COLAM - An Offline Python SLAM Using COLMAP

Yifei Liu,[2] Weirong Chen,[1] Yidan Gao,[1] Kexin Shi[2]
Supervised by Paul-Edouard Sarlin[1]
[1]ETH Zurich [2]University of Zurich

Abstract

The report presents COLAM, an offline python SLAM using COLMAP that is robust, accurate, and highly extensible. We build the new system to leverage the advantages of both COLMAP and ORB-SLAM, the former known for its high-quality reconstruction and the latter for its efficient tracking of sequential data. Our approach was evaluated against COLMAP and ORB-SLAM2 on 20 sequences from TUM-RGBD and KITTI, representing indoor and outdoor environments, respectively. The results show that COLAM achieves a much faster speed than COLMAP, better reconstruction against ORB-SLAM2, and the same level of trajectory accuracy as both. Our code would be available at https://github.com/Chiaki530/COLMAPSLAM.

1. Introduction

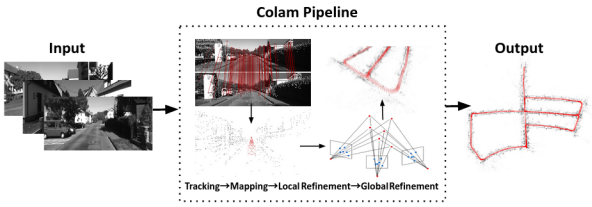


Figure 1: COLAM Pipeline

Simultaneous localization and mapping (SLAM) and Structure from Motion (SfM) are two highly overlapping tasks for building a 3d mapping and getting a trajectory of cameras, with the difference that SLAM processes sequential images very efficiently in real-time and focuses mainly on the trajectory, while SfM processes unordered images and focuses on the 3d mapping, but is significantly slower. Despite their differences, they have both found positions in an enormous range of applications.

However, if one wants to build a large-scale reconstruction like for cities, SLAM is fast but does not provide good enough 3d mapping because it focuses mainly on the trajectory. SfM has a good 3d mapping but is too slow and takes too long to reconstruct a large-scale scene. The appropriate trade-off between these two methods should be an offline but not too slow system that can build a sufficiently good reconstruction for video sequences.

In this work we build on the main ideas from ORB-SLAM[7] and COLMAP[11], to design COLAM, an offline SLAM in python, which has the following properties:
• Faster speed than COLMAP.
• Same level of trajectory accuracy as ORB-SLAM2[8].
• Better reconstruction quality than ORB-SLAM2.
• High extensibility, 100% python, and support customized features and matching methods thanks to hloc toolbox[9].

2. Related Work

NetVLAD[2] is a CNN architecture for weakly supervised place recognition, which outperforms non-learned image representations. We use it to extract global features from images to detect loops.

Hloc[9] is a toolbox for SfM implementing hierarchical localization, leveraging image retrieval and feature matching, and is fast, accurate, and scalable. We slightly adapt hloc for sequential performance and use it to extract and match features as well as to detect loops.

COLMAP[11] is a robust and accurate incremental Structure-from-Motion reconstruction pipeline for unordered images, with user-friendly graphical and command-line interfaces. We use its python binding, pyCOLMAP, in the registration, triangulation, and local refinement modules.

ORB-SLAM[7] is a fast, robust monocular SLAM system able to work in real-time in both indoor and outdoor environments. We leverage ideas from all three threads of the slam system: tracking, mapping, and loop closing, to achieve high efficiency and alleviate scale drift.

Ceres[1] is an open-source C++ library for modeling and solving large, complicated optimization problems. We use its python binding pyCeres for bundle adjustment and pose graph optimization.

3. System Overview

COLAM, as in Figure 2, is composed of four modules: initialization, tracking, local mapping, and loop closure. After initialization, we perform tracking for each frame and pass each selected keyframe to local mapping, loop detection, and closing.

Globally we maintain the Reconstruction Object, the Correspondence Graph Object, and the Covisibility Graph. The first two objects are from COLMAP, and they store all necessary information for the images, keypoints, and map points. The Covisibility Graph is defined in ORB-SLAM as a graph in which nodes are keyframes, and edge weights the number of shared map points between two image nodes. Two keyframes are defined as neighbors if they share > 15 map points.

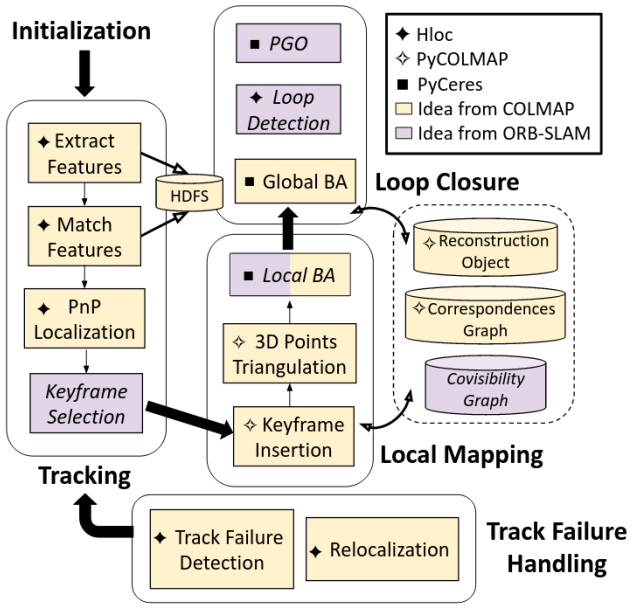


Figure 2: COLAM system overview

4. Initialization

The purpose of map initialization is to compute the relative poses of an initial set of frames and triangulate an initial set of map points. We use the hloc toolbox[9] to extract either SIFT, SuperPoint[3] or customized features, and match them through either nearest neighbor or SuperGlue[10]. Then we use pycolmap[1], a python binding for COLMAP[11], to reconstruct an initial scene and supports both calibrated and uncalibrated cameras.

[1]https://github.com/colmap/pycolmap

5. Tracking

5.1. New Frame Localization

We extract the same kind of features as in initialization and match them with the last three keyframes. Then we estimate the absolute pose using RANSAC and the PnP algorithm[6], followed by a motion-only Bundle Adjustment.

In the case of track failure, which could be due to inadequate features, matches, or 3D point inliers, we handle it by registering more recent frames as keyframes and do relocalization globally.

5.2. Keyframe Check

The goal of keyframing is to balance the accuracy and efficiency of the slam pipeline. We get rid of redundant frames to enable more powerful deep learning-based feature extractor[3] and matcher[10] within the same time span, and if performed well, could filter out frames of poor quality and add robustness to the reconstruction.

To insert a new keyframe, all the conditions must be met as follows:

1. The average optical flow of all matches between the current frame and the last keyframe, normalized against the image width, is larger than 0.02.
2. More than 100 matches score higher than 0.9 if using nearest neighbor and 0.7 if using SuperGlue[9].
3. The percentage of matched keypoints is between 10% to 50%.

Condition 1 and the upper bound of the matched ratio in condition 3 ensures enough movement between keyframes for efficiency, while condition 2 and the lower bound in condition 3 guarantees the tracking quality for accuracy.

To prevent track failure and facilitate refinements using the covisibility graph, we also insert keyframes when more than 4 frames have passed from the last keyframe, and when there are less than 60 3D point inliers during the PnP localization.

6. Local Mapping

6.1. Keyframe Insertion and Triangulation

We add the matches between the new keyframe and past 3 keyframes into the correspondence graph and register the keyframe in the reconstruction object. We do not explicitly update the covisibility graph as it can be inferred from the reconstruction object.

Then we triangulate new map points using the robust and efficient incremental triangulator from COLMAP[11], which leverages transitivity and can handle outlier contaminations.

6.2. Local Bundle Adjustment

We perform local BA in the same fashion as ORB-SLAM that optimizes the current keyframe, all its neighbors in the covisibility graph, and all map points seen by those keyframes. All other keyframes seeing these map points contribute to the loss but keep their poses fixed.

After local BA, we merge tracks and filter modified keyframes and map points to remove outliers as is done in COLMAP.

6.3. Global Bundle Adjustment

We perform a global BA every 20 keyframes and optimize all keyframes and map points. We do it in an iterative fashion by first retriangulating map points, merging tracks and filtering outliers as is done in COLMAP.

7. Loop Closing

We perform loop closing in a similar fashion to ORB-SLAM[7]. We first detect a loop, then compute a similarity transformation to close the loop.

7.1. Loop Candidates Detection

We use NetVLAD[2] to extract a 4096-d and L2-normalised global descriptor for each frame. To compare image similarities, we compute the dot product between two global descriptors and get a similarity score between 0 and 1. A loop is accepted if consecutively three frames have similarity scores > 0.4 with the current keyframe.

7.2. Loop Fusion and Similarity Transformation

In this step we fuse duplicated map points seen by current keyframe and loop candidate, and compute a similarity transformation from these 3D-3D correspondence.

We first solve a Perspective-n-Point (PnP)[6] problem in a RANSAC scheme to relocalize the current keyframe with respect to the loop candidate and candidate's neighbors in the covisibility graph. We store the pose of the current keyframe, and change it to the newly estimated one. The stored pose will be used to move map points after Pose Graph Optimization. Then we further validate inlier PnP correspondences by projecting the 3D points into the image and compare the reprojection error with a strict threshold. If this 3D-2D correspondence is validated, we add the observation to the current keyframe and merge the 3D point with 2D keypoint's corresponding 3D point, if it has one. We compute a similarity transformation from the validated 3D-3D correspondences using Horn's method[5], and get a scale s_{cl} between current keyframe K_c and loop candidate frame K_l to be used in the next step.

7.3. Pose Graph Optimization

We use a 7 DoF pose graph optimization[12] to optimize camera poses, one degree of freedom for scale, three for rotation and three for translation. We add edges between consecutive keyframes, edges in covisibility graph that has > 100 shared 3D points, and the loop edge as constraints. The residual for an edge between keyframe K_i and keyframe K_j is defined as:

$$r_{i,j} = log_{Sim(3)}(S_{ij} S_{jw} S_{iw}^{-1}) \tag{1}$$

, where S_{ij} is the relative Sim(3)[12] transformation from K_j to K_i computed from stored poses while setting scale equal to 1, except for the loop closing edge where S_{cl} is computed from new poses while setting scale equal to s_{cl} in the previous step. $log_{Sim(3)}$[12] transforms an element in Sim(3) into $r_{i,j}$ which is a vector in $\mathbb{R}^7$, and we optimize keyframe poses by minimising the cost:

$$C = \sum_{i,j} r_{i,j}^T \Lambda_{i,j} r_{i,j} \tag{2}$$

, where $\Lambda_{i,j}$ is the information matrix of the edge, and set to identity as in [13].

After the similarities S_{iw}^{corr} are corrected, we use the stored pose T_{iw} to correct the position of a map point x_j by

$$x_j^{corr} = (S_{iw}^{corr})^{-1} T_{iw} x_j \tag{3}$$

, and then S_{iw}^{corr} is transformed back to a corrected pose T_{iw}^{corr} by setting the translation to be its translation divided by its scale, leaving the rotation unchanged.

8. Experiments

We experiment with our method on a diverse set of datasets containing both indoor and outdoor scenes. We compare our method with the ORB-SLAM2[8] and COLMAP[11]. For ORB-SLAM2, we use the monocular setting with default parameters with the given camera parameters. For COLMAP, we adopt the automatic reconstruction pipeline with the video sequence and a single camera option without camera parameters. Following previous work, we evaluate the accuracy of the camera trajectory using the Absolute Trajectory Error (ATE). To evaluate the 3D structures, we visualize the point clouds. We also perform ablation study of our key components.

8.1. Datasets

TUM-RGBD[14] is a large-scale indoor dataset with RGB-D images captured from Microsoft Kinect sensor and ground truth trajectory for evaluating visual odometry and visual SLAM systems. We only use the monocular RGB images with sensor resolution 640×480.

KITTI[4] is an outdoor dataset for autonomous driving and other challenging real-world computer vision benchmark including stereo, optical flow, visual odometry, etc. We use the monocular gray scale images from 'images_0' with resolution 1226×370.

8.2. TUM-RGBD

On TUM-RGBD dataset, results for Absolute Trajectory Error (ATE) and running time are shown in Table 1. The visualization of trajectories and 3D structures are shown in Figure 3 and Figure 4 respectively. Our method can track all five scenes, while COMLAP fails on 'fr1/room' and ORB-SLAM2 fails on 'fr1/room' and 'fr1/desk2', showing the robustness of our method. As for 3D structure, our method performs much better than ORB-SLAM2 and has comparable performance with COLMAP with 3-10 times faster thanks to keyframe selection module and local refinement. Compared with ORB-SLAM2, our method also achieves comparable efficiency while using the Superpoint features.

8.3. KITTI

Besides the indoor scenes, we also experiment on KITTI benchmark to test the loop closure capability. Out of five experimented sequences, 'seq03' and 'seq10' have no loop closure point, 'seq07' and 'seq09' have one loop closure point, and 'seq05' has three loop closure points, which is the hardest case. Absolute Trajectory Error (ATE) and running time are shown in Table 2. Our method achieves the best result on 'seq03' and 'seq05' and 'seq09', and contains a minor gap to the best result on 'seq07' and 'seq10'. As shown in Figure 5, ORB-SLAM2 cannot perform the correct loop closure on 'seq09' while our method successfully detects and close the loop. Comparison of 3D structure is shown in 6, our method can reconstruct more details than ORB-SLAM2. Furthermore, when compared with COLMAP video mode, our method performs better in terms of both accuracy on ATE and efficiency in most cases, which runs 10-15x faster.

8.4. Reconstruction Statistics

Following the COLMAP convention [11], we record the mean number of observations per keyframe and mean track length, and additionally, the keyframe ratio. The results are shown in Figure 7. COLMAP achieves a 100% keyframe ratio as it treats every image as a keyframe, while ORB-SLAM2 and our method achieve a much lower keyframe ratio. For the mean number of observations, COLMAP and ORBSLAM2 extract significantly more features than ours and therefore have more 3D map points to track. In terms of mean track length, COLMAP achieves the highest length since COLMAP does not drop non-keyframe, and the overlapping between consecutive images becomes larger.

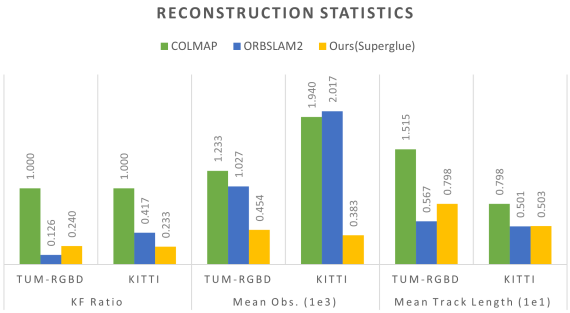


Figure 7: Reconstruction statistics with keyframe ration, mean observations per image, and mean track length.

8.5. Ablation Study

Components. We perform an ablation study on KITTI sequence 05 to study the function of each module in our pipeline. With only the initialization plus PnP to track the new keyframes, the 3D structures of the whole scene quickly collapse after the first turn, as shown in Figure 8 (a). By adding local BA and global BA, the 3D points do not collapse to the same position, and the visual odometry can track the whole sequence while suffering from the accumulative scale drift. (c) takes additional local map into account, and the new keyframe will be dragged to the previous map if sharing the covisibility with the old keyframes. However, such an operation will easily break the continuity of the trajectory. (d) introduces the loop closure on Sim(3) space to solve the scale drift problem and close the loop.

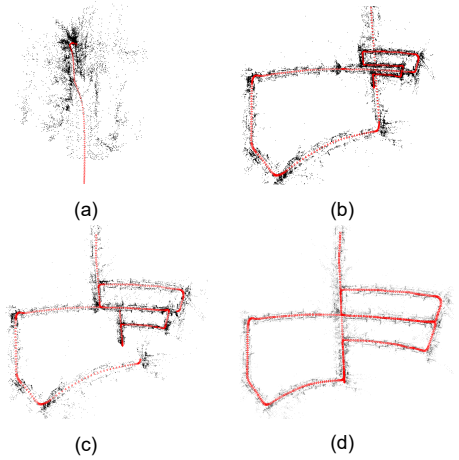
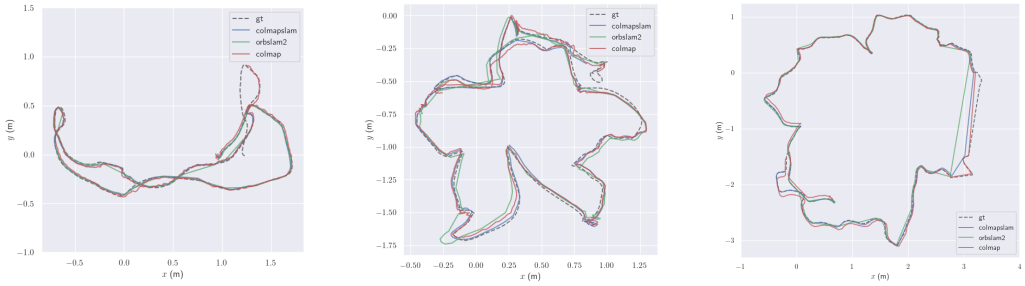


Figure 8: Ablation study of different components on KITTI Seq05. (a) PnP only, (b) PnP + BA, (c) PnP + BA + Local Map Tracking, and (d) PnP + BA + Local Map Tracking + Loop Closure.

SE(3) vs Sim(3) for Loop Closure. We evaluate two constraints of loop closure on either SE(3) space or Sim(3) space. For the SE(3), it only optimizes the pose on SE(3)

Method	fr1/desk		fr1/desk2		fr1/room		fr1/plant		fr2/desk	
	rmse	time	rmse	time	rmse	time	rmse	time	rmse	time
COLMAP	0.044	26	0.112	31	X	69	0.061	200	0.059	1275
ORBSLAM2	**0.040**	1	X	1	X	2	0.119	2	**0.030**	2
Ours (superglue)	0.057	8	**0.109**	9	**0.193**	21	**0.042**	25	0.037	99

Table 1: ATE w.r.t. full transformation (unit-less) and processing time (min) on TUM-RGBD.



(a) fr1/desk (b) fr1/plant (c) fr2/desk

Figure 3: Trajectory visualization on TUM-RGBD

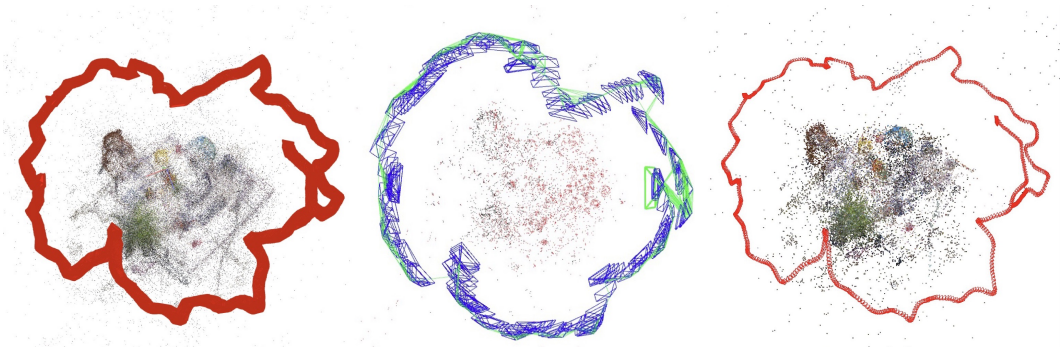
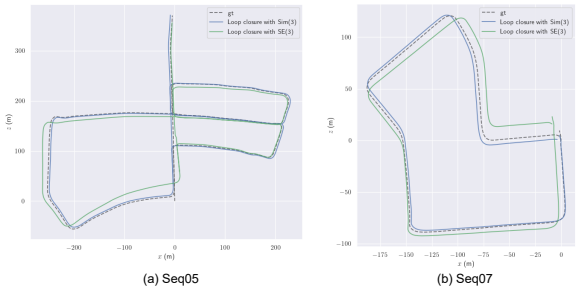


Figure 4: 3D structure visualization on TUM-RGBD fr2/desk (Left: COLMAP, middle: ORB-SLAM2, right: Ours)

Method	seq03		seq05		seq07		seq09		seq10	
	rmse	time	rmse	time	rmse	time	rmse	time	rmse	time
COLMAP	X	153	X	1178	5.198	216	5.700	263	6.150	189
ORBSLAM2	1.027	2	4.338	5	**3.726**	2	53.635	3	**6.138**	3
Ours (superglue)	**0.648**	10	**3.117**	83	3.775	12	**4.494**	22	6.166	18

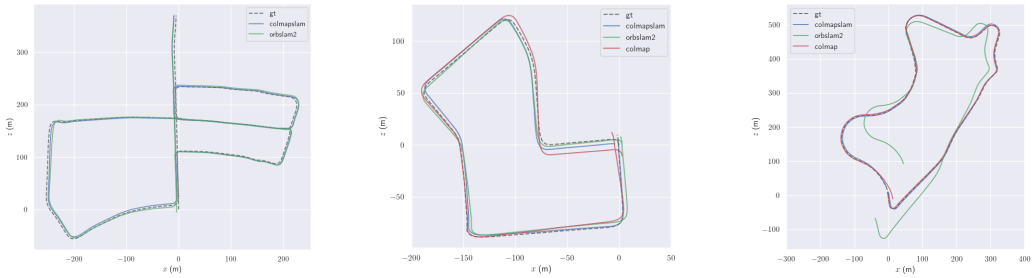
Table 2: ATE w.r.t. full transformation (unit-less) and processing time (min) on KITTI.

with 6 DoF while Sim(3) adds an additional scaling factor than SE(3) and contains 7 DoF, assuming that there may be scale drift. We show both quantitative and qualitative results in Figure 9 and Table 3. As shown in the table, performing PGO on Sim(3) significantly improves the results, which demonstrates the effectiveness of solving the scale drift problem. Performing SE(3) optimization can also close the loop but drag the whole trajectory to an incorrect shape. Also, for SE(3) the cost in PGO can only be minimized to a certain degree and stuck at local minima while using Sim(3) the cost always converges to zero very quickly.



(a) Seq05 (b) Seq07

Figure 9: Ablation study of loop closure constraints on KITTI Seq05.



(a) Seq05 (b) Seq07 (c) Seq09

Figure 5: Trajectory visualization on KITTI

Figure 6: 3D structure visualization on KITTI07 (Left: COLMAP, middle: ORB-SLAM2, right: Ours)

Loop Closure	Seq05	Seq07	Seq09
Ours-SE(3)	12.161	9.065	20.450
Ours-Sim(3)	**3.117**	**3.775**	**4.494**

Table 3: ATE w.r.t. full transformation (unit-less) on KITTI for loop closure constraints.

Time We analyze the time of different parts of our system on kitti sequence 07 as in Figure 10. Inside check keyframe part, 33% is spent on extracing Superpoint, 47.8% on SuperGlue, and the rest on NetVLAD. We do a global BA every 20 keyframes. Note that for longer sequence, the ratio of time took by global BA will grow, and users can persue higher speed by setting the global BA frequency smaller.

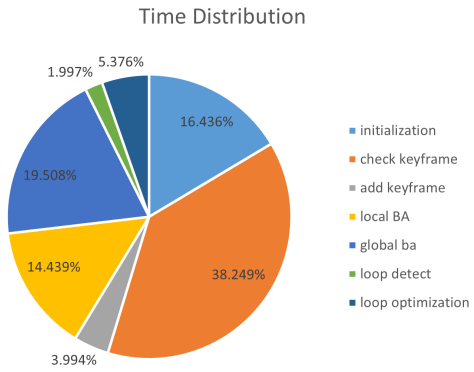


Figure 10: Runtime distribution on KITTI Seq07.

9. Work Distribution

Our distributions are listed as follows:

- Kexin: local/global BA, reproduction of ORB-SLAM2
- Weirong: framework, visualization, BA, experiments
- Yidan: keyframe selection, experiments
- Yifei: initialization, local/global BA, loop closing, profiling and speed optimization.

10. Conclusion

In this project, we develop COLAM, an offline python SLAM based on COLMAP, which achieves comparable reconstruction performance with fewer track failures and a much shorter runtime compared with COLMAP and ORBSLAM. The system is built taking the robustness of COLMAP, the flexibility of Hloc, and the efficiency of ORBSLAM. Starting with COLMAP pipeline, we introduce the keyframe selection, covisibility graph, and loop closure to improve the speed and alleviate scale drift of the vanilla COLMAP. Extensive experiments are conducted on standard SLAM benchmarks, including TUM-RGBD (indoor) and KITTI (outdoor) datasets. In addition, we perform an ablation study on different components and different constraints for loop closure.

References

[1] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 3 2022. 4321

[2] Relja Arandjelovic, Petr Gronát, Akihiko Torii, Tomás Pajdla, and Josef Sivic. Netvlad: CNN architecture for weakly supervised place recognition. *CoRR*, abs/1511.07247, 2015. 4321, 4323

[3] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. *CoRR*, abs/1712.07629, 2017. 4322

[4] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 4324

[5] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of The Optical Society of America A-optics Image Science and Vision*, 4:629–642, 1987. 4323

[6] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal Of Computer Vision*, 81:155–166, 2009. 4322, 4323

[7] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *CoRR*, abs/1502.00956, 2015. 4321, 4323

[8] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017. 4321, 4323

[9] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 4321, 4322

[10] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 4322

[11] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4321, 4322, 4323, 4324

[12] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. Scale drift-aware large scale monocular SLAM. In Yoky Matsuoka, Hugh F. Durrant-Whyte, and José Neira, editors, *Robotics: Science and Systems VI, Universidad de Zaragoza, Zaragoza, Spain, June 27-30, 2010*. The MIT Press, 2010. 4323

[13] Hauke Malte Strasdat. Local accuracy and global consistency for efficient slam. 2012. 4323

[14] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 4323