# 2018 July Reading Reports

## Linear SLAM

This paper provides a new map joining framework which only requires solving **linear least squares problems** and performing nonlinear coordinate transformations. There is no assumption on the structure of the covariance matrices of the local maps. There is no need of an initial guess and no need of iterations. It has been shown that it could solve different versions of SLAM problems (feature-based SLAM, pose graph SLAM and decoupling SLAM). The consistency analysis of Linear SLAM is performed. The theoretical analysis on the computational complexity shows the efficiency. It is also pointed out that one has the freedom to choose the coordinate frame of a local map or the global map.

### Joining Two Pose-feature Maps using Linear Least Squares

The **main idea** is to convert the submaps to the intermediate coordinate to make the objective function linear.

### Sequential Map Joining, Divide and Conquer Map joining and Computatinoal Complexity

The improvements are from no iteration and no initial values, because of the linear form of the objective function. It fuses two maps at a time instead of fusing all together.

### Results

The Consistency Analysis is done by NEES, not analytically proved. The Linear SLAM (LSLAM) can be faster using the Divide and Conquer method, but the estimation and the uncertainty are not as good as the full non-linear method, which is used as the upper bound for the results in the Table 2 and 4. The 3D Pose-feature Map Joining's or feature-only's results are on the simulation with only one test.

The LSLAM implementation is slower than g2o. It is explained as"g2o code is highly optimized and is one of the fastest implementations of Nonlinear LS SLAM".

One thing uncertain is that the accurate under **noisy measurements** or even **outliers**, which may cause linear solution unstable.

### For our related projects

Might be useful to merge multiple Lidar maps. But current implementation doesn't keep features in the map. And Lidar features are different from the ones in image. One way to describe feature can be surfel as [1]. But the stability of the surfel features requires further investigation.

Using LSLAM can obtain suboptimal results efficiently, but not as good as state-of-the-art non-linear method.

[1] Behley J, Stachniss C. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments[J].

---

# LIMO

## Feature extraction and preprocessing

(Block A,B) The feature tracking methodology is viso2. And semantic image is used for dynamic object removing.

## Scale estimation

(Block S) The depth corresponding to detected feature points is extracted from LIDAR. One shot depth estimation approach is applied. LIDAR points are  projected onto the image plane.

1. For each feature point $f$, region around the point $F$, which is
2. segmented as $F_{seg}$ (using histogram by depth) and
3. fitted to a plane $p$ (triangle $F_{\Delta}$ with maximum area). Ground plane will have a different fitting algorithm.
4. Intersect $p$ with the line of sight corresponding to $f$ to get its depth.
5. Perform a test for the estimated depth.

## Frame to frame odometry

(Block C) PnP and fundamental matrix added with Cauchy function are used to form the cost function.

## Backend

(Block D) Keyframes, landmarks are selected by the methods introduced (no Block E).

Keyframes are chosen when motion (rotation) is rapid, and rejected when the mean optical flow is smaller than a fix threshold. The remaining frames are chosen in time intervals of 0.3 s. Connectivity of the keyframes by counting landmarks that connect the current keyframe with the newest keyframe is used to determine the length of the optimization window. The window is bounded on both sides.

Landmarks are categorized in **near, middle and far** bins. Near with the largest optical flow, random (the idea is to use unseen landmarks) for the middle bin. Longest tracking length for the far points. Then a semantic information is used to determine the weight of landmarks (0.9 for vegetation).

Landmark depth is inserted by measured depth, and constrained by the oldest motion.

Robustification and trimmed-least-square-like approach are applied on the problem formulation.

---

# Inverse Depth Parameterization

**Doubling the map state vector size, but it benefits the intialization and distant features.**

Low parallax features (XYZ feature parameterization) are not well represented by the Gaussian distributions implicit.

The single Gaussian representation that the inverse depth parametrization, including undelayed initialization large “infinite” depths, which can be useful for the rotation estimation.

It is important to represent the uncertainty in depth of seemingly infinite features.

6-D state vector $[x_i, y_i, z_i, \theta_i, \phi_i, \rho_i]^T$ is used for each of the feature points. It proves and shows that the inverse depth parameterization can have better linearity in both low or high parallax, if the feature is continuously observed. It also convert inverse depth into XYZ the linearity index is less enough.

---

# V-LOAM

The method is suitable when the sensors moves at a high speed and is subject to significant lighting changes. It is divided into two sections. One is the visual odometry section, which contains depth map registration block. Another one is Lidar odometry. The sensor pose outputs are integration of the transforms from both sections.

The visual odometry solve two types of equations about known or unknown depth to obtain the 6-DOF motion. Then the depthmap is maintained. Associated points form the plane is used to get the projection of a ray on the plane, while the unassociated points are triangulated.

Then the results from visual odometry is refined by the lidar odometry method, which contains sweep to sweep registration and sweep to map registration. The drift of visual odometry is modeled as linear drift. The remaining part is similar with LOAM. There are many tests on the methods, while the method itself is not tight-coupled.

---

# SOFT-SLAM

SOFT-SLAM enables a complete separation of the odometry part and mapping part. It does not share the map between SOFT (stereo odometry algorithm relying on feature tracking) odometry and mapping threads as PTAM, ORB-SLAM2. It yields a constant runtime with global consistency. An exponential filter merges the odometry and mapping parts.

## Odometry Thread

If IMU is available, gyroscopic measurements and one-point RANSAC optimization are applied. Otherwise, a three-point algorithm is applied. For KITTI (five-point Nister algorithm, five-point RANSAC on the left stereo image), the rotation from a single camera can improve localization precision.

The sum of absolute differences (SAD) is used. Circular matching, both left and right images of two consecutive frames corresponding to steps $k-1$ and $k$.
$\mathcal{I}_{k-1}^{L} \rightarrow \mathcal{I}_{k-1}^{R} \rightarrow \mathcal{I}_{k}^{R} \rightarrow \mathcal{I}_{k}^{L} \rightarrow \mathcal{I}_{k-1}^{L}$. Normalized cross correlation (NCC) as an additional check.

Properties includes (i) unique identifier, (ii) age, (iii) current position in image space, (iv) feature strength, (v) class, and (vi) initial descriptor. Spatial and temporal variety policy (careful selection) is realized.

- For no IMU fusion, five-point RANSAC (closed-form) for rotation and is not updated with translation. The translation is first from the one-point RANSAC, and optimized in the Gaussian-Newton algorithm (weighted points).
- With IMU, it uses Kalman filter to estimate the rotation and bias. Use one-point RANSAC to find the inlier features. Finally, it optimize the rotation and translation in the optimization.

A exponential filtering is applied to smooth the outputs.

## Mapping Thread

Put feature management into the frame queue. The keyframe is set according to the **distances** from the last stored keyframe. Then loop closure is checked, following with a sparse set of poses only optimization.

The pose and covariance of the keyframe are proportional to the distance between the previous and the new keyframe. **Temporary** keyframe will be discard after the graph optimization, while full map keyframe will be insert to the graph.

### Loop Closing

Loop closing is from the candidate keyframes with the radius and angle. Histograms of each keyframe consisting of four classes are measured by SAD.

For the candidates:

- Circular matching and NCC.
- Three-point RANSAC.
- Optimization when the number of inliers is larger than the threshold.
- Check if the final linking distance is too large.

Levenberg-Marquard algorithm in g2o to optimize the pose graph. The uncertainties are assumed as proportional to the magnitude of the relative motion (rotation and translation).

---

# ORB-SLAM2

**Stereo keypoints** are defined by three coordinates $\mathbf{x}_s = (u_L, v_L, u_R)$. Then for both stereo or RGB-D cameras, $u_R = u_L - \frac{f_x b}{d}$ is applied.

**Monocular keypoints** are defined by two coordinates $\mathbf{x}_m = (u_L, v_L)$.

As the same as ORB-SLAM, the system performs BA to optimize the camera pose in the tracking thread (**motion-only BA**), to optimize a local window of keyframes and points in the local mapping thread (**local BA**), and after a loop closure to optimize all keyframes and points (**full BA**). Levenberg–Marquardt method is applied. Different projection functions $\pi_m$ and $\pi_s$ are used for these two kinds of keypoints.

Loop closing will abort the full BA optimization. When the loop is closed, the full BA will be optimized again. Nonupdated keyframes and points will be propagated through the spanning tree.

The keyframe insertion is the same as ORB-SLAM. In localization mode, the tracking leverages visual odometry matches (matched with the previous frame, will drift) and matches to map point (drift-free).

Multi-threading system has the nondeterministic nature.

Two metrics are used to evaluate the results, the absolute translation root-mean-square error (RMSE) and the average relative translation and rotation errors.