	SLAP: SLam, but it's not About Poetry
Course	3d Scanning and Motion Capture IN2354
Authors	REBOUD Pierre ge43god@mytum.de
Group Nb.	17
Supervisors	BUROV Andrei, HÖLLEIN Lukas
Date	14.02.2023

Abstract

SLAP (SLam, but it's not About Poetry) is a monocular online Simultaneous Localization and Mapping (SLAM) Algorithm capable of jointly optimizing pose and landmark positions. Though it does not outperform current state of the art by conventional metrics like estimation accuracy, speed, nor robustness to adverse lighting conditions, it runs at under 2k lines of code including docstrings, making it one of the shortest freely available SLAM implementations.

1 Motivation and Idea

SLAM has been a prolific field of research since the first foundational lines of work [1], [2], [3] leveraged ever increasing computational capabilities of modern computers. As computational resources become more widely available and the rise of robotic applications in all domains of life, simple and light-weight localization and mapping frameworks become increasingly important.

2 Related Work

Some of the most influential lines of works of the past decade are listed here for the sake of completeness:

- ORB SLAM v2 [2]
- A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots [1]
- Real-time loop closure in 2D LIDAR SLAM [3]
- Large-Scale Direct SLAM with Stereo Cameras [4]
- Efficient Sparse Pose Adjustment for 2D Mapping [5]

3 Method

SLAP uses a simple architecture, which is composed of the following modules:

- Feature extraction and matching
- Multi-view geometry
- Map and data structures
- Bundle adjustment
- Computational aspects

These are explained in detail in the following subsections.

3.1 Feature Extraction and Matching

SLAP leverages an indirect SLAM [6] approach, where OpenCV's implementation of ORB (Oriented FAST and Rotated BRIEF) features [2] is used to obtain robust features from the monocular video feed. The features are matched across two frames using a k-d-tree, which allows the each feature to be matched to candidate features in the other frame. Lowe's ratio [7] is then used to decide which of the two feature matches is retained, describing the necessary distance improvement necessary to pass the threshold. Match number 1 is selected over match number 2 when the following condition holds:

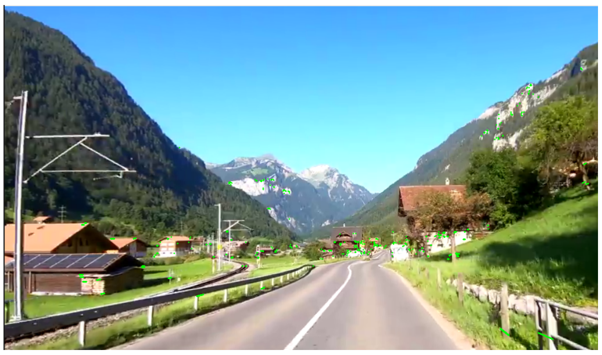


Figure 1: Light green lines outline the 2d landmark correspondences across two frames

$$\text{match\_1.distance} < L * \text{match\_2.distance} \tag{1}$$

, where L represents Lowe's ratio (typical value of 0.8).

The resulting frame matches can be seen in figure 1 as an example.

3.2 Multi-view Geometry

Subsequently, the matched features are used to compute the 3d landmark locations by means of 2-view geometry. We make use of Direct Linear Transformation (DLT, outlined in [8]), where leveraging the re-projection constraint allows to retrieve first 3d pose estimations. In particular, we first construct the A matrix as follows:

$$A = \begin{bmatrix} x * \mathbf{p^{3}}^\intercal - \mathbf{p^{1}}^\intercal \\ y * \mathbf{p^{3}}^\intercal - \mathbf{p^{2}}^\intercal \\ x * \mathbf{p'^{3}}^\intercal - \mathbf{p'^{1}}^\intercal \\ y * \mathbf{p'^{3}}^\intercal - \mathbf{p'^{2}}^\intercal \end{bmatrix} \tag{2}$$

, where x, y are the 2d coordinates of the 2d point in the first camera, $\mathbf{p^{i}}^\intercal$ the i-th row of camera matrix P, and $\mathbf{p'^{i}}^\intercal$ represents the rows of the second camera matrix P'. After having built the matrix, A's Singular Value Decomposition (SVD) is calculated in order to obtain its null-space, which corresponds to the 3d beam pointing from the camera origin through the 2d point measurement. Normalizing by the respective 2d-homogeneous coordinate gives the 3d landmark estimation corresponding to the 2d point.

3.3 Map and Data Structures

Having estimated the spatial landmark locations in the previous step, we move to filling three arrays with the estimated cameras poses, the estimated 3d point locations and the observed 2d point locations respectively. We make sure to keep a record of the correspondences of each estimated 3d landmark to its sightings within the frames of the optimization window. We keep track of all features across time, though only the ones contained within the optimization window (20 frames) are used to refine the estimation during the optimization step.

3.4 Bundle Adjustment

At each map update, we additionally run a Bundle Adjustment (BA) routine which jointly optimizes the landmark locations and the camera poses. For this purpose, we leverage PyTorch's automatic differentiation framework and use the Adam optimization algorithm [9] with a learning rate of 5.10^{-4} to minimize the reprojection loss ϵ. More suitable second order optimization algorithms, such as Levenberg-Marquardt, are not pre-implemented in PyTorch and thus not used. This loss relates point p_i's projection on camera P_j with the measurement m_{ij} in the following way:

$$\epsilon := \|P_j p_i - m_{ij}\|_2^2 \tag{3}$$

where i is the landmark index within the j-th frame.

3.5 Computational Aspects

As we leverage the libraries OpenCV and PyTorch, which are in themselves bindings to much faster C++ code, our implementation does not suffer excessively from the speed of the Python interpreter with its global thread lock. In addition, leveraging the widely used library PyTorch allows to seamlessly stream the necessary data to separate accelerators such as Graphical Processing Units (GPUs), Tensor Processing Units (TPUs), or Metal Performance Shaders (MPS) from Apple to accelerate computation and offload the optimization steps to dedicated hardware.

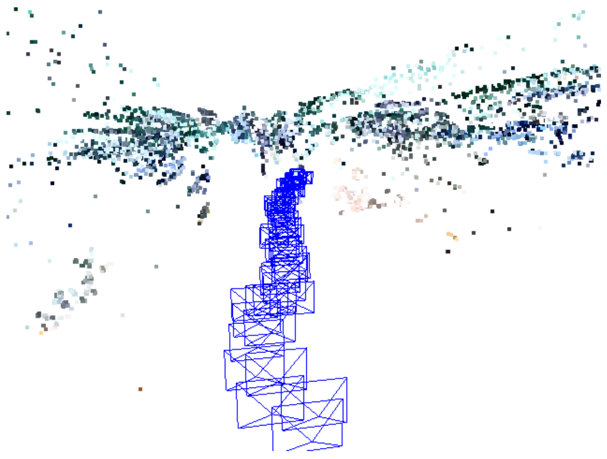
4 Results

SLAP is a very lightweight SLAM implementation. Therefore, traditional metrics by which SLAM algorithms are usually bechmarked against are not in focus. Instead, other light-weight implementations (such as twitchslam) provide a more fitting comparison due to the similarity of the repsective implementations' objectives.

We apply SLAP and twitchslam to 2 different data sets: RGB-D Dataset [10] and a video feed from a dashboard camera of a driving car in winding roads from here. The video excerpt is chosen to contain as little camera exposure changes as possible, in order to increase the number of feature-matches across frames. Additionally, the video showcases both close-by features such as houses and road signs, as well as features located much further away, such as mountain tops.

A qualitative comparison of SLAP's reconstructed map at frame 25 with respect to twitchslam's reconstruction at the same frame is outlined in figure 2.

As can be seen, twitchslam leverages a complex Graph-SLAM framework which allows, among other features, to optimize over parameter distributions instead of point-wise values, like in the case of SLAP. This impacts reconstruction accuracy. On the other hand, SLAP does not leverage such a black-box framework and instead only relies on the automatic differentiation framework PyTorch. This allows the code to provide the reader with an eas-


(a) SLAP reconstruction

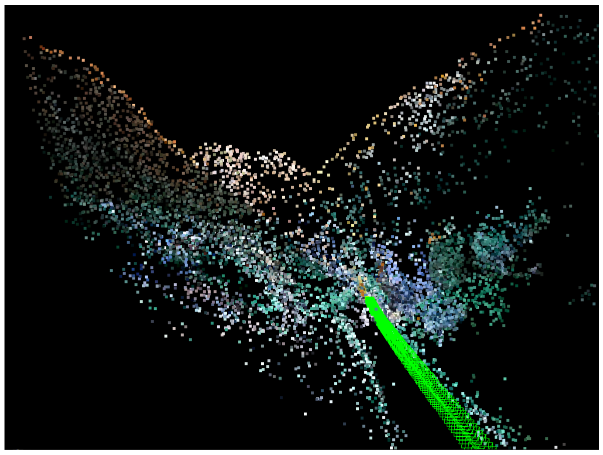

(b) twitchslam reconstruction

Figure 2: Comparison between generated poses and landmarks

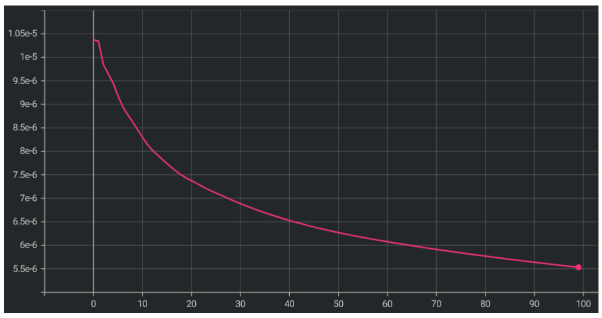


Figure 3: MSE loss across 100 optimization iterations

ier oversight over the different SLAM pipeline modules while retaining a reasonable reconstruction accuracy.

5 Analysis

This section inspects the quantitative performance of SLAP.

5.1 Accuracy

Some important experiments could not be run in-time, including the average drift calculation with respect to a ground-truth trajectory. The RGBD dataset was included for this purpose, however, a small bug in the video reader currently hinders the correct correspondences from working (dashboard video feed works flawlessly). Therefore, the evaluation of this experiment and linked feature-ablation studies is currently not possible.

5.2 Optimization

The optimization over the culled points usually starts off from a good baseline, as point correspondences were refined in the previous steps. Notwithstanding, the optimizer is capable of approximately halving the average re-projection error on the frame's plane, as can be seen in the figure3.

5.3 Outlier Rejection

The matching step of 2d observations at step 3.1 is error-prone and generates erroneous matches. As the optimizer utilizes the Mean Squared Error (MSE) loss, wrong associations are highly over-weighed when compared to valid matches. This leads to the effect visualized in figure 4, where one erroneous match strongly biases the parameter estimation.

As in an error-free case where the normality assumption is applied to measurement errors, $p_i \sim \mathcal{N}(0, \Sigma_{p_i})$ and $m_ij \sim \mathcal{N}(0, \Sigma_{m_{ij}})$ with some uncertainty matrices $\Sigma_{p/m}$, it follows that $\epsilon \sim \chi^2_n$, where n is the number of point correspondences present on the frame j and therefore also the number of degrees of freedom of the χ^2-distribution. As in the real world, erroneous correspondences still occur, ϵ's true distribution is not a χ^2 distribution, as the content of the norm brackets 3 follows a non-zero-mean normal distribution.

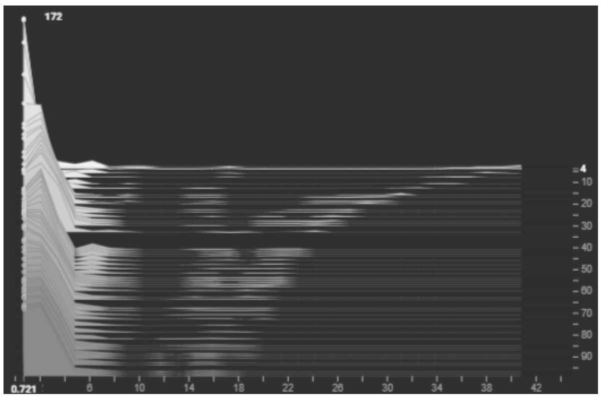


Figure 4: Distribution of the point-wise re-projection errors (width axis, in pixels) at each optimization step (depth axis). Erroneous matches generate large re-projection errors in selected pixels at the first iteration (bottom histogram, right axis side). The optimizer subsequently tries to fit the remaining parameters to this erroneous correspondence, which ruins the parameter estimation (true matches now produce an increased re-projection error)

A common approach to tackle this issue is to apply a threshold, starting from which the correspondences are rejected and the corresponding landmarks pruned from the map. This step requires to manage the trade-off between precision (out of the predicted correspondences, how many will be kept) and recall (out of all true correspondences, how many will be estimated/kept). A possible way to systematically investigate this is the mean Average Precision(mAP), which provides the average precision over different recall values. This method requires the availability of ground-truth information on the point-matches across frames. This information being unavailable in the case of SLAP, the cutoff-threshold is arbitrarily chosen to be 4 pixels. This value seems to keep most correspondences, while pruning the obviously erroneous ones.

5.4 Computational Resources

SLAP is an open source effort without funding. Therefore, its development team does have access to a limited pool of computational resources. The results outlined in this paper are obtained on a 2-core CPU machine from 2015.

6 Future Work

SLAP implements the bare-bone features which are strictly necessary for a SLAM algorithm. Therefore, many features can be implemented to improve the library's performance. For the sake of completeness, some of them are outlined in the following subsections:

6.1 Application of Geometric Constraints

Currently, no separate checks are performed with respect to the plausibility of the estimated re-projections onto the frames. With very little additional effort, it would be possible to check whether the re-projections lie within the camera field of view, as well as in front of the camera. Landmarks not fulfilling these conditions could be removed from the map, as they do not pass this basic plausibility check.

6.2 Optimization Space

SLAP's BA takes place on the 2d coordinate systems of the video's frames. It is possible to bundle all 2d projection lines to infinity and estimate landmark locations from approximate line crossings.

6.3 Optimization in the case of Uncertainty

Each 2d point measurement is plagued with uncertainty. As a landmark corresponds to a null-dimensional feature in Euclidean Space, it's projection on a discretized subset of $\mathbb{N}^2$ (the camera pixel frame) necessarily leads to an error upper-bounded by the frame's resolution. Additional effects, such as optical or atmospheric effects, can act as an amplification to such errors. A common procedure to deal with such artifacts is to measure the noise of landmark and point estimations. This allows to calculate the parameter's respective covariance matrices and optimize not only over single Maximum-Likelihood values, but over distributions, yielding a probabilistic error model. An entire line of research - namely, Bayesian Optimization - tackles these sorts of problems.

6.4 Extended Kalman Filters

Usually, the camera motion can be described by some kinematic model, which enforces additional physical constraints to the optimized multi-view parameter estimations. For instance, when uncertainty estimations are taken into account for the camera's states and for the landmarks, it is possible to fuse the information of both models weighted based on the respective uncertainties, which leads to an improved estimation accuracy and less estimation drift over time.

6.5 Loop Closures

Letting the optimizer run over the entirety of the map allows to obtain loop closures when the camera returns to a location containing already-mapped landmarks. As such features are very computationally intensive and scale in polynomial time with the number of landmarks, this line of work is not among the main goals of SLAP.

7 Conclusion

SLAP is a bare-bone monocular SLAM implementation that focuses on a compact code-base while maintaining core functionalities of SLAM algorithms. It allows to jointly estimate pose and landmark positions within an optimization window on any video feed, given an initial camera intrinsics estimate. Although it does not outperform state-of-the-art SLAM implementations, it is one of the shortest implementations freely available.

References

[1] ThrunSebastian, BurgardWolfram, and FoxDieter. "A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots". In: *Autonomous Robots* (1998).

[2] Raul Mur-Artal and Juan D. Tardos. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33.5 (Oct. 2017), pp. 1255–1262. DOI: `10.1109/tro.2017.2705103`. URL: `https://doi.org/10.1109%2Ftro.2017.2705103`.

[3] Wolfgang Hess et al. "Real-time loop closure in 2D LIDAR SLAM". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 1271–1278. DOI: `10.1109/ICRA.2016.7487258`.

[4] Jakob Engel, Jörg Stückler, and Daniel Cremers. "Large-scale direct SLAM with stereo cameras". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 1935–1942. DOI: `10.1109/IROS.2015.7353631`.

[5] Kurt Konolige et al. "Efficient Sparse Pose Adjustment for 2D mapping". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 22–29. DOI: `10.1109/IROS.2010.5649043`.

[6] Guillaume Bresson et al. "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving". In: *IEEE Transactions on Intelligent Vehicles* 2 (2017), pp. 194–220.

[7] David Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60 (Nov. 2004), pp. 91–. DOI: `10.1023/B:VISI.0000029664.99615.94`.

[8] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press, 2004. DOI: `10.1017/CBO9780511811685`.

[9] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: `10.48550/ARXIV.1412.6980`. URL: `https://arxiv.org/abs/1412.6980`.

[10] J. Sturm et al. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*. Oct. 2012.