Team 4: NAVARCH 568 Final Project Report: SURF-SLAM

Snehal Chavan, Nadha Gafoor, Audrow Nash, Ming-Yuan Yu, and Xinzhe Zhang

I. Introduction

In the past few decades, the area of mobile robotics and autonomous systems has attracted substantial attention from researchers all over the world. Fully autonomous robots have been the dream of researchers over many years. Mobile robotics has applications in various fields such as military, medical, space, entertainment and domestic appliances fields. In those applications, the robots are expected to perform complicated tasks that require navigation in complex and dynamic indoor and outdoor environments without any human input. The robot has to build a map of an environment and at the same time use this map to deduce its location. This problem is called Simultaneous Localization and Mapping (SLAM.)

In recent years the visual SLAM problem has become a focus of the robotics and vision communities. This has been because the cameras are cheaper and simpler to configure as compared to range finders based on LiDAR, RADAR etc. There are two different types of cameras that can be distinguished: Stereo (which involves two cameras) and mono (with a single camera) cameras. One of the main reasons that monocular SLAM is used and researched is because the hardware needed to implement it is much simpler, i.e, it is cheaper and physically smaller than other systems. Like any other SLAM problem, MonoSLAM [3] also consists of steps like finding Visual Odometry, and Loop Closure. There are different ways to approach each of this step which are discussed in the next section.

We propose a visual SLAM algorithm *SURF-SLAM*, which is heavily inspired by ORB-SLAM [12]. Our code is freely available on Github, `https://github.com/audrow/navarch-568-proj/`

II. Survey of related work

Monocular SLAM (MonoSLAM) introduced by Andrew et. al [3] is a possible solution to the monocular camera visual SLAM problem, which obtains a probabilistic 3D Map of camera poses and features using an Extended Kalman Filter (EKF). The sparse feature and Gaussian constant motion assumptions of MonoSLAM reduces its performance under large environment and dynamic motion conditions. MonoSLAM also shares the problem of high computational cost with little new information and the accumulation of linearization errors. On the other hand keyframe-based approaches [15], [9] estimate the map using only selected frames (keyframes) allowing to perform more costly but accurate bundle adjustment (BA) optimizations, as mapping is not tied to frame-rate.

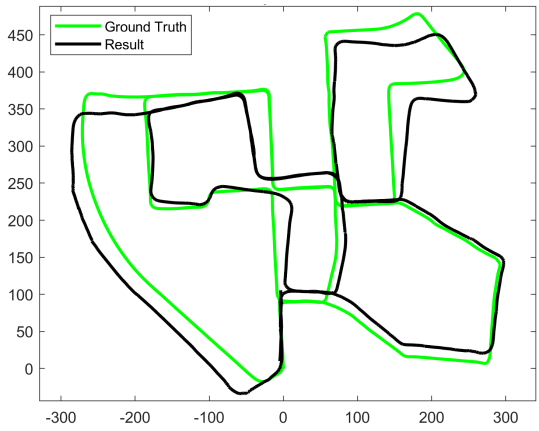

Fig. 1. Result of SURF-SLAM on Sequence 00 in the KITTI dataset.

Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates. [15] and [7] talk about the solutions to this optimization problem. The first attempt to apply bundle adjustment in visual SLAM was carried out by E. Mouragnon in [11].

One of the many approaches to solve the problem of visual odometry was introduced in the work of Klein and Murray [10], known as (PTAM). It proposes to split tracking and mapping into two separate tasks, processed in parallel threads on a dual-core computer. This allows the use of computationally expensive batch optimization techniques not usually associated with real-time operation. One other advantage of PTAM is the use of keyframes. It tracks the camera poses of each frame with existing map points, and updates map by local BA using selected frames with large information. It was proved by [10] that PTAM has better performance than EKF with the allowance of denser features. Unfortunately several factors severely limit its application: lack of loop closing and adequate handling of occlusions, low invariance to viewpoint of the relocalization and the need of human intervention for map bootstrapping.

These challenges are overcome by combining the methods of place recognition in [5], loop closing in [14], and the use of covisibility information in [13]. A visual place recognition is performed using bag of words obtained from FAST+BRIEF features in [5]. Whereas in [14], Hauke and Andrew present a new pose-graph optimization technique which allows for the efficient correction of rotation, translation and scale drift at loop closures. The algorithm proposed in [13] automatically builds a suitable connected graph of

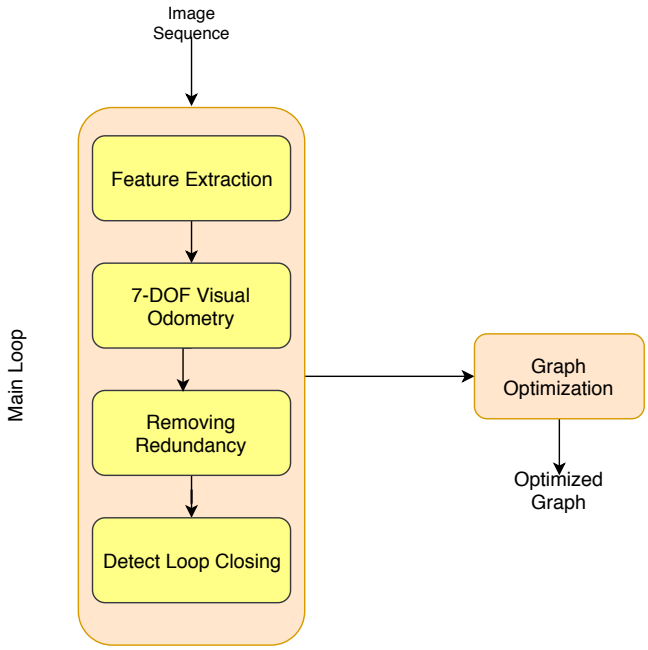

Fig. 2. SURF-SLAM system overview showing main modules of the algorithm. The main loop is called iteratively for each new frame.

keyposes and constraints, dynamically selects inner and outer window membership and optimizes both simultaneously. In extensive simulation environments their method approaches the accuracy of offline bundle adjustment while maintaining constant-time operation, even in the hard case of very loopy monocular camera motion.

III. Problem Statement

Through this project, it is intended to create a keyframe based visual SLAM algorithm with following modules:

- Feature extraction: To obtain robust representation of images as a set of feature points that are rotation, scale and viewpoint invariant.
- Tracking: To get relative motion between frames (visual odometry) with Structure From Motion (SFM).
- Reduce Redundancy: To remove redundant keyframes and maintain an optimal graph structure.
- Loop Closing: To search for loops with every new keyframe.
- Optimization: Optimize the graph with all the visual odometry factors.

IV. Our Approach

The system overview of our implementation is shown in Fig. 2.

A. Feature Extraction

To get a compact representation of images we used feature points and their descriptors. We extracted SURF [1] feature points from the image. It is based on the sum of Haar Wavelet response around the point of interest. It is invariant to scale, rotation and illumination. The descriptors are of length 64. The same features is used by odometry as well as loop closure. As shown in Fig. 3, we match the correspondences in between two frames. The feature points of the first frame are shown in red circles whereas the feature points in the second frame are shown in green crosses. The matched correspondences are drawn using yellow lines. These matched correspondences are further used to calculate the position of the camera in 3D co-ordinate system using structure from motion.

B. 7-DOF Pose

It is shown in [14] that rather than using a 6-DOF pose in $\mathcal{SE}(3)$, a 7-DOF representation in $\mathcal{SIM}(3) = \mathbb{R}\times\mathcal{SE}(3)$ can greatly reduce the scale drifting problem for visual SLAM with monocular camera . We adapt the same approach and represent the camera pose of frame i relative to the first frame by a 7-DOF similarity transform

$$T_i = \begin{bmatrix} e^{\sigma_i} R_i & t_i \\ 0_{1\times 3} & 1 \end{bmatrix} \in \mathcal{SIM}(3)$$

where $\sigma_i \in \mathbb{R}$, $R_i \in \mathcal{SO}(3)$ and $t_i \in \mathbb{R}^3$. The pose of the first frame is set to be a 4×4 identity matrix.

Given a similarity transform T_i, there exist a function f and a *pose vector* p_i such that

$$p_i = f(T_i) = \begin{bmatrix} \sigma_i \\ u_i \\ t_i \end{bmatrix}$$

where u_i is the rotation vector associated with the rotation matrix R_i, which can be calculated using Rodrigues' rotation formula. If the angle of rotation is always normalized to the range $[0, \pi]$, then f is bijective and the inverse f^{-1} exists.

By working directly with p_i, we can use the minimum number of parameters (seven) during optimization in Section IV-F instead of twelve in the case of working with T_i.

C. Pose Graph and Visual Odometry

The map is represented as a pose graph as many of the other approaches [12], [4], [8]. Each node is a keyframe associated with a pose, which is connected to other nodes with visual odometry factors.

Let i be the newly observed frame. First, feature points are extracted and matched between two consecutive frames $i-1$ and i. Second, the fundamental matrix F_i^{i-1} is estimated using RANSAC and the 7-DOF visual odometry $T_i^{i-1} \in \mathcal{SIM}(3)$ can then be obtained. In our implementation, both F_i^{i-1} and T_i^{i-1} are estimated using the build-in functions available in MATLAB. If the algorithm fails to find a valid visual odometry T_i^{i-1} due to lack of matched feature points, a constant velocity model is used by setting $T_i^{i-1} = T_{i-1}^{i-2}$. Third, the pose of the frame i relative to the first frame is initialized by

$$T_i = T_i^{i-1} \cdot T_{i-1}$$

and a connection between frame i and $i-1$ is added to the graph. Finally, if there exists a valid visual odometry T_i^j between non-consecutive frames i and $j \in \{i-n_m, \ldots, i-2\}$, where the constant n_m is usually a small integer, connection between i and j is also added to the graph.

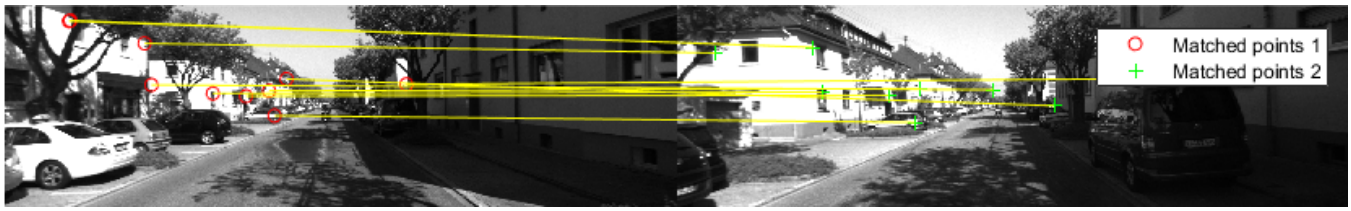

Fig. 3. Matched map-points in two frames using SURF features.

Note that since we only utilize a single camera, the scale of the visual odometry is agnostic. In other words, t_i^{i-1} is always normalized to have a length of 1 and σ_i^{i-1} is always 0. These variables will be refined in the optimization step described in Section IV-F.

D. Remove Redundancy

In SURF-SLAM, we keep inserting keyframes and then we remove the redundant ones. This is called *keyframe culling*. If we do not perform keyframe culling, the complexity of the bundle adjustment grows with the number of keyframes added. Hence, it is important to perform keyframe culling. Provided the map is finite, this method will take care that the number of keyframes converge and that they'll always be bounded. We discard keyframes that share 90% of their map points with at least three other keyframes. This helps us in maintaining a compact graph.

E. Detecting Loop Closure

The graph built in Section IV-C only gives a rough estimate of the camera poses and is far from perfect. It is prone to drift in all the 7 DOFs since it is based solely on visual odometry using monocular camera [14]. One way to mitigate drift is to *close the loop*, which adds extra factors between frames that are potentially very far away from each other.

To detect loop closures we utilize Bag-of-Words (BoW), a common and effective way for place recognition [5]. A large set of SURF descriptors are first being extracted from the dataset a priori, and 10% of them are grouped into n_c clusters using the $k-$means algorithm. The centroids of these n_c clusters are then being used as *visual codewords*. For every newly observed frame i, the frequency of the appearance of each codeword is calculated using a $k-d$ tree algorithm. The resulting histogram $h_i \in \mathbb{R}_+^{n_c}, \sum_{k=1}^{n_c} h_i(k) = 1$, is the BoW representation of the frame and is used to propose a loop closure.

We compare h_i to h_j for all $j \in \{1, \dots, i-l\}$, where the hyperparameter l is the minimum number of frames for a valid loop closure, using the χ^2 distance metric

$$d^2(h_i, h_j) = \sum_{k=1}^{n_c} \frac{(h_i(k) - h_j(k))^2}{h_i(k) + h_j(k) + \epsilon}$$

where ϵ is a small number for numerical stability. Then we try to match features between frame $j^* = \text{argmin}_j\, d^2(h_i, h_j)$ and frame i. If the number of matches is above a certain threshold, the proposed loop closure between i and j^* is confirmed. A connection between frame i and j^* is then added to the graph with the visual odometry factor $T_{j^*}^i$. An example of detected loop closures is shown in Fig. 5.

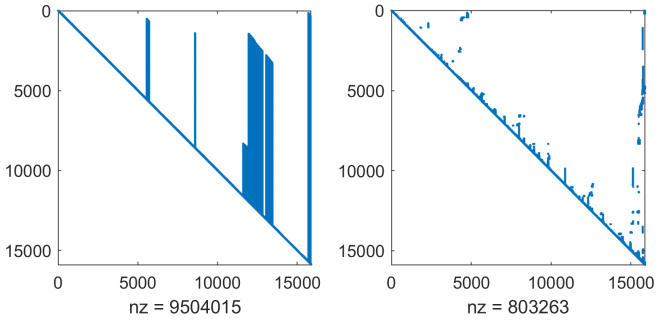

Fig. 4. Cholesky decomposition L of the information matrix $J^\top J$ where `nz` is the number of nonzero elements. The left plot shows nonzeros elements of L without column reordering, and the right plot shows the reordered version of L which is much sparser.

F. Graph Optimization

The pose graph is further optimized offline with the all the visual odometry factors found in Section IV-C and IV-E.

All the poses T_i and visual odometry factors T_j^i are first converted to pose vectors described in Section IV-B. The optimization problem can then be formulated as a nonlinear least squares problem.

$$p^* = \underset{\hat{p}}{\text{argmin}} ||f(T_j^i) - \hat{p}_j^i||_2^2$$

where $p = \begin{bmatrix} p_1^\top & \dots & p_N^\top \end{bmatrix}^\top \in \mathbb{R}^{7N}$, $p_j^i = f(f^{-1}(p_j) \cdot f^{-1}(p_i))$, and N is the number of keyframes in the graph. We use MATLAB's symbolic toolbox to calculate the measu rement Jacobian J analytically and use Levenberg-Marquardt algorithm to solve the problem.

$$(J^\top J + \lambda \cdot \text{diag}(J^\top J))\delta = J^\top (f(T_j^i) - \hat{p}_j^i)$$
$$\hat{p} \leftarrow \hat{p} + \alpha\delta$$

where α is the step size at each iteration. The information matrix $J^\top J$ can also be sparsify using column reordering method COLAMD [2], as shown in Fig. 4.

V. Results

A. Localization Result with KITTI Database

To quantitively evaluate the performance of our algorithm, we choose sequence 00 from KITTI's odometry benchmark [6]. The camera frames and ground truth information are

	ORB-SLAM	SURF-SLAM (ours)
RMSE (m)	6.68	23.65

TABLE I

COMPARISON OF SEQUENCE 00 IN THE KITTI DATASET [6].

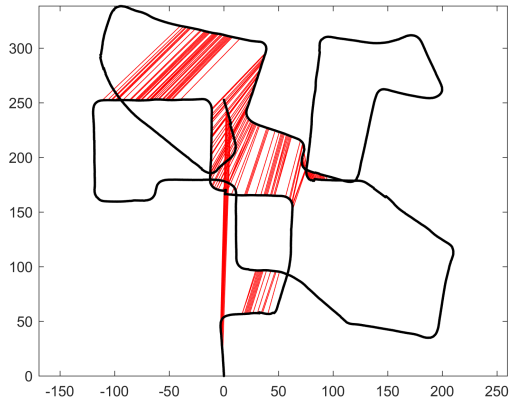

Fig. 5. Visual odometry (black) and detected loop closures (red).

provided by the database, and a reasonable amount of loop closure is included in this test sequence.

Since the monocular camera result doesn't give correct scaling and orientation compared to the ground truth, we first use a similarity transform described in Section IV-B on our result to align with the ground truth.

Fig. 1 shows the result of our algorithm on KITTI sequence 00. As a comparison to Fig. 5, the loop closure was fixed after the optimization and the algorithm gives a reasonable result. The RMSE of the xyz position is calculated, and compared to the result of ORB-SLAM in Table I.

VI. DISCUSSION

A. Analysis of results

Compared to ORB-SLAM, our approach has a relative high RMSE error, as shown in Table I. Nonetheless, SURF-SLAM yields reasonable results. Visual odometry gives good tracking between consecutive frames, BoW provides credible loop closures, and the optimized graph looks reasonably good, as shown in Fig. 1. We achieved a reasonably good loop closure with BoW that prevented the large drifting of odometry. The odometry before and after loop closure can be seen in Fig 5 and 1

B. Limitations and Future Directions

Our implementation has several limitations that stop it from being as generally useful as the open source code from [12].

1) Speed: Our implementation does not run in real-time, about 1 to 2 frames per second, and there are several reasons for this. First, we are using SURF instead of ORB features. [12] states that they found ORB features to be around ten times faster to compute on the KITTI dataset images. Second, our implementation is in MATLAB, and could be sped up significantly by implementing key components in a lower-level language such as C or C++. Also in the future it is possible to have a iterative implementation similar to iSAM [8].

2) Accuracy: The relatively high error shown in TABLE I can be the result of only optimizing the graph after the whole sequence is finished, as oppose to doing local BA frequently mentioned ORB-SLAM. We expect to see lower RMSE values with local graph optimization but unfortunately we do not have time to implement it during the time window of this project.

VII. CONCLUSION

As a course project, we have come to understand visual SLAM through our implementation. We also see the challenges such as implementing the algorithm to run in real-time and performing graph optimization.

In this project, we proposed a monocular visual SLAM algorithm, *SURF-SLAM*, and have an implementation in MATLAB. Although both the speed and accuracy are not as good as ORB-SLAM, SURF-SLAM is still unarguably a complete package for tracking with visual odometry, detecting loop closure, and optimizing the graph.

REFERENCES

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[2] Timothy A Davis, John R Gilbert, Stefan I Larimore, and Esmond G Ng. Algorithm 836: Colamd, a column approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 30(3):377–380, 2004.

[3] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[4] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[5] Dorian Gálvez-López and Juan D Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[6] 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.

[7] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision second edition. *Cambridge University Press*, 2000.

[8] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[9] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234. IEEE, 2007.

[10] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[11] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3d reconstruction. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 363–370. IEEE, 2006.

[12] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[13] Hauke Strasdat, Andrew J Davison, JM Martìnez Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2352–2359. IEEE, 2011.

[14] Hauke Strasdat, JMM Montiel, and Andrew J Davison. Scale drift-aware large scale monocular slam. *Robotics: Science and Systems VI*, 2, 2010.

[15] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer, 1999.