前加载的 rawlog 中的两帧相邻观测值用下面这个类实现对齐：

- mrpt::slam::CICP

之后你可以设置并修改一堆算法参数。

这个模块可以用来通过改变参数来尝试 ICP 算法的性能，有助于对一些具体应用得到一些整定参数，或者仅仅作为一个试错试验。尤其是，整个运算迭代过程可以像看动画一样观看。



【本小节以下内容为译者添加】

你可以用 RawLogViewer 打开任意有激光数据和里程数据的 rawlog 数据文件，比如 MRPT/share/datashets/localization_demo.rawlog 。找到在时间序列上比较相近的两个传感帧，比如 1 和 3 帧，在 ICP 模块里左上角的“first”和“second”内分别写入 1 和 3，选中“Animate step by step”，点击“Run ICP”后即可看到匹配算法的匹配过程动画。

可以在 ICP 参数配置框中更改 ICP 参数，再“Run ICP”查看效果。

更详细的 ICP 算法知识请查阅相关论文资料。

2.6 Actions:里程及不确定性

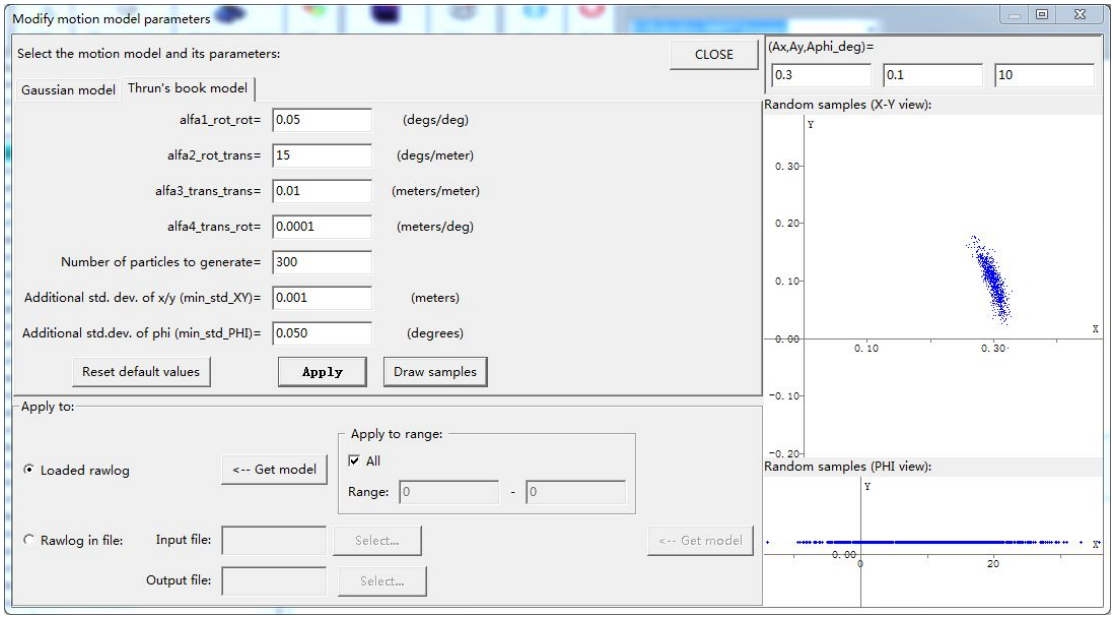
2.6.1 概念

对于每种概率模型的数学解释查看专题教程：（tutorial on the topic）

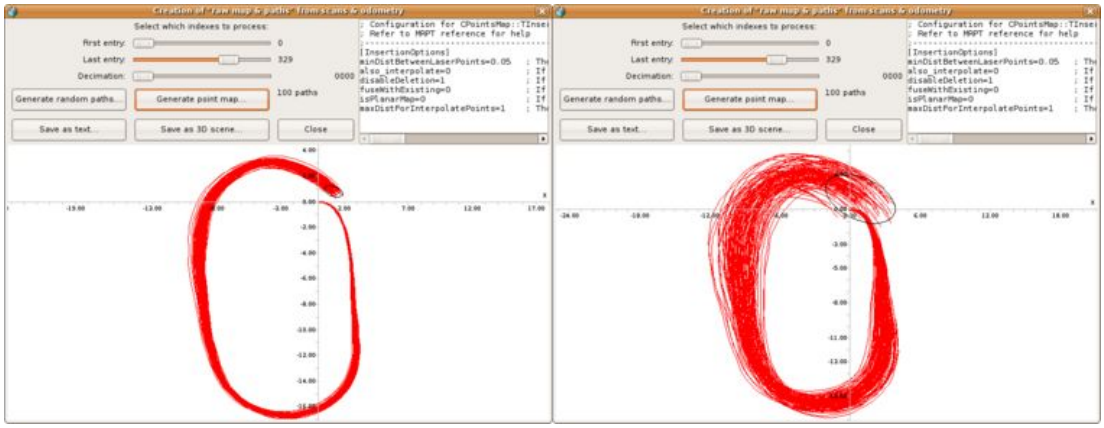
2.6.2 修改概率运动模型

菜单路径: Sensor->Odometry->Modfify motion model...

描　　述：这个窗口内允许试验两种不同的概率运动模型的参数，允许改变这些用于 rawlog 中所有里程增量数据的参数。



对于某些基于粒子滤波的 SLAM 算法，建立尽可能真实的不确定模型非常关键。下一个例子中可以看到，有时候需要增加里程增量的不确定度，以便使路径闭环尽可能包括机器人路径所有潜在的概率分布的真实假设。



注意上面左侧的图片，就是因为对里程数据不确定度设置的偏小（过度自信），导致任何 RBPF SLAM 算法都无法正确探测到路径闭环。

2.6.3 使用扫描匹配生成“里程”

菜单路径：Sensor->Odometry->Recalculate actions with ICP...

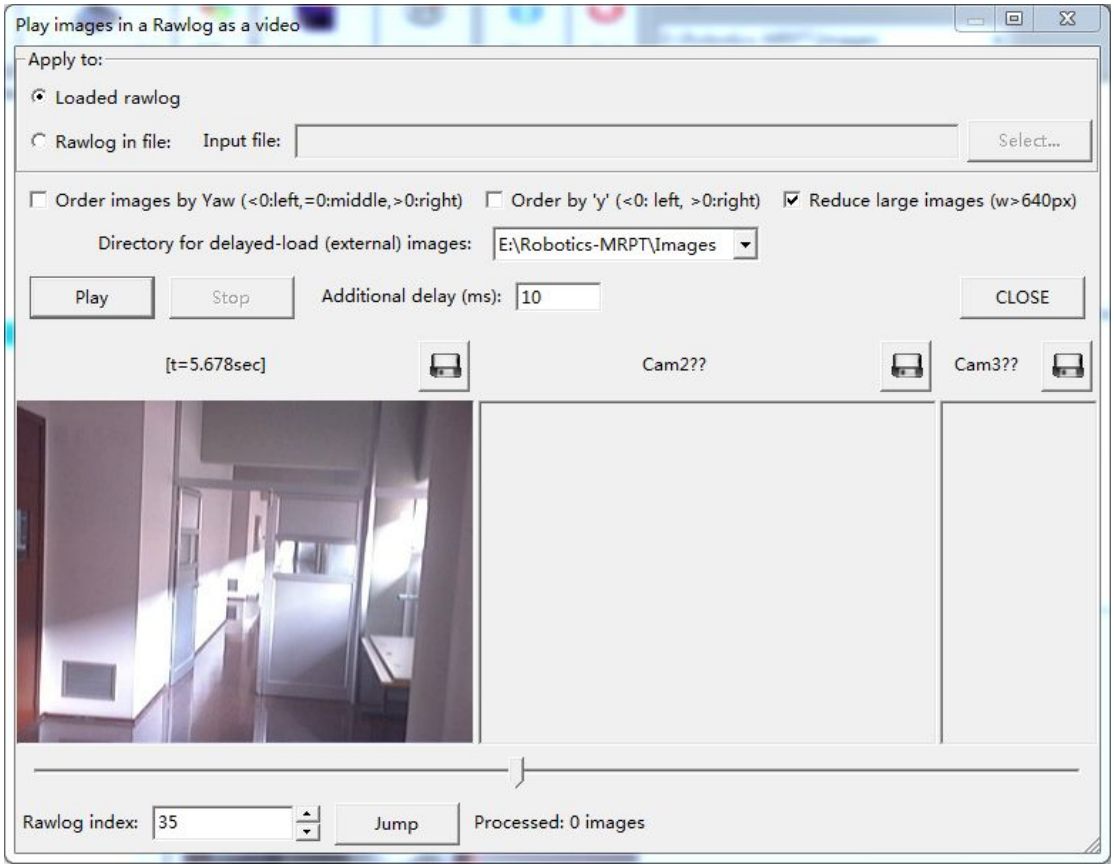
描　　述：该操作允许”纠正“测量法读数，通过执行 ICP 扫描匹配的每一对连续的观擦值。注意，此操作仅对 rawlog 中“SensoryFrame”有效。

【译者注：本部分功能已经在 2.5 节介绍过。】

2.7 视频化显示图像集

菜单路径：Sensor->Image->Show image as a video....

描　　述：这个模块允许可视化所有图像传感器的数据集，作为一个视频流或者整个过程使用滚动条查看。



这个功能可以作用于单目摄像头、立体摄像头 和 3D 摄像头的深度通道。

2.8 使用差分 GPS 数据生成地理路径

这个对 rawlog 的特殊应用在这篇文章中有详细描述。paper

简介：

- 加载 rawlog
- 点击“Raw map”按钮
- 如果需要，标记需要控制的范围。
- 如果不想在 3D 地图中插入所有激光扫描数据，可以设置 decimation>1。
- 点击“Map from RTK GPS”

地图应该出现在底部窗口，点击“Save vehicle path...”，大量信息就可以输出到文本文件，包括重建的真实地图。

第三章 RawLog-Grabber 原始记录抓取器

3.1 描述

RawLog-Grabber 是一个命令行应用，使用通用传感架构去实时采集机器人上各传感器的数据，并考虑了不同传感器不同工作频率的情况。所有数据都是按时间戳排序的，注入到 rawlog 文件中。

这个应用根据配置文件中的参数为每个传感器创建一个采集线程，并将按时间戳排序的观测数据存入 rawlog 文件。这个应用的关键作用是可以从移动机器人上记录原始数据用于离线处理。

可用的“传感器驱动”是指那些从 mrpt::hwdrivers::CGenericSensor 派生的驱动类。到官网查看 MRPT 支持的传感器和设备列表。

3.2 使用

在命令行执行：

```
rawlog-grabber   <config_file.ini>
```

3.3 配置文件

配置文件举例如下所示，注意配置文件中必须有一个[global]区用于定义通用配置参数。

```
[global]
// The prefix can contain a relative or absolute path.
// The final name will be ${rawlog_prefix}_${date}_${time}.rawlog
rawlog_prefix = dataset

// Milliseconds between thread launches
time_between_launches = 800

// SF=1 (or "true"): Enabled= Observations will be grouped by time periods.
// SF=0 (or "false"): Disabled= All the observations are saved independently
//                                          and ordered solely by their timestamps.
use_sensoryframes = false

// Only if "use_sensoryframes=1": The maximum time difference between
// observations within a single sensory-frame.
SF_max_time_span = 0.25 // seconds

// Observations will be processed at the main thread with this period
GRABBER_PERIOD_MS = 1000 // ms
```

然后，下面为每个传感器设定一个参数配置区段(section)，参数名取决于该传感器类中

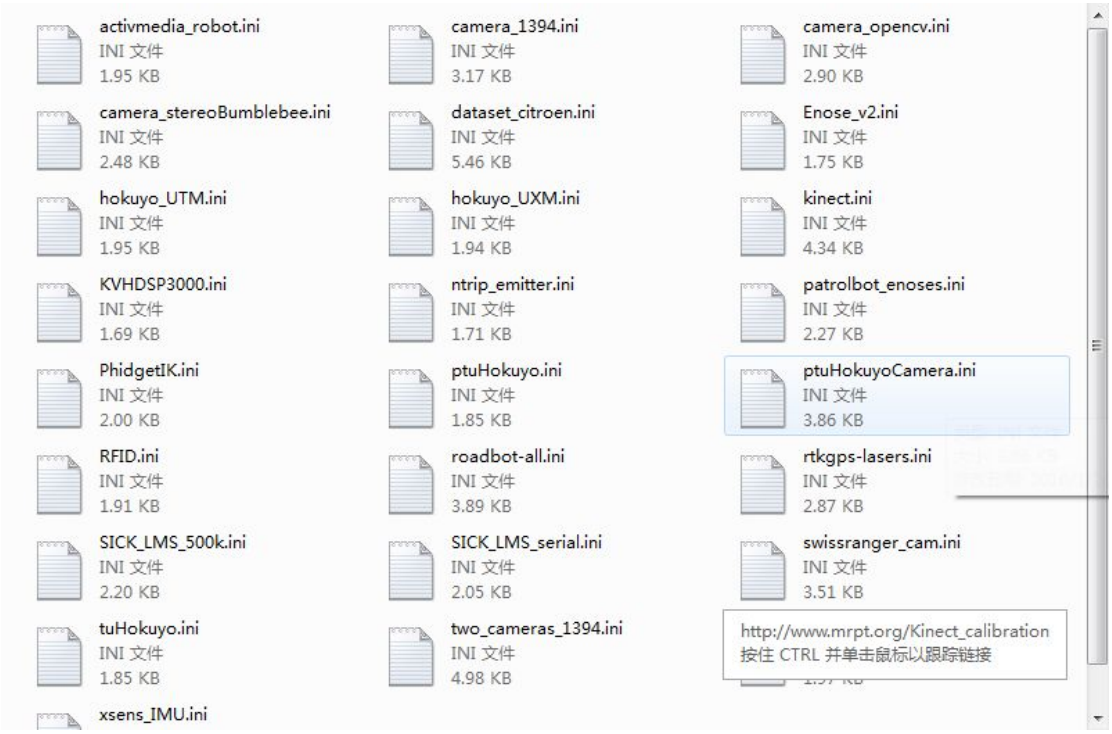
的参数要求。“传感器区段”的基本结构如下所示：

```
[SECTION_NAME]
driver = CLASS_NAME
process_rate = PROCESS_RATE // Hz
sensorLabel = SENSOR_LABEL
// the rest of sensor-specific parameters.... (See the class Doxygen docs)
```

注意事项：

- SECTION_NAME: 所有的区段名字必须是不同的。
- CLASS_NAME: 有效值是从类 CGenericSensor 派生的类名（不带命名空间前缀）。（比如有效值可以是“CCameraSensor”,”CHokuyoURG”,…）
- PROCESS_RATE: 设置在该传感器线程主循环中的最大执行频率。确保大于传感器能发送的最大观测数量。
- SENSOR_LABEL: 任意文本，用于指示是这个传感器发送的观测量。（与 SECTION_NAME 无关，可以和它完全不一样）

mrpt 安装好后，在目录.\mrpt\share\mrpt\config_files\rawlog-grabber 下已经准备好了常用设备的配置文件，用户只需要根据自己的机器人配置修改某些参数即可直接使用。



第四章 RawLog-Edit 原始记录编辑器

4.1 描述

RawLog-Edit 是一个命令行应用程序，机器人数据集（rawlog）处理工具。

4.2 参数

```
rawlog-edit   [--rename-externals] [--stereo-rectify <SENSOR_LABEL,0.5>]
              [--camera-params <SENSOR_LABEL,file.ini>] [--sensors-pose
              <file.ini>] [--generate-pcd] [--generate-3d-pointclouds]
              [--cut] [--export-2d-scans-txt] [--export-imu-txt]
              [--export-gps-txt] [--export-gps-kml] [--keep-label <label[
              ,label...]>] [--remove-label <label[,label...]>]
              [--list-range-bearing] [--remap-timestamps <a;b>]
              [--list-timestamps] [--list-images] [--info]
              [--externalize] [-q] [-w] [--to-time <T1>] [--from-time
              <T0>] [--to-index <N1>] [--from-index <N0>]
              [--text-file-output <out.txt>] [--image-size <COLSxROWS>]
              [--image-format <jpg,png,pgm,...>] [--out-dir <.>] [-o
              <dataset_out.rawlog>] -i <dataset.rawlog> [--] [--version]
              [-h]
```

4.3 用法示例

【译者注：确保已经按照 MRPT 安装说明将“MRPT/bin/”目录添加到系统环境变量 PATH 中。打开命令行终端，执行下面的命令。示例中的 in.rawlog 文件仅作为演示需要，用户可以改为自己的文件名】

- 快速查看一个数据集文件：

```
rawlog-edit   --info   -i   in.rawlog
```

- 将源文件 in.rawlog 中[1000,2000]之间的观测帧切出保存到 out.rawlog 文件中：

```
rawlog-edit   --cut   --from-index 1000   --to-index 2000   -i in.rawlog -o out.rawlog
```

- 将源文件 in.rawlog 中从第一帧到时间戳为 1281619819 之前的帧切出保存到 out.rawlog 文件中：

```
rawlog-edit   --cut   --to-time 1281619819   -i in.rawlog -o out.rawlog
```

- 使用数据集中的 GPS 数据生成 Google Earth KML 文件：

```
rawlog-edit   --export-gps-kml -i in.rawlog
```

- 移除所有名字为“REAR_LASER”的观测帧：

```
rawlog-edit  --keep-label REAR_LASER -i in.rawlog -o out.rawlog
```

- 将数据集中的图片转换为外部存储模式【译注：图片会被保存在/out_images 文件夹中】:

```
rawlog-edit  --externalize -i  in.rawlog -o  out.rawlog

rawlog-edit  --externalize  --image-format  jpg  -i  in.rawlog  -o  out.rawlog
```

4.4 参数详解

```
 --rename-externals
      Op: 重命名 rawlog 文件中的所有外部存储文件的名字(不会更改文件内容).

 --stereo-rectify <SENSOR_LABEL,0.5>
    Op: 使用给定的 SENSOR_LABEL 和 alpha 值，以及存储在观测帧中的参数（必须是校正过的）创建一个 CObservationStereoImages 外部图片集。Alpha 值可以是 -1 表示自动，或[0,1]之间的数。（具体查看 OpenCV 文档中 cvStereoRectify 内容）

      Requires: -o (or --output)

      Optional: --image-format 用于设定图片格式(default=jpg),
                --image-size 用于变更图片尺寸 (example: --image-size 640x480)

 --camera-params <SENSOR_LABEL,file.ini>
      Op: 基于给定的 SENSOR_LABEL，使用 file.ini 文件中的参数改变摄像头 CObservationImage 参数。配置区'[CAMERA_PARAMS]' 用于单目摄像头，'[CAMERA_PARAMS_LEFT]' 和'[CAMERA_PARAMS_RIGHT]' 用于立体摄像头。

      Requires: -o (or --output)

 --sensors-pose <file.ini>
      Op: 基于给定的 file.ini（类似 rawlog-grabber 的配置文件，由传感器标签名 sensorLabel 分别设定传感器的位姿），批量修改摄像头位姿。

      Requires: -o (or --output)

 --generate-pcd
      Op: 使用可转换为点云数据的传感器(laser scans, 3D camera images, etc.)观测值，生成一个 PCD 文件（点云库 PointCloud Library :PCL）。
      Optional: --out-dir 用于改变输出路径(默认: "./")

 --generate-3d-pointclouds
      Op: 在带有测距数据的 CObservation3DRangeScan 对象内（重）生成3D 点云数据。
```

Requires: -o (or --output)

--cut

Op: 在输入的 rawlog 文件中切出一段.

Requires: -o (or --output)

Requires: 至少使用--from-index, --from-time, --to-index,--to-time 的其中一个。每次只能使用一个--from-* 和 --to-* 。

如果只使用了--from-*, 那么 rawlog 会从给定帧保存到末尾帧。

如果只使用了--to-* ，那么 rawlog 会从开始帧保存到给定帧。

--export-2d-scans-txt

Op: 将2D 扫描数据导出为 txt 文件。

2D 扫描观测量中每个不同的传感器标签会生成两个.txt 文件，一个保存时间戳，一个保存测距值。

生成的.txt 文件会保存在 rawlog 文件同目录下，命名为 rawlog 名+sensorlabel 名。

--export-rawdaq-txt

Op: 导出原始 DAQ 说明到 txt 文件。

每个不同的传感器标签和通道生成一个.txt 文件。

生成的.txt 文件会保存在 rawlog 文件同目录下，命名为 rawlog 名+sensorlabel 名。

--export-imu-txt

Op: 将 IMU 数据导出为 txt 文件。

数据集内 IMU 观测量的每个不同的传感器标签生成一个.txt 文件。

会保存在 rawlog 文件同目录下，命名为 rawlog 名 +sensorlabel 名。

--export-gps-txt

Op: 将 GPS 数据导出为 txt 文件。

数据集内 GPS 观测量的每个不同的传感器标签生成一个.txt 文件。

生成的.txt 文件会保存在 rawlog 文件同目录下，命名为 rawlog 名+sensorlabel 名。

--export-gps-kml

Op: 将 GPS 路径导出为 Google Earth KML 文件.

数据集内 GPS 观测量的每个不同的传感器标签生成一个.kml 文件。

生成的.kml 文件会保存在 rawlog 文件同目录下，命名为 rawlog 名+sensorlabel 名。

--keep-label <label[,label...]>

Op: 移除所有不符合给定 label(s)的观测数据。多个 label 可用逗号分隔。

Requires: -o (or --output)

--remove-label <label[,label...]>

Op: 移除所有符合给定 label(s)的观测数据。多个 label 可用逗号分隔。

Requires: -o (or --output)

--list-range-bearing

Op: dump a list of all landmark observations of type range-bearing.

Optionally 输出文本文件可以使用--text-file-output 改变。

--remap-timestamps <a;b>

Op: 将所有时间戳 t 线性映射为'a*t+b'。
参数 'a' 和'b' 必须用分号（;）分隔。

Requires: -o (or --output)

--list-timestamps

Op: 生成一个清单，列出所有观测帧的时间戳、传感器标签、C++类名。

Optionally 输出文本文件可以使用--text-file-output 改变。

--list-images

Op: 生成一个清单，列出数据集内所有外部图片文件。

Optionally 输出文本文件可以使用--text-file-output 改变。

--info

Op: 解析输入文件，并显示信息和统计数据。

--externalize

Op: 转换为外部存储。

Requires: -o (or --output)

Optional: --image-format

-q, --quiet

静默输出。

-w, --overwrite

目标文件已存在时，不提示直接覆盖。

--to-time <T1>

指示--cut 命令作用的终止时间戳。as UNIX timestamp, 可选小数秒。

--from-time <T0>
指示--cut 命令作用的起始时间戳。as UNIX timestamp, 可选小数秒。

--to-index <N1>
指示--cut 命令作用的终止帧索引。

--from-index <N0>
指示--cut 命令作用的起始帧索引。

--text-file-output <out.txt>
Output for a text file

--image-size <COLSxROWS>
Resize output images

--image-format <jpg,png,pgm,...>
External image format

--out-dir <.>
Output directory (used by some commands only)

-o <dataset_out.rawlog>, --output <dataset_out.rawlog>
Output dataset (*.rawlog)

-i <dataset.rawlog>, --input <dataset.rawlog>
(required) Input dataset (required) (*.rawlog)

--, --ignore_rest
Ignores the rest of the labeled arguments following this flag.

--version
Displays version information and exits.

-h, --help
Displays usage information and exits.

第五章 ReactiveNavigationDemo 被动式导航演示

5.1 简介

这个 GUI 程序显示了一个基于占据栅格地图的平面世界，用户可以设定目标点的坐标，运行被动式导航仿真。地图可改为用户提供的任意兼容的地图。所有决定导航系统行为的参数都可以由用户更改测试查看效果。

用户可以在地图上任意位置“右键-->navigation to this point”来设定目标点，点击“simulate”开始运行仿真。注意在第一次运行时，被动导航系统需要先花一会时间建立一个查找表，之后用于实时操作。【译者注：每个新地图只需要建立一次查找表】

这个仿真程序还可以生成导航记录(navigation logs)（就像在真实机器人上一样），这个导航记录文件可以使用应用软件 navlog-viewer 打开、查看、分析。

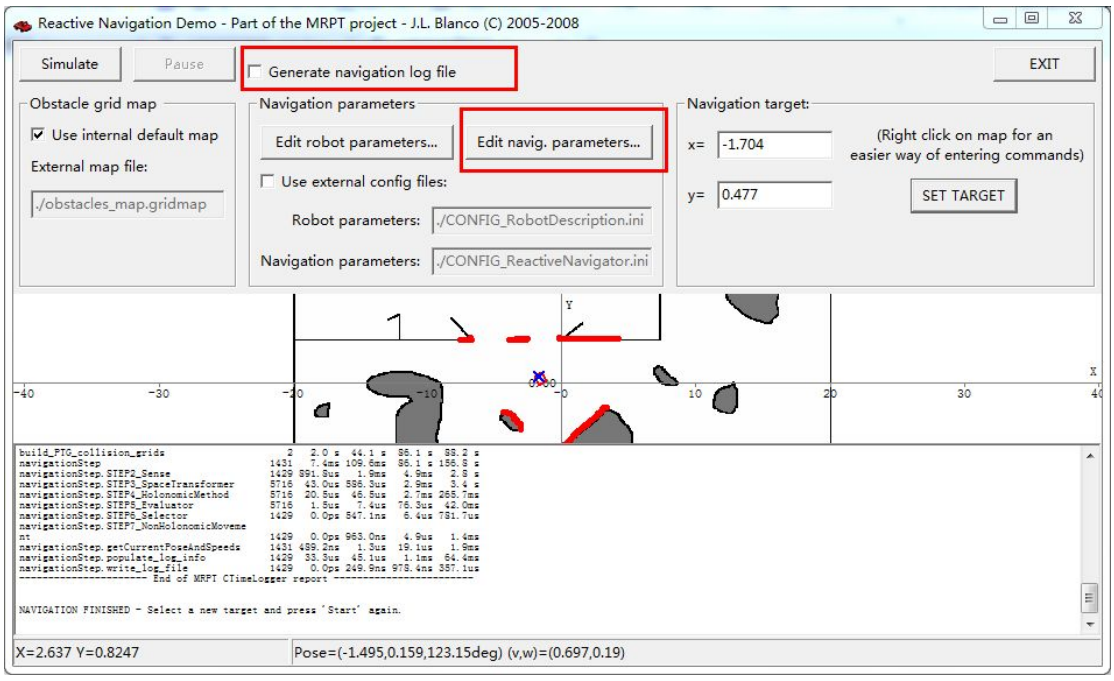
5.2 依赖类

这个应用软件使用 MRPT C++库中的如下几个类：

- mrpt::slam::CRobotSimulator: 用于模拟真实机器人的行为。
- mrpt::slam::COccupancyGridMap2D: 用于存储栅格地图，模拟 2D 激光扫描。
- mrpt::reactivenav::CReactiveNavigationSystem: 这个被动导航主引擎。

这个应用的源码在 MRPT 根目录的“apps/ReactiveNavigationDemo”目录。

5.3 界面介绍



- Simulate：开始仿真
- Pause：暂停仿真
- Obstacle gird map ：障碍栅格地图
  - Use internal default map: 使用默认地图
  - External map file：使用自定义地图
- Navigation parameters：导航参数
  - Edit robot parameter：编辑机器人参数
  - Edit navigation parameter：编辑导航参数
  - Use external config file：使用自定义配置文件
    - Robot parameter：机器人参数配置文件
    - Navigation parameter：导航参数配置文件
- Navigation target：设置导航目标点 x 和 y 坐标

【本章以下部分是由译者添加，不保证绝对准确】

5.4 使用

基本的使用流程是：

- 选择地图：默认地图或者自定义地图
- 分别编辑机器人参数和导航参数，或者使用机器人参数和导航参数的配置文件
- 在地图任意位置单击鼠标右键，选择“Navigation to this point”设置导航目标点。
- 单击“Simulate”，机器人开始导航，到达指定目的地。

【译者注：在 mrpt1.0.2 中默认导航参数中缺少两个参数，直接运行会报错，可以在导航参数最后添加下面两行：

ROBOTMODEL_TAU = 0

ROBOTMODEL_DELAY = 1

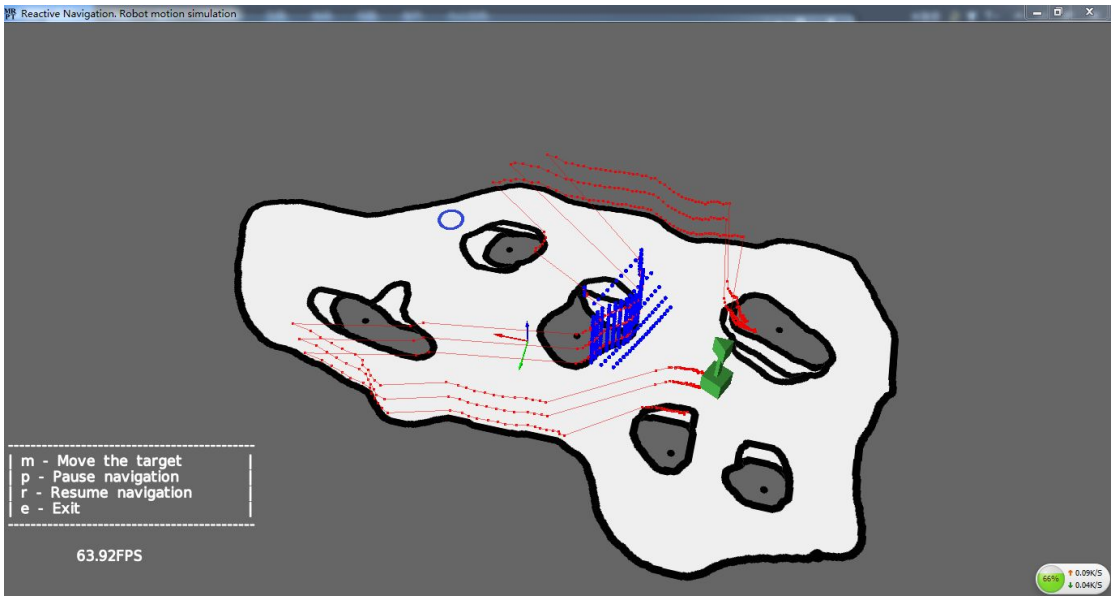
选中“Generate navigation log file”后可以生成导航记录文件，可以在 navlog-viewer 应用软件中打开。可以查看每一步导航的过程，分析每一步的状态，以便优化导航算法和参数。

生成的 navlog 文件在“reactivenav.logs 文件夹下”查找。】

第六章　ReactiveNav3D-Demo 被动导航 3D 演示

6.1 描述

这个 GUI 应用程序是用于测试 3D 被动导航器。实际上是通过给障碍物建立不同高度的多个占据栅格地图来模拟了一个“2.5D 世界”。机器人形状可以被用户定义，2D 或 3D 测距传感器也可以被“加装”到机器人上。在运行时，用户使用命令“m”设定不同的目标点位置，查看被动导航 3D 仿真效果。



提示，可以生成导航记录数据（就像真实机器人上一样），并在 navlog-viewer 中查看分析。

6.2 依赖类

应用程序使用 MRPT C++库中的下面这三个类：

- mrpt::slam::COccupancyGridMap2D: 用于存储栅格地图，模拟 2D 激光扫描。
- mrpt::reactivenav::CReactiveNavigationSystem3D: 这是 3D 被动导航的主引擎。
- mrpt::utils::CRobotSimulator: 用于模拟真实机器人的行为。

应用程序的源码在 mrpt 根目录的“apps/ReactiveNav3D-Demo”目录中。

6.3 命令行选项

虽然这个 GUI 应用程序不需要任何参数就可以运行，但有一些有用的功能可以通过命令行参数被启用。在命令行窗口输入：

```
ReactiveNav3D-Demo –help
```

你可以得到本应用可接受的参数列表：

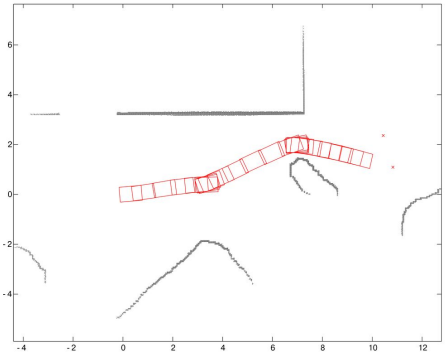
```
USAGE:

   ReactiveNav3D-Demo    [-h] [-c] [-cc ] [-s]

Where:

    -h, --help
      显示用户帮助信息。

    -c,   --config FILE.txt
      载入 FILE.txt 文件作为配置文件(替换默认配置).

    -cc,   --create-config FILE.txt
      将默认配置参数保存到 FILE.txt 文件并退出。
    -s ,   --save-logfile
      使能将导航记录数据保存到文件。可用 Navlog-viewer 查看。
```

第七章 Navlog-Viewer 导航记录查看器

7.1 简介

这个 GUI 应用程序允许用户细致分析在移动机器人导航过程中发生了什么：每一帧探测到的障碍物，被动导航算法计算出的移动方向，等。

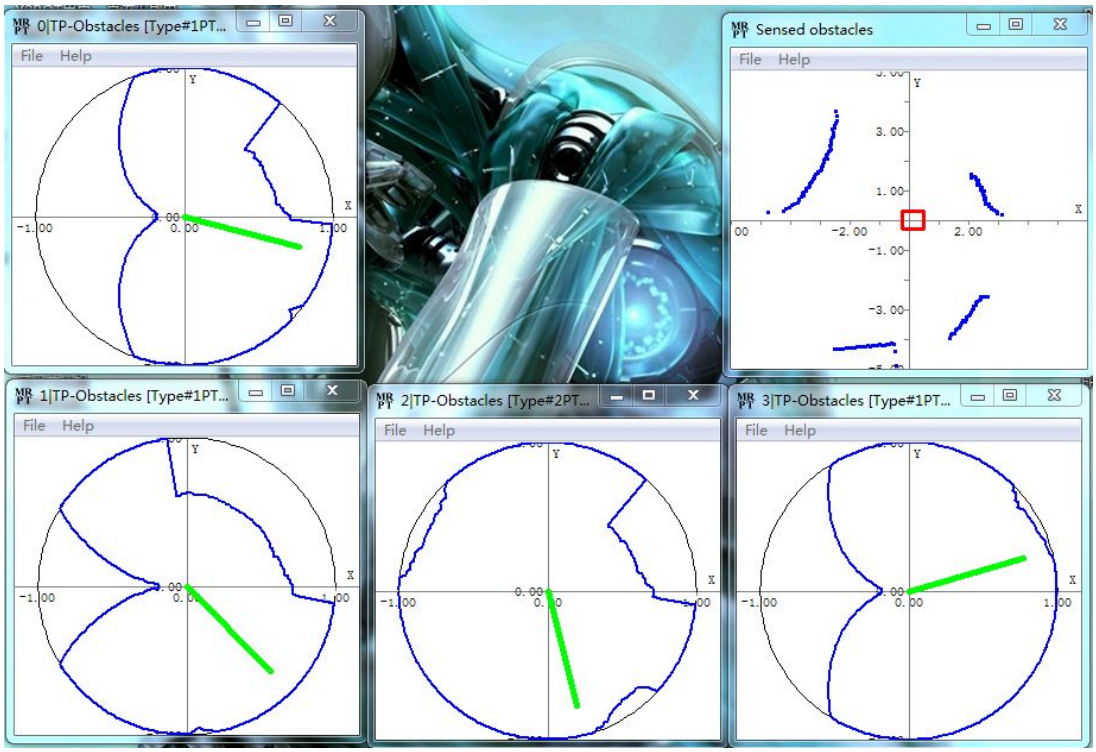
它还可以导出 MATLAB 脚本，用于生成有机器人轨迹信息的高质量图（机器人朝向跟随轨迹变化），以及所有探测障碍物的点云叠加图。如下图所示：



7.2 导入记录

可以使用 ReactiveNavigationDemo 应用软件中生成的导航记录文件，比如上一章中生成的 log_001.reactivenavlog 文件，导入后弹出 5 个窗口如下图所示。一个窗口是以机器人为中心的障碍物探测数据，其它 4 个窗口是被动导航算法计算出的目标方向和可行区域阈值。

可以点击“play”播放整个导航过程，也可以拖动滑块一帧一帧的查看。



第八章 holonomic-navigator-demo 完整约束导航演示

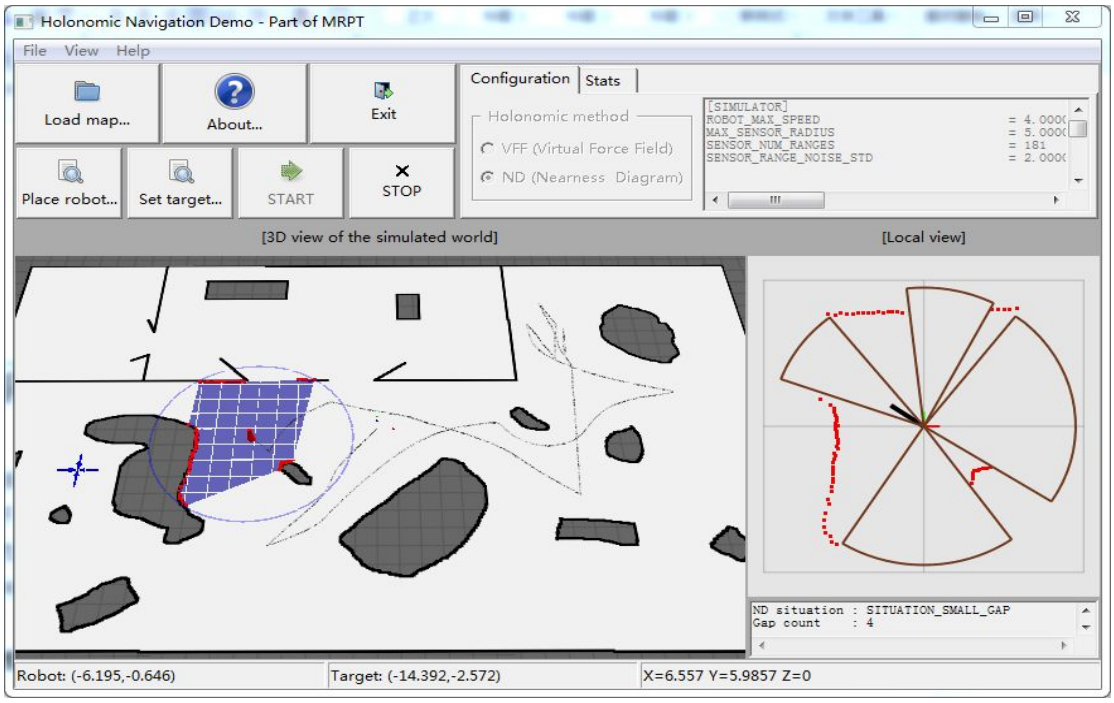
8.1 简介

这个 GUI 应用软件用于演示完整约束导航算法。在 mrpt-reactivenav 中实现。
用户可以使用这个程序 测试、学习、调试 下面这些算法：

- VFF (虚拟力场算法)
- ND(逼近图算法)

【本章以下部分是由译者添加，不保证绝对准确】

8.2 应用



Place robot ：将机器人放在地图中任意位置
Set target：在地图中任意位置设置机器人导航目标
START：机器人开始导航运行
STOP： 机器人停止导航运行
Load map: 加载自定义地图
Exit：退出程序
Configuration：选择完整约束算法：VFF 或 ND

- ROBOT_MAX_SPEED 机器人运行的最大速度
- MAX_SENSOR_RADIUS 360 度激光的最大测距范围（单位米）
- SENSOR_NUM_RANGES 360 度激光的激光束数量
- SENSOR_RANGE_NOISE_STD 传感器噪声参数

第九章 kinect-stereo-calib 立体摄像头校正

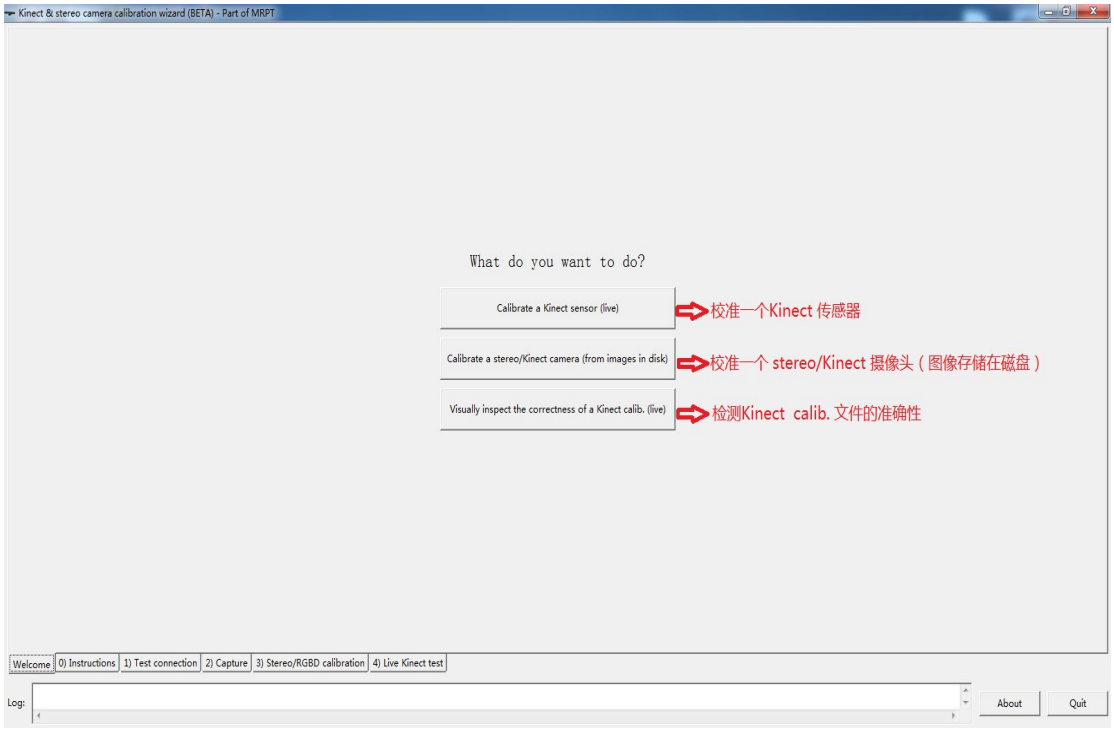
9.1 描述

Kinect_stereo_calib 应用主要用于：

- 使用棋盘校正 kinect 传感器的 intrinsic 和 extrinsic 参数。
- 使用用户已经抓取好的图片校正任何立体摄像头。它支持不同分辨率的摄像头，以及摄像头之间的任何基线和任何旋转。

一个 Levenberg-Marquartd 优化器已经用 C++实现，它也提供作为输出的每个参数估计（内部的数学原理细节见相关论文）。

9.2 校正 Kinect

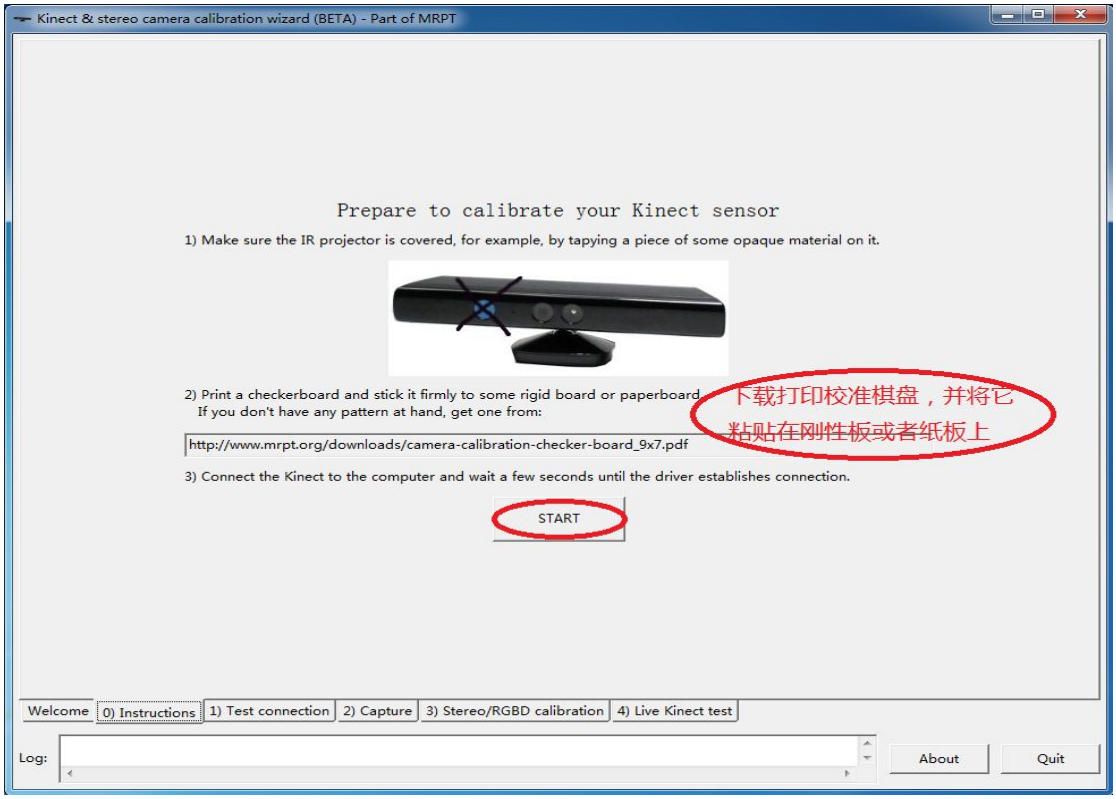


选择“calibrate a kinect sensor”校准 Kinect 传感器。并准备一个最好不少于 7*8 的黑白棋盘固定在墙面或硬基板上【译者注：最好保证棋盘 x 和 y 方向格数不同】。

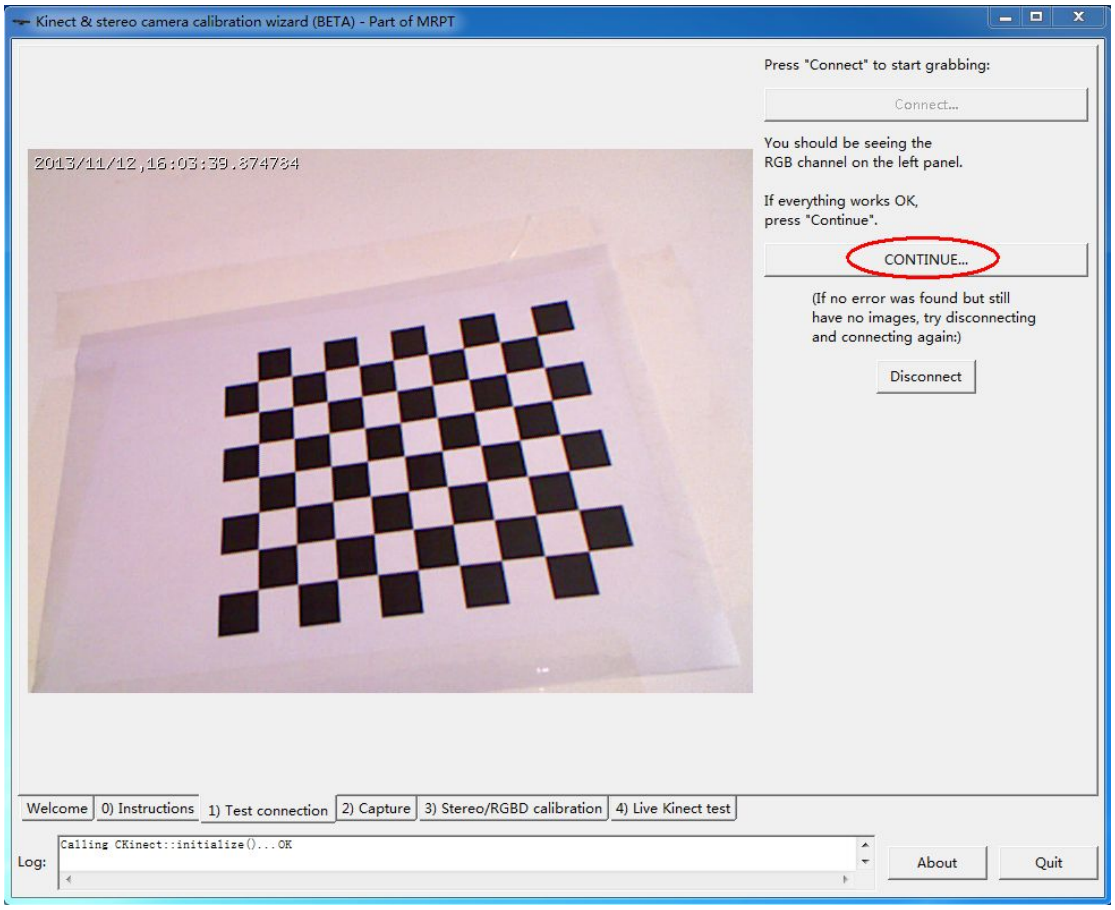
校正用的黑白棋盘可以从下列地址下载得到：

http://www.mrpt.org/downloads/camera-calibration-checker-board_9x7.pdf

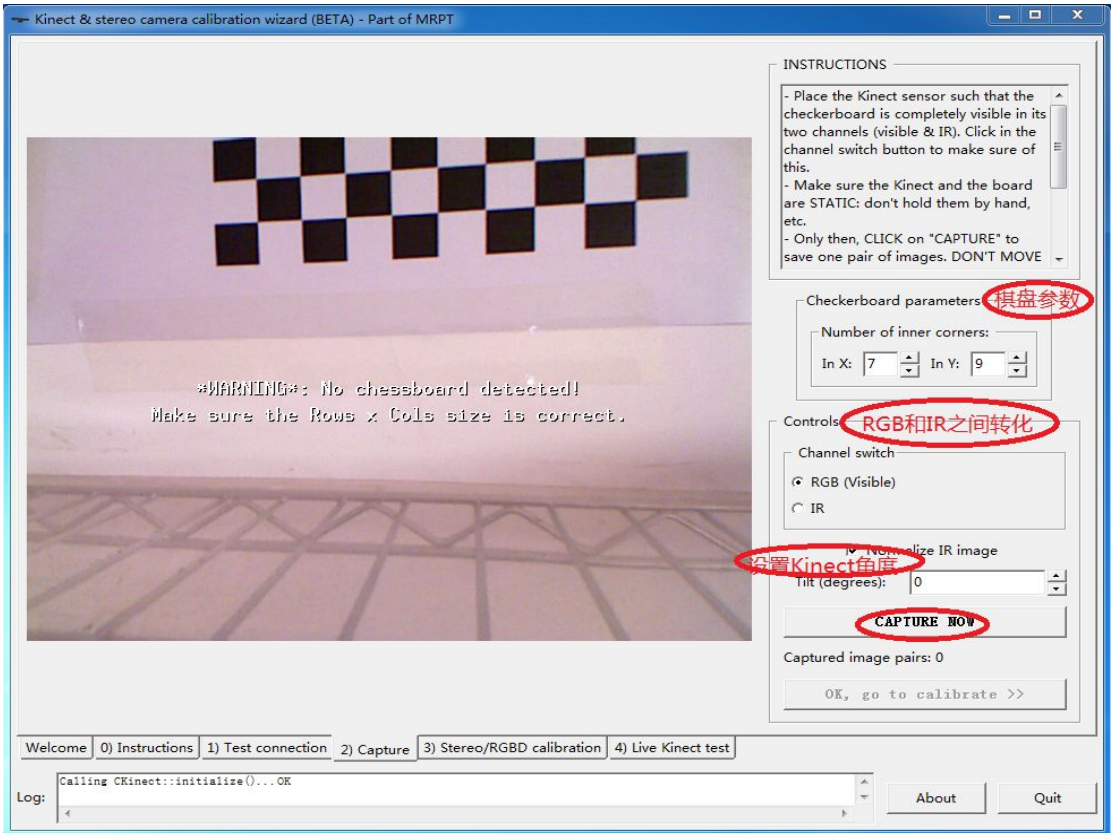
【本章以下部分是由译者添加，不保证绝对准确】



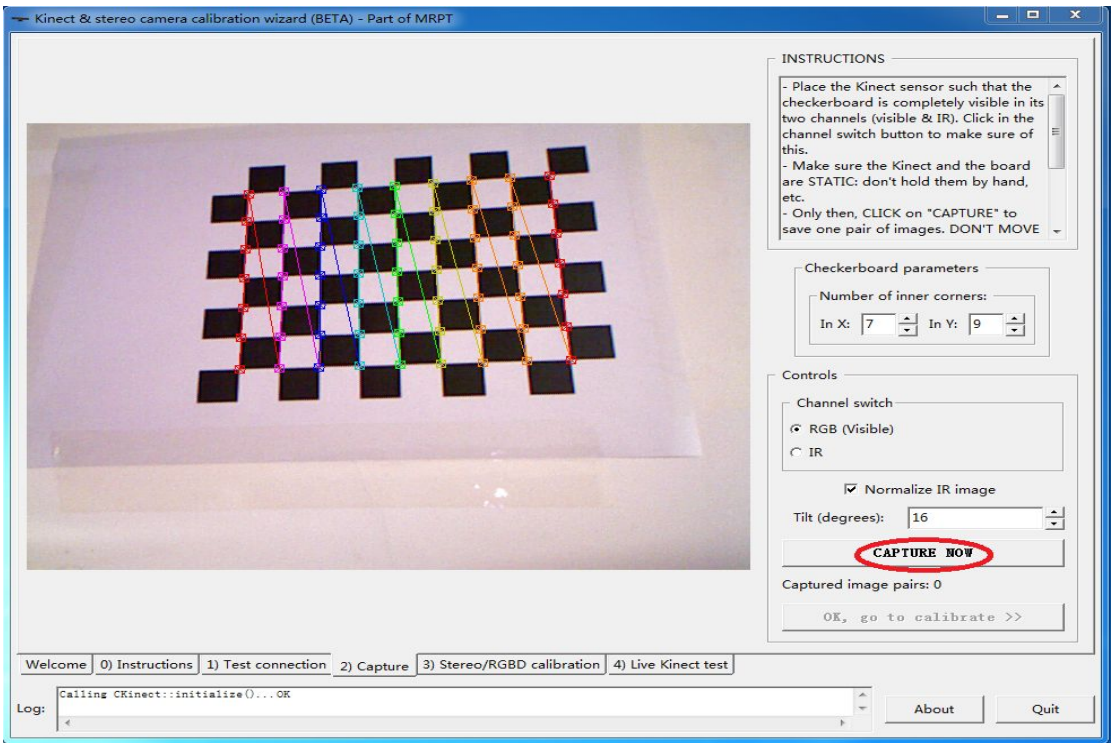
点击“start”后，确认 kinect 已经连接好，点击“connect”。如果一切正常，窗口中将显示 kinect 的 rgb 摄像头图像。



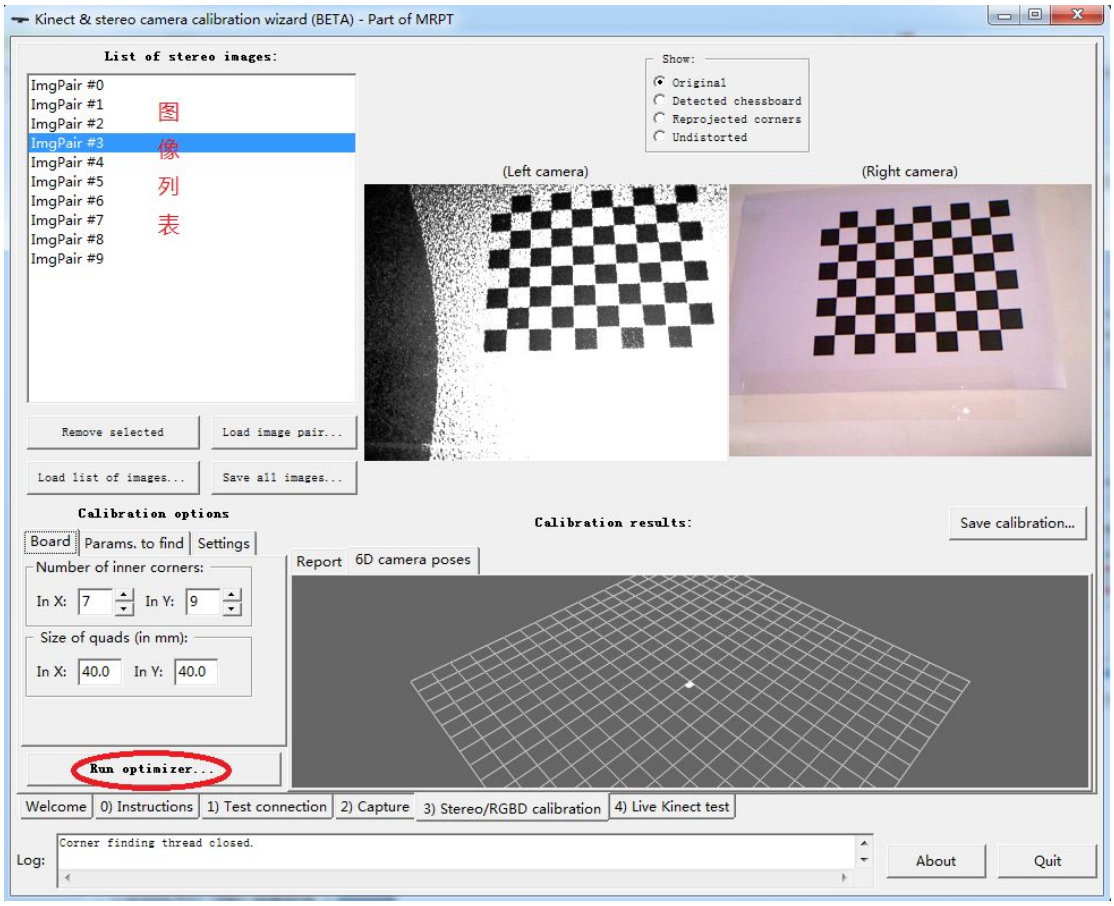
点击“continue”，进入参数设置界面。输入棋盘的棋格数量，选择 kinect 传感通道。



调整棋盘位置，确保 kinect 视野中可以看到整个棋盘，并检测到棋盘角点。点击“capture now”开始抓取棋盘图像。

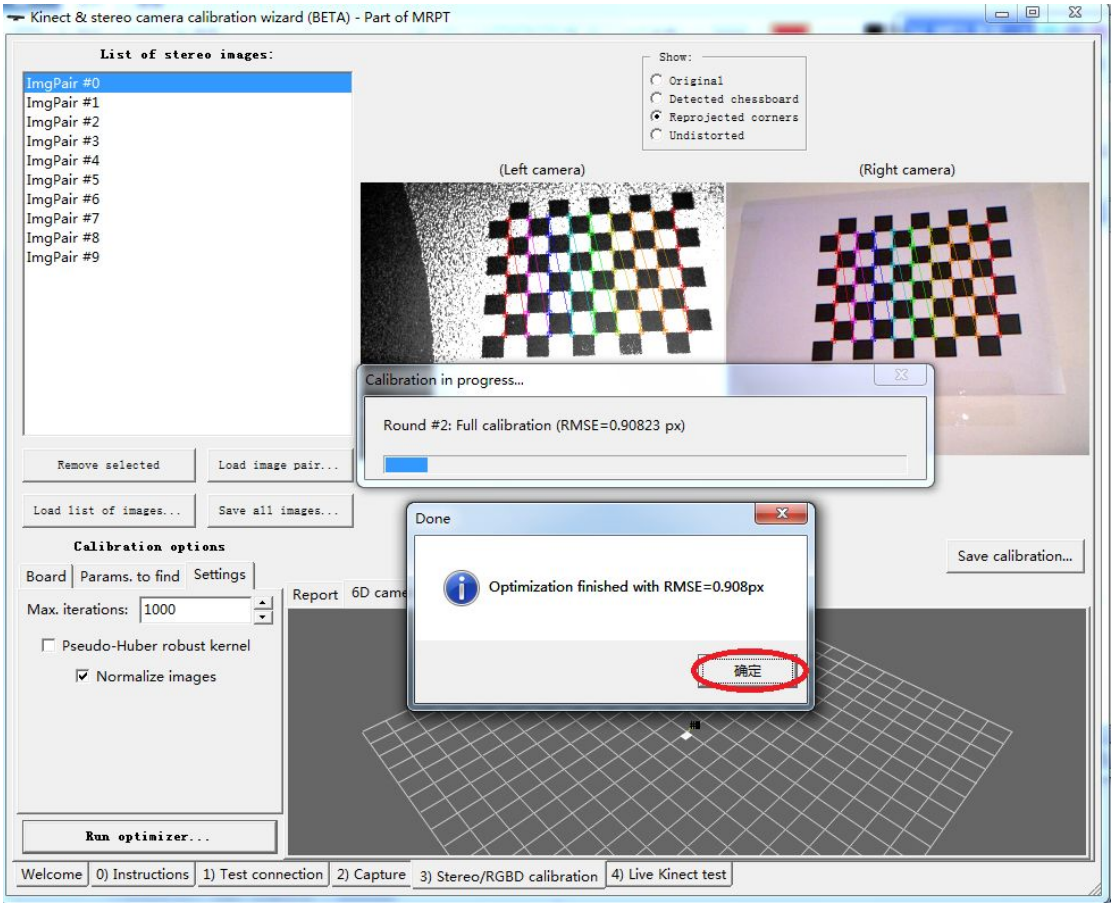


在此过程中，缓慢变换棋盘的倾角和距离，尽可能覆盖比较极端的范围和角度。抓取约 20 张左右的图片即可，点击“ok,go to calibrate”进入校正界面。



- List of stereo images（立体图片对 清单）：
  - Remove selected ：删除选中的图片
  - Load image pair ：载入图片对
  - Load list of images：加载图片清单
  - Save all images：保存所有图片
- Calibrate options（校正选项）：
  - Board
    - Number of inner corners：X 和 Y 坐标方向的内角数量
    - Size of quads：四边形的长度，以毫米为单位
- Calibration results（校准结果）:
  - 6D camera poses：摄像头的 6D 位姿
  - Save calibration：保存校准结果
- Show:
  - Original 原始图像
  - Detected chessboard 检测到棋盘点图像
  - Reprojected corners 重投射角点
  - Undistorted 不失真图像

点击“Run optimizer”开始校正。



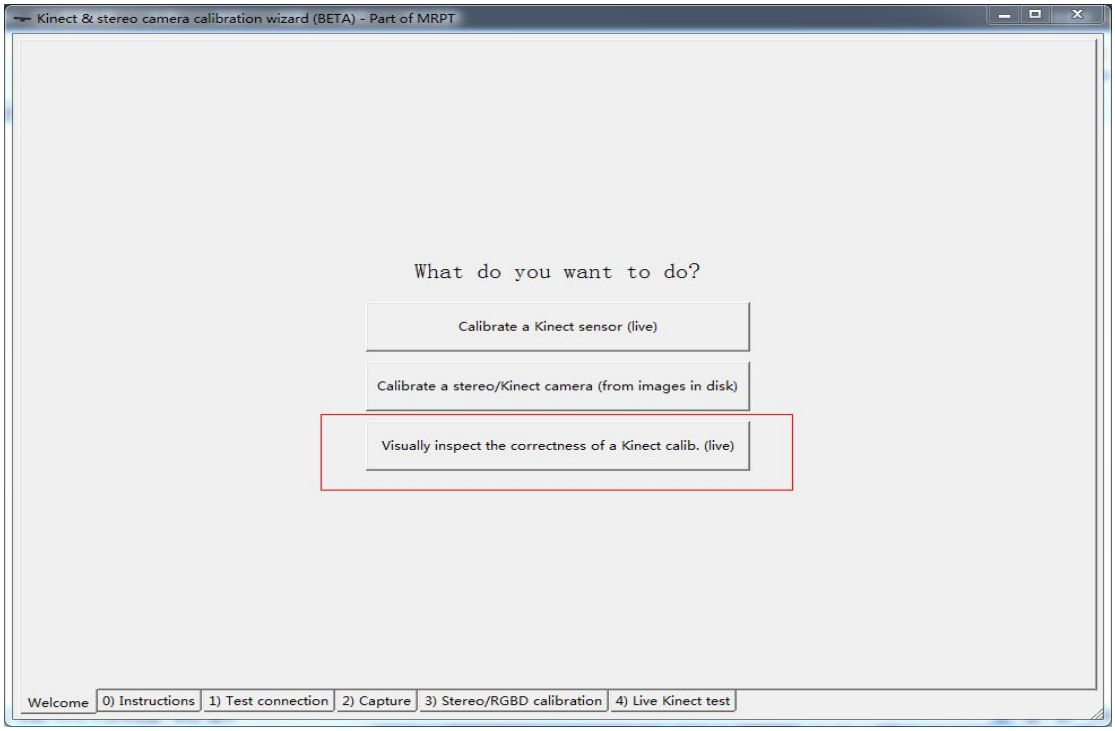
校正完成后，点击“Save calibration”，保存为 calib.ini 文件。文件内容如下：主要是 IR 摄像头和 RGB 摄像头的 intrinsic 和 extrinsic 矩阵参数，以及两摄像头之间的相对位姿。

```
# Left camera (IR/Depth in Kinect) calibration parameters (and 95% confidence intervals):
[CAMERA_PARAMS_LEFT]
resolution = [640 488]
cx              = 337.512109    // +/- 0.019
cy              = 207.624170    // +/- 0.019
fx              = 613.009846    // +/- 0.097
fy              = 601.633895    // +/- 0.144
dist            = [-2.687047e-003 2.487469e-004 0.000000e+000 0.000000e+000 0.000000e+000]
   // The order is: [K1 K2 T1 T2 K3]

# Right camera (RGB in Kinect) calibration parameters    (and 95% confidence intervals):
[CAMERA_PARAMS_RIGHT]
resolution = [640 480]
cx              = 300.993009    // +/- 0.019
cy              = 244.788006    // +/- 0.019
fx              = 537.880026    // +/- 0.109
fy              = 528.542313    // +/- 0.117
dist            = [6.335599e-002 2.619518e-002 0.000000e+000 0.000000e+000 0.000000e+000]
  // The order is: [K1 K2 T1 T2 K3]
```

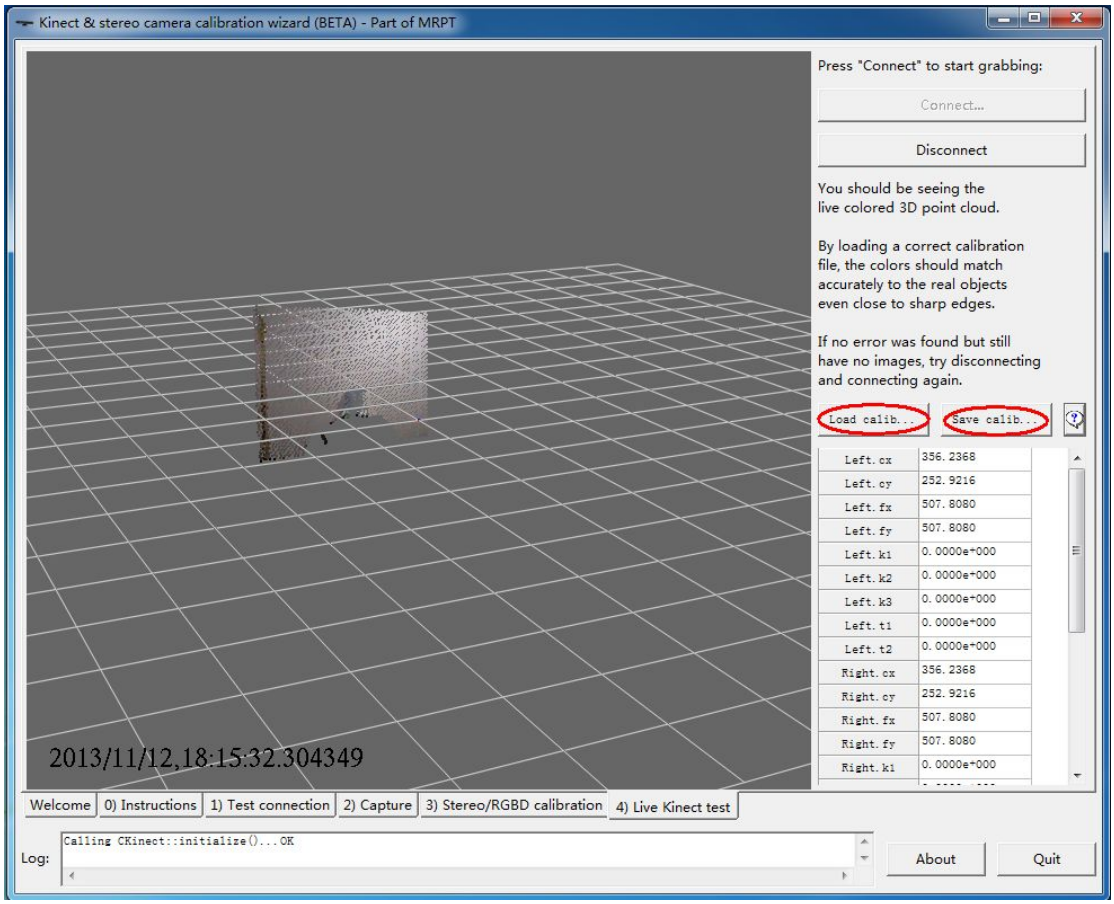
```
# Relative pose of the right camera wrt to the left camera:
# This assumes that both camera frames are such that +Z points
# forwards, and +X and +Y to the right and downwards.
[CAMERA_PARAMS_LEFT2RIGHT_POSE]
translation_only = [5.496772e-002 -1.573980e-003 1.606786e-002]
rotation_matrix_only = [9.9972114716529e-001 -1.0490458542493e-002
-2.1156043819129e-002 ;1.1462080751737e-002 9.9885993034612e-001
4.6340697597177e-002 ;2.0645789288602e-002 -4.6570267644932e-002
9.9870163790600e-001 ]
pose_yaw_pitch_roll = [0.054968 -0.001574 0.016068 0.656883 -1.183001 -2.669815] //
(YPR in degrees)
pose_quaternion = [0.054968 -0.001574 0.016068 0.999660 -0.023236 -0.010454
0.005490]
```

9.3 应用校正结果

关闭应用程序后重新打开，在首界面点击“visually inspect the correctness of a kinect calib”按钮，应用校正参数，检验校正效果。



加载刚刚生成的 calib.ini 文件，导入 kinect 校正参数，可以看到校正结果。也可以在此界面微调参数中的某个值，并将最终参数保存。


Kinect & stereo camera calibration wizard (BETA) - Part of MRPT
Press "Connect" to start grabbing:
Connect...
Disconnect
You should be seeing the live colored 3D point cloud.
By loading a correct calibration file, the colors should match accurately to the real objects even close to sharp edges.
If no error was found but still have no images, try disconnecting and connecting again.
Load calib...
Save calib...
Left.cx 356.2368
Left.cy 252.9216
Left.fx 507.8080
Left.fy 507.8080
Left.k1 0.0000e+000
Left.k2 0.0000e+000
Left.k3 0.0000e+000
Left.t1 0.0000e+000
Left.t2 0.0000e+000
Right.cx 356.2368
Right.cy 252.9216
Right.fx 507.8080
Right.fy 507.8080
Right.k1 0.0000e+000
2013/11/12,18:15:32.304349
Welcome
0) Instructions
1) Test connection
2) Capture
3) Stereo/RGBD calibration
4) Live Kinect test
Log:
Calling CKinect::initialize()...OK
About
Quit

第十章 camera-calib(Camera intrinsic calibration)摄像头校正

10.1 描述

这个 GUI 程序允许用户通过抓取一些棋盘图像计算出一个摄像头的参数，可以实时抓取图像或者是选择之前存储的图像文件。它还可以显示重投射点，无畸变图像和重构的相机位姿。

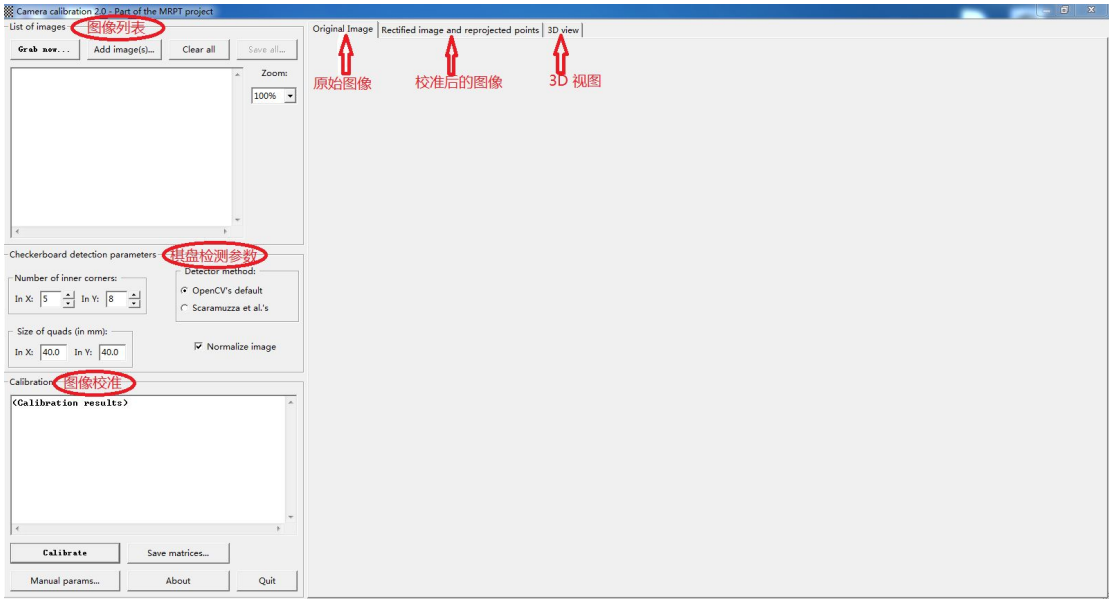
支持的图像源：

- 所有 OpenCV 支持的摄像头（USB 摄像头，Fireware 摄像头...）
- 所有 FFmpeg 支持的摄像头（IP 摄像头）
- 视频文件（任何格式）
- Rawlog 文件（mrpt 指定的一种机器人数据集格式）
- 立体 Bumblebee 摄像头，一次校正其中一个摄像头。
- SwisssRanger ToF 3D 摄像头的深度通道。
- Kinect 的 RGB 摄像头通道和 IR 红外摄像头通道。

你需要打印一张棋盘用于校正，可以在下面链接下载：
camera-calibration-checker-board_9x7.pdf - 9×7 (较高质量，较清晰边界)
注意：在 Linux 下，你可能需要执行 chmod 666 /dev/video1394* 和 chmod 666 /dev/raw1394 以便允许用户 R/W 读写访问 fireware 摄像头，不需要超级用户权限。

【本章以下部分是由译者添加，不保证绝对准确】

10.2 界面介绍



List of image（图像列表）：

- Grab new：获取棋盘校准图像。支持 Camera(opencv)、Camera(FFmpeg)、Camera(custom)、Video file 、Rawlog file、 Bumblebee、 SwissRanger TOF、 Kinect 图像源
- Add image(s)：从文件中选择图像
- Clear all：清除所有图像
- Save all：保存所有图像
- Zoom：图像缩放比例

Checkerboard detection parameters(棋盘检测参数)

- Number of inner corners：X 和 Y 坐标方向的内角数量
- Size of quads：四边形的长度，以毫米为单位
- Detector method ：默认为 OpenCV，还可以选择 Scaramuzza
- Normalize image：图像标准化处理

Calibration：校正

- Calibrate：点击 Calibrate 按钮，图像开始进行校正
- Save matrices：保存生成的矩阵
- Manual params：手动参数
- About：相关介绍
- Quit： 退出程序

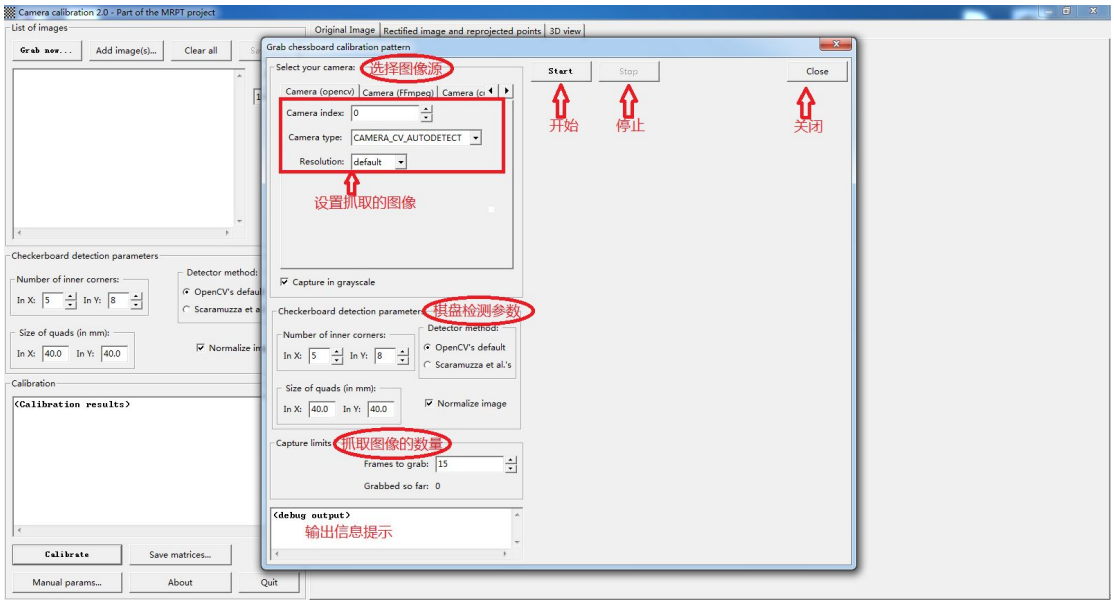
Original Image：原始图像

Rectified image and reprojected points：校正后图像和重投射点

3D View：3D 视图

10.3 校正摄像头过程

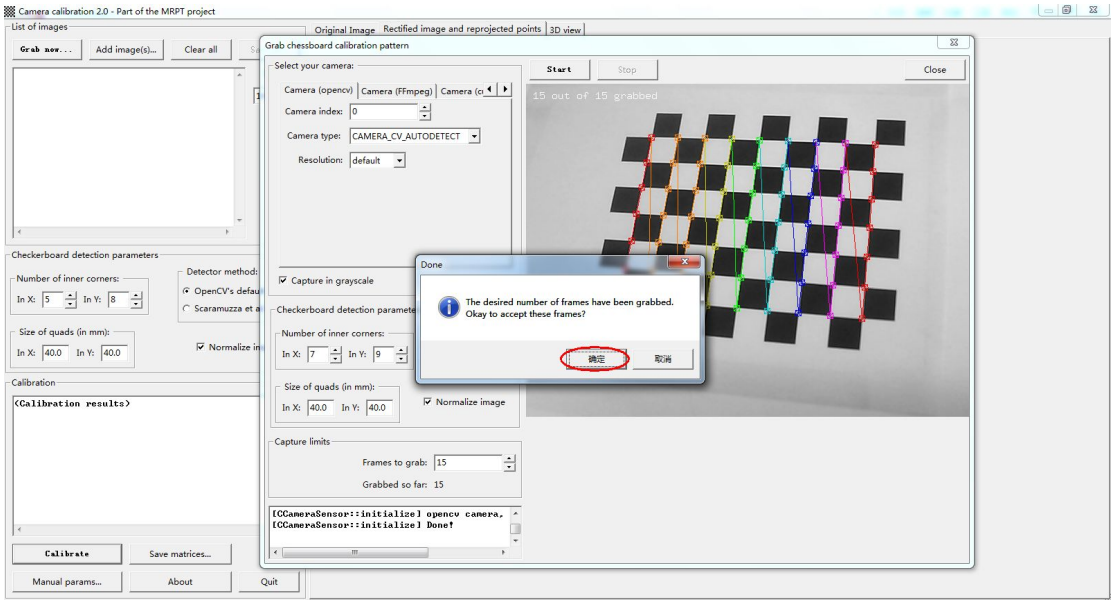
有两种方法获得待校正图像组。第一种方法是：点击“Add image”按钮，从本地文件中加载图片；第二种方法是：点击“Grab new”按钮，从摄像头抓取图像。



选择合适的图像源（这里选择的是 Camera（opencv））后，根据实际情况设置图像参数、棋盘检测参数和获取图像的数量，然后单击“Start”。

在抓取图像过程中，缓慢变换棋盘的角度、距离。使抓取到的图像尽可能清晰，并包括

多种角度状态。这样才能保证校正结果尽可能精确。



抓取完成后，点击“calibrate”开始校正计算，完成后点击“save matrics”保存摄像头校正矩阵到文件中。

生成的文件中是经典的单目摄像头校正矩阵：

- intrinsic_matrix(3X3):

```
9.7295757358689787e+002 0.0000000000000000e+000 3.0243269143764650e+002
0.0000000000000000e+000 9.7059777738372236e+002 2.1709517965559016e+002
0.0000000000000000e+000 0.0000000000000000e+000 1.0000000000000000e+000
```

- distortion_matrix(1X4):

```
-1.0806983471693088e-001 -8.7097085499241034e-001 7.3393774092493300e-003
-1.2111992918481763e-003
```

第十一章 Features-matching 特征点匹配

11.1 描述

这个应用程序是一个用于对两张图片进行特征点探测和匹配测试的小工具，用户可以选择多种特征点探测算法。根据用户指定的特征点描述子，程序会对给定图片计算特征点并建立匹配关联，并显示出所有潜在匹配对。

特征点探测器和描述子都是在 C++类 mrpt::vision::CFeatureExtractor 中实现的。已经实现了下面这些算法：

- 探测器：
  - KLT
  - Harris
  - SIFT
  - SURF
  - FAST
- 描述子：
  - SIFT
  - SURF
  - Intensity spin images
  - Polar coordinates image patch.

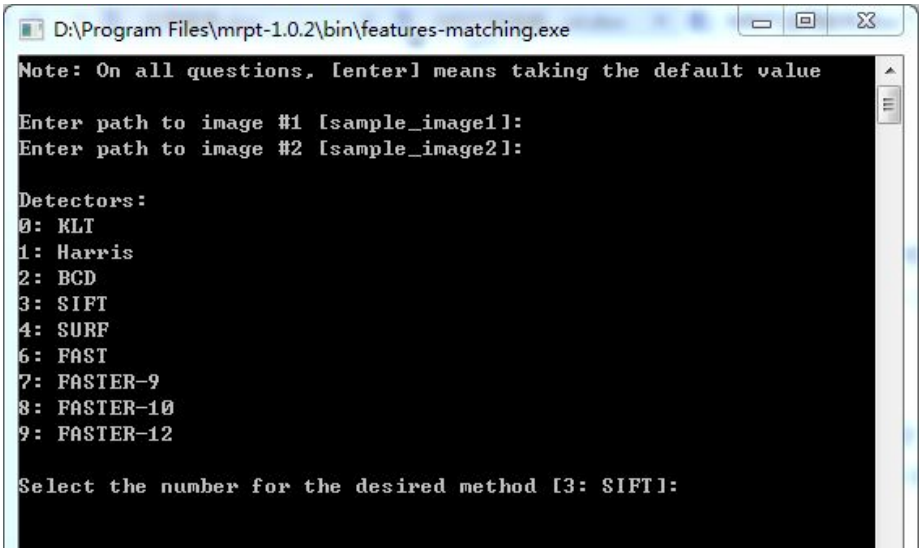
程序运行后，会显示一张图片中的每个特征点和另一张图片中的所有特征点之间的“匹配”结果。

需要注意的是，这个程序只是一个简单演示，没有考究效率问题。

11.2 使用

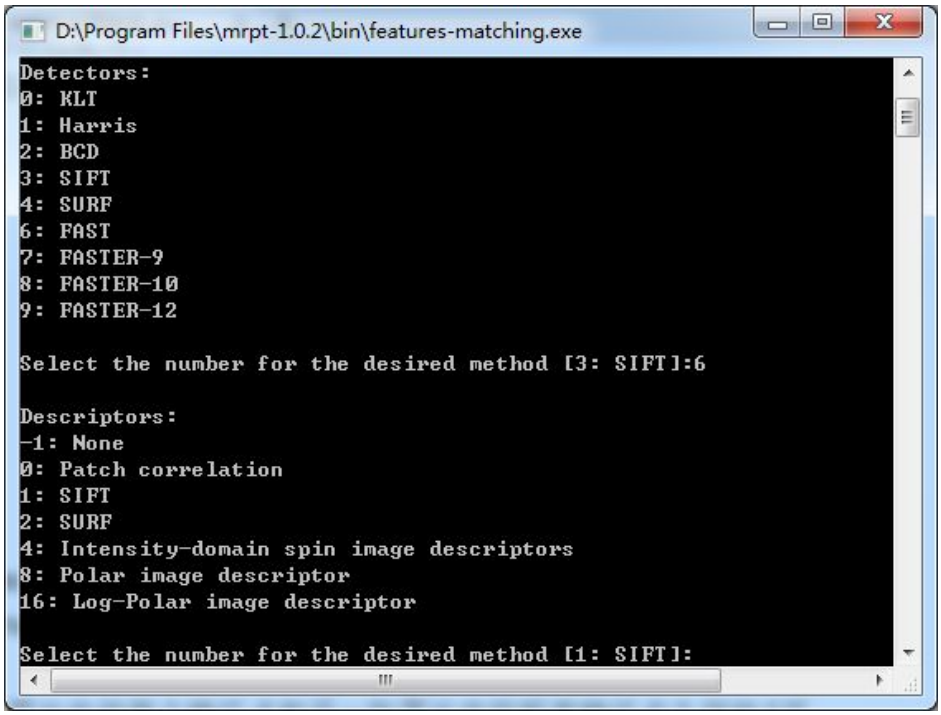
双击 features-matching.exe 运行后，会弹出命令行终端窗口，要求用户输入两张待匹配图片的路径，如果用户不输入图片路径，程序将使用（内置）默认图片。

然后选择特征探测器算法：

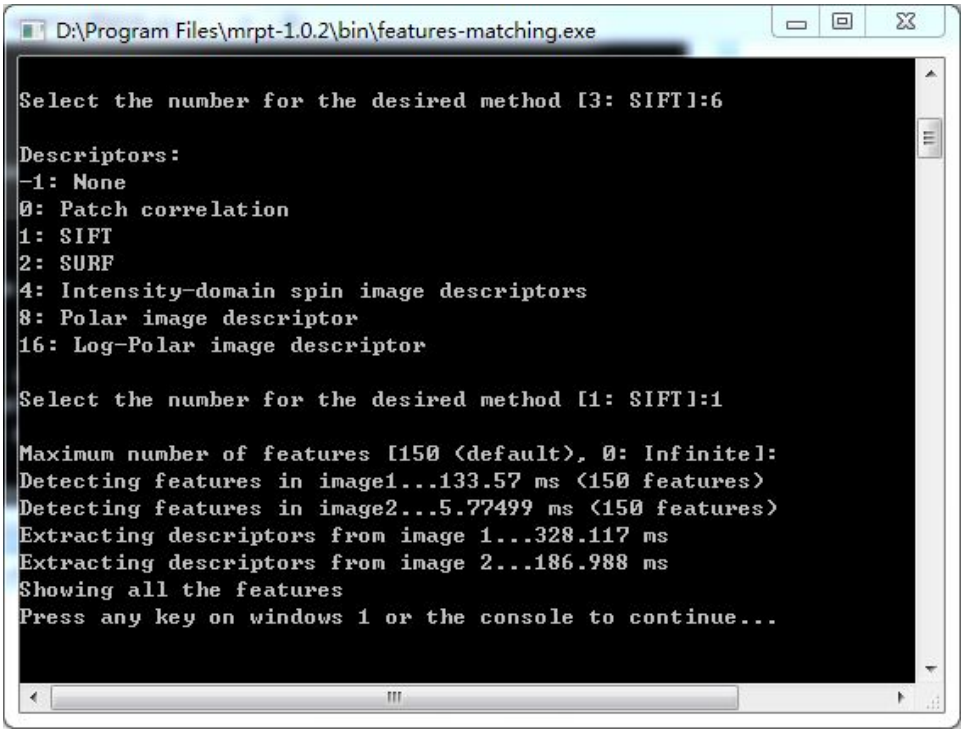


【本章以下部分是由译者添加，不保证绝对准确】

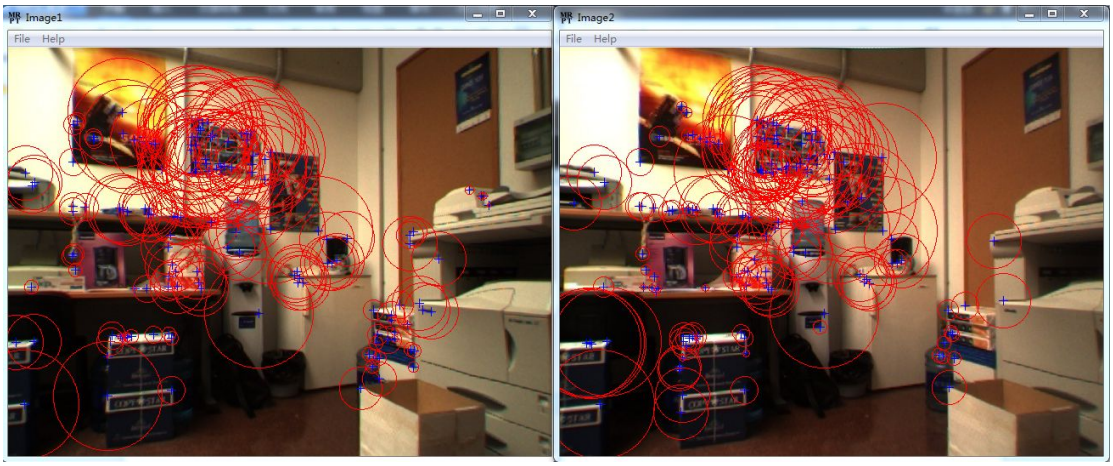
在这里以选择 6 FAST 探测器为例，输入 6 后回车。程序要求选择描述器算法。



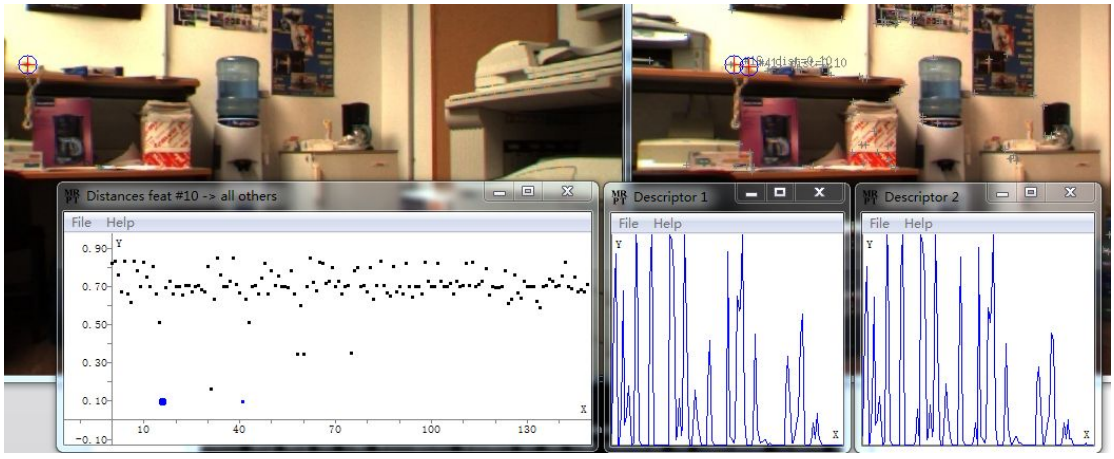
在这里以选择 1 SIFT 描述子为例，输入 1 后回车。再输入限定的特征点数量，默认 150 个，输入 0 表示不限制。输入后开始计算特征点，可以看到第一张图片计算时间约 133ms，时间比较长，效率不能达到实时的效果。第二张图片计算时间只有约 5ms，这是因为第二张图片计算时会利用第一张图片已经计算出来的特征点位置，限制在其附件区域查找，因此计算量锐减计算耗时锐减。



计算结束后会显示两个窗口，分别是两个图片的特征点探测结果标识，蓝十字标识了特征点位置，红色圆圈标识尺度。



继续在命令行终端窗口按任意键，开始一次特征点匹配过程，并弹出 3 个新窗口，一个是匹配到的特征点对应图片上所有特征点的匹配距离（数值越小匹配可信度越高）图表。两外两个小窗口是两张图片的描述子计算结果。



可看到这次匹配结果中，针对左图上的一个特征点，右图上匹配了两个特征点，其中有一个是误匹配的。

用户可以继续在命令行终端按任意键，再开始一次特征点匹配过程。查看不同特征点的匹配对结果。

第十二章 Track-video-features 跟踪视频特征点

12.1 描述

这个应用用于演示在视频流中实时探测和跟踪特征点。

可以直接双击“track-video-features.exe”运行应用程序。或者在命令行终端输入命令：

```
$ track-video-feature
```

弹出一个 GUI 窗口供用户选择视频来源，支持下列视频源：

- OpenCV 支持的摄像头（USB 摄像头，1394 摄像头）
- FFmpeg 支持的网络 IP 摄像头
- 视频文件（比如 avi, mpg, mov 格式）
- Rawlog 文件
- 双目摄像头 Bumblebee
- SwissRangerTOF 的 3D 摄像头
- Kinect

也可以在命令行终端输入命令时带上视频源参数直接指定：

```
$ track−video−features   <DATASET.rawlog>
$ track-video-features    <video_file.{avi/mpg/mp4/flv}>
.....
```

The program is located at <MRPT>/apps/track-video-features/:

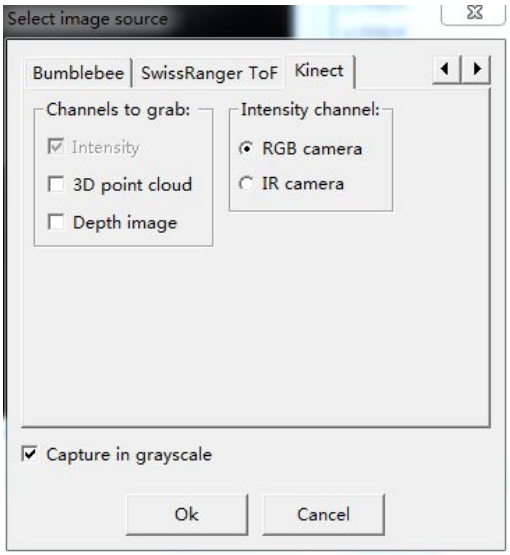
See this tutorial on the related **data structures and classes**.

Test with a sample video sequence published by Andrew Davison among his MonoSLAM works.

【本章以下部分是由译者添加，不保证绝对准确】

12.2 使用

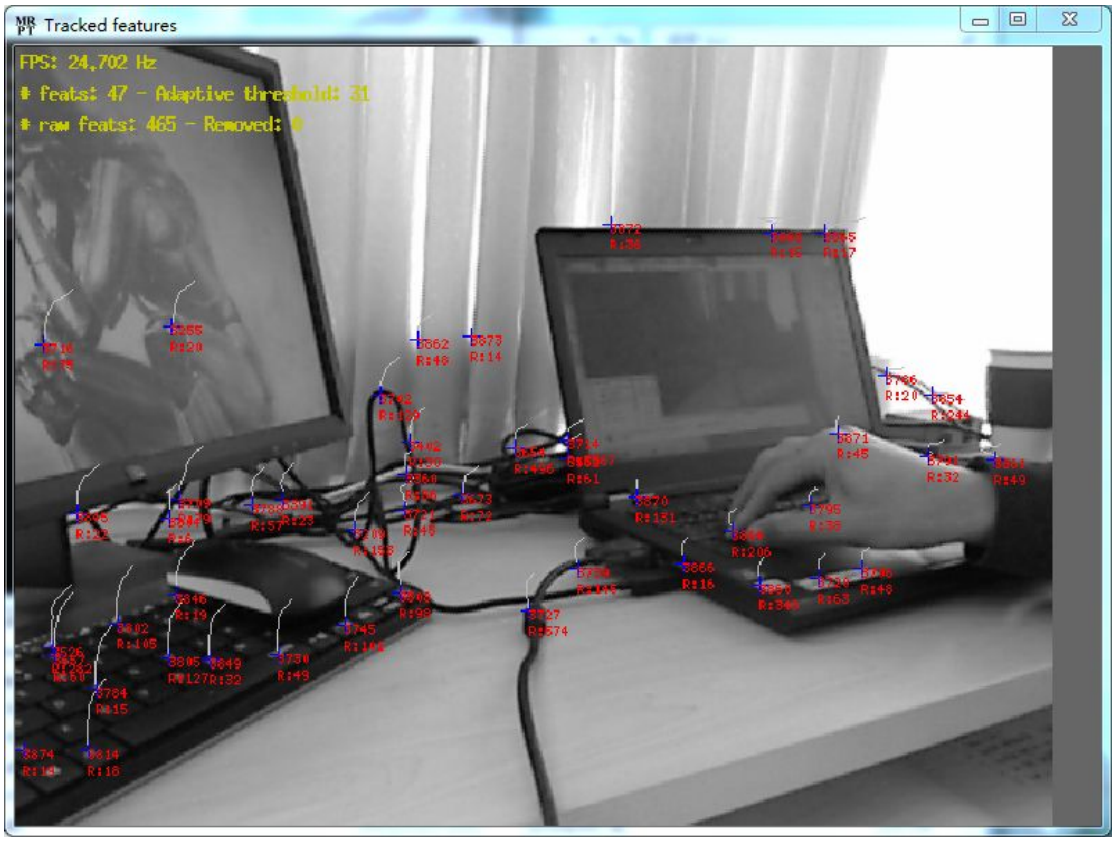
在这里以选择 kinect 作为视频源为例，在窗口中默认是选定 RGB 通道，灰度图。用户可选择是否抓取深度图像，或计算 3D 点云。



确认 kinect 已经连接好，确定后将打开 kinect 的 RGB 摄像头视频，因为之前选定了灰度图模式，所示显示灰度图像。

左上角为计算状态，帧率、自适应阈值、特征点数量。。。

特征点用蓝十字标识位置，红色字体上为特征点编号 ID，下为尺度半径。



源码在 mrpt 根目录的\apps\track-video-features 目录。

第十三章 ICP-slam 应用

13.1 描述

这个应用是对 MRPT C++库中的类 mrpt::slam::CMetricMapBuilderICP 的一个前端实现。这个 slam 算法使用 ICP 算法计算最新扫描观测值在点云地图或占据栅格地图中的对齐，实现增量建图。ICP 算法的实现在类 mrpt::slam::CICP 中。《Tutorial》中有详细介绍。

13.2 使用

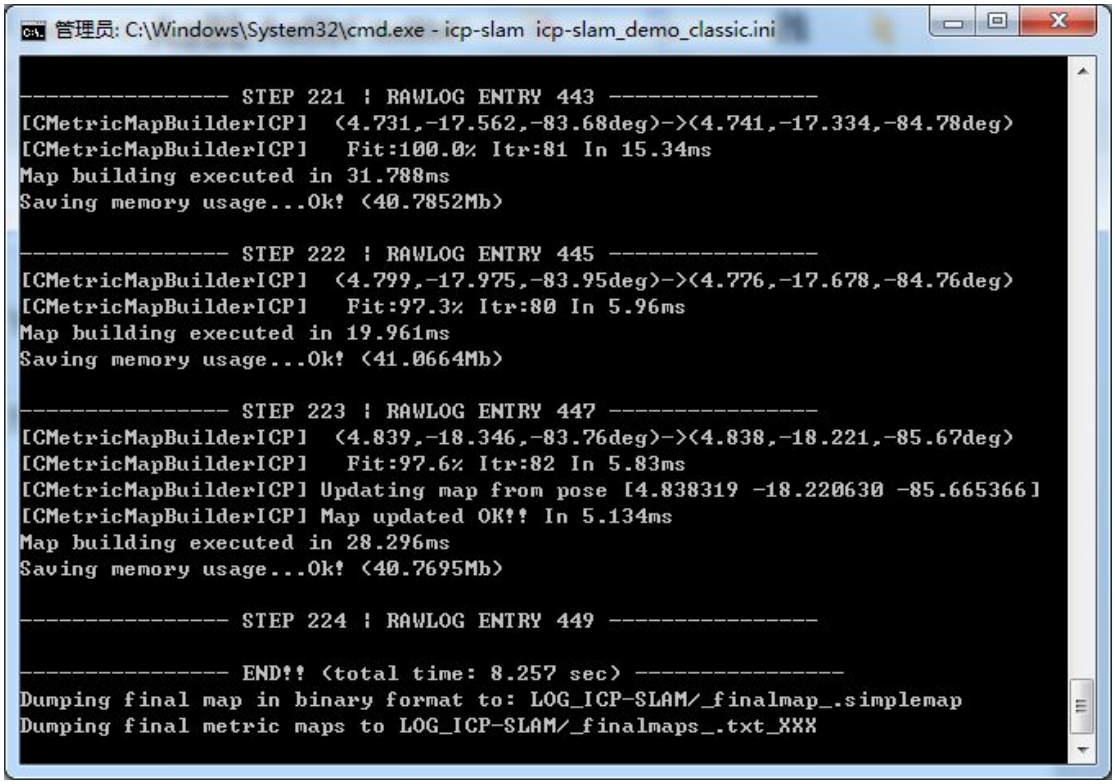
这个应用需要在命令行终端输入命令启动。在 mrpt 安装好后，在目录 MRPT\share\mrpt\config_files\icp-slam 下提供了本应用需要的 ini 配置文件模板。下面的示例使用的配置文件里已设定好 rawlog 文件路径，是 MRPT\share\mrpt\datasets 目录下的文件 2006-01ENE-21-SENA_Telecom Faculty_one_loop_only.rawlog：

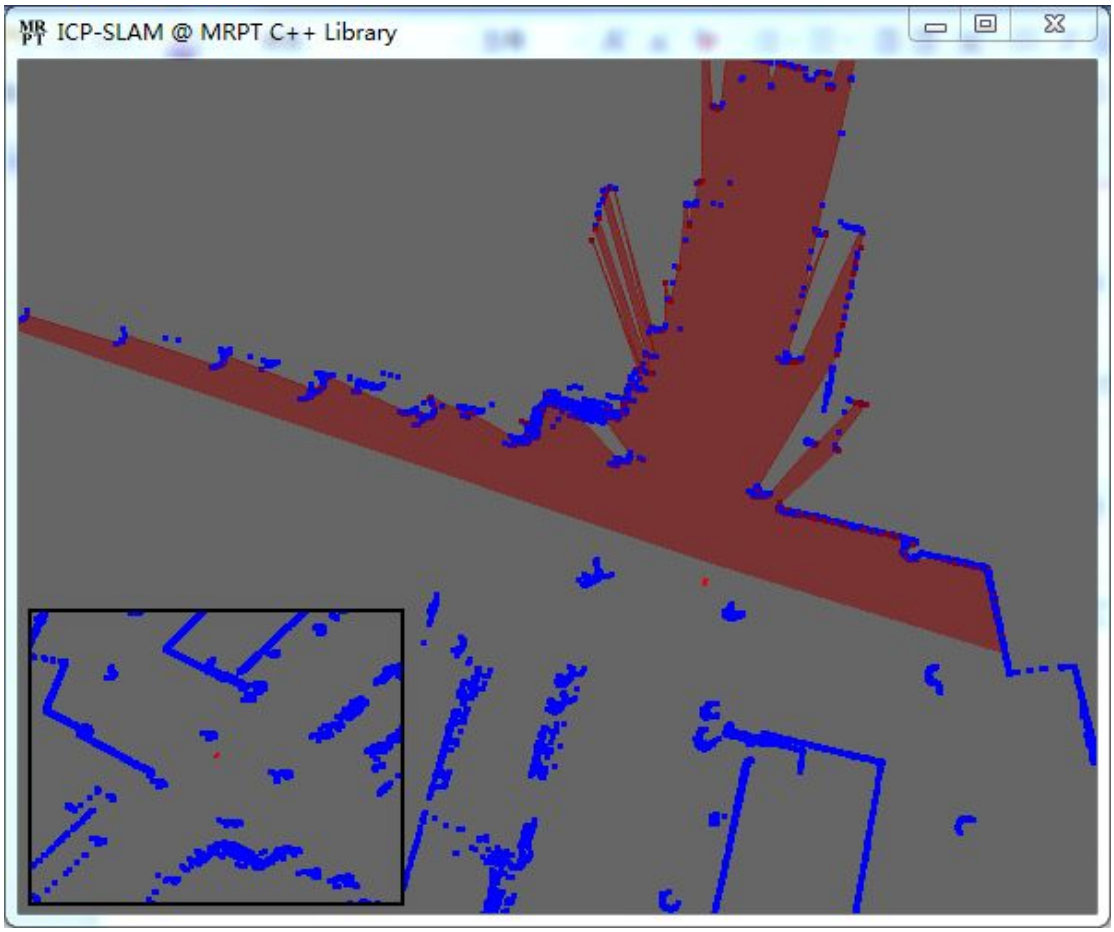
```
$ icp-slam    icp-slam_demo_classic.ini
```

或者用户可以使用自定义的配置文件和自己的 rawlog 数据集：

```
$ icp-slam    icp-slam_YOUR.ini    YOUR_DATASET.rawlog
```

运行后将直接开始 ICP-slam 计算过程，在命令行终端窗口显示过程数据，并在一个 3D 窗口中显示创建好的点云地图（可以修改配置文件以生成栅格地图）。





用户可以打开<MRPT>/share/mrpt/config_files/icp-slam/目录下的配置文件，查看ICP-slam 配置参数。注意 ICP 可以使用两种地图格式：点云地图和占据栅格地图。甚至可以同时创建这两种地图。

提示：在上面 demo 的配置文件 icp-slam_demo_classic.ini 中设定了 rawlog 数据集文件的位置，在<MRPT RootDir>\share\mrpt\datasets 目录下。

第十四章 RBPF-slam 应用

14.1 描述

RBPF-SLAM(Rao-Blackwellized Particle Filter SLAM)

这个“rbpf-slam”应用是对 MRPT C++库中的类 mrpt::slam::CMetricMapBuilderRBPF 的一个前端实现。算法需要的所有参数都是通过命令行中指定的配置文件设定。滤波器处理 rawlog 文件中的 actions 和 observations（数据），可选择生成一些描述滤波器和地图的文件。

RBPF-SLAM 的数学原理以及最新的 RBPF-SLAM 实现可以在《tutorial》里查阅。

14.2 使用

“rbpf-slam”应用是一个命令行程序，处理 rawlog 数据集文件并使用配置文件中指定的粒子滤波算法来建立地图。在 mrpt 安装好后，在/share/mrpt/config_files/rbpf-slam/目录下提供了本应用需要的 ini 配置文件。Rawlog 数据集文件可以使用上一节 ICP-slam 应用中使用的文件，需要确认配置文件中 rawlog_file 变量的值为下面这个路径值：rawlog_file=../../datasets/2006-01ENE-21-SENA_Telecom Faculty_one_loop_only.rawlog

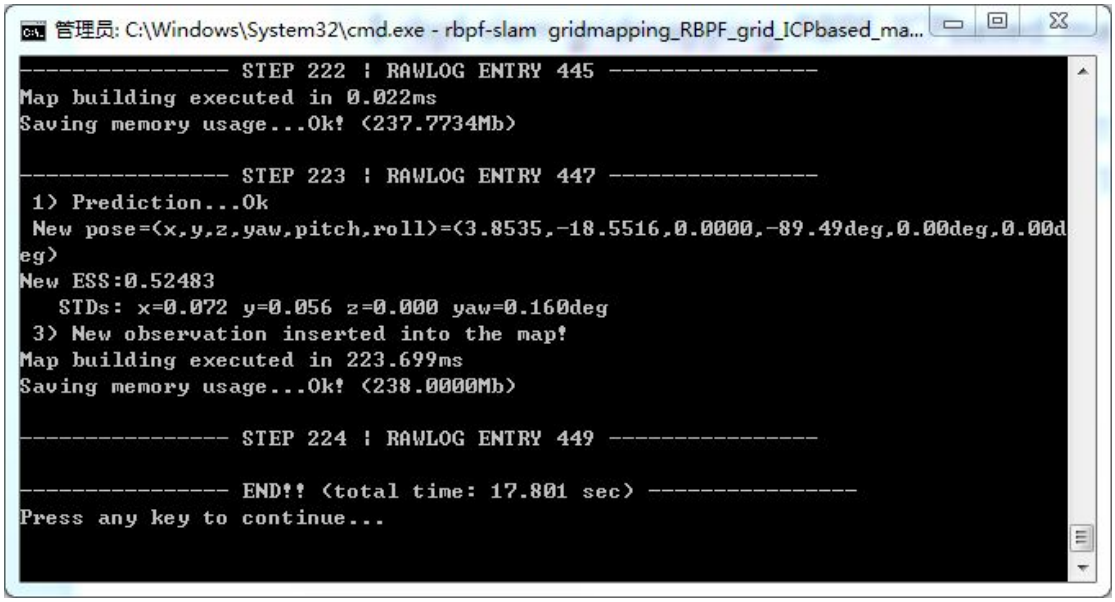
用户也可以指定自己的 rawlog 数据集文件。

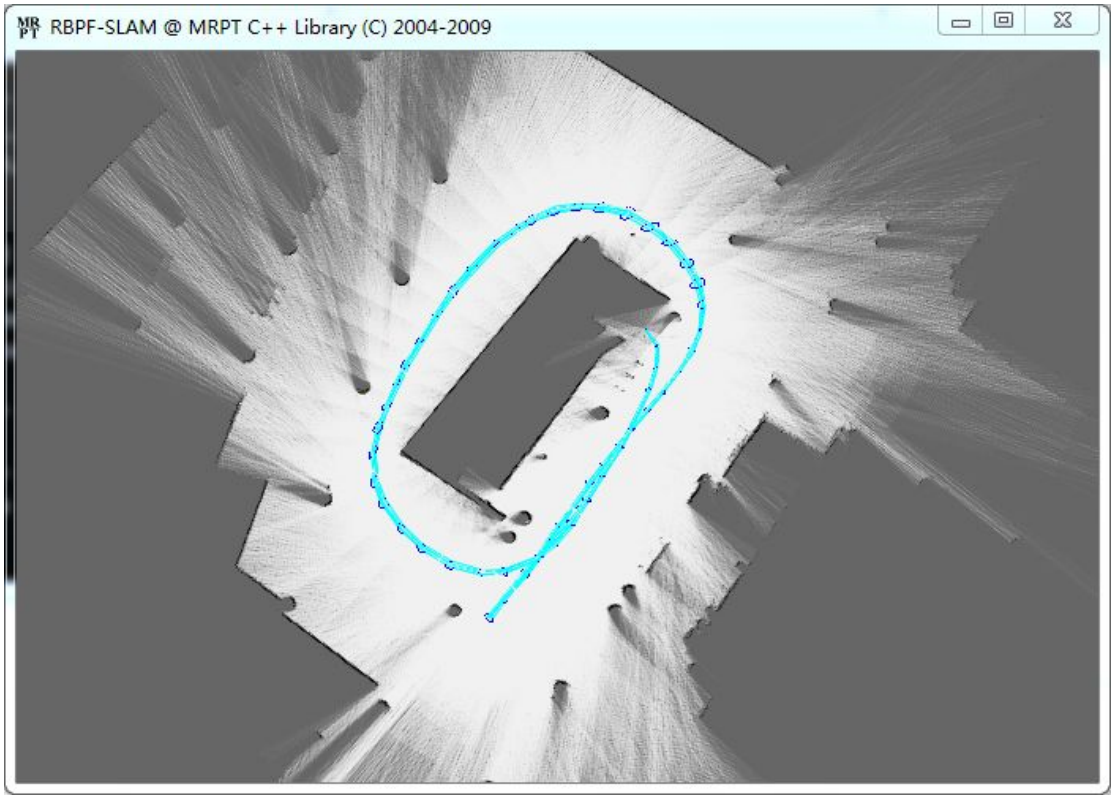
这里以使用 gridmapping_RBPF_grid_ICPbased_malaga.ini 配置文件为例。

打开命令行终端，在命令行运行下面命令：

```
$ rbpf-slam    gridmapping_RBPF_grid_ICPbased_malaga.ini
```

运行后将直接开始 rbpf-slam 计算过程，在命令行终端窗口显示过程数据，并在一个 3D 窗口中显示创建好的栅格地图（可以修改配置文件生成点云地图）。





用户可以打开<MRPT >/share/mrpt/config_files/rbpf-slam/目录下的配置文件，查看rbpf-slam 配置参数。注意 rbpf 可以使用两种地图格式：点云地图和占据栅格地图。甚至可以同时创建这两种地图。

第十五章 KF-slam 应用

15.1 描述

这个应用实现了一个简单的卡尔曼滤波器以解决使用3D扫描测距传感器实现SLAM的问题，并得到机器人完全6D状态空间。这个应用是MRPT C++库中的类mrpt::slam::CRangeBearingKFSLAM的一个前端实现。

算法需要的所有参数都是通过命令行中指定的配置文件设定。滤波器处理rawlog文件中的actions和observations（数据），可选择生成一些滤波器描述文件。

Kf-slam也可以用于解决2维的(Range-Brearing)方位问题，是一种在平面环境下更高效的解决slam问题的方法。

更多关于6D Range-Bearing SLAM的解释查阅相关论文。

15.2 使用

“kf-slam”应用是一个命令行程序，在配置文件中设定所有参数。在mrpt安装好后，在/share/mrpt/config_files/kf-slam/目录下提供了本应用需要的ini配置文件。演示用的Rawlog数据集文件可以在/share/mrpt/dataset/目录下找到。在用户也可以指定自己的rawlog数据集文件。

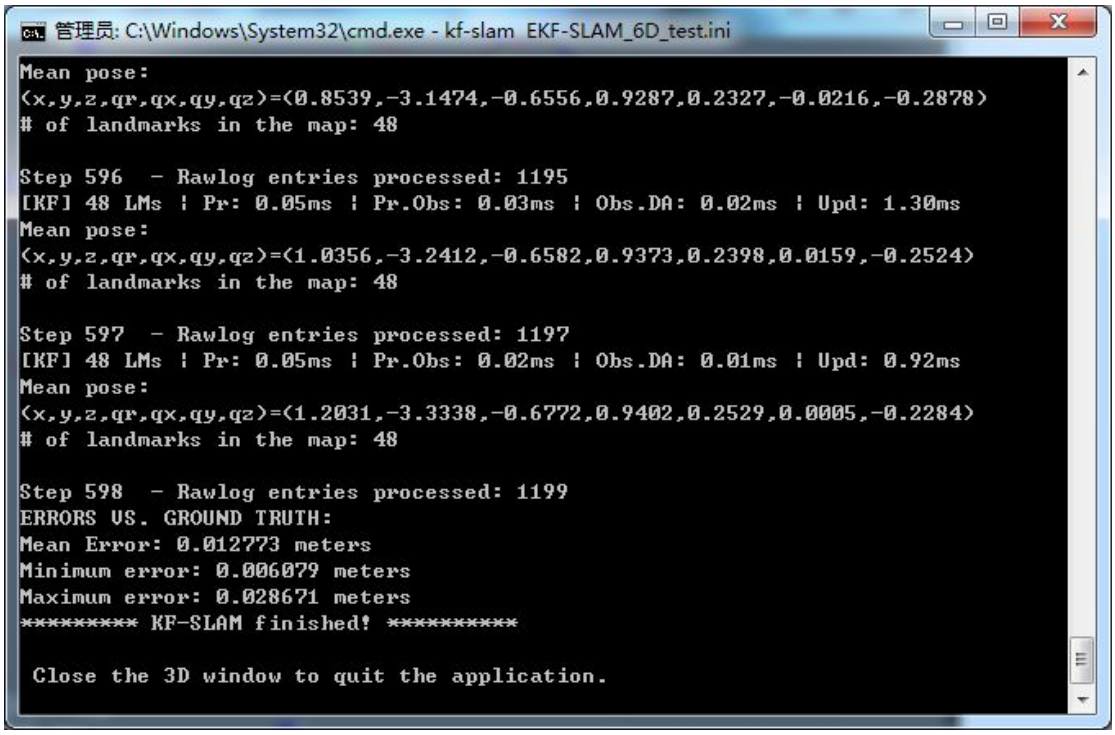
在配置文件中必须设定rawlog-file文件。可以选择设定ground_truth_file文件和ground_truth_file_robot文件，作为误差比较基准数据集。

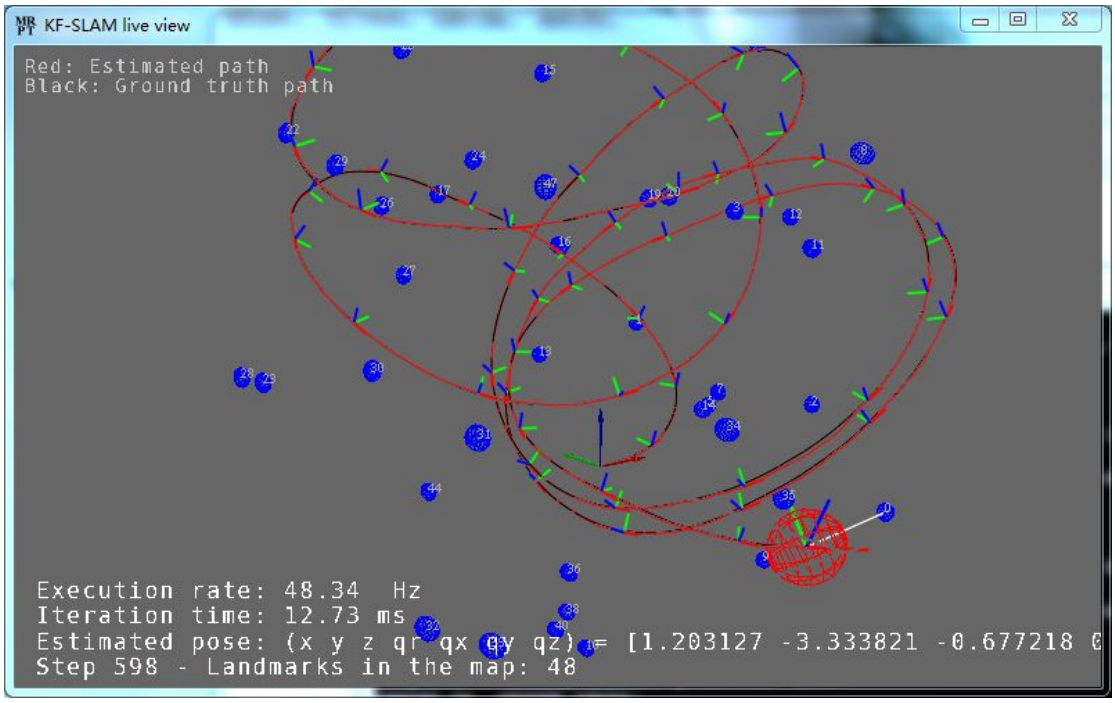
这里以使用EKF-SLAM_6D_test.ini配置文件为例，Rawlog文件以使用kf-slam_6D_demo.rawlog为例。

打开命令行终端，在命令行运行下面命令：

```
$ kf-slam    EKF-SLAM_6D_test.ini
```

运行后将直接开始kf-slam计算过程，在命令行终端窗口显示过程数据，并在一个3D窗口中显示创建好的路标路径地图。





kf-slam 使用路标地图 rawlog 数据，用户可以使用应用“simul-landmarks”模拟生成。模拟器会生成一系列 3D 随机路标，并生成相距预设机器人路径的(Range-Brearing)方位噪声测量。

应用“simul-landmarks”的介绍见下一章。

第十六章 Simul-Landmarks 路标数据模拟器

16.1 描述

这个应用是用于模拟一个在 2D 或 3D(6D)路径上运动的机器人的(Range-bearing observations)方位观测值。这个模拟器生成的合成数据集可以用于 2D“经典”方位(Range-bearing)SLAM 和 3D 立体 SLAM 问题。

路标的位置和数量都是可以通过配置文件配置的，而且机器人路径可以是固定的也可以是随机游走(random walk)。用于路径和传感噪声的随机种子都可以被设置，因此不同的研究者可以使用同一个配置文件精确再现数据集。

生成的 rawlog 数据集中包含 mrpt::slam::CObservationBearingRange 类数据。

16.2 使用

这个应用是一个命令行程序，通过制定配置文件导入参数。在 mrpt 安装好后，在/share/mrpt/config_files/simul-landmarks/目录下提供了本应用需要的 ini 配置文件。这里以使用 simul_landmarks_demo_EKF-SLAM_in_6D.ini 配置文件为例：

打开命令行终端，在命令行输入下面命令：

```
$ simul-landmarks simul_landmarks_demo_EKF-SLAM_in_6D.ini
```

将开始模拟计算路标路径数据，输出文件保存在同目录下的/OUT 文件夹内。这些 rawlog 文件可以用于 kf-slam。同时在 3D 窗口显示路标和路径。