VINS-MONO OPTMIZED: A MONOCULAR VISUAL-INERTIAL STATE ESTIMATOR WITH IMPROVED INITIALIZATION

A Thesis

by

LINGJIE XU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Reza Langari
Co-Chair of Committee,	Srikanth Saripalli
Committee Members,	Dylan Shell
Head of Department,	Andreas A. Polycarpou

December 2018

Major Subject: Mechanical Engineering

ABSTRACT

State estimation is one of the key areas in robotics. It touches a variety of applications in practice such as, aerial vehicle navigation, autonomous driving, augmented reality, and virtual reality. A monocular visual-inertial system (VINS) is one of the popular trends in solving state estimation. By fusing a monocular camera and IMU properly, the system is capable of providing the position and orientation of a vehicle and recovering the scale. One of the challenges for a monocular VINS is estimator initialization due to the inadequacy of direct distance measurement. Based on the work of Hong Kong University of Technology on monocular VINS, a checkerboard pattern is introduced to improve the original initialization process. The checkerboard parameters are used along with the calculated 3D coordinates to replace the original initialization process, leading to higher accuracy. The results demonstrated lowered cross track error and final drift, compared with the original approach.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Prof. Langari, my committee co-chair, Prof. Saripalli and my committee member, Prof. Shell, for their guidance and support throughout the course of this research. I would also like to thank Prof. Chakravorty in the Aerospace Engineering Department for sparing his valuable time joining the defense. Additionally, thanks also go to Sai Vemprala and Subodh Mishra for having valuable discussions with me when questions arose.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

Finally, I would like to thank my parents and my girlfriend, for their encouragement, patience, and love.

CONTRIBUTORS AND FUNDING SOURCES

This work was supervised by a dissertation committee consisting of Professor Langari and Professor Saripalli of the Department of Mechanical Engineering and Professor Shell of the Department of Computer Science and Engineering. All work for the dissertation was completed independently by the student. No outside financial support was gained during the completion of all the work.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

Page

1. INTRODUCTION

In robotics, accurate localization is one of the fundamental challenges. In order to perform autonomous navigation, a robot needs to be constantly aware of its location [1]. Aiming at achieving successful localization, different sensors and techniques are used such as, quadratic encoders, GPS, inertial navigation system (INS), and visual odometry. For different sensors, each has its own limitations. For example, a quadratic encoder would give huge error when dealing with slippage and a GPS would have trouble functioning properly if the vehicle is in an indoor environment or a confined space. Also, INS suffers from significant accumulating drift since the calculation is closely correlated to integration over time. Laser sensor is another type of sensor that is being used in positioning related applications. It is a remote sensing technology for distance measurement that involves transmitting a laser toward the target and then analyzing the reflected light [1].[1] However, it is highly dependent on the surface material and the orientation of surfaces. Another shortcoming of LIDAR as well as high precision INS is that they generally have high costs, putting budget as one of the concerns. To balance between cost and precision, visual odometry (VO) brings great value to localization.

Visual odometry is the process of orientation and position determination by running analysis through a sequence of associated images and a camera is the input source. The reason why visual odometry is a great approach for vehicle localization is due to its balance among cost,

[1] Part of this chapter is reprinted with permission from Review of Visual Odometry: Types, Approaches, Challenges, and Applications (2016), M. O.A. Aqel, M.H. Marhaban, M. I.Saripan, and N.Bt.Ismail, Springer Nature, 2016, 5:1897

[2] Part of this chapter is reprinted with permission from VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator (2018), T. Qin, P. Li, S. Shen, IEEE Trans. Robot., vol. 34, no. 4, pp. 1004–1020, 2018

reliability, and implementation complexity [1].[1] It uses inexpensive device and the accuracy is generally higher than conventional odometry techniques such as quadratic encoder. And it can be incorporated with GPS and/or INS to further increase its accuracy. Another advantage of visual odometry is that it can still function properly in a place where it is not GPS-friendly.

VO can generally be categorized into two categories based on the type the camera it uses: monocular and stereo VO. A stereo camera has two lenses and each lens has a separate image sensor. It is capable of providing the depth information from a single frame so that the image scale is immediately available. However, one of the drawbacks of a stereo camera is that it usually costs more than a conventional camera. Additionally, a stereo camera requires more calibration effort and the calibration errors directly impact the motion estimation process [2]. Compared with stereo camera, a monocular camera reduces the influence the calibration errors have on motion estimation. Due to its inexpensiveness and low deployment complexity, more and more applications lean towards a monocular visual odometry.

As discussed in [2] and [3], monocular VO suffers from scale uncertainty especially on uneven terrains, which leads to degeneracy. To overcome this limitation, a monocular visual-inertial system (VINS) was introduced to eliminate the errors caused by the lack of scale. The minimum system requirement for visual-inertial odometry (VIO) consists of a monocular camera and an inertial measurement unit (IMU) which helps to obtain the scale that is missing in the vision-only methods. The camera provides sequential images of the scene, and IMU provides the user with the accelerations and angular velocities of the motion at high frequency. The reason why a monocular VINS is able to accomplish the state estimation is that IMU can have scale along with the pitch and roll angels observable [4], the variable required for navigational tasks.[2]

According to [4], the scale is not directly observable from monocular camera, making it difficult to directly fuse visual and inertial measurements without good initial values. In addition, it is an inappropriate assumption that a monocular VINS could be launched from a stationary position since the excitation of IMU is required to retrieve scale in the later steps. To make sure a monocular VINS is applicable and the tracking process is accurate, initialization plays a crucial part in the entire process. Hence, a more robust initialization process is needed for monocular VINS.

VINS-mono is a publicly available VIO pipeline that can be applied on different hardware platforms. It was newly developed by Hongkong University of Science and Technology in 2017, and the experimental results on long trajectories were better than state of the art visual inertial algorithm OKVIS.

Since VINS-mono does not have any knowledge about the world frame, all the poses are calculated with respect to the first frame. Bundle adjustment is used to minimize the reprojection error. All the camera poses computed in the vision-only SfM step are then used in the visual-inertial alignment procedure. According to [4], VINS-mono itself had only been tested on drones and is sensitive to the environment. Besides the lack of knowledge of the world frame, during the evaluation tests, several other issues were observed:

- High failure rate: tracking stops and returns to the starting location
- Huge drift before walking after initialization (not an camera-IMU synchronization issue)
- High cross track error on long trajectories

The motivation of this research is to propose an optimized vision-only initialization for VINS-mono to improve its robustness and accuracy. The verification of the improvement will be verified by comparing the cross track errors on different routes and the final drift on long trajectory.

The contribution of this work are:

- A structured scene for vision-only initialization for accurate and consistent initialization
- A world frame based pose estimation

2. RELATED WORK

According to [5], besides VINS-mono, there are several VIO pipelines available to the public: MSCKF, OKVIS, ROVIO, SVO+MSF, and SVO+GTSAM. [3]

MSCKF [6] is a multi-state constraint Kalman Filter for visual-inertial navigation. It is essentially an extended Kalman Filter (EKF) with multiple subsequent variations [6].[4] MSCKF derived a measurement model capable of describing the geometric constraints when multiple camera poses saw a static feature. This model does not require 3D features positions to be included in the state vector of EKF. A group of researchers from University of Pennsylvania implemented MSCKF as the backend of their event-based VIO [7]. MSCKF laid the foundation of many VIO systems and has been uploaded for public access to GitHub at the beginning of 2018.

Open Keyframe-base Visual-inertial SLAM (OKVIS) [8] presented a novel approach to tightly integrate camera measurements with IMU readings in simultaneous localization and mapping (SLAM). [5] OKVIS introduced a key concept called keyframe for drift free estimation.

[3] Part of this chapter is reprinted with permission from A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots (2018), J. Delmerico, D. Scaramuzza, ICRA, 2018

[4] Part of this chapter is printed with permission from A Multi-State Constraint Kalman Filter for Vision-Aided Inertial Navigation (2007), A. I. Mourikis and S. I. Roumeliotis, IEEE Int Conf. on Robotics and Automation, 2007

[5] Part of this chapter is printed with permission from Keyframe-based Visual-inertial Odometry Using Nonlinear Optimization (2015), S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, Int. J. Robot. Research, vol. 34, no. 3, pp. 314–334, 2015

[6] Part of this chapter is reprinted with permission from Robust Visual-inertial Odometry Using a Direct EKF-based Approach (2015), M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, IEEE/RSJ Int Conf. on IROS, 2015

[7] Part of this chapter is reprinted with permission from SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems (2017), C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, IEEE Trans. Robot., vol. 33, no. 2, pp. 249–265, 2017

[8] Part of this chapter is reprinted with permission from A Robust and Modular Multi-sensor Fusion Approach Applied to MAV Navigation (2013), S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, IEE/RSJ Int Conf. on IROS, 2013

[9] Part of this chapter is reprinted with permission from iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree (2012), M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, Int. J. Robot. Research, vol. 31, no. 2, pp. 216–235, 2012

According to [8], a frame is selected to be a keyframe if the ratio between the image area spanned by matched points versus the area spanned by all detected points falls below 50%-60%. This allows the process to keep the visual constraints while still respecting an IMU term [8]. The feature detection part is done by Harris corner detector [9]. And BRISK descriptor [10] is computed in order to associate the data between frames. OKVIS also performs a non-linear optimization using *ceres solver* [11] from Google to minimize the reprojection error. OKVIS was originally tested on a stereo platform, but it can also be extended to a monocular configuration.

ROVIO stands for Robust Visual Inertial Odometry. It used pixel intensity errors of image patches to perform tracking [12].[6] An extended Kalman Filter is closely coupled with multilevel patch feature tracking. The patches are extracted from the images and around FAST corner features [12] [13], where root-centric bearing vectors and distances are parameterized as the 3D positions. ROVIO does not require any initialization procedure, according to [12], due to such robocentric, inverse distance landmark parameterization. This type of power-up-and-go framework was successfully tested in both hand-held and a multi-rotor unmanned vehicle (UAV) experiments. Unlike OKVIS, the pipeline was targeted at monocular VIO.

Aiming at improving the time efficiency and the robustness, Semi-direct VO (SVO) was proposed. It uses direct method to track and triangulate pixels characterized by high image gradients but relies on proven feature-based methods to optimize the structure and motion [14].[7]

[10] Part of this chapter is reprinted with permission from On-manifold Preintegration for Real-Time Visual-Inertial Odometry (2017), C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, IEEE Trans. Robot., vol. 33, no. 1, pp. 1–21, 2017

[11] Part of this chapter is reprinted with permission from Evaluation of Modern Visual SLAM Methods (2015), A. Huletski, D. Kartashov, and K. Krinkin, Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference, 2015

[12] Part of this chapter is reprinted with permission from LSD-SLAM: Large-scale Direct Monocular SLAM (2014), J. Engel, T. Schops, and D. Cremers, Computer Vision-ECCV 2014, pp. 834-849, 2014

SVO also tracks FAST corners [13]. Multi-Sensor Fusion (MSF) is a generic EKF framework. It is designated for fusing different sensor data in state estimation. MSF is capable of processing delayed, relative, and absolute measurements from a theoretically unlimited number of different sensors and sensor types [15].[8] This allows a sensor suite to do self-calibration. By combining SVO and MSF, the estimated pose from SVO is passed to MSF as the output of a generic pose sensor [5]. This output will then be fused with IMU as in [16]. The setup was loose coupling, indicating the need for approximately correct scale of the pose. Because of this, an onboard laser rangefinder was used to estimate the distance. The SVO and MSF combination can be referred as "svomsf" where two individual pipelines communicate through a ROS interface.

iSAM2 is an incremental smoothing and mapping using the Bayes tree. A Bayes tree contains a factored probability density. It is directed and maps to the square root information matrix of SLAM [17].[9] This new data structure can be served as the backend performing online factor graph optimization [5]. By pairing the SVO and MSF combination with iSALM2, the authors in [18] used this system along with pre-integrated IMU for pose estimation.[10] Since IMU measurements come in at high rate, more and more variables need to be taken into consideration during the optimization process. The purpose of pre-integrating IMU between keyframes is to reduce the number of variables introduced to optimization, therefore saves computational cost. [18] also demonstrated the pre-integrated IMU model could be seamlessly integrated into VIO pipeline. Such integration made the incremental-smoothing algorithms applicable [18]. It also helped avoid optimization over 3D points, which further increased time efficiency. The implementation was released as GTSAM 4.0 optimization toolbox [19]. And the system in [18] is referred as "svogtam".

ORB-SLAM [21] is a feature based monocular SLAM system operating in different environments. ORB features [22] are used in this application. Different from some other SLAM algorithms, no external odometry is used [20]. The SLAM system contains 3 modules working in parallel threads [20]:[11]

- tracking thread localizes camera in each new frame and decides new keyframe insertion
- local mapping conducts new frame processing and local bundle adjustment for space reconstruction
- loop closure finds big loops when the new keyframe becomes available

ORB features are extracted from the new camera frame. If there is a successful tracking in the last frame, the initial camera pose estimation will be performed on this frame. The camera poses can be computed using PnP method through 3D-2D correspondences. All correspondences are iteratively determined using the initial pose estimation and feature matches. The camera pose optimization is done by using the initial estimation and all correspondences. The goal of optimization is to minimize the reprojection error using Levenberg-Marquadt algorithm with Huber cost function.

Large-scale direct monocular SLAM (LSD-SLAM) is an RGB image based algorithm. The algorithm is capable of highly accurate pose estimation based on direct image alignment. It can also perform 3D reconstruction in real-time as pose-graph of keyframes with associated semi-dense depth maps [23].[12] LSD-SLAM is essentially developed on the basis of semi-dense monocular visual odometry proposed in [24]. The knowledge about the environment is stored as a topological graph/map [20]. Each node holds an image and probabilistic inverse map. The edges that connect the nodes contain the information about relative scale-aware alignment and corresponding covariance [20]. The observed camera images are approximated by a set of keyframes. As a result, the whole map split into two parts: clusters, and the keyframes that stores cluster-specific data. The explicitly scale-drift aware formulation allows the approach to operate on challenging situations such as large variations in scene scale. The algorithm was successfully tested on a 500m trajectory.

In [28], the authors implemented a monocular visual-inertial state estimator for mobile augmented reality. A visual-inertial system was implemented on a mobile phone whose built-in camera and IMU meets the minimum requirement for the sensor suite. The approach applied the initialization process presented in [32]. The work was claimed to be drift-free in different environments and does not require prior assumption or knowledge regarding the environment. The results showed the improved accuracy compared with OKVIS [8]. However, the comparisons seem to be insufficient. Also, the error for a medium dataset in one direction is higher than OKVIS. Besides, the sensitivity to IMU noise and the assumption of the known orientation for a linear closed-form reduces the robustness of the method. Another issue was the approach did not take camera intrinsic parameters into consideration.

In [29], based on SVO [14], the article claimed to be faster and more accurate compared with OKVIS with no prior landmark points. The test results demonstrated lower errors than the benchmark under easy scenarios. However, the trends of the errors tended to getting closer and closer to OKVIS's as the difficulty of the scenarios went up.

[30] proposed robust visual-inertial state estimation with a quadrotor, claiming to be able to handle the speed up to 4m/s. The experimental results were close to the desired ones. The setup consists of two cameras, one primary and one secondary, compensating the limitation of the monocular vision-based approach. But the approach was based on the assumption of noiseless quantity for camera pose, which could potentially jeopardize the accuracy when not in a controlled

environment. Also, the approach requires slow change in the scale [30], which put another potential constraint on the proposed method. Another issue that is worth mentioning is that the vehicle was only tested indoors when reaching the top speed, leaving the outdoor top speed scenario untested.

A robust initialization was proposed in [31] for aerial robots. The approach uses 5-point algorithm for triangulation [34] and perspective-n- point (PnP) method [33] for pose estimation. Then bundle adjustment [35] is applied to minimize the total re-projection error. And visual-inertial alignment is performed to complete the initialization process. The experimental result turned out to be better than OKVIS under easy scenarios. However, the fluctuation of the errors was bigger than the benchmark in difficult scenario. The outdoor environments were generally challenging for the approach as they require longer time for to complete the initialization [31]. Furthermore, high altitudes would cause IMU measurement degeneracy and made it hard to recover the scale.

While there are more monocular based algorithms such as PTAM [25] nowadays available to the public, this research was based on a new and comparatively accurate monocular visual odometry algorithm, VINS-mono.

VINS-mono [4] came into being in 2017. It is a sliding window monocular visual-inertial state estimator based on non-linear optimization. The system requirement is a monocular camera and an IMU. Similar to OKVIS, it tracks robust corner features as described in [26]. In [26], the authors proposed a more principled definition of feature quality. This helped generating better results than Harris corner detector [9] according to [27]. A couple of new features were added to the estimation framework of VINS-mono. A loosely-coupled sensor fusion initialization was proposed to bootstrap the estimator from arbitrary initial states [5]. IMU measurements are pre-integrated before optimization. And the authors proposed a tightly-coupled VIO for relocalization purpose. The most fragile part of VINS-mono is initialization. The initialization process consists of a sliding window vision-only (vision only SfM) initialization and visual alignment part [4]. The vision-only SfM starts with feature detection and extraction. Between two frames, they need to have more than 30 feature overlap and at least 20 parallaxes to perform 5-point algorithm [34]. After finding the relative rotation and translation between the last frame and frame k, the features will be triangulated frame by frame. And the pose of each frame is computed through PnP method [33].

3. METHODOLOGY

The purpose of this research is to optimize the initialization process of VINS-mono and verify the improvement. In the original work, the initialization starts with a vision-only initialization. The first camera frame is set as the reference as the world frame is. All the camera poses are estimated with respect to the first camera frame. The rotation and translation of each frame relative to the first frame are passed to the visual-inertial alignment process to recover the scale. The method proposed method in this research uses a checkerboard pattern for the vision-only pose estimation. Instead of using the first frame, the world frame is used to estimate the pose of each frame. The world frame feature coordinates are computed based on the checkerboard parameters. By using this proposed vision-only initialization process, it is expected the pose estimation will be more accurate and we can obtain better scales.

The apparatus is Intel RealSense camera ZR300 fisheye mode and it is attached to a laptop. ZR300 is equipped with a camera lens and IMU, and both units are hardware synchronized. Figure 1 describes the gyroscope and accelerometer sensors.

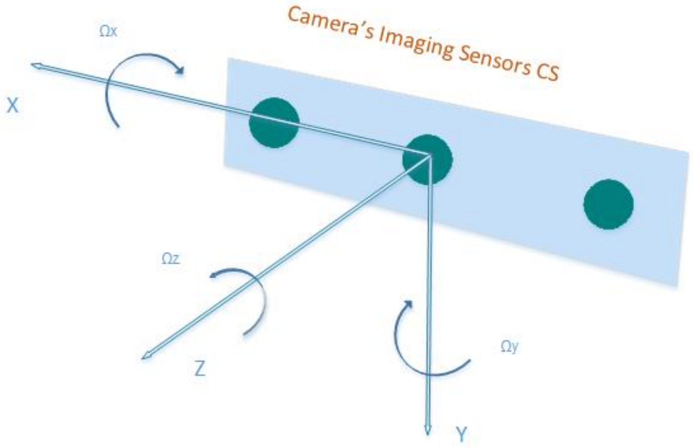

Figure 1. Sensor Descriptions of ZR300

Accelerometer axes: the positive x-axis points to the right, the positive y-axis points down and the positive z-axis points forward. The gyroscope axes are around the accelerometer axes respectively. The laptop would be held in hand and walk different routes. The equipment setup is presented in Figure 2.

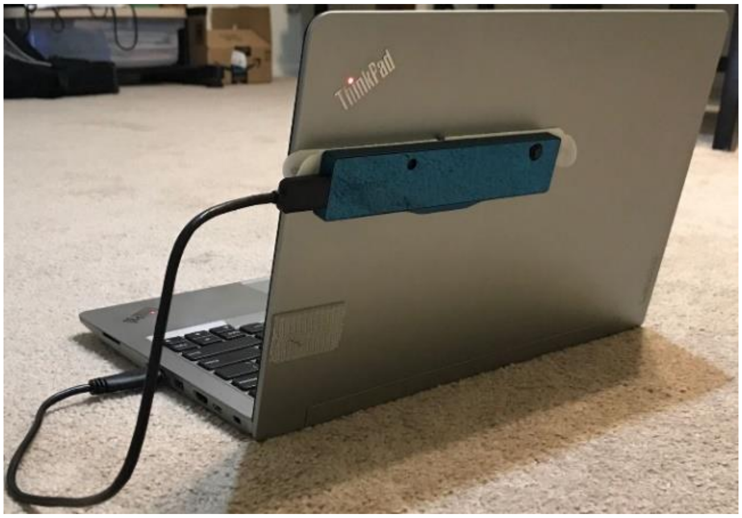

Figure 2. Equipment Setup

Four different routes were tested for the original VINS-mono and the improved version. And each route was to be tested multiple times for each version of VINS-mono to gather quantitative data for error analysis. Let the direction the device starts moving in be Y direction and direction that is perpendicular to Y and parallel to the ground be X. The testing routes were:

1. A straight line of 7.31m, one-way direction
2. A straight line of 7.31m, forward and backward (14.62m)
3. A rectangle loop of 7.8m by 5.4m (26.4m)
4. A rectangle corridor of 13.8m by 60m (147.6m)

All of the testing routes were marked with white tapes and measured using a tape measure. The first three routes were conducted in a lab with textured floorings (see Figure 3).

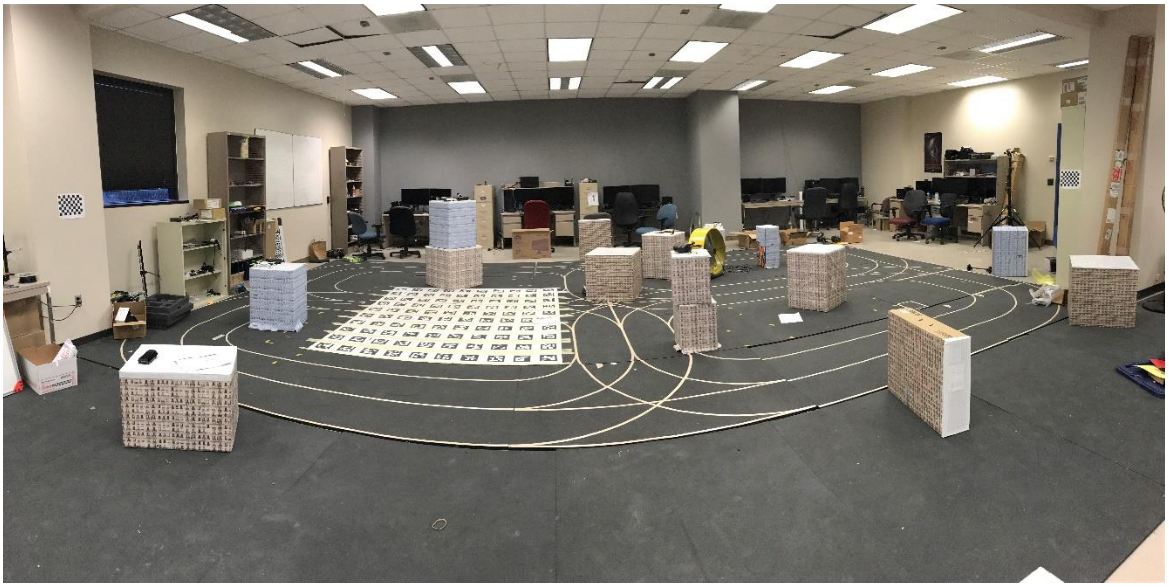

Figure 3. Testing Lab Environment

The 4th route was the corridor on the 4th floors of ENPH building at Texas A&M University. Prior to walking, the camera was pointed slightly downward to have sufficient amount of ground truth. The walk should follow the marked line as steady as possible. When making turns, it is crucial to keep a slow and steady turn to minimize the deviation caused by human.

The checkerboard pattern needs to be in front of the camera when initializing. The checkerboard being used has 6 inner corners per column and 9 inner corners per row with 2.5cm as the side length of each square (see Figure 4).

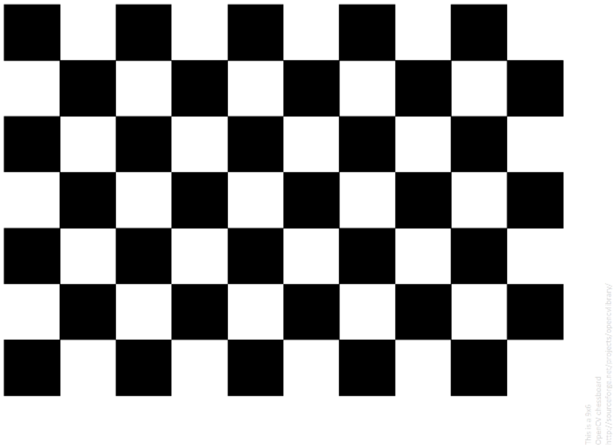

Figure 4. Checkerboard Pattern

The pattern parameters mentioned above are the arguments for checkerboard detection in each frame:

- width: number of inner corners per row
- height: number of inner corners per column
- unitSquare: side length of each square in the pattern

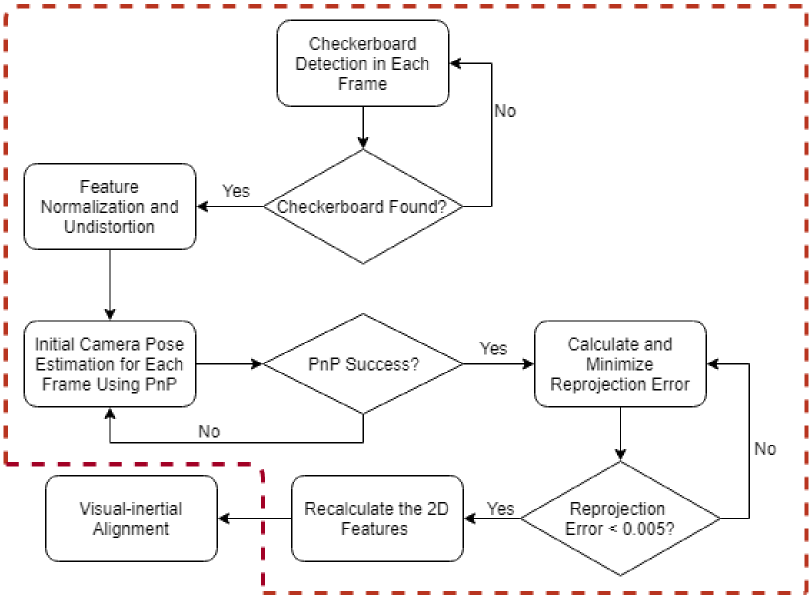
Checkerboard Detection in Each Frame
No
Feature Normalization and Undistortion
Yes
Checkerboard Found?
Initial Camera Pose Estimation for Each Frame Using PnP
PnP Success?
Yes
Calculate and Minimize Reprojection Error
No
No
Visual-inertial Alignment
Recalculate the 2D Features
Yes
Reprojection Error < 0.005?

Figure 5. Block Diagram of Checkerboard Initialization

Figure 5 above demonstrates the flow of the checkerboard initialization process. The checkerboard detection returns 2D coordinate of the detected corners. Since the fisheye camera creates huge distortion on in each frame, it is important to rectify the detected corners. The rectification process can be found in Appendix A. The checkerboard parameters are also used to calculate the 3D coordinates of the inner corners in the world frame. The pseudo code for 3D coordinate computation is in Appendix B. The detected corners and the corresponding 3D coordinate were then passed to solve the perspective-n-point (PnP) problem. In order to check whether the checkboard was successfully detected, the detected inner corner are projected onto the corresponding frame. The PnP method provides an initial calculation of the rotation and translation from camera to world in each frame. To minimize the reprojection error, *ceres* [11] is used to minimize the reprojection error. The cost is the squared L-2 norm (RMS^2) of the reprojection error. A detailed description of the pose optimization can be found in Appendix C. The optimized poses will be passed to the visual-alignment process where the velocity, scale, and gravity are initialized.

4. RESULT

In order to verify the new vison-only initialization improves the accuracy of VINS-mono, two criterion are used: cross track error and final drift. Cross track error is the perpendicular deviation of the experimental data from the reference. And final drift is the deviation of the final position from the starting point. Cross track error can commonly be interpreted as drift. The lower the drift, the better the accuracy. Final drift is taken into consideration since we would want a vehicle that travels in a loop to return to its starting position and be as accurate as possible.

For each testing route, several statistical characteristics are computed for both the original and optimized VINS-mono:

1. the average of the mean and maximum of the cross track errors
2. the standard deviation of the average and the maximum of the cross track errors
3. the root mean square (RMS) of the average cross track error

These characteristics are computed to verify if the accuracy has been improved and how stable the accuracy is. Additionally, the original VINS-Mono was tested on a long trajectory of 700m to evaluate the final drift. To compare the final drift in both X and Y direction on long trajectory, the average and the standard deviation of the final drifts are computed among the tests.

Let the direction where there should be no deviation be x, the cross track errors were calculated using

$$cross\ track\ error = x_{experimental} - x_{reference} \tag{1}$$

For route 1 and 2, since the device essentially follows a straight line, the corresponding reference in x is 0m. For the rectangular shaped route such as route 3 and 4, the references are [0, length of the rectangle, width of the rectangle, 0] m. Each change of the reference is corresponding to a turn.

Finding the turning points in the experimental data is critical for cross track error computation. As a turn was usually slow and steady to minimize the deviation of the camera, there should be a small number of points that were close to each other. And when we resumed moving in X direction after a complete turn, a sudden change of value in X and/or the sign in this direction should be marked as the turning position. The same rule should be applied for the second and third turn in Y direction and in X direction respectively.

Let $p(x, y)$ be one of the positions on an experimental route. The final drift of each test could be calculated by:

$$final\ drift = p_{end} - p_{start} \tag{2}$$

Figure 6 is a demonstration of the successfully detected inner corners.

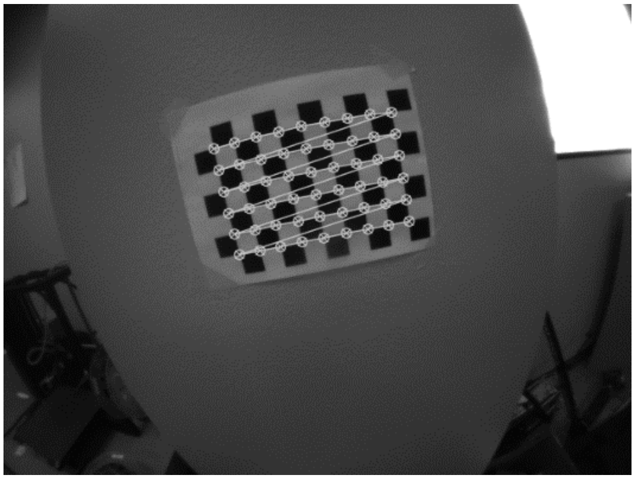

Figure 6. Successful Checkerboard Detection

Each route was tested multiple times and plotted the routes along with the reference for both the original VINS-mono and improved one. Figure 7 – 10 show the tests on both the original and optimized approaches. The references were displayed in bold lines on the graph to exhibit how close the experimental data sets were to the theoretical numbers.

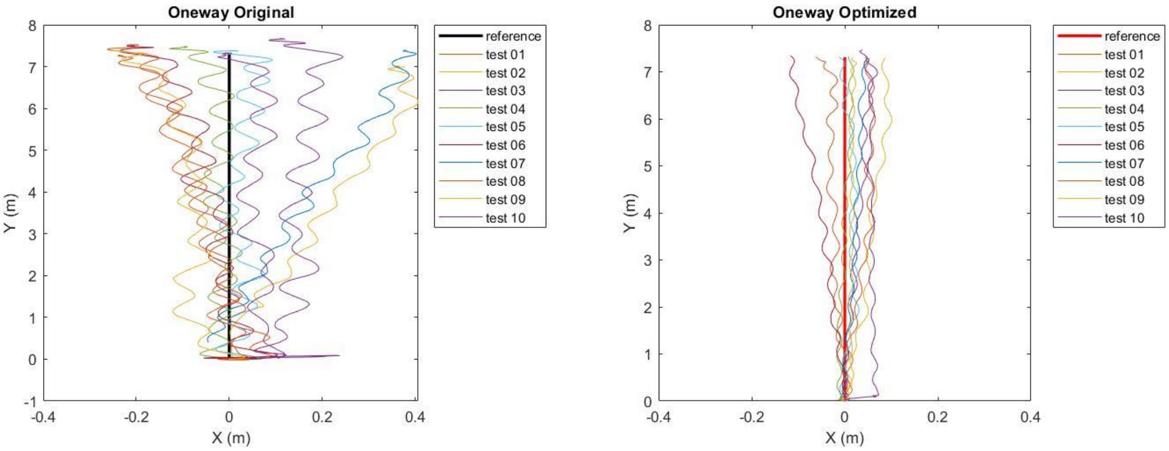

Figure 7. One-way Route Tests vs Reference

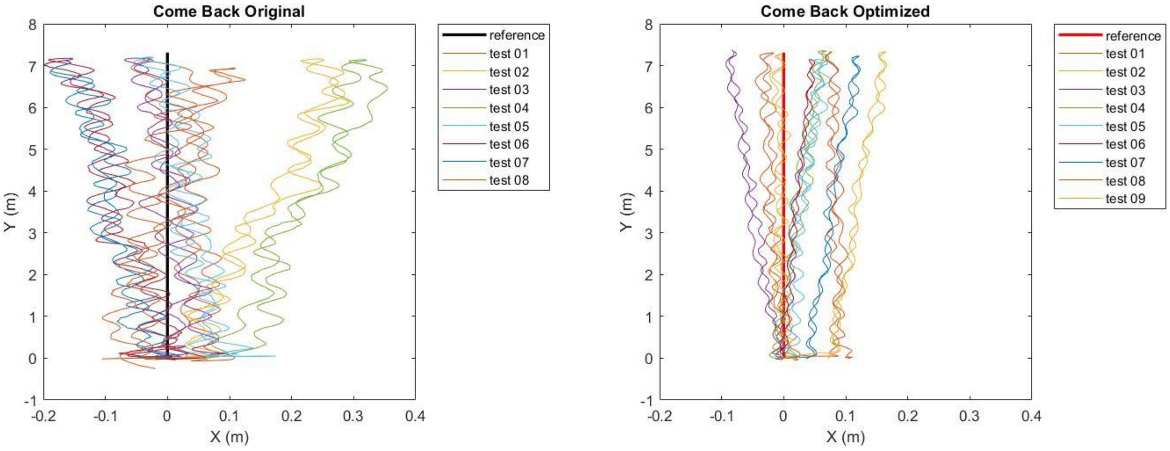

Figure 8. Forward and Backward Route Tests vs Reference

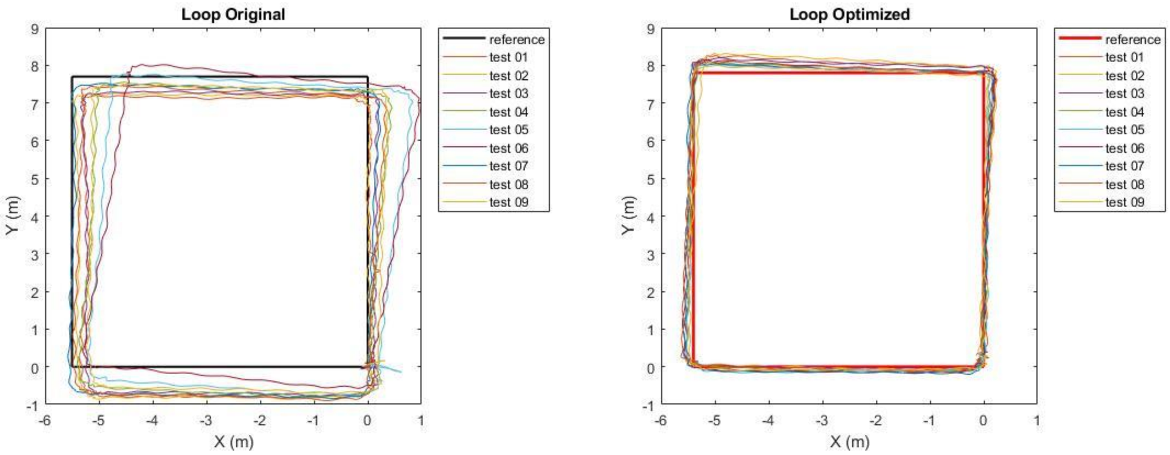

Figure 9. Small Rectangular Route Tests vs Reference

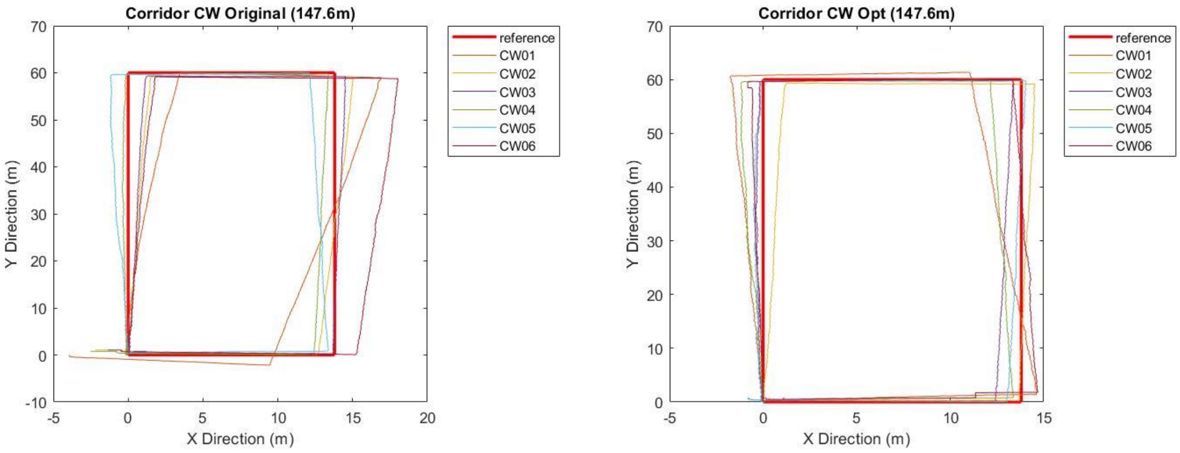

Figure 10. Corridor Tests vs Reference

A general observation based on the route graphs above is that the optimized initialization process gave better accuracy. The routes that were tested with the improved initialization stay closer to the corresponding references. From Figure 7, the drift is not completely eliminated. However, it can be seen that the drift of the proposed method varies in a much smaller range compared with the original VINS-mono. If we ignore the results that are much greater drift than the others, for 7.31m route, the drift of improved VINS-mono stays within approximately [-0.1, 0.1]m while the original fluctuates between -0.2m and 0.2m. The similar result can also be observed for the forward and backward route in Figure 8. The drift after improvement stays between -0.09m and 0.1m while the original version fluctuates between -0.2m to 0.1m. The improvement on accuracy is significant on the small rectangular shaped route presented in Figure 9. Even if we ignore the most skewed experimental route in the data collected from original VINS-mono, the drift can still goes up to 1m. However, all of the tests experimented using the optimized VINS-mono stay very close to the reference with only about 0.2m drift at most. The test results are also better for the long corridor. For both versions of VINS-mono, there is one experiment that is significantly skewed. By taking the greatly skewed experiments out the consideration, the remaining tests stay closer using the optimized VINS-mono. The maximum drift is about 2m using the improved initialization while it is roughly 5m for the original approach.

To further solidify the improved accuracy of the optimized VINS-mono over the original, Figure 11 – 14 show the cross error analysis on each route. The drifts were taken by their absolute values.

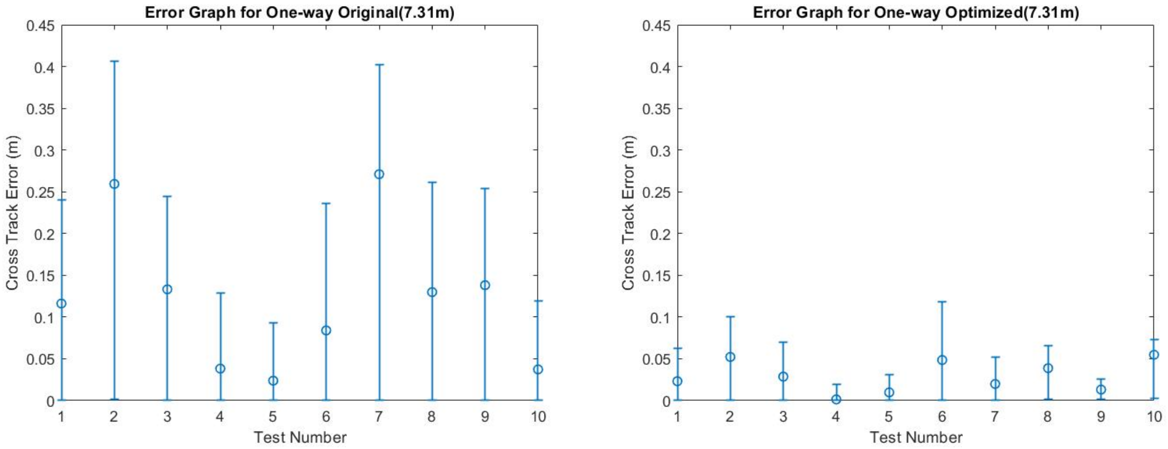

Figure 11. Cross Track Error of One-way Tests (left: original, right: optimized)

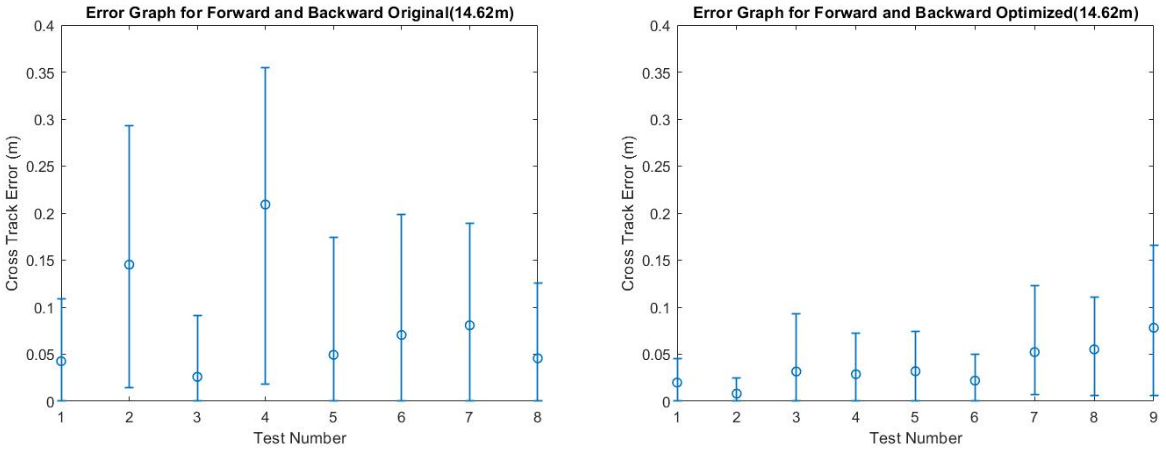

Figure 12. Cross Track Error for Forward and Backward Tests (left: original, right: optimized)

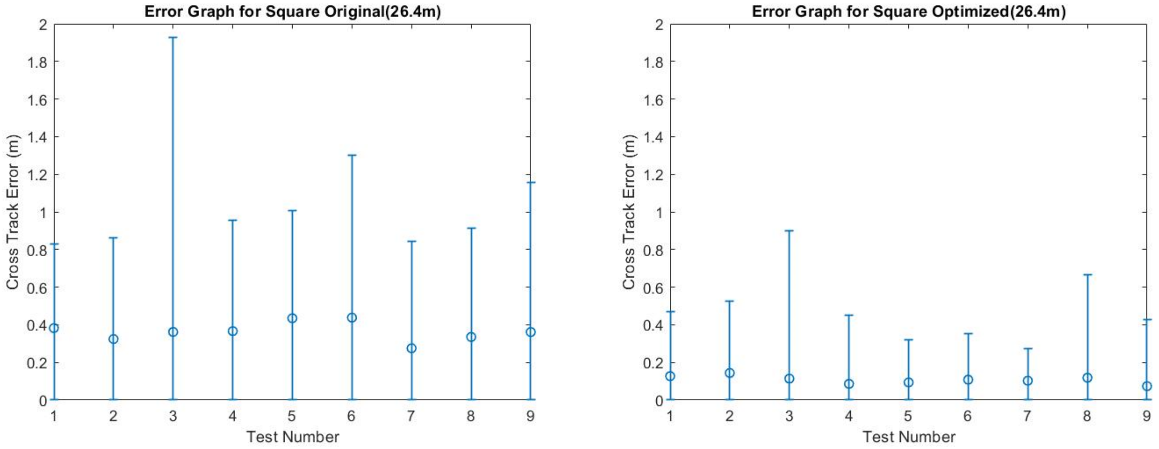

Figure 13. Cross Track Error for Small Rectangle Tests (left: original, right: optimized)

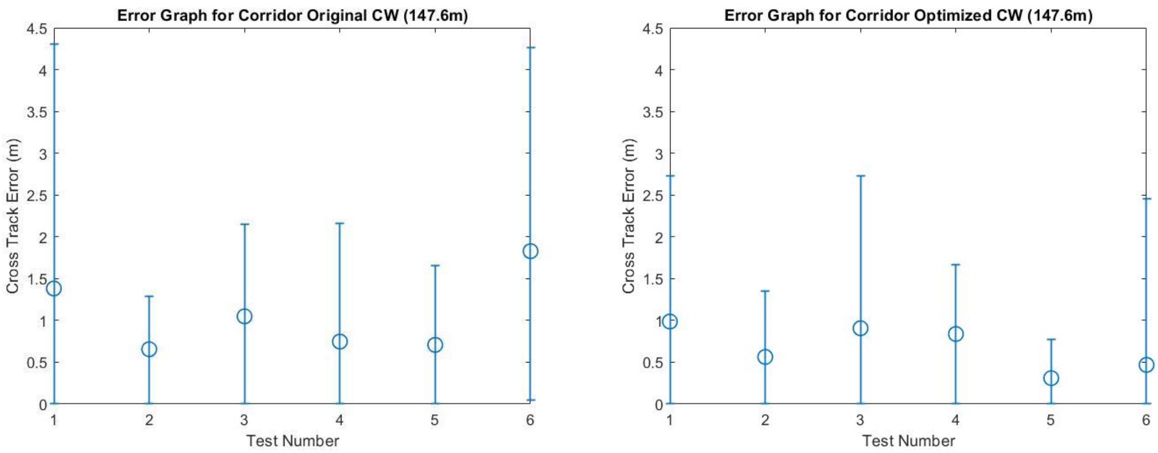

Figure 14. Cross Track Error for Corridor Tests (left: original, right: optimized)

The error bar graphs serve as further evidence of the corresponding route plots. In general, the average and maximum cross track error for each route have decreased. Also, it can be observed that the average cross track error for each route fluctuates much less compared with the original VINS-mono. For example, the average error stays below 0.05m for one-way tests on the optimized VINS-mono, unlike the original that has greater up and downs. Most importantly, the maximum cross track errors are significantly reduced in all cases. Such is a huge improvement that in longer routes such as the corridor, the there is an over 1m reduction in the maximum cross track error.

To verify the decrement of the cross track error of the optimized VINS-mono, Table 1 shows the averages and standard deviations of the cross track error averages and maximum values. And Table 2 verifies the improvement from an RMS point of view.

Table 1. Comparison of Cross Track Analysis on VINS-monos

Route Type		Straight (m)			Forward and Back (m)			Small Loop (m)			Corridor (m)	
		mean	max		mean	max		mean	max		mean	max
Original	Avg	0.123	0.239		0.084	0.192		0.364	1.088		1.061	2.637
	Stdev	0.086	0.107		0.063	0.091		0.051	0.351		0.465	1.316
Optimized	Avg	0.029	0.062		0.036	0.085		0.107	0.487		0.682	1.888
	Stdev	0.019	0.032		0.022	0.044		0.021	0.194		0.267	0.778

Table 2. Comparison of RMS of Error Average

Route Type	Straight (m)	Forward and Back (m)	Small Loop (m)	Corridor (m)
	RMS_mean	RMS_mean	RMS_mean	RMS_mean
Original	0.148	0.102	0.367	1.143
Optimized	0.034	0.042	0.109	0.724

As showed in Table 1, compared with the original VINS-mono, the average of the mean cross track error has a big decrement. The biggest drop of the average cross track is at 74.6% and the minimum drop can still reach up to 35.7%. The biggest drop of the average maximum cross track error is 74.06% and the minimum drop is at 28.4%. The biggest and least error reduction happen at the one-way test and corridor test respectively. From an RMS point of view displayed in Table 2, each route has quite obvious RMS reduction using the checkboard initialization. In other word, the cross track error is lower on each route with the optimized vison-only initialization process.

The original VINS-mono had a final drift of [-5.47, 2.76] m on a long trajectory of 700m without loop closure, taking 0.88% of the total length. On the long corridor tests (147.6m), the optimized VINS-mono has an average final drift of [-0.018, 0.821] m in X and Y direction with standard deviation of [-0.136, 0.774] m respectively. The average final drift takes 0.56% of the total route length. The maximum final drift is [-0.716, 0.773] m if not considering the significantly skewed test, taking 0.71% of the total trajectory length.

5. DISCUSSION

It is a common issue that visual odometry has drift no matter what kind of algorithm it uses. And the drift would normally grow bigger as the travel distance increases. One may say the cross track error for the forward and backward route is lower than the one-way route while the total length is longer. This could have been caused by the loop detection algorithm embedded in VINS-mono. Loop detection is triggered if the same place is revisited, and the goal is to minimize the tracking error. As a result, it makes sense that the cross track error of route 2 is lower than that of route 1. Also, if taking a look at route 3, another route where loop detection was triggered, its cross track error is actually greater than route 2. So increase in travel distance increases cross track error is still true. One also may say that the cross track error is not improved a lot on the long corridor test. However, it is worth mentioning that obvious cross track error happens after the second turn. From the second turn to the third turn, this route section has glasses on one side and low textured wall on the other. The insufficient amount of strong feature and highly reflective surface could be the reason to the least cross track error drop. But over 1m reduction in the maximum error could be considered a big enough improvement. Besides, the noticeable drop in the averages of the mean and maximum error show the solid improvement. The decrease in the RMS also further proves that the optimized initialization works. Another factor to be taken into consideration is the goal is to use VINS-mono on unmanned ground vehicle. The original VINS-mono presented [-0.032, 0.09] m final drift on 700m route with loop closure. However, the loop closure of VINS-mono was never triggered during the corridor test. This could have been caused by the insufficient lighting of the corridor, making it hard to detect repeated features. Another reason could be the lack of strong features in the corridor since the tiles and walls do not have sufficient amount of textures. But improvement in final drift was demonstrated even without the loop closure. Additionally, if VINS-mono is applied on unmanned ground vehicle, there could be relatively more forgiveness for the cross track error and final drift. This is because an autonomous car has much bigger dimensions than a drone, making cross track error and final drift more flexible.

6. CONCLUSION AND FUTURE WORK

In this research, an optimized vision-only initialization was proposed based on VINS-mono. It uses a checkerboard pattern as a structured scene to perform a world frame based pose estimation. The results demonstrated lowered cross track on different types of indoor route. Significant differences between the original and optimized VINS-mono were demonstrated in statistics. And the final drift was successfully and greatly decreased on long trajectory. However, further investigation is needed to determine the cause of failure to trigger the loop closure when it was supposed to be triggered on both the original and optimized VINS-mono. Furthermore, in order to solidify the robustness of the optimized VINS-mono, it should be tested on different vehicle platforms such as drones, self-driving cars and hand-held device. It should also be tested in an outdoor environment to make sure it is capable of handling more complexed scenes.

The original VINS-mono is incorporated with a self-recovery when the tracking is interrupted. It automatically starts re-initialization to resume the tracking. However, since the optimized initialization requires a checkerboard pattern in front of the camera, the self-recovery algorithm on the original VINS-mono would not function properly. Although the failure rate with optimized VINS-mono, based on the experience of experimenting, is much lower than the original, it would be optimal to have a new self-recovery algorithm in the future.

During the tests, it was observed that there was noticeable latency between tracking and actually motion. Reduce the time cost is another task that is work considering.

REFERENCES

[1] Review of Visual Odometry: Types, Approaches, Challenges, and Applications (2016), M. O.A. Aqel, M.H. Marhaban, M. I.Saripan, and N.Bt.Ismail, Springer Nature, 2016, 5:1897, DOI 10.1186/s40064-016-3573-7

[2] Monocular Visual Odometry Using A Planar Road Model to Solve Scale Ambiguity (2011), Kitt BM, Rehder J, Chambers AD et al, Proceedings of the European Conference on Mobile Robots, Örebro University, Sweden, p 43–48

[3] Feature Localization Refinement for Improved Visual Odometry Accuracy (2011), A. Cumani, Int. J Circuits Syst Signal Process 5(2):151–158

[4] VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator (2018), T. Qin, P. Li, S. Shen, IEEE Trans. Robot., vol. 34, no. 4, pp. 1004–1020, 2018

[5] A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots (2018), J. Delmerico, D. Scaramuzza, ICRA, 2018, DOI 10.1109/ICRA.2018.8460664

[6] A Multi-State Constraint Kalman Filter for Vision-Aided Inertial Navigation (2007), A. I. Mourikis and S. I. Roumeliotis, IEEE Int. Conf. on Robotics and Automation, 2007, DOI 10.1109/ROBOT.2007.364024

[7] Event-based Visual Inertial Odometry (2017), A. Zhu, N. Atanasov, and K. Daniilidis, CVPR, 2017, DOI 10.1109/CVPR.2017.616

[8] Keyframe-based Visual-inertial Odometry Using Nonlinear Optimization (2015), S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, Int. J. Robot. Research, vol. 34, no. 3, pp. 314–334, 2015

[9] A Combined Corner and Edge Detector (1988), C. Harris and M. Stephens, Proc. Alvey Vision Conf., 1988, pp. 147–151

[10] BRISK: Binary Robust Invariant Scalable Keypoints (2011), S. Leutenegger, M. Chli, and R. Siegwart, ICCV, 2011, DOI 10.1109/ICCV.2011.6126542

[11] A. Agarwal, K. Mierle, and Others, “Ceres solver,” http://ceres-solver.org

[12] Robust Visual-inertial Odometry Using a Direct EKF-based Approach (2015), M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, IEEE/RSJ Int. Conf. on IROS, 2015, DOI 10.1109/IROS.2015.7353389

[13] Faster and Better: A Machine Learning Approach to Corner Detection (2010), E. Rosten, R. Porter, and T. Drummond, IEEE Trans. Pattern Anal. Machine Intell., vol. 32, no. 1, pp. 105-119, 2010, DOI 10.1109/TPAMI.2008.275

[14] SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems (2017), C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, IEEE Trans. Robot., vol. 33, no. 2, pp. 249–265, 2017, DOI 10.1109/TRO.2016.2623335

[15] A Robust and Modular Multi-sensor Fusion Approach Applied to MAV Navigation (2013), S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart, IEE/RSJ Int. Conf. on IROS, 2013, DOI 10.1109/IROS.2013.6696917

[16] Autonomous, Vision-based Flight and Live Dense 3D Mapping with A Quadrotor MAV (2016), M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, J. Field Robot., vol. 33, no. 4, pp. 431–450, 2016

[17] iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree (2012), M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, Int. J. Robot. Research, vol. 31, no. 2, pp. 216–235, 2012

[18] On-manifold Preintegration for Real-Time Visual-Inertial Odometry (2017), C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, IEEE Trans. Robot., vol. 33, no. 1, pp. 1–21, 2017, DOI 10.1109/TRO.2016.2597321

[19] Factor Graphs and GTSAM: A Hands-on Introduction (2012), F. Dellaert, Georgia Institute of Technology, Tech. Rep. GT-RIM-CP&R-2012-002, Sep. 2012

[20] Evaluation of Modern Visual SLAM Methods (2015), A. Huletski, D. Kartashov, and K. Krinkin, Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference, 2015, DOI 10.1109/AINL-ISMW-FRUCT.2015.7382963

[21] ORB-SLAM: A Versatile and Accurate Monocular SLAM System (2015), Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós, IEEE Trans. Robot., vol. 31, no. 5, pp. 1147-1163, 2015, DOI 10.1109/TRO.2015.2463671

[22] ORB: An Efficient Alernative to SIFT or SURF (2011), E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, ICCV, 2011

[23] LSD-SLAM: Large-scale Direct Monocular SLAM (2014), J. Engel, T. Schops, and D. Cremers, Computer Vision-ECCV 2014, pp. 834-849, 2014

[24] Semi-dense Visual Odometry for a Monocular Camera (2013), J.Engel, J. Sturm, and D. Cremers, ICCV, 2013, DOI 10.1109/ICCV.2013.183

[25] Parallel Tracking and Mapping for Small AR Workspace (2007), G. Klein, D. Murray, Proc. International Symposium on Mixed and Augmented Reality, ISMAR, 2007, DOI 10.1109/ISMAR.2007.4538852

[26] Good Features to Track (1994), J. Shi and C. Tomasi, CVPR, 1994, DOI 10.1109/CVPR.1994.323794

[27] Shi-Tomasi Corner Detector & Good Features to Track, https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_shi_tomasi/py_shi_tomasi.html

[28] Monocular Visual-Inertial State Estimation For Mobile Augmented Reality (2017), P. Li, T. Qin, B. Hu, F. Zhu, and S. Shen, ISMAR, 2017, DOI 10.1109/ISMAR.2017.18

[29] Monocular Visual-Inertial State Estimation on 3D Large-Scale Scenes for UAVs Navigation (2017), J. Su, X. Li, Y. Ye, and Y. Li, SSRR, 2017, DOI 10.1109/SSRR.2017.8088162

[30] Vision-Based State Estimation and Trajectory Control Towards High-Speed Flight with a Quadrotor (2013), S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, Robotics: Science and Systems, 2013, DOI 10.15607/RSS.2013.IX.032

[31] Robust Initialization of Monocular Visual-Inertial Estimation On Aerial Robots (2017), T. Qin, S. Shen, IEEE/RSJ Int. Conf. on IROS, DOI 10.1109/IROS.2017.8206284

[32] Monocular Visual-Inertial State Estimation with Online Initialization and Camera-IMU Extrinsic Calibration (2017), Z. Yang, S. Shen, IEEE Trans. on Automation Science And Engineering, vol. 14, no. 1, 2017, DOI 10.1109/TASE.2016.2550621

[33] Epnp: An Accurate O (n) Solution to the PnP Problem (2009), V. Lepetit, F. Moreno-Noguer, and P. Fua, Int. Journal of Computer Vision, vol. 81, no. 2, pp. 155–166, 2009

[34] An Efficient Solution to the Five-point Relative Pose Problem (2004), D. Nist´er, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 6, pp. 756–770, 2004, DOI 10.1109/TPAMI.2004.17

[35] Bundle Adjustment -a Modern Synthesis (1999), B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbonin, Int. Workshop on Vision Algorithms. Springer, 1999, pp. 298–372.

[36] CamOdoCal: Automatic Intrinsic and Extrinsic Calibration of a Rig with Multiple Generic Cameras and Odometry (2013), L. Heng, B. Li, and M. Pollefeys, Proc. IEEE/RSJ Int. Conf. on IROS, 2013, DOI 10.1109/IROS.2013.6696592

[37] A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses (2006), J. Kannala, S.S. Brandt, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 28, no. 8 , 2006, DOI 10.1109/TPAMI.2006.153

APPENDIX A

FEATURE UNDISTORTION

Camera Calibration: Equidistant fisheye model (CamOdoCal by Lionel Heng [36])

- Projection Parameters

$$\begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = \begin{bmatrix} 336.2536 \\ 236.0392 \end{bmatrix}, \begin{bmatrix} m_u \\ m_v \end{bmatrix} = \begin{bmatrix} 277.2371 \\ 276.9978 \end{bmatrix}$$

where $(u_0, v_0)^T$ is the principal point, m_u and m_v are the number of pixels per unit distance in horizontal and vertical directions.

- Distortion Coefficients

$$k_1 = 1$$

$$[k_2, k_3, k_4, k_5] = [1.72804e-2, \ -2.55052e-2, \ 2.26214e-2, -7.33559e-3]$$

Feature Normalization and Undistortion

- Lift detected points to normalized plane

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1/m_u & 0 \\ 0 & 1/m_v \end{bmatrix} \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix}$$

- Use $[x, y]^T$ and the distortion factor to calculate θ and φ, where θ is the angle between the principal axis and incoming ray

- Rectified 2D feature

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} sin\theta cos\varphi \\ sin\theta cos\varphi \\ cos\theta \end{bmatrix} => \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

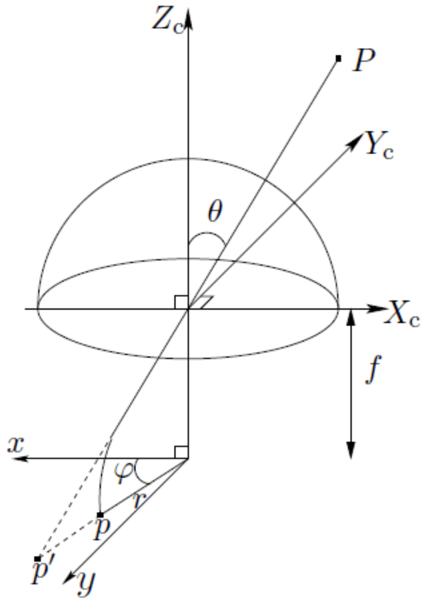

Figure 15. Radially Symmetric Model (Reprinted from [37])

APPENDIX B

3D COORDINATE CALCULATION PSEUDO CODE

Calculation of 3D Cooridinates

Let the inner corner in the first row and first column be the origin of world frame, the pseudocode below is how the 3D coordinates are computed:

```
if the corner number is less than the total number of inner corners
{
   the x coordinate of this corner = (corner number % width) * unitSquare
   the y coordinate of this corner = (corner number / width) * unitSquare
   the z coordinate of this corner = 0
   add this corner to the 3D corners
}
```

APPENDIX C

POSE OPTIMIZATION PROCESS

Find Optimized Poses for Vision-only Sfm

- Detect inner corners of checkerboard in each frame
- Use PnP method to solve the camera pose in each frame using the detected corners and 3D coordinates
- Reproject 3D coordinates to camera frame

$$w\_r\_cam = R^T$$

$$w\_t\_cam = -(w\_r\_cam * T)$$

- Use *ceres* to optimize the camera poses by minimizing the squared RMS between the reprojected and detected coordinates

$$P_{reprojected}(x, y, z) \Rightarrow \ p_{reprojected}\left(\frac{x}{z}, \frac{y}{z}, 1\right)$$

$$p_{detected}(u, v)$$

$$RMS = \sqrt{\frac{\Sigma(p_{reprojected} - p_{detected})^2}{n}}$$

- Recalculate the 2D coordinates of 3D corners in each frame when RMS^2 < 0.005 and pass optimized poses down to visual-inertial alignment