Miguel Angel Syll

Development of an Android Application for Photogrammetric 3D Reconstruction

Duration of the Thesis: 6 months
Completion: July, 2014
Supervisor & Examiner: Prof. Dr. –Ing. Dieter Fritsch

1. Abstract

This thesis attempts to describe the necessary steps to program an application to take pictures, organize and process them in order to generate and visualize a three-dimensional colored point cloud. The steps to create the application are explained, using the argumentative logic to every decision and fulfill the stated objective in the best possible way, balancing the available technology and the quality of results.

The general requirement "Development of an Android Application for Photogrammetric 3D Reconstruction" was analyzed from the point of view of Software Engineering methodology trying to fulfill the requirements related to Software Quality (Somerville, 2011): Reliability, Efficiency, Security, Maintainability and (adequate) Size. Moreover, Android systems and their features are outlined, and the fundamental concepts of Photogrammetry, Computer Vision and Structure from Motion (SFM) are explained.

After the analysis stage a new product named AndroidSFM is designed and programmed, that is composed by an Android application and a Server application.

2. Overview of Software Engineering Definitions and Software Quality

2.1. Software Engineering

Somerville (2011) defines software engineering "as a engineering discipline that is concerned with all aspects of software production". By this short phrase is hard to see or understand the real scopes of software engineering, therefore another ones is cited.

According to an old IBM Systems Journal publication (Mills, 1980) Software engineering may be defined as "the systematic design and development of software products and the management of the software process. Software engineering has as one of its primary objectives the production of programs that meet specifications, and are demonstrably accurate, produced on time, and within budget".
This is a nice definition because it takes into consideration different terms like: *Systematic, design, development, management, specifications, accurate, time and budget*. All terms indicate that developing a software is necessary not only to bring together a computer, a programming language and a programmer, but also a methodology to fulfill some known requirements taking into account quality, time and money. Of course, there are other definitions, shorter or more extended, but the important thing is that to develop a piece of software implies more issues than only to program.

In conjunction with the term software engineering, more linked with programming, appears the term Software development process, also known as a Software Development Life-Cycle (SDLC), that is a structure imposed on the development of a software product.

For Somerville (2011) the systematic approach that is used in software engineering is sometimes called a software process. A software process is a sequence of activities that leads to the production of a software product. There are four fundamental activities that are common to all software processes. These activities are:

1. **Software specification**, where customers and engineers define the software that is to be produced and the constraints on its operation.
2. **Software development**, where the software is designed and programmed.
3. **Software validation**, where the software is checked to ensure that it is what the customer requires.
4. **Software evolution**, where the software is modified to reflect changing customer and market requirements.

Different types of systems need different development processes. For example, real-time software in an aircraft has to be completely specified before the development begins. In e-commerce systems, the specification and the program are usually developed together. Consequently, these generic activities may be organized in different ways and described at different levels of detail depending on the type of software being developed.

Several general models exist to streamline the development process. Each one has its pros and cons, and it is up to the development team to adopt the most appropriate one for the project. Sometimes a combination of the models may be more suitable. Here is a list with the generic processes mostly used:

- Waterfall model
- Iterative and incremental development.
- Reuse-oriented software engineering

### 2.2. Software Quality

It seems to be clear that for software development companies and for customers a "software" is a product, therefore, it is a candidate to be qualified or evaluated in some way. This is in agreement with Pressman (2005) who states: "Software functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. That attribute can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product".

Historically, the structure, classification and terminology of attributes and metrics applicable to software quality management have been derived or extracted from the ISO 9126-3 and the subsequent ISO 25000:2005 quality model. Based on these models, the Consortium for IT Software Quality (CISQ) has defined five major desirable structural characteristics needed for a piece of software to provide business value: **Reliability, Efficiency, Security, Maintainability and (adequate) Size**.
The dependence tree between software quality characteristics and their measurable attributes is represented in the next diagram, where each of the 5 characteristics that matter for the user (left and right) or owner of the business system depends on measurable attributes (center):

![Software quality example diagram.](image)

Figure 2.1: Software quality example diagram.

In order to add more references, Somerville (2011) states in the next table the Essential Attributes of good software:

<table>
<thead>
<tr>
<th>Product characteristics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability</td>
<td>Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.</td>
</tr>
<tr>
<td>Dependability and security</td>
<td>Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.</td>
</tr>
<tr>
<td>Acceptability</td>
<td>Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.</td>
</tr>
</tbody>
</table>

Table 2.1: Essential Attributes of good software according to Somerville (2011)

More information about these topics can be obtained from books of Software Engineering. However, as a summary, the most important things to keep in mind are that Software engineering
- Is an engineering discipline
- Generates a product or products (software/s).

Also Software
- Has a life cycle and it is subject to a methodology to be generated/developed.
- Has a process that also can be improved
- Can be qualified, evaluated or measured using different parameters.
3. **Development Process of the Application**

At the beginning of this chapter it is important to point out that each one of the next stages will deliver "results" in the way as "special features" of the final product.

It is the intension to quote "results" quite often, because not only the final product will be the result but also each one of the conclusions (output of this chapter) used to generate it.

Then, in order to generate a product that fulfills some software quality requirements, it seems suitable to apply some software methodology.

As was enunciated on the overview about Software Engineering there are three general process models

- Waterfall model
- Iterative and incremental development.
- Reuse-oriented software engineering

Taking into account the definitions of each one the Waterfall model was chosen.

![Waterfall model and its stages](image)

Figure 3.1: Waterfall model and its stages

In this work we will pass through the different stages of the Waterfall model in order to obtain a product, taking also into account to try satisfying the six dimensions of software quality, which are: **Reliability**, **Efficiency**, **Security**, **Maintainability**, **Size** and finally **Acceptability**.

### 3.1. Requirement Analysis and Proposed Software

**Requirement:**

*To develop an Android Application for Photogrammetric 3D Reconstruction that works on smartphones and tablets.*

In this case no more requirements were specified, then, the final product should gather at least the next features:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operative System</td>
<td>Android. Version were not defined</td>
</tr>
<tr>
<td>Devices</td>
<td>Smartphones and tablets. Hardware characteristics were not defined.</td>
</tr>
<tr>
<td>Task</td>
<td>3D Reconstruction. The way to achieve this goal was not defined.</td>
</tr>
</tbody>
</table>

Table 3.1: Summary of requirements.
As we can see, the requirements are too general, hence the developer (in its role as consultant) have to analyze the requirement and present to the customer the available alternatives with its advantages and disadvantages.

At this stage it is important to mention a crucial technical factor about Android, that can have direct influence over design, programming and final product. This factor is that the Android operative system runs on devices with different hardware configurations.

Manufacturers like Samsung, LG, Huawei, HTC (to name only few) fabric diverse kind 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 states a big difference with respect to develop on iOS where any hardware is defined beforehand.

Another important issue is to know the process to make 3D Reconstructions, i.e. inputs, outputs, process step by step, memory usage, disk usage, speed, etc. Taking into account some concepts about Photogrammetry and Computer Vision it is possible to create a table with some information and some questions.

<table>
<thead>
<tr>
<th>Item</th>
<th>Assumptions and possible questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inputs</td>
<td>Pictures: resolution and size? Monochromatic or color? Amount of pictures? Sensor Information. Camera is mandatory. But what about GPS, Accelerometers, magnetic field sensor?</td>
</tr>
<tr>
<td>Language</td>
<td>JAVA, C++ or both?</td>
</tr>
<tr>
<td>External Tools</td>
<td>Is it possible to use external tools to perform the programming? i.e. OpenCV, PCL Library, VTK Libraries, etc? Will be results visualized with a external software or is necessary to program the viewer as well?</td>
</tr>
<tr>
<td>Runtime Features</td>
<td>Processes have to run in foreground or background? Can the user perform other tasks with the program or other program meanwhile the software is processing? Intermediate results (like matching results) have to showed or only final result (point cloud and report)? What is the expected time to process and see the results?</td>
</tr>
<tr>
<td>Outputs</td>
<td>Language Interfaces? Matching files, NVM file, Point Cloud or all? Where will be stored the results? What about the qualities of the results? Must they be similar or better than results obtained using desktop software?</td>
</tr>
</tbody>
</table>

Table 3.2: Interpretation of requirements.

As you can see, from the general requirements arise a lot of questions. This is not strange, generally customers have a general idea about the product they want, and it is the developer who has to suggest the way to reach the objective. Then, using the available information, we will try to solve the showed questions.

3.1.2. Inputs:
Due to an amazing development of new and existing devices, each time more sensors delivering more information, even when the information is not used by the application. Maybe in the future or for other applications, several sensor information could be useful.

With current devices it could be useful to collect, at least:

- Pictures, Colors and Resolution
- Angles and accuracy of accelerometers and compasses
- GPS information and provider.

Other inputs are those necessary to organize the information, like the project name, place description, etc.
The amount of information stored will depend mainly on the amount of pictures and its resolutions, which in turn depends on the kind of project to be photographed.

3.1.2. Language:
Due to the device diversity JAVA should be used and C++, depending on the necessity to make special calculations or when speed is an important factor. For this work we focus on programming the interface, sensor management, communications, Input and Output storage in JAVA - the matching and reconstruction processes is performed using C++. However, the keys to select the programming languages are also based on the Runtime Features.

3.1.3. Runtime Features
At this point 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 simultaneously. Even when one application is running on foreground, there are other tasks running i.e. Google Mail, Calendar, Alarm, Antivirus and other applications installed by the user. All of them consume memory and use disk space. Then, important resources like memory availability depends on how many applications are running.
- Second, applications and services have different priorities. For example, even when an application 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 Android 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 few memory (to avoid being killed); then it is necessary to have an idea about the 3D reconstruction process tasks and its demands of processor, memory and finally time consumption.

3.1.4. Outputs
- In order to generate an Application according with the 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, a report or a point cloud should be showed.

3.2. Evaluation test of an Application for feature detection and matching running 100% on the device

In order to evaluate the resource consumption it was created a little application to perform the feature detection and matching, one of the first tasks necessary to realize the stage of Feature Tracking.

Three devices where evaluated and 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 it was developed a little program using OpenCV to detect features and compute matchings. At the moment of this evaluation, feature detectors SIFT and SURE were not available in the Android OpenCV version, however, SIFT and SURE were implemented patching OpenCV.
As is possible to appreciate in the graph:

- Tasks with zero value where not performed because the program crashed due to lack of memory (SIFT).
- Each task consumes excessive time in comparison with a computer. VisualSFM used 7.8 seconds for each image using SIFT.
- The graph does not mention that in some cases not all pictures were processed and also that in all devices the application crashed due to lack of memory.

This short analysis point out, that the reconstruction process could not be carried out and even when it were possible, it would use too much time. Generally speaking, reconstruction projects require high resolution images and many pictures.

### 3.3. Proposed Software

Taking into account the previous sections, it can be concluded for the development of the application that two scenarios are feasible:

- A stand-alone application, that will run on the device and will perform all the processes to generate results.
- A data collection system linked to a Server application. The device collects the information and sends the information to a Server where the information is processed and the results are sent back to the device to be visualized. This alternative can be programmed using Asynchronous Remote Procedure Calls (Asynchronous RPC) (Ananda, 1992).

<table>
<thead>
<tr>
<th>Quality Item</th>
<th>Stand Alone</th>
<th>Client Server</th>
</tr>
</thead>
</table>
| Reliability        | Depend only on the device, however, to complete the process of reconstruction will depend also on:  
|                    |   - Number of images and its resolution.  
|                    |   - Hardware configuration of the device and Android version.  
|                    |   - Available Memory  
|                    |   - If the level of complexity of matching algorithms is higher  
| Software reliability | is the probability that software will work properly in a specified environment and for a given amount of time | Depend on the device, server and the connection with the server.  
|                    | In this case, to complete the process of reconstruction will not depend on:  
|                    |   - Number of images and resolution.  
|                    |   - Hardware configuration of the device or Android version.  
|                    |   - Available Memory  
|                    |   - Additionally other applications can be used. The user, for
then more probabilities to fail. | example, could take meanwhile more pictures.
---|---
**Efficiency** |  |  
- Appropriate interactions with expensive and/or remote resources  
- Data access performance and data management  
- Memory, network and disk space management |  
- Low interaction with remote resources.  
- Higher usage of data access, memory and disk space.  
- Intermediate interaction with remote resources (only to send pictures and get the results).  
- Lower usage of data access, memory and disk space.

**Security** |  |  
Information is stored in the device. Information is on risk if the device fail or is lost. | Information is automatically backed up on remote Server.

**Maintainability** |  |  
It should be easier to maintain only one application. | Due to there are two applications (one on device and the other on server) the maintainability should be more complex.

**Size** | Big Size | Small Size

**Acceptability** |  |  
Similar for both two alternatives. It will depend on the expectations of the users based on: time of processing, rate fails, quality or the results.

Table 3.3: Qualitative comparison.

According to Table 3.3 the Client-Server application presents more advantages. It is also possible to enumerate other benefits, because the reconstruction process is realized on a Server, for example:

- Processing on a server, better hardware configurations can be used, for example: more processors, higher RAM, GPU and CUDA. Then the application would have (indirectly) the performance of a desktop application.
- Results can be delivered faster and it is also possible to process a higher amount of pictures with higher resolutions. Of course, the speed will also depend on hardware capabilities and the software used 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 information would be processed using the best current algorithms.
- The time of development will be shorter and the results are benefitting from existing PC applications. When the PC reconstruction applications are upgraded, then the functionality might benefit as well.

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

- Dense reconstruction using PMVS or SURE should be performed on a PC. Then, results would be preliminary.
• Just in case, that the expectations of the results are at the same level or similar to the results obtained with PMVS or SURE, a new module can be programmed. Additionally, the code necessary to perform this task should be provided and authorized by the authors. Is this possible? Maybe yes, but it seems, that each time there are more things necessary, and we are working with devices with limitations.

• Even when in future devices will offer increasingly chances to program a Stand Alone Application, PC hardware will follow the same trends and also desktop applications could be better. Based on these conclusions, there is obviously a high probability, that the development of applications for 3D reconstruction on devices are always behind of PC applications because of the hardware differences.

Maybe, the most important advantages of the second alternative are related to the flexibility to run on different hardware configurations and that it will be possible to perform other tasks on the device simultaneously. These aspects have an important weight for users, who use different kind of devices and different kind of applications.

As a conclusion, the second alternative was developed. The following activities are performed:

• System and Software Design
  o Functional analysis
  o UML - Use Case diagram
  o UML Sequence diagram
  o UML Class Diagram

• Programming and Testing (Software Validation)

4. Products and Results

In this chapter some screen shots and diagrams of the software named AndroidSFM are presented, as the result of this master thesis.

4.1. AndroidSFM on Device
Figure 4.1: Main Screen Design on Smartphones

- Create Project
- Camera Calibration
- Preferences
- Exit

List of Projects

Picture Tasks

Project Picture Folders
Organized according to resolutions

Figure 4.2: Main Screen on Tablets (Fragments)

Figure 4.3: Camera screen to take pictures and additional information.
Figure 4.4: Point cloud of House in Maichingen result example (in smartphone).

Figure 4.5: Screen of Matching Utility.
4.2. AndroidSFM on Server

Figure 4.6: Screen of Server App processing a task with SURE.

5. Discussion and Conclusions

As was mentioned in earlier paragraphs, software development is a process that follows a methodology, which is quite flexible and provides a set of useful tools that allows obtaining clarity of the characteristics and scope of the product being generated.

Thanks to this methodology, general requirements can be transformed into each step in more concrete things, which allows meeting the initial objectives of the client in a better way, even extending them.

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

After a careful analysis in order to get better results fulfilling the Software Quality requirements, the idea of using RPC (Remote Procedure Call) was born, applied in the field of mobile devices – it is named here 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.
Today’s mobile devices are equipped with special sensors (GPS, cameras, accelerometers, magnetic compass, gyroscope, pressure sensor, temperature sensor, proximity sensor, humidity sensor, gravity sensor, light sensor or sensors extras that can be assembled), which - when used - would collect field information, environmental information, seismic data, etc.; All data is georeferenced.

The product AndroidSFM is only 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 the sensors.

With regard to the Android operating system, using Asynchronous MRPC, it is the author’s vision, that the system could multiply its use for generating more useful information 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 also in Close range photogrammetry. Additionally, sensor information could be added to increase the collected information (4D?)

6. References