The software tool for comparison and configuration of nonlinear optimization techniques in ORB-SLAM

Boris Makaev and Ilya Afanasyev
Innopolis University, Innopolis, 420500, Russia
{b.makaev, i.afanasyev}@innopolis.ru

***Abstract*—This paper presents the study on development of a software tool with a graphical user interface (GUI) to compare nonlinear optimization methods in ORB-SLAM library. It uses g2o graph optimization framework to solve five optimization problems while tracking SLAM: (a) Global Bundle Adjustment (GBA), (b) Local Bundle Adjustment (LBA), (c) Relative Simulation Optimization (SO), (d) Pose Graph Optimization over SO(3) Constraints, and (e) Essential Graph Optimization. By default, ORB-SLAM solves these problems using Levenberg-Marquardt method, but our tool allows to select optimization methods between Levenberg-Marquardt, Powell's Dogleg and Gauss-Newton algorithms to result in the best configuration solution when working with a video dataset. The developed application allows a user to configure the optimization tasks in ORB-SLAM and run an experiment to observe the performance evaluation of the current configuration. In addition, the ORB-SLAM functionality has been extended using Ceres solver optimization framework. The tests of various datasets showed that the developed application is a convenient and effective tool for comparing different solutions and finding the fastest and most accurate ORB-SLAM configuration.**

I. Introduction

Nowadays, computer vision (CV) algorithms are used in many areas of robotics, such as autonomous cars, unmanned aerial vehicles (UAVs), mobile robot navigation and many others. The number of such applications that use CV techniques continues to grow rapidly to the tremendous work done by many studies. Many algorithms, such as algorithms for Simultaneous Localization and Mapping (SLAM) and Visual Odometry were implemented in order to allow robotic systems to perform visual tracking, assess their position and trajectory relative to the world, build 3D map in which they are located. The data for these algorithms come from sensors such as a camera, inertial measurement units (accelerometers, gyroscopes, etc.). But the problem is how to increase the performance of such algorithms. They perform complex mathematical calculations, and it is very important to choose the right methods to solve specific problems. In particular, some SLAM problems can be formulated using nonlinear optimization. Such problems include minimization of reprojection error, bundle adjustment, etc. Various methods for solving nonlinear optimization problems can be performed differently depending on the type of problem that they are trying to solve. Thus, it is important to compare different optimization methods in order to choose the most effective.

This paper attempts to establish a connection between the choice of nonlinear optimization methods in SLAM and the overall performance of SLAM algorithms. As the case study, ORB-SLAM [1], [2] was chosen to be the base SLAM system, which implements its nonlinear optimization tasks using the graph optimization framework (g2o). The g2o optimization framework implements three nonlinear optimization methods: Levenberg-Marquardt, Gauss-Newton, and Dogleg. This paper attempts to expand the selection of optimization methods for ORB-SLAM by implementing optimization tasks using Ceres solver optimization framework. We are looking for answers to two main questions:

- Does the choice of the optimization methods affect the overall performance of the ORB-SLAM library?
- Does the choice of the optimization framework that implements the solution of optimization problems affect the performance of ORB-SLAM?

The goals that we set to achieve were defined as follows:

- To develop a Comparison application that allows the user to customize optimization methods in ORB-SLAM.
- To extend ORB-SLAM with new functions, i.e. allow solving optimization problems by using Ceres-solver optimization framework.
- To conduct experiments in order to assess and compare different nonlinear optimization methods.

Our main contribution is the developed application, which allows the user to compare different methods of nonlinear optimization in ORB-SLAM, which can also be expanded to bring new functionality.

The Section II defines important concepts and reviews related literature. The Section III describes how we compare ORB-SLAM configurations with various nonlinear optimization methods. This section introduces the ORB-SLAM extensions by implementing nonlinear optimization tasks using the Ceres solver optimization framework. Moreover, this section introduces the Comparison application written in Qt that allows the user to configure ORB-SLAM, save the performance results and visualize them. The Section V evaluates the operation of the Comparison application and then compares results of various experiments which show how the choice of nonlinear optimization methods affects the performance of ORB-SLAM. The Section VI concludes this paper, summarizing results and discussing future works that could be done to extend the functionality of the application.

978-1-7281-3021-7/19/$31.00 ©2019 IEEE
DOI 10.1109/DeSE.2019.00095

II. Related Works

Nocedal and Wright [3] describe two classes of existing nonlinear optimization methods that are conceptually complement each other. The *Line Search* methods first try to find a descent direction along which the objective function will be reduced, and then compute a step size, which decides how far to move in this direction. The descent direction can be calculated by various methods, such as gradient descent, Newton's method, and Quasi-Newton method. The step size can be determined either accurately or inaccurately. The *Trust-Region* methods approximate the objective function using a model function (often quadratic) over a subset of the search space known as the trust region. If the model function succeeds in minimizing the true objective function the trust region is expanded, or otherwise it is contracted and the model optimization problem is solved again.

Ceres solver [4] is nonlinear optimization framework that is developed and maintained by Google. It is written in C++ as almost all solvers, because it requires high-speed calculations without great overhead. Currently, Ceres solver can solve (1) nonlinear least squares problems with bounds constraints, and (2) general unconstrained optimization problems.

Giorgio Grisetti et al. [5], [6] presented their new g2o optimization framework for solving nonlinear least squares problem in graph or hyper-graph space. They state that this method of solving the problem is convenient when trying to optimize SLAM solutions or other robot systems. g2o implements two algorithms for solving nonlinear least squares optimization problems: Levenberg-Marquardt and Gauss-Newton as well as Powell's dogleg.

There are many different types of solutions for SLAM problems that can be outlined. Jakob Engel et al. [7] present their taxonomy of currently popular types of methods that solve problems of SLAM and Visual Odometry. They stated that methods for solving the SLAM problem can have some characteristic properties:

- **Sparse + Indirect**: generate 3D geometry from a set of keypoint matches. Examples of such works: monoSLAM [8], PTAM [9], ORB-SLAM [10].
- **Dense + Indirect**: evaluate 3D geometry from dense optical flow field. Example: [11].
- **Dense + Direct**: these methods use photometric error and geometric shape before evaluating the geometry. Examples are DTAM [12], REMODE [13] and LSD-SLAM [14].
- **Sparse + Direct**: these methods include a photometric error, but without use of geometric prior. See: DSO [7].

According to Durrant-Whyte and Bailey[15], **Simultaneous Localization and Mapping** is the method of building or updating a map of an unknown environment while simultaneously tracking the location of the agent in the constructed map. Thus, solving the SLAM problem allows a mobile robot with sensors such as a camera or laser to construct a map of the environment, as well as determine its current location.

Not so long ago, the new study on visual-inertial SLAM systems was provided by Haoimin Liu et al. [16]. They have developed and implemented the new SLAM system, which they called **Incremental, Consistent and Efficient Bundle Adjustment** (ICE-BA).This is a completely fledged SLAM system for which they wrote their own optimization solver to optimize nonlinear cost functions. **Bundle Adjustment** [17] is the problem of simultaneously improving the scene coordinates, relative motion parameters, and optical characteristics of the camera(s) employed to acquire images using the optimality criterion, which includes the corresponding image projections of all points. Existing solutions include **ICE-BA for Visual-Inertial SLAM** [18], that uses the sparseness and a unique matrix structure to optimize the sliding window-based bundle adjustment and offers a new marginalization strategy to improve global consistency.

In 2015, Raul Mur-Artal et al. [1] introduced their newly developed feature-based monocular SLAM system. It implements such features as tracking, mapping, relocalization and loop closing. *Tracking thread* localizes the camera with each frame and decides when to insert a new keyframe. *The local mapping thread* processes new keyframes and performs local bundle adjustment to achieve optimal reconstruction in the environment of the camera pose. Local mapping is also responsible for selecting redundant keyframes. *Loop closing thread* searches for loops with each new keyframe.

Often, researches only look for testing their developed SLAM algorithms outside of a robot environment. Because, it is much simpler and more convenient to test their prototype algorithms on existing datasets. Thus, researchers from all over the world create various datasets that can contain (1) image sequences for mono/stereo camera, (2) readings and measurements with IMU, (3) ground truth trajectory for evaluating SLAM algorithm performance, and (4) calibration files. Known datasets include [19], [20], [20], [21], [22]. The performance and accuracy of vision-based SLAM algorithms were evaluated in the papers [23], [24], [25], [26] for indoor and [27], [28] for outdoor applications, where camera-based trajectories of mobile platforms were analyzed and compared.

III. Methodology

The approach we are developing to compare the selected nonlinear optimization methods is the approach of changing existing SLAM system. To successfully solve this problem, it was decided: (1) to select a suitable existing SLAM library, (2) to analyse its default nonlinear optimization methods, (3) to choose and implement new nonlinear optimization methods using another optimization framework, (4) to develop a simple application with GUI that would make it easy to configure SLAM optimization methods, (5) to save metrics that evaluate the performance of SLAM algorithm with certain databases, and (6) to provide tools to visualize metrics, errors for selected configurations. Design approaches for these issues are described in this section.

A. ORB-SLAM

As discussed in the section II, for a successful SLAM system it is needed to choose both the front-end that extracts and processes features out of images, and the back-end that refines reprojection errors and solves other optimization problems, such as global or local bundle adjustment, etc. There are many various SLAM systems, feature extractors and optimization problem solvers that we can choose from. For this paper ORB-SLAM2 library[1] was selected as the base SLAM system. It has own ORB feature extractor that serves as the front-end of the SLAM system, and g2o graph optimization framework is used as the back-end that solves optimization problems.

Using ORB-SLAM as the starting point, it is necessary to determine what part of the system should be modified in order to compare nonlinear optimization methods. In order to perform comparison we need to change code in the back-end of the system, particularly in the optimization process. Due to the fact that ORB-SLAM uses g2o framework to solve optimization tasks, the main work should be related to changing the configuration of g2o solvers and other functions. The g2o optimization framework implements linear and nonlinear solvers that can be used as an operating method to perform the optimization process. In particular, g2o implements the following nonlinear optimization methods: Levenberg-Marquardt, Powell's Dogleg and Gauss-Newton methods.

The back-end of ORB-SLAM solves a number of optimization problems. We are interested in the nonlinear optimization problems that are present there. There are five nonlinear optimization problems that are solved during SLAM tracking: Global Bundle Adjustment (GBA), Local Bundle Adjustment (LBA), Pose Graph Optimization over SO(3) Constraints, Relative Sim(3) Optimization and Essential Graph Optimization. By default, ORB-SLAM solves these nonlinear optimization problems using Levenberg-Marquardt method. Thus, this means that to test other optimization methods, we need to call various g2o functions that are responsible for corresponding optimization methods.

B. The Extension of ORB-SLAM Optimization

As a part of the research, it was decided to extend the functionality of the ORB-SLAM library. The motivation behind this decision is that g2o optimization framework provides a limited amount of methods for solving nonlinear optimization problems. It was decided to use the Ceres solver as the second way to perform optimization tasks in ORB-SLAM. It has more flexible code base, that allows fine-tuning of optimization models and it implements more optimization methods that also can be easily configured. In addition, we need to expand ORB-SLAM in such way that we could easily evaluate the performance of ORB-SLAM using new optimization methods.

As it could be seen in the figure 1, there is the ORB-SLAM implementation that has default g2o optimization framework binding. The extension of this default architecture is the inclusion of the Ceres solver into the ORB-SLAM environment. Further, for each optimization task defined for ORB-SLAM, the corresponding model is written using the Ceres solver.

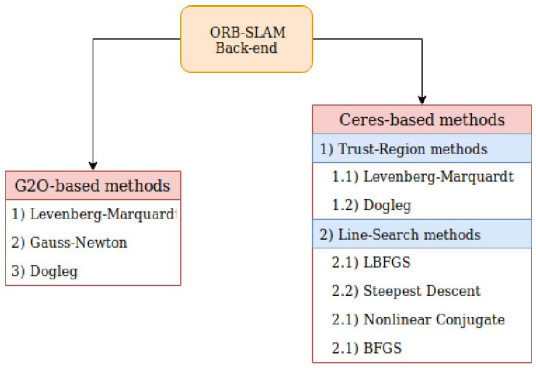

Fig. 1: The scheme of all possible optimization problems that could be solved in ORB-SLAM

Later, the parameters of the optimization problem solvers will be easily adjusted in the Comparison application, which would be also developed presented in the paper.

C. The Comparison application

This section describes one of the main goals of this paper: the development and implementation of the Comparison application that would allow to evaluate the influence of nonlinear optimization methods on the performance of ORB-SLAM.

First of all, it is necessary to determine the performance indicator of the nonlinear optimization problem. In fact, there may be several performance indicators in ORB-SLAM. The most obvious is the time taken to complete the optimization task. It is quite simple to assess the execution time of each optimization task by simply putting a timer inside the corresponding code, where the optimization function is called from. Another performance criterion that can be chosen is the overall error between a ground truth trajectory from a dataset and the reconstructed trajectory. Another important performance indicator is CPU consumption, since different optimization methods take different CPU resources in order to solve the corresponding tasks. These criteria were chosen to evaluate the performance of the nonlinear optimization problem.

The motivation for the design and development of such a framework is related to the nature of comparing various nonlinear optimization methods in ORB-SLAM. There are 5 optimization problems that are defined for solving in ORB-SLAM. Given the vast amount of optimization methods for each of these optimization problems, the problem arises: there are many combinations of different optimization methods used together. Compiling ORB-SLAM together with various optimization methods will take a lot of time. That is why the idea of generalizing the ORB-SLAM library for different nonlinear optimization methods and creating a GUI application to configure the SLAM system is really important.

The figure 2 shows the prototype structure of the Comparison application. The architecture of such an application will be discussed in the following order:

GUI application is the part of the desktop application that allows the user to set the necessary parameters, for exam-

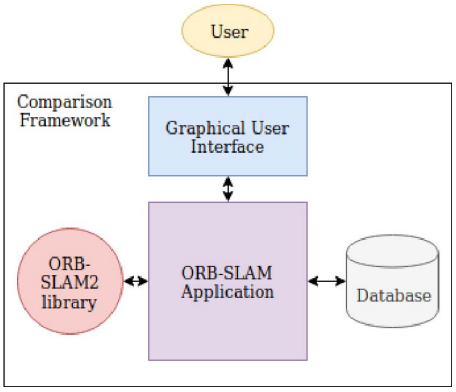

Fig. 2: High-level prototype design of Comparison framework

ple, which optimization method will be used in a particular optimization task; what dataset will be used to evaluate the current ORB-SLAM configuration; number of features, etc. When the user finally determines all the parameters that will be set, the experiment will be visualized. After the experiment, one can access the results of previous experiments, visualize and compare the evaluation metrics using visualization tools.

ORB-SLAM Application is core framework that uses ORB-SLAM2 library to process an image sequence (dataset) to produce the point cloud map and trajectory. Default examples are created for each type of dataset when building the ORB-SLAM2 library from source. Ideally, there is the need to implement one general application that would take different datasets.

ORB-SLAM2 Library is the core of ORB-SLAM that implements ORB feature extractor and the rest of the back-end. This paper defines that ORB-SLAM back-end will consist of g2o-based optimization methods and Ceres-based optimization methods. Each of the newly defined method must be controlled externally, particularly from higher level.

Database is the repository that would contain all the results of the ORB-SLAM experiments. Later, these results could be further visualized in GUI of the Comparison application.

IV. System Setup

We used the MSI GX60-1AC laptop with the following specifications: AMD A10-4600m **CPU**, 8GB DDR3 1600 MHz **RAM**, AMD Radeon HD7970M **GPU** (presented in the Table I). The operating system used is Ubuntu 18.04 LTS. The following sections will emphasize which software packages and libraries were installed on the laptop. All of these packages had to be installed before starting the development of the Comparative application and the solver extensions.

TABLE I: MSI GX60 laptop specification

MSI GX60-1AC	
Operating System	Ubuntu 18.04 LTS
CPU	AMD A10-4600m
GPU	Radeon HD7970M
RAM	8 GB DDR3

V. Comparison Application and Experiments

A. The Comparison Application

This section demonstrates the case study of the Comparison application step by step. The batch of experiments will be launched in order to get representative results that will subsequently be used to evaluate various ORB-SLAM configurations. They will be compared using metrics discussed earlier in the Section III, e.g. CPU consumption, time performance for various ORB-SLAM tasks in milliseconds, absolute pose error (APE) between generated trajectory and ground truth trajectory (also called reference), etc. This section is mainly evaluates the GUI application itself. The design issues and possible improvements will be discussed further.

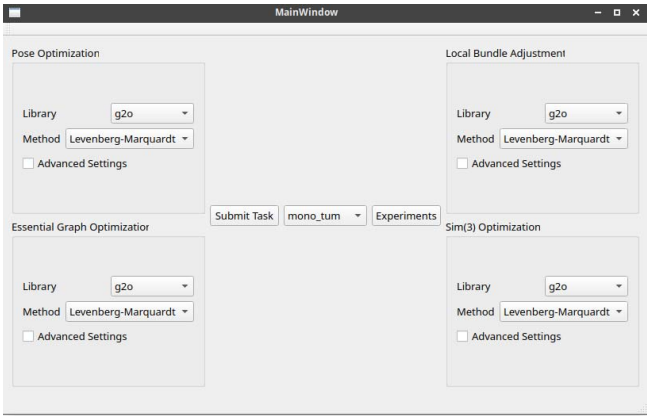

Fig. 3: Initial Page of comparison application

B. Testing the application

This subsection will describe a typical case study that shows all the functionality of the Comparison application. The figure 3 shows the initial windows that give an opportunity to configure each of the four ORB-SLAM optimization tasks. After setting the optimization method of Essential graph optimization to g2o Gauss-Newton, the experiment can be started by clicking the "Submit Task" button (Fig. 3). After clicking the "Submit button", the Comparison application collects the *json* file with the configuration and starts the ORB-SLAM application process. Then ORB-SLAM is initialized with the new configuration and launches the visualization windows: first, to view the point cloud and trajectory, then to display the tracked points in the current image (Fig. 5). After ORB-SLAM completes its work, the results are saved in *json* format, and then the Comparison application shows them and, using all the information about the experiment that it had, inserts all this information into the database. Now the user can click the Experiments button in the main window, which will bring him to the results window (Fig. 4), where the Table fields correspond to the experimental results. If the user clicks the "Visualize trajectory" button in the Results window, the result of launching *evo_traj* with the tested experiments will be shown. In particular, it will show computed trajectories in comparison with the ground truth trajectory (Fig. 6).

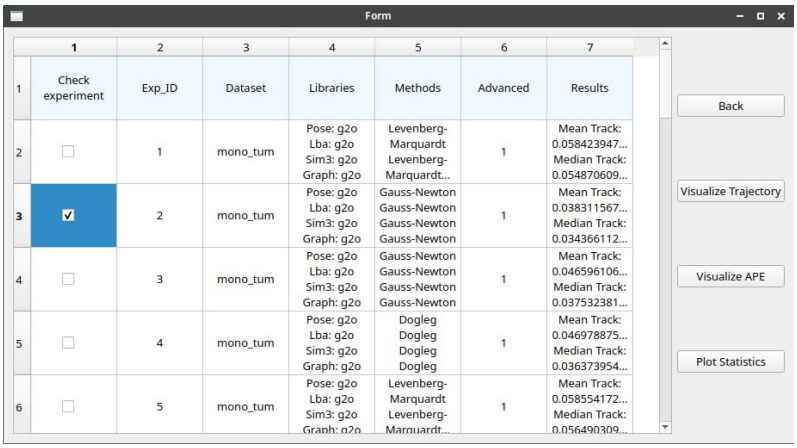

Fig. 4: The 2nd window that shows all the previous results and allows to check the experiments and visualize results

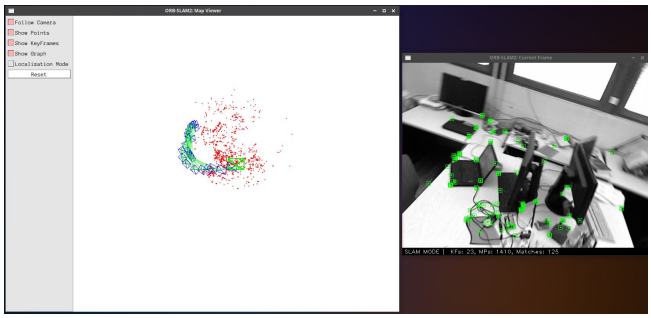

Fig. 5: ORB-SLAM display of camera tracking and position

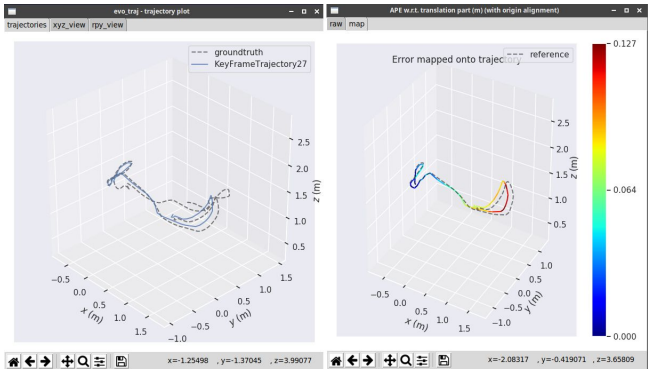

Fig. 6: Plot the experiments: (left) Comparison of the computed and reference trajectories, and (right) Absolute pose error for these trajectories

C. Running the Experiments

Now when the experiments were competed, the Comparison application helps to analyze the results. Let's look at the first question: "Which g2o combination of optimization methods is the best for *mono_tum* and for all optimization tasks?" To answer this question, 11 experiments were conducted. Fig. 7 shows the results (median/mean tracking time). We see that the second experiment, where Essential Graph Optimization problem is solved by Gauss-Newton optimization method, works well in terms of tracking time, and this ORB-SLAM configuration computes the most accurate trajectory. The next question is "Which Ceres combination of optimization methods is the best for *mono_tum* and for all optimization tasks?" The results are shown in Fig. 8. We see that the total tracking time is slightly slower than in g2o-based optimization tasks.

id	Graph	Sim(3)	LBA	Pose	Mean (ms)	Median (ms)	APE (m) - rmse	Path length (m)
1	LM	LM	LM	LM	58	55	0.3	4.5
2	GN	LM	LM	LM	59	58	0.068	7.7
3	DL	LM	LM	LM	59	58	0.42	6.1
4	LM	GN	LM	LM	59	55	0.15	5.3
5	LM	DL	LM	LM	62	61	0.09	7.8
6	LM	LM	GN	LM	63	62	0.11	5.8
7	LM	LM	DL	LM	74	71	0.025	8.4
8	LM	LM	LM	GN	60	59	0.39	4
9	LM	LM	LM	DL	76	79	0.48	3.9
10	GN	GN	GN	GN	Track Lost	Track Lost	Track Lost	Track Lost
11	DL	DL	DL	DL	Track Lost	Track Lost	Track Lost	Track Lost

Fig. 7: The test results, where LM - Levenberg-Marquardt; GN - Gauss-Newton; DL - Dogleg; APE - Absolute Pose Error. Green indicates the best results for median/mean tracking time or for Absolute Pose Error and covered path; Red shows unsuccessful experiments; Blue indicates the trade-off cases

VI. Conclusions

This paper discusses how the configuration of non-linear ORB-SLAM optimization methods affects performance. During our studies the next significant results were obtained:

Extending ORB-SLAM nonlinear methods: As the part of the study, it was decided to expand ORB-SLAM with Ceres-based optimization problems. This extended the research scope by allowing to compare two optimization frameworks and analyze which framework affects SLAM system performance more strongly.

id	Graph	Sim(3)	LBA	Pose	Mean (ms)	Median (ms)	APE (m) - rmse	Path length (m)
1	LM	LM	LM	LM	70	68	0.09	7.9
2	t-DL	LM	LM	LM	61	61	0.078	5.473
3	LM	t-DL	LM	LM	64	65	0.1	5.5
4	LM	LM	t-DL	LM	70	71	0.09	7.6
5	LM	LM	LM	t-DL	70	71	0.15	6.8
6	t-DL	t-DL	t-DL	t-DL	68	68	0.095	7.8
7	s-DL	LM	LM	LM	66	68	0.06	7.7
8	LM	s-DL	LM	LM	84	74	0.07	7.8
9	LM	LM	s-DL	LM	68	69	0.08	7.8
10	LM	LM	LM	s-DL	63	71	0.04	5.7
11	s-DL	s-DL	s-DL	DL	71	71	0.065	8

Fig. 8: The test results, where LM - Levenberg-Marquardt; t-DL - Traditional Dogleg; s-DL - Subspace Dogleg; APE - Absolute Pose Error. Green indicates the best results of median/mean tracking time; Blue indicates the trade-off cases

Design and implementation of Comparison application is one of the most important parts of this study. It allows testing different configurations of ORB-SLAM, e.g. run ORB-SLAM on a particular dataset with a specific configuration of nonlinear optimization methods, showing the evaluation results (mean/median tracking time, absolute pose error) and visualizing the results. The implementation of Comparison application included the development of Qt-based GUI application, the design and deployment of the database for storing results, and the expansion of ORB-SLAM library to work with Comparison application. The code for the Comparison framework can be found here [29].

Running experiments: Several experiments were carried out to test the Comparison application and find out which configuration of nonlinear optimization methods was optimal for a case study. For example, for the question 1, the results showed that the configuration where local bundle adjustment is solved using Dogleg optimization method performs better than others in terms of Absolute pose error (APE). The default configuration for g2o-based optimization problems (all problems are solved using Levenberg-Marquardt) is found the best in terms of mean/median tracking time, but it generates inaccurate trajectory with high APE.

The Comparison application can be further extended with new functionalities, such as:

- Add several experiments into the batch before starting the experiments, which will reduce the time that the user spends on configuring and conducting experiments.
- Add a permutation flag to each property (e.g., an optimization method), which would create a certain number of configurations with each permutation. This would automate testing multiple configurationsfor user convenience.

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] R. Mur-Artal, J. D. Tardos, J. M. M. Montiel, and D. Galvez-Lopez, "Real-time slam for monocular, stereo and rgb-d cameras, with loop detection and relocalization capabilities," https://github.com/raulmur/ORB_SLAM2.

[3] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[4] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[5] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *ICRA*. IEEE, 2011, pp. 3607–3613.

[6] R. Kuemmerle and et al., "g2o: A general framework for graph optimization," https://github.com/RainerKuemmerle/g2o.

[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans on PAMI*, vol. 40, no. 3, pp. 611–625, 2017.

[8] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 1052–1067, 2007.

[9] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," *Intl. Symposium on Mixed and Augmented Reality*, 2007.

[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[11] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun, "Dense monocular depth estimation in complex dynamic scenes," in *CVPR*, 2016, pp. 4058–4066.

[12] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2320–2327.

[13] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time."

[14] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *ECCV*. Springer, 2014, pp. 834–849.

[15] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robot Autom Mag*, vol. 13, pp. 99–110, 2006.

[16] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1974–1982.

[17] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustmenta modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[18] H. Liu, M. Chen, Y. Bao, and Z. Wang, "Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam," https://github.com/baidu/ICE-BA.

[19] "Technical university of munich (tum) datasets for slam," https://vision.in.tum.de/data/datasets.

[20] J. Sturm, W. Burgard, and D. Cremers, "Evaluating egomotion and structure-from-motion approaches using the TUM RGB-D benchmark," in *Proc. of the Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RJS International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[21] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[22] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *CVPR*, 2015, pp. 3061–3070.

[23] A. Buyval, I. Afanasyev, and E. Magid, "Comparative analysis of ros-based monocular slam methods for indoor navigation," in *ICMV*, vol. 10341, 2017, p. 103411K.

[24] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ros-based visual slam methods in homogeneous indoor environment," in *Positioning, Navigation and Communications (WPNC)*. IEEE, 2017, pp. 1–6.

[25] M. Filipenko and I. Afanasyev, "Comparison of various slam systems for mobile robot in an indoor environment," in *International Conference on Intelligent Systems*, 2018.

[26] R. Giubilato, S. Chiodini, M. Pertile, and S. Debei, "An evaluation of ros-compatible stereo visual slam methods on a nvidia jetson tx2," *Measurement*, vol. 140, pp. 161–170, 2019.

[27] A. Bokovoy and K. Yakovlev, "Original loop-closure detection algorithm for monocular vslam," in *AIST*. Springer, 2017, pp. 210–220.

[28] A. Gabdullin, G. Shvedov, M. Ivanou, and I. Afanasyev, "Analysis of onboard sensor-based odometry for a quadrotor uav in outdoor environment," in *Int. Conf. on Artif. Life and Robotics (ICAROB)*, 2018.

[29] B. Makaev, "Comparison framework of nonlinear optimization methods for orb-slam," https://github.com/borisqa00/ComparisonFramework.