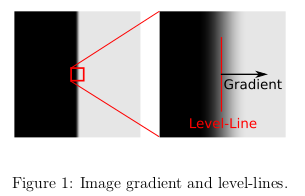
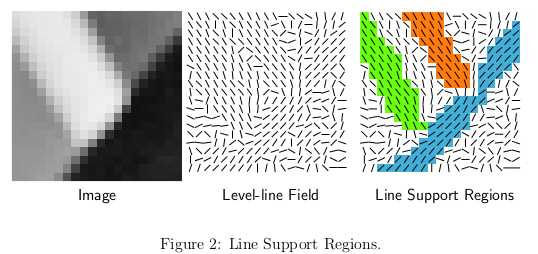
## LSD原理解析：

**直线段检测算法---LSD：a Line Segment Detector**

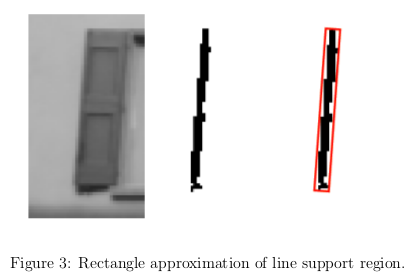
LSD的核心是像素合并于误差控制。利用合并像素来检测直线段并不是什么新鲜的方法，但是合并像素的方法通常运算量较大。LSD号称是能在线性时间（linear-time）内得到亚像素级准确度的直线段检测算法。LSD虽然号称不需人工设置任何参数，但是实际使用时，可以设置采样率和判断俩像素是否合并的方向差。我们知道，检测图像中的直线其实就是寻找图像中梯度变化较大的像素。LSD的目标在于检测图像中局部的直的轮廓，这也是我们称之为直线分割的原因。轮廓是图像中的某些特殊区域，在这些区域，图像的灰度从黑到白或者从白到黑的剧烈变化。因此，梯度和level-line是两个重要的概念，如下图所示：



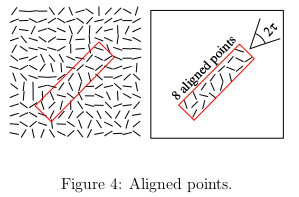
算法首先计算每个像素的level-line angle（此文章下面的[2]）以构成一个level-line 场。该场被分割为连通的若干个部分，它们方向近似相同并且在容忍度τ内，这样可以得到一系列regions，这些 regions被称为 line support regions（支持域）。如下图所示：



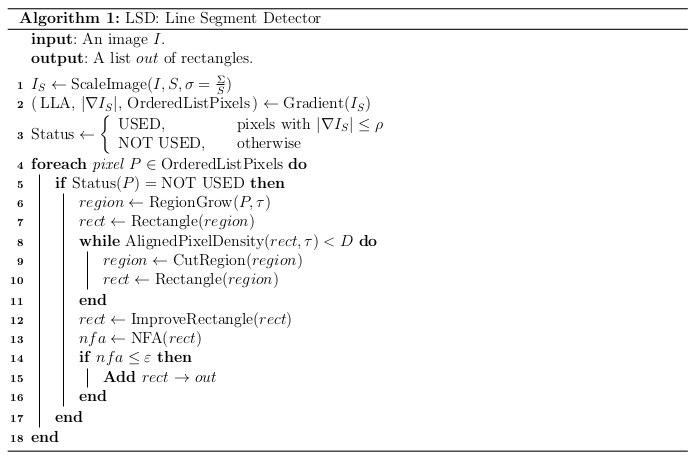
每一个line support region其实就是一组像素，它也是直线段（line segment）的候选。同时，对于这个line support region，我们可以观察它的最小外接矩形。直观上来讲，当一组像素构成的区域，特别细长时，那么这组像素更加可能是直线段。line support region的一个主惯性轴作为矩形的主方向，矩形的大小选择为覆盖整个区域。



矩形中的像素的level-line angle与最小外接矩形的主方向的角度差在容忍（tolerance）τ内的话，那么这个点被称作"aligned point"（同性点）。通过统计最小外接矩形内的所有像素数n和其内的alinedg points个数k，用来判定这个line support region是否是一个直线段。判定的准则使用的是a contrario approach and the Helmholtz principle。



LSD算法的具体解释：



输入：灰度图

输出：一系列的直线分割结果。

1.以 s=0.8的尺度对输入图像进行高斯核采样。

2.计算每一个点的梯度值以及梯度方向（level-line orientation），其中gx和gy分别为水平和垂直方向梯度。

3.根据梯度值对所有点进行伪排序（pseudo-ordered），建立状态列表，所有点设置为UNUSED。

 [设置图像梯度强度范围到[0, 1023](大于1023的梯度强度，强制设置为1023),创建1024个链表，遍历整个梯度图，根据梯度强度，相同梯度值像素坐标放入同一张链表中,将1024个链表，按从大到小顺序，合成一张大链表(首部为1023链表，尾部为0)]

4.将梯度值小于ρ的点状态表中相应位置设置为USED。

5.取出链表头部存储图像坐标位置，作为种子像素，

do：

    a.以seed为起点，根据梯度角方向相似(搜索周围UNUSED并且方向在阈值[ -τ, τ]范围内的点)，进行区域扩散。(每扩散一个像素，将该像素坐标从链表中删除，并且做标记(状态改为USED)，之后新的区域扩散，无法再扩散到在像素)。

    b.将扩散区域进行矩形拟合,R。

    c.判断同性点（aligned pt）密度是否满足阈值D，若不满足，截断（cut）R变为多个矩形框，直至满足。

    d.计算NFA(拟合矩形精度误差)。

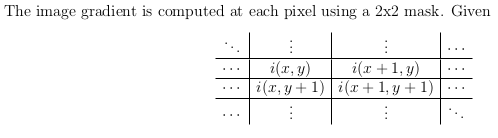
    e.改变R使NFA的值更小直至NFA <= ε ，R加入输出列表;如果改变了还不满足或者矩形区域太小，舍去。

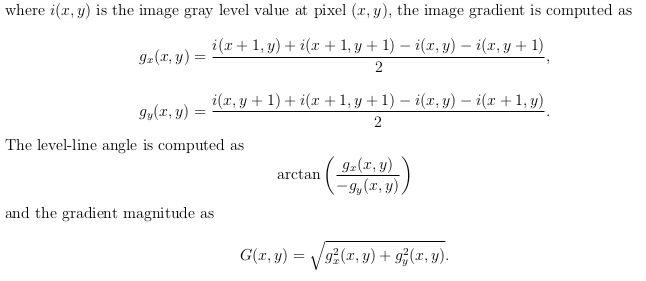
6.继续回到第5步，从链表中，找到下一个种子点，从剩下图像进行区域扩散，至到遍历完全图，得到所有检测到的直线。

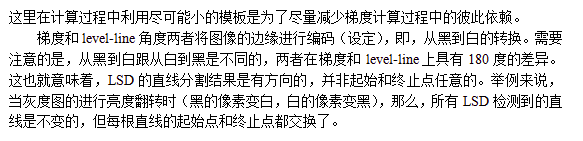
【1】将输入的图像缩小为原来大小的80%，即缩放尺度为80%（S=0.8），目的在于减弱甚至消除很多图像中出现的锯齿效应。注意：80%的缩放因子意味着x和y方向都缩放为原来的80%，即总的像素为原来的64%。

    缩小是通过高斯采样来进行的：图像首先采用高斯核进行滤波从而避免锯齿效应，再进行降采样。高斯核的标准差是由 Σ / S来决定的，此处的S是缩放因子，参数Σ设置为0.6，以此来保持锯齿效应和图像模糊之间的平衡。

【2】梯度计算





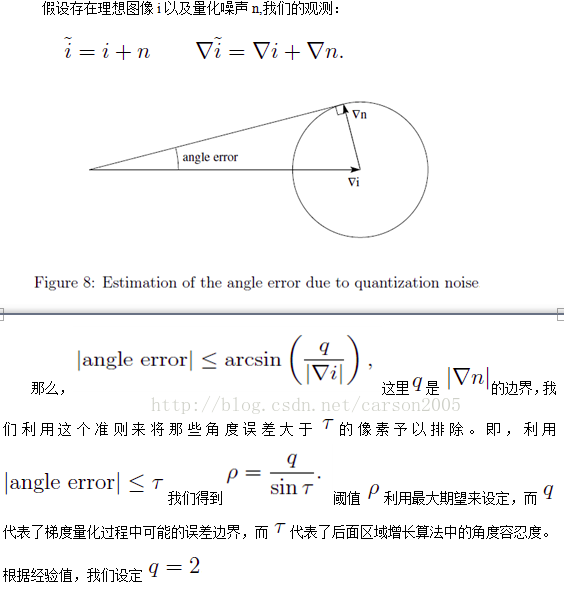


【3】梯度排序

梯度值越大，越是显著的边缘点，更适合作为种子点。但是对梯度值进行完全排序是一个时耗性很高的工作。因此简单的将梯度值划分为1024个等级（bins），这1024个等级涵盖了梯度由0~255的变化范围，这种排序是一个线性的时耗。LSD首先将最大梯度的像素作为种子点，种子点从梯度值最高的bin开始搜索，依次往下，直至所有点标记为UNUSED。

【4】梯度阈值**（小梯度值抑制）**

小梯度值点往往出现在平滑区域，或者仅仅是噪声。不在关注的范围内，但是他们的存在往往会严重影响直线角度的计算。在LSD计算过程中，梯度幅值小于ρ的像素点将被拒绝参与line support region或者矩形的构建过程。

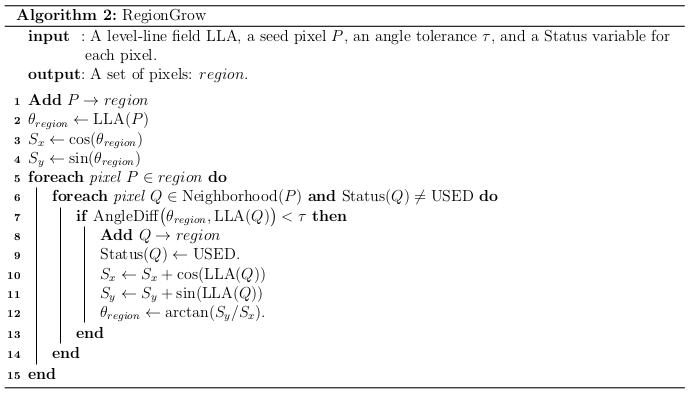


【5】区域增长

line support region是通过合并方向近似相同的像素得到。其实在这里，这个合并的过程更多的是依赖于区域生长算法。

输入：level-line 场LLA，种子像素P，和角度容忍度 τ以及对应每个像素的状态变量

输出：一组像素区域

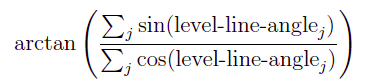


区域生长算法用于生成一个line support region。最初的区域角θregion是种子点（P）的level-line angle，递归的，将已经在区域的像素（P）的未使用的邻居用来测试，

28-344633087.png

and the ones whose level-line angle is equal to the region angle θregion up to a tolerance τ are added to the region.

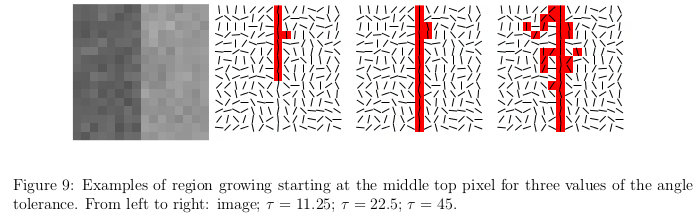
每次添加一个新的像素到该区域，区域的角度就被更新为：



下标j：用于遍历区域中的所有像素，如此持续进行，直到没有任何像素添加到矩形当中。

 角度容忍度 τ设置为22.5度,或者说π/8弧度，so it was set to obtain p = 1/8。在该误差容忍度范围内的像素点都将被选择到矩形中，这是因为他们在很大程度上跟矩形的方向保持一致。

如下图所示，实验发现，22.5是一个较好的参数。



【6】矩形估计

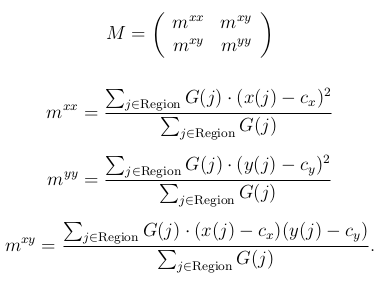
上一步形成line-support region一系列相邻的离散点，需要将他们包含在一个矩形框内R，这个可以看做宽度为R的宽，长度为R的长的候选直线。R选择能包含line-support region的最小矩形，所有点的梯度规范化值平均计算重心，R长轴的方向设置为R的方向。

矩形的中心(c x , c y )：



G（j）是像素j的梯度幅值，下标j用于遍历矩形区域内的所有像素。

矩形的主方向被设置为矩阵的最小特征值对应的特征向量的角度：



【7】NFA计算

**自己的理解：**其实这个NFA，就是通过你的观察图中矩形中的aligned points所占比例的大小与引入的一个新的模型 （acontrario model）的aligned points作比较的概率，来约束这个矩形是不是可以作为一条“线段”。（就如同作者所希望的，这个aligned points越多，越有可能是一条“线段”）

**矩形验证：**

用来验证矩形是否可以作为检测线段的方法是基于a contrario approach and the Helmholtz principle .

the Helmholtz principle：在完美噪声图像图像中不应该检测到目标。

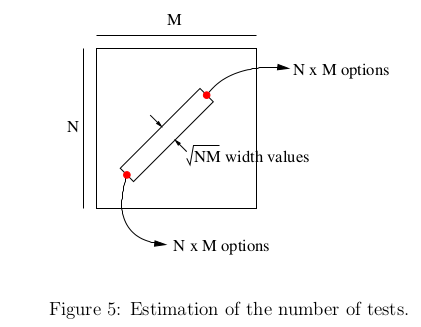
a  contrario approach：一个不会检测到目标的噪声图像。

对于本课题，contrario model  H0是一个像素值为level-line angle的图像，其level-line angle随机分解为独立且服从平均分布于[0,2π]。

（即主要有以下两个属性： （1）{LLA(j)}，是 level-line场，其中j是像素，由独立随机变量组成；（2）LLA(j)在[0,2π]上均匀分布。 ）

这里用NFA（Number of False Alarms）来评判observe img中某个候选R小于contrario model中相同位置R里同性点（aligned  points）的数量的概率，如果NFA的值很大，认为在观测图像中aligned points比contrario model中aligned points小的概率很大，将其认为是common，平常的，背景中的一部分。如果NFA的值很小，认为目标是相对突出（rare）的，是一个合适的“直线”。

35-904366074.png



其中，Ntest为当前大小（n\*m）图像中直线（矩形框）的数量。在n\*m的图像中直线的起点和终点分别有m\*n种选择，所以一共有(n\*m）\*(n\*m)种起点和终点搭配。线段的宽度为(n\*m)^0.5，因此在m\*n大小的图像中有（n\*m）^2.5 种不同直线。The precision p is initially set to the value τ /π.We will note γ the number different p values potentially tried. Each rectangle with each p value is a different test.Thus, the final number of tests is

7-1402636477.png

PH0是对应 contrario model H0 的一个概率，I是在模型H0下的随机图像。（H0是图像梯度方向的噪声模型而不是一个图像的噪声模型。）

k(r,I) 为contrario model ，I 中 r 矩形里aligned pt的数量。

k(r,i) 为observe img，i 中 r 矩形里aligned pt的数量。

一个关键概念是p-aligned points，是指矩形中的level-line angle与最小外接矩形的主方向的角度差在容忍（tolerance）pπ内的像素点。 precision p最初设置的值为τ /π（角度正负容忍误差为τ ，总容忍误差为2τ 。那么在contrario model中，某个点为aligned point的概率为 p=2\*τ / 2\*π =τ / π），

k(r, I) 服从二项分布，所以：

922-10262499.png

所以，NFA的数量与矩形r有关：

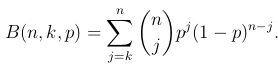
85-740528326.png

简化r和i的公式：

0-1382740193.png

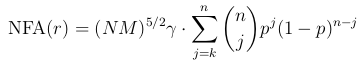
其中，N和M是采样过后图像的列和行，n：矩形的像素的总数，k：p-aligned点数。

二项分布：

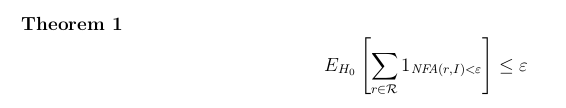


**解释：**设k(r,i)=k。那么，在 I 中的 r 矩形里，总像素个数为 n，I 中的 r 矩形里aligned pt个数k(r,I)大于等于k的话，可选择的值为k(r,i)、k(r,i)+1、k(r,i)+2，......n。

NFA计算：



**若NFA(r) ≤ ε，那么可以认为结果有效。**As stated before,we set **ε = 1** .



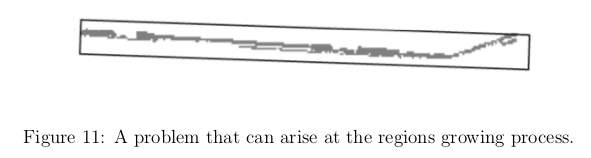
其中，E是期望算子，1是指示符函数，r是考虑的矩形集合，而I是H0上的随机图像。

 该定理指出在 contrario model H0上的ε-meaningful 矩形的平均数量小于ε。因此，噪声检测的次数通过ε控制的并且可以根据需要往小的调整。

换句话说，这表明LSD满足亥姆霍兹原理（具体证明可以看原文）。

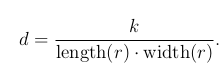
【8】Aligned Points Density

在某些情况下，这τ角度容忍度的方法产生一个错误的解释。这个问题可能会出现两直线边缘存在于图像形成一个夹角小于容忍度τ。如下图显示了一个线支持区域（灰色）和与其对应的矩形的例子：



在LSD中，这个问题是通过检测有问题的线支持区域并将其切割成两个较小的区域来处理的，希望在适当的位置将该区域分割出来以解决问题。这个“角度问题”的检测是基于矩形中同性点的密度。

当这个问题不存在时，矩形非常适合于线支持区域，并且同性点的密度很高。另一方面，当“角问题”出现时，正如前面的图所看到的，同性点的密度很低。同样，当一个稍微弯曲的边被一系列直边近似地逼近时，近似的程度（多少线段被用来覆盖曲线的一部分）与同性点的密度有关；因此，对齐点密度也与线段近似曲线的精度有关。

同性点密度的计算公式：

规定一个阈值D ，在这里设置D=0.7，文章认为这个数字既能保证同一个R中的同性点属性相近，也能保证R不会被过分的分割为小的矩形。

若d > D（同性点密度阈值），accepted。否则，需要将R截断。

R截断的方法有两种：“缩小角度误差阈值”与“缩小区域半径”的方法。在这两种方法中，区域中的一部分像素被保留，而另一些则重新标记为NOT USED，因此它们可以在将来的线支持区域中再次使用。

缩小角度容忍阈值：简单的将τ值缩小，再次从当前R的seed开始搜寻，看是否满足要求。

缩小区域半径：逐渐去除远离种子的像素，直到满足标准或区域太小而被拒绝为止。当线支持区域对应于一条曲线时，该方法效果最好，并且在满足密度准则之前需要减少区域，通常意味着对曲线有一定程度的近似。

【9】矩形的改进

如果当前的R仍旧不能满足NFA(要求：**NFA(r) ≤ ε**)，以下的方法将对其进行改进。考虑到在有些情况下，删除line-support region中的一个点会减少R的 length-1个点（想象为对角线）。对不满足NFA的R，采取以下策略：

1.减小p=p/2

2.短边减少一行

3.长边减少一行

4.长边减少另一行

5.减小p=p/2

直至满足NFA（**NFA(r) ≤ ε**）或 R过小被拒 或 p为原来的1/32。

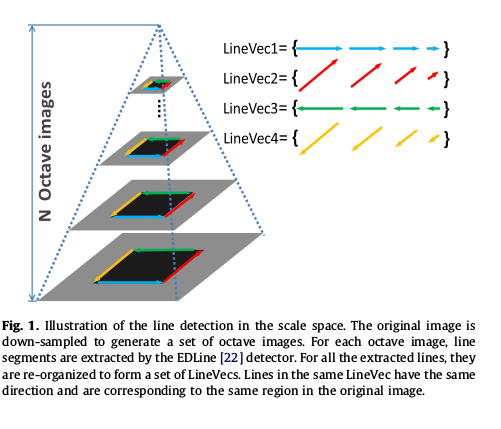
## LBD原理解析：

参考文章：An efficient and robust line segment matching approach based on LBD descriptor and pairwise geometric consistency ----Lilian Zhang  , Reinhard Koch

***第二部分：Line detection and description***

1.尺度空间中提取线段

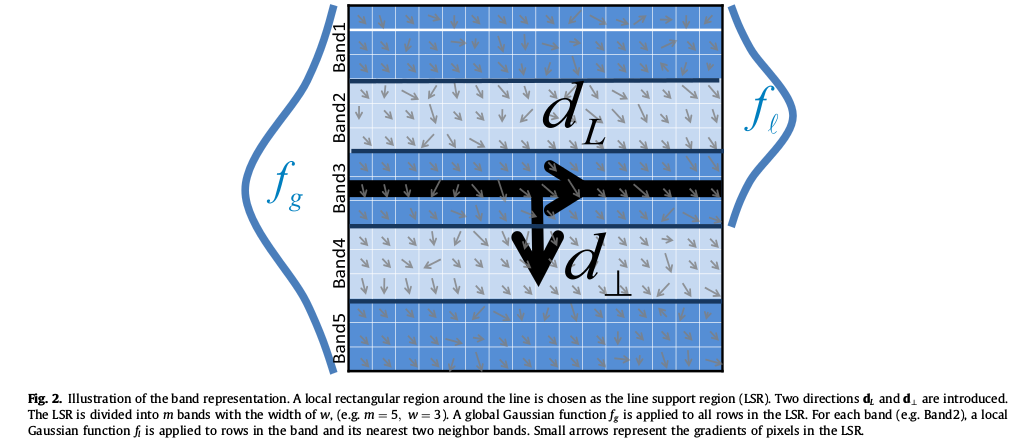
为了克服在线检测碎片问题和提高大尺度变化的性能，我们的检测框架采用由对原始图像进行高斯下采样（ with a set of 尺度因子and 高斯模糊 ）的尺度空间金字塔组成（N层图像）。我们首先用**Edline算法**每层的尺度空间产生一组线。每条直线都有一个方向，它是由从直线左侧到右侧的大部分边缘像素点的梯度构成的。**然后通过在尺度空间中找到对应的直线来组织它们。**在尺度空间中所有的线，他们被分配一个唯一的ID，如果在图像中他们相关相同的事件，将其存储在一个称为LineVecs的向量中（即图像的同一区域具有相同的方向）。线检测方法 不同于Wang et al.[5] ,通过重新组织从尺度空间提取的所有的线段形成LineVecs，从而降低了图匹配问题的维数。最终提取的结果是一组LineVecs如下图：



如图1所示，每个Linevec可能包括尺度空间的多条线。为描述一个LineVec局部外观，对于其中的每条线，我们会在被提取的线段的每层图中产生一种线描述符。

2.条带（Band）来表示线的支持域

 在octave image给出了线段，描述符将从线段支持域（LSR）计算。该支持区域被划分为一组条带{B1；B2；...；Bm}，每个条带都是LSR的子区域并且他们之间的平行的，条代数m的和每个条带的宽度w，条带的长度等于线段的长度，如下图当m =5 ，w=3的LSR例子。



类似于 MSLD[11],引入两个方向构成a local 2D coordinate frame（局部二维坐标系），用于区分具有相反梯度方向的平行线并使描述符旋转不变。根据线的方向为5-2068899866.png ，正交方向（垂直方向） 7-1907679277.png被定义为5-2068899866.png顺时针垂直的方向。该线的中点被选为这个局部坐标系的原点。LSR中的每个像素的梯度投影到这个局部框架8-1695992466.png，其中的6-1873297768.png：image frame中的像素梯度， 88-784999590.png：local frame中的像素梯度。

对于SIFT[23] and MSLD，两个高斯函数应用在每一行沿着7-1907679277.png的方向。首先，全局权重系数0610-5960595.png是在LSR的第i行，其中2-1255934013.png是第i行到LSR中心行的距离，00-596931028.png

第二，考虑一个条带Bj，以及他在相邻的条带Bj-1，Bj+1 ,局部权重系数

75-447249163.png，在Bj第k行，其中8-1187390907.png第k行到Bj中心行的距离，5-1096567077.png

全球的高斯窗的目的是给予在LSR中沿7-1907679277.png方向远离线的微小变化梯度很小的关注度（敏感性）。

局部高斯窗的目的是减少边界效应，它避免了像素从一个条带移动到下一个的描述符突然改变。

此方法描述子区域的优点：首先，它对5-2068899866.png方向的小的位置变化更有鲁棒性，因为在这种情况下，带内图像的大部分内容保持不变，只有带边界略有变化。注意，这个特性很重要，因为一般来说，由于线路端点不稳定，线的位置精度在方向5-2068899866.png中要比方向7-1907679277.png低。

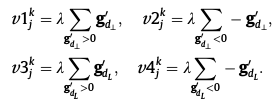
第二，它的计算效率更高，因为每个条带之间在5-2068899866.png方向没有重叠，高斯权重直接应用于每一行，而不是每个像素。

 3.构造条带描述符

（the Line Band Descriptor） LBD:00-393997882.png

每条条带的描述子BDj：通过其最近的两相邻行的条带Bj-1 ; Bj+1来计算 。**特别是，对于在顶部和底部的条带带B1和Bm，在计算B1和Bm的描述子时行在LSR之外不会被考虑在内。（所以下面n的取值会把j=1||m单独出来）**

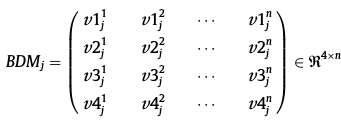
BDj的计算（其实就是描述的上下左右四个方向）：



k：表示在条带Bj的第k行，我们累积的像素的梯度在这行；

2-2125257240.png：高斯权重。

条带描述矩阵the band description matrix (BDM)：



7-2119028302.png

BDj由BDMj矩阵的均值向量Mj and标准方差 Sj 得到：

2-1343838481.png

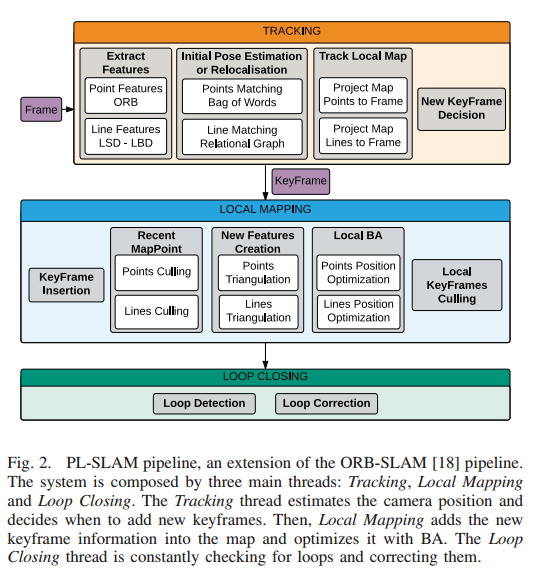
LBD的均值部分和标准方差部分由于其大小不同，分别进行规范化处理。此外，为减少非线性光照变化的影响，对LBD每个维度的进行抑制，使它小于一个阈值（经验：0.4的是一个很好的值）。最后，我们重新规范约束向量得到单元的LBD。

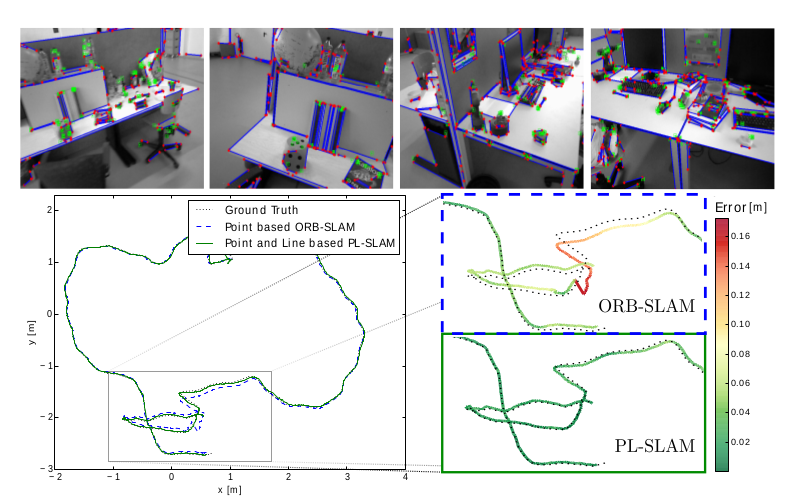
## PL-SLAM ：

将ORBSLAM与线特征结合，允许point和line同时跟踪；

提出一个0新的初始化方法：从3个连续的图像的line对应关系来估计近似初始映射，这是基于三个连续帧之间的恒定旋转假设，并且这些旋转相对

较小。（一般通过单应矩阵或基础矩阵估计）；

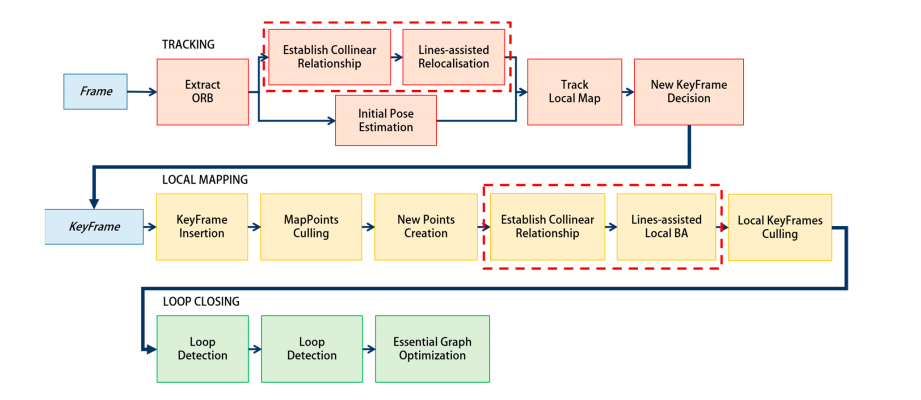




## LAP-SLAM：

<https://doi.org/10.3390/electronics8020243>

基于线辅助点的单目slam，在orbslam基础上，修改了匹配原则，加快匹配速度



**三个改进点：使用点共线和线匹配的方法；提出一个line辅助束的调整方法；改进PnP**

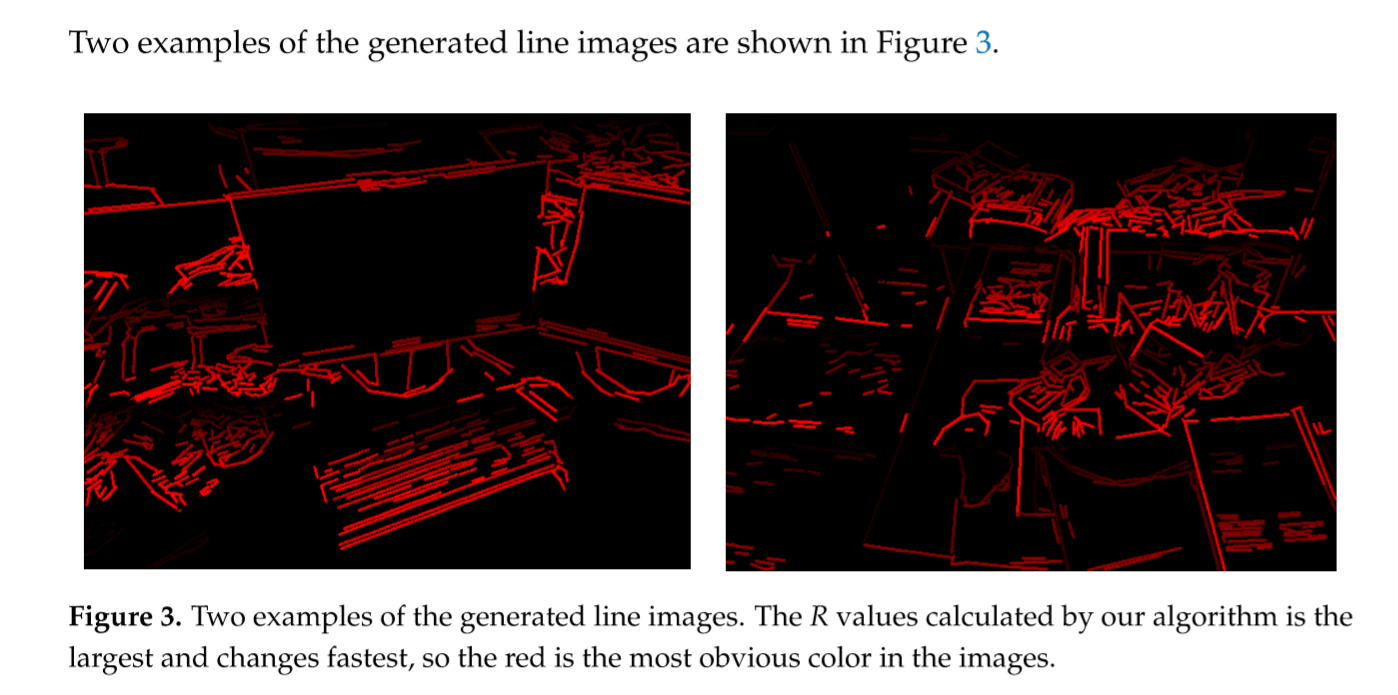
**一个实用的直线匹配算法和点的共线关系获取算法，一个直线辅助束平差方法，一个考虑直线和点的共线关系的PNP方法。**

由于人造环境中充满了丰富的边缘和线性的形状，线的面积远大于点，在计算时的负担更重。通过线上的匹配点来匹配线段。

直接法优势：弥补基于特征的方法在纹理较差的环境中可能会失去跟踪，直接法通过捕获图像之间可能的姿态变化空间 优化 顺序图像配准的光度误差。

直接法劣势：对图像噪声、几何失真、大位移、光照变化等多个因素更为敏感，因此，基于特征的方法更适合作为用于鲁棒和精确的姿态跟踪的SLAM解决方案。

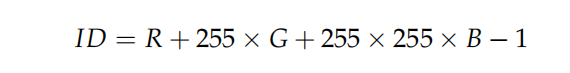
与基于点的VSLAM相比，大多数线辅助VSLAM的附加部分是线的检测、匹配和三角剖分。High computation和the potential shift along the direction of lines是目前面临的两个主要问题。



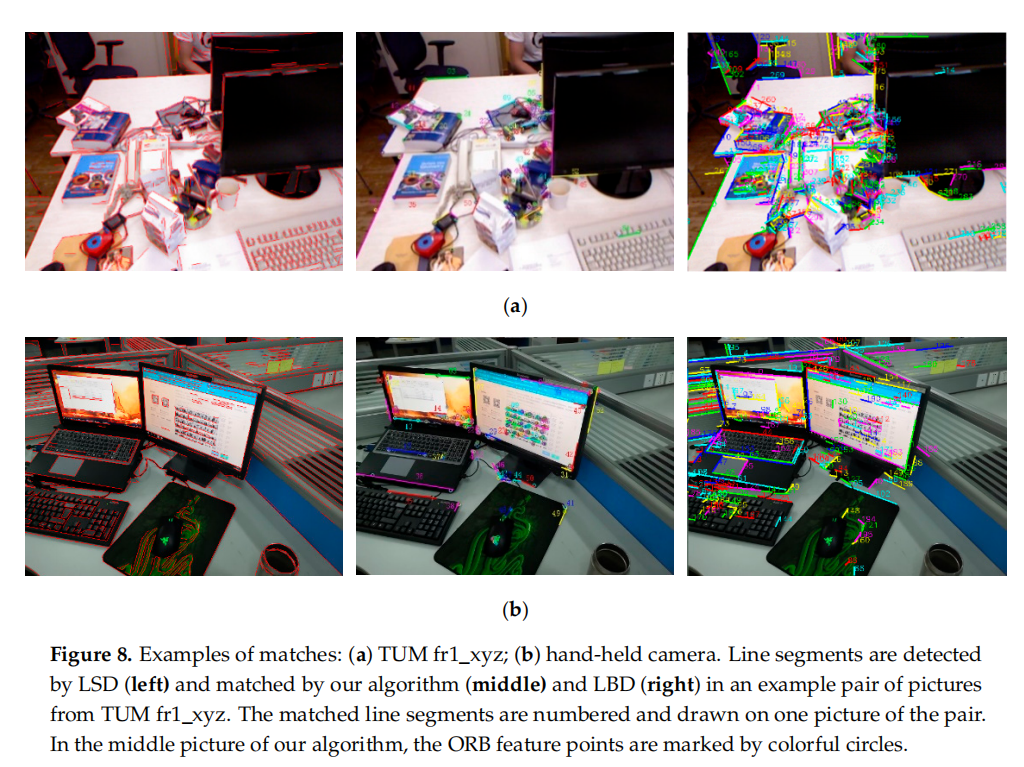
Tracking thread 计算相机的姿态并创建新的关键帧。等速运动模型用于猜测当前相机的姿势，并初步将当前帧与前一帧匹配。如果跟踪丢失，将使用位置识别模块重新定位摄像机。在ORB-SLAM中，这通常是通过EPNP[23]算法实现的，这是PNP问题的非迭代解。在LAP-SLAM中，我们在检测到的线与其对应点之间添加了重投影误差。局部映射线程主要用于局部束调整。通过关键帧插入、地图点剔除、新点创建等操作，利用LSD检测关键帧中的线段。然后，我们搜索直线段上的匹配点。例如，如果在一个关键帧的一条直线段上检测到两个或多个匹配点，并且在另一个关键帧的另一条直线上检测到相同的点，则这两条直线段是匹配的，并且这些直线段上的所有点都属于一条三维直线。可以估计三维线的初始坐标，然后在三维线上调整属于该三维线的点的位置。然后利用线辅助BA对摄像机的姿态、点线坐标进行优化，其代价函数由线点的重投影误差、点与线之间的距离组成。回环检测线程在本地映射线程之后处理关键帧，并尝试仅使用点功能检查循环。

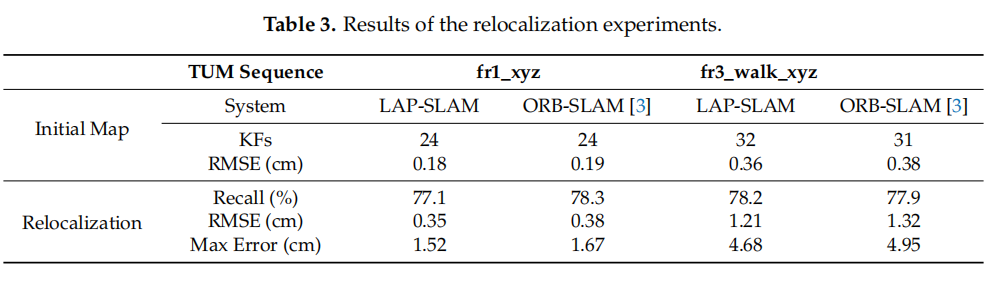
共线关系(collinear relationship)用于局部BA和线辅助全局重定位。共线关系不是实时建立的，而是在需要的时候。

投影后，如果线图像上的投影点通过上述测试，则系统读取投影点的BGR值。如果所有值都为零，则表示该点不会投影到直线上。如果没有，则该点将投影到直线段上。



如果要匹配线段，则它们的共享点应大于两个，以防唯一共享点位于两条交叉线的交叉处。



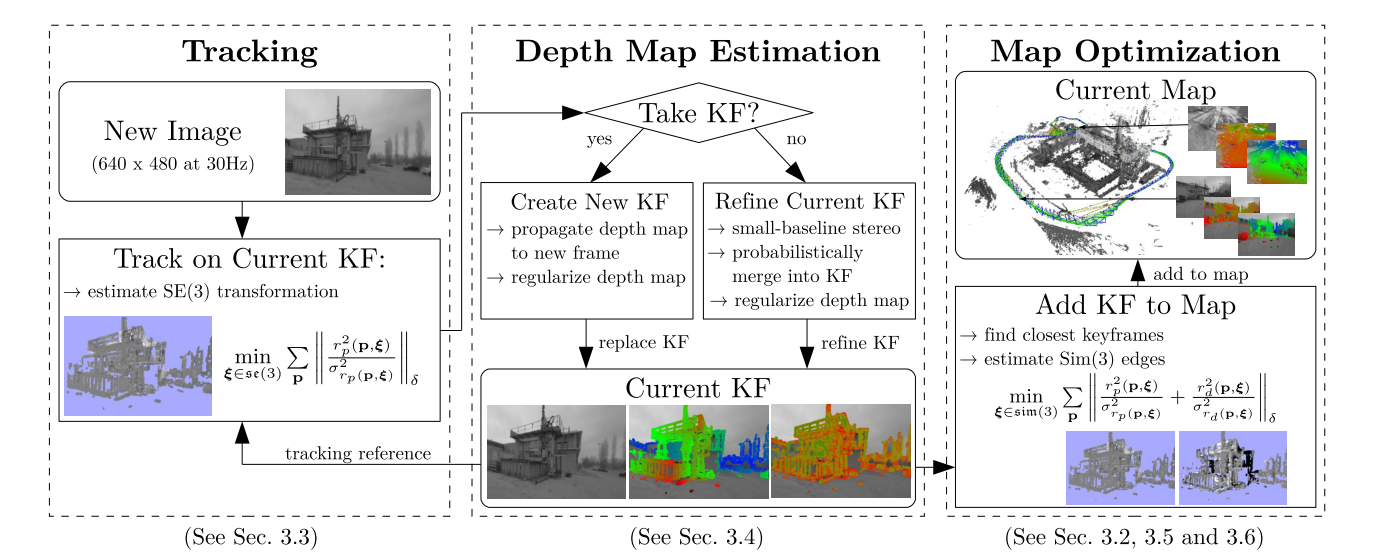


## LLD（可学习型描述符）

可学习型线描述符的设计是使用公开数据集利用最小线参数化来获取线数据集并在深而轻量级的完全卷积神经网络之上训练，将结果应用于ORB-SLAM2中来提高精度。

## LSD-SLAM（已在linux上配置完成）：

LSD-SLAM可以分为三大模块：Tracking、Depth Map Estimation和Map Optimization，如下图所示。三个部分分别位于论文[2]中3.3，3.4和第3章其余部分。但Depth Map Estimation部分详细的算法是在[1]中介绍的。Map Optimization部分是整个算法的核心。下面我们来分别看一下这三大模块。



**1 Tracking**

直接法即最小化光度误差，论文进行了一些小的改进，如使用的是方差归一化后的光度误差（variance-normalized photometric error）：

Ep(ξji)=∑p∈ΩDi∥∥∥r2p(p,ξji)σ2rp(p,ξji)∥∥∥δ(1)

rp(p,ξji):=Ii(p)−Ij(ω(p,Di(p),ξji))(2)

σ2rp(p,ξji):=2σ2I+(∂rp(p,ξji)∂Di(p))2Vi(p)(3)

可以看出改进的地方就是在分母部分加了个方差（方差的推导可以参考[3]），然后还使用了Huber核函数，总之目的就是为了更好更快地解求位姿。需要注意的是Tracking部分求的是se(3)上的位姿。

**2 Depth Map Estimation**

这部分主要是用Tracking跟踪后的帧更新或构建深度图，分两种情况：

* 构建关键帧时，则构建新的深度图（Depth Map Creation）
* 不构建关键帧时，则更新当前关键帧的深度图（Depth Map Refinement）

其中，是否构建关键帧是通过计算当前帧与当前关键帧的距离判断的，大于一定距离时才构建新的关键帧。距离定义如下，W为对角阵，目的是加权：

dist(ξji):=ξTjiWξji(4)

在进一步介绍之前，有必要重申一下一些定义：**关键帧（key frame）**是相隔一定距离的具有代表性的帧，每一个关键帧对应一个深度图；**参考帧（reference frame）**是用于和当前帧做立体匹配的帧，一个关键帧后面一般有许多参考帧。接下来就分别介绍一下上述的两种情况。

**2.1 深度图的更新**

当我们获得一个新的帧，并且判断不构建新的关键帧，则进行当前关键帧深度图的更新过程。这部分论文中称为基于立体匹配的深度更新（Stereo-Based Depth Map Update），具体可分为以下四个步骤：

* 根据一些标准选择出“好的”像素
* 为每个像素自适应选择最合适的参考帧
* 极线上进行立体匹配
* 深度观测融合

原论文打乱了顺序，这里我们按上述顺序依次介绍。

**2.1.1 选择出“好的”像素**

这部分论文中称为不确定性估计（Uncertainty Estimation），主要通过建模（对逆深度的不确定度进行估计），并以此为依据挑选出好的像素。我们知道在直接法中，逆深度与四个变量有关，可以表示成它们的的函数：

d∗=d(I0,I1,ξ,π)(5)

其中I0，I1分别为参考帧和当前帧两个图像，ξ表示两帧间位姿变换，π为相机内参。为进一步分析各个变量的影响，将整个立体匹配过程分为三步：

1.计算在参考帧中的极线

2.在极线上找到最好的匹配位置λ∗∈R(视差)

3.通过λ∗求出逆深度d∗

这三个步骤分别涉及三个误差：**几何视差误差**，由ξ和π中的噪声将影响第１步极线的位置，从而导致匹配点位置的误差；**光度视差误差**，在图像I0和I1上的噪声将影响第２步匹配位置的求取；**逆深度计算误差**，逆深度误差除了与上述两个误差有关外，还与基线、像素位置等有关。接下来将对这几个误差一一建模分析。

**a.几何视差误差**

**几何视差误差（Geometric disparity error）**，记为ϵλ。这一部分论文从几何和代数上分析了几何视差误差的具体形式。

首先我们从**代数**上分析。理论上，几何视差误差可以准确地计算出，但计算复杂度较大，不够高效，故使用了近似的误差模型进行了代替。首先，极线可定义为：

L:={l0+λ(lxly)∣λ∈S}(6)

其中，λ为视差，l:=(lx,ly)T是归一化的极线方向向量，l0是极线上对应无穷远点的图像点（但我认为当作极点更容易理解，不影响结果，只是视差的具体数值不一样）。设我们要解求的点在极线和灰度等值线（isocurve）的交点上，在局部我们将等值线近似为直线，同时梯度方向近似不变，则有：

l0+λ(lxly)＝!g0+γ(−gygx)γ∈R(7)

这里g0为等值线上的一点，g0处的灰度值等于或接近待匹配像素的灰度值。g:=(gx,gy)T表示归一化梯度方向向量，则(−gy,gx)T表示归一化等值线方向向量。故等式左边是沿极线搜索到的匹配点坐标，等式右边表示该点位于等值线上。由于下面会分析图像噪声的影响，所以这里假设g0和g不受噪声影响。现在用这些相关量将λ∗表示出来(将式(7)两边同左乘gT并移项得):

λ∗(l0)=⟨g,g0−l0⟩⟨g,l⟩(8)

由[协方差传播率](https://en.wikipedia.org/wiki/Propagation_of_uncertainty)可得：

σ2λ(ξ,π)=Jλ∗(l0)(σ2l00σ2l)JTλ∗(l0)(9)

由式(8)得：

Jλ∗(l0)=∂λ∗(l0)∂l0=−gT⟨g,l⟩(10)

代入式(9)得：

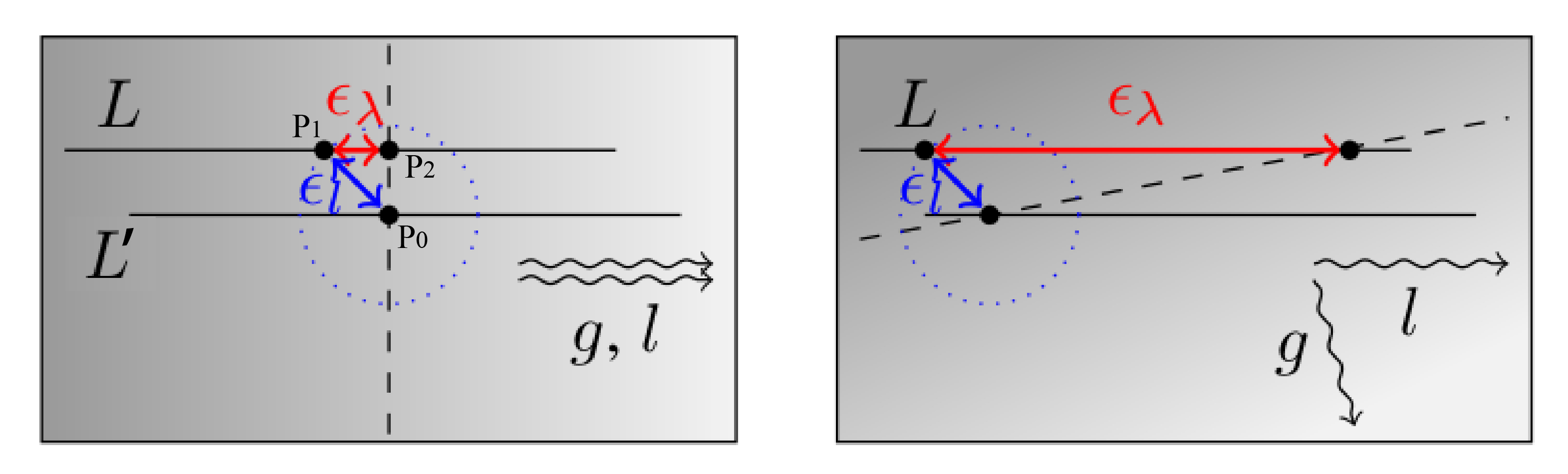
σ2λ(ξ,π)=⟨g,g⟩⋅σ2l⟨g,l⟩2=σ2l⟨g,l⟩2(11)

其中，σ2l为极线位置误差ϵl的方差（ϵl可以看作位姿扰动的影响，其示意图见下面）。总之，从式(11)我们可以得出以下结论：

* 位姿扰动造成的误差σ2l越大，则几何视差误差越大
* 梯度g和极线l的夹角越小（内积越大），几何视差误差越小

代数上分析完后，再从**几何**上分析一下，毕竟数形结合才能更好理解，否则这个地方理解起来有点抽象。如下图所示，P0是假设没有误差时匹配到的点，L′是无误差时的极线。但是由于位姿扰动影响，在图上表现为P0偏移了ϵl这么远（论文认为位姿扰动只影响极点位置计算，但极线方向不受影响，所以极点l0等极线上的点都偏移了这么远），论文假设l0受各向同性的高斯噪声，因此P0偏移后的位置可能出现在蓝色虚圆的任意一点P1。偏移后极线为L，虚直线为等值线。那么现在，在极线L上要匹配到同样灰度值的点，就得**多走**ϵλ这么一段距离，这就是几何视差误差。

从图上我们可以清晰看出梯度方向与极线方向夹角大小对几何视差误差的影响，可以得出和上面代数分析部分一样的两个结论。



**b.光度视差误差**

**光度视差误差（Photometric disparity error）**这一部分论文也从几何和代数上分析了几何视差误差的具体形式。首先我们从**代数**上分析。在立体匹配时我们使用SSD误差（见2.1.3节），即：

λ∗＝minλ(iref−Ip(λ))2(12)

其中iref是参考帧像素灰度，Ip(λ)是极线上视差λ处的像素灰度。假设实际中可以获得一个比较好的初值λ0，则将Ip(λ)一阶泰勒展开：

Ip(λ)≈Ip(λ0)+gpΔλ(13)

则误差函数变为：

λ∗＝minλ(iref−Ip(λ0)−gpΔλ)2(14)

可以解出Δλ，则有：

λ∗(I)=λ0+Δλ=λ0+(iref−Ip(λ0))g−1p(15)

注意这里gp是图像Ip极线上的梯度，因此是一维的。这里同样不考虑梯度的噪声，只考虑两个图像的噪声，则有：

σ2λ(I)=Jλ∗(I)(σ2i00σ2i)JTλ∗(I)=2σ2ig2p(16)

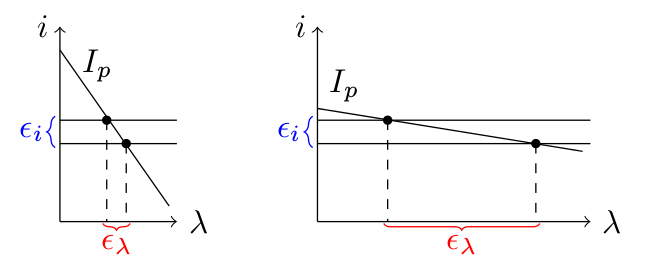
这里由于同时对关键帧和参考帧的灰度都计算了噪声，二者误差相互独立，因此会有２这个系数。故也可以这么求：

σ2λ(I)=Var(λ∗(I))=(Var(iref)+Var(Ip))g−2p=2σ2ig2p　(17)

其中σ2i是图像的高斯噪声的方差。从上式可得结论：

* 图像噪声越大，光度视差误差越大
* 极线方向上梯度越大，光度视差误差越小

代数上分析完后，再从**几何**上分析一下。如下图所示，比较直观。直线斜率的绝对值表示极线上梯度大小，当梯度值越大时，可以看出光度视差误差越小。直接法因为是靠图像梯度来不断调整位姿的，因此梯度必须较大，这样才能在优化中较快较好地收敛。这部分内容在《十四讲》中也有提到。



**c.逆深度计算误差**

这部分主要是讲基线、像素位置等对逆深度误差的影响。首先，视差λ越大，我们知道物体的深度越小，那么逆深度越大。论文中认为当旋转角度较小时，λ和d近似成正比例关系。而λ误差方差可以表示为几何视差误差和光度视差误差方差之和（因为两者独立），故就有下面的式子：

σ2d,obs=α2(σ2λ(ξ,π)+σ2λ(I))(18)

其中α是权重。更准确说，α是当前像素的深度观测值不确定性，其定义如下：

α:=∂d∂λ(19)

其中分子为逆深度搜索范围，分母为极线段搜索范围。如果一个小的极线上的变化会导致深度变化大，那就是说此时不确定性大，故权重大一些。这个概念我们在学习三角化时早已领略过：即平移太小，三角化的精度不够。因此，平移小的像素，就让它误差的权重大一些，我们不太想要这种像素。α具体计算公式推导可以参考[4]。考虑到在极线上搜索匹配点的时候，是使用了多个点，因此这里给出了逆深度误差的上限：

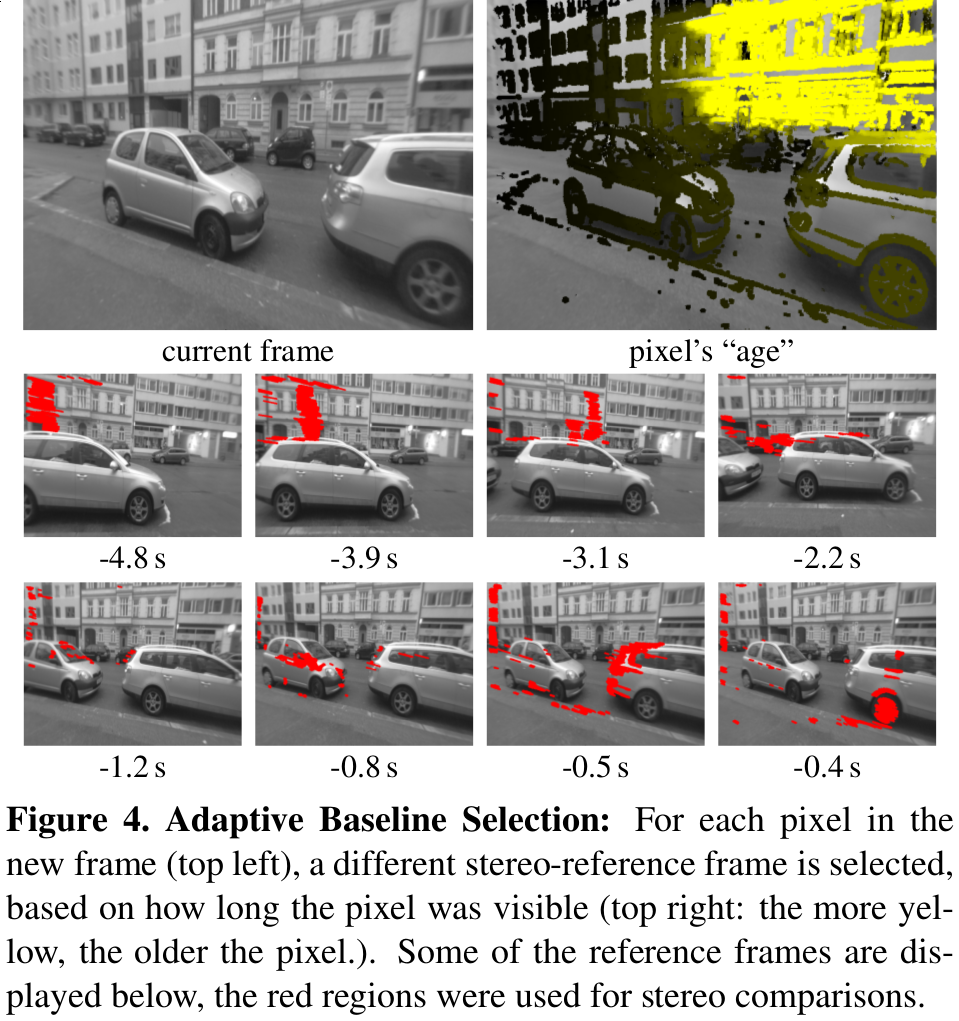
σ2d,obs≤α2(min[σ2λ(ξ,π)]+min[σ2λ(I)])(20)

总结一下：几何视差误差与**梯度方向**有关，光度视差误差与**梯度大小**有关，权重α与**基线长度像素位置等**有关。这三个因素共同影响立体观测精度，我们可以据此只把好的像素的观测结果保留，并利用这计算的方差进行地图的更新（见2.1.4节）。

**2.1.2 为每个像素自适应选择参考帧**

挑选好像素后，需要为每个像素找合适的参考帧。为什么需要找合适的参考帧？因为我们知道在立体视觉中：

* 小基线，准确度accuracy高，精度precision低
* 大基线，准确度accuracy低(有局部最小值，易误匹配)，精度precision高



因此，必须**尽可能**选择距离当前帧较远的帧作为参考帧，但是当极线上超过一定距离还没有匹配到，就需要**退而求其次**，选择距离较近的、新获得的这些帧作为参考帧。论文提出一种自适应的方法，下面使用例子来说明这个过程：

* 当前0s我们获得一个帧（current frame）
* 直接法估计了位姿，并挑选了一些好的像素，如右上角图所示
* 找最老的帧，比如直接从当前的关键帧开始匹配，但可能没有匹配到，故使用比较新的帧来做极线搜索，结果找到了-4.8s的帧，那么这一小部分像素的参考帧就是-4.8s时的这个帧了。这部分像素因为跨度比较大，因此像素年龄最大，立体观测的精度应该也是最高的。
* 但是别的部分的像素还没找到属于自己的参考帧，因此退而求其次，找更新一点的帧，结果找到了-3.9s的帧
* 因为时间越近的帧差别越小，如-0.4s的帧和0s的帧差异很小，所以**一定**会匹配成功（短基线准确度高），只是精度没那么高罢了。所以反复进行上面的过程直到**所有**好的像素都找到对应的参考帧。
* 计算深度并进一步更新地图。

**2.1.3 立体匹配策略**

立体匹配是在当前帧（current frame）和参考帧（reference frame）之间进行的，而不是关键帧（key frame），不过关键帧也可以作为一部分像素的参考帧。

在立体匹配的过程中，论文采用了极线段上５个采样点计算SSD误差的方式。采用５个点主要的方式很大程度上提高了匹配的效率。由于这５个点是相邻的，在极线段上移动的时候，每次只需要更新一个点的值，这就非常高效了。在立体匹配的误差计算方式上有很多中方法，论文中使用了SSD（Sum of Squared Distance）误差：

ESSD(u)=∑i[I1(xi+u)−I0(xi)]2(21)

**2.1.4　深度观测融合**

通过当前帧匹配的像素为深度提供一个新的观测值，然后就可以把当前观测的深度融合到关键帧的深度地图中去。这里有两种情况：当对应像素点没有深度先验时则由新的观测值构建新的先验；当已经有先验值的话，则把新观测值融合到先验中去。在这个融合的过程，使用了两个高斯分布乘法的方式，在《十四讲》chap 13中也有介绍：对于给定先验N(dp,σ2p)以及有噪声的观测值N(do,σ2o)，给出后验估计：

N(σ2pdo+σ2odpσ2p+σ2o,σ2pσ2oσ2p+σ2o)(22)

**2.2 深度图的传播**

在构建新的关键帧时，进行深度图的传播。假设两帧间的旋转是很小的，逆深度就可以近似为：

d1(d0)=(d−10−tz)−1(23)

这里的tz是相机沿着光轴方向的位移，同样根据协方差传播率则有：

σ2d1=Jd1σ2d0JTd1+σ2p=(d1d0)4σ2d0+σ2p　(24)

这里σ2p的为预测不确定性。需要注意代码中在实际求逆深度的时候，是考虑旋转的，把参考关键帧上的点通过se(3)变换到当前新的关键帧上来，然后求逆深度。但是求方差则是用上述近似方法求的。

一个像素点只能有一个逆深度的先验值，在得到一个新的观测值的时候，根据方差来判断对新的观测值融合或者舍弃，论文中称这部分为Collision handling：

* 当两个逆深度观测值的差值小于2σ的时候，则认为观测有效，根据(22)融合
* 否则，离相机较远的这个深度估计认为是被遮挡（occluded）了的（逆深度假设无效），并且舍弃该逆深度信息。

**2.3 其他**

论文还进行了一些其他处理，如把每个像素与其周边的加权深度作为改点的深度值，假如两个邻接深度之间的差值远大于2σ，他们便不做这个处理。处理之后，各自的方差不变。这里其实是一个Edge-preserving smoothing，也叫地图的正则化。另外，论文中还用了outlier removal。

**3 Map Optimization**

这部分在论文中叫建图一致性约束（constraint acquisition），是算法的核心（解决尺度问题）。因为长距离会出现尺度漂移，因此建图一致性约束也就是做闭环检测和全局优化。我们先看一下关键帧之间的sim(3)位姿估计。

首先，将每个关键帧地图的平均逆深度归一化为１。然后最小化误差函数来对齐不同的关键帧，误差函数如下，其实相比se(3)跟踪时的误差项多了一个尺度项：

E(ξji)=∑p∈ΩDi∥∥∥r2p(p,ξji)σ2rp(p,ξji)+r2d(p,ξji)σ2rd(p,ξji)∥∥∥δ(25)

这里的光度残差和方差的定义和se(3)跟踪是一样的:

rp(p,ξji):=Ii(p)−Ij(ω(p,Di(p),ξji))(26)

σ2rp(p,ξji):=2σ2I+(∂rp(p,ξji)∂Di(p))2Vi(p)(27)

以及深度残差和方差定义如下，可以看做深度图的光度残差：

rd(p,ξji)=[p′]3−Dj([p′]1,2)(28)

σ2rd(p,ξji)=Vj([p′]1,2)(∂rd(p,ξji)∂Dj([p′]1,2))+Vi(p)(∂rd(p,ξji)∂Di(p))　(29)

这里的p′:=ωs(p,Di(p),ξji)表示从图像帧i变换到图像帧j上的点。这里在求解sim(3)的时候也是同求se(3)一样，使用了迭代变权重高斯牛顿算法。此外，同样也使用了Huber核函数。

下面看一下回环检测部分。由于论文中采用的是直接法，虽然代码中有fabmap检测闭环的部分，但是其默认检测闭环的方式是帧与帧之间做双向跟踪（Reciprocal tracking check）。首先搜索最近的10个关键帧及一些外观较像的帧作为候选帧，对每一个候选帧都计算其与当前关键帧彼此跟踪的sim(3)，然后计算两者间的马氏距离的平方：

e(ξjki,ξijk):=(ξjki∘ξijk)T(Σjki+AdjjkiΣijkAdjTjki)−1(ξjki∘ξijk)(30)

详细的介绍可以参考[5]。当距离足够小时表明相似度较高，就将这一帧插入全局地图中。最后执行图优化（g2o中的pose graph optimization），边为连接关系，节点为关键帧，即优化：

E(ξW1⋯ξWn)=∑(ξji,Σ)∈ε(ξji∘ξ−1Wi∘ξWj)TΣ−1ji(ξji∘ξ−1Wi∘ξWj)(31)

在大的回环检测时，为了更好地优化、增大收敛半径，论文提出可以使用关键点提供初始值、高效二阶最小化（Efficient Second Order Minimization，ESM）和由粗到细（Coarse-to-Fine）这三种方法，这里就不再详细介绍了。