Photogrammetric 3D reconstruction using mobile imaging

Dieter Fritsch, Miguel Syll
Institute for Photogrammetry, University of Stuttgart, Geschwister-Scholl-Strasse 24D, D-70174 Stuttgart, Germany

ABSTRACT

In our paper we demonstrate the development of an Android Application (AndroidSfM) for photogrammetric 3D reconstruction that works on smartphones and tablets likewise. The photos are taken with mobile devices, and can thereafter directly be calibrated using standard calibration algorithms of photogrammetry and computer vision, on that device. Due to still limited computing resources on mobile devices, a client-server handshake using Dropbox transfers the photos to the sever to run AndroidSfM for the pose estimation of all photos by Structure-from-Motion and, thereafter, uses the oriented bunch of photos for dense point cloud estimation by dense image matching algorithms. The result is transferred back to the mobile device for visualization and ad-hoc on-screen measurements.

Keywords: Mobile Android devices, image-based data collection, camera calibration, feature detection, Structure-from-Motion, Semi-Global Matching, dense image matching, point cloud generation.

1. INTRODUCTION

Mobile computing fascinates people of all age groups, as the success of smartphones has proven it for 10 years. Three main vendors of operating systems will dominate the market for mobile devices for the next five years: Android (Google, close to 70%), iOS (Apple, close to 20%), and Windows (Microsoft, close to 10%). Mobile devices are becoming more and more powerful – they allow for "always on – anywhere, anytime", creativity, and public understanding of complex issues. A prediction for the next five years expects 1 TB of storage capacity, fast x-core processors (x>4), communication speeds of up to 1Gbit/sec, and the integration of many sensors - besides compatible GNSS receivers for GPS, Glonass, Baidou and Galileo – like Inertial Navigation (IMU), advanced pedometers, air temperature, air pressure, and many more.

The performance of camera systems on mobile devices is outstanding – vendors have realized that it is not only the increase of resolution needed to win the Mpixels race – it is the size of the pixel pitch which should be >2µm to get enough photons catched, also during critical illumination conditions. Due to these developments, no one is questioning that nowadays mobile phones and tablets provide most of the photo shootings of closed and open (internet) photo collections. Therefore vision-based technologies integrated in mobile devices could be a next step to add value to the massive photo and video streams. Some apps are already integrating inertial and position informations, like horizontal and vertical angles, additional to the lat/lon or X,Y,Z coordinates (see fig. 1). This information can be used in photogrammetric bundle block adjustments as so-called "weak datum (exterior orientation)" as well as in Structure-from-Motion algorithms of computer vision, for which it is called "pose" parameters.

Figure 1. Mobile device screen to take pictures and additional pose information using AndroidSfM

In the following we demonstrate the development of an Android application for photogrammetric 3D reconstruction that works on smartphones and tablets likewise, seen from a point of view of software engineering. Android was chosen as it will dominate the usage of mobile operating systems over the next five years . The photos are taken with mobile devices, and can thereafter directly be calibrated using standard calibration algorithms of photogrammetry and computer vision, on that device. Due to still limited computing resources on mobile devices, a client-server handshake using Dropbox transfers the photos to the sever to run AndroidSfM for the pose estimation of all photos by Structure-from-Motion and, thereafter, uses the oriented bunch of photos for dense point cloud estimation by dense image matching algorithms. The result is transferred back to the mobile device for visualization and ad-hoc on-screen measurements. This client-server architecture is providing fast and reliable results for all situations having an Internet connection available, it can be programmed using Asynchronous Remote Procedure Calls (Asynchronous RPC) [4]. The strength and weaknesses for choosing this architecture is given below:
An analysis using classical quality items driving software development: *reliability*, *efficiency*, *security*, *maintainability*, *size* and *acceptability* will come to the following conclusions:

- *Reliability:* The complete 3D reconstruction process will not depend on the number of photos and resolution, neither the hardware configuration of the mobile device nor its Android version. Is also independent of available device memory. Parallel usage seems feasible, as the user can shoot more photos during processing the first shots.
- *Efficiency:* There are only intermediate interactions with remote resources (only to send photos and get the results). Furthermore, it benefits lower usage of data access, memory and disk space.
- *Security:* All data and information are automatically backed-up on a remote server.
- *Maintainability:* As two applications are running (one on the device and one on the server) this is more complex than running on the mobile device only.
- *Size:* The app running on the mobile device is a small size
- *Acceptibility:* It will depend on the expectations of the user based on time of processing, fail rates, and quality of results.

Moreover, it is possible to enumerate further benefits because the 3D reconstruction process is realized on a server:

- By processing on a server, better hardware configurations can be used: for example, more processors, higher RAM, GPU and CUDA. Then the application will have (indirectly) the performance of a desktop application.
- Results can be delivered faster and it is also possible to process a larger number of photos and higher resolutions. Of course, the processing time will also depend on hardware capabilities and the software running on the server.
- The reconstruction can be implemented using other software tools like VisualSfM or Bundler, in addition with dense reconstruction software like PMVS or SURE. Then the data is processed using the best currently available algorithms for pose estimation and dense image matching.
- The time of development will be shorter, and already a first app version would be as powerful as existing PC applications. When the desktop applications are upgraded, the app functionality will also benefit.

If a Stand Alone Application is completely programmed on the device, the following problems could appear:

- The dense 3D reconstruction using PMVS or SURE should be performed in any case onto a desktop. Then the results of the device would be preliminary.
- If we expect that the results of the 3D reconstruction on the device should have the same quality as those resulting from a desktop, then a new module has to be programmed. Additionally, code necessary to perform this task should be provided and authorized by the authors, which could be a problem.
- Even if we expect that future mobile devices are becoming more powerful and increase the chances to program a Stand Alone Application, desktop hardware will follow the same trend, and its programs will also benefit from these developments.

Most probably, as a conclusion, is to say, that the development of 3D reconstruction algorithms on mobile devices will always be behind desktop applications due to hardware differences.

The paper is structured as follows. After the introduction with the detailed rationale behind our chosen client-server architecture, a comprehensive analysis of related work in computer vision and photogrammetry is presented. Although both disciplines have been made for many years substantial developments independent from each other, there is a trend that both are now benefitting from each other by chaining Structure-from-Motion approaches with Dense Image

Matching algorithms, which leads to synergies of both disciplines. Chapter 3 outlines in detail the reasons to develop AndroidSfM as a client-server architecture. Based on the Waterfall model of software engineering the requirement definition is investigated very carefully. The demonstration of the AndroidSfM client-server app is given in chapter 4, discussing the setup of the photo shooting and the usage of two dense image matching algorithms: PMVS and SURE. Finally, our conclusions and an outlook on future work are given.

2. RELATED WORK

Automation for vision-based systems has a long tradition in computer vision and photogrammetry. As computer vision is still a young discipline compared with photogrammetry, is has developed fascinating methods and algorithms during the last 50 years. With the developments of analytical photogrammetry in the 1950s a new paradigm was created, leading to similar developments. The common ground are the collinearity equations, solved by Projective Geometry in computer vision and Perspective (Euclidean) Geometry in photogrammetry (see fig. 2)

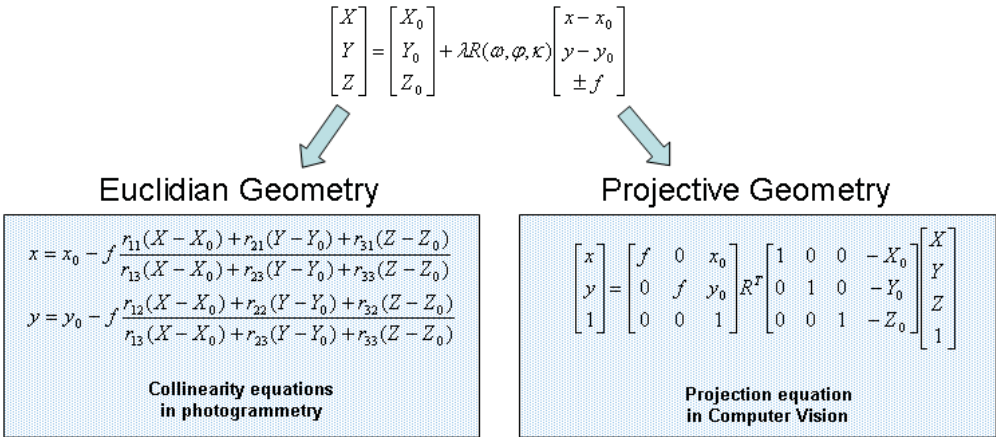


Figure 2. Computer Vision and its relation to the collinearity equations

The main advantage of using computer vision algorithms is related to usually linear analytical techniques and to use more robust algorithms (e.g. RANSAC) for feature tracking and matching. In case of self calibration, it calculates the three classical internal orientation parameters of the camera (x_0, y_0, f) and could also allow for a skew c - it does not require any control points. Point determination accuracy is not at the foreground in computer vision, as it drives photogrammetry, solving the linearized collinearity equations by bundle block adjustments, with a sufficient number of control points.

2.1 Computer Vision

The idea to implement the whole processing pipeline of data collection, feature detection and matching, pose estimation by Structure-from-Motion and finally using dense image matching algorithms on a mobile device is not new. Representative for these ongoing developments is [23]. This became feasible because of many fascinating developments delivering puzzle for puzzle, which are highlighted in the following. [5] delivered a comprehensive investigation for camera calibration for off-the-shelf TV cameras and lenses. At the end of 1990s, a first comprehensive collection of geometric methods of computer vision is given [6]. With the invention of Semi-global matching the door was opened to generate very dense point clouds [9]. A comparison of multi-view stereo reconstruction algorithms is given in [11]. SIFT and SURF operators for fast feature detection and matching are analyzed in [10]. Precision became more important – [12] offers a new and flexible technique for camera calibration. Programming the Patch-based Multi-view Stereo (PMVS) offered for the first time an Open Source code for dense point cloud generation [13], a study and new code with the CMP-MVS solution was complemented by [14]. Finally, the VisualSfM software put all main developments for pose estimation together and offers links to PMVS (CMP-MVS) and SURE [22].

2.2 Photogrammetry

With the brilliant work of D.C. Brown in the 1950s photogrammetry became an analytical discipline. He investigated also camera calibration [1] by introducing additional parameters to come to ultra-precise point determination. This work was deeply investigated in [2] and transferred to airborne photogrammetry in [3], to use additional orthogonal algebraic polynomials to overcome complex calibration issues. Furthermore, the spatial resection problem of photogrammetry - called pose estimation in computer vision – is presented in [7]. An extension of Semi-Global Matching (SGM) has been under development at the Institute for Photogrammetry, University of Stuttgart, since 2010 – the solution is a tSGM (tube SGM) and the software is called SURE[16, 20, 21]. A thorough test using several packages for dense image matching is performed in [18]. A complete new solution for camera self calibration, for airborne and close range photogrammetry, is presented by [19], leading to less correlations between the parameters of interior orientation. A new approach for Structure-from-Motion, based on graph theory, is presented in [17].

3. DEVELOPMENT OF ANDROIDSFM

When we started, we had the task in mind "to develop an Android Application for photogrammetric 3D reconstruction that works on smartphones and tablets". The desired output should be a high resolution, colored point cloud to be visualized on a mobile Android device, to check the point cloud's completeness and, just in case, to shoot more photos closing data gaps.

As software engineering is proposing three general process models, which are

- **Waterfall model.** This takes the fundamental process activities of specification, development, validation, and evolution and represents them as separate process phases such as requirements specification, software design, implementation, testing and so on.
- **Iterative and Incremental development.** This approach interweaves the activities of specification, development, and validation. The system is developed as a series of versions (increments), with each version adding functionality to the previous version.
- **Reuse-oriented software engineering.** This approach is based on the existence of a significant number of reusable components into a system rather than developing them from scratch.

Following the previous definitions, it seems suitable to use the Waterfall model, to be differentiated in requirements definition, system and software design, implementation and unit testing, integration and system testing, and operation and maintenance. Reuse-oriented software engineering could be used, if we reuse other systems. Iterative and Incremental development cannot be applied as we will not pass through different versions. For the time being, we generated only one version. During the product development process we tried to satisfy the six dimensions of software quality as well (chapter 1): *Reliability*, *Efficiency*, *Security*, *Maintainability*, *Size*, and *Acceptability*.

3.1 Requirements definition

For the definition of the requirement stage, the introductory task ought to be more detailed, as the final product should gather at least the next features (see table 1). As we can see, the requirements are too general, hence the developer (in its role as a consultant) should analyze the requirement and present to the customer the available alternatives with its advantages and disadvantages.

Table 1. Summary of requirements

Requirement	**Detail**
Operating System	Android, Version not defined
Devices	Smartphones and tablets. Hardware characteristics were not defined
Task	3D Reconstruction. The way to achieve this goal was not defined.

At this stage it is important to mention a crucial technical factor about Android that can have direct impact over the design, programming and the final product. This factor is that the Android operating system runs on devices with different hardware configurations. Manufacturers like Samsung, LG, Huawei, HTC (to name only a few) fabricate diverse kinds of devices with/without sensors, different camera resolutions, different processors and RAM and even with different Android versions. Developers have to take these aspects into consideration when they have to develop a product. This makes a big difference with respect to develop on iOS where hardware is defined beforehand.

Another important issue is to know the process to make photogrammetric 3D reconstruction, i.e. inputs, outputs, process step by step, memory usage, disk usage, speed, etc. Taking into account the overview showed in chapter 2 it is possible to create a table with some information and some questions (see table 2).

Table 2. Interpretation of requirements

Item	**Assumptions and possible questions**
Inputs	Pictures: resolution and size? Monochromatic or color? Number of pictures? Sensor information. Camera is mandatory. But what about GPS, Accelerometrs, magnet field sensor etc.?
Language	JAVA, C++ or both?
External Tools	Is it possible to use external tools to perform the programming? I.e. OpenCV, PCL Library, VTK Libraries, etc.? Will the results be visualized with an external sofware or to be programmed by ourselves?
Runtime Features	Processes to run in foreground or background? Can the user perform other tasks with the program or other programs meanwhile the software is processing? Intermediate results (like feature matching) have to be showed or the final point cloud and report only? What is the expected time to process and see the results?
Outputs	Language interfaces? Matching files, NVM file, point clouds or all? Where will the results be stzored? What about quality of the results ? Must they be similar or better than results obtained by using desktop software?

In the following we will resolve the questions of table 2.

Inputs: The performance of mobile devices will change according to the progress in innovation and developments. It could be useful to use all sensor information a mobile device has to offer, even if it is not used for the special application. Therefore it could be useful to collect at least: photos, colors and high resolution, angles and accuracy of accelerometers and compass, GPS information and provider. Furthermore, meta information is necessary as well, like project name, place descripption, etc.

Language: Due to device diversity, JAVA should be used and C++ whether is necessary to make special calculations or when speed is a important factor. For this work: Interface, sensors management, communications, Input and Output storage should be programmed using JAVA. Matching and Reconstruction processes could be performed using C++. However, the keys to select the Language are based also on the Runtime Features.

Runtime features: At this point it is important to give more information about how Android works and how hardware and software coexist.

- First of all it is necessary to mention that Android performs many tasks simltaneously. Even when one application is running in the foreground, there are other tasks running, i.e. Google Mail, Calendar, Alarm, Antivirus and those applications installed by the user. All of them consume memory and use disk space. Then, important resources like memory availability depend on how many applications are running.
- Second, applications and services have different priorities. For exmaple, even when an app runs in foreground it will be paused when a new phone call entries. As you can see phone calls have a higher priority.
- Third, according to the available resources (memory or disk space) the Andorid system can kill an application (without any warning).

These three aspects put some limitations on applications. If our application must run on different Android versions, different hardware configurations and also the use of less memory (to avoid being killed); then it is necessary to have an idea about the 3D reconstruction process tasks and its demands for the processor, memory and finally, time consumption.
In order to evaluate the usage of resources a first pilot application was developed to perform feature detection and matching – one of the firsts tasks necessary to realize the stage of feature tracking.
Three devices have been evaluated: The Samsung S3 I9300, Huawei ASCEND G510, Tablet Samsung GT p7510 (see table 3).

Table 3. Device features

Item	**Samsung S3 I9300**	**Huawei ASCEND G510**	**Tablet Samsung GT p7510**
CPU	Quad-core 1.4 GHz Cortex-A9	Dual-core 1.2 GHz Cortex-A5	Dual-core 1 GHz Cortex-A9
Storage	16/32/64GB	4GB	16/32/64GB
Memory	1 GB RAM	512 MB RAM	1 GB RAM
Camera	8 MP, 3264 x 2448 pixels, autofocus, LED flash	5 MP, 2592 x 1940 pixels, autofocus, LED flash	3.15 MP, 2048 x 1536 pixels, autofocus, LED flash
Sensors	Accelerometer, gyro, proximity, compass, barometer	Accelerometer, proximity, compass	Accelerometer, gyro, compass
GPS	Yes	Yes	Yes
Android	Android OS, v4.0.4 (Ice Cream Sandwich), 4.3 (Jelly Bean), planned upgrade to v4.4.2 (KitKat)	Android OS, v4.1 (Jelly Bean)	Android OS, v3.0 (Honeycomb), upgradable to v4.0 (Ice Cream Sandwich)

The objective of the evaluation was to measure the time consumption for each device performing detection and matching tasks over 4 images of 1536 x 2048 pixels (6 combinations) . To do this, a small program was developed using OpenCV to detect features and compute matchings. At the moment of this evaluation the SIFT detector and SURE was not available in the Android OpenCV version, however, SIFT and SURE were implemented patching OpenCV. The results of this first analysis are given in fig. 2.

As seen in fig. 3 tasks with zero value were not performed because the program crashed due to the lack of memory (SIFT). Each task consumes excessive time in comparison with a desktop application, e.g. Visual SFM used 7.8sec for each image using SIFT. The graph does not mention that in some cases not all photos could be processed because the application crashed due to the lack of memory.

Outputs: In order to generate an application according with international standards, English should be the language interface. Data results could be stored in the device or in a remote folder (Dropbox, Google Drive of FTP folder). Regarding to the detail and quality of results it will depend on the 3D Reconstruction process memory/disk consumption, however, at least should be showed a report or a point cloud.

After analysing carefully the requirement definitions, taking into account the previous paragraphs, it can be concluded that for the development of AndroidSfM two scenarios are feasible:

- A Stand-alone application that will run on the mobile device and will perform all the processes to generate results.
- Data collection system linked to a server application. The mobile device collects the data and sends it to a server, where it is processed and the results are sent back to the device to be visualized.

In chapter 1 we already outlined the strength and weaknesses of a client-server approach which is designed in the next section.

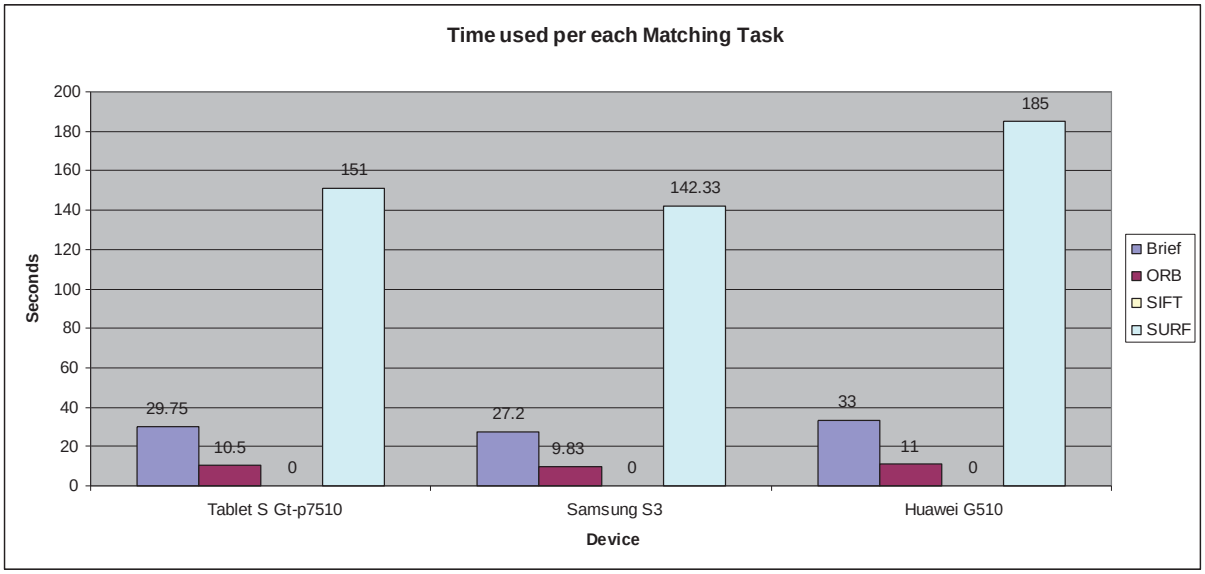


Figure 3. Feature matching evaluations

3.2 System and software design

The application for photogrammetric 3D reconstruction should contain the following functions: The Android App will cover all tasks associated to collect the photos, preprocess it and send it to the server for final processing. This App should also care for the visualization of point clouds. The Server App processes all data and deliver the results back to the mobile device (see fig. 3).

Fig. 4 shows the boundaries for each application (light blue boxes). The user can setup the configuration of the Server App, i.e. the path of a Dropbox shared folder, path for VisualSfM, etc on the other hand the user of the Android App organizes the photos and configures general options of the system including the reconstruction options (performed by the Server, i.e. software for coarse reconstruction, software for dense reconstruction, quality of results, send camera calibration parameters, setup the shared folder to excahnge the files (Dropbox, Google drive, a shared folder in a network, etc.). Furthermore, an order for 3D reconstructions is given, that triggers a service to send the task and photos and alo download the results when they are available.

A thorough analysis of a sequence diagram for the 3D reconstruction comes out with only one time dependency. As shown in fig. 5, the shared folder is a necessary resource to exchange information between both systems. It could be located in Dropbox or Google Drive, then the times of synchronization have to be included. The shared folder can also

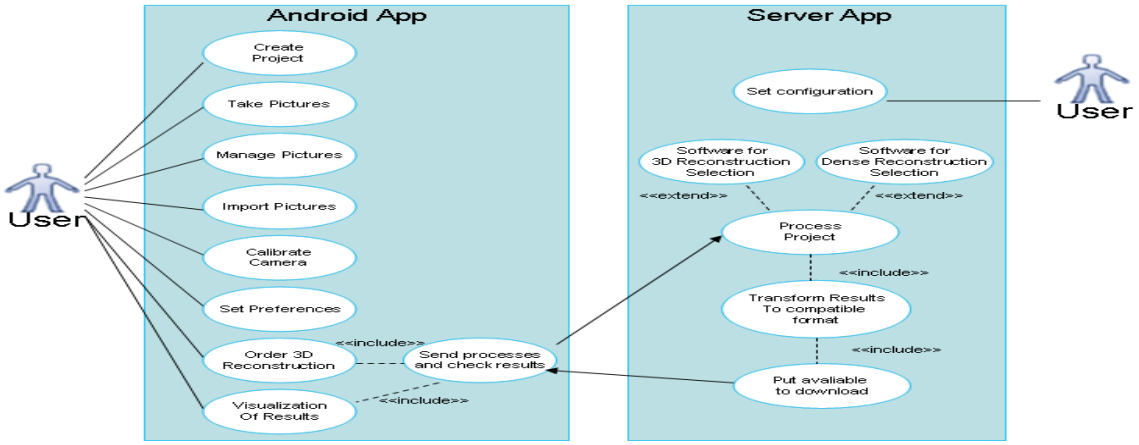


Figure 4. Use case diagram (using UML [8].

be located on the server in a local network, which would lower the times as the Android application would be connected directly to the Server application.

The time used to transfer and process information between each task is variable and it will depend on:

(1) The bandwidth between the device and the shared folder and between the shared folder and the Server. It will be the same in the opposite direction.
(2) Amount of pictures and size of them, which influences the uploading and processing time.
(3) Server hardware performance. The better the performance the less time is used to process the data.
(4) Quality of the results. A dense point cloud also implies a big file and more time for downloads.

The metadata (Class Project Management: id, name, description, datetime, mainpicture, lat, lon, #pictures, active, procedures) and Class Picture: id, project_id, name, lat, lon, pitch, rdl, bearing) are stored on the Android device using a class named shared preferences. The following information is stored: session credetials to access the shared folder, which means, user name and password to access Dropbox and/or Google Drive, the calibration and distortion matrix, current camera resolution, and general preferences oft he system. None of the metadata is stored on the Server, as all this

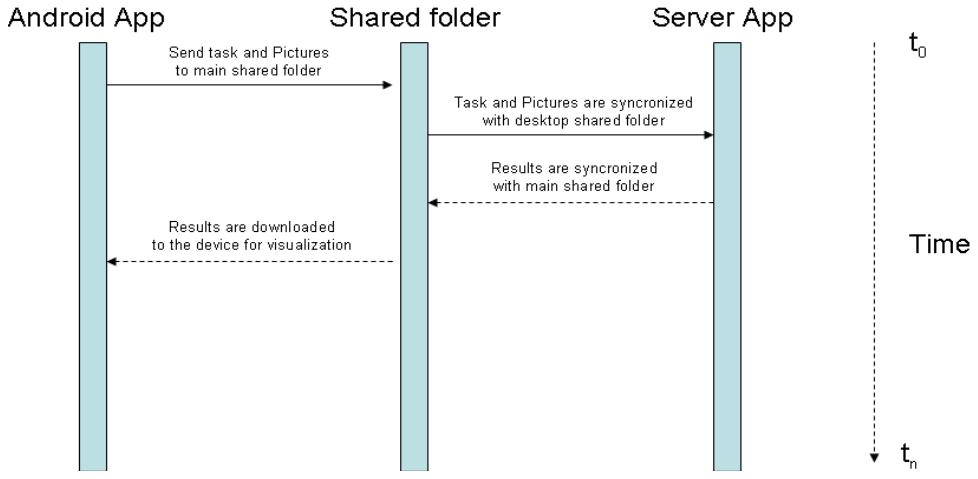


Figure 5. Sequence diagram for 3D reconstruction

information is handshaked by the Android device. Furthermore, additional tasks have been designed and implemented, such as: Implement prototypes, showing the software units woring, design screens and their functionalities for each stage oft he software (screen for photo shooting, photo organizer, project organizer, point cloud viewer).

3.3 Implementation and unit testing

After the system and software design the software has to be implemented and validated. As our project is small, the testing stage will be performed in only one stage that considers unit and integral testing. As in our system, the inputs are photos and user preferences, the testing activities are not difficult to implement. In fact, the structure of AndroidSfM was designed to avoid failures, for example, if projects are separated, then it is not possible to mix photos of different projects, and if photos of different resolutions are separated, then it is not possible to mix photos of different resolutions. Moreover, the application should not send information to the Server, if the Dropbox is not installed, the Server app is not configured, there is only one photo, and there is no internet connection.
On the other hand, the Server application was also designed to avoid failures, for example, if Dropbox is not installed, the Server application is not configured, there is only one photo, and there is no internet connection.

3.4 Operation and Maintenance

It was decided to test the AndroidSfM app during teaching courses of the curricula geodesy, geoinformatics and GEOENGINE at the University of Stuttgart.

4. RESULTS OF ANDROIDSFM

The resulting AndroidSfM App is presented in the following, on the client side by table 4, some screen shots and diagrams, and on the server side by table 5, and fig. 6-9. So far, it has been proven that the design and implementations are stable, robust and powerful. The client-server architecture runs smooth within normal internet environments. It seems that this setup is very comfortable and can be used to increase the number of photos incrementally, if the user is not yet satisfied with the completeness of the point clouds.

4.1 AndroidSfm on mobile device (client)

The main parameters for the implementation can be seen in table 4. The camera screen for taking photos is depicted in fig.1, for which a small app has been written to render the photo by additional information, provided by other smartphone sensors (GPS, accelerometer, etc.)

Table 4. AndroidSfM on device features

Item	**Description**
Programming language	JAVA
IDE	Eclipse Version: Kepler Service Release 1 Build id: 20130919-0819
Classes	31
Main class	ProjectsActivity.java
Layouts	24
Android compatibility	minSdkVersion 11 (Android 3.0.x HONEYCOMB) targetSdkVersion 18 (Android 4.3 JELLY_BEAN_MR2)
Dependencies	Universal Image Loader Dropbox API OpenCV
Devices tested	Samsung S3 I9300 Huawei ASCEND G510

	Tablet Samsung GT-p7510
Application requirements	CAMERA WRITE_EXTERNAL_STORAGE INTERNET ACCESS_NETWORK_STATE ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION READ_GSERVICES MAPS_RECEIVE
Codification style	Headers with general description and comments between lines
External applications	Kiwiviewer 1.0 to visualize point cloud information Any kind of Picture viewer to view pictures.
Data storage	SDCard and Dropbox folder for pictures SQLLite for Projects and Picture information Preferences are stored locally on the device using public interface SharedPreferences

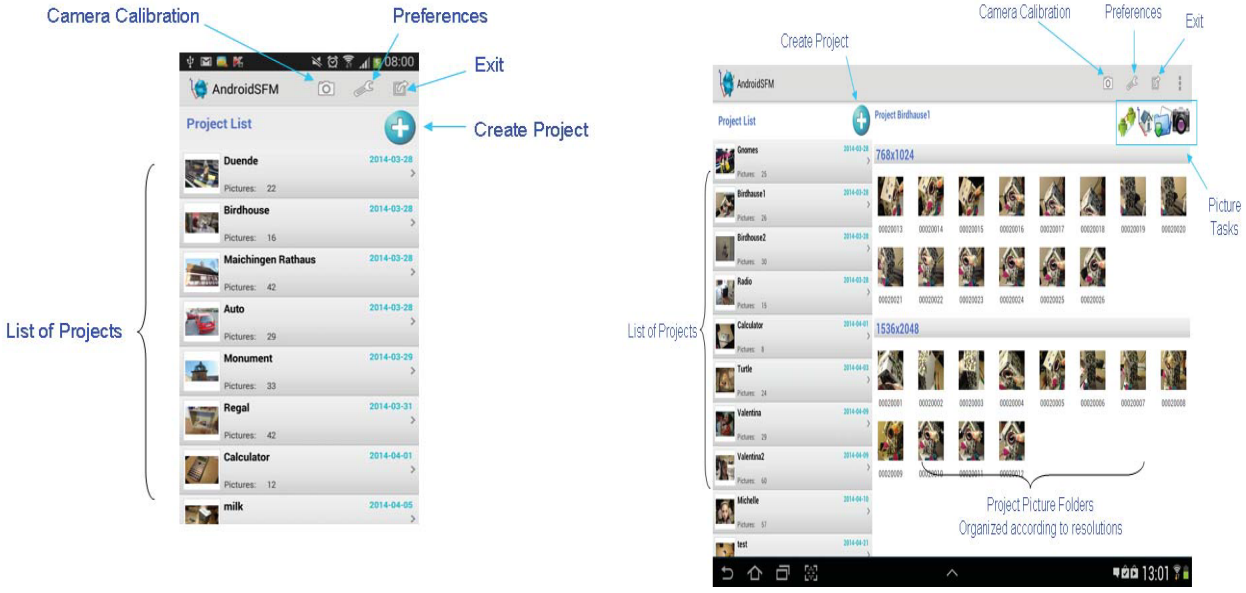
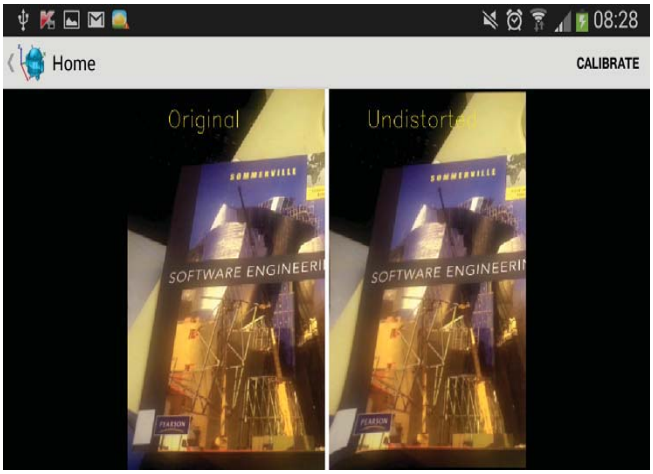


Figure 6. Main screens (a) on smartphones (b) on tablets



cameracalibration.xml*

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<Camera_Matrix type_id="opencv-matrix">
<rows>3</rows>
<cols>3</cols>
<dt>d</dt>
<data>528.5136377416345 0 319.5
  0 528.5136377416345 239.5
  0 0 1</data>
</Camera_Matrix>
<Distortion_Coefficients type_id="opencv-matrix">
<rows>5</rows>
<cols>1</cols>
<dt>d</dt>
<data>0.09437625535174626 0.2356656329311826 0 0 -0.9828585140497332</data>
</Distortion_Coefficients>
```

Figure 7. Camera calibration (a) calibration pattern (upper left), (b) calibration results (upper right), (c) parameters (lower center)

Figure 8. Point cloud of Maichingen house on smartphone (a) PMVS (b) SURE

4.2 AndroidSfM on Server

The AndroidSfM on Server has the following features (see table 5) - fig. 9 demonstrates the processing mode.

Table 5. AndroidSfM on Server features

Item	Description
Programming language	VB NET
IDE	Microsoft Visual Studio 2010 Version 10.0.40219.1 SP1Rel Microsoft .NET Framework Version 4.5.50938 SP1Rel
Classes	1
Main class	Main
Codification style	Headers with general description and comments between lines in the same way as device. //Launch camera Calibration Activity Intent intent = new Intent(this, calibration.CameraCalibrationActivity.class);
External applications	Kiwiviewer 1.0 to visualize point clouds Any kind of picture viewer to view pictures
Data storage	As application settings

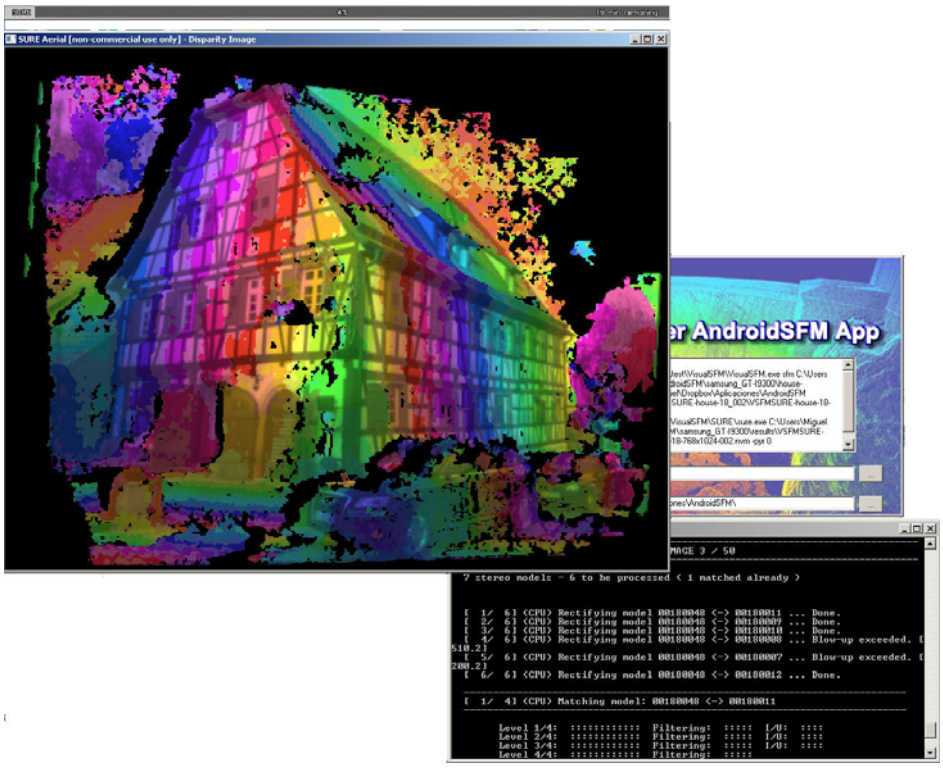


Figure 9. Screen of Server App processing a task with VisualSfM and SURE

5. CONCLUSIONS AND OUTLOOK

As was mentioned previously, software development is a process that follows a methodology, which is quite flexible and provides a set of useful tools that allow, as you progress in the design and analysis, to obtain clarity of the characteristics and scope of the product being generated.

Even if for this work two applications were developed (one on server and another on an Android device), it is important to note that this study does not say that an application for photogrammetric 3D reconstruction cannot be 100% developed onto a mobile device, or if it was created, will not serve all required information. It only states that, taking into account the analysis, that the best option currently available is to use a client-server model to take advantage of already developed software, the hardware potential of a computer, and at the same time does not demand too much from a device.

After a careful analysis to get better results fulfilling Software Quality requirements, the idea of using RPC (Remote Procedure Calls) applied in the field of mobile devices, we propose the term: **Mobile Remote Procedure Calls (MRPC)**. Applications using **Asynchronous MRPC** could link two worlds, mobile devices and computers to perform three-dimensional reconstruction processes, process surveying data or even to process information in advance while performing other tasks or traveling.

The product **AndroidSFM** is an example or first stage of an application to make 3D reconstructions (indirectly), therefore, it is possible to add new features such as:

- Add new reconstruction applications (Bundler, Agisoft Photoscan, etc)
- Enable new ways to exchange files (Google drive, local folders, etc..)
- Perform other tasks remotely on the point clouds using information collected by further sensors.

With regard to the Android operating system, using **Asynchronous MRPC**, we think that the system could multiply its usage for generating useful information also in the area of Computer Vision. It would be interesting, for example, to analyze the possible results of using arrays of synchronized devices (at different angles and overlaps as implemented in Ultracam) but in close range photogrammetry. Additionally, any further sensor information could be added to increase the collected information contents (4D, 5D, 6D?)

REFERENCES

[1] Brown, D.C., “Close-range camera calibration,” Photogrammetric Engineering, 37(8), 855-866 (1971).

[2] Faig W., “Calibration of close-range photogrammetry systems: Mathematical formulation. Photogrammetric Engineering and Remote Sensing, 41(12):1479–1486 (1975).

[3] Ebner, H., “Self-Calibrating block adjustment, Bildmessung und Luftbildwesen (BuL), 44(4), 128-139 (1976).

[4] Ananda A. Tay B., Koh E. , “A survey of asynchronous remote procedure calls,” ACM Digital Library Operating Systems Review, ACM New York, Newsletter 26(2), 92-109 (1992).

[5] Tsai R.Y., “A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and lenses,”. IEEE Journal of Robotics and Automation, 221-244 (1992).

[6] Hartley, R., Zisserman, A., [Multiple View Geometry in Computer Vision], Cambridge University Press, Cambridge, UK, 655p (2000).

[7] Grussenmeyer P. & Al Khalil O.,”Solutions for exterior orientation in photogrammetry, a review. The Photogrammetric Record, 17, 615–634 (2002).

[8] Booch G., Rumbaugh J., Jacobson I., [Unified Modeling Language User Guide], 2nd Edition. Addison-Wesley. 496p (2005).

[9] Hirschmüller, H. [2005]. Accurate and efficient stereo processing by Semi Global Matching and Mutual Information. IEEE Conference for Computer Vision and Pattern Recognition (CVPR), 2, 807-814 (2005).

[10] Bay, H., “SURF: Speeded Up Robust Features”. Computer Vision – ECCV 2006. Lecture Notes in Computer Science Vol. 3951, Springer, Berlin, 404-417 (2006).
[11] Seitz S., Curless B., Diebel J., Scharstein D., Szeliski R., “A comparison and evaluation of Multi-View stereo reconstruction algorithms. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1, 519-528 (2006).
[12] Zhengyou Z., “A flexible new technique for camera calibration”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22, 1330-1334 (2009).
[13] Furukawa Y., Ponce, J., “Accurate, dense, and robust multiview stereopsis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(8), 331-343 (2010).
[14] Jancosek, M., Pajdla T. ,”Multi-view reconstruction preserving weakly-supported surfaces,” The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3121-3128 (2011).
[15] Lélis D., Milan D., Saragih J., “Mastering OpenCV with Practical Computer Vision Projects”. Published by Packt Publishing Ltd. ISBN 978-1-84951-782-9 (2012).
[16] Rothermel, M., Wenzel, K., Fritsch, D., Haala, N., “SURE: Photogrammetric surface reconstruction from imagery“, Proceedings Low-cost 3D (LC3D), Berlin (2012).
[17] Abdel-Wahab, M., Wenzel, ., Fritsch, D., “Automated and accurate orientation of large unordered datasets for close range Cultural Heritage data recording“, Photogrammetrie – Fernerkundung – Geoinformation (PFG), 6, 679-690 (2012).
[18] Ahmadabadian, A.H., Robson, S., Boehm, J., Shortis, M., Wenzel, K., Fritsch, D., “A comparison of dense matching algorithms for scaled surface reconstruction using stereo camera rigs”, ISPRS Journal of Photogrammetry and Remote Sensing 78, 157-167 (2013)
[19] Tang, R. Fritsch, D., “Correlation Analysis of Camera Self-Calibration in Close Range Photogrammetry”, The Photogrammetric Record, Special Issue: DIAMOND JUBILEE ISSUE, Vol. 28, Issue 141, 86-95 (2013).
[20] Wenzel K., Rothemel M., Fritsch D., “Image acquisition and model selection for Multi-View stereo. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XL-5/W1, (2013)
[21] Wenzel K., Rothemel M., Haala N., Fritsch D., “SURE – The ifp software for dense image matching, “ Photogrammetric Week '13, Ed. D. Fritsch, Wichmann, Berlin/Offenbach, 59-70 (2013).
[22] Wu, C., “Towards linear-time incremental Structure from Motion”, 3DV (2013).
[23] Kolev, K., Tanskanen, P., Speciale, P., Pollefeys, M., “Turning mobile phones into 3D scanners”, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3946-3953 (2014).