CLion是以IntelliJ为基础，专为开发C及C++所设计的跨平台IDE，可以在Windows、Linux及MacOS使用，我是在双系统下 ubuntu20.04下 演示的

linux平台clion安装

- 下载

1. 官网下载 Download CLion: A Smart Cross-Platform IDE for C and C++ (jetbrains.com)
2. wget https://download.jetbrains.8686c.com/cpp/CLion-2016.2.2.tar.gz

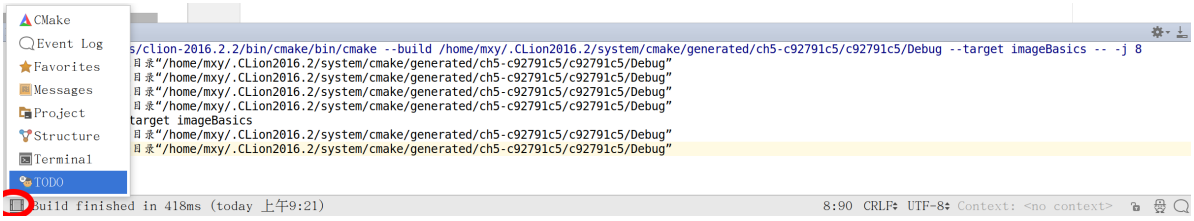
- 解压: `tar -zxvf CLion-X.tar.gz`
- 运行clion.sh脚本打开clion

```
cd clion-201X/bin/
./clion.sh
```

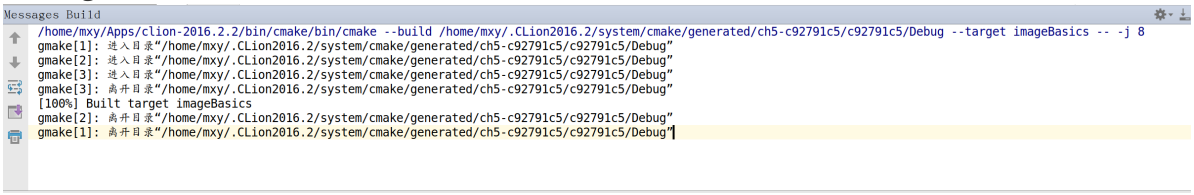
学校邮箱 申请免费账户

工具窗口工具栏

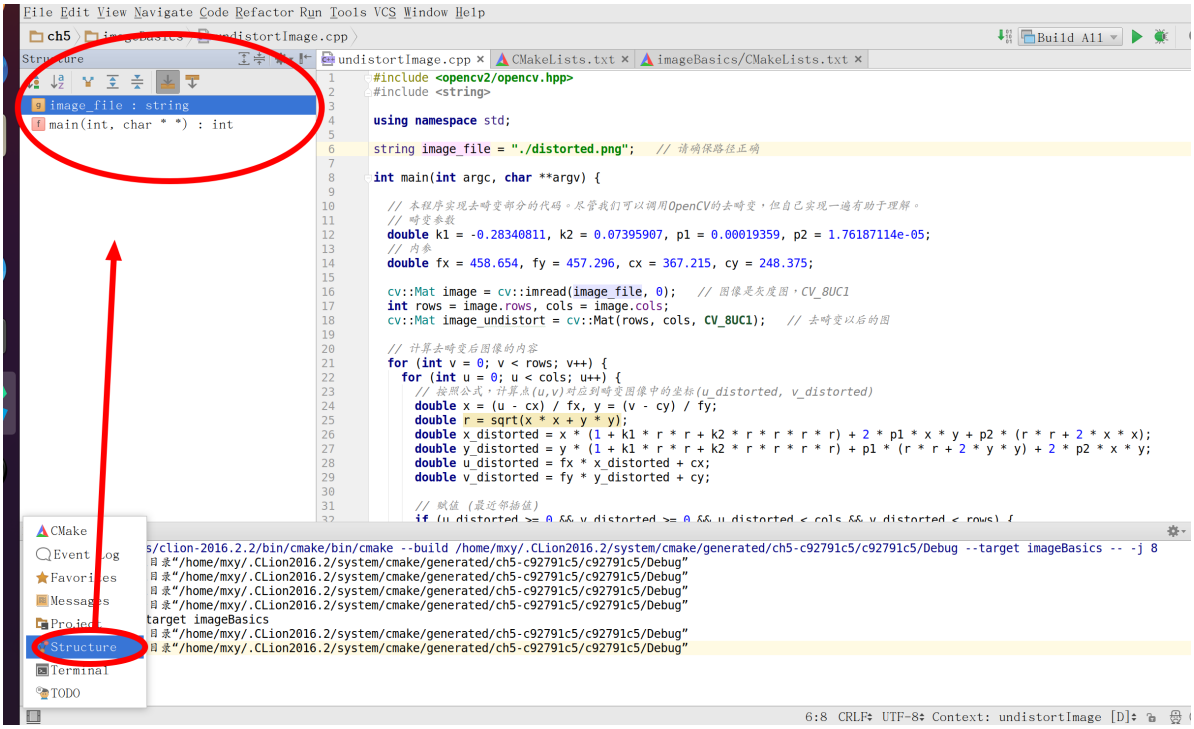
CLion工具窗口在工作区的底部和侧面。这些辅助窗口使您可以从不同角度查看项目，并提供对典型开发任务的访问。其中包括项目管理，源代码搜索和导航，运行和调试，与版本控制系统的集成以及许多其他特定任务。
鼠标悬停在此处，会打开一个菜单，可快速访问工具窗口;若单击该按钮，则会显示工窗口具栏，再次单击会隐藏。



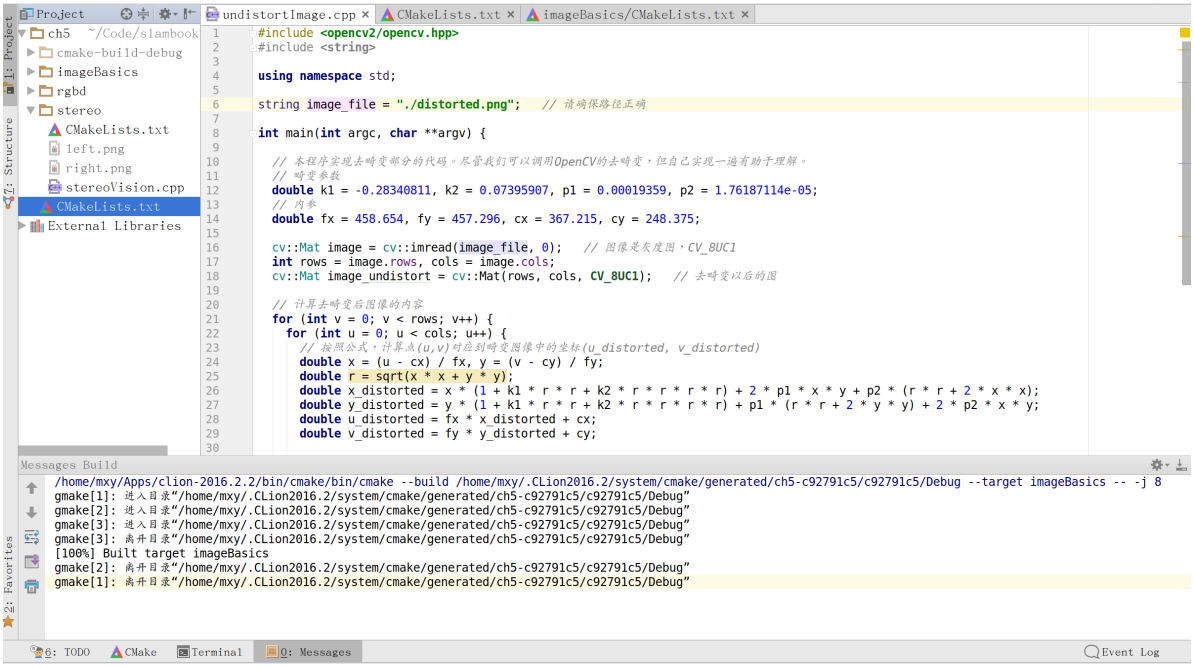
messages信息



structure

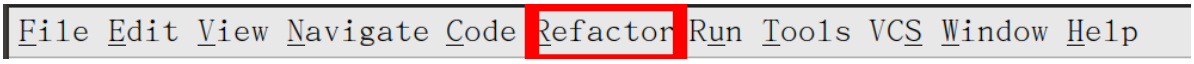


project



菜单栏

Refactor



Ctrl+Alt+V 自动赋值

refactor->extract->variable
把选中的语句自动赋值给一个变量

```
                // 根据双目模型计算 point 的位置
                double x = (u - cx) / fx;
                double y = (v - cy) / fy;
//                double depth = fx * b / (disparity.at<float>(v, u));
//                point[0] = x * depth;
//                point[1] = y * depth;
//                point[2] = depth;
//
                point[0] = x * (fx * b / (disparity.at<float>(v, u)));
                point[1] = y * (fx * b / (disparity.at<float>(v, u)));
                point[2] = (fx * b / (disparity.at<float>(v, u)));
                pointcloud.push_back(point);
            }
```

Ctrl+Alt+V 后

```
//
            point[0] = x * (fx * b / (disparity.at<float>(v, u)));
            point[1] = y * (fx * b / (disparity.at<float>(v, u)));
            point[2] = (fx * b / (disparity.at<float>(v, u)));
            pointcloud.push_back(point);
        }
```

Multiple occurrences found
Replace this occurrence only
Replace all 3 occurrences

Ctrl+Alt+P

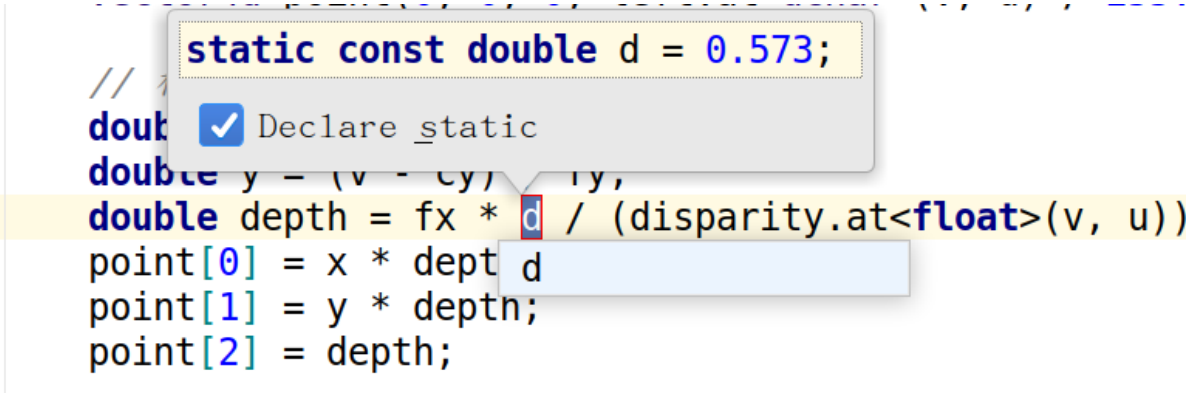
refactor->extract->parameters
将新参数加入到函数声明中

Ctrl+Alt+C 定义常量，便于修改参数

refactor->extract->constant

```
double depth = fx * 0.573 / (disparity.at<float>(v, u));
```

按下快捷键，

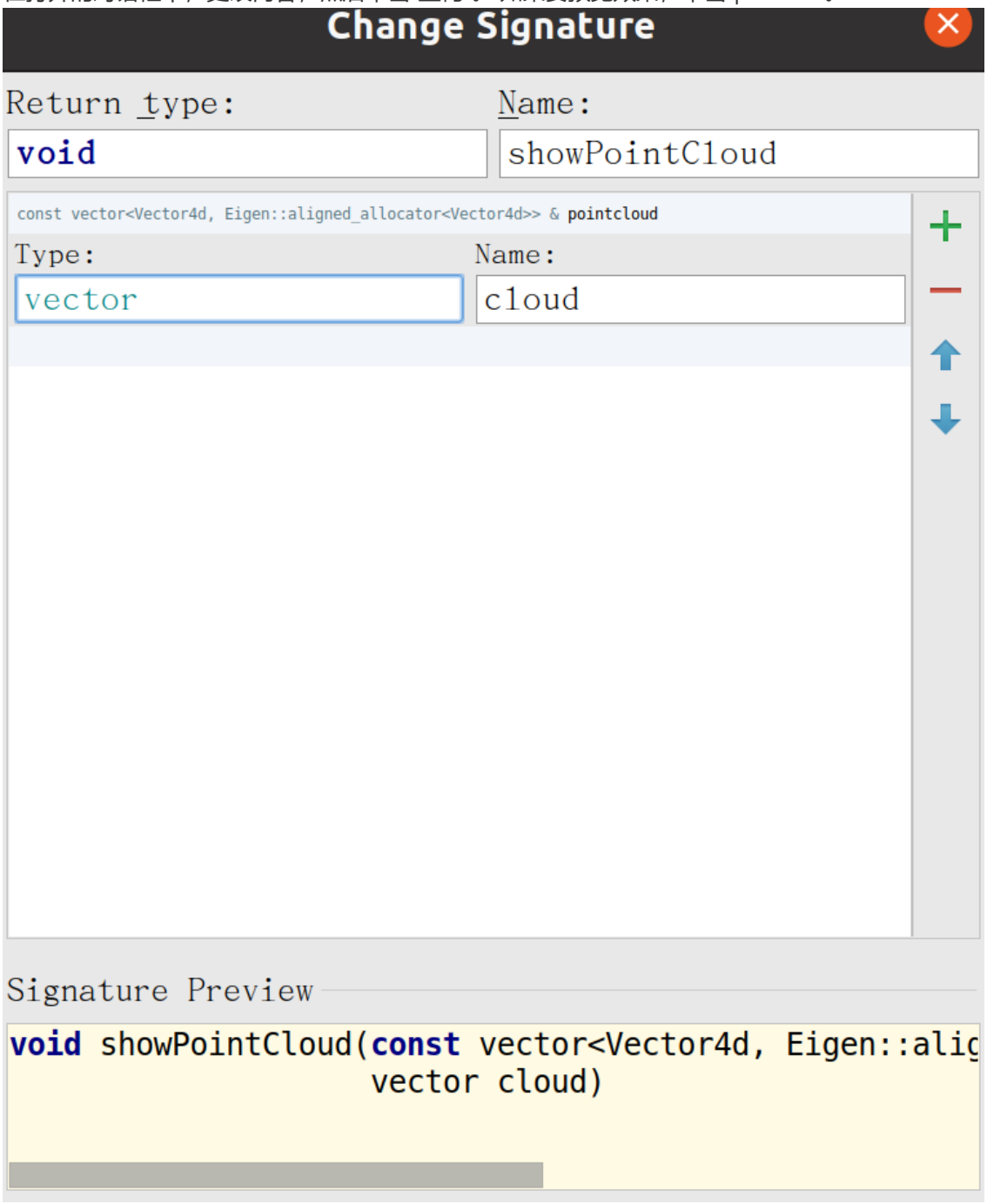


ctrl+F6 重构

refactor-> change signature
可以应用于方法签名或类签名
对于Class，重构可以修改其类型参数；对于method，此重构可以更改方法名称，添加，删除，重新排序和重命名参数和异常（exceptions），并通过调用层次结构传播新的参数和异常。

1. 选择要更改其签名的方法或类。
2. Refactor | Change Signature (Ctrl+F6)

3. 在打开的对话框中，更改内容，然后单击“重构”。 如果要预览效果，单击“preview”。



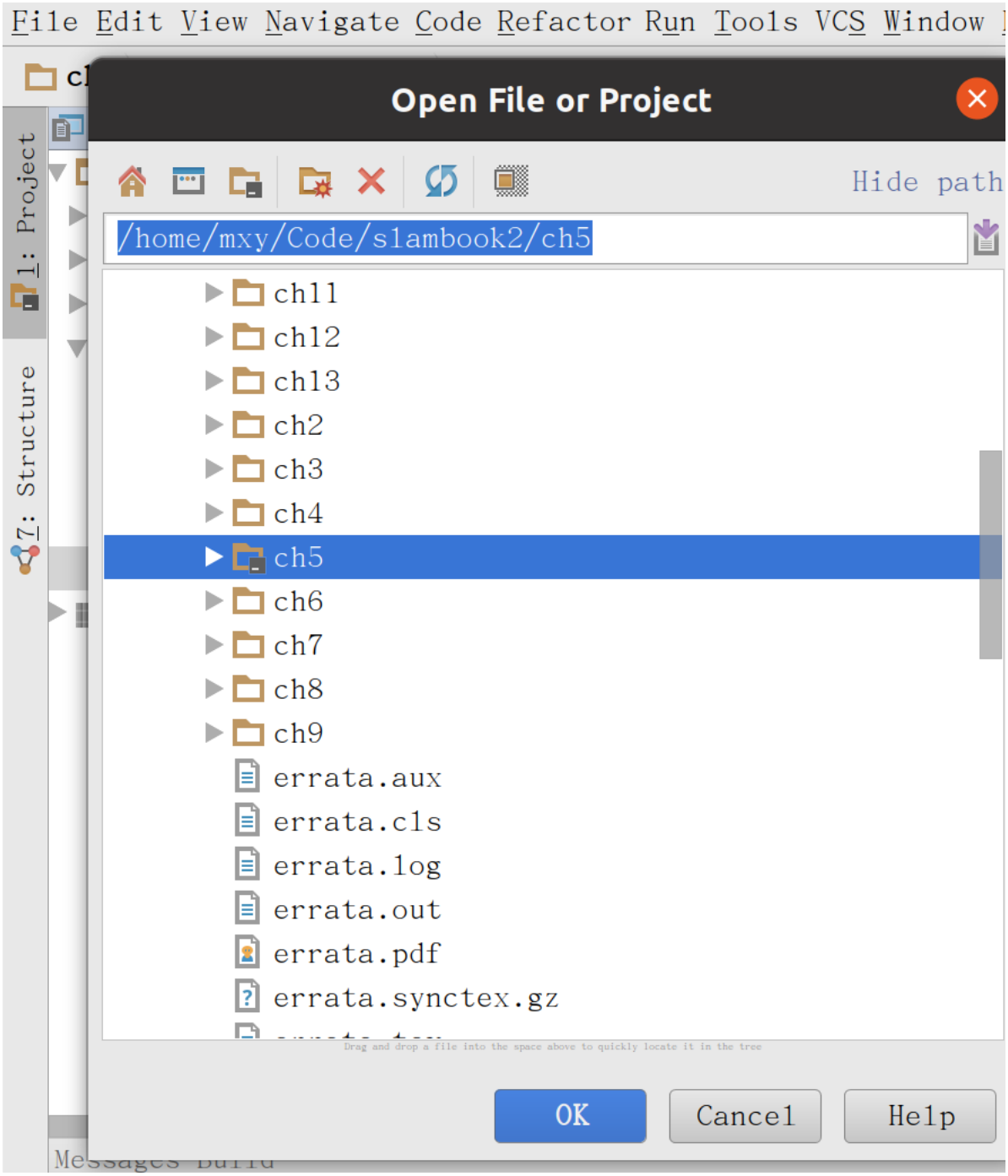
基础调试方法演示

clion默认的调试工具试GDB，也可以在Setting中修改为其他的调试工具。(clion把GDB的命令行调试给界面化了)
演示的代码是slambook里的chap5 (图像的基本去畸变和双目视觉)

打开项目

file->open

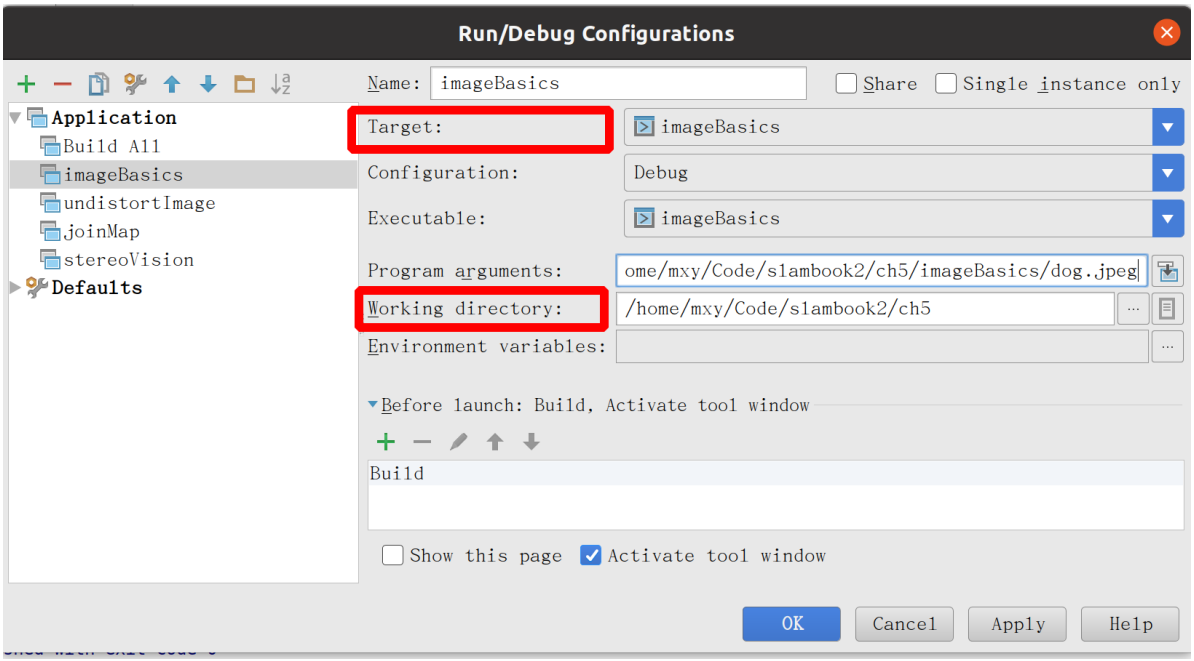
cmakelist会自动把工程根据设置进行链接，



配置参数

```
1 // 读取argv[1]指定的图像
2 cv::Mat image;
3 image = cv::imread(argv[1]); //cv::imread函数读取指定路径下的图像
```

run->edit configuration



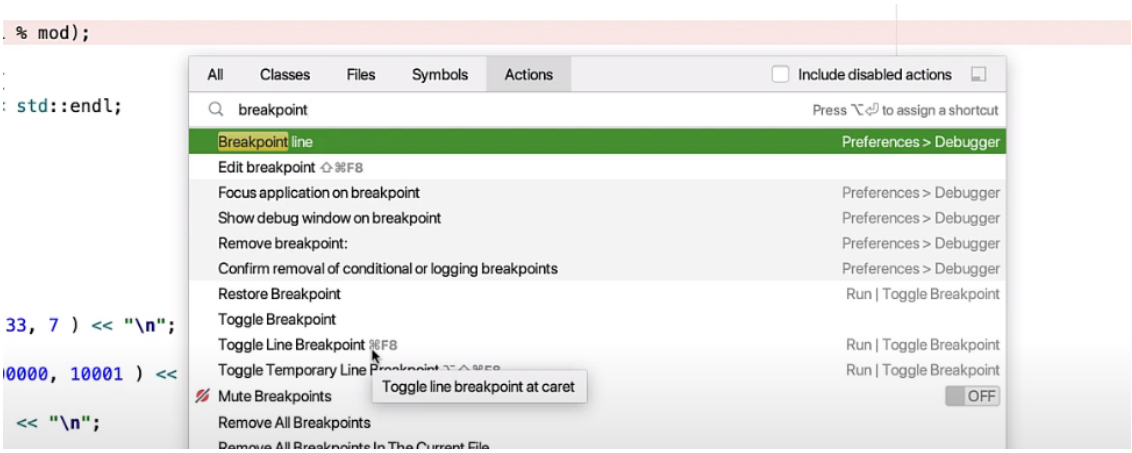
需要设置编译目标，因为工程里可同时存在多个可执行文件

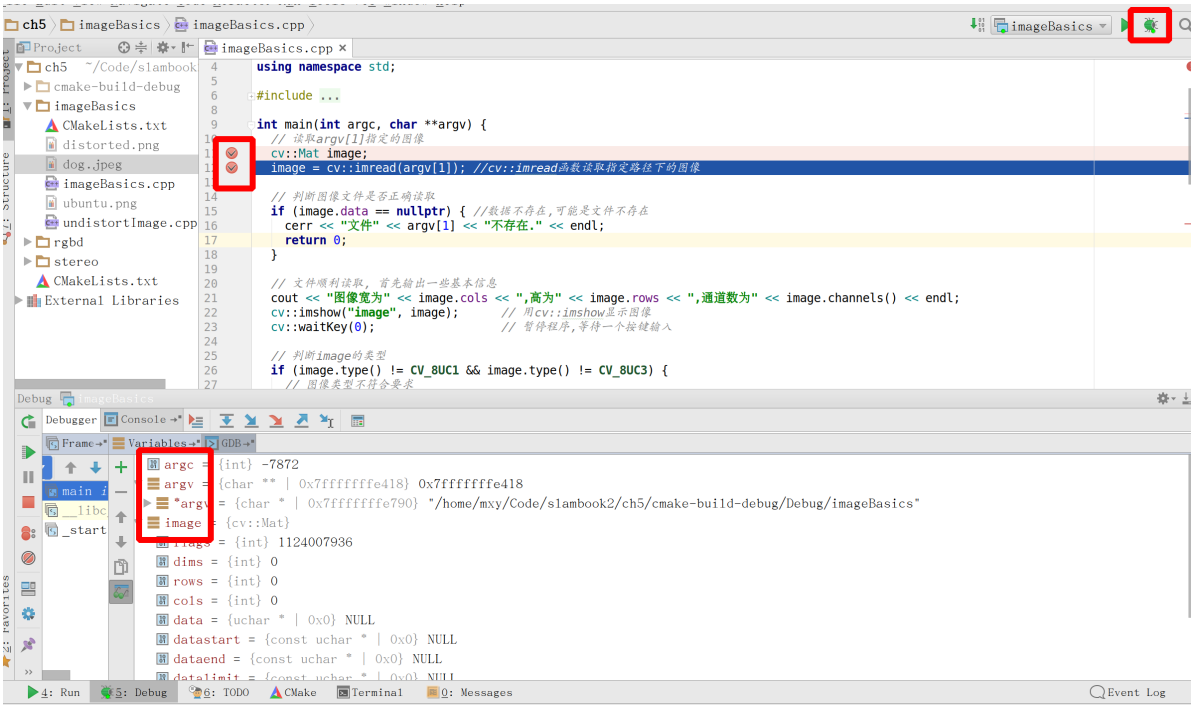
- [ ] 添加视频： clion运行图像基础操作.mkv #todo

断点调试

设置断点

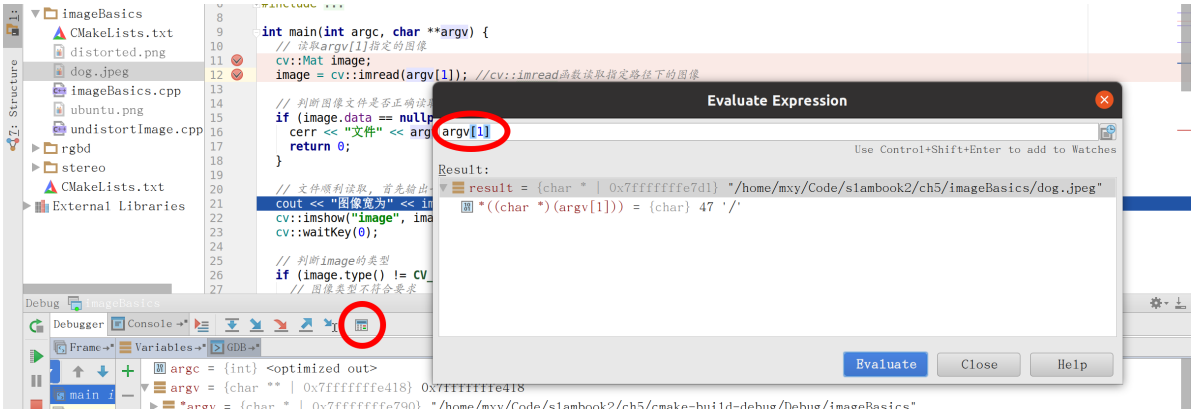
- 鼠标单击
- run->toggle breakpoint ->line breakpoint
- find





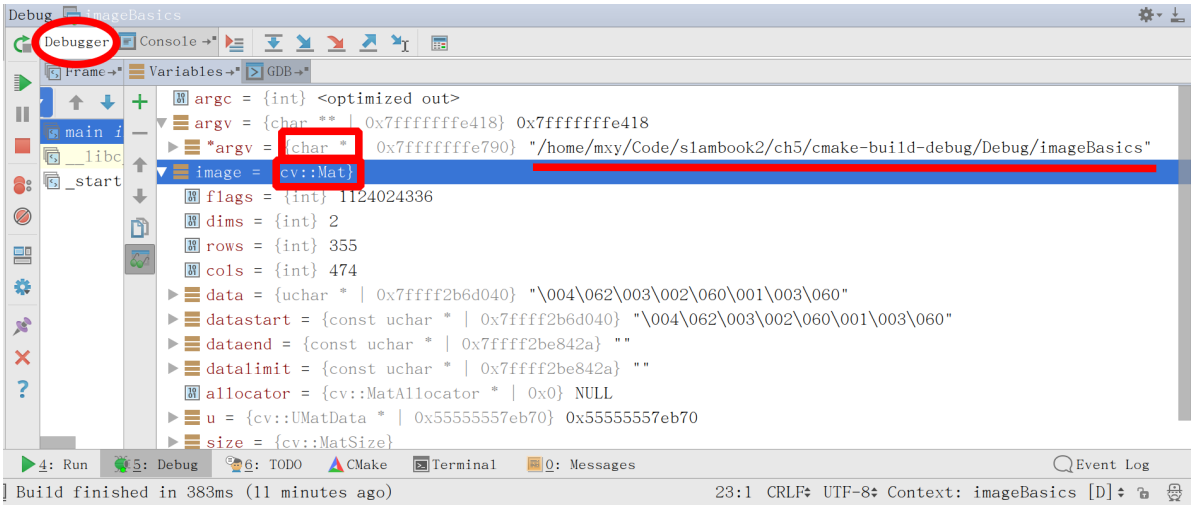
Evaluate Expression

输入你要查看的数据，并点击Evaluate



调试窗口Debugger

可以看到变量的类型和值

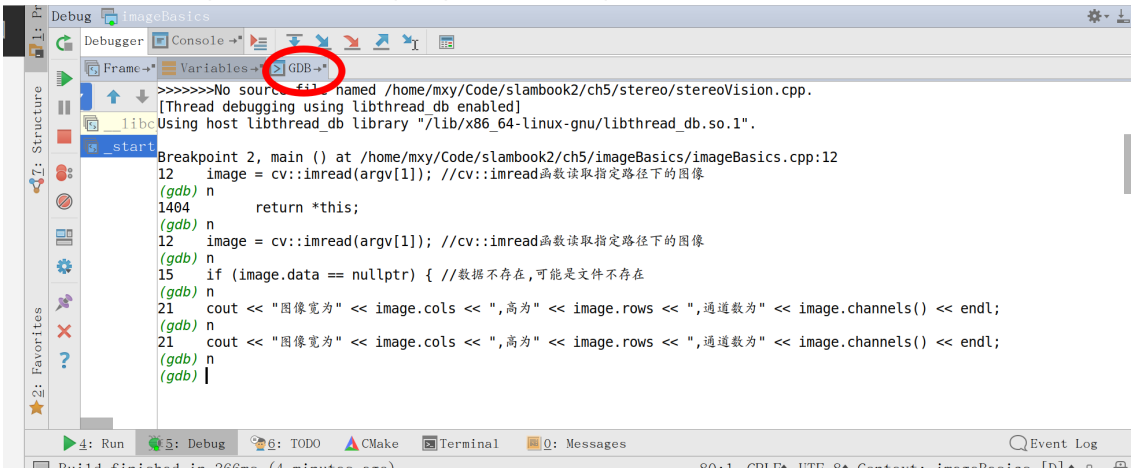


console窗口：输出cout的值

单步调试

1. step over

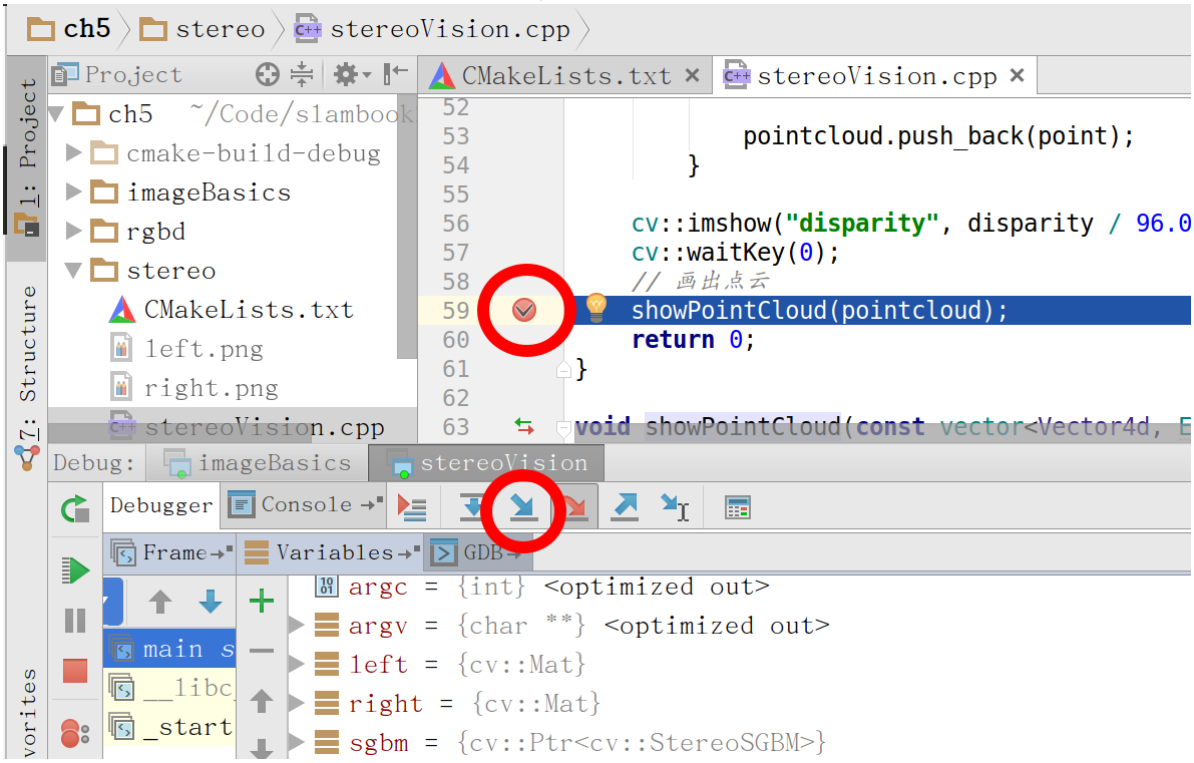
2. GDB窗口，输入n(不进入函数体内)或s(进入函数体内)即可 单步调试



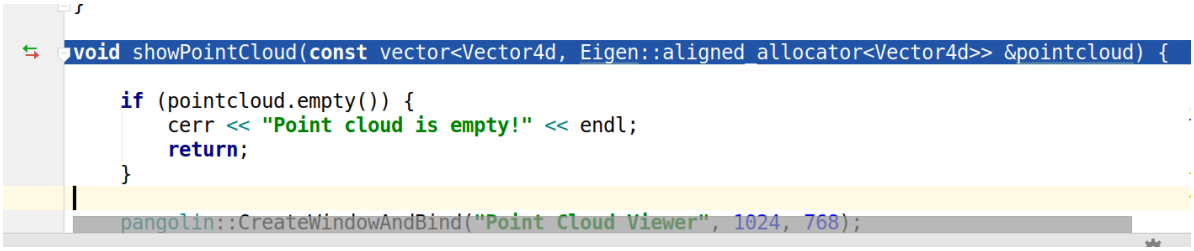
step into

逐语句执行，如果该行有自定义的方法，则进入该方法内部继续执行，需要注意如果是类库中的方法，则不会进入方法内部。

以双目视觉的为例，在56行设置断点，点击step into 会进入方法内部



如下图



force step into

强制单步跳入方法内部，和step into功能类似，主要区别在于：如果当前行有任何方法，则不管该方法是我们自定义还是**类库**提供的，都能跳入到方法内部继续执行

step out

从方法内部跳出，返主函数

run to cursor

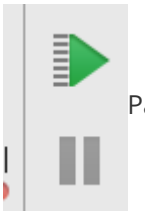
自动调试到光标所在行

断点管理区

重新调试

点击该按钮会停止目前的应用,并且重新启动.

暂停和恢复



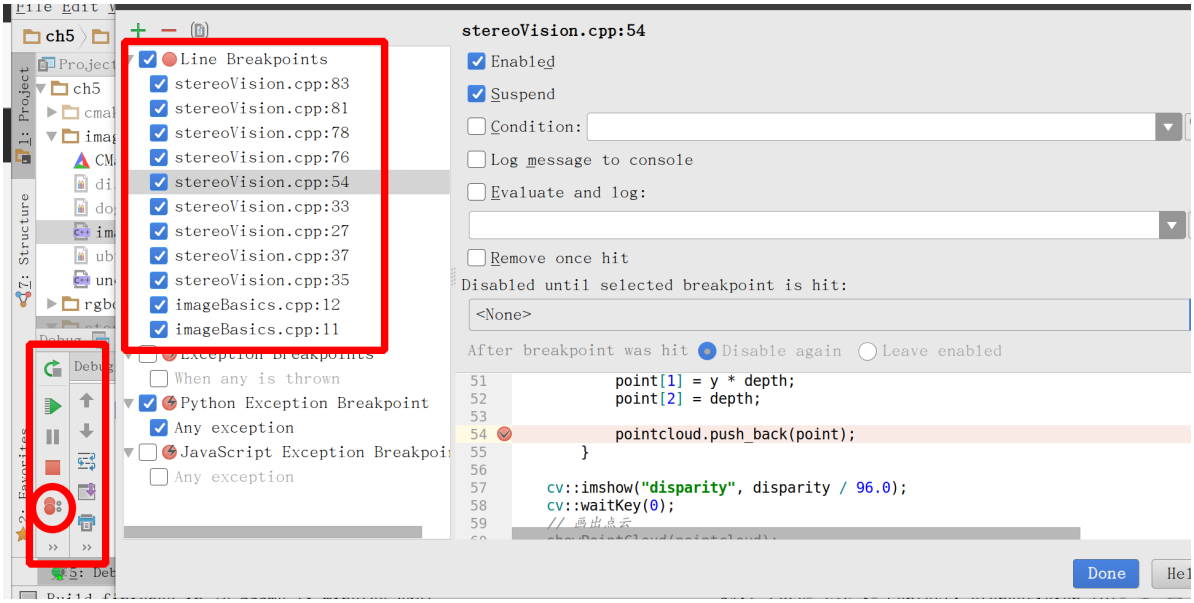
Pause Program(下）暂停应用的执行；

Resume Program（上）

1. 恢复暂停的程序
2. 设置多个断点的情况下，点击此按钮程序会从当前断点移动到下一个断点处，两个断点之间的代码自动被执行

显示所有断点

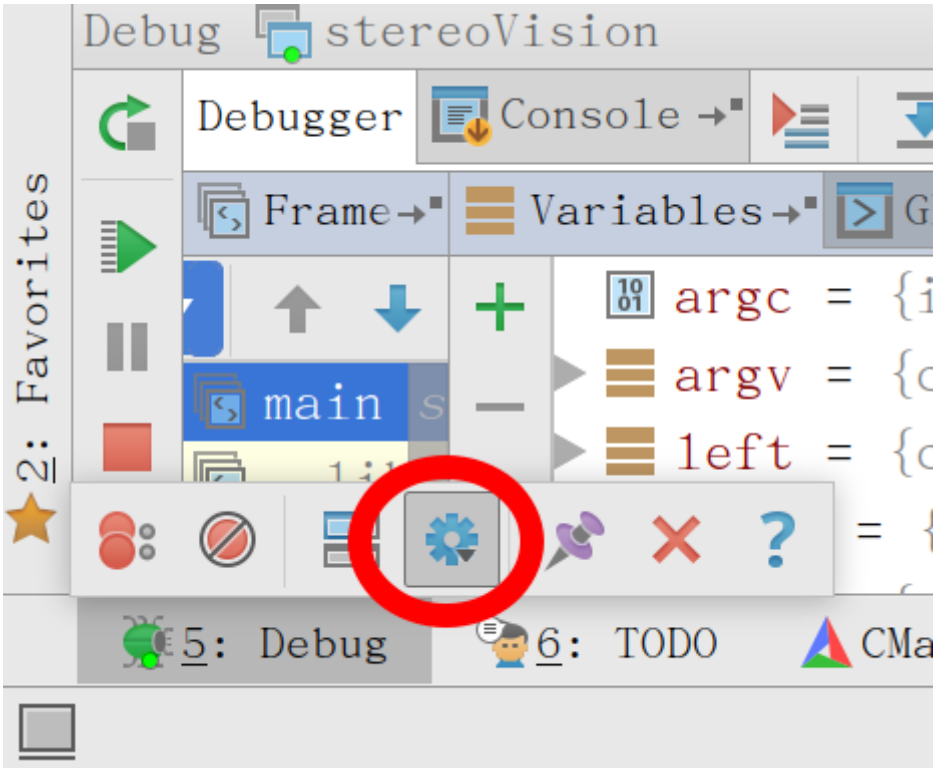
设置的断点比较多，一个页面显示不完全，可以点击左边两个圈圈的按钮，能直观看到所有的断点，可对断点进行删除、修改属性等操作



一键清除所有断点

不用傻乎乎的手动取消所有断点了

自动变量



inline |

```
    // 读取图像
    cv::Mat left = cv::imread(left_file, 0);   left: cv::Mat
    cv::Mat right = cv::imread(right_file, 0);   right: cv::Mat
    cv::Ptr<cv::StereoSGBM> sgbm = cv::StereoSGBM::create(   sgbm: cv::Ptr<cv::StereoSGBM>
        0, 96, 9, 8 * 9 * 9, 32 * 9 * 9, 1, 63, 10, 100, 32);     // 神奇的参数
    cv::Mat disparity_sgbm, disparity;   disparity_sgbm: cv::Mat   disparity: cv::Mat
    sgbm->compute(left, right, disparity_sgbm);
    disparity_sgbm.convertTo(disparity, CV_32F, 1.0 / 16.0f);

    cout << "读取图片完成." << endl;
```

|
|--|--|

ubuntu下录屏软件

Simple Screen Recorder

```
#添加源
sudo add-apt-repository ppa:maarten-baert/simplescreenrecorder
#更新源
sudo apt-get update
#安装
sudo apt-get install simplescreenrecorder
```

我分享中的小视频都是用这个软件录制的

Q&A

命令行 和IDE 输出路径不同

命令行默认的输出路径是 二进制文件所在的路径

如果把IDE的working directory设置为二进制所在路径，则输出路径是一致的

release模式 VS debug模式

cmake debug和release设置

方式1：显式指定

```
mkdir Release
cd Release
cmake -DCMAKE_BUILD_TYPE=Release ..
make
```

或者

```
mkdir Debug
cd Debug
cmake -DCMAKE_BUILD_TYPE=Debug ..
make
```

方法2：在Cmakelists.txt 中设置编译模式为debug or Release

```
SET(CMAKE_BUILD_TYPE "Debug”)
or
SET(CMAKE_BUILD_TYPE "Release")
```

debug和release模式的不同

debug模式会给定义的变量等额外的分配内存，在调试时能看到具体的变量值，速度慢

release模式会引入很多优化，断点不可用

一般情况下使用debug进行开发，release进行编译

【相关博客】【C++】Debug模式和Release模式的区别 - Y先森0.0 - 博客园 (cnblogs.com)

cmakelists其他语法

```
# CMAKE_CXX_COMPILER：指定C++编译器
# 如果是gcc编译器,则在编译选项中加入c++11支持
set(CMAKE_CXX_FLAGS "-std=c++11 ${CMAKE_CXX_FLAGS}")
```