北京理工大學珠海學院
ZHUHAI CAMPUS,BEIJING INSTITUTE OF TECHNOLOGY

毕业设计（论文）

并行跟踪与地图生成技术在 android 上实现

学院：信息学院

专业：信息工程

姓名：魏家立　　学号：120103021036

指导老师：唐佳林　　职称：讲师

中国 • 珠海

二〇一六年六月

北京理工大学珠海学院毕业设计

诚信承诺书

本人郑重承诺：我所呈交的毕业设计《并行跟踪与地图生成技术在 android 上实现》是在指导教师的指导下，独立开展研究取得的成果，文中引用他人的观点和材料，均在文后按顺序列出其参考文献，设计使用的数据真实可靠。

承诺人签名：_____

日期:_____年_____月_____日

并行跟踪与地图生成技术在 android 上实现

摘　要

本文展示了一种在未知环境中估计摄像头姿态的方法。这种尝试是由 SLAM 算法改编而来的，而 SLAM 就是为机器人探索开发的，所以我们提出的这个系统也是专为在小空间范围跟踪一个手持摄像头设计的。我们分开跟踪和地图建立在两个独立任务，在多核电脑上并行运行。tracking 线程处理不稳定的手持姿态，mapping 线程处理出现在帧上的 3D 特征点。计算上，这种算法允许一部分性能花销的优化技术与实时操作无关：这个系统的输出是产生有数千个标记点的详细地图。这些标记可以以帧速速度被精确，鲁棒地跟踪。

本文的研究工作包括:

（1）基于单目摄像头的并行跟踪及地图建立算法(PTAM)的研究。

（2）移植到 android 上的技术细节。

（3）利用 Ogre 引擎实现更好的现实增强效果；利用 bullet 实现物理碰撞。

关键词：非线性最小二乘；高斯牛顿法；PTAM；Android NDK；渲染管线

Parallel Tracking And Mapping System Porting To Android

Abstract

This paper presents a method of estimating camera pose in an un- known scene. While this has previously been attempted by adapting SLAM algorithms developed for robotic exploration, we propose a system specifically designed to track a hand-held camera in a small AR workspace. We propose to split tracking and mapping into two separate tasks, processed in parallel threads on a dual-core com- puter: one thread deals with the task of robustly tracking erratic hand-held motion, while the other produces a 3D map of point features from previously observed video frames. This allows the use of computationally expensive batch optimisation techniques not usually associated with real-time operation: The result is a system that produces detailed maps with thousands of landmarks which can be tracked at frame-rate, with an accuracy and robustness rivalling that of state-of-the-art model-based systems.

Main works in this paper can be described as follow:

(1) the research to monocular camera parallel tracking and maps, in the study of the tracking system.

(2) Porting the PTAM system to android.

(3)Using OGRE engine to render.

KeyWord: NonLinearWLS;Gauss-NewthonMethod;OpenGL; Android ndk

目　录

1 绪论

机器人的诞生和机器人学的建立和发展是 20 世纪自动控制最具说服力的成就，是20 世界人类科学技术进步的重大成果，而作为机器人中重要分支之一的移动机器人更是给人们带来了无限的惊喜。移动机器人的研究始于 20 世纪 60 年代末期。斯坦福研究所（SRI）的 Nils NVILSSEN 和 Charles Rosen 等人，在 1966 年至 1972 年中研制出了取名 Shakey 的自主移动机器人，其目的是研究、应用人工智能技术以及在复杂环境下机器人系统的自主推理、规划和控制；20 世纪 70 年代末，随着计算机的应用和传感器技术的发展，移动机器人研究又出现了新的高潮；20 世纪 90 年代以来，以研制高水平的环境信息传感器和信息处理技术，高适应性的移动机器人控制技术、真实环境下的规划技术为标志，开展了移动机器人更高层次的研究[1]。目前，移动机器人正向着具有自组织、自学习、自适应的智能化方向发展，导航能力的高低是移动机器人智能化水平的重要体现。

1.1 同步定位与建立地图(SLAM)

本文展示了一种在未知环境中估计摄像头姿态的方法。这种尝试是由 SLAM 算法改编而来的，而 SLAM 就是为机器人探索开发的，所以本文提出的这个系统也是专为在小空间范围跟踪一个手持摄像头设计的．我们的系统分开跟踪和地图建立在两个独立任务，在多核电脑上并行运行。tracking 线程处理不稳定的手持姿态，mapping 线程处理出现在帧上的 3D 特征点。计算上，这种算法允许一部分性能花销的优化技术与实时操作无关：这个系统的输出是产生有数千个标记点的详细地图。这些标记可以以帧速速度被精确，鲁棒地跟踪。

SLAM 的动机在于不需要先验地图而能精确定位，并且它维持的地图能够随环境的变化进行扩展和自适应。SLAM 通过机器人的传感器探测环境特征，然后由机器人的位姿估计特征位置，并把地图特征存入地图。当特征被重复观测时，特征位置的不确定性逐渐降低，此时可用这些特征位置去提高机器人的位姿估计，使得能够得到收敛的位姿估计与环境地图。

近十几年来，SLAM 发展非常迅速，已经成为了自主导航领域的研究热点，许多方法均得到了成功的实践，硕果累累。以下根据 SLAM 中问题模型的描述差异将 SLAM 算法分成了 4 类：

1．基于状态空间，包括扩展的卡尔曼滤波算法以及压缩扩展的卡尔曼滤波算法等。

2．基于样本集，包括基于粒子滤波的 slam 算法，DP-SLAM 算法，Fasti．OSLAM

算法，Fast2．OSLAM 算法等。

3．基于差异表达，例如扫描匹配 SLAM 算法。

4．基于信息空间，包括信息滤波 SLAM 算法以及稀疏连接 SLAM 算法等。

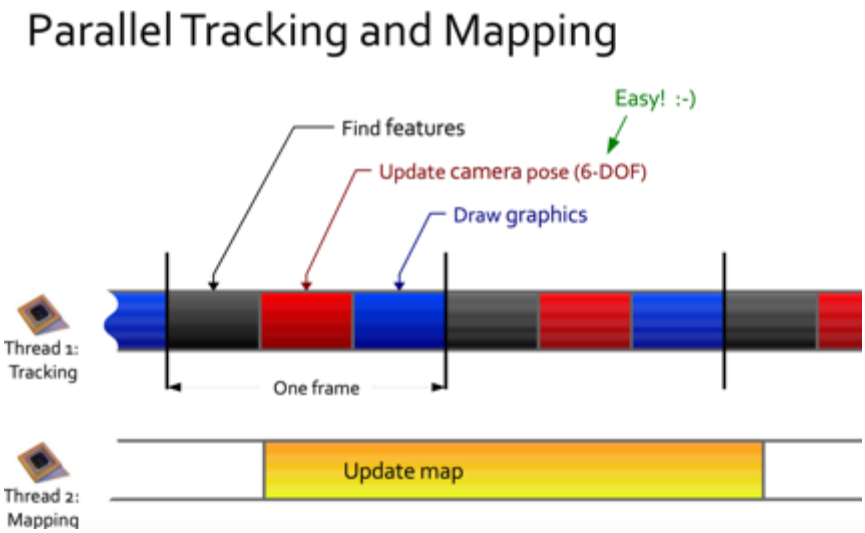
在上述四种问题模型中，比较流行的模型包括基于状态空间的 SLAN1 算法以及基于样本集的 SLAM 算法，其中又以基于扩展的卡尔曼滤波算法和 FastSLAM 最为经典。

1.2 增强现实

增强现实（Augmented Reality,简称 AR）是一种实时地计算摄影机影像的位置及角度并加上相应图像的技术，这种技术的目标是在屏幕上把虚拟世界套在现实世界并进行互动。这种技术由 1990 年提出。随着随身电子产品运算能力的提升，由于其虚实结合的独特性，目前在许多领域都有巨大的发展潜力和应用前景.

1.3 并行跟踪与地图建立

本章将研究并行跟踪及地图建立（PTAM）算法，PTAM 是 Parallel Tracking and Mapping 的简称，由英国牛津大学的 Georg Klein 和 David Murray 于 2007 年在 ISMR 提出[2]。PTAM 系统利用双核处理器进行并行处理技术，将跟踪与绘制分散到两个线程中，一个线程负责跟踪相机的姿态，同时绘制虚拟的模型，另一个线程负责建立场景的模型和绘制场景的地图。



前者会产生一个跟踪的结果，需要保证实时性，而后者则为前者提供数据源，可以是非实时的。这样进行拆分，使得某些只能在 SFM 中使用的算法也能够应用到 SLAM 中，从而实现无需预先地图的增强现实功能。简单来说，PTAM 可以理解为使用单目摄像头和立体几何算法代替 SLAM 中的传感器模型。

1.4 PTAM 与现实增强

PTAM 和 AR 没有必然联系。PTAM 中，Geory Klein 博士只是在建立地图用 OpenGL 画出了一对眼球在地图原点。

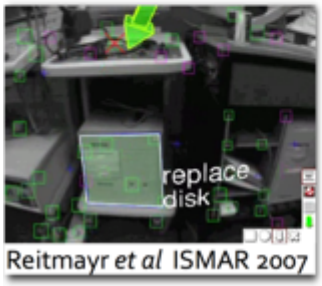
而如今的 AR SDK，如 Vuforia，ARTookit 等，已经有各种强大的功能。包括物体识别，加载更丰富表现力更强的三维格式。但是 PTAM 实现的 AR 与这些 SDK 的基本原理是不一样的，同时能实现一些它们无法完成的功能。

首先这些 AR 工具包的基本功能需要一个特定的靶标（maker），例如一个带图案的长方形纸片，然后识别这个 maker，假设出它四个顶点的空间坐标，根据照相机模型计算出当前位置的视图矩阵。这种方法稳定高效，各家 SDK 都是基于这种算法做改进，例如 Vuforia 就添加了智能识别卡片的功能，不需要特定图案。

但是这些方法在跟踪时都不能离开 maker，也就是摄像头画面看不到卡片跟踪就丢失。同时对现实周围的环境也没有感知能力。而 PTAM 就能做到对周围环境感知，同时能实时地找到画面内稀疏特征点的三维坐标。近年来 SLAM 应用于 AR 的成果：


Davison *et al* 2004

Williams *et al* ICCV 2007



Reitmayr *et al* ISMAR 2007


Checklov *et al* ISMAR 2007

1.5 研究目标和内容

本文的研究内容包括：

单目标定原理；特征提取；斑点搜索方法；仿射变换及模拟三维变换；模板搜索匹配；基与坐标变化；姿态估计；最小二乘；高斯牛顿法解非线性最小二乘；高斯牛顿法推导；M 估计迭权迭代；雅可比矩阵；图像雅可比矩阵；PTAM 跟踪方法；重定位方法；Bundle Adjustment；立体几何；对极几何；对极搜索；基础矩阵与本征矩阵；

五点法求解基础矩阵；单应性矩阵分解 RT 矩阵；RANSAC；线性摄像机模型；渲染引擎与游戏引擎；渲染管线编程；OpenGL 编程。

基于单目摄像头的并行跟踪及地图建立算法（PTAM）的研究是此次研究的主要内容，PTAM 是个设计精良的软件系统，其实现使用了很多视觉理论与算法，而 PTAM 论文是发表在 ECCV 顶级会议的文章，所以 Georg 博士在论文中并没有过多描述实现细节。

此次研究具体编程工作：

（1）编译运行调试原版 ptam 源码。

（2）移植到 android 上的技术细节。

（3）利用 Ogre 引擎实现更好的现实增强效果。

2 线性针孔摄像机模型与标定

2.1 照相机模型

照相机模型是照相机标定的基础和核心，只有确定了成像的模型，才能进一步的确定出照相机的内外参数。成像模型是三维空间中的物体到像平面的投影。理想的投影成像模型是在光学中心的投影，因此也被称为是针孔模型或者是线性摄相机模型。针孔模型假设物体的表面反射光均会经过一个小孔，然后投影到像平面上，这符合光的直线传播原理。针孔模型是由光心、光轴以及成像平面组成的。由于针孔比较小，成像的透光量不足，所以需长时间曝光，并且得到的图像并不十分清晰。实际的摄像系统通常是由透镜或透镜组成。这两个模型有着相同的成像关系，像点是图像平面和光心连线的交点。因此，可以用针孔模型作为照相机成像的模型。

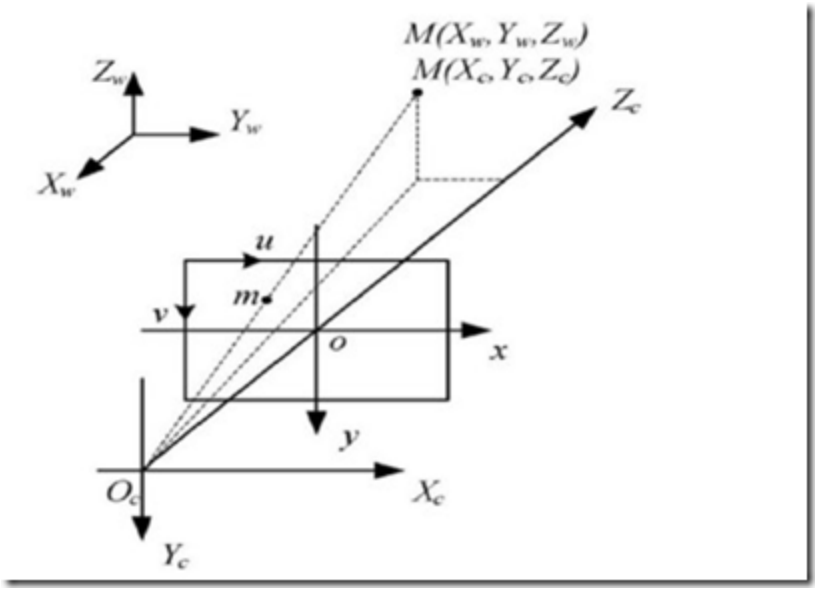


图 2-1

如图所示，在计算机图形学和渲染引擎中，照相机模型主要由 3 个矩阵组成：模型矩阵，视图矩阵和投影矩阵。在计算机视觉中，还需要应用相机内参，把图像经过映射变换到图像平面。需要注意 PTAM 中跟踪线程和 Mapping 线程使用的线性照相机模型的投影矩阵是正交投影的方式。也就是点投影到图像坐标系的位置和摄像机坐标系下投影到 z=1 平面的位置是一样的。而得到结果后，把结果渲染到屏幕上时则使用透视投影的方式，这样的投影结果符合人眼观测时近大远小的效果。

为了映射地图上的点到图像上，首先我们把这些点从世界坐标系变换到以摄像机中心为中心点的摄像机坐标系，此时的摄像机的世界坐标位置由运动先验模型估计而来。通过左乘一个 4x4 矩阵完成这个变化。

$$p_{jC} = E_{cw}p_{jw} \tag{1}$$

矩阵E_{CW}包含一个旋转和一个平移分量，它是一个 SE3 类型变量。

2.1.1 摄像头畸变与标定

摄像头由于光学透镜的特性使得成像存在着径向畸变，可由三个参数 k1，k2，k3 确定。由于装配方面的误差，传感器与光学镜头之间并非完全平行，因此成像存在切向畸变，可由两个参数 p1，p2 确定。单个摄像头的定标主要是计算出摄像头的内参。焦距 f 和成像原点 cx，cy、五个畸变参数。一般只需要计算出 k1，k2，p1，p2，对于鱼眼镜头等径向畸变特别大的才需要计算 k3。以及外参。为了投影当前点到图像，我们使用一个已矫正的摄像机模型:

$$\binom{u_i}{v_i} = CamProj(E_{CW}p_{iw}) \tag{2}$$

采用针孔摄像机的投影方法，它可以描述摄像机的镜头畸变。假设 camera 参数是已知的：f_u，f_v 代表焦距，u_0，v_0 代表光心，w 表示曲变系数(在我们的简单模型中先设为 1)。

可以得出 CamProj 投影函数表达式：

$$\binom{u}{v} = \mathrm{CamProj}\begin{pmatrix} x_c \\ y_c \\ z_c \\ 1 \end{pmatrix} = \binom{u_0}{v_0} + \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix}\begin{pmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \end{pmatrix} \tag{3}$$

摄像机姿态的变化可以等价于给视图矩阵左乘一个运动矩阵 M:

$$E'_{CW} = ME_{CW} = exp(\mu)E_{CW} \tag{4}$$

M 和E_{cw}一样，是有 6 个自由度的 4x4 矩阵，可以最小参数化表示，通过 exp 函数转换为一个六维向量保存。通常前 3 个为位移，后三个代表绕 x，y，z 轴旋转的量化。在后续应用中，要完成跟踪，一个基本条件就是对式[3]的姿态的变化E_{cw}求微分。对矩阵求导等于对矩阵中的元素分别求导，以 u 为例:

u = CamProj(x，y，z，1)求微分得

$$\Delta u = \lambda\frac{z_c\Delta x + x_c\Delta z}{z^2},\ \lambda = f_u \tag{5}$$

同理计算Δv_c，整理可得

$$\begin{bmatrix}\Delta u\\ \Delta v\end{bmatrix} = \mathbf{F}\{\frac{1}{z_c}\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} - \frac{\Delta z}{z_c^2}\begin{bmatrix}x\\ y\end{bmatrix}\} \tag{5}$$

F 为焦距矩阵$\begin{bmatrix}f_u & 0\\ 0 & f_v\end{bmatrix}$，$\Delta x$，$\Delta y$，$\Delta z$分别为运动微分 motion 的 x，y，z 部分，下标 c 表示该点用摄像机坐标系表示时的位置。

2.2 准备跟踪数据

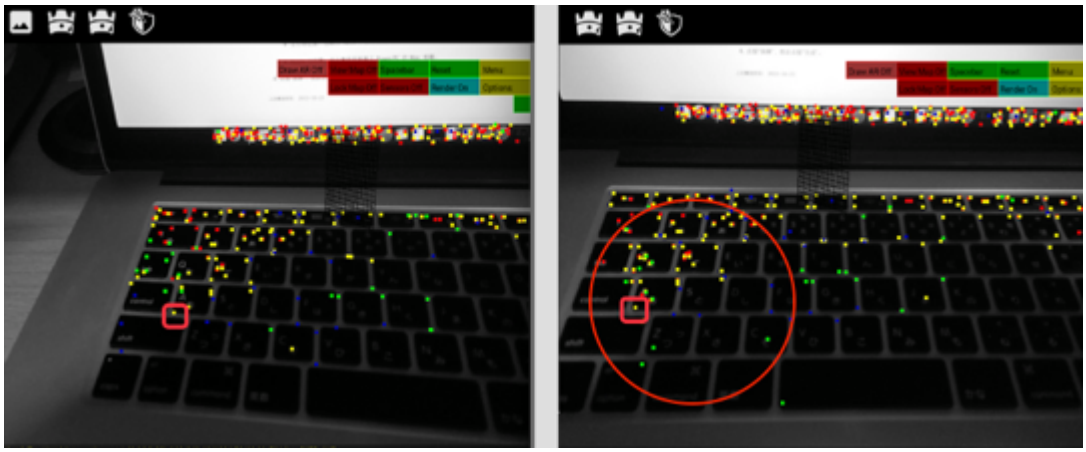
下一章将解析跟踪线程的流程。但实际工作中，构建线程是同时工作的。在跟踪之前已经完成了构建地图的初始化工作。此时我们至少有以下数据用于进行跟踪:

- 每个点的世界坐标系位置(mm)。
- 第一次观测到该点的帧的 ID 索引，称此关键帧为该点的源帧。
- 该点在帧图像中的位置(像素)，记为 rootPos(u，v)。
- rootPos 的右边像素(u+1，v)和下一像素(u，v+1)在平面 P 中的投影。平面 P 在下一节定义。
- 每一关键帧的视图矩阵。
- 每一关键帧观测到的点的索引集合。
- 每个关键帧的灰度图像金字塔，每个四层。

实际上，保存这些数据就相当于保存地图，下次启动时可以直接加载地图免去初始化工作，直接开始跟踪。我们可以保存除这些数据外更多的中间变量免去加载地图后的部分计算。

3 斑块查找

在跟踪线程中，使用者变换摄像头姿态或位置得到新关键帧，新关键帧需要重新查找特征点，找到与上一帧匹配的点以实现点跟踪。为了快速精确地找到邻帧对应特征点，PTAM 使用一种基于斑块搜索的特征匹配方法。首先从照相机采集一帧灰度图像，由运动估计模型计算一个估计姿态。将地图中的特征点投影到图像上。然后进行斑块搜索。



图[3.1] 斑块搜索

其基本原理就是从源图像中生成一个搜索模板，该模板可以补偿摄像机运动造成的尺度和角度变换。用该模板在目标图像中搜索找到最佳的匹配位置。具体步骤如下：

1. 在当前帧查找 FAST 特征点。 特征查找方法有多种，本文不再赘述。
2. 计算源帧在当前姿态下的一个变形矩阵 A。
3. 通过 A 的行列式决定搜索层。
4. 通过 A 生成搜索模板。
5. 使用该图像模板在当前帧一定范围内搜索匹配，找到斑点位置。
6. 使用反向合成法计算该点所在的亚像素精确位置。

3.1 仿射变换

本节介绍第 2 步，介绍前了解一种常见的 2D 变换：仿射变换。在目标跟踪应用中，图像目标通常会发生旋转和缩放等变化，而另一方面，目标与图像采集设备之间的距离通常较远，其三维变化常可以用二维变化来近似，所以，在兼顾算法效果和计算效率的考虑之下，经常可以选择使用了六参数的仿射变换运动模型。 在我们的模型中，摄像机运动变换导致的图像变换也可以用仿射变化近似模拟，参考图[3-1]。

仿射变换是射影几何中的一种基本变换。仿射变换有 6 个自由度，分别是两个平移,两个旋转，一个错切,和一个缩放。

二维欧氏空间上的仿射变换可以表示为：

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} T_u \\ T_y \end{bmatrix} \tag{6}$$

当源帧图像生成模板时执行上面的这些投影可以确保扭曲矩阵能补偿透视投影、尺度变化以及镜头畸变造成的图像影响。A 即为上式中 a_n 组成的矩阵。它表示旋转，缩放，错切的合成变换。

仿射变换具有 6 个自由度。如果要对仿射矩阵线性求解，至少需要 6 对点对应。我们没有 6 组准确的点对应，但是有每个点关于空间的投影关系。利用这些关系也能求得仿射矩阵。

在求解仿射矩阵之前，下节引入一些概念，以更好地理解仿射变换乃至射影变换的实质。

3.2 基与坐标变换

假设向量与矩阵乘积 Mx，我们可以把这个形式的矩阵 M 理解为对向量 x 的基——即坐标系的声明，向量 x 可看作空间的一点。乘积的结果向量，就是 x 在标准正交基 I 下的空间坐标，或者是坐标值不变，只是换了一个基。

一个变换 T(x)，其 M 下的变换矩阵为 A，则变换过程可以表示为 MAx，其位置就是 MAx，而变换后的坐标，从数值上来讲也可以认为是相对位置没有变而基变了：从 x 以 M 为基变换到了 x 以 MA 为基。变换是相对的。所以这个变换 T 的矩阵 A，也可以理解为是对基 M 的一个变换。也就是运动的实质，就是基变换。仿射变换的实质也是基变换。

从这个角度理解，正交基下的变换有特殊的性质，以仿射变换为例，对点(1,0)做 2D 变换，可以表示为：

$$p = \begin{bmatrix} m1 & m2 \\ m3 & m4 \end{bmatrix} [1,0]^T = I[m1, m3]^T \tag{7}$$

上式 x 方向单位向量做了个基变换得到一个向量，这个向量正好是原基的第一列。该基的 Y 方向化为单位正交基的坐标为[m2,m4]。也就是正交基下，两个向量按列排序组合即可组成新的基。

3.3 计算扭曲矩阵

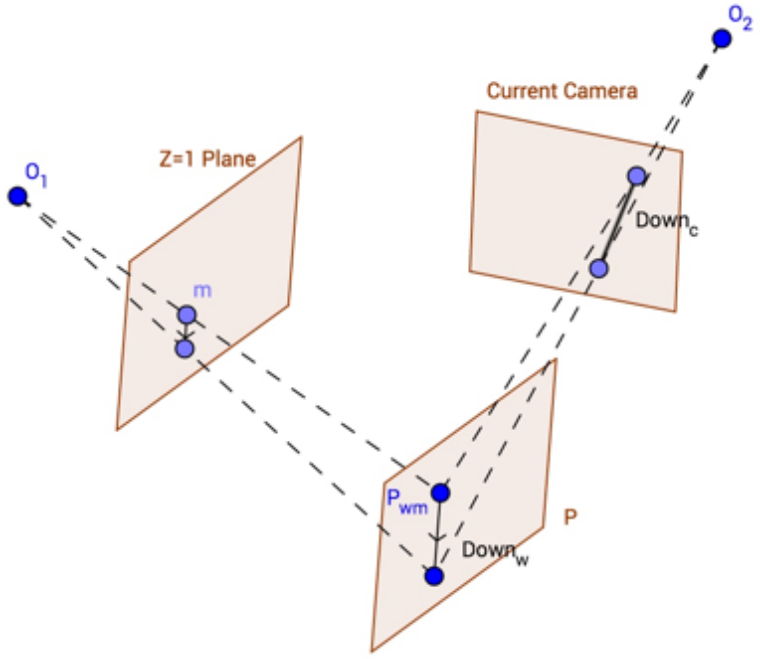
本节介绍 PTAM 中计算仿射变换矩阵的过程，同时也能证明，在静态场景下，仅摄像机的移动造成的图像变换可以用仿射变换来代替。下面简称第一次找到该点的关键帧为源帧，该帧对应的视图矩阵为源视图矩阵，记为E_{cws}。当前帧的视图矩阵用E_{cw}

表示。但是当前还没有估算姿态，所以这里使用的是上一帧的姿态。

我们执行仿射变化是为了用 2D 变换近似模拟 3D 变换，最终目的是为了在两幅不同摄像机姿态的图像中找到对应着同一三维点的一对像素。

每个地图点有一个从属关键帧的摄像机坐标系的位置。这个位置是不精确的。我们把该位置看做一个平面中的一个点。假设这个平面为 P。这个平面的法向量指向源帧摄像机。还假设一个平面 z=1，垂直于摄像机坐标系的 z 轴，这个就是经摄像机内参矫正后的图像平面。平面 P 平行于 Z=1 平面。

如图所示：



图[3-1] 仿射变换近似替代三维变换

取匹配点，也是斑点中心点 m(u,v,1)，的正下方一个像素(u,v+1,1)，连接该像素和匹配点得到一个竖直向下的向量 d。还已知 m 点在世界坐标系的位置，应用原始帧的针孔摄像机模型，可以计算出向量 d 在平面 P 的投影$\mathrm{Down_w}$。如果取 m 右边一像素，同样可得“右向量”在 P 的投影$\mathrm{Down_r}$。显然 P 离摄像机越远，上述向量的模越小。然后将$\mathrm{Down_w}$左乘E_{cw}得到该向量在当前视图的投影向量：

$$\mathbf{Down_c} = [\Delta u_d, \Delta v_d]^T = E_{cw} Down_w \tag{8}$$

由前一节矩阵与基的结论可知。$[\Delta u_d, \Delta u_r]$为原向量(0,1)在新的基下的表示，该基即可描述仿射变换，仿射变换的实质也就是基变换。同理，下标 r 表示$Right_w$方向的投影，$[\Delta u_d, \Delta u_r]$对应（1，0）。以斑点中心点为原点，把它们组成一对新的基，于是

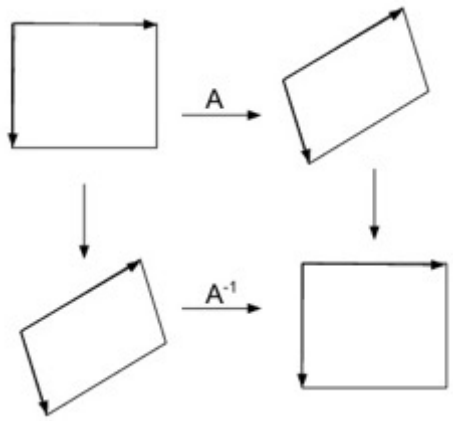
我们通过投影关系得到了一个等价于仿射变换矩阵的基。那么该基下的点在正交基下的值为：

$$[u_c, v_c]^T = \begin{bmatrix} \Delta u_d & \Delta u_r \\ \Delta v_d & \Delta v_r \end{bmatrix} [u_s - u_0, v_s - v_0]^T \tag{9}$$

图[3-2] 的上半部分是上式的结果，显然变换的结果转为笛卡尔坐标系后是个平行四边形。为了后续 SSD 计算方便，我们需要的是一个 8x8 的方形模板。所以还要做一个对基做一个变换，以产生一个正交基下的坐标值，用 A 表示等式右边的 2x2 矩阵，它表示仿射变换的一部分，做如下基变化：

$$I\begin{pmatrix} u_T \\ v_T \end{pmatrix} = A\begin{pmatrix} u_S \\ v_S \end{pmatrix} \rightarrow I\begin{pmatrix} u_S \\ v_S \end{pmatrix} = A^{-1}\begin{pmatrix} u_T \\ v_T \end{pmatrix} \tag{10}$$

最终得到A^{-1} 即是我们需要的扭曲矩阵。



图[3-2] 基变换

同时这两个单位投影向量组成的平行的四边形面积即为矩阵 A 的行列式：$S = \Delta U_R * \Delta V_D - \Delta V_R * \Delta U_D$。当当前摄像机比源帧摄像机位置更靠近斑点时，面积更大。此时画面尺度变大，根据这个面积值大小决定使用源帧图像金字塔的那一层，以更好地和当前图像匹配。

有了基，我们还需要坐标值才能得到变换后的结果。

式子中，下标 T 表示变换后的坐标值，S 表示源图像。现在使用u_T, v_T, A^{-1}计算得到源帧下对应的四个点。将它们围成的平行四边形采样，双线性插值到目标图像，即得到这个 mapPoint 在当前视图中的 8x8 像素的搜索模板。这个过程可以使用 TooN 库

CVD::transform(aLevels[p.nSourceLevel].im,mimTemplate,m2,vec(p.irCenter),vec(mirCenter));

的 transform 函数完成。

参数分别为源图像，目标图像，扭曲矩阵，源图像采样中心点（对应图[3-2]的原点），目标图像中心点中心点。采样方向和单位长度由扭曲矩阵决定，尺度由目标图像大小决定。

以上就是扭曲矩阵 A 补偿尺度和角度变化的原理。

3.4 模板搜索

模板匹配步骤可以应用上一步生成模板在目标图像中寻找匹配位置，找到的点和源点形成一对点对应，用于后续姿态估计算法中。上一节使用源点对应源帧的图像、变形矩阵、该点在图像上的投影位置，生成一个 8x8 像素的模板。然后计算图像金字塔层数，从目标图像的图像金字塔层中得到目标图层，以源位置为中心，在一个合适的半径内搜索，找到最匹配的位置。搜索过程如下：首先在该范围内找到所有角点，在此系统中角点是该图像的 FAST 特征点，然后在角点位置应用该模板计算 SSD 算法得分，迭代每个点选择差异值最小的角点作为最佳匹配位置。假如这个得分小余某个预设的预置，那么就认为在该范围找到了该斑块和点对应。

3.5 计算亚像素位置

在上面所说的搜索中，当搜索层数大于 0 时，得到的模板最佳匹配位置是所在图层的像素位置。后续跟踪时还需要它的第 0 层的图像坐标，这个位置可以使用根据图像邻点关系，使用反向合成法[18]做图像对齐，然后获得第 0 层坐标，得到的坐标称为亚像素位置。

4 姿态估计

在斑点搜索之后，我们有一群新找到的点集在当前视图的投影，由于测量误差的关系，点集的坐标位置并不是精准的，我们需要从这些不太准确的数据中估算新的摄像机姿态和位置。把这群点集的坐标值作为观测值，把运动模型计算的点位置作为估计值。求某个运动参数向量，使估计值与观测值的残差平方和最小，即为我们要计算的姿态更新增量[4]。

姿态估计分为粗搜索和精搜索两个阶段。粗搜索搜索范围更广，使用点数较少。精搜索搜索范围较窄，使用上千个点样本。两个阶段解决的一样的问题，但精搜索样本大对计算资源消耗极大，因此引入一些优化方法。

4.1 姿态更新

斑块搜索中得到的斑块第 0 层位置 u,v 作为独立的测量值，把 j 组测量值线性组合成一个向量：$P_j = \{u_1, v_1, u_2, v_2, \ldots u_j, v_j\}$。而运动模型估计的 u,v 为：

$$[\hat{u_j}, \hat{v_j}]^T = CamProj(exp(\mu)E_{cw}\mathbf{p_j}) \tag{11}$$

用 ^ 区分观测值和估计值。观测值为斑块搜索中找到的点坐标位置。e_j代表重投影误差向量。CamProj 函数可以将摄像机坐标系中的值转到图像坐标系，则测量残差为：

$$r_{2j} = u_j - \hat{u_j}; r_{2j+1} = v_j - \hat{v_j} \tag{12}$$

那么解$\sum_i r_i = min$就是一个最小二乘问题。把参数向量$\mu = [\alpha, \beta, \gamma, T_x, T_y, T_z]$作为待估计参数。$\hat{u_j}(p_j), \hat{v_j}(p_j)$展开是一个非线性方程组。即估计运动参数的实质就是求解非线性最优解问题。下面介绍用高斯牛顿法求解非线性最小二乘问题。

4.2 最小二乘

最小二乘问题出现在许多应用中。它可以被看作是一个最优化问题（optimization problem）。例如，我们搜集了一些数据，然后想用这些数据对一个模型进行拟合，并使得这些数据和求解到的模型在某种意义上达到最佳拟合。一般可以将最小二乘问题紧凑地表示为：

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) \tag{13}$$

其中 x 是模型参数组成的向量，r 被称为残差向量（residual vector），可以表示预期值和测量值之间的差异，是由单独的残差所组成的列向量:

$$r_j(x) = y(\tilde{x}_j|x) - \tilde{y}_j \tag{14}$$

$$y(\alpha|x) = x_1X_1(\alpha) + x_2X_2(\alpha) + \cdots + x_nX_n(\alpha) \tag{15}$$

函数 y 是我们的模型函数（或期望模型），它根据给定的样本点$\tilde{x_j}$以及模型参数 x 返回预期值。这个值再和$\tilde{y}_j$进行比较，这里$\tilde{y}_j$为观测值。波浪线标记(~)用来区分从模型参数(x)和模型函数(y)获获取的数据点。

4.2.1 设计矩阵

最小二乘问题可以用矩阵表示，引入设计矩阵 A：

$$A_{ji} = X_i(\tilde{x}_j) = \frac{\partial r_j}{\partial x_i} = \frac{\partial}{\partial x_i}\left[y(\tilde{x}_j|x) - \tilde{y}_j\right] = X_i(\tilde{x}_j) \tag{16}$$

x 是要拟合模型的参数向量，X 是基函数（basis functions）。基函数可以是参数向量 x 的任何函数。把最小二乘化为矩阵形式：

$$\| r \|^2 = r^T r = (Ax - \tilde{y})^T(Ax - \tilde{y}) = x^T A^T Ax - 2x^T A^T \tilde{y} + \tilde{y}^T \tilde{y} \tag{17}$$

要对$\| r \|^2$进行最优化求解。可以通过计算上述表达式对于参数 x 的偏导数，并令它等于零：

$$2A^T Ax - 2A^T \tilde{y} = 0 \tag{18}$$

求解得到：

$$x = (A^T A)^{-1} A^T \tilde{y} \tag{19}$$

4.2.2 非线性最小二乘

对于线性最小二乘问题，我们将问题求解归结到对方程组 Ax=b 进行求解，而对于非线性最小二乘问题，我们将会有诸如 A(x)x=b 形式的方程组，即设计矩阵也依赖于模型参数。在非线性的情况下能够使用的最好方法是为参数向量提供一个初始值，然后让这个初始值逐步趋近于正确的解向量。这种方法会在每一步对参数向量进行更新，并重复这个过程直到收敛，因此是一种迭代算法。此时r_j是一个函数向量，它对参数向量求导的矩阵称为雅可比矩阵。在最小二乘问题中，它的定义和设计矩阵的定义相同：

$$J_{j,i}(x) = \left[\frac{\partial}{\partial x_i} r_j\right] \tag{20}$$

但与设计矩阵不同的是，它依赖于模型参数 x。由定义可知，我们可以直接从我们的模型函数构造雅可比矩阵：

$$J(x)=\begin{bmatrix}\frac{\partial}{\partial x_1}y(\tilde{x}_1|x) & \frac{\partial}{\partial x_2}y(\tilde{x}_1|x) & \cdots & \frac{\partial}{\partial x_n}y(\tilde{x}_1|x)\\ \frac{\partial}{\partial x_1}y(\tilde{x}_2|x) & \frac{\partial}{\partial x_2}y(\tilde{x}_2|x) & \cdots & \frac{\partial}{\partial x_n}y(\tilde{x}_2|x)\\ \vdots & \vdots & \ddots & \vdots\\ \frac{\partial}{\partial x_1}y(\tilde{x}_m|x) & \frac{\partial}{\partial x_2}y(\tilde{x}_n|x) & \cdots & \frac{\partial}{\partial x_n}y(\tilde{x}_m|x)\end{bmatrix}$$

每一种迭代算法在开始计算时都需要提供参数向量 x 的一个初始估计值 x(0)，并用下面的形式计算在当前迭代步骤中新的参数估计值：

$$x_{(k+1)}=x_{(k)}+p_{(k)} \tag{21}$$

p_k表示第 k 次迭代的修正因子，x（k）表示第 k 次迭代的值。此外，目标函数的函数值应该是在每一步变得更小：

$$f(x+p)<f(x) \tag{22}$$

通过对不等式左边进行一阶泰勒展开我们有：

$$\begin{array}{ccc} f(x)+g(x)^T p & < & f(x)\\ g(x)^T p & < & 0\end{array} \tag{23}$$

上式中：

g(x)表示目标函数的梯度向量，即目标函数，或当前对目标函数的近似，相对于模型参数的一阶导数的向量。它描述了目标函数在某点 x 的斜率。

这样就给出了以目标函数的梯度表示的减小条件。

同时给出目标函数的海森矩阵 H(x)的定义：目标函数（或当前对目标函数的近似）相对于每一种参数组合的二阶偏导数矩阵。它描述了目标函数在某点 x 的曲率（curvature）。目标函数的二阶偏导可以用 hessian 矩阵近似表达。

$$\nabla^2 f(x)\approx J(x)^T J(x)=H(x) \tag{24}$$

4.3 高斯牛顿法

应用高斯牛顿法可以解决非线性最小二乘问题。由于 fi(x)非线性性，此时按梯度等于 0,得到的是一个非线性方程组，求解困难。常用的基本思想是用一系列线性最小二乘问题求解该非线性最小二乘问题：设 x(k)是解的第 k 次近似，在 x(k)处将函数 fi(x)线性化，把问题化为线性最小二乘问题，求出第 k+1 次近似解 x(k+1);再从 x(k+1) 出发，重复此过程，直到达到迭代终止准则。

首先原方程可以化解为二次式最优解的问题：

$$f(x+p) \approx f(x) + p^T \nabla f(x) + \frac{1}{2} p^T \nabla^2 f(x) p = m \tag{25}$$

尝试模仿牛顿法得到牛顿法迭代步骤公式，实际就是上式对向量 p 求导令等于 0,然后变形，得到牛顿迭代定义：

$$p_{(k)} = -\nabla^2 f_{(k)}^{-1} \nabla f_{(k)} = -H_{(k)}^{-1} g_{(k)} \tag{26}$$

此时按牛顿法的思路用 f 的 hessian 矩阵计算上式结果太复杂。如上节所述，用雅可比矩阵近似替代目标函数的二阶偏导。所以上式也可以写为：

$$\begin{aligned} \mathrm{H_k p_k^{GN}} &= -\mathrm{g_k} \\ \mathrm{J_k^T J_k p_k^{GN}} &= -\mathrm{J_k^T r_k} \end{aligned} \tag{27}$$

对上式求解 P 即为高斯牛顿法迭代定义。完整的迭代过程如下：

1. 取初始点 x (1) ,置精度要求 ε ,令 k=1;

2. 计算雅可比矩阵$\mathrm{J_k}$；

3. 若$\left\|J_k r\right\| \le \varepsilon$,则停止计算，x(k)为所求 k 的极小点;否则解以上线性方程组 。得到 Gauss-Newton 方向p_k^{GN}；||表示取范数。

4. 令$x(k+1) = x(k) + p_k^{GN}, k = k+1$转 2;

由迭代步骤可知解非线性最小二乘最优化问题仅需要雅可比矩阵和残差向量。实际编程中使用 TooN 库的 wls 类解姿态更新的最小二乘问题。

4.4 图像雅可比

现在我们的问题就变成前一节所说的纯数学问题：求出哪些摄像机运动参数能让目标函数达到最小值，参数向量μ相当于上一节的 x,观测值$\hat{u_j}$相当于$y(\hat{x_j}, x)$,而$\hat{x_j}$表示一组观测输入，比如点所在的世界坐标系x_w, y_w, z_w。

把运动估计模型得到的$\hat{u_1}, \hat{v_1} \ldots \ , \hat{u_j}, \hat{v_j}$ 组成为一个函数向量 F，F 对μ求偏导即为求解最小二乘所需的雅可比矩阵。首先 Camproj 函数分别对参数求偏导，用 X 代表该参数：

$$\frac{\partial u}{\partial X} = \frac{\Delta u}{\Delta X}; X = \mu[n] \tag{28}$$

由针孔摄像机模型可知：

$$p_c' = ME_{cw}p_w = Mp_c \tag{29}$$

M 为 4 阶运动矩阵。需要注意的，M 可以分解。例如第 l+1 次迭代时，M 可以分解为

$$\begin{bmatrix} r11 & r12 & r13 & \sum_j T_{xj} \\ r21 & r22 & r23 & \sum_j T_{yj} \\ r31 & r32 & r33 & \sum_j T_{zj} \\ 0 & 0 & 0 & 1 \end{bmatrix} = [\mathbf{R}_l, \mathbf{T}_l] \ldots [\mathbf{R}_2, \mathbf{T}_2][\mathbf{R}_1, \mathbf{T}_1]$$

$$p_{l+1} = Mp_c = [\mathbf{R}_l, \mathbf{T}_l]p_l \tag{30}$$

即第 l+1 次迭代时，p_c' 可以通过上一次迭代结果作为初值。记$p_c' = [x_c', y_c', z_c']$向量$p_c'$求微分有：

$$\Delta motion = [\Delta x, \Delta y, \Delta z]^T = \frac{\partial p_c'}{\partial X}\Delta X \tag{31}$$

分别将$\Delta motion$的$\Delta x, \Delta z$分量代入[4]式，得$\hat{u}$对变量 X 求偏导的通用公式：

$$\frac{\partial\, u}{\partial X} = \lambda \frac{z_c \frac{\partial x_c'}{\partial X} + x_c \frac{\partial z_c'}{\partial X}}{z^2} \tag{32}$$

λ为 x 方向的焦距$\lambda = f_x$。$记motion = [\frac{\partial y_c'}{\partial X}, \frac{\partial x_c'}{\partial X}, \frac{\partial z_c'}{\partial X}]$。物理意义上，motion 可以理解为摄像机做某个微分运动时造成该点在摄像机坐标系下的运动量。摄像机微分运动分为微分平移和微分旋转。

当 X 为 Tx 时，P_c' 对 Tx 求偏导得

$$\frac{\partial x_c'}{\partial T_x} = \frac{\partial(m_{11}x + m_{12}y + m_{13}z + T_x)}{\partial T_x} = 1$$

$$\frac{\partial y_c'}{\partial T_x} = 0$$

$$\frac{\partial z_c'}{\partial T_x} = 0$$

$.motion_{Tx} = [1,0,0]$，同理得

$$motion_{Ty} = [0,1,0], motion_{Tz} = [0,0,1]$$

当 X 为绕 x 轴的旋转量α时，矩阵Mp_c对α求偏导：

$$motion_{\alpha} = \left(\frac{\partial R}{\partial \alpha} + \frac{\partial T}{\partial \alpha}\right)\mathbf{p_c}$$

第二项为 0,第一项为绕 x 轴旋转的微分旋转矩阵。因为微分旋转量很小，所以有：

$$\sin(\delta\alpha) = 0, \cos(\delta\alpha) = 1$$

微分旋转矩阵$rot(x,\alpha)$为：

$$\frac{\partial R}{\partial \alpha} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & sin(\delta\alpha) & -cos(\delta\alpha) & 0 \\ 0 & cos(\delta\alpha) & sin(\delta\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

代入求偏导通用公式得：

$$motion_{\alpha} = [0, -z, y]$$

同理可得，对绕 y 轴旋转量β求偏导：

$$motion_{\beta} = [z, 0, -x]$$

对绕 z 轴旋转 γ求偏导:

$$motion_{\gamma} = [-y, x, 0]$$

将 motion 代入[35]式即得到$\hat{u_j}$ 对参数向量$[T_x, T_y, T_z, \alpha, \beta, \gamma]$ 求偏导的结果，$\hat{v_j}$ 对参数向量求偏导的的过程也类似。遍历所有点对参数向量求导便得到一个雅可比矩阵：

$$\begin{bmatrix} \frac{\partial \hat{u_1}}{\partial T_x} & \frac{\partial \hat{u_1}}{\partial T_y} & \frac{\partial \hat{u_1}}{\partial T_z} & \frac{\partial \hat{u_1}}{\partial \alpha} & \frac{\partial \hat{u_1}}{\partial \beta} & \frac{\partial \hat{u_1}}{\partial \gamma} \\ \frac{\partial \hat{v_1}}{\partial T_x} & \frac{\partial \hat{v_1}}{\partial T_y} & \frac{\partial \hat{v_1}}{\partial T_z} & \frac{\partial \hat{v_1}}{\partial \alpha} & \frac{\partial \hat{v_1}}{\partial \beta} & \frac{\partial \hat{v_1}}{\partial \gamma} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial \hat{u_j}}{\partial T_x} & \frac{\partial \hat{u_j}}{\partial T_y} & \frac{\partial \hat{u_j}}{\partial T_z} & \frac{\partial \hat{u_1}}{\partial \alpha} & \frac{\partial \hat{u_j}}{\partial \beta} & \frac{\partial \hat{u_j}}{\partial \gamma} \\ \frac{\partial \hat{v_j}}{\partial T_x} & \frac{\partial \hat{v_j}}{\partial T_y} & \frac{\partial \hat{v_j}}{\partial T_z} & \frac{\partial \hat{v_j}}{\partial \alpha} & \frac{\partial \hat{v_j}}{\partial \beta} & \frac{\partial \hat{v_j}}{\partial \gamma} \end{bmatrix}$$

将此雅可比矩阵和残差 r_j代入高斯牛顿迭代定义式，求得修正因子 P_N。编程实现上，采用 TooN 库的 wls 类可以简化以上操作。使用修正因子修正$\mu_{j+1} = \mu_j + P_N$即完成一次迭代。之后再运行上面的过程，投影获得估计值，计算雅可比矩阵，代入高斯牛顿迭代定义式。直到修正因子收敛，完成一次姿态更新。

4.5 精搜索时使用线性最小二乘

PTAM 跟踪系统中，第一次搜索仅选取至多 40 个跟踪点，在大范围内搜索以更新姿态。第二次为精搜索，在精搜索中，我们要计算数千个点。如果每次迭代都重新投影这些点，重新计算它们组成的非线性方程组的雅可比矩阵。那么消耗的计算资源是巨大的。PTAM 采用的策略是先非线性计算一次，再线性计算两次，作为一个周期。通常三个周期后修复因子已经到了10^{-7}精度的极小值。最后再做一次非线性计算，完成一次精搜索的姿态更新。

在前面的非线性解最优解中，提到了目标方程是非线性的，以其中求 u 变量为例：

$$u_j = Camproj(\mu, p_c) = u_0 + f_u \frac{x_c}{z_c} \tag{33}$$

$$[x_c, y_c, z_c]^T = ME_{cw}\mathbf{p_{wj}}, M = exp(\mu) \tag{34}$$

显然 Camproj 关于 $\boldsymbol{\mu}$的方程是非线性方程。用 F 表示$Camproj(p_{1\ldots j})$方程组，$\mathbf{x}$表示μ参数向量，J 为该方程组的雅可比矩阵。F 是个多元方程，F(x)在 $\mathbf{p}$ 处的多元一次泰勒展开为[17]:

$$F(\mathbf{x}) \approx \tilde{F}(\mathbf{x}) = F(\mathbf{p}) + J_F(\mathbf{p}) \cdot (\mathbf{x} - \mathbf{p}) \tag{35}$$

这是个线性方程，变形一下得到：

$$\tilde{F}(\mathbf{x}) - F(\mathbf{p}) = J_F(\mathbf{p}) \cdot \mathbf{P_N} \tag{36}$$

上式展示了用近似的线性方程逼近结果。等式左边可以是$\Delta\hat{u_j}$或$\Delta\hat{v_j}$。其逼近的方向是上一次非线性逼近时计算的各个梯度方向，增量也是上一次非线性迭代计算的 P，重新计算残差即完成一次近似逼近的迭代。这样的逼近的结果是近似的，但是有可能不准甚至远离结果，这取决于得到的修正引子$\Delta\mu$,当它取极小值时，得到的结果和重计算投影公式结果相同。取值越大，越偏离$F(\mathbf{p} + \mathbf{P_n})$的结果。所以不能做多次近似逼近。

伪代码如下：

```
迭代10次：
    第0 4 9次迭代时：
        bNonLinearIteration = true;
    否则：
        bNonLinearIteration = false;
    迭代可用点集：
        if(bNonLinearIteration)
            重新投影计算该点图像坐标系估计位置;
        else
            使用[3.1式]线性逼近更新:v2Image += m26Jacobian * v6;
    if(bNonLinearIteration)
        重新计算雅可比矩阵J;
    v6Update = CalcPoseUpdate(J,r)
    mse3CamFromWorld = SE3<>::exp(v6Update) * mse3CamFromWorld; //修正
                    视图矩阵
    迭代loop
```

4.6 稳健回归(M 估计)方法简介

由于测量中系统可能存在粗差。个别异常大的残差的出现会导致平方和迅速增大。为了达到平方和为最小的目的，估值必然要迁就那些离群值[13]。如果用增长较慢的残差函数代替平方和函数，也许就能获得一定抗粗差性的估计值。M 估计正是基于这样的想法，在 60 年代由 Huber 提出来[8]。

M-estimators 是一类广泛估计函数，定义为所给数据上的最小和函数。最小平方估计和极大似然估计都是 M 估计法[8]。M 估计法由鲁棒的数据作为运行保证。一般地，一个 M-estimation 定义为一个估计函数为 0 的情况。这个估计函数经常是一些统计函数。比如令一个由参数定义的极大似然函数为 0，因此一个极大似然估计值往往是一个能量函数取得极值得点。在很多应用中，这样的 M 估计可以用于参数估计。

在最小二乘中，观测方程为：

$$r^T Pr = min$$

其中权重矩阵 *P=diag(p1,p2...)*。也就是每一项多乘以一个权重因子。P 为权重引子，由权函数计算获得。权函数是关于残差的函数。在 PTAM 中，我们使用 tukey 发明的 biweight 权函数做迭权迭代：

$$w_i = \begin{cases} 0 & \text{if } |u_i/c| > 1 \\ (1-|u_i/2|)^2 & \text{if } |u_i/c| \leq 1 \end{cases}$$

式中 c 一般 4.685。$u_I = \frac{e_I}{s} = 0.6745 \cdot \frac{e_i}{med(\ \ |e_i - med(e_i)|\ \ )}$。med()为中位数，s 为残差尺度。在跟踪系统中，TooN 库的 wls 类已经封装好了计算最小二乘时加入权重的方法。综合上一节，只需要迭代每个点时这样添加数据：

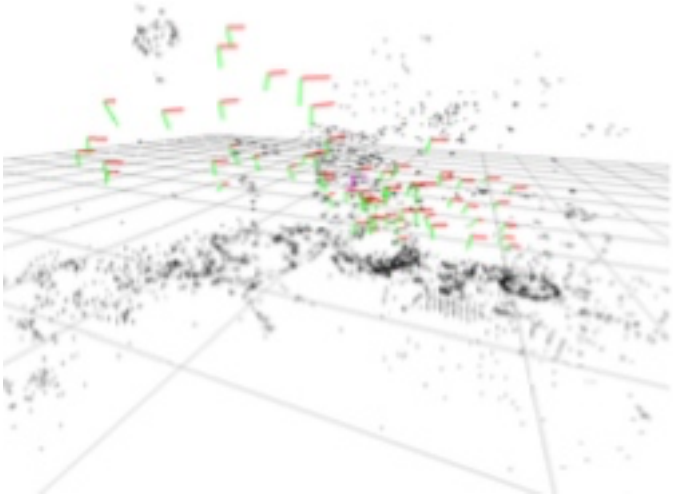
*wls.add_mJ(v2[0], TD.dSqrtInvNoise * m26Jac[0], dWeight);*

参数分别是上一节计算得到的残差，雅可比矩阵的第 i 个方程的梯度向量，权重。添加完后调用 wls.compute()得到结果。其实质就是求解高斯牛顿迭代定义式中的修正因子向量。

5 重构地图

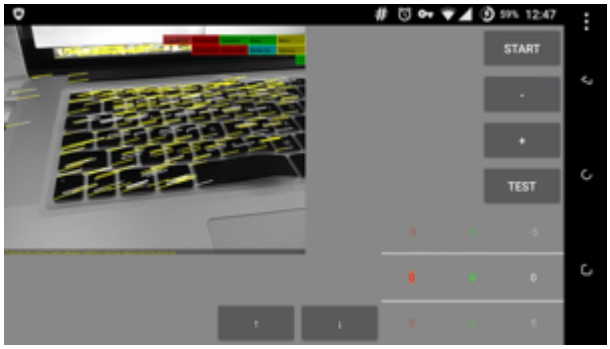
地图构建系统主要是建立 3D 点云图的过程。地图的创建发生在两个不同的阶段:第一个阶段，绘制出初始的地图;第二个阶段，通过等待关键顿，加入新的特征点，不断的对初始地图进行扩增和更新。与跟踪线程不同，Mapping 线程不要求实时性。

用可视化工具来看地图，也就是第二章提到的跟踪前准备数据，地图是这样的:



图[5-1] 地图数据可视化

在地图的初始绘制阶段，可以使用 5 点算法或单应性分解生成初始化地图，但是需要使用者的配合。使用者首先将网络相机放到需要跟踪的场景中，然后按下一个键，通知系统第一个关键帧已经获取，接下来，使用者需要慢而平稳的将相机平移一段距离，然后再次按下一个键，这时第二个关键帧就被记录下来。在使用者移动网络相机的过程中，第一帧中的特征点会被跟踪，以获得其在第二帧上的对应位置。这样做，系统就获得了一组二维点的对应关系，然后估计两视图相对[R|T]矩阵和三角化特征点，



最后通过边界调整生成最终的地图。

图[5-2] 平移相机初始化地图

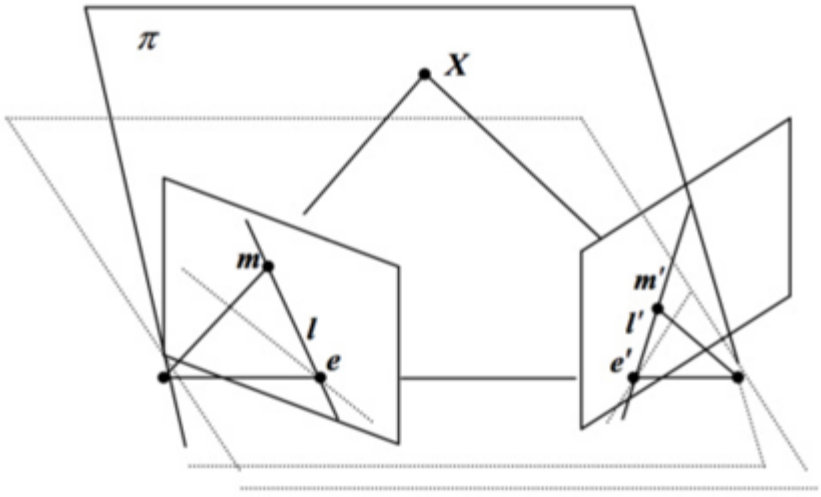
有两种方法可以估计当前[R|T]矩阵。一种方法是先求单应性矩阵和极点得到基础矩阵。然后对基础矩阵做 SVD 分解得到它的 RT 矩阵[14]。

另一种方法是直接 Homography 矩阵分解得到 [R|T]矩阵[15]。实际 PTAM 中后一版本的代码里用的都是这种方法。

5.1 对极几何、基础矩阵与本征矩阵

在本节中，主要介绍两个视点像间的几何关系,即所谓的两图像像间的极几何[1]。极几何是两图像的点线关联关系。对应点在对应线上，这种关联关系 可用所谓 的基本矩阵 代数描述。 基本矩阵有 7 个自由度,从 8 对像点对应可以线性唯一 确定。在射影等价意义下,基本矩阵确定两像所对应的摄像机矩阵。以下对极几何中的一些概念:

- 基线：指左右两摄像机光心和的连线。
- 极平面：指任一空间点和两摄像机光心三点确定的平面，如平面 π 就是一个极平面，所有的极平面相交于基线。
- 极点：指基线与左右两图像平面的交点，用 e 和 e'表示。
- 极线：指极平面与左右两图像平面的交线，左图像或右图像中所有极线相交于极点。



图[5-3] 双视图对极几何

本征矩阵(Essential Matrix) E：它包含了物理空间中两个摄像机相关的旋转(R)和平移信息(T)。T 和 R 描述了一台摄像机相对于另外一台摄像机在全局坐标系中的相对位置。

基础矩阵(Fundamental Matrix)F：除了包含 E 的信息外，还包含了两个摄像机的内参数。由于 F 包含了这些内参数，因此它可以在像素坐标系将两个摄像机关联起来。

本征矩阵和基础矩阵具有如下性质:

$$(P_r)^T E P_l = 0 p_r^T F p_l = 0$$

Pr,Pl 为摄像机坐标系下的点 m 位置，pr,pl 为图像坐标系下的齐次坐标。

我们观察摄像机内参数矩阵 M，如果图像没有畸变，即 cx,cy 为 0，并且焦距进行了归一化处理，那么 M 就成了单位阵，此时本征矩阵 F 就等于基础矩阵 E，即 F=E。

5.2 基础矩阵估计的研究

目前已经有了很多利用图像点对来估算基础矩阵的方法。基础矩阵有 7 个自由度，从 8 对图像点对应可以线性唯一确定。基础矩阵估计方法可以分为 8 点算法和最小点对应算法。8 点算法主要有 Longuet-Higgins 的 8 点算法和 Hartley 的改进 8 点算法。前者计算简单，易于实现，但对噪声异常敏感; 后者则通过在计算前对二维数据进行规范化处理，减小了噪声对实验结果的影响。OpenCV 中主要使用这类方法。

然而实际情况中，不可能得到精确的图像点对应。DavidNister 提出的 5 点算法，基本思想是两幅图像之间的运动为纯平移运动时，给定 5 对图像对应点，则可以线性确定本质矩阵，通过与基础矩阵的转化关系，便可以确定基础矩阵。相比“8 点算法”和“6 点算法”，“5 点算法”所需求的点数更少，且对于平面图像不退化，实现精度更高。

5.3 地图初始化

估计基础矩阵可以还原[R|T]矩阵。另一种方法是对单应性矩阵做分解。基于分解单应性矩阵的方法，PTAM 初始化地图过程主要有以下几个步骤:

1. 从匹配到的点集中，随机选择至少 4 对点，并用它们计算这两幅图像的单应性。迭代几次选择最好的单应性矩阵。评分方法:遍历所有匹配点，用当前单应性矩阵得到第二帧的点位置，和观测值取方差和结果即为当前单应性得分。得分反映了当前射影变换和当前单应性矩阵的耦合度。

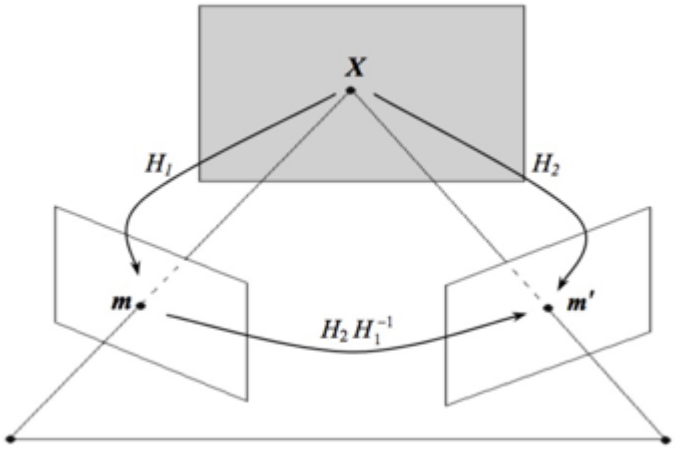
2. 上一步的得分计算实质也是个最小二乘的最优解问题。采用姿态估计时一样的策略，再次应用加权最小二乘剔除离群点集，可以进一步优化当前单应性矩阵。

3. 分解单应性矩阵。

4. 在可视条件内选择最好的分解。得到当前摄像头的视图矩阵。

5.4 单应性矩阵分解

如图所示，设 π 是不通过两摄像机任一光心的空间平面，令 X 是平面 π 的任一点，它在摄像机下的像为 m，记为齐次坐标[x,y,1]，右相机的像为 m' 。用'区分左视图和右视图。图中 m 和 m'所在的平面为 π 在两摄像机下的图像。



图[5-4] 两视图间的单应性

平面 π 到两个像平面之间存在两个单应性矩阵 H1,H2。使得$m = H_1X, m' = H_2X$。由于 π 不过任一光心，所以 m,m'之间存在一个二维射影变换 $H = H_2H_1^{-1}$使得

$$m' = Hm \tag{37}$$

假设两摄像机内参矩阵为K, K'，第二个摄像机相对第一个的位置为(R,t)，尝试求解 H 和 RT 矩阵的关系。

设 X 在摄像机坐标系下 坐标为$P_x(X_c, Y_c, Z_c)$则有：

$$X_c/x = Y_c/y = Z_c$$

n 为 M 所在平面法向量，d 为光心到该平面距离，X 所在平面方程为:

$$n^T P_x = d$$

用 R,T 表示两摄像机之间的旋转平移矩阵，则P_X, P_X'满足:

$$P_x' = \mathbf{R}P_x + \mathbf{t} \tag{38}$$

联立以上式子即可得单应性矩阵:

$$m' = (d\mathbf{R} + \mathbf{tn^T})m \rightarrow H = R + \frac{tn^T}{d} \tag{39}$$

5.4.1 计算单应性矩阵

以上的图像坐标是未经校正的，假设相机内参矩阵为 K，标定后的坐标记为 n，则有：

$$n = Kmn' = Mn = KHK^{-1}n$$

为了求解上式的，可以代入 4 个点直接线性求解，直接给出公式:

$$\begin{bmatrix} u_1' \\ v_1' \\ u_2' \\ v_2' \\ u_3' \\ v_3' \\ u_4' \\ v_4' \end{bmatrix} = \begin{bmatrix} u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1u_1' & -u_1'v_1' \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1v_1' & -v_1v_1' \\ \cdots & & & & & & & \\ \cdots & & & & & & & \\ \cdots & & & & & & & \\ \cdots & & & & & & & \end{bmatrix} \cdot \begin{bmatrix} m11 \\ m12 \\ m13 \\ m21 \\ m22 \\ m23 \\ m31 \\ m32 \end{bmatrix}$$

再由内参关系式得$H = K^{-1}MK$。由前面的推导可知，三维空间的图像单应性矩阵包含旋转平移信息，我们希望从它得到两摄像机的相对旋转平移矩阵。Faugeras 提出了一种方法[15]，对矩阵 H 做 SVD 分解。根据奇异值不同取值情况和可视区域条件可以找出最优[R|T]矩阵。

5.5 计算主平面和 PCA 法线估计

计算得到[R|T]矩阵后，即可对初始帧进行三角搜索和捆绑调整等操作，这部分内容在后面更新帧的章节中介绍。问题是，做完这些处理后，当前视图是相对第一个关键帧的。我们需要将地图旋转平移到合适的位置。好让世界坐标系原点在摄像机视野内。这一步通过 RANSAC 完成:

随机 3 个点集构成一个平面，假设它是我们想要的主平面，然后用其它剩下的点做一致性测试，测试的方法:

- 计算 3 点平面法向量，归一化三点构面的法向量。

- 计算剩余点到平面距离 **d = n p**。剔除离群点。计算剔除后的距离之和作为一次迭代得分结果。

- 反复随机选择 100 组，取得分最低的一组进行下一步。

- 使用内点作为 PCA 数据集输入。对这团点云做法线估计[16]，最终确定主平面。

使用 PCA 做法线估计时，此数据集的的协方差矩阵为:

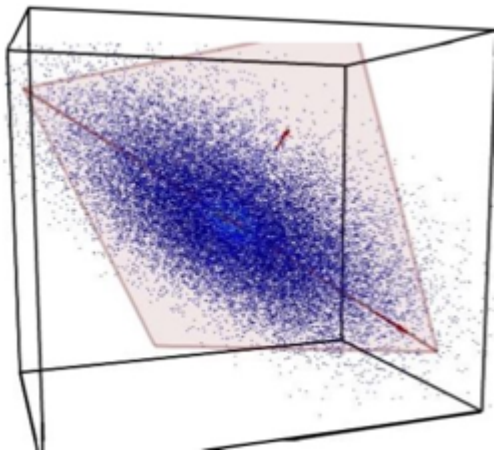
$$C = \frac{1}{k}\sum_{i=1}^{k} (p_i - \overline{p}) \cdot (p_i - \overline{p})^T \tag{40}$$

其中，p 为世界坐标系中的位置向量[x,y,z]，k 是点云的点个数，$\overline{p}$为点云坐标平均值。

协方差矩阵的每个元素代表着各个随机变量之间的相关性。每个元素可表示为 $\Sigma_{ij} = \mathrm{cov}(X_i, X_j) = \mathrm{E}[(X_i - \mu_i)(X_j - \mu_j)^{\mathrm{T}}]$，当除对角元素外的其它元素为 0 时，各个随机变量互相独立。因此协方差的特征向量可以表示数据间的相关性，对协方差矩阵特征分解得:

$$C \cdot \mathbf{v_j} = \lambda_j \cdot \mathbf{v_j}, j\epsilon 0,1,2$$

λ_j为协方差矩阵的第 j 个特征向量，v 为第 j 个特征向量。C 为实对称矩阵，故分解的特征向量两两正交。而 PCA 分解中，特征值越大，对应点集在特征向量上的正交投影与中心点差值越大。也就是最大限度地降维保留了点信息。于是最大的两个特征值对应的特征向量构成主平面。特征值最小的，点云投影的能量最小，对应的特征向量即是目标法向量 N。如图所示:



图[5-5] 三维 PCA

估计得法向量后，以它为世界坐标系的 Z 轴，把点集坐标位置和摄像机姿态变换到合适的位置。当前点的世界坐标系位置在第一帧位置。要把坐标系旋转平移到图中所示坐标系，变换方程为:

$$p'_w = Rp_w + T$$

而目标位置变换到原点则为:

$$p'_w = R^{-1}p_w - T$$

在第三章中基变换可以拓展到三维，每个旋转矩阵可以看做 3 个基向量的列线性组合。R 即是新坐标系的坐标轴向量在原坐标系中的值的线性组合。以 **N** 为 z 轴，构建三个相互正交的向量。(需要注意这三个向量是原基下的值，只有原基是单位正交基时才能直接计算新的基)。由于 R 是正交矩阵，$R^{-1} = R^T$, 直接将 3 个向量填入转置矩阵即得行向量即得 R^{-1}：

$$\begin{aligned} R^T[1] &= [1,0,0] - N \cdot ([1,0,0] \cdot N) \\ R^T[2] &= N \cdot R[1] \\ R^T[3] &= N \end{aligned} \tag{41}$$

该坐标系的位置即当前点云的位置均值:

$$T = [\overline{x}, \overline{y}, \overline{z}] \tag{42}$$

除了变换所有点，每一帧的视图矩阵也要变化，使视图区域内的点在摄像机坐标系中的位置不变，令M'_{cw}为新的视图矩阵则:

$$M_{cw}p_w = M'_{cw}p'_w \tag{43}$$

代入$p_{w'} = [R|T]p_w$得:

$$M'_{cw} = M_{cw}[R^T| - T]^{-1}$$

遍历所有关键帧（当前只有起始帧两帧），按上式更新它们对应的视图矩阵即完成地图初始化工作。

5.6 添加关键帧

最初的地图仅包含两个关键帧，并且描述的是一个较小空间。由于照相机的移动而远离其初始位置，新的关键帧和地图特征被添加到系统中，进而扩增地图。可以被添加的关键侦需要满足以下条件:跟踪质量必须好，与被添加的上一个关键侦时间间隔必须超过 20 帧;相机的位置必须以最小距离远离已加入地图的特征点。最小距离的要求避免了照相机恒定不动而破坏地图的问题，并且保证了相邻关键帧的间空间距离很接近。

同时在插入新帧时，当出现第一次观察到的点，也就是在前面的关键帧集中都无法找到对应的点，我们需要新观察到的点的信息：首先判断新观测到的点符合候选点定义，然后计算这个点在世界坐标系的位置，最后把它归类到当前关键帧。计算世界坐标系的过程叫三角搜索。在这之前需要一对点对应。找点对应的过程和跟踪线程相似。只是现在对所有的特征点做筛选，然后再做斑点搜索。

5.7 对极约束

出于性能和精度要求，先使用对极约束缩小搜索点集范围。首先需要：

1. 每帧图像的 4 个金字塔层分别保存有所有特征点位置。

2. 使用 Shi-Tomasi 得分筛选跟踪质量高的候选点。

3. 排除 10*10 像素内重复的点。

立体匹配中，为了降低可能匹配点对的数量，提出了极线约束，即左图像上的任一点，在右图像上的可能匹配点只可能位于极线上。以图 5-1 演示，对极约束的定义是：

已知视图 A 的点 m，视图 B 对应的点 m'必定坐落在对应的极线 l'上。极线约束使得立体匹配搜索范围从二维平面降到一维直线，大大减少了搜索时间，在立体匹配中占据着重要的地位。

由于测量粗差关系，实际 m'不可能准确落在 l'上。但是求出 l'对缩小搜索范围还是有意义的。首先用下标 s，t 区分源帧和目标帧。把源帧的候选点 m 经摄像机反校准(CamUnproj)反投影(Unproj)得到源帧摄像机坐标系下 z=1 平面上的值：

$$p_u = Unproj_s(CamUnproj(m)) \tag{44}$$

转为目标帧的摄像机坐标系，M 为视图矩阵，O 为原点：

$$p'_u = M_t M_s^{-1} p_u; O_s = M_t M_s^{-1}[T_s] \tag{45}$$

获得一个合适范围内的向量：

$$\mathbf{r} = (p'_u - O_s) \cdot len \tag{46}$$

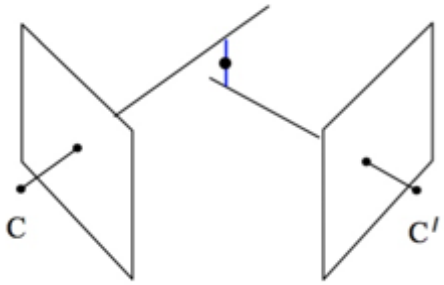
投影到目标帧的视图上，即得到极线 l：

$$\mathbf{l} = proj_t(r) \tag{47}$$

后续把在 **l** 附近的点加入搜索点集，使用斑点搜索生成 m 的搜索模板，在点集中找到 ZSSD 得分最低的点即为一对匹配点。

5.8 奇异值分解解决三角搜索问题

本节介绍用上节得到的匹配点对三维重构它们的三维位置。



图[5-6] 三角搜索

在双视图几何中，候选点符合公式$\mathbf{x} = P\mathbf{X}$和$\mathbf{x}' = P'\mathrm{X}$，P 是视图矩阵，x[x,y]为图像中的坐标的齐次形式，X 即为所求的三维坐标值。则 X 满足

$$PX - [x, y, 1]^T -= [amin, amin, 1]^T \tag{48}$$

amin 表示绝对值最小，以 P[i]表示 P 矩阵的第 i 行，写成方程组：

$$xP_{[3]}\mathbf{X} - P_{[1]}\mathbf{X} = min \tag{49}$$

$$yP_{[3]}\mathbf{X} - P_{[2]}\mathbf{X} = min$$

同理，另一个视图中的 x'也有类似的公式。把它们组合成矩阵表示：

$$||\mathbf{AX}|| = amin, A = \begin{bmatrix} xP_{[3]} - P_{[1]} \\ yP_{[3]} - P_{[2]} \\ x'P'_{[3]} - P'_{[1]} \\ y'P'_{[3]} - P'_{[2]} \end{bmatrix}$$

为了确保有非零解，加入约束条件||X||=1，还有另一个约束条件是以齐次坐标 p 表示位置，则$pX_{[4]} = X$，下标 4 表示 X 第 4 个元素 。对目标函数变形：

$$||\mathbf{AX}||^2 = X^T A^T AX = min \tag{50}$$

等式右边$\mathbf{A^T A}$是个二次型矩阵。加入约束条件，使用拉格朗日乘数法[17]令

$$L(x, \lambda) = X^T \mathbf{A^T A} X - \lambda(X^T X - 1) \tag{51}$$

取极值则有：

$$\frac{\partial L}{\partial X} = 2\mathbf{A^T A}X - 2\lambda X = 0$$

$$\frac{\partial L}{\partial \lambda} = 1 - X^T X = 0$$

知 L 取极值时，X 为$A^T A$特征值 λ 特征向量，此时$||AX||^2 = \lambda X^T X = \lambda$。对 A 做 SVD 分解，得到 V 矩阵即为$A^T A$特征向量集。令$\lambda$最小，即取 V 的最后一列作为所求的 X 向量。最后令 X 满足齐次坐标的约束条件得到三维坐标。

$$\begin{aligned} A &= UDV^T \\ X &= V_{[4]}^T \\ p &= \frac{X}{X[4]} \end{aligned} \tag{52}$$

遍历当前帧每个可用候选点，计算它们的三维坐标，即可得到一个当前帧的稀疏的点云。

5.9 捆绑调整（Bundle Adjustment）简介

捆绑调整是系统优化地图的方法。通过捆绑调整迭代来降低误差，以提高鲁棒性。这种方法不仅可以对照相机的位姿进行优化，同时还可以对场景中的所有特征点进行优化。基于全局优化的思路，PTAM 有几种捆绑调整策略：

全局捆绑调整是对所有的关键帧进行的；而局部调整仅是对于某个新关键帧，以及其周围相邻的四个关键帧用以优化调整；以及对离群点做调整或者抛弃。如果关键帧被添加到地图时，捆绑调整正在进行，那么，立即中断捆绑调整，并将新关键帧添加到地图中后继续进行捆绑调整操作。事实上构建地图线程大部分编码和耗时都在这一步骤，但对帧捆绑调整的过程和姿态估计内容类似，本节不再赘述。

6 Android 移植与拓展

6.1 Android 系统

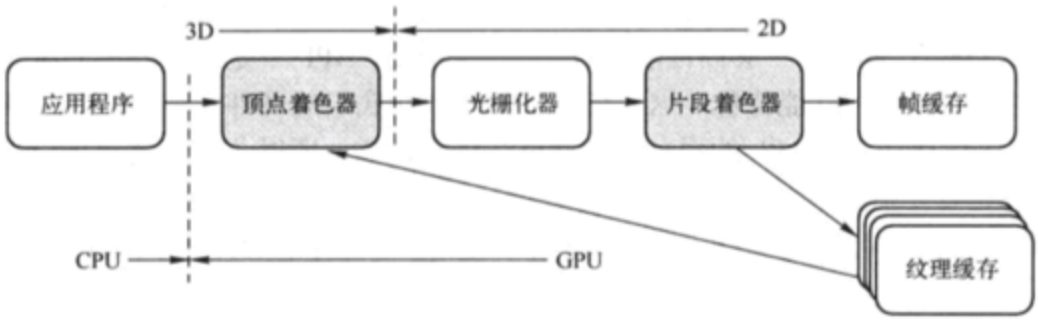
Android 是当今人人皆知的移动操作系统，而 Android NDK,即 Native Development Kit，是 android 开发原生层的工具。众所周知，Android 程序运行在 Dalvik 虚拟机中，NDK 允许用户使用类似 C/C++之类的原生代码语言执行部分程序。NDK 包括了：

从 C/C++生成原生代码库所需要的工具和 build files。使用 ndk 可以将一致的原生库嵌入在 Android 设备上部署的应用程序包文件中。而且支持所有未来 Android 平台的一系列原生系统头文件和库。本节讨论如何将已在 x86 平台上实现的 PTAM 代码移植到 Android 设备。

6.2 OpenGL 与渲染管线

OpenGL™ 是行业领域中最为广泛接纳的 2D/3D 图形 API, 其自诞生至今已催生了各种计算机平台及设备上的数千优秀应用程序。OpenGL™ 是独立于视窗操作系统或其它操作系统的，亦是网络透明的。

图形流水线是 GPU 工作的通用模型。它以某种形式表示的的三维场景为输入，输出一个二维的光栅图像到显示器。OpenGL 作为接口，进一步定义了流水线中各功能与



硬件的关系，以及实现这些功能的具体方法。下图是简化的图形流水线模型：

图[6-1]

GLSL - OpenGL Shading Language 也称作 GLslang，是一个以 C 语言为基础的高阶着色语言。它是由 OpenGL ARB 所建立，提供开发者对绘图管线更多的直接控制，而无需使用汇编语言或硬件规格语言。可编程的即是指上图中的顶点着色器单元和片段着色器单元处理可以使用 GLSL 编程描述。

使用 openGL(ES)和可编程管线可以模拟实现照相机模型，相比第二章描述的线性

照相机模型，只是少了最后的摄像机内参矩阵。PTAM 系统中，OpenGL 主要负责的工作包括使用可编程渲染管线渲染摄像头帧数据，渲染地图点，使数据可视化，基于 PTAM 构建的三维世界渲染三维模型，实现 AR 效果。

6.3 PTAM 移植

把 PTAM 移植到 android 主要有以下几个工作：

1. 移植 libcvd TTooN gvars3 lapack libpng tinyxml2 库的 ndk 版本。

2. 实现 VideoSourceForAndroid 代码。

Android JNI 层无法直接调用摄像头。因此需要在 java 层实现读取摄像头原始数据(NV21 格式)。后台新建一个 TextureSurface 伪浏览。将读取的数据传到 jni 层转为 CVD::Image 格式。之后就可以在 PTAM 中使用以及用 GLSurfaceView 渲染摄像头画面了。

3. 将使用 OpenGL 接口的代码用 openGLES2.0 接口重新封装。第一版本的 PTAM 使用的是 OpenGL 的代码，当时还在使用固定管线。而现代移动设备已经普及了 GLES2.0,所以必须编写对应的 Shader 程序，同时用新的接口编写。以及为了提高运行速度，此次移植还编写了一个直接渲染摄像头 NV21 格式的帧数据的 Shader 程序。免去了用 CPU 格式转换的时间。这部分是移植最主要的工作。

4. 触摸点击按钮等更多辅助业务逻辑代码。

5. (可选)使用手机的 IMU 数据代替旋转矩阵。

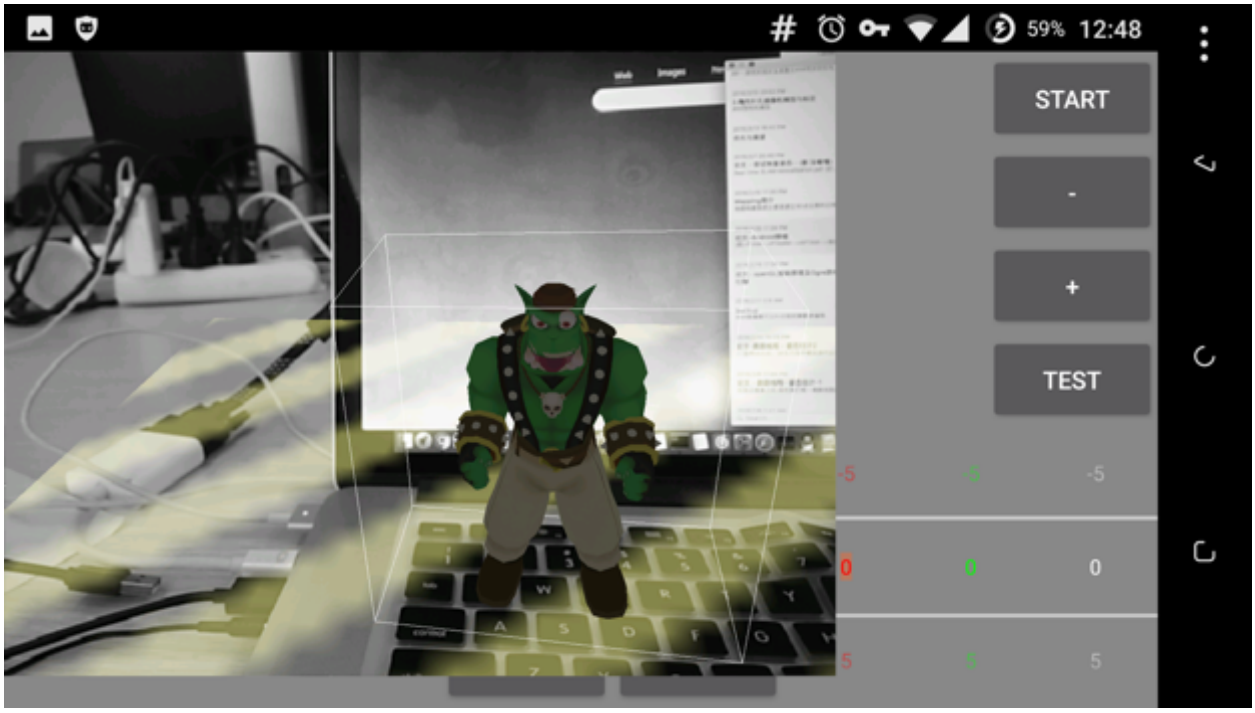
6.4 OGRE 引擎

OGRE（Object-Oriented Graphics Rendering Engine，面向对象图形渲染引擎） 又叫做 OGRE 3D。OGRE 是面向场景的、灵活的图像引擎。它是开源的，免费的，跨平台图形渲染引擎。通常在 Unix 系平台上，它所做的是对 openGL 接口更进一步地封装，使使用者能轻松地绘制 3D 人物，地形，特效等 OpenGL 需要非常繁琐的操作才能实现的事情。

ogre 是为游戏开发设计的图形引擎，为了便利开发，它实现了一套包含角色管理，场景管理的 API,以及一套完善的材质系统和材质脚本。由于篇幅限制，本章不讲解这些接口，隐藏具体编程细节。总结为了在 Ogre 中实现 AR 效果主要有以下流程：

开启渲染线程→帧回调 loop→从跟踪线程获得帧画面和当前姿态→渲染当前画面到纹理
→渲染纹理到 Ogre 的 2D 平面对象→将当前姿态转为 Ogre 图形系统的视图矩阵
→绘制 Ogre 场景，人物，帧动画等

最终我们实现的效果：



图[6-2]

6.5 辅助地图初始化

虽然地图初始化时有 RANSAC 测试和 PCA 估计主平面，已经能相对智能的找到地面。但是面对特征不明显或者纹理重复的场景依然无能为力。本次移植加入手动调整的功能，使主平面能主观地落到想要的位置。实现方法和地图初始化时一样，对所有点做调整和对所有帧的视图矩阵做调整。

7 总结与展望

移动机器人技术是一个不断发展和创新的技术，机器人的研究内容涉及到多个学科的内容。PTAM 增强现实系统的鲁棒性和准确性已经达到了相当高的水平。结合当今机器人研究的发展潮流和国内外的研究内容，作者认为 PTAM 还有以下几个方面还需要进一步研究：

1. 本文中对同时定位与制图的研究都是假设环境中的特征是静态的，小范围的，对于大规模、动态环境下的 SLAM 不适用。

2. 跟踪系统的重定位功能太慢。而且是基于 3 个旋转自由度内寻找查找，不利于大范围移动的重定位。

3. PTAM 移植到手机后可以利用高端手机的 IMU 数据更好的跟踪。例如姿态更新部分，直接使用 IMU 产生的旋转矩阵，只对位移偏量进行估计。

4. 同样的，。如果 IMU 数据足够准确。当旋转量过大时可以直接判断为跟踪丢失。直接开始重定位。

5. 基于视觉的跟踪系统在应用上有不可避免的限制。帧图像中，单调重复的纹理容易使得跟踪丢失。

PTAM 在跟踪质量和地图重建的算法优化上已经出现不少方法，如 ORBSlam 就是以 PTAM 为基础的 SLAM 工程，实现了大范围的跟踪与重建。本文实际工作没有优化方法，而是提出一些 PTAM 在 AR 方面的拓展与改良。例如把 PTAM 与 Ogre 游戏引擎结合起来，实现在移动平台上体验 AR 游戏，这是本文实际编程工作中的重点。实际上，将来加入物理引擎后，结合 Ogre 引擎后能完全基于此系统做 AR 游戏开发了。甚至近年来，有开源组织把 Ogre 和 Bullet 等项目结合在一个 gamekit 工具集中，开发者能使用此工具直接加载 blender 文件。仿照 Unity 的开发模式，实现 PC 上写游戏脚本，跨平台运行，开发起 AR 应用来将会非常方便，而且免费。

SLAM 方面，近年来 SLAM 发展迅速，衍生出各种版本的地图数据。如 PTAM 稀疏点集，DTAM 稠密的点集，基于稠密点集主成分分析构建的多平面。这些地图也能加入到 AR 系统中做诸如碰撞反馈之类的应用。具有良好的可扩展性。

参考文献

[1] 吴福朝. 计算机视觉中的数学方法[M]. 科学出版社, 2008.

[2] Klein, G., & Murray, D. (2007, November). Parallel tracking and mapping for small AR workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on* (pp. 225-234). IEEE.

[3] Klein G, Murray D. Parallel tracking and mapping on a camera phone[C]//Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on. IEEE, 2009: 83-86.

[4] G. Bleser, H. Wuest, and D. Stricker. Online camera pose estimation in partially known and dynamic scenes. In Proc. 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'06), San Diego, CA, October 2006.

[5] Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. In Proc. 11th IEEEInternational Conference on Computer Vision (ICCV'07), Rio de Janeiro, October 2007.

[6] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, second edition, 2004.

[7] C. Engels, H. Stewenius, and D. Nistér. Bundle adjustment rules. In Photogrammetric Computer Vision (PCV'06), 2006.

[8] P. Huber. Robust Statistics. Wiley, 1981.

[9] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In Proc. 9th European Conference on Computer Vision (ECCV'06), Graz, May 2006.

[10] J. Shi and C. Tomasi. Good features to track. In Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR '94), pages 593-600. IEEE Computer Society, 1994.

[11] Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. International Journal of Pattern Recognition and Artificial Intelligence, 2(3):485-508, 1988.

[12] 赵毅力. 使用 Math.NET 求解线性和非线性最小二乘问题[EB/OL].http://cs2.swfc.edu.cn/~zyl/?p=875#nonlinear-gn

[13] 实用测量数据处理方法[M]. 测绘出版社, 2000.

[14] Stewenius H, Engels C, Nistér D. Recent developments on direct relative orientation[J]. ISPRS Journal of Photogrammetry and Remote Sensing, 2006, 60(4): 284-294.

[15] Faugeras O D, Lustman F. Motion and structure from motion in a piecewise planar environment[J]. International Journal of Pattern Recognition and Artificial Intelligence, 1988, 2(03): 485-508.

[16] Shlens J. A tutorial on principal component analysis[J]. arXiv preprint arXiv:1404.1100, 2014.

[17] 张禾瑞, 郝钢新. 高等代数 第五版[M]. 北京: 高等教育出版社, 2007.

[18] Zhao H. Inverse Compositional Method[J].

致　谢

本次的毕业设计论文是在唐佳林老师的帮助和指导下完成的，唐老师作为一位在教学岗位上兢兢业业、在教学方式上富有耐心、在科研领域上严谨求实的教师，在我的整个毕业设计的实验和论文的写作过程中，从未停止过对我的教导。在整个毕业设计期间，引导我不断的发现问题，为我解答疑惑，鼓励我大胆创新，使我在这一段宝贵的大学时光中，既开阔了视野，增长了知识，又培养了良好的实验习惯和务实的科研精神。

同时还感谢我在观梦科技有限公司和进化动力数码有限公司实习时遇到的同事。他们在计算机视觉方面的专业素质给我提供了很大的帮助。他们是王婷如、刘钊、倪攀。在此，我向他们表示最由衷的感谢。

如今我在北京理工大学珠海学院的大学生生活即将结束，回顾这四年来的学习经历，期间有过失落，更也有过收获。面对这些失落我学会了放正心态，面对这些收获我学会了放宽心态。这些都是学校教给我的一生的财富。为此，我向曾热心帮助过我的所有老师和同学表示最诚挚的感谢。

在这次的毕业设计过程中，我遇到了很多在平时专业课学习上从未碰到过的难题，得到了老师•，使我的毕业论文如今能得以完成，再次向所有帮助过我的老师和同学表示诚挚的感谢，感谢你们对我的热情帮助！

最后，对在百忙之中参加答辩并给与建议的各位老师、教授表示最诚挚的谢意！

由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师予以批评和指正！