**代码问题：**

**一、 单目稠密重建**

**1.** **十四讲第十二章单目稠密建图：**

**错误：安装sophus库时执行make指令后出现错误：**

**/home/liyan/Sophus/sophus/so2.cpp:32:26: error: lvalue required as left operand of assignment**

**unit\_complex\_.real() = 1.;**

**^~**

**/home/liyan/Sophus/sophus/so2.cpp:33:26: error: lvalue required as left operand of assignment**

**unit\_complex\_.imag() = 0.;**

**解决方法：**

在出现错误的地方修改为

unit\_complex\_.real(1.);

unit\_complex\_.imag(0.);

然后执行make命令，编译成功

**2.** **十四讲第十二章单目稠密建图：**

**错误：在编译执行单目稠密重建时遇到错误：**

**/home/liyan/dense\_mono/dense\_mapping.cpp:10:10: fatal error: sophus/se3.hpp:** **没有那个文件或目录**

**#include <sophus/se3.hpp>**

**原因：**安装的sophus库是非模板类型，而单目稠密重建的代码中要求sophus为模板类型.非模板类型相应文件后缀为.h文件，而模板性sophus的相应文件后缀为.hpp文件。故编译出现错误，需要卸载非模板性sophus，重新安装模板性sophus库。

**解决方法：**卸载非模板sophus：简单粗暴直接删除文件夹

安装模板类sophus库：

git clone https://github.com/strasdat/Sophus.git

cd Sophus/

mkdir build

cd build

cmake ..

make

sudo make install

很顺利编译成功

**3.** **十四讲第十二章单目稠密建图：**

**编译结果中显示长篇：**

**CMakeFiles/dense\_mapping.dir/dense\_mapping.cpp.o：在函数‘unsigned long long fmt::v9::detail::width\_checker<fmt::v9::detail::error\_handler>::operator()<float, 0>(float) [clone .isra.272]’中：**

**dense\_mapping.cpp:(.text.unlikely+0xc)：对‘fmt::v9::detail::throw\_format\_error(char const\*)’未定义的引用**

**CMakeFiles/dense\_mapping.dir/dense\_mapping.cpp.o：在函数‘unsigned long long fmt::v9::detail::precision\_checker<fmt::v9::detail::error\_handler>::operator()<float, 0>(float) [clone .isra.283]’中：**

**dense\_mapping.cpp:(.text.unlikely+0x1c)：对‘fmt::v9::detail::throw\_format\_error(char const\*)’未定义的引用**

**CMakeFiles/dense\_mapping.dir/dense\_mapping.cpp.o：在函数‘Sophus::SO3Base<Sophus::SO3<double, 0> >::normalize() [clone .part.626]’中：**

**dense\_mapping.cpp:(.text.unlikely+0x99)：对‘fmt::v9::vprint(fmt::v9::basic\_string\_view<char>, fmt::v9::basic\_format\_args<fmt::v9::basic\_format\_context<fmt::v9::appender, char> >)’未定义的引用**

**在结尾处显示：**

**collect2: error: ld returned 1 exit status**

**CMakeFiles/dense\_mapping.dir/build.make:111: recipe for target 'dense\_mapping' failed**

**make[2]: \*\*\* [dense\_mapping] Error 1**

**CMakeFiles/Makefile2:67: recipe for target 'CMakeFiles/dense\_mapping.dir/all' failed**

**make[1]: \*\*\* [CMakeFiles/dense\_mapping.dir/all] Error 2**

**Makefile:83: recipe for target 'all' failed**

**make: \*\*\* [all] Error 2**

**原因：**error1显示生成dense\_mapping可执行文件失败，采用gcc命令生成可执行文件时遇到同样的问题。未定义一般是库或者链接的问题。据说这个问题能解决的前提是安装fmt库后重新编译然后重新安装sophus库并重新编译，重点是重新编译。

**解决方法：**应该在CmakeList.txt文件中声明fmt库并写入链接。

include\_directories(${FMT\_INCLUDE\_DIRS})

find\_package(FMT REQUIRED)

add\_executable(dense\_mapping dense\_mapping.cpp)

target\_link\_libraries(dense\_mapping ${THIRD\_PARTY\_LIBS} fmt::fmt)

将程序用到的库写在链接里。

**4.** **十四讲第十二章单目稠密建图：**

**错误：bash: ./dense\_mapping:** **没有那个文件或目录**

**原因：**在CMakeLists.txt文件中

add\_executable(dense\_mapping dense\_mapping.cpp)

如果编译成功的话，能生成可执行文件dense\_mapping，出现了3中的错误，并没有生成可执行文件，所以运行的时候会出现没有该项文件的错误

**解决方法：解决3中的问题即可生成可执行文件。编译过程中，绿色字代表正在链接，白色字代表生成相对应的可执行文件。**

**二、 视觉里程计（光流法）**

**5. 十四讲第八章光流法：**

**编译时出现**

**/home/liyan/文档/slambook2-master/ch8/direct\_method.cpp:67:10: error: ‘mutex’ in namespace ‘std’ does not name a type**

**std::mutex hessian\_mutex;**

**^~~~~**

**解决方法：**

添加头文件：

#include <thread>

#include <mutex>

#include <unistd.h>

**6. 十四讲第八章光流法：**

**编译时出现**

**error: ‘chrono’ has not been declared**

**解决方法：**

在.cpp文件中增加

#include <algorithm>

#include <chrono>

**7. 十四讲第八章光流法：**

**编译时出现**

**/home/liyan/文档/slambook2-master/ch8/direct\_method.cpp:290:9: error: ‘unique\_lock’ was not declared in this scope**

**unique\_lock<mutex> lck(hessian\_mutex);**

**解决方法：**把fmt库的问题解决了这个问题就消失了。我也不知道怎么回事。

**PS：**

1. **学习资源**

视觉SLAM十四讲

课程

代码

1. **论文写作技巧**
2. **题目选择**

毕业论文如果要改题目，最迟中期检查前

适用场景精确一些，对技术的选择会更有方向

题目中体现创新点

1. **创新点**
2. **内容**

第一位是工作量，第二位是创新点

论文审核更加严格

题目中的内容在正文中都要有体现

研究内容与题目要切合

一个方面做出深度，不要做两个方面但都不精

1. **视觉SLAM2的研究方向**

稠密建图

语义地图构建

**语义地图构建学习记录**

1. **问题记录：**
   * + 1. **什么是端到端**

端到端可能指输入图片或视频，输出机器人的动作、导航、避障等。例如未知环境中进行导航。

* + - 1. **深度学习主要运用在什么环节上**

深度学习替换或辅助视觉SLAM中的一个或几个环节。

* + - 1. **深度学习和机器学习的区别**
      2. **什么是语义信息**

维基百科：在传媒行业指语言文字提供的信息，在计算机行业指有意义的数据提供的信息，在科学哲学领域泛指一种有意义的语言、文字、图片等等提供的信息。

图像中的语义分为视觉层、对象层和概念层：

* 视觉层，即颜色纹理形状等，被称为底层特征语义。例如一张图片中的区域。
* 对象层，通常包含了属性特征。例如一张图片中的沙子、人、海水、蓝天。
* 概念层，指人类能理解的信息。例如这一幅图描绘的是海滩。

在视觉SLAM上加上语义信息可以衍生出两个方向：图像语义分割和语义地图构建。

* + - 1. **上下文信息**

也可以叫做上下文特征，在图像中一个像素点肯定不是孤立的，一个像素点跟它周围的像素点都有联系，上下文信息指的是像素与其周围像素间的联系。

* + - 1. **语义信息具体怎么用**

在视觉SLAM上加上语义信息可以衍生出两个方向：图像语义分割和语义地图构建。

* + - 1. ORB-SLAM2能否用于语义地图构建
      2. 适用的开源代码和数据集有哪些
      3. **对象级语义信息**

神经网络的引入，借助语义信息，将数据关联从传统的像素级提高到了对象级。

* + - 1. 像素级语义分割
      2. 什么是地图的全局一致性
      3. **什么是时间一致性**

时间一致性是各种语义建图方法中的一个共同属性

* + - 1. **什么是拓扑地图**

**拓扑地图**更强调地图元素间的关系。拓扑地图是一个图（Graph），由节点和边组成，只考虑节点间的连通性。放松了地图对精确位置的需求，去掉了地图的细节问题，是一种更紧凑的表达方式。

缺点：不擅长表达具有复杂结构的地图。

* + - 1. **什么是度量地图**

度量地图强调精确地表示地图中物体的位置关系，通常用**稀疏（sparse）**和**稠密（dense）**进行分类

* **稀疏地图：**只有代表性的特征点，比如角点信息，并不代表所有物体；
* **稠密地图：**通常按某种分辨率，由许多小块组成。二维度量地图由许多小格子（Grid），三维由许多小方块（Voxel）。

**Ps：**比如说一张桌子，稀疏点云地图呈现的是四个桌角，稠密点云地图呈现的是一个桌面。大规模的度量地图有时候会出现一致性问题。

* + - 1. **什么是实例分割**

同时具备语义分割和目标检测的特点，即既需要进行像素层面上的分割，又需要判断种类。

* + - 1. **动态特征的匹配**
      2. **除了深度学习还有什么方法构建语义地图**
      3. **基于什么方向建立语义地图**

比如说自动驾驶，考虑有没有研究条件

* + - 1. **视觉SLAM的评价指标**

相对位姿误差

绝对轨迹误差

* + - 1. **经典VSLAM与语义VSLAM在实时性、精度、鲁棒性上的差距**

1. **传统VSLAM：**
2. **组成：**

视觉传感器、前端（视觉里程计）、后端（非线性优化）、回环检测、地图构建

1. **性能**

传统的视觉SLAM在建图的实时性上已经取得不错的结果，在建图精度和系统鲁棒性上有提高的空间。尤其是系统的鲁棒性。

1. **不足与改进方向**

视觉SLAM分为特征法和直接法。特征点法依赖于特征点的提取和匹配，过程较为复杂。直接法依赖于像素强度值。因此在光照变化和纹理稀疏的环境建图结果不尽人意。

1. **相关方案：**

LSD-SLAM，

RGBD-SLAMV2，

ORB-SLAM

Code-SLAM/DVSO：基于立体相机

语义信息多与视觉里程计和回环检测结合，用于提高建图精确度和系统鲁棒性。

1. **地图的构建**
2. **度量地图：**

度量地图强调精确地表示地图中物体的位置关系，通常用**稀疏（sparse）**和**稠密（dense）**进行分类

稀疏地图：只有代表性的特征点，比如角点信息，并不代表所有物体；

稠密地图：通常按某种分辨率，由许多小块组成。二维度量地图由许多小格子（Grid），三维由许多小方块（Voxel）。

**Ps：**比如说一张桌子，稀疏点云地图呈现的是四个桌角，稠密点云地图呈现的是一个桌面。大规模的度量地图有时候会出现一致性问题。

1. **拓扑地图：**

拓扑地图更强调地图元素间的关系。拓扑地图是一个图（Graph），由节点和边组成，只考虑节点间的连通性。放松了地图对精确位置的需求，去掉了地图的细节问题，是一种更紧凑的表达方式。

缺点：不擅长表达具有复杂结构的地图。

1. **语义SLAM：**

实际上是采用深度学习的方法代替传统视觉SLAM中的某些环节，来提取大量语义信息，通过提取到的语义信息来完成任务，例如语义分割和语义地图构建。

1. 语义信息提取的方法
2. 目标检测算法
3. 语义分割算法
4. 语义地图构建
5. 语义地图的构建分为面向场景的语义地图构建和面向对象的语义地图构建。

面向场景的语义地图：

面向对象的语义地图：

待解决问题：实时性和动态场景

1. 深度估计

有监督学习

无监督学习

趋势：弱监督学习和无监督学习

**视觉SLAM十四讲（第2版）**

* **第一章 资源获取**
* **第二章 初识SLAM**

**2.1 简介**

**传统VSLAM：**

1. **主要任务**

**定位：**知道自己处于什么位置和姿态

**建图：**知道周围环境

1. **相机：**

**单目相机：**以二维形式记录三维世界。只能通过相机的运动估计环境，因此具有尺度不确定性。

**双目相机：**通过两个摄像头拍摄图像的差异计算距离，需要大量计算，且可靠性低。

**深度相机：**通过自身红外功能得到距离。

**2.2 经典视觉SLAM框架**

1. **组成：**

**视觉传感器：**读取相机信息和预处理；

**前端（视觉里程计）：**估计相邻图像之间相机的运动，以及局部地图；

**后端（非线性优化）：**接受视觉里程计的信息和回环检测的信息，对他们进行优化。以构建全局一致的地图；

**回环检测：**判断机器人是否到达过先前的位置，将信息传给后端，校准累积误差；

**地图构建：**根据估计的轨迹，构建地图。

**Ps：**

**相关方案：**LSD-SLAM，RGBD-SLAMV2，ORB-SLAM

**缺陷：**弱纹理环境、光照变化、动态目标

**2.2.1 视觉里程计**

第七章、第八章

**2.2.2 后端优化**

第九章、第十章

**2.2.3 回环检测**

第十一章

**2.2.4 建图**

第十二章

**2.3 视觉SLAM数学问题**

第三章、第四章、第五章、第六章

**2.4 编程基础**

**Cmake：**【基于VSCode和CMake实现C/C++开发 | Linux篇】 https://www.bilibili.com/video/BV1fy4y1b7TC/?share\_source=copy\_web&vd\_source=780b556eb327416c32428160ab3c7206

**ROS：**【【Autolabor初级教程】ROS机器人入门】 https://www.bilibili.com/video/BV1Ci4y1L7ZZ/?share\_source=copy\_web&vd\_source=780b556eb327416c32428160ab3c7206

**语言：**

* **第七章 视觉里程计1（特征点法）**

视觉里程计的算法：**特征点法**和**直接法**

**特征点法**具有稳定，对光债、动态物体不敏感的优势，是视觉里程计的主流方法。

**7.1 特征点法**

**7.1.1 特征点**

1. **典型特征点：SIFT、SURF、ORB**

**SIFT：**尺度不变特征变换。可适用于光照、尺度、旋转等场景，但计算量巨大，无法做到实时性。

**SURF：**

**ORB：**改进FAST关键点和BRIEF描述子。具有旋转、尺度不变性的同时具备实时性。

1. **特征点的组成：关键点**和**描述子**

**关键点：**该特征点在图像的位置，有些还包括朝向、大小等信息。

**描述子：**是一个向量，描述特征点周围像素信息。

**Ps：**外观相似的特征点应该具有相似的描述子，或者说，只要两个特征点的描述子在向量空间上距离相近，就可以认为他们是同样的特征点。

**7.1.2 ORB特征**

**ORB：**改进FAST关键点和BRIEF描述子。

**ORB特征提取步骤：**

FAST角点提取：找出图像中的角点。ORB中计算了特征点的主方向，为BRIEF描述子增加了旋转不变性。

BRIEF描述子：对前一步提取的特征点的周围图像区域进行描述。ORB中的BRIEF使用了上一步的方向信息。

1. **FAST关键点**

FAST关键点本不具有方向信息和尺度不变性。

**尺度不变性**通过构建**图像金字塔**解决；

**方向性**通过**灰度质心法**实现。

改进后的FAST称为Oriented FAST，应用在ORB中。

1. **BRIEF描述子**

BRIEF是一种二进制描述子，计算快速，储存方便。

ORB中的描述子是Steer BRIEF描述子，利用关键点提取到的方向信息，具有良好的旋转不变性。

**7.1.3 特征匹配**

特征匹配解决了SLAM中的数据关联问题。

方法：快速近似最近邻算法（FLANN）

**Ps：**误匹配广泛存在，是SLAM中的一个瓶颈问题

**7.2 实践：特征提取和匹配**

**7.3 2D-2D：对极几何**

使用单目相机时，只能得到2D像素坐标，根据两组2D点**估计相机运动**的问题用对极几何来解决。

**7.3.1 对极约束**

**7.3.2 本质矩阵**

**7.3.3 单应矩阵**

**7.4 实践：对极约束求解相机运动**

**尺度不确定性：**

**初始化的旋转问题：**

**对于多余8点的情况：**

**7.5 三角测量**

通过三角测量的方法估计**地图点深度**。

**7.6 实践：三角测量**

**7.7 3D-2D：PnP**

PnP是求解3D到2D点运动的方法。对极几何存在着初始化、纯旋转和尺度的问题。特征点的3D位置可以由三角化或RGB-D相机的深度图确定。在双目相机或RGB-D的视觉里程计中可以用PnP估计相机运动。在单目视觉里程计中必须先进行初始化，才能使用PnP。3D-2D方法不需要使用对极约束。

PnP的解决方法有P3P、直接线性变换DLT、EPnP、UPnP、光束法平差BA。

**7.7.1 直接线性变换**

**7.7.2 P3P**

**7.7.3 最小化重投影误差求解PnP**

* **第八章 视觉里程计2（直接法）**

**8.1 直接法的引出**

特征点法缺点：

关键点的提取和描述子的计算非常耗时；

特征点丢弃了图像中大部分的可用信息；

在弱纹理场景不能提取到足够的特征点用于匹配；

只能构建稀疏地图。

直接法思路：

保留特征点，只计算关键点，不计算描述子，使用光流法跟踪特征点运动，替换了描述子的工作，但估计相机运动时仍然使用对极几何等；

只计算关键点，不计算描述子，使用直接法计算特征点在下一刻图像中的位置，在直接法中通过图像的像素灰度信息同时估计相机运动和点的投影，不需要提取到的点是角点 。

直接法：

根据像素亮度信息估计相机运动，可以完全不用计算关键点和描述子。直接法可以分为稀疏、半稠密和稠密三种。

* **第九章 后端优化1**

**9.1 概述**

SLAM中后端求解的方法大致可以分为两类，一类是基于**滤波器**的方法，另一类是**非线性**的优化方法。如果考虑马尔可夫性，即k时刻的状态只与k-1时刻相关，则得到以卡尔曼滤波器为代表的的基于滤波器的方法。另一种方法则考虑k时刻与之前所有状态间的关系，即非线性优化框架。

目前，视觉SLAM主流后端求解方法为基于非线性优化的方法。

**9.2 BA与图优化**

**9.3 实践：Ceres BA**

**9.4 实践：g2o求解BA**

**9.5 小结**

* **第十二章 建图**

**12.1 概述**

1. **地图的用处：**
2. **定位：**由稀疏地图就能实现。
3. **导航：**机器人能够在地图中进行路径规划，知道哪些位置可以通过，哪些位置不能通过，这项任务所需要的地图形式至少是一种稠密地图，稀疏地图无法完成此项任务。
4. **避障：**仅有特征点无法判断某个特征点是否为障碍物，因此此项任务同样需要稠密地图。避障与导航相比，更注重局部的、动态的障碍物的处理。
5. **重建：**此项任务也需要稠密地图，但不局限于稠密地图，如果想要重建物体表面的纹理，仅用稠密地图也是不够的。
6. **交互：**交互指人与地图间的交互。在地图中放置一些虚拟物体，并与虚拟物体产生互动。需要机器人对地图有更高的认知，也就是语义地图。

**Ps：**例如一张桌子，稀疏地图可能只建模出四个角，稠密地图可能能够建模出一个桌面。

**12.2 单目稠密重建**

**12.2.1 立体视觉**

1. **三种相机**
2. **单目相机：**估计相机运动，利用三角化计算像素的距离。
3. **双目相机：**利用左右目视觉差计算像素距离。
4. **RGB-D相机：**直接获取像素距离。

使用单目相机和双目相机的方式称为立体视觉，计算量巨大，可靠性相对较低。对于稠密地图的重建，常用的是RGB-D相机，但是也会被量程、应用范围和光照所影响。但在大场景、室外环境中，立体视觉要比RGB-D效果更好。

在稠密深度图估计中，无法把每个像素都当作特征点计算描述子，因此**匹配（**如何确定第一幅图的某像素出现在其他图中的位置**）**是非常重要的一环。需要用到**极线搜索**和**块匹配**技术。

当清楚某个像素点在图中的位置时，就可以用三角测量法确定深度，构建稠密地图需要使用很多次三角测量法进行深度估计的收敛。当深度估计从一个不确定的值收敛到一个稳定值，就是**深度滤波技术**。

**12.2.2 极线搜索和块匹配**

1. **极线搜索：**

极线：十四讲P309

在特征点法中通过特征匹配找到不同图像中匹配的特征点。而此时没有描述子，只能在极线上搜索与特征点相似的点进行匹配。听起来与直接法类似，在直接法中，比较单个像素的亮度值不一定可靠，同样在极线中也可能出现多个相似的特征点。

像素点的亮度值区分不一定可靠，即可以考虑用像素块进行区分，在相似的特征点周围取合适大小的像素块，进行**块匹配**。

1. **计算块与块之间的差异：**
2. **SAD：**差的绝对值之和；0相似，1不相似；
3. **SSD：**平方和；0相似，1不相似；
4. **NCC：**相关性；0不相似，1相似。
5. **去均值SSD、NCC：**减少计算量，或许会出现整体更亮但仍然相似的情况，相对更可靠。
6. **深度滤波器**

搜索距离较长时，得到的非凸函数有许多个峰值，但真实匹配的点只有一个，这种情况下倾向于使用概率分布描述深度值。在不断对不同图像进行极线搜索时，估计的深度分布将发生怎样的变化—深度滤波器。

**12.2.3 高斯分布的深度滤波器**

对像素点深度估计有滤波器和非线性优化两种思路，非线性优化效果较好但计算量大、耗时长，因此实时性差；建图部分通常采用计算量较少的滤波器。

**12.2.4估计稠密深度的完整过程**

1. 假设所有像素点的深度满足某个初始的高斯分布；
2. 当新数据产生时，通过极线搜索和块匹配确定投影点位置；
3. 根据几何关系计算三角化后的深度及不确定性；
4. 将当前观测融合进上一次的估计中。若收敛则停止计算，否则返回第2步。

**12.3 实践：单目稠密重建**

数据集：remode\_test\_data

遇到的问题

总结：

1. 块匹配的正确与否依赖于图像块是否具有区分度。如果图像是一块黑或一块白，那么无法通过梯度提取到合适的特征点，例如图像中的打印机表面，就出现了不该出现的条纹，打印机表面的深度信息就是不正确的。
2. 立体视觉的重建质量十分依赖于纹理环境。如果只关心某个像素周围的小块的话，无法在现有的算法上加 以改进并解决。
3. 在像素梯度垂直于极线的场景中，匹配程度都是一样的，得不到有效匹配；在像素梯度平行与极线的场景中，梯度变化明显，可以精准匹配。在实际应用中，多数情况介于这两种情况之间，极线匹配的不确定性随着像素梯度方向与极线方向变化，这就是像素的误差。



**12.4**

**RGB-D稠密建图**

1. RGB-D相机主流建图方式
2. 点云地图：
3. 三角网格、面片：
4. 体素：

**12.4.1 实践：点云地图**

**SLAM2课程**

**第一章 简介**

**第二章 视觉里程计**

**01讲 SLAM系统构造函数**

void 定义函数：

ComputePyramid：构建图像金字塔

**02讲 图像金字塔每层如何分配特征点数**

**03讲 ORB特征点简介、如何构造灰度质心圆**

**04讲 ORB构造函数仿函数、图像帧构造函数**

**05讲 构建图像金字塔并进行扩充**

**06讲 提取ORB特征点**

**07讲 四叉树实现特征点均匀分布**

**08讲 利用灰度质心计算ORB特征点方向、实现旋转**

**09讲 ORB描述子Steer BRIEF计算原理及代码实现**

**10讲 去畸变、计算图像边界、特征点网格划分**

* **计算ORB特征点核心代码：**

step1 ：检查图像有效性，查看图像是否为空

step2 ：构建图像金字塔

step3 ：特征点提取和匹配，计算特征点并均匀化

step4 ：计算描述子

step5 ：对图片进行高斯模糊

step6 ：计算高斯模糊后的描述子

**第三章 单目初始化**

**第01讲 为什么需要地图初始化**

单目初始化需要一定的时间，双目和RGB-D相机在一开始就可以完成初始化。

单目中的点的坐标是相对坐标，无法获取两个特征点间的坐标差；而双目和RGB-D中特征点的坐标是绝对坐标，可以获取两个特征点的距离。

**第02讲 单目地图初始化流程处理**

跟踪：Tracking.cc

初始化函数：Initializer.cc

单目初始化函数：MonocularInitialization() // 视频3.3

特征匹配器：ORBmatcher

步骤：

**step1：**判断是否已经初始化，如果未开始初始化，判断当特征点数量大于100时，开始初始化。

**step2：**初始化过后的第二帧，之间进入else，判断第二帧特征点是否达到100，只有连续两帧图像特征点达到100，才能开始初始化进程；

**step3：**构造特征匹配器：ORBmatcher，寻找匹配的特征点；

// 最佳和次佳特征点评分阈值，跟踪时一般为0.7，这里比较宽松，比值越大，说明最佳和次佳差别越小，当前特征点区分度越差，这里因为是初始化，所以设置为0.9；

ORBmatcher matcher(0.9,true);

**step4：**验证匹配结果，如果初始化的两帧特征点太少，则重新初始化；

**step5：**通过H模型或者F模型进行单目初始化，得到两帧之间的相对运动、初始MapPoints；

**step6：**初始化成功后，删除那些无法进行三角化的匹配点；

**step7：**将初始化的第一帧作为世界坐标系，因此第一帧变换矩阵为单位矩阵；

**step8：**创建初始化地图点MapPoints；

**第03讲 单目初始化中RANSAC随机选择点集**

**第04讲 单目初始化中数据归一化**

* 为什么要归一化

为了将所有特征点都放在统一坐标系下；

矩阵A是利用8点法求得，利用8点法求基础矩阵不稳定的一个主要原因是原始图像像素点坐标组成的系数矩阵A不好造成的，而A不好是因为像素点的齐次坐标各个分量的数量级相差太大，因此提出改进8点法，在应用8点法之前先进行归一化处理，使得原始图像坐标作同向性变换。

* 归一化步骤

首先，对每幅图的坐标进行平移，使图像中匹配的点的形心平移到原点；

接着对坐标系进行缩放使得各个分量总体上有一样的平均值，各个坐标轴的缩放相同

**第05讲 单目初始化中直接线性变换求解单应矩阵**

**第06讲 单目初始化中计算并根据 得分寻找最佳单应矩阵**

**第07讲 计算并根据得分寻找最佳基础矩阵**

**第08讲 卡方检验原理及其在ORB-SLAM2中的应用**

**第09讲 单目初始化中从单应矩阵恢复位姿**

**第10讲 单目初始化中从基础矩阵得到最佳位姿及三维点**

**第11讲 单目初始化成功构建初始化地图及单目尺度归一化**