论文集来源

https://zhuanlan.zhihu.com/p/115599978/ 吴艳敏

传统SLAM： 特征点、半直接、直接法

Feasure_based

S-PTAM(双目 PTAM)

S-PTAM: Stereo Parallel Tracking and Mapping

摘要：

1.对多种环境鲁棒：室内、室外、动态物体、光环境、大回环、高速运动

2.对多种Feature和Descriptor的测试。

3.一种新的初始化方法

内容：

1.特征点测试了多种

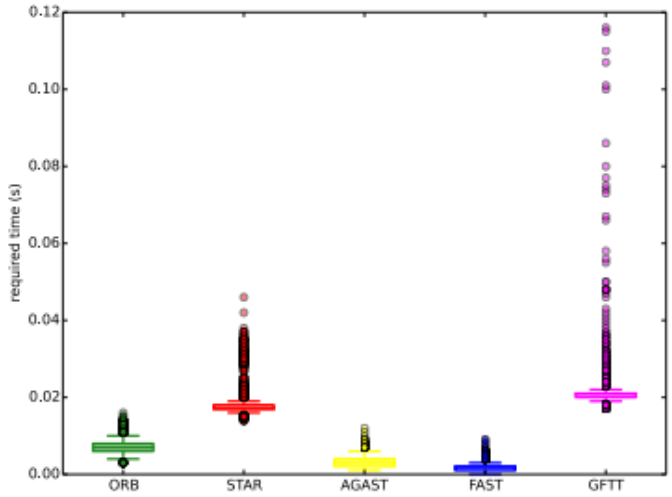
2.DeadReckon+PnP 前端 ScanToMap

3.Local Map BA ScanToMap

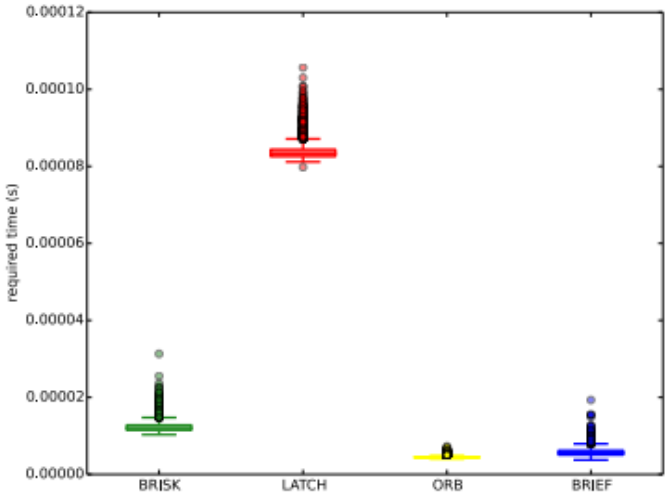
4.回环BoWds(Apperance) + P3P+内点数量检测(Geometric Verification)

5.相关实验结果：

Feature 和 Descriptor 计算时间

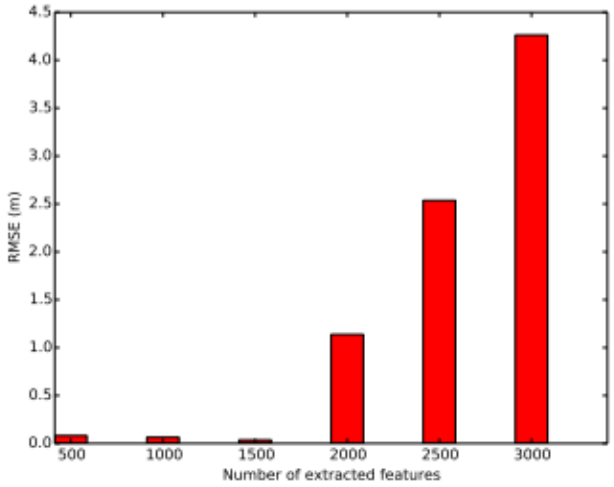


(a) Feature detection time for each image

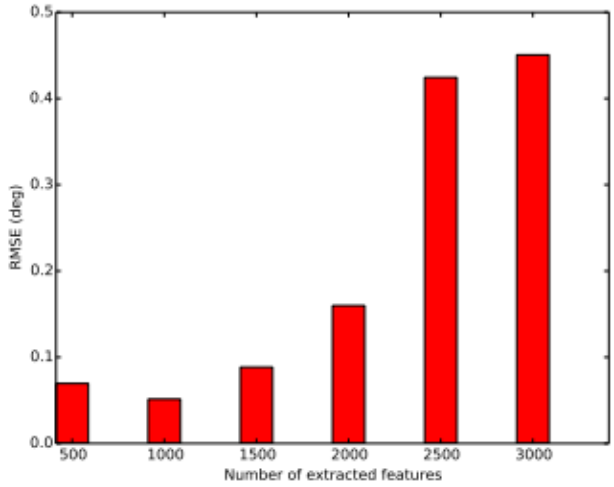


(b) Descriptor extraction time for each feature

Error 和 特征提取量的关系



(a) Translation RMSE errors.



(b) Rotation RMSE errors.

总结：

中规中距的SLAM算法，对于所说的鲁棒性没有看到源码前，表示怀疑。

ORB 的提取速度已经意外的快了。

令人意外的是特征点的数量并不是越多越好，怀疑是引入误匹配的原因。

开源：

https://github.com/lrse/sptam

ARM-VO (Mono)

ARM-VO: an efcient monocular visual odometry for ground vehicles on ARM CPUs

摘要：

在Rasperry 3B上，可以实时跑的VO。

内容：

a.并行网格Fast特征提取

b.并行KLT追踪

c.特征点矫正

d.F和H+RANSAC+GRIC1 这里选的是一个rough H

e.H的快速estimation[2]

f.多线程RT验证

g.Scale resolving 没看懂

实现：

a.KLT 用 OpenCV 有Neon C 加速 TBB

b.FAST 从 OpenCV 的SSE2 换成NEON Fast Threshold 是15 TBB

c.H 计算和check OpenMP

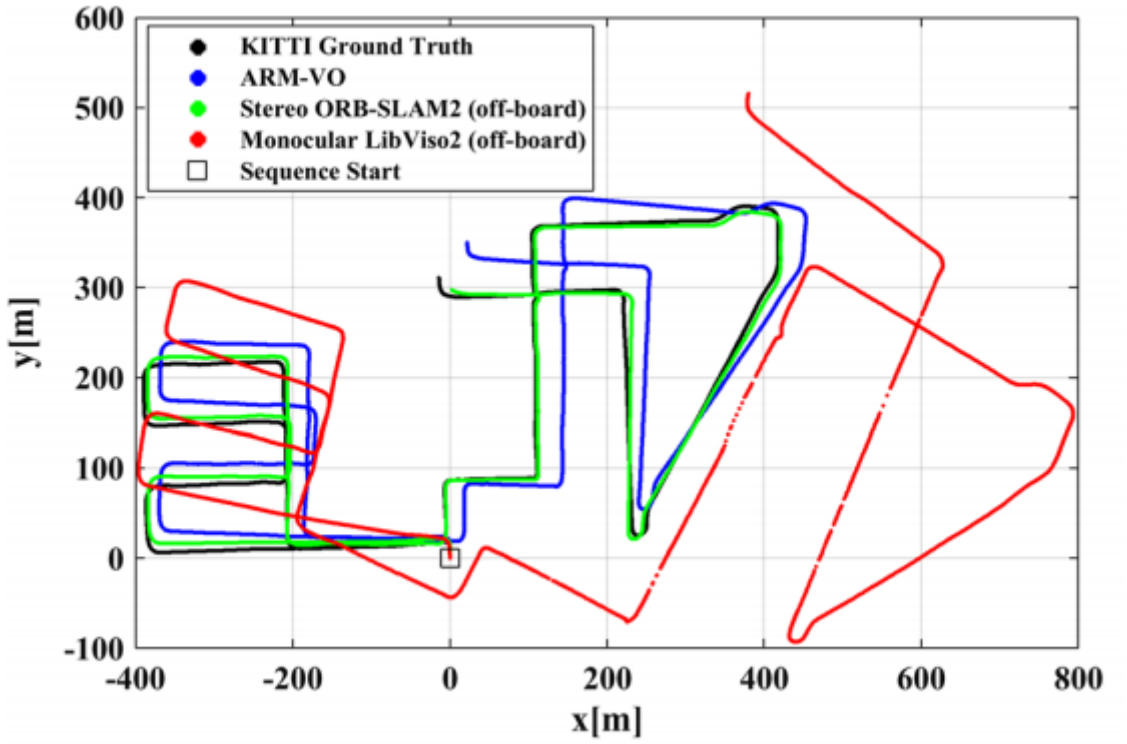
d.对OpenCV源代码进行了更改,删除 check 和 double->float

效果：

Table 3 The performance of each method on Raspberry Pi 3 represented by average FPS

	ARM-VO	LibViso2	ORB-SLAM2
Avg. FPS	8	2	1.5

The image resolution is 1392×512. LibViso2 and ORB-SLAM2 lose track because of large inter-frame motion induced by low processing speed



引用：

[1]GRIC：An assessment of information criteria for motion model selection. In: 1997 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings. IEEE

[2]Márquez-Neila, P., et al.: Speeding-up homography estimation in mobile devices. J. Real Time Image Process.

总结：

速度快，但是精度有限。GRIC用于选择HF模型需要看一下

开源：

https://github.com/zanazakaryaie/ARM-VO

Semi-Direct

SVO/SVO2 (Mono/Stereo、SVO2 有机会精读一下)

SVO: fast semi-direct monocular visual odometry SVO

SVO: Semi-Direct Visual Odometry for Monocular
and Multi-Camera Systems

摘要：

SVO1 FASTFeature + PLK

1. 后端的深度估计器
   SVO2 比SVO1 添加了
2. 一个edgelet 的特征，但还是继续使用光流进行跟踪
3. IMU Piror 误差项

内容：

a. 栅格FAST提取 (网友说有提取策略有一些问题)

b. Direct法 构建光度误差进行匹配(网友说初始化有光流)

c. FrameToFrame 之后 FrameToMap (一个inverse_xxx 的操作 计算Jacobian加速)

d. 因为单木的尺度未知，在地图线程有一个深度滤波(难点，网友说深度不容易收敛)

开源:

https://github.com/HeYijia/svo_edgelet

贺一家修改的SVO 版本 添加 Edgelet

https://github.com/uzh-rpg/rpg_svo_example

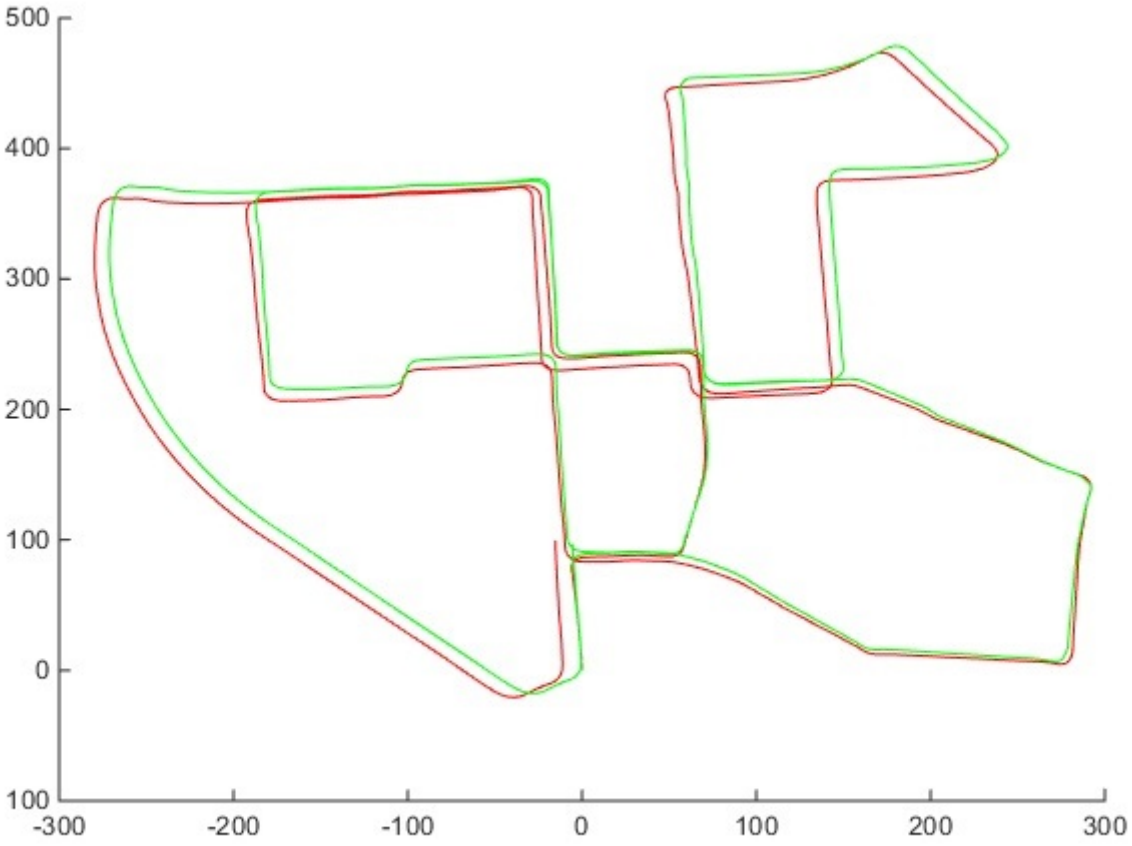
用于SVO2.0 的使用 SVO2.0 只有Binary文件

https://github.com/uzh-rpg/rpg_svo

SVO1.0 阉割版源码

Tips:

网友修改过后的SVO1.0 还是有潜力的



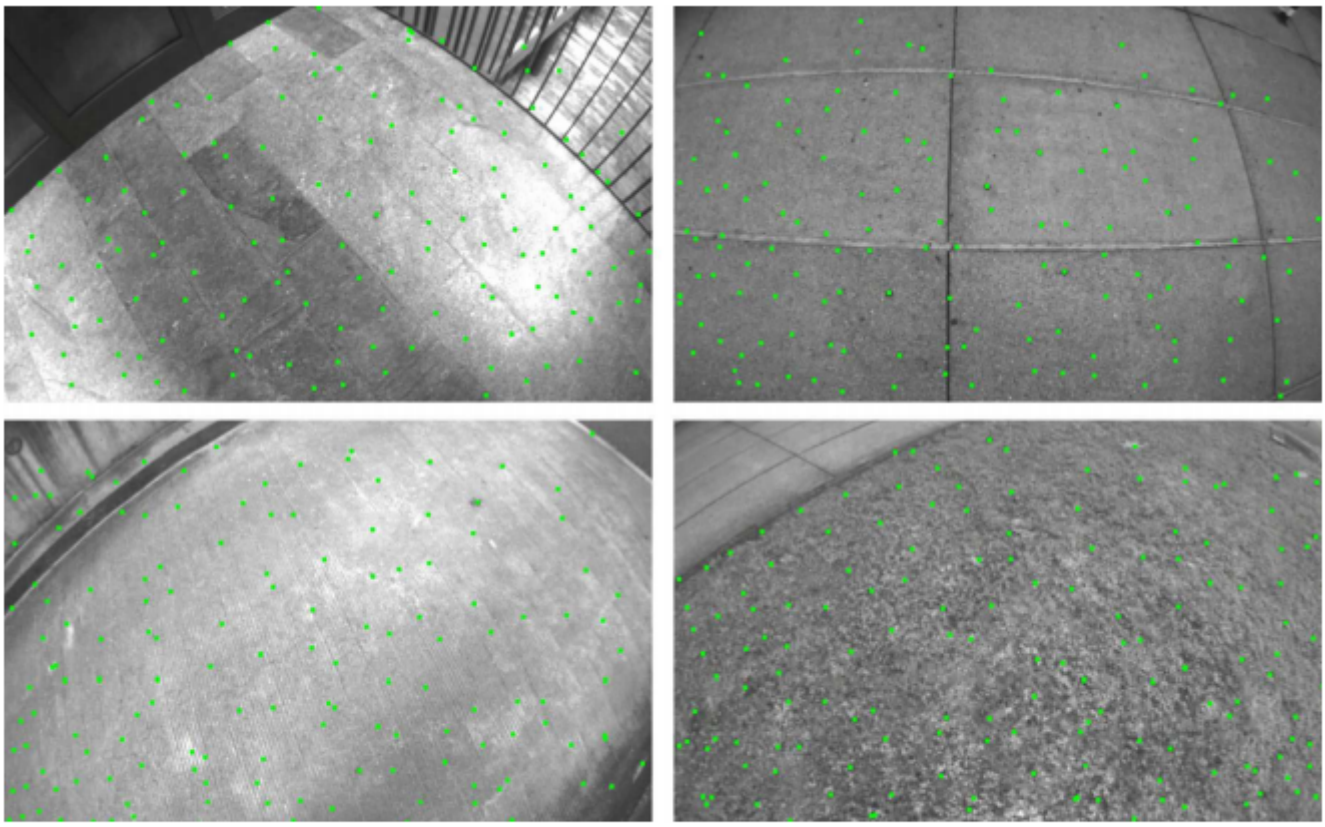


Fig. 15: Successful tracking in scenes of high-frequency texture.

1. 看论文效果感觉也不是完全的栅格提取
2. Mapping 的概率点更新的方式 需要进行修改

总结：

不知道SVO 的Mapping 线程的深度滤波是不是可以直接用Stereo DSO 的Stereo 误差项直接补掉

Direct

DSO(Mono)/LDSO (精读，之前不了解)

Direct Sparse Odometry
LDSO: Direct Sparse Odometry with Loop Closure
摘要：

感觉是添加了光度参数的直接法。

LDSO 则是使用词袋进行回环提取。

内容：

a.添加光度之后的直接法公式

光度误差可以用以下公式计算：

$$r = w_h \left(I_2 [x_2] - (a_{21} I_1 [x_1] + b_{21}) \right)$$

其中 w_h 是 Huber 权重，I_1, I_2 分别是影像1、2，x_1, x_2 分别是空间中一点 X 在影像上的像素坐标。

其中 x_2 可以写作

$$x_2 = f(x_1, \xi_{21}, \rho_1)$$

即 x_2 是由 x_1 投影而来，投影的过程中需要两帧之间的相对位姿 ξ_{21} 和 x_1 在 1 中的逆深度 ρ_1。

b.应对模糊的邻域选取

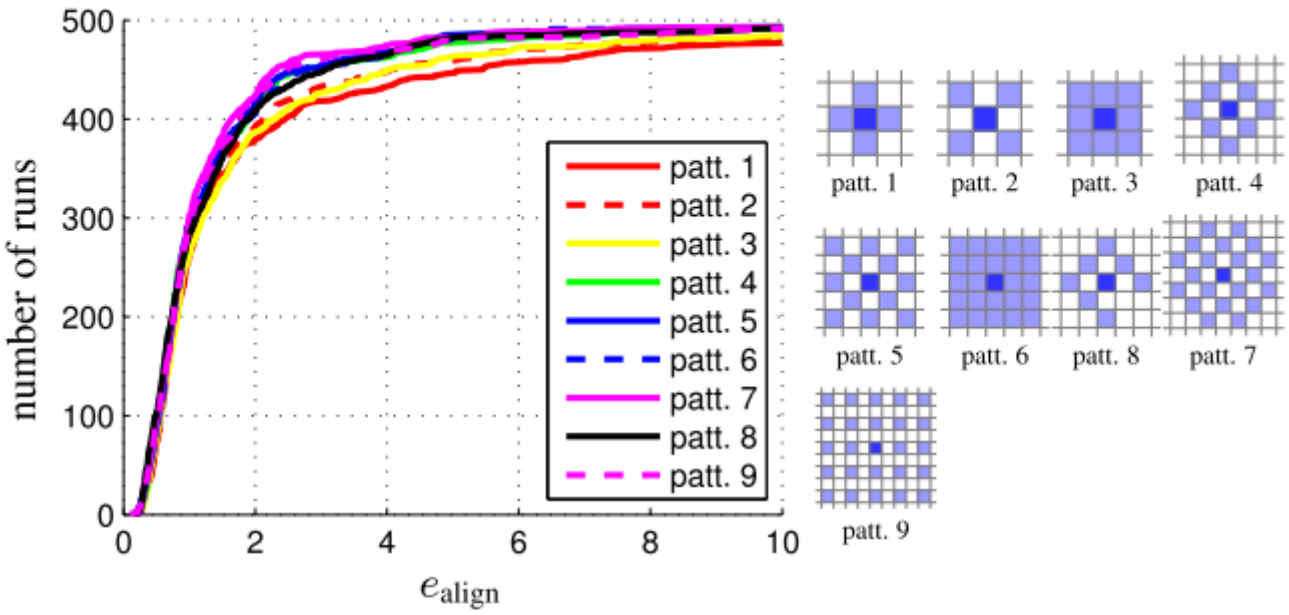


Fig. 19. *Residual pattern.* Errors on the TUM-monoVO dataset for some of the evaluated patterns $\mathcal{N}_\mathbf{p}$. Using only a 3×3 neighborhood seems to perform slightly worse—using more than the proposed 8-pixel pattern however seems to have little benefit—at the same time, using a larger neighbourhood increases the computational demands. Note that these results may vary with low-level properties of the used camera and lens, such as the point spread function.

最终使用Pattern 8 右下角的缺失主要是为了SSE加速的宽度对齐

c.active frame策略

效果：DSO/LDSO

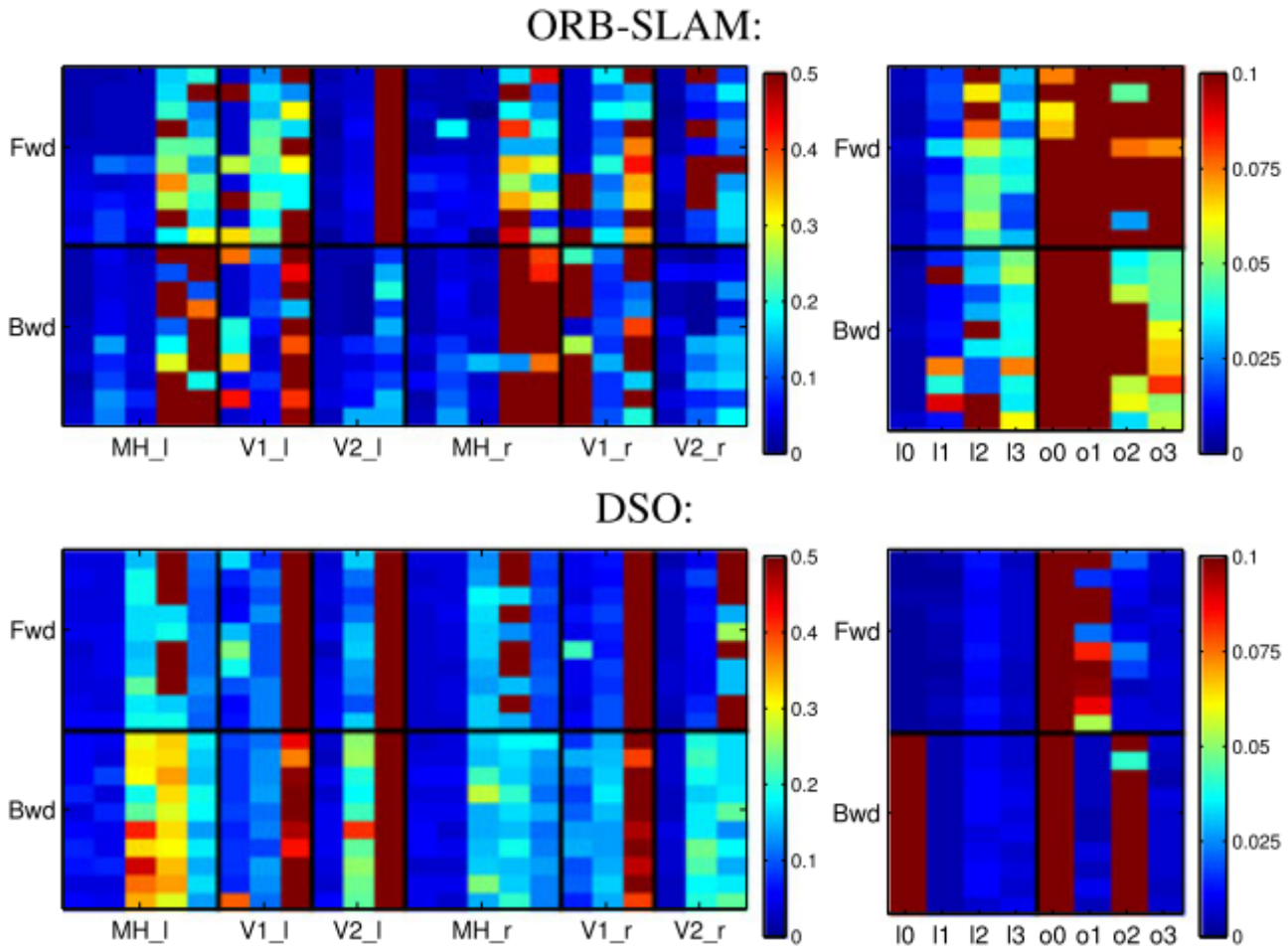


Fig. 13. *Full evaluation results.* All error values for the EuRoC MAV dataset (left) and the ICL_NUIM dataset (right): Each square corresponds to the (color-coded) absolute trajectory error e_{ate} over the full sequence. We run each of the 11 + 8 sequences (horizontal axis) forwards ("Fwd") and backwards ("Bwd"), 10 times each (vertical axis); for the EuRoC MAV dataset we further use the left and the right image stream. Fig. 10 shows these error values aggregated as cumulative error plot (bold, continuous lines).

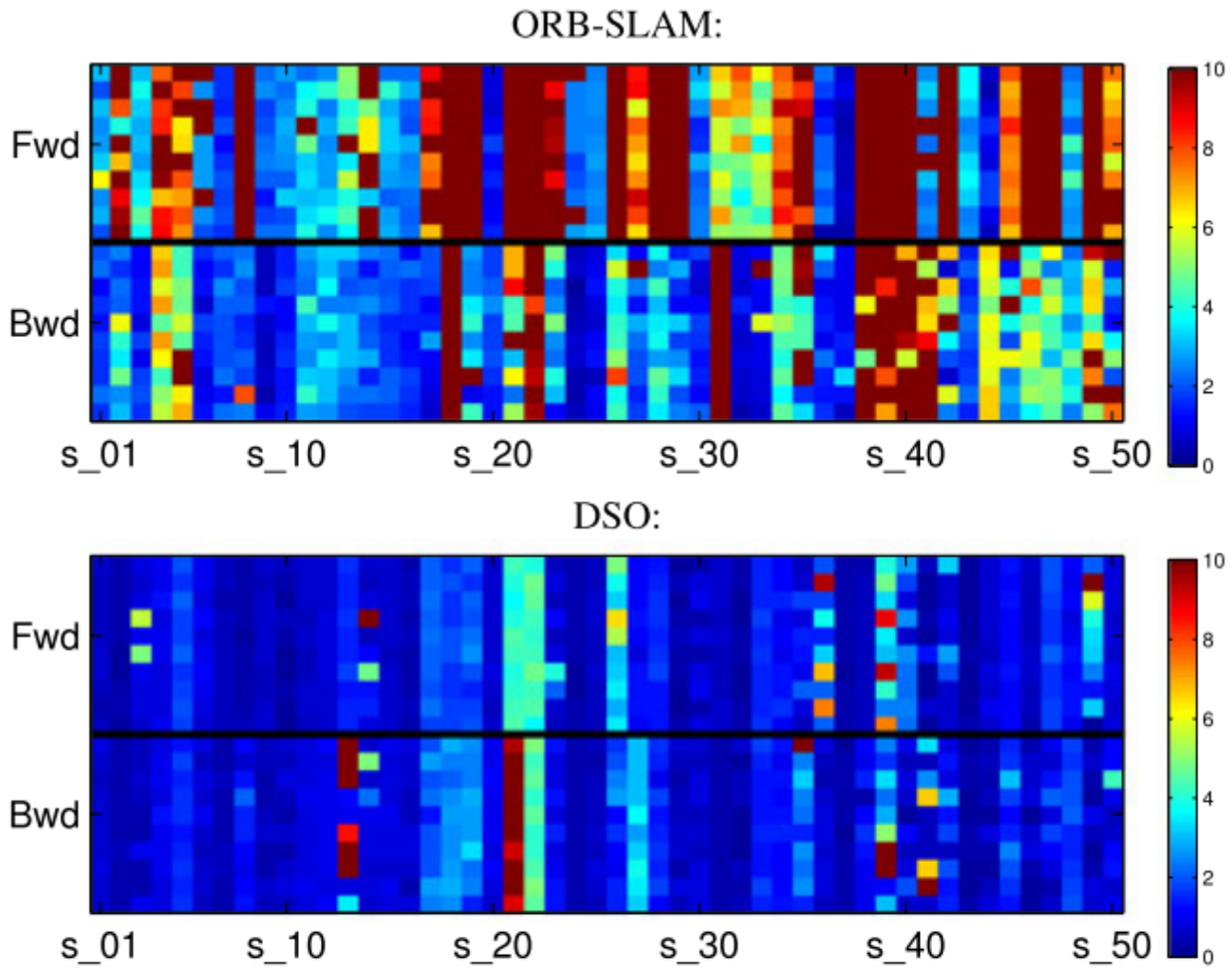


Fig. 14. *Full evaluation results.* All error values for the TUM-monoVO dataset (also see Fig. 11). Each square corresponds to the (color-coded) alignment error e_{align}, as defined in [8]. We run each of the 50 sequences (horizontal axis) forwards ("Fwd") and backwards ("Bwd"), 10 times each (vertical axis). Fig. 12 shows all these error values aggregated as cumulative error plot (bold, continuous lines).

在数据集TUM-monoVO dataset the synthetic ICL_NUIM dataset 比ORBSLAM2 表现更好，在EuRoc上不太行

原因如下：

a. TUM-monoVO 和 ICL-NUIM 有比较好的光度标定,EuRoc 没有这个标定

b. EuRoc 的小回环比较多，更适合SLAM 系统而不是一个里程计

Sequence	Mono DSO	LDSO	ORB-SLAM2
00	126.7	9.322	**8.27**
01	165.03	**11.68**	x
02	138.7	31.98	**26.86**
03	4.77	2.85	**1.21**
04	1.08	1.22	**0.77**
05	49.85	**5.1**	7.91
06	113.57	13.55	**12.54**
07	27.99	**2.96**	3.44
08	120.17	129.02	**46.81**
09	74.29	**21.64**	76.54
10	16.32	17.36	**6.61**

开源：

https://github.com/JakobEngel/dso

一个单目的版本 基本无法使用
双目版本 没有开源
https://github.com/tum-vision/LDSO

https://github.com/JiatianWu/stereo-dso

网友的Stereo 版本 使用这个版本跑Kiiti 在笔记本上帧率大概4-5帧/s

总结：

a. 精度：感觉精度一般，但是直接法对于处理低纹理区域肯定是有好处的。

b. 稳定性：
光度比ORBSLAM2 更加鲁棒

几何没有ORB_SLAM2 稳定

稳定性取决于初值是否精准，感觉对于VIO 来说，就涉及到了 IMU 和 图像帧率的问题

c. 速度：

d. 硬件平台：

最大的贡献在于，直接法以更整体、更优雅的方式处理了数据关联问题。

处理了特征点法在

环境中存在许多重复纹理；

环境中缺乏角点，出现许多边缘或光线变量不明显区域；的问题

但是感觉这个问题是不是光流处理掉的？

总体上讲，感觉更加符合对变量进行建模，进行估计的方法。对于光度进行建模是一个很好的想法。代码没有进行阅读的情况下,不知道是进行光度标定之后当作常数来使用，还是真的当作变量进行实时优化,有机会可以看看代码。

博客已经证实 是将两个光度参数 加入优化了。

那么新的问题就出现了，光度的概率模型如何确定？ 还是需要看看代码

为了防止运动模糊使用中心点邻域的方式。

Stereo DSO

Stereo DSO:
Large-Scale Direct Sparse Visual Odometry with Stereo Cameras

摘要：

1. 在实时优化的过程中添加 Stereo 对应的 ActivePoint 逆深度和光度参数估计

内容：

$$\begin{aligned}\boldsymbol{\xi} = (&\mathbf{T}_{0,\dots,N_f-1}, d_{0,\dots,N_p-1}, \mathbf{c},\\ &a^L_{0,\dots,N_f-1}, b^L_{0,\dots,N_f-1},\\ &a^R_{0,\dots,N_f-1}, b^R_{0,\dots,N_f-1}),\end{aligned} \tag{10}$$

$$E = \sum_{i \in \mathcal{F}} \sum_{\mathbf{p} \in \mathcal{P}_i} \Big(\sum_{j \in obs^t(\mathbf{p})} E_{ij}^{\mathbf{p}} + \lambda E_{is}^{\mathbf{p}} \Big), \qquad (15)$$

Static Stereo. For static stereo the residual is modified to

$$r_k^s = I_i^R[\mathbf{p}'(\mathbf{T}_{ji}, d, \mathbf{c})] - b_i^R - \frac{e^{a_i^R}}{e^{a_i^L}}(I_i[\mathbf{p}] - b_i^L). \qquad (14)$$

Temporal Multi-View Stereo. Each residual from temporal multi-view stereo is defined as

$$r_k^t = I_j[\mathbf{p}'(\mathbf{T}_i, \mathbf{T}_j, d, \mathbf{c})] - b_j - \frac{e^{a_j}}{e^{a_i}}(I_i[\mathbf{p}] - b_i). \qquad (12)$$

效果：

	St. DSO		ORB-SLAM2		St. LSD-VO	
Seq.	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
00	0.84	**0.26**	**0.83**	0.29	1.09	0.42
01	1.43	**0.09**	**1.38**	0.20	2.13	0.37
02	**0.78**	**0.21**	0.81	0.28	1.09	0.37
03	0.92	**0.16**	**0.71**	0.17	1.16	0.32
04	0.65	**0.15**	0.45	0.18	**0.42**	0.34
05	0.68	**0.19**	**0.64**	0.26	0.90	0.34
06	**0.67**	**0.20**	0.82	0.25	1.28	0.43
07	0.83	**0.36**	**0.78**	0.42	1.25	0.79
08	**0.98**	**0.25**	1.07	0.31	1.24	0.38
09	0.98	**0.18**	**0.82**	0.25	1.22	0.28
10	**0.49**	**0.18**	0.58	0.28	0.75	0.34
mean	0.84	0.20	0.81	0.26	1.14	0.40

Table 1. Comparison of accuracy on KITTI training set. t_{rel} translational RMSE (%), r_{rel} rotational RMSE (degree per $100m$). Both are average over $100m$ to $800m$ intervals. Best results are shown as bold numbers.

总结：

计算量大，但是准，问题是没有开源代码，只有一个开源三方库

DSM

Direct Sparse Mapping

摘要：

1. 利用光度信息的一致性地图？
2. 局部共视图来选取active frame 的策略
3. PBA 需要从粗到细的来进行收敛？ 有啥用？
4. 使用T分布来进行outlier 剔除

效果：

TABLE I: RMS ATE [m] using forward videos for left (l) and right (r) sequences. (×) means failure and (-) no available data.

Seq.	ORB-SLAM [7]	DSO [1]	LDSO [14]	DSM-SW	DSM	DSM (Global PBA)
MH1_l	0.070	0.046	0.053	0.054	**0.039**	0.042
MH2_l	0.066	0.046	0.062	0.041	**0.036**	0.035
MH3_l	0.071	0.172	0.114	0.123	**0.055**	0.040
MH4_l	0.081	3.810	0.152	0.179	**0.057**	0.055
MH5_l	**0.060**	0.110	0.085	0.139	0.067	0.054
V11_l	**0.015**	0.089	0.099	0.099	0.095	0.092
V12_l	**0.020**	0.107	0.087	0.124	0.059	0.060
V13_l	×	0.903	0.536	0.888	**0.076**	0.068
V21_l	**0.015**	0.044	0.066	0.061	0.056	0.060
V22_l	**0.017**	0.132	0.078	0.123	0.057	0.053
V23_l	×	1.152	×	1.081	**0.784**	0.681
MH1_r	-	**0.037**	0.050	0.054	0.045	0.039
MH2_r	-	0.041	0.051	0.039	**0.039**	0.034
MH3_r	-	0.159	0.095	0.187	**0.048**	0.035
MH4_r	-	3.045	0.129	0.188	**0.058**	0.052
MH5_r	-	0.092	0.087	0.131	**0.064**	0.052
V11_r	-	0.047	0.662	0.031	**0.014**	0.012
V12_r	-	0.080	0.208	0.118	**0.046**	0.043
V13_r	-	1.270	0.642	1.313	**0.045**	0.037
V21_r	-	**0.027**	0.040	0.032	0.034	0.030
V22_r	-	0.059	0.068	0.314	**0.057**	0.052
V23_r	-	0.540	**0.171**	0.889	0.528	0.482

TABLE II: Processing time and keyframe frecuency.

Operation	Median [ms]	Mean [ms]	St.D. [ms]
Frame & Point Tracking	7.44	7.45	0.31
Local PBA	888.77	908.53	121.10
Keyframe Period	396.28	397.22	177.51

开源：

https://github.com/jzubizarreta/dsm

视频效果看起啦 一个LocalPBA 需要 500 -1000ms 实时性堪忧

引用：

[1]Robust odometry estimation for
RGB-D cameras 2013

总结：

1. 没有仔细看 之后看
2. 这个地方可以再看看
3. 就是加了金字塔 DSO 没加吗？？
4. 就是在使用Huber核的基础上，对于RBGD相机来说在uv表示中，会更加合适感觉和DSO一样[1]
   如果光度误差在分布可信域的95%之内，就说明是。
   可信域是用图像中所有点的广度误差构建出来的
   感觉中规中矩,远没有大组的DSO 做的细致

scale_optimization 拓展DSO单目到双目

其实有一篇Stereo DSO 但是代码没有开源就不进行具体介绍
但是有一个三方库可以使用

Extending Monocular Visual Odometry
to
Stereo Camera Systems by Scale
Optimization

摘要：

1. 一种拓展DSO单目到双目的新方法
2. 在光度误差上添加scale 优化

内容：

1. 只是在误差项添加了一个Stereo(scale)项，将尺度信息引入

$$E_{scale} = \sum_{\mathbf{p} \in P} E_{\mathbf{p}scale} \tag{3}$$

$$E_{\mathbf{p}scale} = w_{\mathbf{p}} ||I_1[\Pi_1(\mathbf{T}_{stereo} \cdot s\Pi_0^{-1}(\mathbf{p}, d_{\mathbf{p}}))] - I_0[\mathbf{p}]||_\gamma \tag{4}$$

$$E_{photo} = \sum_{i \in F} \sum_{\mathbf{p} \in P_i} \sum_{j \in obs(\mathbf{p})} E_{\mathbf{p}j} \tag{1}$$

$$E_{\mathbf{p}j} = \sum_{\mathbf{p} \in N_{\mathbf{p}}} w_{\mathbf{p}} ||I_j[\Pi_0(\mathbf{R}\Pi_0^{-1}(\mathbf{p}, d_{\mathbf{p}}) + \mathbf{t})] - I_i[\mathbf{p}]||_\gamma \tag{2}$$

2. DSO最主要的光度还是没有考虑，感觉这个可能是速度快和为了双目模块化的原因

效果：

Seq.	t_{rel} (%)	r_{rel} (deg)	S.O. S.M. (ms)	BA (ms)	TPF (ms)	Pts
00	1.35	0.27	**2.25**	**124.07**	**141.95**	2164.28
	0.83	0.27	12.40	147.63	189.88	1658.76
01	2.72	0.13	**1.85**	**66.38**	**76.42**	1437.18
	1.78	**0.11**	10.75	73.32	123.27	1133.11
02	1.10	0.22	**2.23**	121.42	**164.16**	2019.06
	0.79	**0.21**	11.28	**111.71**	171.52	1426.85
03	3.17	**0.15**	**2.44**	115.32	**97.78**	2241.95
	1.01	0.16	11.34	**109.05**	109.47	1592.40
04	1.73	0.21	**2.09**	**87.40**	**122.44**	1926.45
	1.01	**0.19**	11.15	104.69	160.64	1599.01
05	1.69	0.20	**2.11**	**108.60**	**119.62**	2028.98
	0.82	**0.18**	11.20	113.93	145.99	1647.78
06	**1.66**	0.19	**2.09**	**85.05**	**110.03**	1718.27
	9.19	**0.17**	11.05	90.44	125.06	1270.94
07	2.50	**0.32**	**2.22**	**113.50**	**113.77**	2153.60
	1.03	0.33	10.66	119.33	134.84	1716.57
08	1.72	**0.26**	**2.08**	**109.24**	**126.58**	1945.50
	1.04	0.27	11.15	118.36	155.55	1497.82
09	1.88	0.22	**2.04**	**98.95**	**130.83**	1900.30
	0.98	**0.19**	11.22	102.23	150.79	1457.92
10	1.02	0.21	**1.89**	**88.84**	**102.70**	1775.74
	0.61	**0.19**	10.60	93.38	117.87	1357.89

Table 1: Error and run-time comparisons on the KITTI dataset. For each sequence, **the upper line is the result of SO-DSO, and the lower line is for Stereo DSO**. t_{rel} is translational RMSE(%); r_{rel} is rotational RMSE (degree per 100m). Results are averaged over $100m$ to $800m$ intervals. S.O. is the run-time of scale optimization; S.M. is the run-time of stereo matching; BA is the bundle adjustment run-time; TPF is the time per frame (not just keyframe); Pts is the number of 3D points in the bundle adjustment.

开源：

https://github.com/jiawei-mo/scale_optimization

总结：

方式简单粗暴，效果比放出来的DSO单目开源好的多，但是和没有开源的Stereo DSO 差距也还是很明显的

这个框架应该是还有优化空间的。目前看论文，感觉没有引入光度参数,应该更多的是为了双目模块化方面进行考虑

但是i7 -6 系的140ms 每帧的计算速度还是很让人头痛

和Stereo DSO 的不同是 Stereo 只是估计每一个frame的scale，但是Stereo 是在对每一个active Point 的逆深度进行估计，运算量更大。

BAD_SLAM(RGBD)

BAD SLAM: Bundle Adjusted Direct RGB-D SLAM

摘要：

总结

目前传统的内容看下来，主要的方向是

1. 引入新的变量模型 光度等
2. 使用新的边缘化策略
3. 讨论更合理的概率分布

VIO

MSCKF系列 (精读、比较适合嵌入式平台)

A Multi-State Constraint Kalman Filter
for Vision-aided Inertial Navigation (MSCKF1.0)

Improving the Accuracy of EKF-Based Visual-Inertial Odometry(MSCKF 2.0)

Robust Stereo Visual Inertial Odometry for Fast
Autonomous Flight(S-MSCKF)

OpenVINS(S-MSCKF的各种优化大杂烩)

摘要：

解决状态维数爆炸的情况(MSCKF1.0)
和别的VIO 框架一致，但是低算力，高稳定性(S-MSCKF)

内容：

1. 特征点+ PLK光流跟踪(内部作了不少优化)
2. 状态扩增(IMU预积分之后，EKF使用观测进行更新之前)
3. 状态更新:

    a. 对跟踪点进行三角化(优化方式的三角化)
    b. 通过这个三角化后的点，构建和各个滑床内frame 的重投影误差,来构建EKF 的更新方程
4. 边缘化

引用:

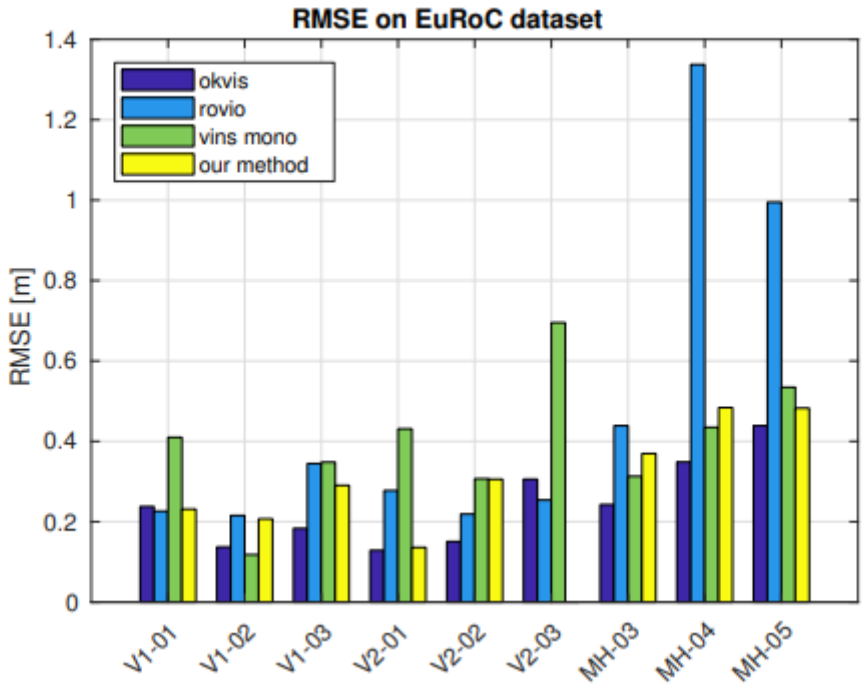
a. VIO 四个自由度优化分析：

[1] M. Li and A. I. Mourikis, “High-precision, consistent ekf-based visualinertial odometry,” The International Journal of Robotics Research,
vol. 32, no. 6, pp. 690–711, 2013.

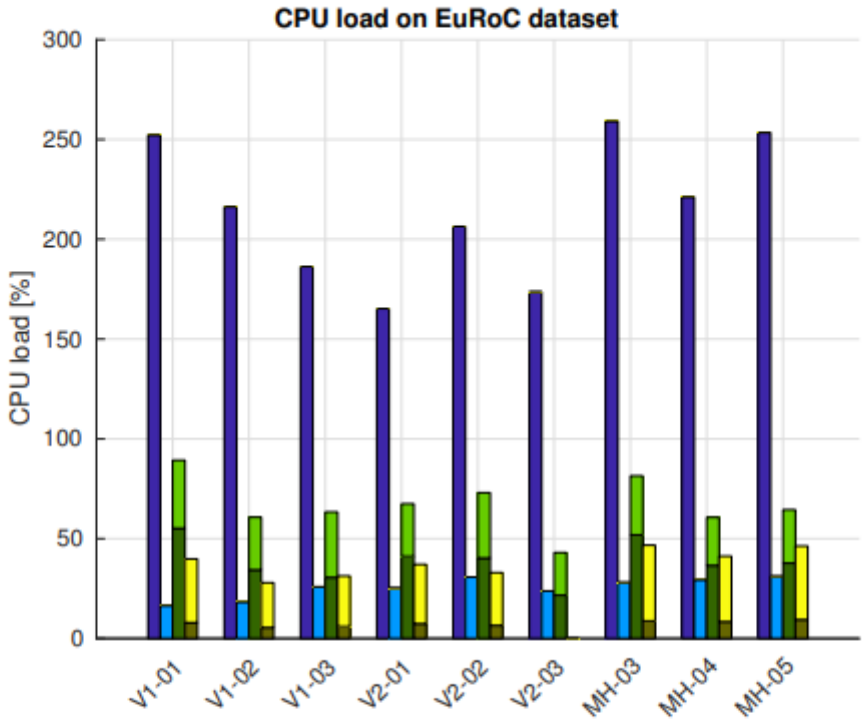
[2]G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, “Observability-based
rules for designing consistent ekf slam estimators,” The International
Journal of Robotics Research, vol. 29, no. 5, pp. 502–528, 2010.

效果：

quad-core i76770HQ 2.6Hz. 20HZ 的Stereo 和200 HZ的IMU



(a)



(b)

Fig. 2: (a) Root Mean Square Error (RMSE) and (b) average CPU load of OKVIS, ROVIO, VINS-MONO, and the proposed method on the EuRoC dataset. The parameters used for each method are the same as the values given in the corresponding Github repositories. Statistics are averaged over five runs on each dataset. For VINS-MONO and S-MSCKF, the frontend and backend are run as separate ROS nodes. The lighter color represents the CPU usage of the frontend while the darker color represents the backend. Note that the backend of VINS-MONO is run at 10hz because of limited CPU power.

看起来 OKVIS 基本不用看了 ROVIO 可以看一下 S-MSCKF 效果还是不错的
开源：

https://github.com/KumarRobotics/msckf_vio

OpenVINS

Tips：

优化型三角化的trick在下面的网址内有说明

https://zhuanlan.zhihu.com/p/77040286

总结：

阅读相关内容需要看一下MSCKF，MSCKF2.0，看视频效果确实不错。不知道真实跑是什么样子,可能会用这个算法进行落地了

VI-DSO

Direct Sparse Visual-Inertial Odometry using Dynamic Marginalization
DSO的IMU拓展版

摘要：

1. VI-DSO系统
2. 一种新颖初始化方式
3. dynamic marginalization的策略 还是在更新滑窗的策略
   不知道和直接套用isam2 比 哪个更加优秀
4. 在EcRoc上的多种测试

内容：

开源:

https://github.com/RonaldSun/VI-Stereo-DSO
网友开发版

总结：

因为没有开源，只有一个网友实现版，不好多加评论，感觉光度误差和IMU 之间的方差关系需要手动调参，有些失望。中规中据

VI-ORB

Visual-Inertial Monocular SLAM with Map Reuse

摘要：

1. IMU+ORB 耦合
2. 一个新的IMU初始化方法

内容：

1. IMU 的递推还是Forster 的预积分模型
2. IMU初始化(顺序，应该是从高可观性开始，一直标到低可观性上)：
   a. 先标Gyro_bias
   b. 之后是scale 和 重力大小
   c. accel bias 和重力方向
   d. IMU速度

开源：

均为网友实现的版本

https://github.com/jingpang/LearnVIORB

https://github.com/YoujieXia/VI_ORB_SLAM2

这个版本加不加IMU 效果不大

BASALT(看不懂)

Visual-Inertial Mapping with Non-Linear Factor Recovery

摘要：

LARVIO

Monocular Visual-Inertial Odometry with
an Unbiased Linear System Model and Robust
Feature Tracking Front-End

Lightweight hybrid visual-inertial odometry with closed-form zero velocity update

摘要：

1. 一个描述子辅助的LK光流
2. 更多的稳定性改进
3. ZUPT 零速抑制
   内容：
4. 相当于对匹配的LK点对过一次描述子来再去一次outlier
   具体就是把 ORB 的方向描述使用，Shi-Tomasi的角点方向

效果：

Table 4. Results of proposed and state-of-the-art VIOs using EuRoC dataset. Ten runs on each sequence and the means of positioning RMSEs (m) are calculated.

	MH_01	MH_02	MH_03	MH_04	MH_05	V1_01	V1_02	V1_03	V2_01	V2_02	V2_03
VINS-MONO	**0.159**	**0.182**	**0.199**	**0.350**	**0.313**	0.090	0.110	0.188	**0.089**	0.163	0.305
ROVIO	0.250	0.653	0.449	1.007	1.448	0.159	0.198	0.172	0.299	0.642	**0.190**
OKVIS	0.376	0.378	0.277	0.323	0.451	**0.087**	0.157	0.224	0.132	0.185	0.305
Proposed	0.289	0.258	0.331	0.394	0.423	0.117	**0.089**	**0.134**	0.097	**0.140**	0.211

Table 5. Average processing time (ms) and rate (Hz) of visual front-end and EKF/optimization back-end of our implementation and the state-of-the-art using the EuRoC dataset.

	Sequence	MH_01		MH_02		MH_03		MH_04		MH_05		V1_01		V1_02		V1_03		V2_01		V2_02		V2_03	
		Time	Rate	Time	Rate	Time	Rate	Time	Rate	Time	Rate	Time	Rate	Time	Rate	Time	Rate	Time	Rate	Time	Rate	Time	Rate
VINS-MONO	front-end	18.0	55	18.3	55	18.6	54	19.3	52	21.3	47	20.2	49	21.4	47	23.2	43	22.3	45	23.8	42	30.6	33
	back-end	50.2	20	50.9	20	50.1	20	50.1	20	53.0	19	53.1	19	45.9	22	37.9	26	54.4	18	48.3	21	33.4	30
ROVIO	front-end	2.0	505	1.9	526	2.0	497	2.1	476	2.0	490	1.9	538	2.0	508	2.1	481	2.0	503	2.0	510	2.0	478
	back-end	15.9	63	15.9	63	15.9	63	15.9	63	15.7	63	15.9	63	15.9	63	15.9	63	15.9	63	15.9	63	15.9	63
OKVIS	front-end	46.7	21	45.3	22	47.4	21	40.9	24	41.4	24	38.5	26	38.8	26	31.3	32	38.8	26	37.3	27	31.4	32
	back-end	39.8	25	39.4	25	39.9	25	32.1	31	33.1	30	30.6	33	25.5	39	19.2	52	29.6	34	27.9	36	18.0	56
Proposed	front-end	16.2	62	16.5	61	15.9	63	16.1	62	15.7	64	15.7	64	15.3	65	16.4	61	15.8	63	15.9	63	17.3	58
	back-end	5.5	182	5.9	169	6.1	164	5.5	181	6.0	166	5.7	174	5.4	185	4.9	203	5.7	176	5.6	178	4.6	218

新版

Algorithm	RMSE of estimated trajectories (m)									
	MH_01	MH_02	MH_03	MH_04	MH_05	V1_01	V1_02	V1_03	V2_01	V2_02	V2_03
Proposed	**0.1440**	0.2054	**0.1612**	**0.2696**	**0.3061**	**0.0883**	0.0783	0.1245	0.1008	**0.0970**	0.2257
3D IDP	0.2794	0.2843	0.2478	0.3452	0.5130	0.1480	0.0893	0.1461	0.1499	0.1108	0.2450
MSCKF	0.1947	0.2817	0.2138	0.3193	0.4022	0.0956	**0.0767**	**0.1238**	0.1442	0.1086	**0.2121**
VINS-MONO[26]	0.1590	**0.1817**	0.1987	0.3504	0.3139	0.0900	0.1098	0.1875	**0.0886**	0.1632	0.3051
R-VIO[29]	0.3582	1.0437	0.3192	2.0287	0.7064	0.1260	0.1275	0.3074	0.1302	0.2439	0.4523

总结：

虽然精度上不是特别好，感觉更多的是在，稳定性上 的贡献。
看一下ROVIO 为什么前端这么快

总结

主要的方向

1. 更鲁棒的初始化
2. IMU融合方式都用一遍
3. 继续换边缘化策略
   这里到底是使用策略还是使用isam2 的贝叶斯树还有待商榷

Sensor Fusion

Visual + Odom

se2lam(精读，适合扫地机环境)

Visual-Odometric Localization and Mapping for Ground Vehicles Using SE(2)-XYZ Constraints

摘要：

1. 引入一个SE2-XYZ 约束
2. 引入一个SE2 上的预积分
3. 开源了一个框架

内容：

1. 引入 Oberservatin 上的约束

效果：

TABLE I

ESTIMATION ERRORS STATISTICS (RMSE)

	Odom.	ORB-SLAM	SE2C	SE2
DATASET ROOM				
x err. (mm)	541.24	135.33	62.44	61.106
y err. (mm)	1028.83	371.01	93.15	59.021
ϕ err. (rad)	0.24835	0.12809	0.01567	0.01181
trans. err.(mm)	1162.51	394.93	112.15	84.956
accuracy*	4.469%	1.343%	0.381%	0.288%
DATASET WAREHOUSE				
x err. (mm)	1615.19	1038.66	304.22	171.129
y err. (mm)	460.90	507.13	393.87	279.766
ϕ err. (rad)	0.10062	0.17149	0.03924	0.04921
trans. err.(mm)	1679.67	1155.85	497.68	327.955
accuracy	1.142%	0.787%	0.339%	0.223%

*Accuracy is calculated by the translation error over the travel distance in *one loop*.

开源：

https://github.com/izhengfan/se2lam

总结：

添加新约束形式

multiple Featue

RESLAM (Edge)

RESLAM: A real-time robust edge-based SLAM system

摘要：

1. RGBD Edge-Based SLAM system

内容：

1. 前端 EBVO Frame To Frame ：

   a. Canny 做Edge检测

   b. 金字塔+DT 构建重投影误差优化Pose

   这里应该是使用了RGBD的深度信息

   c. 初值确定方法(因为先验信息较弱，这里的计算估计在实际当中会出问题)
2. LocalMap Frame To Map

   a. 进一步优化pixel的深度和内参
3. 回环：

   a. 检测回环 random-fern-based bag of words

   效果：

Comparison of the Absolute Trajectory Error $[cm]$

	BF [9]	RGBDTAM [8]	DVO-SLAM [7]	EF [6]	ORB2-SLAM [4]	RGBDSLAM [5]	REVO [10]	Edge+ICP [20]	Our Methods			
Seq.	Direct				Features		Edges		VO	LM	LC	All
fr1/xyz	-	1.0	1.1	1.1	**0.8**	1.3	6.8	1.6	7.4	2.4	2.2	1.1
fr1/desk2	-	**4.2**	4.6	4.8	2.2	4.2	8.2	6.0	7.1	6.3	5.4	4.8
fr2/desk	-	2.7	1.7	7.1	**0.9**	5.7	8.9	9.5	3.5	3.4	3.6	1.9
fr2/xyz	**0.4**	0.7	1.8	1.1	0.8	0.8	1.0	-	0.9	0.5	0.8	0.5
fr3/office	2.2	2.7	3.5	1.7	**1.0**	3.2	11.0	-	13	7.8	4.2	3.5
icl/lr-kt0	**0.6**	-	10.4	0.9	0.8	2.6	24	5.4	6.5	3.2	2.2	2.1
icl/lr-kt1	**0.4**	-	2.9	0.9	1.6	0.8	2.3	0.9	1.3	1.9	2.5	1.7

TABLE II

THE RMSE OF ATE IN $[cm]$ FOR BUNDLEFUSION (BF) [9], RGBDTAM [8], DVO-SLAM [7], ELASTICFUSION (EF) [6],ORB-SLAM2 [4], RGBD-SLAM [5], REVO [10] AND EDGE+ICP [20] COMPARED TO OUR METHOD WITH VO, LOCAL MAPPING (LM), LOOP CLOSURE (LC) AND ALL (VO+LM+LC) ON THE TUM-RGBD [27] AND ICL-NUIM [28] DATASETS.

开源：

https://github.com/fabianschenk/RESLAM

总结：

效果不佳

EDVO(Edge)

Edge-Direct Visual Odometry

摘要：

1. 对边缘进行直接法

内容：

效果：

Seq.	ours Canny$_{KF}$	ours Canny$_{FF}$	ours LoG$_{FF}$	ours Sobel$_{FF}$	ours SE$_{FF}$	REVO[19] SE$_{FF}$	REVO[19] SE$_{KF}$	DSLAM[11] ICP+Gray	ORB2[18] Feat.	SLAM[6] Feat.	D-EA[1] Canny
Relative Pose Error (RPE) [m/s]											
fr1/xyz	0.02228	0.02768	0.02821	0.02712	0.03289	0.03202	0.01957	0.02661	**0.01470**	0.04193	0.04942
fr1/rpy	-	0.03126	**0.02714**	0.03694	0.03041	0.03553	0.04037	0.04865	0.03221	0.07028	0.16150
fr1/desk	**0.02664**	0.03022	0.03022	0.03598	0.03056	0.07800	0.22196	0.04429	0.06178	0.05346	0.10654
fr1/desk2	-	**0.04387**	0.04953	0.05566	0.04490	0.07056	0.06703	0.05722	0.06535	0.06955	0.20117
fr1/room	-	0.04830	0.06239	0.05240	0.05006	0.04816	**0.04272**	0.06427	0.07081	0.06666	0.21649
fr1/plant	-	**0.02736**	0.02752	0.04171	0.05006	0.03063	0.02381	0.04362	0.04218	0.03789	0.34099
fr2/desk	**0.01237**	0.01375	0.01375	0.01800	0.03021	0.01426	0.02453	0.03248	0.03067	0.01400	0.09968
Absolute Trajectory Error (ATE) [m]											
fr1/xyz	0.04567	0.04478	0.04461	0.05260	0.04115	0.09011	0.05375	0.05760	**0.00882**	0.01347	0.13006
fr1/rpy	-	0.05561	0.03982	0.04795	0.04983	0.08933	0.07684	0.16341	0.08090	**0.02874**	0.14822
fr1/desk	0.03133	0.05387	0.05358	0.05977	0.04802	0.18648	0.54789	0.18251	0.09091	**0.02583**	0.16376
fr1/desk2	-	0.06798	0.08384	0.08189	0.06271	0.16866	0.18163	0.18861	0.10090	**0.04256**	0.44886
fr1/room	-	0.27382	0.34505	0.31586	0.34167	0.30594	0.28897	0.21559	0.20282	**0.10117**	0.60361
fr1/plant	-	0.07559	0.06708	0.09975	0.06560	0.07300	**0.05623**	0.12216	0.07234	0.06388	0.56927
fr2/desk	0.12540	0.16664	0.18830	0.19074	0.27464	0.32902	0.09590	0.46796	0.38657	**0.09505**	0.94546

Table 1. Comparison of the performance of our system using three different types of edges. Blue denotes best performing frame-to-frame VO, excluding SLAM or keyframe systems. **Bold** denotes best performing system overall. A dashed line indicates that using keyframes did not improve performance.

开源：

https://github.com/kevinchristensen1/EdgeDirectVO

深度学习SLAM