VO系统综述

MulinB按：最近在学习SLAM算法，这里作为阅读笔记记录和总结一下。这里关注的主要是基于视觉的Visual SLAM或Visual Odometry，也包括一部分图像和IMU融合的Visual-Inertial Odometry相关算法。

注：下文中部分链接指向[Google Scholar](https://scholar.google.com/)及[Youtube](https://www.youtube.com/)，有些用户可能无法访问。

# 一、概述

Simultaneous Localization and Mapping (SLAM)原本是Robotics领域用来做机器人定位的，最早的SLAM算法其实是没有用视觉camera的（Robotics领域一般用Laser Range Finder来做SLAM）。本文主要关注基于camera图像做SLAM的算法，即Visual SLAM算法。本文对SLAM和Visual SLAM不做区分。

其实SLAM是一个研究了30年的topic了，然而到目前仍然没有广泛走进人们的生活。个人认为，其中一个原因是SLAM对实时性要求比较高，而要做到比较精确、稳定、可靠、适合多种场景的方案一般计算量相对较大，目前移动式设备的计算能力还不足够支撑这么大的计算量，为了达到实时性能，往往需要在精确度和稳定性上做些牺牲。因此在具体的应用中，往往需要根据移动设备所具有的传感器组合、计算能力、用户场景等，选择和深度定制合适的SLAM算法。比如，无人驾驶汽车和手机端AR类应用的SLAM算法就非常不同。无人驾驶汽车中，传感器的使用可以更多样化，其中在建地图时和实时定位时的算法又很不一样，需要根据具体情况下使用的传感器类型和可以获得的地图辅助数据类型等信息进行综合性的方案设计，以达到精度和稳定性最大化。而在手机端的AR类应用中，如何利用标配的单目+MEMS IMU在旗舰类手机上得到更精确、更稳定的算法是关键任务。本文主要focus在后者。

关于SLAM的基础入门，这里不多啰嗦，列一些不错的学习资源如下：

Tomasz Malisiewicz的博客有一篇不错[关于SLAM的文章](http://www.computervisionblog.com/2016/01/why-slam-matters-future-of-real-time.html)，其中很多link的资源都很好

国内有个不错的SLAM组织，叫“[泡泡机器人](http://www.slamcn.org/index.php/%E6%B3%A1%E6%B3%A1%E6%9C%BA%E5%99%A8%E4%BA%BA%E5%85%AC%E5%BC%80%E8%AF%BE%E7%A8%8B-%E5%B7%B2%E5%8F%91%E5%B8%83)”，其中有很多不错的SLAM课程，可以关注其微信公众号查看往期课程。另外该组织的不少成员都有博客，有很多不错的算法详细介绍的文章

另外再列一些比较强的研究SLAM的组织和牛人：

[ICL的Andrew Davison组](https://www.doc.ic.ac.uk/~ajd/index.html)：老牌SLAM研究组，专注SLAM十几年，很多SLAM领域的大牛都出自这个组，比如KinectFusion和DTAM的作者[Richard Newcombe](https://scholar.google.com.hk/citations?user=MhowvPkAAAAJ&hl=zh-CN)、G2O的作者[Hauke Strasdat](http://hauke.strasdat.net/php/pubs.php)等

[Oxford的ActiveVision组](http://www.robots.ox.ac.uk/ActiveVision/index.html)：PTAM出自这个组，Andrew Davison也毕业自这个组

[Gatech的Frank Dellaert组](http://www.cc.gatech.edu/~dellaert/FrankDellaert/Research.html)：GTSAM出自这个组

[TUM的Daniel Cremers组](http://vision.in.tum.de/research/vslam)：LSD-SLAM出自这个组

[UZH的Davide Scaramuzza](http://rpg.ifi.uzh.ch/publications.html)：SVO和OCamCalib出自这个组

[ETH Zurich的ASL组](http://www.asl.ethz.ch/)：OKVIS和BRISK的作者[Stefan Leutenegger](http://wp.doc.ic.ac.uk/sleutene/)毕业自这个实验室

[Monash Univ的Tom Drummond](https://scholar.google.com.au/citations?user=6sWGL5wAAAAJ)：他曾经在Cambridge做过教授，FAST detector的作者[Edward Rosten](http://www.edwardrosten.com/)是他的学生，PTAM的作者[Georg Klein](http://www.robots.ox.ac.uk/~gk/)也是他的学生

[Juan D. Tardos](https://scholar.google.com/citations?user=TVf1gw0AAAAJ)：ORB-SLAM出自他的学生

[UMN MARS Lab的Stergios Roumeliotis](http://mars.cs.umn.edu/)：MSCKF 1.0出自当时还在这个实验室读书的Anastasios Mourikis，另外一篇[关于quaternion algebra的非常棒的tutorial](http://www-users.cs.umn.edu/~trawny/Publications/Quaternions_3D.pdf)也出自这个实验室

[Anastasios Mourikis](http://www.ee.ucr.edu/~mourikis/)：MSCKF 1.0出自他之手，MSCKF 2.0出自他的学生[Mingyang Li](https://sites.google.com/site/mingyangli009/publications)

# 二、Monocular SLAM算法笔记

Visual SLAM的两篇开山之作应该是[Davison ICCV 2003](https://www.doc.ic.ac.uk/~ajd/Publications/davison_iccv2003.pdf)和[Nister CVPR 2004 Visual Odometry](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5767&rep=rep1&type=pdf)的两篇paper了。Andrew Davison是从Robotics领域的视角出发，将经典的non-camera的SLAM算法应用到使用single camera做SLAM的方案中，使用了经典的EKF框架，经过后续改进就是[TPAMI 2007的MonoSLAM[2]](https://www.doc.ic.ac.uk/~ajd/Publications/davison_etal_pami2007.pdf)算法。David Nister是Computer Vision领域3D Vision的大牛（著名的[五点法](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.1518&rep=rep1&type=pdf)就是他的作品），他的[CVPR 2004 Visual Odometry paper](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5767&rep=rep1&type=pdf)是从3D Vision的视角出发设计的一个SLAM方案，整个方案设计的比较简单，只是着重连续较少帧的相对camera pose的估计，并没有完整的地图维护机制，因此容易造成drift。不过其3D Vision的算法和思想是后续很多的SLAM算法中都使用到的，这里先从Nister的这篇VO paper开始介绍。

## Nister VO (2004) [[1]](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5767&rep=rep1&type=pdf)

下面是Nister这篇VO paper的算法流程：

TrackFeatures[使用Harris提取角点，用patch correlation做match]EstimateInitCameraPose[使用五点法从2D点匹配估计相机相对pose，然后使用三帧图像做BA]TriangulateMapPoints[用点的track轨迹来三角化出点的3D位置，并估计相机pose相对之前轨迹的scale]TrackFromMapPoints[根据已经三角化的3D点，使用三点法P3P求后面帧的pose，并用Local BA迭代refine]RefineMapPoints[根据新的track结果，进一步refine 3D points的位置]End of InnerLoop?[重复TrackFromMap和RefinePoints数次]End of OuterLoop?[重复从EstimateInitPose到RefinePoints数次]InsertFirewallAndReset[插入Firewall阻止误差累积，重新从第一步开始VO]yesnoyesno

Nister的这篇paper重点介绍了vision相关的算法及实现细节（包括如何高效实现角点提取和match等），但在系统流程上有些介绍的并不详细，比如上面InnerLoop和OuterLoop的次数和条件等。算法中的很多思想在后续的Visual SLAM算法中都有沿用，比如2D to 2D用五点法计算初始pose、3D to 2D用P3P计算pose、2D correspondence -> essential matrix -> pose -> triangulation -> 3D to 2D correspondence -> P3P -> optimization的流程及子模块的迭代方式等（并不是说所有这些3D Vision算法都是Nister首创，重要的是整个流程在Visual SLAM中的应用）。

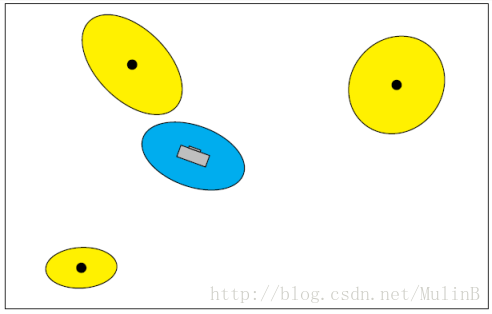
算法优点：没有对motion model的假设，主要靠三点法P3P来计算相机pose；在计算过程中引入多次bundle adjustment提高精度

算法缺点：算法流程具体细节描述的不太清晰；没有比较fancy的map points维护机制；插入firewall阻止误差累积的方式比较原始，无法消除drift；方法比较简单，没有跟踪丢失后重定位、回环检测等现代SLAM方法中比较常见的机制

## MonoSLAM (2003-2007) [[2]](https://www.doc.ic.ac.uk/~ajd/Publications/davison_etal_pami2007.pdf) ([code available](https://github.com/hanmekim/SceneLib2))

Andrew Davison的MonoSLAM是将传统机器人领域中基于laser range-finder的EKF-SLAM应用到了single camera的SLAM中，算法相对于Nister VO更加完整，其中关键的提高在于通过guided feature matching取代invariant feature matching提高了计算速度。[这里](https://www.doc.ic.ac.uk/~ajd/Scene/Release/monoslamtutorial.pdf)有个介绍MonoSLAM的不错的slides，在Github上有[MonoSLAM的代码](https://github.com/hanmekim/SceneLib2)。另外，更入门的材料，关于Kalman Filter和Particle Filter的介绍，最容易intuitively理解的是udacity上[Sebastian Thrun的视频课程](https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373)。另外，[这里](http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/)有一篇非常不错的对Kalman Filter介绍的blog。

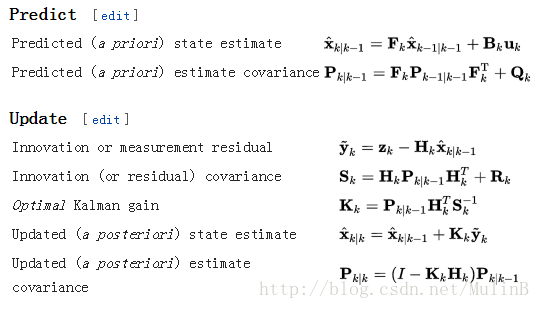
具体点讲，在MonoSLAM中，每一个state由一个当前状态的最佳估计state vector x^，以及一个表示该状态uncertainty的协方差矩阵 P 表示。其state vector包括相机的pose、velocity、angular velocity、以及场景中所有3D map points的坐标。该模型是一个对于state vector中参数的single modal multivariate Gaussian modeling，借用一个上述slides的图表示：



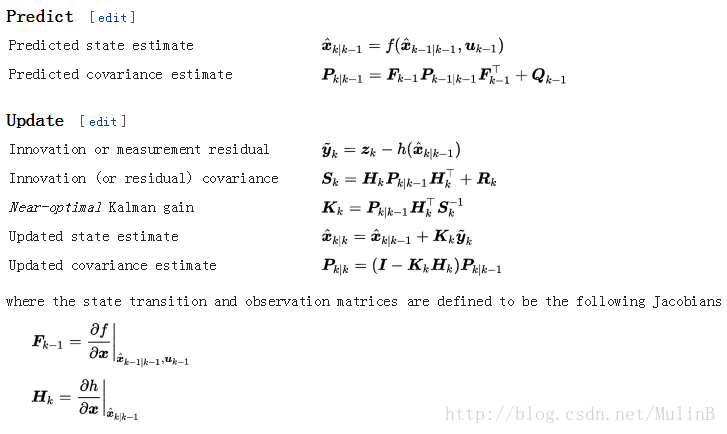
假设场景中有 N 个3D map points，那么每次更新上述协方差矩阵的计算复杂度就是 O(N2)，为了维持在realtime的performance，通常需要控制3D map points的个数 N<100。虽然这表面看起来不是很高效，paper中介绍了原因如下：之所以选择一个full covariance matrix而不选择使用covariance submatrix减小计算，是因为MonoSLAM的目标是在一间room大小的场景中进行repeatable localization，这种情况下相机的view会较多的与彼此重叠，会有频繁的loop closure发生，这种情况下用一个full covariance matrix会更加精确，而且100个左右的map points足够表征一间room大小的场景。

MonoSLAM中对于特征的检测和匹配如下：特征点检测使用的是Shi-Tomasi corner detector (类似于Harris detector，[这里](http://aishack.in/tutorials/shitomasi-corner-detector/)有其区别)，特征匹配是靠warped image patch进行[NCC匹配](https://cn.mathworks.com/help/images/ref/normxcorr2.html)（每个patch template保存其所在view的surface orientation，匹配时根据后续帧的pose将该patch template投影过去），paper中特意提到该patch template一旦初始化将不进行更新（为防止drift）。更进一步，patch template的surface orientation也是通过一个单独的Kalman filter进行estimate的(Section 3.8)。

基于Kalman filter的算法在每个时刻的计算一般分为两步：第一步，predict step；第二步，update step。在第一步中一般是根据运动模型或者控制模型预测state vector，是uncertainty propagation的过程，其中用到的运动模型或者控制模型需要根据实际场景设置其uncertainty的参数，直接影响covariance matrix的计算。在第二步中是根据观测到的结果来估计最佳state vector并减小uncertainty。借用wikipedia的公式，Kalman filter的计算过程如下：



作为对比，EKF是将上述predict step中的变换矩阵 F 更换成非线性的函数 f(−)，将update step中的观测矩阵 H 更换成非线性的函数 h(−)，EKF的计算流程如下：



看到这么多符号，我一般就跪了。幸运的是，MonoSLAM paper里Kalman filter的符号(Section 3.4-3.5)以及上述wikipedia的符号都和[这篇牛逼的深入浅出的blog文章](http://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/)里的符号是一致的，对照着该blog文章可以大概领悟到MonoSLAM paper里是在干啥(进一步整明白还要再看看代码)。值得注意的是，在计算Kalman filter update step时的innovation covariance matrix S 可以用来选择保留每一帧中信息量较大的feature。

最后，在MonoSLAM中比较有趣的是其3D map point的triangulation（paper中叫feature initialization, Section 3.6），由于可以保存3D点的uncertainty，当parallax不够大时，3D点的depth不确定性很大，可以保存其所在的3D line的信息，随着观测的增多及parallax足够大，最终再转化成一个3D点的Gaussian distribution。估计后面的SVO的depth filter的思想也是从这里借鉴的，只是在filter-based SLAM中由于uncertainty是显式表达出来的，所以比较直观，而在keyframe-based SLAM中就没那么直观了。提到这两类SLAM的比较，filter-based SLAM and keyframe-based SLAM, 比较深入的分析可以参见Hauke Strasdat的[Why Filter IVC 2012 paper [7]](https://www.doc.ic.ac.uk/~ajd/Publications/strasdat_etal_ivc2012.pdf)。注意其中keyframe-based SLAM有时也被叫做BA-based SLAM或optimization-based SLAM，Strasdat的paper中用的典型例子是Ethan Eade and Tom Drummond的[ICCV 2007 paper [3]](http://ethaneade.com/eadeICCV2007.pdf)。值得注意的是，在那篇paper中，虽然使用的是BA-based的方法，其实对于3d map point也是有uncertainty modeling的（使用的是information matrix，类似于covariance matrix，其区别可以参见Strasdat的Why Filter IVC 2012 paper [7]）。顺便提一句，[Ethan Eade的个人主页](http://ethaneade.com/)上有不错的Lie Group and Exponential Mapping的总结文档 :-)

整个MonoSLAM的流程图如下：

SystemInit[使用已知target上的几个已知corner的位置来初始化]KalmanFilterPredict[使用motion model计算state vector及covariance matrix]MakeMeasurement[根据prediction的state vector进行guided feature matching]KalmanFilterUpdate[根据match的结果update state vector及covariance matrix]ManageMapPoints[根据一些准则删除不太好的3D map points]TriangulateMapPoints[Bayesian计算uncertainty并triangulate map points]##### End of image stream? #####Endyesno

Davison的这篇MonoSLAM是他之前几篇会议paper的综合，描述了一个相对比较完整的Filter-based Visual SLAM系统，包括用motion model预测的camera pose来进行guided feature matching（paper里称为active search）、patch matching时考虑surface normal、3D map points的维护等。在paper中也提到了可以较容易的integrate gyro数据，因为其state vector里有angular velocity的参数，可以直接在Kalman filter中增加一步measurement update将gyro数据加入即可。整个系统可以在1.6GHz的Pentium M处理器上处理320x240的图像接近60fps，速度快的让人吃惊（速度快得益于其处理的feature个数比较少，平均每帧图像大约只处理12个feature）。

算法优点：速度快；可以显式建模map points的uncertainty；较容易融合gyro数据

算法缺点：Filter-based方法的缺点是只维护一个当前时刻的state，不能利用history数据进行optimization；每帧跟踪的map points太少，比较容易丢失；当需要精度较高时，需要更多的map points，这是计算复杂度会增大很快(参见[7])；在camera不太动时，会有jitter(参见PTAM paper[4]的对比分析)；系统初始化方法比较原始，属于半自动形式

## PTAM (2007-2008) [[4](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2007ISMAR.pdf),[5](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2008ECCV.pdf),[6](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2009ISMAR.pdf)] ([code available](https://github.com/Oxford-PTAM/PTAM-GPL))

PTAM是keyframe-based SLAM派系里最出名的一个算法，作者是Georg Klein和David Murray。[David Murray](https://scholar.google.com/citations?user=O5QreiwAAAAJ&hl=en)是Oxford的教授，是Andrew Davison读博士期间的导师。[Georg Klein](http://www.robots.ox.ac.uk/~gk/)和[Ethan Eade](http://ethaneade.com/)都是师从当年在Cambridge的[Tom Drummond](http://twd20g.blogspot.com/)，两位好像都在Microsoft的HoloLens工作过。有点扯远了。PTAM名声大噪的原因是，它开创了多线程SLAM的时代，后来的多数keyframe-based SLAM都是基于这个框架。PTAM受到了广泛采用bundle adjustment (BA)的Nister算法的启发，将tracking和mapping分成两个单独的线程，这样既可以不影响tracking的实时体验，又可以在mapping线程中放心使用BA来提高精度（另外BA也没有必要对所有frame做，只对一些keyframe进行BA即可）。这样以来，由于BA的引入，PTAM的精度得到了大幅提高，连Davison自己都承认MonoSLAM被PTAM clearly beaten了（参见[这个slides](http://wp.doc.ic.ac.uk/thefutureofslam/wp-content/uploads/sites/93/2015/12/slides_ajd.pdf)）。

PTAM的数据结构主要包括keyframe和3D map point。Keyframe保存的是camera pose及一个4-level的image pyramid (从640x480到80x60)。3D map point保存的是3D坐标、patch normal、以及来自哪个keyframe的哪一层。系统运行时通常有大约100个keyframes和几千个3D map points。

PTAM的mapping线程流程如下：

MapInit [其实在tracking线程里，代码里叫TrackForInitialMap][用户指定两帧，检测FAST corner并match correspondences，然后使用Nister五点法计算pose并triangulate map points，最后检测dominant plane。注意，代码里不是用五点法初始化，而是用homography初始化的。]HasNewKeyframe?[根据几个准则判断是否加入新keyframe]ProcessNewKeyFrame [代码在AddKeyFrameFromTopOfQueue函数内][将在tracking线程中与该keyframe匹配成功的map points做关联，这一般在传统的SLAM中叫做data association]ReFindInSingleKeyFrame[将更多的map points投影到keyframe并匹配，加入成功的observation，即data association]ThinCornerCandidates[由于tracking线程计算量限制，只能用较快的FAST corner detector，所以在mapping线程这里可以从容的使用Shi-Tomasi或Harris score对corner进行筛选，减少一些质量不高的corner，为下一步做准备]TriangulateMapPoints [代码里叫AddPointEpipolar][在当前keyframe与上一个keyframe之间匹配correspondence：从一帧的corner point出发，在另一帧的epipolar line上用SSD搜索匹配点，然后进行triangulation计算得到3D map point]HandleBadPoints[检查所有3D map points的质量，处理掉一些不好的点]SleepForAFewMilliseconds[线程让出CPU几个ms并准备进入下一轮]BundleAdjustRecent[小范围的Local BA，这里的待优化参数是刚加入的keyframe和与其紧密相邻的几个keyframes、以及在这些keyframe中可以观测到的3D map points，作为constraint的是所有可以观测到这些待优化3D map points的keyframes]ReFindNewlyMade[这一步也是data association，即将刚刚创建的3D map points与之前的keyframes关联起来，因为刚刚创建的3D map points只是从相邻的两个keyframes triangulate出来的，只与这两个keyframes有关联]BundleAdjustAll[如果Local BA做完了并且还没有新keyframe过来，就放个大招]yesno

PTAM的tracking线程流程如下：

DetectCorner[将image做成image pyramid，并在每一层上detect FAST corners]ApplyMotionModel[根据运动模型预测camera pose]TrackMapCoarse[将3D map points投影到当前frame并在coarse level上进行SSD匹配，这里匹配用的template也是经过affine warp的image patch]OptimizePose[进行Gauss-Newton迭代优化camera pose]TrackMapFine[同上TrackMapCoarse, 在fine level上用更多的点进行匹配]OptimizePose[进行Gauss-Newton迭代优化camera pose]UpdateMotionModel[根据camera pose结果, 更新运动模型]AssessTrackingQuality[评估tracking的质量，作为判断是否跟踪丢失或插入关键帧的依据]

除了上述在[paper [4]](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2007ISMAR.pdf)中描述的tracking和mapping，[这里的code](https://github.com/Oxford-PTAM/PTAM-GPL)还实现了[paper [5]](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2008ECCV.pdf)中的relocalization模块。大致来讲，在track丢失后，用small blurry image （40x30的经过gaussian blur及mean-subtracted的小图）进行SSD匹配来检测是否与之前的keyframe匹配度较高，如果检测到匹配的keyframe，通过先恢复出2D的SE(2)参数再进而用类似于unscented transform恢复出SO(3)参数得到camera pose。

PTAM在paper [4]中report的速度是在Intel Core2 Duo E6700 2.66 GHz处理器上处理640x480的frame能达到30fps，tracking中主要时间花费在3D到2D投影后的搜索correspondence上。另外mapping线程中的bundle adjustment一般还是比较慢，为了跟上map expanding，global BA会经常被打断。注意这里的CPU其实比MonoSLAM用的CPU要强不少，甚至比现在主流智能手机的CPU都要强。PTAM的作者后来将该算法做了一些裁剪和定制应用到了一个iPhone 3G上（处理器412MHz，比E6700慢15-30x速度），参见[paper [6]](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2009ISMAR.pdf)。

算法优点：精度比MonoSLAM高，分两个独立线程，在不影响tracking体验的情况下mapping引入BA

算法缺点：tracking依赖于map points (3D to 2D tracking)，这样如果如果fast motion时map expansion太慢会导致tracking丢失；没有model 3D map points的uncertainty

## ORB-SLAM (2014-2015) [[8]](http://webdiis.unizar.es/~raulmur/MurMontielTardosTRO15.pdf) ([code available](https://github.com/raulmur/ORB_SLAM2))

最近比较火的ORB-SLAM算法，是keyframe-based SLAM派系的一个集大成者。ORB-SLAM算法基本沿用了PTAM的框架，将近几年来被验证有效的module都加了进来，做出一个稳定性和精度很高、可以用于室内/室外和小规模/大规模等各种场景的全能系统，刷爆各种benchmark，并开源了质量很高的代码，还摘得了IEEE Transactions on Robotics的年度best paper award，可谓牛逼哄哄。关于这个算法，网上能找到很多学习资料甚至代码解读。

ORB-SLAM算法与PTAM的主要不同之处在于以下几个方面：

提出一个自动做map initialization的算法：通过correspondence同时计算homography matrix和fundamental matrix，再通过一个heuristic的准则判断属于哪种情况，并用相应情况下的算法计算初始pose。个人觉得这一点是ORB-SLAM这篇paper最大的贡献（当然其代码开源的贡献也非常大，这里只是说paper的原创点）。

在tracking和mapping两个线程之外加入第三个线程做loop closing，并在做loop closing时加入scale作为优化参数修正scale drift。

设计了更加高效和更适用于large scale场景的keyframe管理机制（比如covisibility graph、essential graph等），设计了更细致的keyframe/map points的创建、筛选准则等。

Loop detection和relocalization使用了更先进的place recognition方法（bag of words），取代了PTAM中的small blurry image方法。

Feature matching用ORB feature matching取代了PTAM原始的patch matching。

ORB-SLAM的算法流程跟PTAM有些细微的不同，这里大致列一下。也先从mapping线程开始（为了与PTAM流程图保持一致，下图里的步骤名称沿用上面PTAM流程图的）：

MapInit [其实在tracking线程里，代码里叫MonocularInitialization][分别使用八点法计算fundmental matrix和用DLT计算homography matrix，用heuristic判断属于哪种model适用，然后用相应的model估计出pose，并triangulate map points]HasNewKeyframe?[根据几个准则判断是否加入新keyframe]ProcessNewKeyFrame[将在tracking线程中与该keyframe匹配成功的map points做关联，即data association]HandleBadPoints [代码里叫MapPointCulling][检查刚加入的新3D map points的质量，处理掉一些不好的点]TriangulateMapPoints [代码里叫CreateNewMapPoints][在covisibility graph内的keyframes之间匹配correspondence：从一帧的corner point出发，在另一帧的epipolar line上搜索匹配点，然后进行triangulation计算得到3D map point]SleepForAFewMilliseconds[线程让出CPU几个ms并准备进入下一轮]SearchInNeighbors[这一步也是data association，即将刚刚创建的3D map points与相邻两级内的keyframes关联起来，将重复的map points合并]BundleAdjustRecent[小范围的Local BA，这里的待优化参数是刚加入的keyframe和与其紧密相邻的几个keyframes、以及在这些keyframe中可以观测到的3D map points，作为constraint的是所有可以观测到这些待优化3D map points的keyframes]KeyFrameCulling[基于一些heuristic准则对之前所有的keyframes做个筛查]yesno

上图可以看出ORB-SLAM的mapping中与PTAM的最大的不同之处是，ORB-SLAM的TriangulateMapPoints创建更多的map points时使用了更多的keyframes（即covisibility graph中的所有keyframes）。另外，global BA从mapping线程移到了loop closing时另起一个线程做（没错，其实系统在做global BA时同时会有四个线程在运行）。

ORB-SLAM的tracking线程如下（为了简化，这里省去了tracking lost时的relocalization的分支）：

ExtractORB[将image做成image pyramid，并在每一层上extract ORB features，这里有个小trick，将image划分成grid，在每个grid里单独提feature，保证所有的grid都包含至少5个features]ApplyMotionModel[根据运动模型预测camera pose]TrackWithMotionModel[将上一个frame里观测到的3D map points投影到当前frame并进行匹配，注意这里与PTAM的TrackMap的区别，PTAM里是将所有的map points进行投影，这里只用到了上一个frame的，所以后面另外加了一步TrackLocalMap][如果没有运动模型或者TrackWithMotionModel失败，则调用TrackReferenceKeyFrame，将上一个keyframe里观测到的map points投影并匹配]OptimizePose[进行Levenberg迭代优化camera pose]TrackLocalMap[收集更多的相邻keyframes并将其能观察到的所有map points投影到当前帧进行匹配]OptimizePose[进行Levenberg迭代优化camera pose]AssessTrackingQuality[评估tracking的质量，作为判断是否跟踪丢失或插入关键帧的依据]UpdateMotionModel[根据camera pose结果, 更新运动模型]

上图可以看出，ORB-SLAM的tracking中与PTAM的最大的不同之处是，当track当前帧时，ORB-SLAM只用了上一帧能观测到的map points来投影匹配，而PTAM直接用所有地图点来匹配，这样在地图点较多时显然会效率较低。所以ORB-SLAM是做出了更适合large scale SLAM的设计。

ORB-SLAM的loop closing流程比较简单，基本就是loop detection -> estimate Sim(3) transformation -> loop fusion -> optimize essential graph。这里有很多细节需要注意，所幸的是ORB-SLAM公布的代码注释很丰富，很容易follow，这里细节不再多说。

从paper中report的速度上来看，ORB-SLAM能在Intel Core i7-4700MQ (4 cores 2.40GHz)上track 512x382的视频流达到30fps。鉴于这个CPU比PTAM的CPU要强不少，这个算法应该比PTAM慢不少。

算法优点：具有自动初始化的功能；精度和稳定性都较高；keyframe和map points的管理机制也比较完善；系统比较完整，可以用在large scale SLAM中

算法缺点：tracking仍然依赖于map points (3D to 2D tracking)，这样如果fast motion时map expansion太慢会导致tracking丢失；依然没有model 3D map points的uncertainty，有些情况下可能会由于初始化的不好造成错误累积

## LSD-SLAM (2013-2014) [[10](http://vision.in.tum.de/_media/spezial/bib/engel2013iccv.pdf), [12](http://vision.in.tum.de/_media/spezial/bib/engel14eccv.pdf)] ([code available](https://github.com/tum-vision/lsd_slam))

自从2011年的ICCV上，Andrew Davison的学生Richard Newcombe带着[DTAM [9]](https://www.robots.ox.ac.uk/~vgg/rg/papers/newcombe_davison__2011__dtam.pdf)用[牛逼哄哄的效果](https://www.youtube.com/watch?v=Df9WhgibCQA)完爆了PTAM之后，众人意识到更dense的tracking和mapping会大幅度提高SLAM的精度和鲁棒性。直觉上想想，如果整幅图像的信息都可以拿来用，的确信息量会比提一些feature多很多。另外，Computer Vision界研究了多年的传统的image alignment及optical flow的很多subpixel精度的经验都可以借鉴，也会帮助提高精度。自此，dense及direct这两个概念开始在SLAM中火了起来（其实这两个概念在SfM中可以追溯到很久之前），dense指的是尽量多的利用image信息，direct是相对应于feature matching/projection的direct image alignment方法（通常feature-based methods是最小化reprojection error，而direct image alignment methods中是最小化photometric error）。由于DTAM的计算量有点大，在GTX 480 GPU上处理640x480视频流勉强可以到30fps，也诞生了一些做了trade-off能达到更快的计算速度的方法，比如[semi-dense VO [10]](http://vision.in.tum.de/_media/spezial/bib/engel2013iccv.pdf)，[semi-direct VO (SVO) [11]](http://rpg.ifi.uzh.ch/docs/ICRA14_Forster.pdf)等。

Jakob Engel在ICCV 2013提出的[semi-dense VO [10]](http://vision.in.tum.de/_media/spezial/bib/engel2013iccv.pdf)方法其实就是最近很火的[LSD-SLAM](https://github.com/tum-vision/lsd_slam)的前身。这个算法基本是将他们组之前基于RGBD camera的[DVO](https://github.com/tum-vision/dvo_slam)思想用到了monocular camera中。算法思想如下：算法仍然分为两个独立的线程分别做tracking和mapping，假设已知前两帧图像对应的depth map（这个初始的depth map可以通过传统的correspondence方法计算或者初始化为随机值），tracking线程通过最小化photometric error来求解相机姿态（对于已有depth map的图像I1中每个有depth d的点x，通过未知姿态参数P，可以求出其在新一帧图像I2中相应的位置x~，然后该点的photometric error即为||I1(x)−I2(x~)||），mapping则通过类似于Kalman filter的机制来predict-update每个点的depth及其uncertainty（其中predict step是通过estimated pose来propagate到下一帧对应点的depth，update step是基于每个点选择不同的reference frame匹配并计算depth，注意其中uncertainty的propagation和update比较有趣，包括observe时对于geometric uncertainty和photometric uncertainty的引入）。该VO算法还被作者移植到手机上做AR，在一篇[ISMAR paper](http://vision.in.tum.de/_media/spezial/bib/schoeps14ismar.pdf)中能找到，不过从[演示视频](https://www.youtube.com/watch?v=X0hx2vxxTMg)中看来感觉有点卡顿和抖动。

[LSD-SLAM [12]](http://vision.in.tum.de/_media/spezial/bib/engel14eccv.pdf)就是在上面这个VO算法基础上，加入keyframe的管理及优化、loop closure等机制使得上面的VO算法真正成为一个完整的SLAM系统。在LSD-SLAM中：

地图是由一系列keyframe（存储semi-dense的depth map和depth variance map）以及keyframes之间的Sim(3)变换组成

在对新来的一帧图像做tracking时，用上面的VO算法从最近的keyframe估计其SE(3) pose

当需要创建新keyframe时，用上面的VO算法估计其depth map及depth variance map

当把keyframe加入地图时，估计该keyframe与其他keyframe之间的Sim(3)变换

用[OpenFABMAP](https://github.com/arrenglover/openfabmap)做loop detection

值得注意的几个细节是：

在tracking的photometric error中，用intensity variance及depth variance来做normalize

每个keyframe存储depth map时将其归一化（mean inverse depth为1），在keyframe之间的edge中存储scale的关系（即Sim(3)变换）

在estimate keyframe to keyframe的Sim(3)变换时，需要在cost function中引入depth error来对scale做constraint

关于LSD-SLAM的算法解析和代码详解，网上可以找到很多文章，这里就不再啰嗦。

算法优点：对map uncertainty的model比较好；对于low texutre场景应该比较鲁棒；可以建相对比较dense的地图

算法缺点：算法目前计算量还是有点大，用i7的CPU勉强可以达到实时；从AR的效果看似乎抖动比较明显，似乎精度仍不如ORB-SLAM

## SVO (2014) [[11]](http://rpg.ifi.uzh.ch/docs/ICRA14_Forster.pdf) ([code available](https://github.com/uzh-rpg/rpg_svo))

[SVO (semi-direct VO) [11]](http://rpg.ifi.uzh.ch/docs/ICRA14_Forster.pdf)是另一个由direct method催生的VO算法，注意跟上述的[semi-dense VO [10]](http://vision.in.tum.de/_media/spezial/bib/engel2013iccv.pdf)名字很像，但其实是个很不一样的算法。SVO是一种hybrid的方法，结合了direct method和feature-based method的成分，算法虽然依赖于feature correpondence，但是避免了显式计算feature matching和outlier handling，因此也节约了不少无谓的计算量，所以算法的速度可以做到非常快，在嵌入式的平台ARM Cortex A9 1.6GHz CPU上处理752x480的frame能达到55fps（在laptop的Intel i7 2.8GHz CPU上可以高达300fps）。算法依然是分为两个独立的线程分别做tracking和mapping，下面分别介绍。

SVO的mapping也是采用了probabilistic model，跟LSD-SLAM不同的是，SVO的depth filter是用的Gaussian distribution + Uniform distribution的mixture model，而LSD-SLAM是基于Gaussian distribution的model（类似于Kalman filter）。SVO的depth filter算法在[paper [19]](http://george-vogiatzis.org/publications/ivcj2010.pdf)中可以找到详细推导过程。SVO的mapping线程流程比较简单，主要算法在updateSeeds里：

MapInit[类似PTAM的bootstrap：计算homography并triangulate 3D map points，前两个frame作为keyframe创建depth filter seeds]UpdateSeeds[每一个frame都会调用updateSeeds来更新已有keyframe里的depth filter seeds，通过在epipolar line上搜索匹配patch来做data association，如果seed的uncertainty足够小就创建新的3D map point]InitializeSeeds[如果是keyframe，需要另外初始化一些depth filter seeds：在image pyramid每一层的Shi-Tomasi score比较高的FAST corner位置初始化depth filter seed，这些seeds的depth设置成scene mean depth，variance设置成比较大的值]WaitForNewFrame[##############等待下一个frame coming##############]

SVO的tracking比较有趣，流程如下：

SparseImgAlign[通过类似于LSD-SLAM中的直接最小化photometric error方法求出current frame相对于previous frame的姿态，其中cost function里的data term是previous frame中已知depth的点投影到current frame，用点附近的4x4的small patch计算photometric error]ReprojectMapAndAlignFeature[将之前keyframes中的地图点都投影到current frame，用LK算法search每个点更精确的subpixel匹配位置，注意这里并不限制在epipolar line上，用点附近8x8的affine warped patch计算LK的error]OptimizePose [motion-only BA][用跟PTAM类似的最小化reprojection error的方法优化pose]OptimizeMapPoints [structure-only BA][用最小化reprojection error的方法优化3D map points]BundleAdjustRecent [full BA, optional][最后做一个local BA一起优化pose和3D map points]AssessTrackingQuality[评估tracking的质量，作为判断是否跟踪丢失或插入关键帧的依据]

由上可以看到SVO的tracking中没有用到motion model，在第一步的alignment中用到了direct method中的方法求pose，第二步中用了经典的optical flow算法将匹配计算到subpixel精度，第三步后面又用了传统SLAM算法的最小化reprojection error的方法去优化。整个过程去掉了显式的feature matching和RANSAC类的outlier handling算法，很高效。

SVO paper中有几个有趣的讨论值得注意一下：

Tracking为何不只使用第一步的算法：因为第二步其实是用地图中更多的map points来做tracking，会减少drift，类似于ORB-SLAM中的TrackLocalMap的步骤

Tracking为何不直接从其第二步开始，跳过第一步：虽然第二步的LK算法可以单独作为tracking的算法，但是第一步的sparse direct image alignment的作用是可以减少LK算法的搜索范围并大量减少匹配的outlier，从而提高了搜索速度和省去了RANSAC的计算过程

由于depth filter的使用，tracking时使用的map points是已经converge的比较可靠的点，所以每帧可以使用比PTAM更少的点来做tracking，从而速度更快

另外，mapping线程基本只做depth filter seed update，在seed较少时速度可以很快，甚至比tracking速度都快，所以可以每一帧都送到mapping线程进行处理

[SVO的代码](https://github.com/uzh-rpg/rpg_svo)在Github上可以找到，代码不多比较容易读，入口可以在Frame\_handler\_mono.cpp中找到，tracking的主要代码在processFrame函数中，mapping的主要代码在Depth\_filter.cpp中的updateSeeds。

算法优点：速度快；对map uncertainty的model比较好；对运动模型没有假设

算法缺点：由于track的feature比较少，有些情况下会丢失

## DSO (2016) [[13]](http://vision.in.tum.de/_media/spezial/bib/engel2016dso.pdf) ([code available](https://github.com/JakobEngel/dso))

[DSO [13]](http://vision.in.tum.de/_media/spezial/bib/engel2016dso.pdf)是LSD-SLAM的作者Jakob Engel最近放出的另一个大杀器，从其[展示的实验结果](https://www.youtube.com/watch?v=C6-xwSOOdqQ)看，无论是robustness，或是accuracy，或是计算速度，都完爆LSD-SLAM和ORB-SLAM，上个月又放出了[code](https://github.com/JakobEngel/dso)。从其paper上看，DSO比LSD-SLAM提高的主要原因有以下几个方面：

受[OKVIS [16]](https://spiral.imperial.ac.uk/bitstream/10044/1/23413/2/ijrr2014_revision_1.pdf)的启发，将LSD-SLAM中的depth map estimation从Kalman filter替换成bounded window optimization（仍然主要是photometric error minimization），这样使得depth的估计更加精确，个人认为这个是精度提高的主要原因

在上述的optimization中，加入了更多的参数一起优化，除了camera pose和point depth value之外，还包括camera intrinsics以及考虑曝光时间等因素的affine brightness transfer parameters （当optimization的keyframes之间距离比较远时，曝光参数的不同可能会影响photometric error）

设计了一套比较fancy的point selection的机制，相对于LSD-SLAM使用了更少的点，采样更sparse，这样使得计算量减少很多，另外也加入了显式的outlier detection机制

采用了类似于ORB-SLAM的更先进的keyframe管理机制，另外也采用了类似OKVIS的marginalization机制

在track新来的一帧时，虽然跟LSD-SLAM一样也是从最近的一个keyframe来estimate pose，但是在此之前将所有该keyframe可以看到的active points都project过来，这样增多了可以用来track新frame的线索

考虑了camera的photometric calibration因素，比如lens vignetting以及nonlinear response function等，由于这些因素会直接对photometric error产生影响

需要注意的是，上面的optimization只是针对keyframes做的，并不是每一个新来的frame都会做。另外，在paper中对于direct methods和feature-based/indirect methods之间的对比noise实验比较有趣：direct methods对geometric noise比较敏感，比如rolling shutter等；而indirect methods对photometric noise比较敏感，比如blur等。这点其实说明，在普通的智能手机上（一般采用rolling shutter camera），基于PTAM类的算法可能效果更好；而在基于global shutter的机器人相机中，direct methods可能越来越受欢迎。

算法的计算速度上来看，正常设置可以在Intel i7-4910MQ CPU上处理640x480图像达到realtime的速度，参数低配时可以处理424x320图像达到5倍realtime的速度（不知道具体多少，150fps?）。

算法优点：精度高，鲁棒性好，drift小，适用于大规模SLAM

算法缺点：计算量大；对rolling shutter比较敏感；由于optimization不是针对每个frame都做，而只是针对keyframes做，估计用在AR中会出现类似于LSD-SLAM的卡顿，因为非关键帧的pose估计不如关键帧的准确（TODO： 这点没有实测，只是猜测，有待考证）

名字缩写太多，我有点凌乱了，做个区分：

[DVO](https://github.com/tum-vision/dvo_slam): TUM的基于RGBD camera的VO方法

[SVO](https://github.com/uzh-rpg/rpg_svo): Gatech的基于semi-direct的hybrid VO方法

[DSO](https://github.com/JakobEngel/dso): TUM的direct sparse VO方法

# 三、Visual-Inertial Odometry算法笔记

使用monocular camera + IMU的方案来做SLAM/Odometry，一般被称作Visual-Inertial Odometry (VIO)或者Visual-Inertial Navigation System (VINS)。这一类paper大多出自Robotics社区，主要focus在如何更好的在Visual SLAM中融合IMU数据。IMU数据不单可以帮助resolve单目的scale ambiguity，一般情况下还可以提高SLAM的精度和鲁棒性。需要注意的是，想要从IMU数据获得准确的姿态没那么容易，一般需要做sensor fusion，从经典的[complementary filter](http://www.pieter-jan.com/node/11)做gyroscope、accelerometer、magnetometer的融合，再到[Mahony filter](http://www.olliw.eu/2013/imu-data-fusing/)等更复杂的融合算法，有很多可以选择的算法，其精度和复杂度也各不相同。现在的Android系统里一般可以直接获得手机姿态，至于其中用了哪种融合算法本人还没有仔细研究过，精度也有待考察。在Robotics社区的VIO paper中，一般是直接用原始的IMU数据或者经过简单滤波的数据，一般需要对IMU的bias进行建模（尤其在MEMS IMU中，所谓的零飘和溫飘对精度影响很大，有些要求比较高的情况下甚至需要将其置于恒温状态工作）。

## MSCKF (2007-2013) [[14](http://www.ee.ucr.edu/~mourikis/papers/MourikisRoumeliotis-ICRA07.pdf),[15](http://www.ee.ucr.edu/~mourikis/papers/Li2013IJRR.pdf)]

基于Kalman filter的MSCKF跟EKF-based SLAM一样也是出自Robotics社区，从[MSCKF 1.0 [14]](http://www.ee.ucr.edu/~mourikis/papers/MourikisRoumeliotis-ICRA07.pdf)到[MSCKF 2.0 [15]](http://www.ee.ucr.edu/~mourikis/papers/Li2013IJRR.pdf)，精度得到了不错的提高，据说[Google Project Tango](https://developers.google.com/tango/apis/c/c-motion-tracking)中的SLAM算法就是用的MSCKF算法。

传统的EKF-based SLAM做IMU融合时，跟前面介绍的MonoSLAM类似，一般是每个时刻的state vector保存当前的pose、velocity、以及3D map points坐标等（IMU融合时一般还会加入IMU的bias），然后用IMU做predict step，再用image frame中观测3D map points的观测误差做update step。MSCKF的motivation是，EKF的每次update step是基于3D map points在单帧frame里观测的，如果能基于其在多帧中的观测效果应该会好（有点类似于local bundle adjustment的思想）。所以MSCKF的改进如下：predict step跟EKF一样，但是将update step推迟到某一个3D map point在多个frame中观测之后进行计算，在update之前每接收到一个frame，只是将state vector扩充并加入当前frame的pose estimate。这个思想基本类似于local bundle adjustment（或者sliding window smoothing），在update step时，相当于基于多次观测同时优化pose和3D map point。具体细节可以参考paper [[15]](http://www.ee.ucr.edu/~mourikis/papers/Li2013IJRR.pdf)。

## OKVIS (2013-2014) [[16]](https://spiral.imperial.ac.uk/bitstream/10044/1/23413/2/ijrr2014_revision_1.pdf) ([code available](https://github.com/ethz-asl/okvis))

相对应于MSCKF的filter-based SLAM派系，OKVIS是keyframe-based SLAM派系做visual-inertial sensor fusion的代表。从MSCKF的思想基本可以猜出，OKVIS是将image观测和imu观测显式formulate成优化问题，一起去优化求解pose和3D map point。的确如此，OKVIS的优化目标函数包括一个reprojection error term和一个imu integration error term，其中已知的观测数据是每两帧之间的feature matching以及这两帧之间的所有imu采样数据的积分（注意imu采样频率一般高于视频frame rate），待求的是camera pose和3D map point，优化针对的是一个bounded window内的frames（包括最近的几个frames和几个keyframes）。

需要注意的是，在这个optimization problem中，对uncertainty的建模还是蛮复杂的。首先是对imu的gyro和accelerometer的bias都需要建模，并在积分的过程中将uncertainty也积分，所以推导两帧之间的imu integration error时，需要用类似于Kalman filter中predict step里的uncertainty propagation方式去计算covariance。另外，imu的kinematics微分方程也是挺多数学公式，这又涉及到[捷联惯性导航(strapdown inertial navigation)](https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf)中相关的很多知识，推导起来不是很容易。这可以另起一个topic去学习了。

OKVIS使用keyframe的motivation是，由于optimization算法速度的限制，优化不能针对太多frames一起，所以尽量把一些信息量少的frames给marginalization掉，只留下一些keyframes之间的constraints。关于marginalization的机制也挺有趣，具体参见paper [[16]](https://spiral.imperial.ac.uk/bitstream/10044/1/23413/2/ijrr2014_revision_1.pdf)。

ETH Zurich的ASL组另外有一篇基于EKF的VIO paper，叫[ROVIO [17]](http://e-collection.library.ethz.ch/eserv/eth:48374/eth-48374-01.pdf)，也有[code](https://github.com/ethz-asl/rovio)，具体还没细看，听说鲁棒性不错。

## IMU Preintegration (2015-2016) [[18]](http://www.cc.gatech.edu/~dellaert/pubs/Forster16tro.pdf) ([code available in GTSAM 4.0](https://bitbucket.org/gtborg/gtsam))

从OKVIS的算法思想中可以看出，在优化的目标函数中，两个视频帧之间的多个imu采样数据被积分成一个constraint，这样可以减少求解optimization的次数。然而OKVIS中的imu积分是基于前一个视频帧的estimated pose，这样在进行optimization迭代求解时，当这个estimated pose发生变化时，需要重新进行imu积分。为了加速计算，这自然而然可以想到imu preintegraion的方案，也就是将imu积分得到一个不依赖于前一个视频帧estimated pose的constraint。当然与之而来的还有如何将uncertainty也做类似的propagation（考虑imu的bias建模），以及如何计算在optimization过程中需要的Jacobians。相关的推导和理论在[paper [18]](http://www.cc.gatech.edu/~dellaert/pubs/Forster16tro.pdf)中有详细的过程。在OKVIS的代码[ImuError.cpp](https://github.com/ethz-asl/okvis/tree/master/okvis_ceres/src)和GTSAM 4.0的代码[ManifoldPreintegration.cpp](https://bitbucket.org/gtborg/gtsam/src/c21186c6212798e665da6b5015296713ddfe8c1d/gtsam/navigation/ManifoldPreintegration.cpp?at=master&fileviewer=file-view-default)中可以分别看到对应的代码。

[1]. David Nister, Oleg Naroditsky, and James Bergen. [Visual Odometry](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.5767&rep=rep1&type=pdf). CVPR 2004.

[2]. Andrew Davison, Ian Reid, Nicholas Molton, and Olivier Stasse. [MonoSLAM: Real-time single camera SLAM](https://www.doc.ic.ac.uk/~ajd/Publications/davison_etal_pami2007.pdf). TPAMI 2007.

[3]. Ethan Eade and Tom Drummond. [Monocular SLAM as a Graph of Coalesced Observations](http://ethaneade.com/eadeICCV2007.pdf). ICCV 2007.

[4]. Georg Klein and David Murray. [Parallel Tracking and Mapping for Small AR Workspaces](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2007ISMAR.pdf). ISMAR 2007.

[5]. Georg Klein and David Murray. [Improving the Agility of Keyframe-based SLAM](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2008ECCV.pdf). ECCV 2008.

[6]. Georg Klein and David Murray. [Parallel Tracking and Mapping on a Camera Phone](http://www.robots.ox.ac.uk/~gk/publications/KleinMurray2009ISMAR.pdf). ISMAR 2009.

[7]. Hauke Strasdat, J.M.M. Montiel, and Andrew Davison. [Visual SLAM: Why Filter?](https://www.doc.ic.ac.uk/~ajd/Publications/strasdat_etal_ivc2012.pdf). Image and Vision Computing 2012.

[8]. Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. [ORB-SLAM: A Versatile and Accurate Monocular SLAM System](http://webdiis.unizar.es/~raulmur/MurMontielTardosTRO15.pdf). IEEE Transactions on Robotics 2015.

[9]. Richard Newcombe, Steven Lovegrove, and Andrew Davison. [DTAM: Dense Tracking and Mapping in Real-Time](https://www.robots.ox.ac.uk/~vgg/rg/papers/newcombe_davison__2011__dtam.pdf). ICCV 2011.

[10]. Jakob Engel, Jurgen Sturm, and Daniel Cremers. [Semi-Dense Visual Odometry for a Monocular Camera](http://vision.in.tum.de/_media/spezial/bib/engel2013iccv.pdf). ICCV 2013.

[11]. Christian Forster, Matia Pizzoli, and Davide Scaramuzza. [SVO: Fast Semi-Direct Monocular Visual Odometry](http://rpg.ifi.uzh.ch/docs/ICRA14_Forster.pdf). ICRA 2014.

[12]. Jakob Engel, Thomas Schops, and Daniel Cremers. [LSD-SLAM: Large-Scale Direct Monocular SLAM](http://vision.in.tum.de/_media/spezial/bib/engel14eccv.pdf). ECCV 2014.

[13]. Jakob Engel, Vladlen Koltun, and Daniel Cremers. [Direct Sparse Odometry](http://vision.in.tum.de/_media/spezial/bib/engel2016dso.pdf). In arXiv:1607.02565, 2016.

[14]. Anastasios Mourikis, Stergios Roumeliotis. [A multi-state constraint Kalman filter for vision-aided inertial navigation](http://www.ee.ucr.edu/~mourikis/papers/MourikisRoumeliotis-ICRA07.pdf). ICRA 2007.

[15]. Mingyang Li, Anastasios Mourikis. [High-Precision, Consistent EKF-based Visual-Inertial Odometry](http://www.ee.ucr.edu/~mourikis/papers/Li2013IJRR.pdf). International Journal of Robotics Research 2013.

[16]. Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Timothy Furgale. [Keyframe-based visual–inertial odometry using nonlinear optimization](https://spiral.imperial.ac.uk/bitstream/10044/1/23413/2/ijrr2014_revision_1.pdf). The International Journal of Robotics Research 2014.

[17]. Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. [Robust Visual Inertial Odometry Using a Direct EKF-Based Approach](http://e-collection.library.ethz.ch/eserv/eth:48374/eth-48374-01.pdf). IROS 2015.

[18]. Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. [On-Manifold Preintegration for Real-Time Visual-Inertial Odometry](http://www.cc.gatech.edu/~dellaert/pubs/Forster16tro.pdf). IEEE Transactions on Robotics 2016.

[19]. George Vogiatzis, Carlos Hernandez. [Video-based, Real-Time Multi View Stereo](http://george-vogiatzis.org/publications/ivcj2010.pdf). Image and Vision Computing 2011. ([Supplementary material](http://george-vogiatzis.org/publications/ivcj2010supp.pdf))