Simultaneous Localization and Mapping

Low-cost Solution Based on HUAWEI P9

VE/VM450 Capstone Design Projects

Summer 2016

Project Duration:

May 16th, 2016 — Aug 10th, 2016

Final Project Report

Prepared by:

Jie Mei
Junjie Shen
Xi Chen
Yi Zhang
Yuhao Li
Undergraduate Engineering Students
UM-SJTU Joint Institute
Team 10

Prepared for:

Dr. Chengbin Ma
Assistant Professor, VM450 Course Coordinator
UM-SJTU Joint Institute

Mr. Zhijie Wang
Company Mentor
Huawei IoV Innovation Centre

Abstract

In this project, we are going to focus on the possibility of SLAM algorithm on HUAWEI P9. The goal is to develop a mobile app on Android platform. The app should be able to map, model, and localize the surroundings using the sensors embedded in the phone such as camera, gyroscope, and accelerometers while the phone is being carried around. The core of our system is a visual SLAM algorithm. Based on accuracy and cell phone storage capacity, we choose ORB-SLAM to be the framework. One camera, one accelerometer and one gyroscope on Huawei P9 are serving as input devices. A tightly coupled visual-inertial sensor fusion algorithm is applied to obtain a camera pose estimation based on the visual and inertial inputs, to be able to work robustly in both slow motion and crowded scenes. We also applied 3D map reconstruction offline on a computer to visualize the SLAM result. Totally speaking, our application is able to realize SLAM on Huawei P9 and the 3D map reconstruction system offline is capable of visualization. Peak CPU usage is 1% beyond our expectation. All other specifications, including localization error, localization time, peak memory usage, internet bandwidth usage, user learning time, and cost have been satisfied. This project indicates a bright future of self-driving car based on smart phones.

Table of Contents

1. INTRODUCTION

Simultaneous localization and mapping has long been a hot topic in which people in past years discover different approaches to improve accuracy and functionality of mapping surroundings as the sensor moves around geographically. This project focuses on the possibility on SLAM algorithms on mobile phones, specifically, Huawei P9. The main goal is to develop an mobile application runnable on Android environment that will map, model, and localize the surroundings of the user as he/she moves around using his/her mobile phone. Worth noticing, the SLAM algorithm should, ideally, be able to tell if the user has entered a loop, meaning he/she has entered an area which he/she has set foot on before. Therefore, the application will be able to identify the situation if the user has moved in a cycle and can successfully localize the user at the same location in the generated map as the point he/she was at previously.

Furthermore, since throughout the project the experiments will be conducted on mobile phones (and should be afterwards in regular situations), the application will rely on several censors on Huawei P9 for information as input to the algorithm. These include the camera and inertia components inside the mobile phone. The camera will take frequent shots of the surroundings and help identify key points, while the inertia components will help gain information such as acceleration that contributes to the calculation of the relative geographic location to any previous point.

Finally, while the application should be able to run on any phone with Android environment, for this project, the team will focus on the performance on Huawei P9, taking advantage of its high performance hardware, aiming to enhance its functionality on this specific phone.

Currently there is a lack of well-known benchmark applicable to SLAM development and application on mobile phones, but in general the team will work on enhancement of accuracy of the mapping and localization in terms of comparison to benchmark. In short, the project team commits to developing high-performance, accurate Android application that provides simultaneous localization and mapping on HUAWEI P9.

2. Literature Search

Monocular visual SLAM studies the problem of utilizing visual information from a single camera to build a sparse or dense 3D model of the scene while traveling through it, and simultaneously reconstruct the trajectory. The problem was initially solved by filtering [1], [2], [3], [4] which processes every frame to jointly calculate the map feature locations and the camera pose. It has the shortages of extensive computation in processing consecutive frames with rarely new information. On the other hand, keyframe-based approaches [5], [6] estimate the map using only selected frames (keyframes), making it possible to save computational resource and perform more accurate bundle adjustment optimizations. The work of LSD-SLAM by Engel et. al [7] proposes to build large scale semi-dense maps, optimizing directly over image pixel intensities whose results are very impressive. They are able to operate in real time, without using GPU acceleration and also to build a dense map. A more recent work of feature-based visual SLAM, known as ORB-SLAM [8], achieves better results than LSD-SLAM with less memory constraints and less computational resources by performing bundle adjustment over extracted visual features. These features enjoy nice properties, such viewpoint invariance and illumination invariance, which are fundamental to performance of visual SLAM systems.

Even though visual signal is great for estimating slow motions, it is subject to outliers such as fast and sudden motions. On the other hand, Inertial Measurement Units (IMUs) generate noisy but outlier-free measurement, allowing to robustly estimate fast and sudden motions. Thus combining the complementary natures of visual information and inertial measurements can contribute positively to robust inference of camera pose.

There has been extensive research investigation in visual-inertial state estimation. Firstly, it is straightforward to apply variations of Bayesian filters [9], [10], to loosely fuse visual and inertial measurements. Such fusion smoothies the vision-based tracking, and the inertial measurements for short-term motion prediction when visual tracking fails. These methods usually lead to suboptimal results even though at lower computational complexity. Recent advancements in visual-inertial fusion show that tightly-coupled methods are able to achieve much better accuracy in terms of estimation accuracy [11], [12]. Thus we will adopt tightly-coupled methods in our design.

3. Customer Requirements & Engineering Specifications

This section describes the customer requirements and engineering specifications of our design. Since the final protocol of our design is expected to be an Android application that applies our modified SLAM algorithm, our target customers are smartphone users who are interested in SLAM algorithms and self-driving technologies. The customer requirements and specifications are hence based on the common characteristics of smartphone users and the requirements from the computer sponsor HUAWEI.

3.1 Customer Requirements

- *Accurate Localization*

Our SLAM algorithm should provide users with accurate location information when they are using their mobile phones to locate themselves in a new environment.

- *Real-time Localization and Mapping*

Our mobile application should provide real-time location and construct a real-time map. This means a user should be able to obtain the location information and a map of his/her surroundings immediately after the environment information was captured by the smartphone.

- *Reasonable Hardware Usage*

This requirement is set for the mobile application we are going to develop. Considering that the mobile device we use has limited hardware resource compared with a low-end computer, and our application needs to share hardware resource with other applications ran on the mobile device, we need to carefully design our application so that it can use minimum hardware resource to achieve the performance requirement.

- *Minimum Internet Usage*

This requirement is also set for the mobile application. Our application may access internet to obtain the GPS information or map information. However, the internet access on a mobile device consumes internet data and power. Therefore, the internet access should be minimized for the users.

- *User-friendly Interface*

A user-friendly Interface is very important for a mobile application to attract new users. This also applies to our mobile application.

3.2 Engineering Specifications

From the customer requirements we derive the following Engineering specifications. Note that we have modifications on the engineering specifications.

- *Localization time [ms]*

The localization time refers to the time for the SLAM system to calculate the camera pose with the environment information. According to the company sponsor Huawei, the target value of this specification should be less than 100ms.

- *Localization error [mm]*

The localization error is the error in estimation of the camera frame's translation and orientation. Both the relative translation and orientation error should be controlled under 5%.

- *CPU Usage and Memory Usage [%]*

CPU usage and memory usage are two metrics used to quantify the hardware usage of an android application. The CPU usage is expected to be less than 50% and the memory usage is expected to be less than 25%.

- *Internet Bandwidth Usage [%]*

Internet Bandwidth usage is a metrics used to quantify the internet usage of an application. It is expected to be less than 10%.

- *Average user learning time [min]*

We decided to use average user learning time as an indicator of the user-friendliness, since it is relatively easy to measure the user learning time and the user learning time is positively correlated with the user-friendliness.

3.3 Quality Function Deployment (QFD) Chart

The QFD chart is shown as following. It is based on the engineering specifications set before DR#2.

	Weight (1-10)	Localization Time	Localization error	Internet Bandwidth Usage	CPU Usage	Memory Usage	Average user learning time	Benchmarks LSD-SLAM	ORB-SLAM	RGB-SLAM
Accurate Localization	10				3	3		3	5	3
Real-Time Localization and Mapping	9	9	9		3	3		4	5	4
User-friendly Interface	8						9		2	
Minimum Internet Usage	7			9	6	6				
Resonable Hardware Usage	7			9	9	9			3	
Measurement Unit		ms	ms	kb/s	%	%	min			
Target Value		28	350	30	25	25	10			
Total		81	81	126	162	162	72			
Normalized		0.12	0.12	0.18	0.24	0.24	0.11			

Figure 3.1: QFD Chart.

The QFD was developed according to the procedure described in the lecture slide.

First, we make binary comparisons for the customer requirements and calculate the customer weights for each customer requirement. As is seen from the chart, the two most important requirements are Accurate Localization and Real-Time Localization. In Fact, these two are the primary requirements set by the company sponsor Huawei.

Then we cross correlate each engineer specification and have these correlations: 1. Tracking time and local mapping time are negatively correlated with the CPU usage and memory usage, since as we increase our hardware usage, the calculation time will be reduced and hence the tracking time and the local mapping time will be reduced. 2. Local mapping is medium negatively correlated with the internet bandwidth usage, since increasing internet bandwidth can reduce the time spent on accessing remote database/GPS and therefore reduce the local mapping time. 3. Absolute KeyFrame Trajectory RMSE is positively related with the CPU usage and memory usage, as increasing the computation accuracy requires more hardware resource.

Next we correlate the customer requirements to the engineering specification. As is mentioned before, Accurate localization will be judged according to the value of absolute KeyFrame Trajectory RMSE; Real-time localization will be judged according to the value of average tracking time and average local mapping time.

Finally, we make the importance rating and normalized target values for each engineering specifications, which are shown in the QFD chart.

4. Concept Generation

The core of our system is a visual SLAM algorithm. There are two state-of-the-art visual SLAM algorithms, namely ORB-SLAM ([8]) and LSD-SLAM ([7]). Here we show a detailed comparison between them. To satisfy the accuracy and storage requirement, we select ORB-SLAM as the framework of our system.

Notably, besides a camera, an Inertial Measurement Unit is also available on a Huawei P9, which consists of an accelerometer and a gyroscope. The IMU measurements can help visual SLAM algorithm better handle fast and sudden motions. Thus we decide to use both sensors in the design.

Table 4.1: Comparison between ORB-SLAM and LSD-SLAM.

	Feature-based SLAM (ORB-SLAM)	**Direct SLAM (LSD-SLAM)**
Matching method	Sparse visual features	Image pixel intensities
Number of Points	300	100,000
Strength	Excellent accuracy Robust in dynamic scene	Robust in low-texture areas Robust under motion blur
Weakness	Subject to motion blur Unstable in low-texture areas	Unstable in dynamic scenes Subject to strong viewpoint/illumination changes

To utilize the merits of both sensors and to make them work seamlessly and robustly, a visual-inertial sensor fusion algorithm is applied. Currently, there are also two main stream methods for this purpose, so called loosely-coupled and tightly-coupled methods. As the names suggest, the loosely-coupled methods ignore intrinsic structure of the problem but apply variants of Kalman filters (EKF, UKF, etc.) straightforwardly ([12]). On the other side, the tightly-coupled methods fully utilize the structure and express the problem as an optimization objective ([14]). In general, tightly-coupled methods have much better accuracy but with higher computational burden, due to solving a non-linear least square problem.

However, because the ORB-SLAM is formulated as an optimization problem which is in a similar form of tightly-coupled methods, the extra computational burden can be ignored and thus the higher accuracy comes at free.

In the validation section, we provide a detailed comparison among vision-only, loose-coupled and tightly-coupled methods, which clearly shows the superiority of tightly-coupled methods.

5. Design Description

This section describes the selected concept in detail. Three main parts are discussed, System Overview, Android Application, and 3D Map Reconstruction.

5.1 System Overview

Our system is the first SLAM system, in academic corpus, that successfully combines visual-inertial sensor fusion with the most accurate and the most comprehensive SLAM algorithm (ORB-SLAM). Before us, many researchers employed visual-inertial fusion on specific application, such as aerial vehicle control and therefore chose to embed a visual-inertial fusion algorithm into a dense SLAM system ([15]) and ignore local mapping and loop-closing for simplicity. In contrary, as a general SLAM solution, our system utilizes every single possible technique, including visual-inertial fusion, local mapping, loop-closing, two-way marginalization to boost SLAM accuracy.

The design uses both camera and Inertial Measurement Unit (IMU) on Huawei P9 to feed data into the SLAM system. In order to obtain robust estimation of camera pose, the visual SLAM algorithm (ORB-SLAM) is augmented with a visual-inertial data fusion algorithm to take advantage of the complementary natures of visual and inertial measurements. Specifically, our system is able to work robustly under fast and slow motions of the mobile devices as well as in crowded scenes.

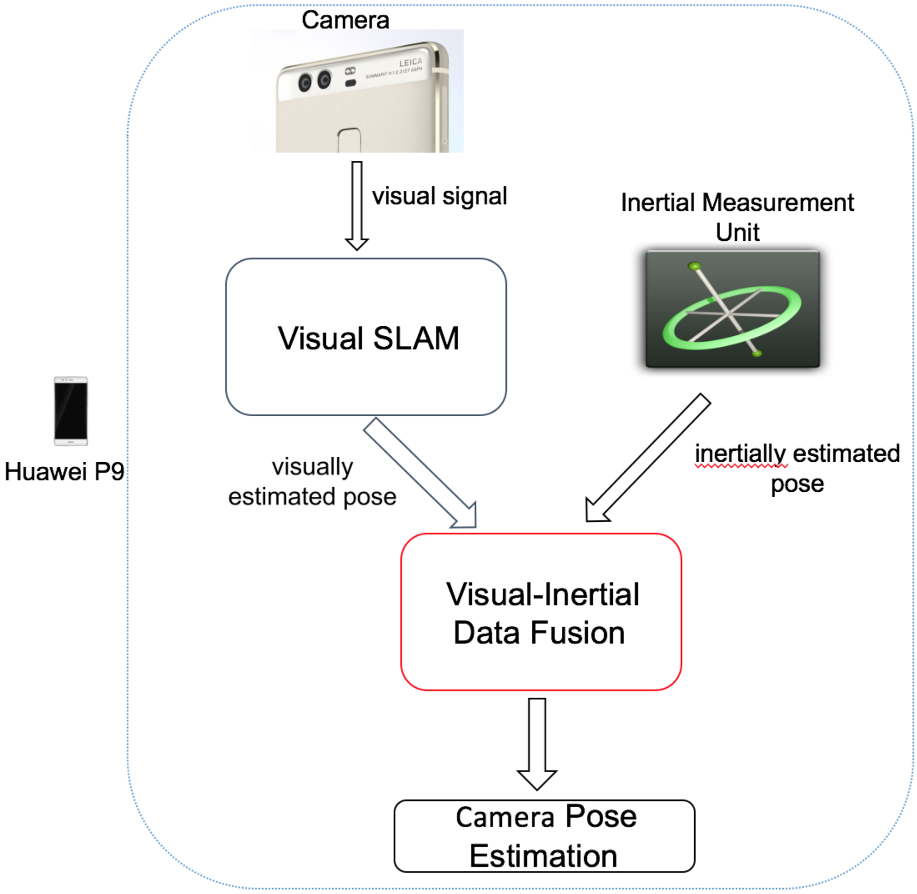


Figure 5.1: System Overview.

It's well-known ([15]) that visual signal can be utilized to estimate slow motions but it usually loses tracking under fast and sudden motions. On the other hand, inertial signal is noisy and subject to bias drift but it's able to estimate fast motions using proper filtering mechanism.

In our system, specifically, the visual signal captured by the camera goes through a visual SLAM algorithm, ORB-SLAM in this case, to generate visually estimated poses. At the mean time, the IMU generates noisy but outlier-free inertial measurements of the camera pose as well. Then a visual-inertial data fusion algorithm is adopted to take advantage of the complementary natures of both signals. The pose estimations going out of the pipeline are expected to be robust under both slow and fast motions.

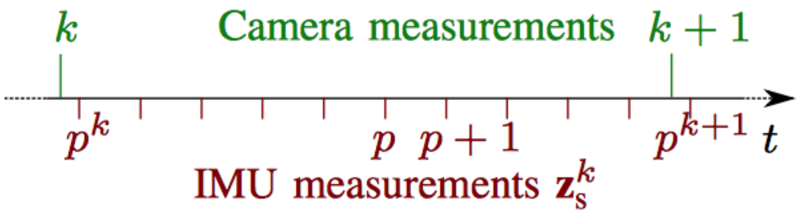


Figure 5.2: Asynchronous measurement rates of camera and IMU.

For sake of limited space, we omit the details of common visual SLAM algorithms and we refer interested readers to a complete survey [14] for details. However, we emphasize our unique contribution, with detailed mathematical derivation, of augmenting the state-of-the-art feature based visual SLAM algorithm (ORB-SLAM) with inertial information to circumventing the well-known limitations faced by all visual SLAM systems, such as scale ambiguity and poor fast motion estimation.

From a high-level perspective, the tightly coupled visual-inertial SLAM can be captured by a non-linear least square optimization problem over state vectors χ which is formally defined as

$$\begin{aligned}\mathcal{X} &= [\mathbf{x}_0^0, \mathbf{x}_1^0, ..., \mathbf{x}_N^0] \\ \mathbf{x}_k^0 &= [\mathbf{p}_k^0, \mathbf{v}^k, \mathbf{q}_k^0] \\ \mathbf{p}_0^0 &= [0,0,0], \ \mathbf{q}_k^0 = [0,0,0,1].\end{aligned}$$

$$\begin{aligned}\min_{\mathcal{X}} \ (\mathbf{b}_p - \mathbf{\Lambda}_p \mathcal{X}) + \sum_{k \in S_i} ||r_{S_i}(\hat{\mathbf{z}}_{k+1}^k, \mathcal{X})||^2_{\mathbf{P}_{k+1}^k} + \\ \sum_{(i,j) \in S_c} ||r_{S_c}(\hat{\mathbf{z}}_i^j, \mathcal{X})||^2_{(\mathbf{W}_i^j)^{-1}\mathbf{P}_i^j}\end{aligned}$$

The state vectors χ represents a trajectory of the camera frame, which consist of the position, rotation, and velocity estimates from the initial time instant 0 to the current time instant k. In the optimization object, the first term is a prior regularization on the χ, calculated from two-way marginalization ([15]). The third term is the core of a visual SLAM algorithms, which searches for plausible camera poses that could explain the matched visual feature points among consecutive frames. Most importantly, the second term represents inertially measured motion constraints between two consecutive frames. For example, we can simply integrate the readings between two frames to get an approximate estimation of the rotate the camera encountered during the period. As a matter of fact that gyroscope readings are accurate in short term, we know the final solution to the optimization above shouldn't deviate too much from this estimation. The specific form for the inertial regularization term will be defined soon.

In practice, on Huawei P9, the IMU measurement rates (~100Hz) is much higher than camera measurement rate (~30Hz) and these rates vary through time. As a result, our system assumes asynchronous measurement rates of camera and IMU, as shown in Figure 5.2. Specifically, a non-exhaustively running thread keeps collecting and processing IMU measurements between consecutive captured camera frames, to generate inertially measured motion constraints.

Mathematically, assuming the axes of camera and IMU are perfectly aligned, between the k-th and the k+1-th key camera frames, the inertial information is summarized as a 10-dimensional vector $\hat{z}_{k+1}^{k}$:

$$\hat{\mathbf{z}}_{k+1}^{k} = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{k+1}^{k} \\ \hat{\boldsymbol{\beta}}_{k+1}^{k} \\ \hat{\mathbf{q}}_{k+1}^{k} \end{bmatrix} = \begin{bmatrix} \iint_{t\in[k,k+1]} \mathbf{R}_t^k \mathbf{a}_t^b dt^2 \\ \int_{t\in[k,k+1]} \mathbf{R}_t^k \mathbf{a}_t^b dt \\ \int_{t\in[k,k+1]} \frac{1}{2} \begin{bmatrix} -\lfloor \boldsymbol{\omega}^{b_t} \times \rfloor & \boldsymbol{\omega}^{b_t} \\ -\boldsymbol{\omega}^{b_t\,T} & \mathbf{0} \end{bmatrix} \mathbf{q}_t^k dt \end{bmatrix}$$

where a_t^b and ω^{b_t} are the instantaneous linear acceleration and angular velocity in the IMU body frame at time t respectively, while R_t^k is the 3D rotation matrix of the camera frame from time instant t with respect to time instant k, and q_t^k is its corresponding quaternion representation.

Finally, we can define the regularization term for the inertial measurements as:

$$\begin{aligned} r_{S_i}(\hat{\mathbf{z}}_{k+1}^{k}, \mathcal{X}) &= \begin{bmatrix} \delta\boldsymbol{\alpha}_{k+1}^{k} \\ \delta\boldsymbol{\beta}_{k+1}^{k} \\ \delta\boldsymbol{\theta}_{k+1}^{k} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{R}_0^k(\mathbf{p}_{k+1}^0 - \mathbf{p}_k^0 + \mathbf{g}^0 \frac{\Delta t^2}{2}) - \mathbf{v}^k \Delta t - \hat{\boldsymbol{\alpha}}_{k+1}^{k} \\ \mathbf{R}_0^k(\mathbf{R}_{k+1}^0 \mathbf{v}^{k+1} + \mathbf{g}^0 \Delta t) - \mathbf{v}^k - \hat{\boldsymbol{\beta}}_{k+1}^{k} \\ 2[(\hat{\mathbf{q}}_{k+1}^{k})^{-1} (\mathbf{q}_k^0)^{-1} \mathbf{q}_{k+1}^0]_{xyz} \end{bmatrix} \end{aligned}$$

where the vector g^0 is the gravity and the vector v^k is the velocity of the camera frame at time instant k.

As the core of ORB-SLAM system is solving Bundle Adjustment problems (similar to the third term in the optimization objective), adding two more regularization terms incur little extra computational complexity and makes the overall optimization problem more well-behaved by eliminating the scale ambiguity.

5.2 Android Application

The main prototype of our project is the Android application to deliver. The application, developed and tested on Android studio, is run on HUAWEI P9. The application, in short, lets users use camera to select certain view angles and the application correspondingly will analyze and process the images to produce respective key points and visual pose and other SLAM results.

Figure 5.3: App welcome splash screen.

As far as the application works to the user, the application first opens and shows a welcome splash screen to indicate to the user who developed and/or supported the application and show a logo of the application. Afterwards the screen will appear as a double-layered camera screen with the smaller screen appearing as if the user is just videoing the surroundings and the bigger screen will show the same images but with key points appearing as small dots and visual pose estimation appearing as green strokes. This is the application reading information from the camera of the phone's surroundings and analyzing to produce visual SLAM results. At the meantime, inertia measurement sensors will also help the process by integrating its inertially estimated pose and visually estimated pose from the camera to achieve the final estimation of the camera pose.

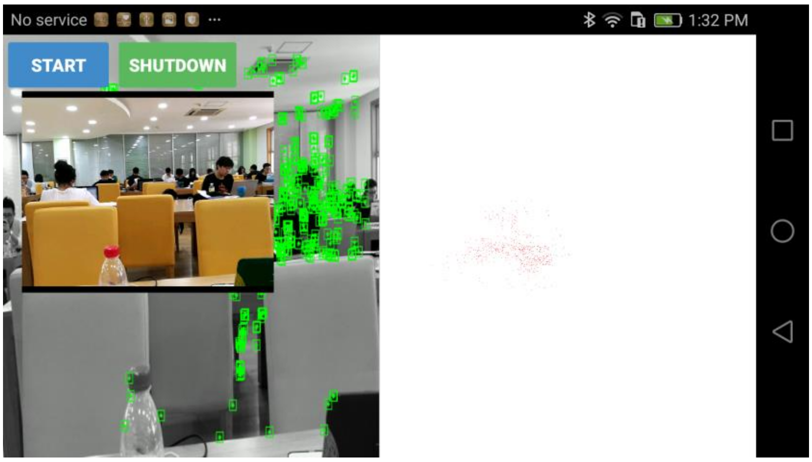


Figure 5.4: App GUI.

The android application architecture is roughly a two-layer architecture, which is shown in the following figure. On the top of the layers is the JAVA layer, which provides the user-interface, interaction with the camera and inertia measurement units, result visualization. In the JAVA layer, we use OpenCV camera modules and android sensor APIs to capture camera frames from camera and sensor information from inertia units. Furthermore, we use OpenGL to visualize the camera poses and map points. We decided to implement these functions in the JAVA layer since they are not performance critical. At the bottom is the C++ native layer, which consists of the SLAM module and the Dalvik Virtual Machine. The Dalvik virtual machine provides communication between the JAVA layer and the actual hardware. The SLAM system calculates the map points in each frame and estimate the camera poses based on the input information. We decided to use the state of the art ORB-SLAM system and add our novel data-fusion algorithm into it so that it can process both inertia information and visual information. We chose to implement the SLAM system with native C++, because it is performance-critical.

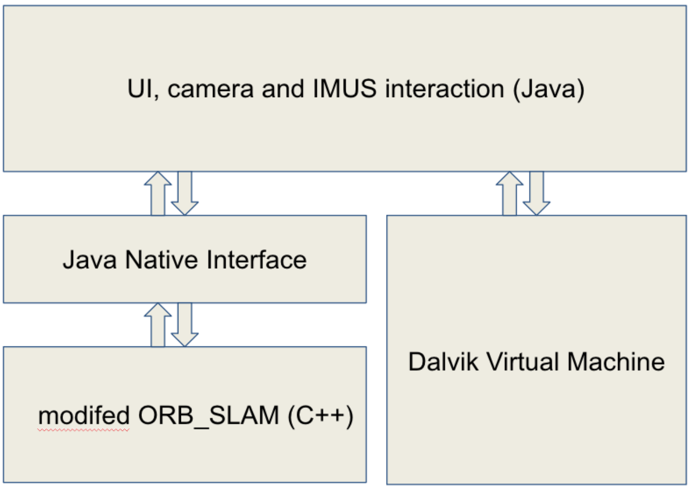


Figure 5.5: App Architecture.

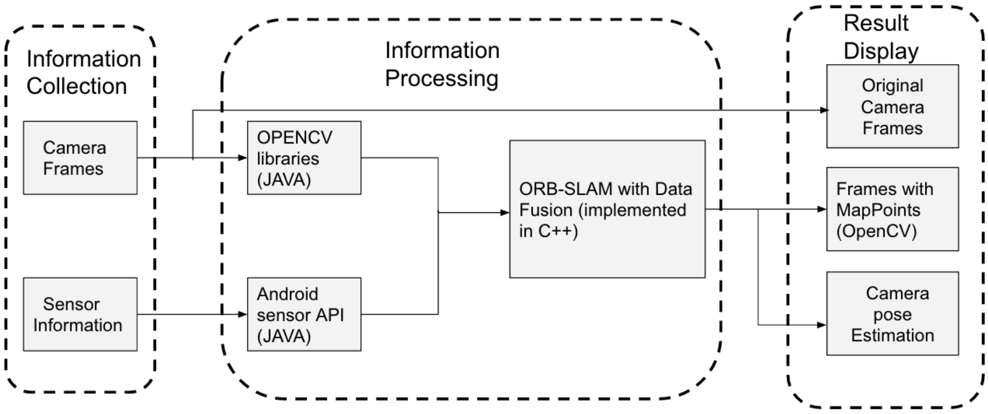


Figure 5.6: Data-flow in the App.

The application works in all scenarios geographically in that whether the environment is crowded or roomy, whether the user is moving fast or slow while using the application, while the user is moving among a vast or small area, the application will work just as normally, smoothly, and correctly.

5.3 3D Map Reconstruction

In order to visualize our SLAM result, we applied 3D Map Reconstruction to make it more realistic.[1] Figure 5.7 shows the flow chart of 3D Map Reconstruction. More details are illustrated in APPENDIX B.

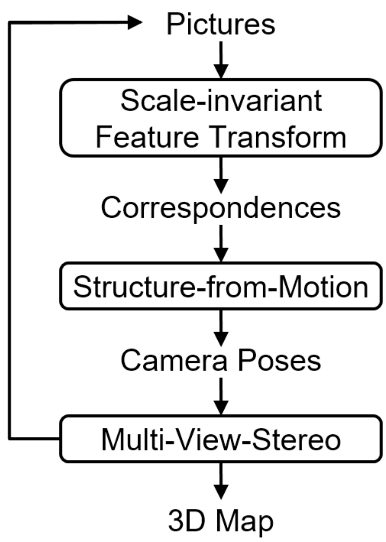


Figure 5.7: 3D Map Reconstruction Flow Chart.

[1] In DR#3, Prof. Ma asked why 3D Map Reconstruction is applied since it is not a requirement.

- *Scale-invariant Feature Transform*

First, we took a video of the structure by HUAWEI P9. Then, we captured photos from the video. For each photo, we identified correspondences/feature points, especially corner points, using the algorithm called SIFT (Scale-invariant Feature Transform developed by David Lowe). Furthermore, we could match these feature points between any pair of these photos.

- *Structure-from-Motion*

With the matched feature points, we could calculate the camera poses. The algorithm, called SfM (Struture-from-Motion developed by Noah Snavely), mainly applied the knowledge of rigid body motion in 3D, Relative Motion, and triangulation to reflect the camera motion between two adjacent photos. Taking the initial camera pose as the key coordinate system, we were able to get all the camera poses (each one corresponding to each photo) from the calculated camera motions. The next step is just to transform all the feature points from the auxiliary coordinate systems into the key coordinate system so as to obtain sparse point clouds of the 3D map.

- *Multi-View-Stereo*

The sparse point clouds are not as visualized as we actually can see from the structure. Therefore, the algorithm called MVS (Multi-View-Stereo developed by Yasutaka Furukawa) is implemented subsequently. The main idea is to transform sparse point clouds into dense point clouds by iteratively expanding feature points to nearby locations [13].

Figure 5.8 shows the 3D map reconstruction of JI Building. Figure 5.9 shows the 3D map reconstruction of Yu Liming Center. The quality is influenced by the video clarity, distortion, brightness, etc. Although it is not perfect, the location can be clearly identified and we think that’s enough for our application.

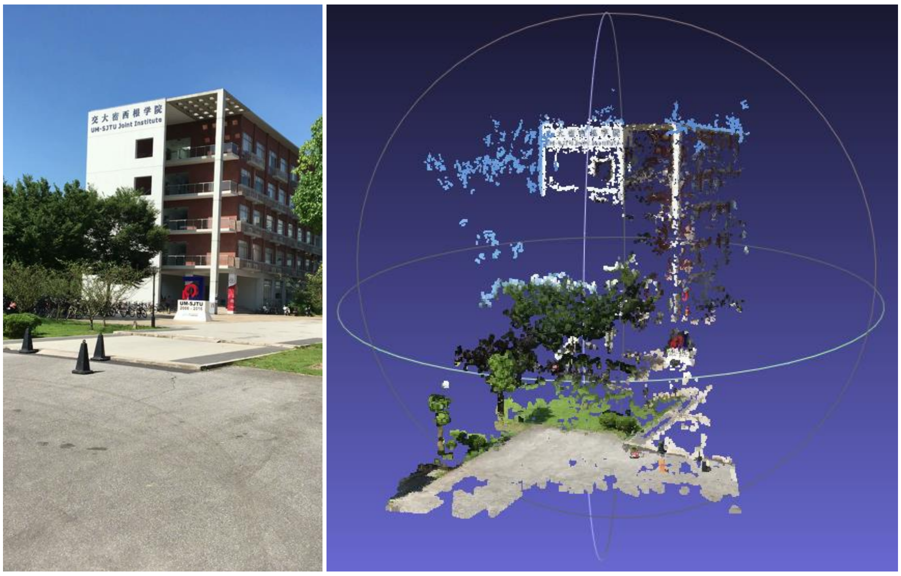


Figure 5.8: 3D Map Reconstruction of JI Building.

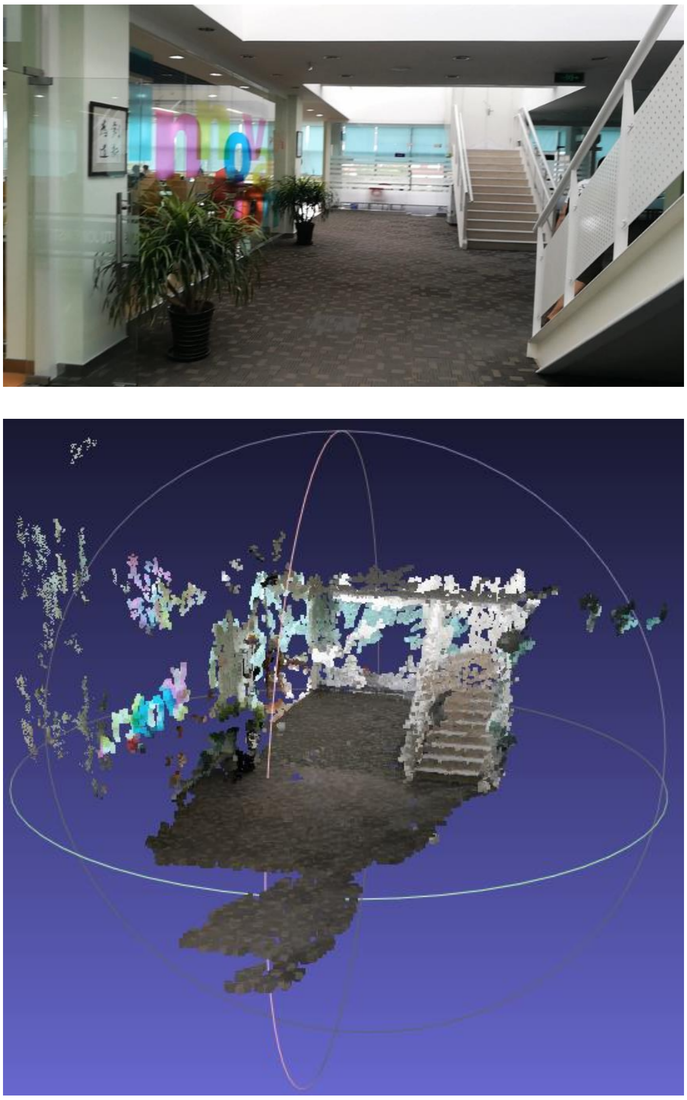


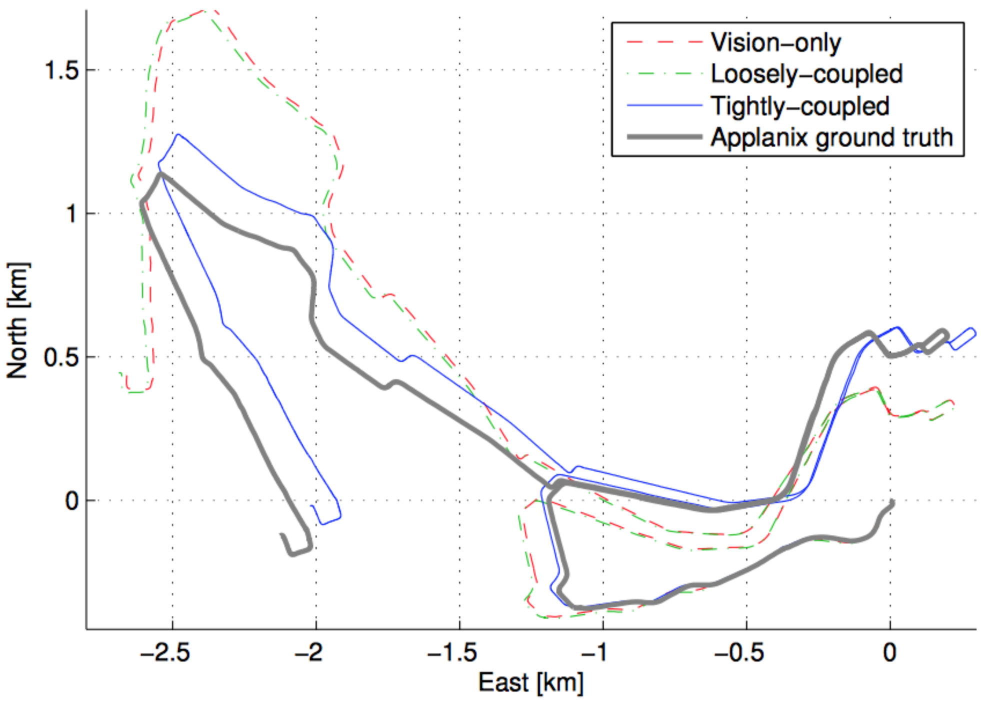
Figure 5.9: 3D Map Reconstruction of Yu Liming Center.

6. Validation Plan

Upon finishing implementing, we test for following engineering specifications:

- Localization error
- Frame rate
- Localization time
- CPU usage
- Memory usage
- User learning time
- Cost

Firstly, we validate our design choices in terms of localization errors by comparing vision-only, loosely-coupled and tightly-coupled (ours). As a matter of fact that there is no ground truth data available in real-life environment, we run all of the experiments on a PC using EuRoC MAV dataset ([16]) which is an open benchmark for visual-inertial SLAM. Since the localization accuracy depends mainly on the algorithm instead of specific platforms. We are confident that the test results on a PCs can truly reflect the competence of the algorithm on a mobile device.



Above is a 2D map constructed using different SLAM methods on a long term route in the EuRoC MAV dataset. This result is firstly reported by [17] and we replicate their experimental result by our own. As expected, the loosely-coupled approach shows approximately no performance gain over the vision-only method, while our tightly-coupled method exhibits the closest route to the ground truth.

We also run several short-term indoor experiments to get quantitative results on localization error. As we found out, the translation error <= 5% and orientation error <= 2% which both decrease linearly as the distance traveled increases.

To measure the localization time, we monitored the frame rate of the SLAM system, i.e., how many camera frames our mobile system can process per second, and used the relation $Localization\ time = \frac{1}{frames\ rate}$ to obtain the localization time. We tested the mobile application in three different scenes, including Yu Liming Center, a hallway in the dormitory and the road next to the JI building. Based on our observation, the frame rate of our system is never below 15 fps in different scenes, so we concluded that the localization time of our system should not be greater than 66.7 ms, which is well below the target value required by the company sponsor.

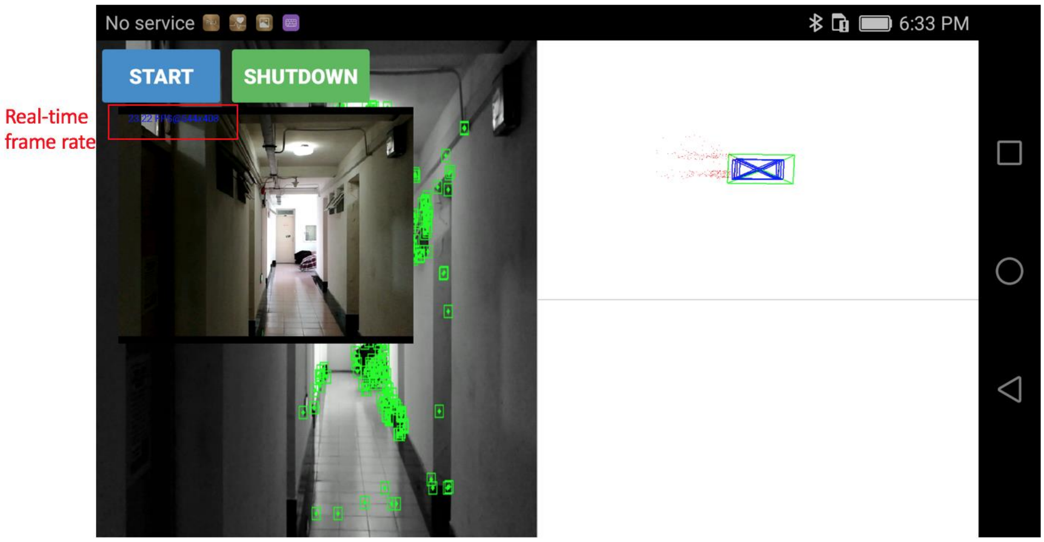


Figure 6.1: Monitoring real-time frame rate

To measure the CPU usage and Memory usage, we utilized the Android Debugging Bridge (ADB) to set up shell access to the Huawei P9 device, and then monitored the CPU usage and memory usage via shell access from the laptop using the **top** command in the shell. The **top** command displays real-time information of processes running on the android kernel, including the CPU usage (in %) and the physical memory usage (in KB). We monitored the real-time CPU usage and memory usage while we tested the mobile application in three different scenes mentioned above.

Based on our observation, the peak CPU usage is 51% and the peak physical memory usage is around 508MB (12.4% of the total physical memory) among three different scenes. Therefore, we conclude that the peak CPU usage slightly exceeds the constraint and the peak memory usage is well below the constraint.

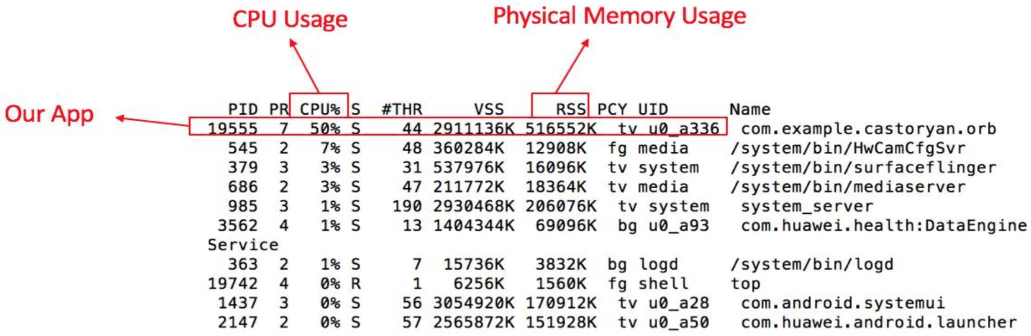


Figure 6.2: Monitorint the CPU and Memory Usage with adb

To measure the user learning time, we randomly picked 10 people and measured the average time for to them to learn to use our android application. The result showed that all the users learnt to use the application within 1min. This is due to our simple UI design.

For internet bandwidth usage, since the mobile application does not use network, the internet bandwidth usage is 0%.

For cost, since we only spend 3,688 RMB on our Huawei P9 phone, the cost in this project is 3,688 RMB.

To sum up, followings are our outcomes for the validation results.

- **Localization error**:

  translation = 5% <= 5%, orientation 2% < 5%

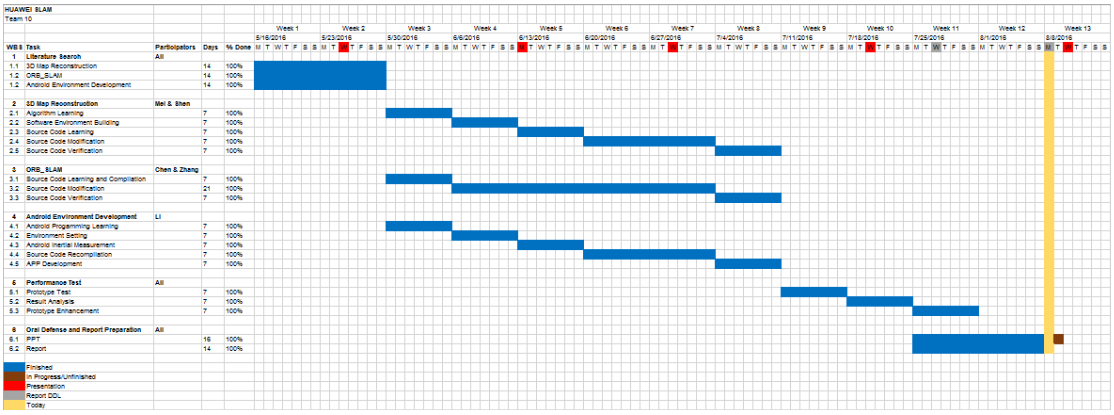
- **Localization time** = 66.7 ms < 100ms
- **Peak CPU usage** = 51% > 50%
- **Peak Memory usage** = 12.4% < 25%
- **Internet Bandwidth usage** = 0% < 10%
- **User learning time** = 1min < 10min
- **Cost** = 3,688 RMB < 5,000 RMB

7. Project Plan

This section presents the schedule of our project in the form of Gantt Chart, each team member's work assignment, and the budget.

7.1 Gantt Chart

Figure 7.1 shows the Gantt Chat of our project. By June 16, we finished our literature research. That is, we had every material we needed and every tool we used. As scheduled, By June 30, we managed to migrate the ORB_SLAM source code into visual-inertial algorithm. Later, we focused on the recompilation of the source code from Linux platform to Android platform. This assignment lasted for about two weeks. At meantime, the development of GUI was taken into action as well. Then, at the end of July, we started to test the performance of our prototype. Especially, accuracy and efficiency were considered as two most important characteristics.



HUAWEI SLAM																		
Team 10																		
					Week 12							Week 13						
					8/1/2016							8/8/2016						
WBS	**Task**	**Participators**	**Days**	**% Done**	M	T	W	T	F	S	S	M	T	W	T	F	S	S
1	**Literature Search**	**All**																
1.1	3D Map Reconstruction		14	100%														
1.2	ORB_SLAM		14	100%														
1.2	Android Environment Development		14	100%														
2	**3D Map Reconstruction**	**Mei & Shen**																
2.1	Algorithm Learning		7	100%														
2.2	Software Environment Building		7	100%														
2.3	Source Code Learning		7	100%														
2.4	Source Code Modification		7	100%														
2.5	Source Code Verification		7	100%														
3	**ORB_SLAM**	**Chen & Zhang**																
3.1	Source Code Learning and Compilation		7	100%														
3.2	Source Code Modification		21	100%														
3.3	Source Code Verification		7	100%														
4	**Android Environment Development**	**Li**																
4.1	Android Progamming Learning		7	100%														
4.2	Environment Setting		7	100%														
4.3	Android Inertial Measurement		7	100%														
4.4	Source Code Recompilation		7	100%														
4.5	APP Development		7	100%														
5	**Performance Test**	**All**																
5.1	Prototype Test		7	100%														
5.2	Result Analysis		7	100%														
5.3	Prototype Enhancement		7	100%														
6	**Oral Defense and Report Preparation**	**All**																
6.1	PPT		16	100%														
6.2	Report		14	100%														
	Finished																	
	In Progress/Unfinished																	
	Presentation																	
	Report DDL																	
	Today																	

Figure 7.1: Gantt Chart.

We completed our project at the beginning of August. Currently, we are just on schedule and preparing ourselves for the oral defense and final report.

7.2 Work Assignment

Mei and Shen are focusing on the implementation of 3D map reconstruction algorithm. It is called multi-view-stereo (MVS), developed by Yasutaka Furukawa, a Japanese professor.

Chen and Zhang are learning the structure of ORB_SLAM source code. Later, they will concentrate on the modification and development for our use.

Li is working on the recompilation of the source code from Linux platform to Android platform. As mentioned, our project will be finally operated on HUAWEI P9, which is an Android-based smartphone.

7.3 Budget

Note that our project is a pure software development and the only hardware we will use is an Android-based smartphone, which can be borrowed conveniently. After discussion with Prof. Chengbin Ma, we bought HUAWEI P9 for 3,688 RMB. As a result, the total cost of our project is 3,688 RMB by now.

8. Conclusion

Our application is able to realize SLAM on Huawei P9 and the 3D map reconstruction system offline is capable of visualization. Peak CPU usage is 1% beyond our expectation. All other specifications, including localization error, localization time, peak memory usage, internet bandwidth usage, user learning time, and cost have been satisfied.

9. Acknowledgements

This project was carried out under the direction of Mr. Zhijie Wang from HUAWEI, together with the help of Prof. Chengbin Ma and Prof. Mian Li from UM-SJTU Joint Institute. Many thanks to them!

REFERENCES

[1] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pp. 1052–1067, 2007.

[2] J. Civera, A. J. Davison, and J. M. M. Montiel, “Inverse depth parametrization for monocular SLAM,” IEEE Transactions on Robotics, vol. 24, no. 5, pp. 932–945, 2008.

[3] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, “Structure from motion causally integrated over time,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 4, pp. 523–535, 2002.

[4] E. Eade and T. Drummond, “Scalable monocular SLAM,” in IEEE Com- puter Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, New York City, USA, June 2006, pp. 469–476.

[5] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, “Real time localization and 3d reconstruction,” in Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on, vol. 1, 2006, pp. 363–370.

[6] G. Klein and D. Murray, “Parallel tracking and mapping for small AR workspaces,” in IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), Nara, Japan, November 2007, pp. 225–234.

[7] J. Engel, T. Scho ̈ps, and D. Cremers, “LSD-SLAM: Large-scale direct monocular SLAM,” in European Conference on Computer Vision (ECCV), Zurich, Switzerland, September 2014, pp. 834–849.

[8] Mur-Artal, Raul, J. M. M. Montiel, and Juan D. Tardós. "Orb-slam: a versatile and accurate monocular slam system." *IEEE Transactions on Robotics* 31.5 (2015): 1147-1163.

[9] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, “Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor,” in Proc. of Robot.: Sci. and Syst., Berlin, Germany, 2013.

[10] D. Scaramuzza, M. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, A. Martinelli, M. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss, “Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments,” IEEE Robot. Autom. Mag., vol. 21, no. 3, 2014.

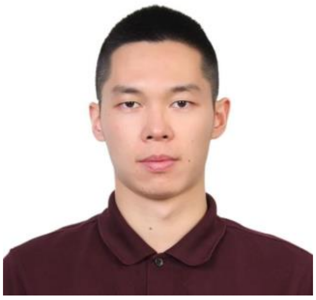
[11] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, “Consistency analysis and improvement of vision-aided inertial navigation,” IEEE Trans. Robot., vol. 30, no. 1, pp. 158–176, Feb. 2014.

[12] M. Li and A. Mourikis, “High-precision, consistent EKF-based visual-inertial odometry,” Intl. J. Robot. Research, vol. 32, no. 6, pp. 690–711, May 2013.

[13] R. Fergus, “Multi-view Stereo & Structure from Motion,” [Online]. Available: http://cs.nyu.edu/~fergus/teaching/vision/11_12_multiview.pdf.
[14] Aulinas, Josep, et al. "The SLAM problem: a survey." CCIA. 2008.
[15] Y. Ling, T. Liu, S. Shen, “Aggressive Quadrotor Flight Using Dense Visual-Inertial Fusion.” ICRA. 2016.
[16] Burri, Michael and Nikolic, Janosch and Gohl, Pascal and Schneider, Thomas and Rehder, Joern and Omari, Sammy and Achtelik, Markus W and Siegwart, Roland, “The EuRoC micro aerial vehicle datasets”, International Journal of Robotic Research, 2016.
[17] Leutenegger, Stefan, et al. "Keyframe-based visual–inertial odometry using nonlinear optimization." The International Journal of Robotics Research, 2015.

APPENDIX A: Biographical Sketches

1. Jie Mei



This is Jie Mei. I am a graduate student learning Naval Architecture and Marine Engineering in University of Michigan, and an undergraduate student learning Mechanical Engineering in SJTU-UM Joint Institute. I am going to finish my graduate school in May, 2017. However, I do not have a very clear future plan yet. Due to the poor oil market, getting a job in ship building or oil company is nearly impossible. So I am switching to mechanical engineering again starting from next semester. My life is composed of study, basketball, reading, and visiting my girlfriend, who is working in Hong Kong. I need to keep hard working on hunting for a decent job to live with her, with some dignity.

2. Junjie Shen

I am Junjie Shen, a senior ME student at UM-SJTU Joint Institute. I am born in Shanghai and have never left for long. My parents both major in ME and this may be one of the main reason why I choose ME as my career. As a teenager, I enjoyed building models and making origamis. I also attended a lot of science and technology competitions and won many of them, which contributes to another prime reason why I choose ME. Essentially, I succeeded in going to the Joint Institute. After four-year ME undergraduate education, I am capable of using CAD software, LabVIEW, MATLAB, etc. My research of interest lies in Systems and Control, Mechanisms Design, and Robotics. After graduation, I plan to further my graduate education in the United States. In fact, I have been admitted as a PhD student in the Mechanical and Aerospace Engineering Department at UCLA, which is a great honor. I will continue to major in ME and especially concentrate on the area of robotics. Subsequent to my PhD program, I plan to come back to China to continue working on projects related to my dissertation. Practical application area of my research will be robotics and the Chinese robotics market is in a great need of researchers. At my leisure time, I like doing sports, particularly, basketball and jogging. I am also fond of movies and music for relaxation. Thank you!

3. Xi Chen

I am Xi Chen, a senior ECE student at UM-SJTU Joint Institute and just graduated as a CSE student at University of Michigan, Ann Arbor. I'm born in Tianjin, a northern city, and moved to Shanghai when I was 9. My parents are both in ME and this may be one of the main reasons I chose ECE instead of ME. As a teenager, I never really considered either ME or ECE as a future career, even after I graduated from high school. In a way, I entered the Joint Institute without knowing what I want to do in my later career in advance. With my parents objection I opted out ME and chose ECE and pretty much liked it freshman and sophomore year. After data structure and algorithm class in UMich, I realized I want to be in the CS area. After two years at SJTU and two years at UMich, I'm capable of programming in C/C++, Python, and some PHP, Javascript, and a bit of Java. My interest lies in Natural language processing. After graduation I plan to start my career and join Microsoft in Seattle as a software engineer in Windows operating system team. At my leisure time, I'm a huge movie fan and I love eating at newest and best restaurants and bakeries. I also love skiing, which I'm not able to enjoy in Shanghai, unfortunately. Thank you!

4. Yi Zhang

I'm Yi Zhang, a senior ECE student at UM-SJTU Joint Institute, and have received a BSE degree in Computer Science from University of Michigan, Ann Arbor. I'm specifically interested in Artificial Intelligence and I have been involved in front-end research of machine learning under mentorship of Prof. Honglak Lee since my junior year. Fortunately, I have published two papers at top conferences of Machine Learning during my undergraduate years. After graduation, I'll pursue a PhD degree in Computer Science at Princeton University and will hopefully continue to contribute to Artificial Intelligence research. At leisure time, I play basketball and work out a lot. I'm also a big fan of movie and music.

5. Yuhao Li

I am Yuhao Li, a senior ECE student at UM-SJTU Joint Institute. I just received my B.S.E. in Computer Engineering from the University of Michigan. After graduation, I am going to pursue a Ph.D. in Computer Science at Duke University. My research interest lies in the interface of computer architecture and computer system. Specifically, I am interested in research topics about large-scale datacenter design, such as energy efficiency, resource allocation, etc. I am a huge fan of TV series. I have been supporting Arsenal F.C. (an English professional football club) for 14 years.

APPENDIX B: 3D Map Reconstruction

VisualSFM (http://ccwu.me/vsfm/) is an open source software developed by Changchang Wu under GPL licence. It is used for 3D reconstruction using SfM. The system integrates several projects to obtain sparse point clouds of the structure. However, for dense reconstruction, this program requires the execution of Yasutaka Furukawa's CMVS/PMVS tool chain, which can be obtained from http://francemapping.free.fr/Portfolio/Prog3D/CMVS.html.

Figure B-1 shows the operation process overview.

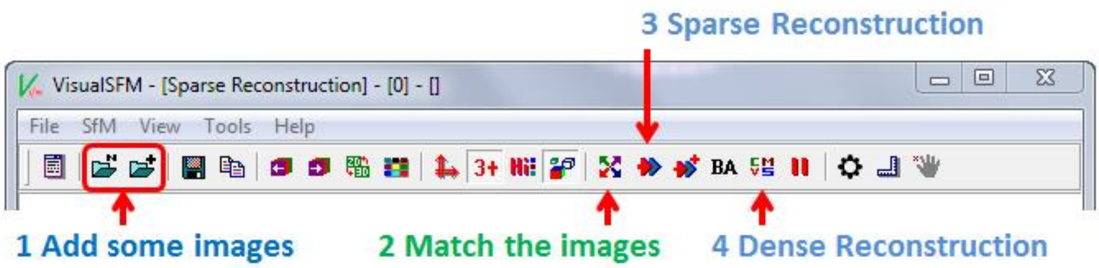
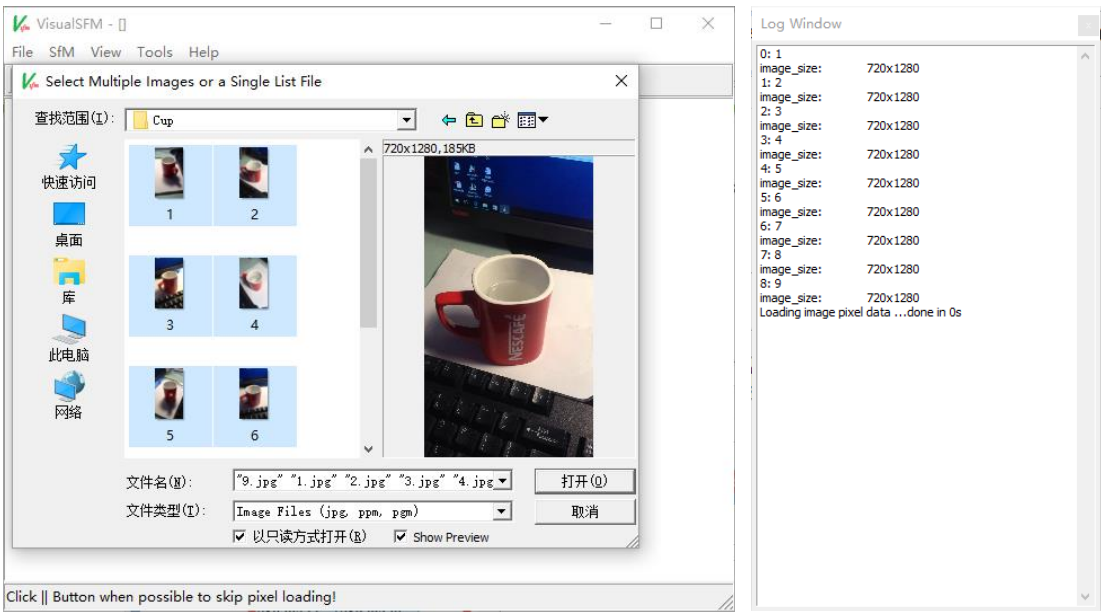


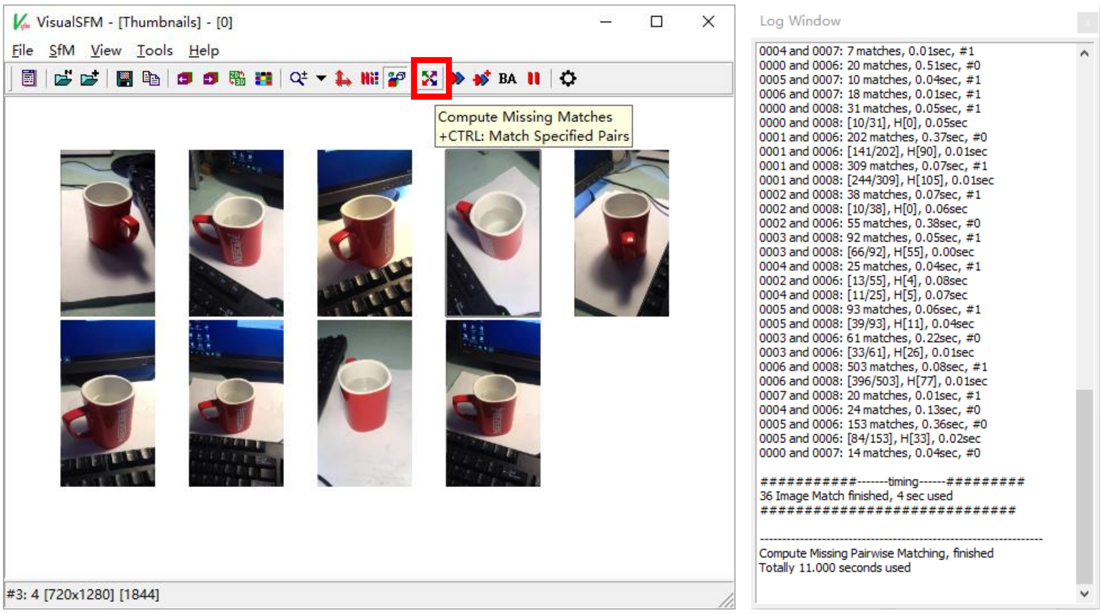
Figure B-1: Operation Process Overview. Retrieved from http://ccwu.me/vsfm/.

Step 1: Add some images.

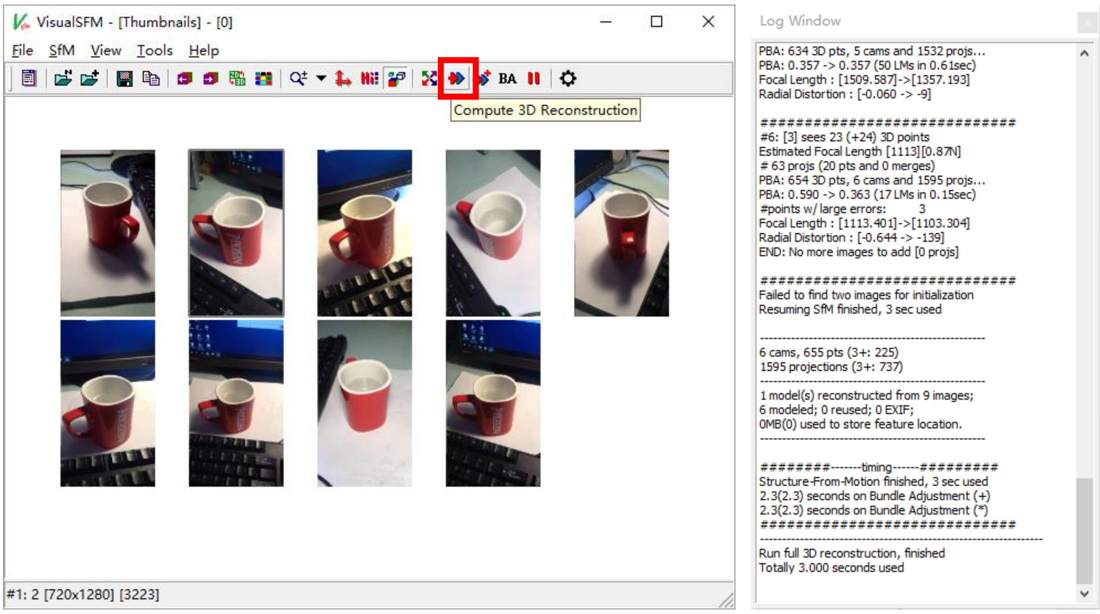


You can see the image data from the log window.

Step 2: Match the images.

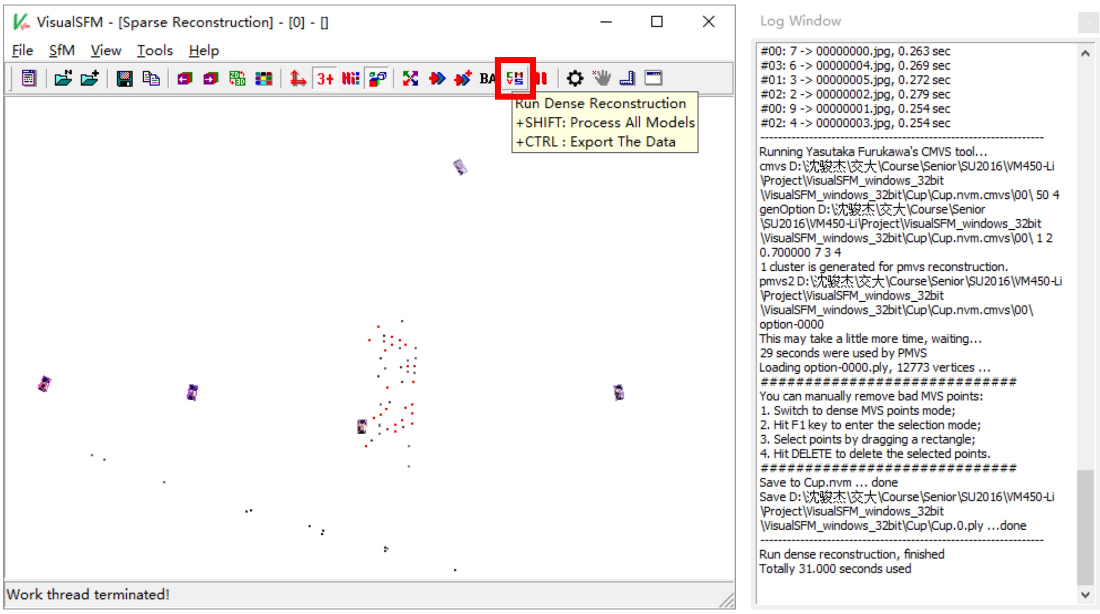


Step 3: Sparse Reconstruction.

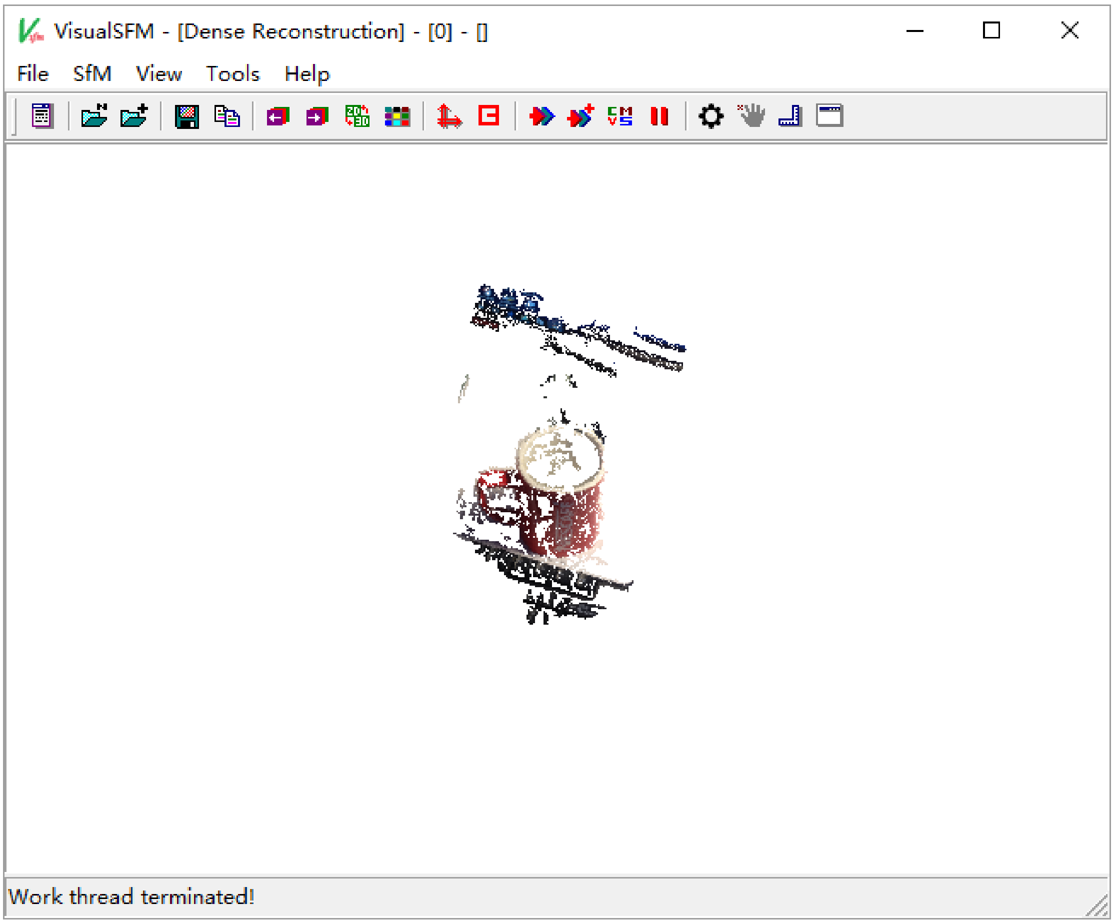


Camera poses are calculated first.

Step 4: Dense Reconstruction.



Step 5: Press Tab to see the result.



The corresponding 3D Model file (.ply) can be found in the root directory.