<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

Monocular-SLAM从当初使用卡尔曼滤波器发展到现在的keyframe-based SLAM系统。Strasdatet总结了必滤波器的方法更加精确的 keyframe-based技术，其代表作之一就是PTAM。

但是PTAM的出现并不代表单目SLAM已经做好了，因为地图只是用来计算相机的姿势和位置，他们对于从不同的角度进行观察的相机进行重定位并没有用处（原本是一帧帧图像获取进来，获取的特征点也是二维的，因此当另一个相机的viewpoint如果偏差距所有的keyframe都比较远则无法进行定位）。这个问题在动态摄像的时候显现的更加严重。为了解决这个限制。我们认为地图的可识别能力是十分关键的，但是PTAM类似的系统在这些方面的缺陷是十分大的。

我们之前继承了一套在PTAM上十分有效的环境识别系统来做热localisation和loop closing，给予ORB特征的词袋算法（Bags of Binary Words for Fast Place Recognition in Image Sequences），能够在实时重定位的情况下保持高度的viewport不变性.

与Stradat相似的是，我们的系统也使用了的covisibility信息来操作大尺度，同时使用keyframe之间covisibility信息来优化我们的环境识别的效果，继承了一个删减冗余keyframe的算法，？so it is multiple-pass efficient？系统主要有三个任务：tracking、local mapping、loop closer，三个线程独立，所以可以在多核的机器里面跑。

一、Map

A. Map points or 3D ORB

从3D重建中析取map points，即图片中的ORB特征，so map points are triangulation of FAST corners???。这个patch的大小和朝向取决于FAST被探测到的尺度空间的大小和key points的方向。每个map point都有一个相关的patch向量

n（观测方向向量的平均值、the rays that join the point with the keyframe camera centers）。一个map point可能与多个key frame中的ORB特征相关，所以一个map point有多个descriptor。（但是最终多个ORB会合成一个D，作为这个特征点三维空间上的描述子）

当不存在错误的匹配时（即贴在这个map point上的所有ORB特征都是正确的），这些描述子根据其viewport的不同而体现出这同一个map point的不同appearances。为了加速数据的关联（匹配？），每个map point存一个代表性的描述子，为到所有相关描述子的最短距离（这样为某个frame中的ORB寻找map point时较快）。并且存最大和最小距离dmax和dmin来判定它能够被观察到的最大和最小的尺度（即该点能够被观测到的极限距离）。

B. Keyframes

每个keyframe都存下frame中所有的ORB特征（不管有没有和某个map point关联上，所以会存在一些free ORB，这些free ORB可能会与后面继续跟进的keyframe，或者loop的时候可能转变成map point），以及这个frame所对应的camera pose，keyframe在被发现已经冗余后将被删除。

C. Covisiblity graph

covisibility info用来制作local map和加速环境识别，作为无向图存储，点是keyframe，当两个keyframe之间共享了至少Θ得map points则有边，所以Θ决定了这个图的联通程度，Strasdat提出是15~30。边权值为共享的map points数量。（由于是根据map point来定边的，而在loop closure没有发生的时候，map points也往往是由时域上连续的帧和成的，所以当camera从一个场景转移到另一个相似的场景的时候covisibility graph并不会将这两个局部连起来，当loop closure真的发生的时候，再对covisibility graph进行处理，将两个局部合成一个局部，loop fusion）

二、Place Recognition

place recognizer用来探测回环并且用来重定位。我们之前使用词袋的方法（bags

of binary words，使用k-mean建立分层树，预先训练分类器）。我们对于之前的工作主要有两个提升：

1.返回多个可能的candidate而不只是一个相似性分数最高的

2.使用covisibility info替代掉所有的临时信息（要求对当前帧Ki进行场景识别，不只是对比两帧之间的分数，而是对某一keyframe进行分组（在covisibility graph上直接与该keyframe相连的分为一组），对该组的匹配分数进行求和，这样求和的分组不再是一个连续时间段（不是根据时间常数计算，而是根据covisibility graph计算），以一组keyframe表示一个place，增加匹配的鲁棒性）

A. Recognition Database

数据库包含反向索引（分层词袋树上每个节点都有索引到包含它的keyframe，并且存下该节点在keyframe中所占的权重，用来快速查找common visual points的keyframe，注意这里词袋树上的节点只是视觉效果上相似的特征点，而并不一定属于一个map point）。新的keyframe在local mapping处理完它（说明该帧已经确定是keyframe了）的时候，由loop closer插入数据库中（需要判断loop detection）。

当relocalisation或者loop closer访问数据库的时候，数据库将对所有与query image享有共同的视觉点（即在词袋树上有节点同时指向这两帧，说明这两帧中有些特征点从视觉上来看是很相似的）的keyframe给出一个相似性的分数。（由于视觉点并不意味着是同一个map point，这样意味着视觉上相似度比较高的帧都会被计算出来，虽然可能在covisibility graph上并没有边直接相连，这样可以用来完成loop detection）由于两个keyframe之间可能出现视野重叠（即相似度比较高），所以不会只有一个keyframe有一个很高的分数，而是会出现若干个分数很高的keyframe。我们建立了一个covisibility group，对在covisibility graph中直接相连的keyframes的分数求和（这样对于定位来说更好，因为在covisibility graph上直接相连说明这些帧组成了一个place，可以比较好的反应这个place的相似度而不只是其中某一帧的特性），用这个作为keyframe的匹配结果（实际上是找到了这个keyframe应当出现在map中的那个环境中）。在5和1中使用的是时间上相近的帧来求和，这样对于时间上不连续，但是看的仍然是同一个地点的keyframe没有处理（比如loop closure完成之后某些不在连续时间中插入的帧实际上在空间上是连续的，能够组成一个place）。而且我们不是直接取最好的匹配，比最高分数不少于90%的我们都作为candidate。

B. Efficient and refined ORB matching

在relocalisation和loop detection中访问数据库后，我们都会各自计算camera pose和similarity transformation，用来验证geometrical validation。我们需要先计算两张图片间ORB的对应关系。暴力匹配能够在不减少重要性能的情况下加速一个数量级（如果我们只对于在词袋树上second level的ORB特征进行检测）。初始响应再被orientation consistency test改善一下[5]。

C. Covisibility Consistency

loop closure的鲁棒性十分重要，假的loop closure会毁了整个图。[1]提出了一个temporal consistency test，系统等待时间上连续的一系列匹配才接受这个loop closure。这里我们使用covisibility consistency test取而代之，两个keyframe是连续的取决于他们俩之间在covisibility graph上是否有边相连。

三、Tracking

为两个相邻的帧找到homography或者essential matrix里初始化地图（刚开始扫描的时候）。一旦初始地图存在了，tracking会对每一个进来的帧进行camera pose评估。

A. ORB extaction

建立scale pyramid，并且使用FAST对每层进行角点探测，然后再计算descriptor。

这里涉及许多参数：金字塔的层级、两层之间的scale factor，ORB的抽取数目，detector的threshold，这些都决定了被保留的key point的性质（数目等）。这些参数对于特征的尺度不变性，其他任务的计算花销，和在image中的keypoint分布都有直接影响。我们使用的参数：8层、1.2的scale factor、1000ORB。为了最大化keypoint在image中的离散程度，对图片进行网格划分，并且每个格子保留相同数目的点。如果某个格子不够，则其他格子可以保留更多（意思是总数量相同）

B. Track from previous frame

当最后的一帧被成功跟踪，那么对于camera pose首要考虑的评估是从最后的那帧过来的（在上一帧（不一定是keyframe）中我们找到了很多ORB特征，将那些被绑定到map point上的ORB拿出来作为candidates（说明这些ORB是比较明显且稳定的特征，被多个帧探测到了），然后在相同的图片位置上进行小范围搜索以获得匹配（两帧之间图像差距比较小，这样做可以有效的减少搜索量）），为了防止由于相机的快速移动而没有找到足够的响应点，当响应点数目不够时，增大搜索ORB的范围。初始响应将使用orientation consistency test进行修正。这时我们有一系列的3d或者2d的响应点，使用RANSAC来解决PnP。

C. Relocalisation:Track from scratch

如果跟踪丢失，我们将frame转化为bag of words并且查询数据库来获取relocalisation candidates。如果有多个地点（在Recognition Database中提到，返回的不只是最好的，还有90%以上的）与当前帧类似，则会得到多个candidates。为每个 candidate keyframe和当前帧之间都计算ORB correspondence（在词袋树上ORB matching）。然后对这些响应都分别使用RANSAC，并且算出他们的camera pose(PnP)，如果我们找到了一个camera pose，那我们就可以继续跟踪了。（因为Recognition Database只是根据BoW特征查询得到结果，并没有考虑epipolar constraint，所以可能存在多个地点都是假的，比如两个地方都有几本相同的书，但是书的摆放不同，这样就可以又好又快的识别出来对的那个位置 ）

D. Track the local map

现在我们可以在大世界中进行camera pose的评估了，我们可以将map（树上面挂着的那些map points么？）投影到frame中并且搜索更多的map point correspondences。和PTAM中将所有map point都投影进去不同的是，我们使用Strasdat的方法投影到一个local map中去（double window）。local map包含一个frame set K1，这些frame与当前frame共享map point，然后获取一个covisibility graph中K1点的邻点集K2（也在local map中，所以local map分为两个集合，K1和K2）。在连通性很强的途中，K2会很大，所以我们限制了K2的大小（即我们获取

的K1的邻点的数目）。local map还有一个reference keyframe Kref∈K1，它与当前帧共享的map points最多。

现在对所有在K中的map points按照如下方式在当前frame中进行搜索（用来获取当前frame中对于local map有用的信息）：

1.计算map point在当前帧中的投影位置，如果超出了image就丢掉

2.计算map point的法向n和当前的视线v，当v·n < cos(45)就丢弃掉（看的方向与点的法向小于45度集成在map point上的那些ORB就ambiguous了）

3.计算该点离相机中心的距离，如果不在dmin~dmax之间也抛弃掉

4.计算它在frame中的尺度d/dmin（dmax、dmin实际上是有pyramid的最底层和最上层来决定的？）

5.在预测的scale（就是上面算出来的）中比较描述子D（map point中取平均值的那个）和帧中的ORB

6.将map point和best match连起来（将当前的ORB feature也加入到triangulated map point中去）

这样一套下来之后就将当前帧与local map中的map points全部关联起来了，之后就可以根据这个关系用最小化误差做优化定位

E. Final pose optimization

tracking的最后一步是优化camera pose，使用上面的B或者C我们获得一个initial estimation，并且在D中获得了这个frame和local map points直接的所有ORB对应关系。使用g2o的Huber cost function完成minimizes the reprojection error maintaining fixed the map point positions，优化之后就是camera pose的准确信息。

F. New keyframe decision

跟踪成功之后我们应选择是否我们将它作为新的keyframe插入。下面是作为新keyframe插入所必须满足的条件：

1.上一次重定位必须在20帧之前（即重定位后20帧内不可选做keyframe）

2.当前帧至少track50个点（local map中的map points吗？）

3.当前帧和Kref相比，track了不到90%的点（信息量很大）

4.距上一个keyframe至少10帧，并且local mapping线程已经处理完了上一个keyframe。或者过去了20帧，并且有一个信号给local mapping要求做local BA（inner window）

在这里我们和PTAM不一样的是，并没有使用和其他keyframe的距离相关的条件。条件3是visual change，条件4是为了尽可能快的插入帧，条件1保证了good precious relocalisation（说明上次的relocalisation是稳定的，不会重复做不必要的）以及条件2保证了good tracking

四、local mapping

local mapping线程需要处理新的keyframe、更新并优化他们的local neighborhood（covisibility graph？）。新的keyframe由tracking线程插入到队列中，这个队列理想状态下应该只包含上一个插入的keyframe。

local mapping按照如下决策处理oldest keyframe：

A. Bags of Words conversion

将keyframe转化为BoW，并且得到词袋树的相关信息。

B. Triangulate new map points

新的map points通过triangulating不同keyframes中的ORB产生。PTAM只是用最近的keyframe来triangulate，这种方法的视差最小。而我们使用的是在covisibility graph中享有相同点最多的N个邻居。需要限制邻点数量是因为联通性较高的图邻居太多，计算花销不能够承受。对于当前keyframe每个自由的ORB（没有绑定到map point上面的），我们将它与其他keyframe中的自由ORB进行descriptor的比较（在同一个二级词袋树节点下的，优化搜算量），抛弃掉那些不符合epipolar constraints的。一旦找到了一对就将他们triangulate。如果视角小于1度就抛弃掉（没有用处）。一开始map point只是从两个keyframe中被看到，但是随后如同上方的track the local map中不断被更新和使用（预测scale、重投影等等）。用来增加local map的稳定性。

C. Update the local area

map point由接下来的3个keyframe来检测是否错误，并且必须满足一下两个条件才不会被丢弃：

1.tracking必须在超过25%的frames中在被预测的地方找到这个点

2.如果map point被多于一个keyframe经过，他必须至少被3个keyframe看到。

当keyframe中的点产生了新的衡量时，要重新计算covisibility graph的边和描述子D

D. Local Bundle Adjustment

和PTAM类似，我们使用local BA优化当前keyframe、在covisibility graph中和它相连的所有keyframe、以及在这些keyframe中能够被观测到的map points。而那些能够观测到这些点，但是不与正在处理的这个keyframe直接相连的其他keyframes，则在保持自身不变的情况下参与到优化中。（即优化只对这个核心图进行修改，图的外围视为固定条件）。使用g2o中的LMA和Huber robust cost function来实现非线性优化。优化后重新为每个map point计算向量n、dmin、dmax。

E. Prune redundant keyframes

抛弃掉一些90%的map points都能够在其他的keyframes看到（相同的或者finer的尺度）

五、Loop Closer

loop closer线程用来探测地图中的loops，并且使用全局优化来保持全局的连续性。还通过local mapping处理的最后一张处理的keyframe Ki并且完成一些任务。

A. Loop detection

首先我们计算Ki和它的邻居（covisibility graph上）的BoW相似性，并获得最低分smin（如果某个frame分数比这个低，说明那个frame根本就不是Ki附近）。并且所有与Ki直接相连的都会被抛弃掉（在做loop fusion之前本身就是直接相连的，说明这些keyframe本来就被视为在同一个环境中的，在做loop detection，我们的任务是探测是否本来应该在同一个环境中的keyframe在我们mapping的时候却被重复建图了（我们走到了原点），当我们检测到这种情况，则应当将这两个重复的图整合起来，这就是loop detection然后loop fusion）。为了增强鲁棒性，连续三次探测成功才行。（词袋树只是用来聚类用的，方便查询，跟map points没有必然关系，covisibility graph相邻实际上在时间上还是比较近的，只是不再规定一个时间常量来做covisibility graph，这样的话拿着相机在原地晃悠这些相同点会被映射到同一个map point上，但是你走开之后遇到相似的特征则不会被聚到同一个map point上）

B. Compute the similarity transformation

弹幕SLAM有七个自由度：三个旋转、三个平移、以及一个尺度。因此为了封闭一个环，我们需要计算当前帧Ki和the loop keyframe Kl的相似转换来获得loop中的累计误差。这个相似计算也用来计算loop的几何有效性(geometrical validation)。

在重定位时，如果有多个地方的外观相似，则可能对于当前帧存在多个loop candidates。首先我们计算当前帧和the loop keyframe之间的ORB correspondences，按照efficient ORB matching。这样我们队每个loop candidates都有3D到2D的对应。我们队每个candidate都做RANSAC来取得相似转换。然后选取一个相似变换Sil，这个变换有足够多的inliers使得我们能够接受这个loop candidate。

C. Synchronization with local mapping thread

在做这个loop纠正之前，loop closer向loop mapping发送一个信号让它在完成手上的工作后停止工作。在local mapping停止工作之后loop closer再开始loop纠正的工作。（因为loop纠正会影响keyframe的pose，如果local mapping线程不停止工作，则他使用的数据是混乱的）

D Loop fusion

loop correction的第一步就是融合重复的map points并且在covisibility map中插入新的边（这样两个局部就被融合到一起了）。一开始当前的keyframe的pose Tiw会被相似转换Sil纠正并且这个纠正会被传播到Ki的邻居上去（将变换要接起来），这样loop的两端就对齐的接起来了。所有在loop keyframe及其邻居中看到的map points会被投影到KI和他的邻居上去，并且在投影出进行小范围的搜索来匹配（因为loop会有误差，所以这些匹配点是对不准的，进行小范围搜索配准后误差项用来BA优化），和track local map的原理一样处理。当所有的map points匹配完之后，并且那些inliers在计算Sil的时候会被融合。所有参与了融合的keyframe都要更新在covisibility graph中他们的边并且有效的创建边来使得loop closure合并（更新边是因为原来的边权值等会有改变，加上边就使得loop的两头合拢了）。

E Covisibility pose graph optimization

为了使得loop闭合并且纠正那些累计的错误，我们必须使用全局优化。但是由于在使用BA时robust cost function是必须的（否则的话outlier会对整个图的优化产生十分大的影响，所以必须对outliers鲁邦），我们需要算出一个初始解使得优

化是收敛的。初始解将在covisibility graph的pose graph optimization的结果中产生，？？并且在初始解中loop closure edges的误差会沿着图分布？？pose graph optimization用来在相似变换的基础上实现纠正scale drift？？这个方法更具一般性，并且它只是一个将keyframe与之前的keyframe连起来的一个环，关于residuals的定义以及优化过程仍然是相同的？？

（看一下他们之前的工作、以及那个pose graph optimization的工作，scale drift）

F Global Bundle Adjustment

整个地图在使用了pose graph optimization的种子之后实现了优化。这个优化和local bundle adjustment中的优化是相同的，但是这里所有的keyframe和map points都得到优化。所有优化都完毕之后local mapping再次开始工作