1. 概括

2. 源代码文件所名

```
.
├── CMakeLists.txt
├── CMakeLists.txt.user
├── configdialog.cpp “设置”对话框源文件
├── configdialog.h “设置”对话框头文件
├── configdialog.ui “设置对话框”UI文件
├── helpdialog.cpp “帮助”对话框
├── helpdialog.h
├── helpdialog.ui
├── main.cpp “main”函数
├── mainwindow.cpp “主界面”
├── mainwindow.h
├── mainwindow.ui
├── rebulider.cpp “点云重建”
├── rebulider.h
├── recognizer.cpp “物体识别”
├── recognizer.h
├── slam.cpp “ORB-SLAM”
└── slam.h
```

1) ReBulider 点云重建

主要函数：

```
void processNewDepthFrame(cv::Mat depthImg, Sophus::SE3f pose);
```

说明：当主线程完成SLAM估计之后，会将得到的位姿估计结果和深度图像以信号的形式发送到点云重建线程。在上面的函数里，需要对点云作处理。

2) ReCongnizer 物体识别

主要函数：

```
void signalProcessColorFrameFinished(cv::Mat dstImg);
```

当完成一帧图像的识别后，以信号的方式发送到主线程。主线程接收的到后，又将新的一帧图像发送过来，进而通过下面的函数作处理。这样做的目的是考虑到速度较慢，识别不能实时完成。

```
void processNewColorFrame(cv::Mat srcImg);
```

3) Slam ORB-SLAM

主要函数：

```
void createSlamSystem(ConfigDialog *config);
```

说明：收到主线程开始SLAM的信号后，进行SLAM 系统对象的创建。这一步需要加载配置文件、词袋文件，故速度较慢，所以放到子线程里。创建完成后，通过下面的信号通知主线程。

```
void createSlamSystemFinished(float scale);
```

主线程收到SLAM系统创建完成的信号后，开始往slam系统里输入数据帧。下面的函数对其进行处理。

```
void processNewFrame(cv::Mat colorImg, cv::Mat depthImg, double tframe);
```

处理完成后，通过下面的函数通知主线程。主线程进而发送新的数据帧。

```
void processNewFrameFinished(Sophus::SE3f pose);
```

3. 注意

每一个处理过程都已放入到对应的线程中去，你需要作的是：

- 点云重建：实现***processNewDepthFrame***函数；
- 物体识别：实现***processNewColorFrame***函数；
- ORB-SLAM：在***processNewFrame***函数里实现窗口的生成，通过写代码重新绘制或者修改ORB-SLAM源代码来调用。