



Case number:

UNIVERSITY OF MOHAMED BOUDIAF - M'SILA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

COMPUTER SIENCE DEPARTEMENT

Research Paper

**Presented in partial fulfilment of the Requirements for the Degree
of Master**

Field: Mathematics and computer science

Branch: Computer science

Specialty: ICT

By: HEMMAK MALEK

TOPIC

**Design and Implementation of an Android
Application for Creating Panoramic Images**

Members jury :

Mr. DEBBI ImadAddine

Dr. BOUGUERRA Abdelbaki

Mr. KAMEL Mohamed

University of M'sila President.

University of M'sila Supervisor.

University of M'sila Examiner.

Class: 2015/2016

Table of contents :

Figures List	1
Equations List	2
Tables List	3
GENERAL INTRODUCTION	4
CHAPTER1: OVERVIEW OF DOMAINE AND STATE OF ART	7
1. Introduction	8
2. Why Smartphone	8
3. Android	8
4. What is Image Stitching	10
5. The difficulties of Image stitching	10
6. What is Panorama	11
7. Panoramic cameras	11
7.1. Using a physical support	11
7.2. Professional panoramic cameras	11
7.3. Multi-lenses camera	12
8. Image stitching and feature extraction applications	12
8.1. Video stabilization	12
8.2. Face recognition	13
8.3. Gesture interface	13
8.4. Virtual Reality	14
8.5. Maps Applications	14
8.6. Robotics	14
8.7. Medical	14
9. Conclusion	15
CHAPTER2: IMAGE STITCHING	16
1. Introduction	17
2. Image features	17
3. Introduction to Image stitching	17
3.1. Generating Panorama Photos (Hewlett-Packard Labs)	17

3.2. Stitching of X-ray Images (UPPSALA university).....	18
4. Main components of image stitching	18
4.1. Calibration	19
4.2. Registration (Alignment)	19
4.3. Blending	20
5. Image stitching techniques	20
5.1. Direct techniques	20
5.2. Feature-based techniques	20
5.2.1. Scale-Invariant Feature Transform (SIFT)	21
5.2.2. Speeded Up Robust Features (SURF)	24
6. Comparison	27
6.1. Performance comparison	28
6.2. Detector performance comparison	28
7. Conclusion	28
CHAPTER3: DESIGN AND IMPLEMENTATION	29
1. Introduction	30
2. Tools and software used	30
2.1. Android SDK	30
2.2. Android NDK	30
2.3. Android Studio	30
2.3.1. Gradle	31
2.4. OpenCV	31
2.4.1. Core	31
2.4.2. Imageproc	32
2.4.3. Video.....	32
2.4.4. Features2d	32
2.4.5. Objdetect	32
2.4.6. Highgui	32
2.5. ArgoUML	32
3. Generating panorama process	33

4. Panorama image processing	34
4.1. Image acquisition	35
4.2. Feature extraction	35
4.3. Feature matching	35
4.4. Image alignment	35
4.5. Blending	35
4.6. Cropping	35
5. UML Classes Diagram	36
5.1. Main Activity Class	37
5.2. Preview Class	37
5.3. Prepare Static Class	37
6. Application Interface.....	38
6.1. Main interface	38
6.2. Configuration interface	39
7. Functionalities.....	40
7.1. Main button	40
7.2. Android device orientation angle guiding lines	40
7.3. Camera Autofocus state	40
7.4. Camera Autofocus lock	40
7.5. Taken panorama pictures	41
7.6. Notifies user on panorama has created	41
7.7. Tilt Guiding	41
7.8. Matching verification	42
7.9. Gallery	42
7.10.Parameters	42
7.10.1. Input images resolution	42
7.10.2. Panorama quality	42
7.10.3. Autofocus lock feature	43
7.10.4. Ability to re-compute a previous panorama with different configurations	43
8. Dealing with Device Sensors	43

8.1. Accelerometer	43
8.2. Magnetometer	43
8.3. Low Pass Filter	44
9. Implementation.....	45
9.1. Dealing with Sensors	45
9.2. Guiding Angle	46
9.3. Dealing with android AsyncTask	47
10.Power points.....	47
11.Conclusion	48
CHAPTER4: TESTS AND RESULTS	49
1. Device used (Meizu M2 Note)	50
2. Device performance	50
3. Resolution and Time test	51
4. Resolution Error Rate	51
5. Panorama Quality test	52
6. Autofocus lock test.....	53
7. Manual picture taking vs. automatic picture taking	54
8. Resolution comparison	54
9. Panorama camera applications comparison	55
10.Device angle effects	56
11.Some results	57
12.Future goals	57
13.GENERAL CONCLUSION	59
Bibliography	62

FIGURES, EQUATIONS AND TABLES

Figures List :

Figure 1 Statistics about the most used camera type in taking pictures, According to [1]	8
Figure 2 Android Operating System Architecture.....	9
Figure 3 Statistics about Mobile Operating System Market Share, According to [3].....	9
Figure 4 A Set of Pictures that will be combined using Image Stitching.....	10
Figure 5 The set of pictures in the previous figure after image stitching. Extracted from [4]...	10
Figure 6 Bushman Panoramic Gobi head, extracted from [6].....	11
Figure 7 PanoScan MARK III camera.	11
Figure 8 The Facebook Surround 360 Camera. Extracted from [7].....	12
Figure 9 Represents before and After Applying video stabilization.	12
Figure 10 An Example of Face Recognition Application, a kid trying to access his dad's phone. Extracted from [8].	13
Figure 11 A man playing a game on Xbox Kinect, using his hand geastures. Extracted from [9].	13
Figure 12 Example of two Satellite images begin stitched together. Extracted from [10].	14
Figure 13 Panorama Image Result of Arch national park in Utah by Hewlett-Packard Labs Paper [11].	17
Figure 14 Represents Image Stitching Application in X-ray, of stitching three images, By [12].	18
Figure 15 Image stitching main steps: Calibration, Registration, Blending.....	18
Figure 16 Basic pinhole camera model. Extracted from [13].	19
Figure 17 the image before blending. Extracted from [13].	20
Figure 18 panorama mosaic after applying a blending method called multi-band blending. Extracted from [13].	20
Figure 20 Laplacian of a Gaussian. Extracted from [15].	22
Figure 19 Represent the comparison between a point and it's 8 neighbors in the same scale and it's 9 neighbors of up and down scale, to detect the maxima or minima, I_1 is the downscale and I_3 is the upscale	22
Figure 21 Represents SIFT orientation vectors, image extracted from [16].	23
Figure 22 Generating Image descriptors.	23
Figure 23 the integral images (the sum of all pixels that are used later in Hessian-matrix) before and after applying Hessian detector's filter (square-shaped).....	24
Figure 24 Grayscale image in the left and the image after applying box filter. Image extracted from [18].....	25
Figure 25 Determining Orientation in SURF.	25
Figure 26 Example of image and SURF features detected. Extracted from [19].....	26
Figure 27 Matching two images. Extracted from [12].	26
Figure 28 The Test Image of [17], size of the image is 800*640 pixel.....	27

Figure 29 A simple code of Gradle configurations, that it shows a simple gradle configuration file that is responsible to define build configurations.	31
Figure 30 Generating Panorama Process, this diagram represents between 4 classes of our application and the user.	33
Figure 31 An state-transition diagram of our panorama image processing.	34
Figure 32 Our main UML classes diagram.	36
Figure 33 Our application's main interface.	38
Figure 34 Interface of configurations.	39
Figure 35 Main Button.	40
Figure 36 Focus states, Not Ready: when Camera Focus is still processing, Ready: when Camera Focus processing is done.	40
Figure 37 Represents Number of taken pictures, that will be stitched to make the panorama picture.	41
Figure 38 Represents Tilt Guiding Arrows.	41
Figure 39 Represents where the phone should be tilted to.	41
Figure 40 Two icons to inform the user about the matching result.	42
Figure 41 Gallery Button. The user can touch it to access panoramic images.	42
Figure 42 Shows device orientation angles and tilting angle (Azimuth).	44
Figure 43 A noisy random signal (blue) and its low pass filter (shown in red).	44
Figure 44 Represent the user phone in our tests, Meizu M2 Note.	50
Figure 45 A Graph of the relation between Pictures Resolution and Time to make a panorama.	51
Figure 46 A Graph of Picture Resolution Error Rate (High Quality Panorama Processing).	51
Figure 47 Represents the detected errors.	52
Figure 48 How the is the correct image stitching, Extracted from the most better image resolution 13MP and Higher Processing.	52
Figure 49 Graph represents work resolution (of the stitching process) and the time to stitch images.	53
Figure 50 A picture of the computer science department, before we added autofocus lock feature. Notice on the left side of the picture it's a bit blurry.	54
Figure 51 effect of camera angle(tilt) on image stitching. The red rectangle represents the safe area of the picture, and the circle shows how much the camera was curved to the front.	56
Figure 52 More guided panorama image, notice the comers are straight which clarifies that the device was in a good orientation.	56
Figure 53 A result of the application, panorama image of science of computer faculty. Taken may 2016.	57
Figure 54 Another result, taken in the university. Taken May 2016.	57

Equations List :

Equation 1 Applying Guassian filter (G) to an Image (I), with a scale value (σ).	21
Equation 2 Difference Of Gaussian of two images, one with scale of k-times the other.....	21
Equation 3 location of the extremum point, achieved by the laplacian.....	22
Equation 4 Gradient magnitude.....	23
Equation 5 Orientation equation.....	23
Equation 6 Integral image.	24
Equation 7 Hessian-matrix that is combining the four integral images: xx, xy, xy, yy.	25
Equation 8 Determinant of Hessian-matrix.	25
Equation 9 Mathematical low pass filter euqation, $f(x)$ is the recived frequency, α is the filter size that $0 \leq \alpha \leq 1$	44

Tables List :

Table 1 An overview comparison between SIFT and SURF. Extracted from [20].....	27
Table 2 Represents comparision of processing time between SIFT and SURF. Extracted from [20].	28
Table 3 Comparison between Fast-Hessian and Difference-Of-Gaussian detectors. Extracted from [20].....	28
Table 4 Represents our main test device comparison with other Smartphone.....	50
Table 5 Represents our tests data of Panorama Quality and Resolution within performance time	53
Table 6 A comparision between our application and others	55

INTRODUCTION

GENERAL INTRODUCTION

The last decade has been marked by the widespread of mobile devices. A typical example is the proliferation of Smartphones, which have been an integral part of the daily life of people. Besides their communication purpose, smartphones are used for entertainment, navigation, and as pick and shoot cameras among other things. They are quickly replacing advanced digital cameras, thanks to their processing capabilities and the quality of their sensors. Moreover, with the quick development of mobile operating systems such as Android and iOS, it is common now to find image processing applications that provide a lot of functionalities that used to be available only on desktops.

In this Master's thesis, we are going to design and develop an application for creating panoramic images that can run on Android smartphones. The aim is to take several pictures using the smartphone camera and then combine them to produce a picture that has a wider field of view than what can be taken by the on-board camera. The application is designed to help the user to take the best possible individual pictures.

The key idea to produce a combined picture is based on image stitching technique, where images are searched for correspondences between them, and then aligned according to these correspondences. A final blending processes is applied to smoothen the artificial stitching marks. Image stitching is a well-known technique used in domains such as robotics, astronomy and medical imaging.

This thesis is comprised of four chapters, organized as follows:

- **Chapter 1 (Overview of domain and state of art):** We review the fundamentals of the Android OS. We will also review work related to panoramic imaging and image stitching.
- **Chapter 2 (Image stitching):** we will go through the principals of image stitching with more details. In this chapter, we will explain image features and see two popular feature extraction methods: SIFT (Scale-Invariant Feature Transform) and SURF (Speeded Up Robust Features).
- **Chapter 3 (Implementation and Design):** in this chapter, we give the details of our application design and implementation. We will show the UML diagrams

of the different java classes as well as the interfaces and different parameters used to control the internal algorithms used by the application.

- **Chapter 4 (Tests and Results):** in this chapter, we show results of running our application on some data sets and discuss the obtained performance. The data sets represent pictures taken under different conditions (indoors. Outdoors. Lighting,...etc.).

CHAPTER 1:
OVERVIEW OF DOMAIN AND STATE OF ART

1. Introduction

Since the very fast hardware development, and the lighter weight and thinner smart devices like smart-phones and smart-watches. new powerful and light weight electronic chips has been made. Nowadays, Companies are challenging each other to make even more better camera chips and more powerful GPUs and CPUs. Mobile applications developers are trying to keep the level with the development by profiting the maximum of these equipments advantages. All this has made the devices a well competitors to laptops and PC's performances.

2. Why Smartphone

Current Smartphone have an improved hardware with good computation, this made it possible to develop mobile applications that perform reasonably well compared to ordinary computers. Smartphone has been the most popular photo cameras, in comparing with digital cameras, film cameras or even smart cameras and smart watches. Also, a Smartphone is daily used by people, as shown in the figure 1 below.

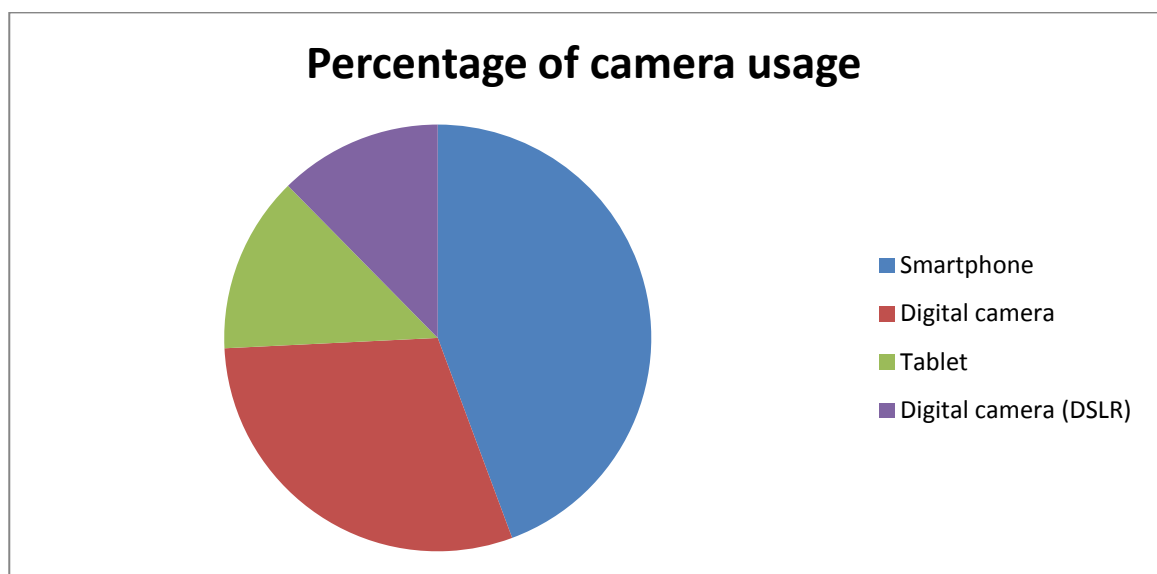


Figure 1 Statistics about the most used camera type in taking pictures, According to [1]

3. Android

Android is a mobile operation system (OS) that is designed for touch screen devices. It's a very customizable and easy to use, with widgets, backgrounds and shortcuts in home screen, and a numberless different applications and gestures that are inspired from real human gestures like swiping, holding and pressing. It supports more than a billion devices across the globe and owned by Google.

It has Linux kernel that provides a level of abstraction between the device hardware and all the essential hardware drivers, also it handles networking and power management. On top of Linux kernel there is a set of libraries like SQLite database, multimedia libraries, Internet security...etc., within android libraries as well these java-based libraries include building, graphics, drawing and database access. On top of that comes the Application framework layer that provides many higher-level services to applications in the form of Java classes, like activity manager, content providers, resource manager...etc. At the top layer we find Applications where all applications are installed to such contacts, books, browser, games applications etc. the figure below describes these layers. [2].

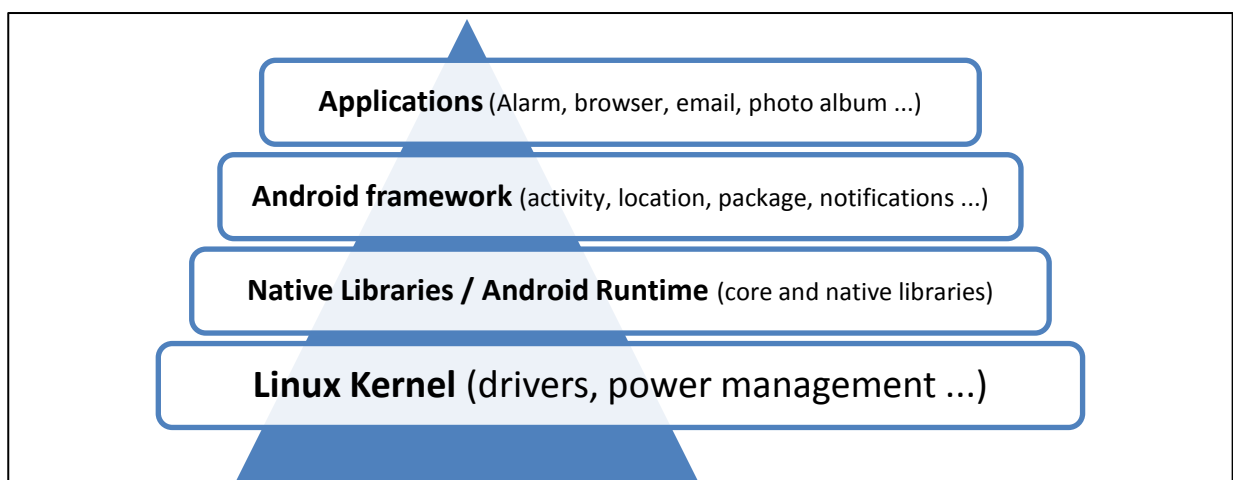


Figure 2 Android Operating System Architecture.

Because, due to its open-source licensing, Android development is very rapid. it is the most leading mobile OS in the world. more than 50% of Smartphone are Android based. Also its system supports background processing, provides a rich user interface library and supports 2D and 3D graphics using the OpenGL-ES (short OpenGL) standard.

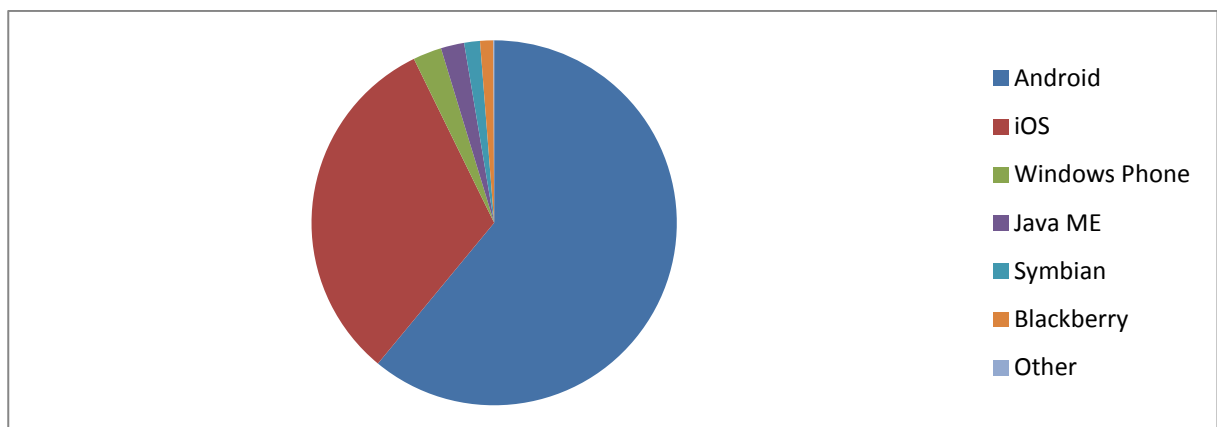


Figure 3 Statistics about Mobile Operating System Market Share, According to [3].

4. What is Image Stitching

Image stitching is the process of combining several images together, to create a wide field of view images, due to the camera's limits unlike the human eyes FOV. Image Stitching allows us to create photos that are 360° wide and even more like a spherical photo that can be used in virtual reality applications, all with a single and simple limited FOV camera. Example in the two figures below.



Figure 4 A Set of Pictures that will be combined using Image Stitching.



Figure 5 The set of pictures in the previous figure after image stitching. Extracted from [4].

5. The difficulties of Image stitching

Not enough overlap between frames and Distortion can make the images much harder to stitch. Moving object also can cause blending errors, because there will be a time difference between shooting the frames. Also there is the lack of visible features when photo has large areas without many obvious features like blank wall or clear blue sky, also exposure problems can cause the same thing. Highly repetitive identical features like an empty football stadium, can make such a confusion in stitching process. Extracted from [5].

6. What is Panorama

Panorama is any wide field of view representation of a physical space. A combination of limited angle views. In digital terms panorama is made of stitching images together to create high resolution and wide FOV images. you can see Figure 5.

7. Panoramic cameras

There are a several solution before android panoramic camera, but their major problem is that they cost too much, and they are not a daily use device also too big and heavy to handle that can't be with the user all the time, unlike the Smartphone that can even fit in pocket.

7.1. Using a physical support

There are many camera supports for panorama shooting, for example like **Bushman Panoramic Gobi head** that it's a good quality cheap support to DSLR Digital Cameras that its price is around 166\$ (not including the camera), in figure 6 below.



Figure 6 Bushman Panoramic Gobi head, extracted from [6].

7.2. Professional Panoramic cameras

PanoScan MARK III That is an expensive professional panoramic camera that can produce 360 degree images and supports QuickTime VR. with ability to shoot in low light, automatic white balance, and a powerful rotation drive that operates in any position. Priced at 40.000\$.



Figure 7 PanoScan MARK III camera.

7.3. Multi-lenses Camera

Well-known companies like Facebook, Samsung, LG are starting to develop Multi lenses cameras for taking a 360° videos and photos that's used later in Virtual Reality. These devices mostly don't need any image stitching.



Figure 8 The Facebook Surround 360 Camera. Extracted from [7].

8. Image Stitching and Feature Extraction Applications

8.1. Video stabilization

Many cameras have been supporting a video stabilization features, it depends mainly on features (key points) extraction and then re-positioning the video frame according to a stable Key points. This process can be done while recording the video or after. it's main goal is to avoid the disturbances (camera shake) in a video.



Figure 9 Represents before and After Applying video stabilization.

8.2. Face recognition

Face recognition uses feature extraction and match the same in image stitching. it's base idea is to extract face image features of a case and search for the same or closest matched features in the database. it's well used in Smartphone, and home smart security systems.



Figure 10 An Example of Face Recognition Application, a kid trying to access his dad's phone. Extracted from [8].

8.3. Gesture interface

It's been always around in science-fiction movies, where the actor seems to control in human-machine interface with hand gestures and movements. But, today It became a real life application. Smartphone, smart house systems, smart TVs and game consoles had been supporting these technologies for a while. However, it mainly depends on feature extraction and comparison exactly the same like in image stitching. The figure below shows an experience with Xbox KINECT.



Figure 11 A man playing a game on Xbox Kinect, using his hand geastures. Extracted from [9].

8.4. Virtual Reality (Augmented Reality)

It's a computer technology that replicates an environment, real or imagined, and simulates a user's physical presence that environment in a way that allows user to interact with. Image stitching and object removal is well known in VR domain. These technologies usually use panorama images like We are going to generate by our application, to simulate a reality scene.

8.5. Maps Applications

It's actually stitching flat images, a high resolution photo mosaics that used in digital maps that was captured by satellites. Like Google Maps.



Figure 12 Example of two Satellite images begin stitched together. Extracted from [10].

8.6. Robotics

Image stitching is also used in robotics. In a multi-agent system in coordination of agents for example robots that have to detect objects and figure out their way in the environment according to multi images weather from the same robot agent or from multiple robots.

8.7. Medical

Image stitching is very common in the medical field. it has been used to generate panoramic X-ray images due to the limits of medical equipment, panorama is obligated to do in certain cases. Unlike any other panoramic application preciseness in generating a medical panorama is number one priority.

9. Conclusion

Today, Android is the most leading operating system in the market share, also the most common and used camera is the Smartphone's. An android camera application will Target a huge amount of users, a panoramic application with a good quality will also attracts professional photographers. In the next chapter we are going to see some of image stitching that will be used to generate panorama pictures.

CHAPTER 2:

IMAGE STITCHING

1. Introduction

The goal of this chapter is to study some image features extraction and matching methods like SIFT and SURF, and see a simple comparison between them.

2. Image features

A Feature is used to indicate any location in the image which exhibits a relevant property. it is an interest point, an image location that is interesting for the application at hand. its main purpose is to select locations (points, edges, curves or regions) in the image that are likely to be useful candidates. it's an image location that exhibits the following properties.

3. Introduction to Image Stitching

Image stitching is the process of combining a number of images taken side to side into a composite image (panoramic picture). the difference between the field of view of human visual system and the camera is huge (135 x 200 degrees for human and only 35 x 50 degrees for a typical camera, it depends on the lenses used). Therefore, image stitching aim is to achieve a wide level of field of view although with keeping the quality of images and sometimes with a better resolution result.

Image stitching applications vary from panoramic camera applications, medical imaging applications, satellite imaging, robotics,..etc.

3.1. Generating Panorama Photos (Hewlett-Packard Labs)

A paper by HP that presents techniques to handle some practical issues when generation panorama photos like finding the stitching line (stitching line is the area that corresponds with two images on stitching process, this area will be later blended by some filters and methods), and moving objects in pictures in time, by [11]. The Figure 13 below represents one of their results.

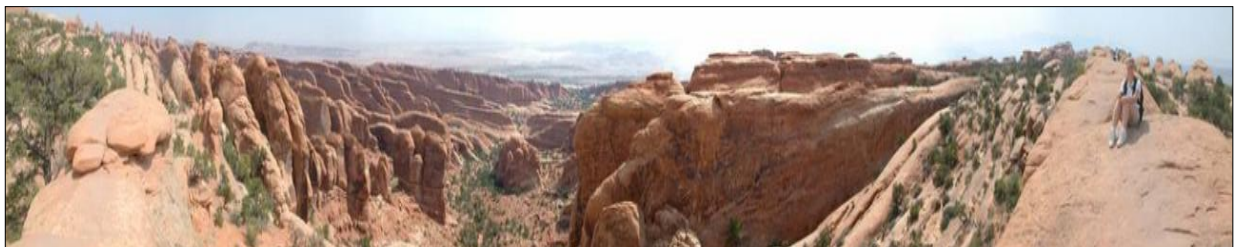


Figure 13 Panorama Image Result of Arch national park in Utah by Hewlett-Packard Labs Paper [11].

3.2. Stitching of X-ray Images (UPPSALA university)

Medical imagery systems usually have a limited field of view, resulting in images that do not cover large parts of patient's body. This paper [12] studied HARRIS, SIFT and SURF and modified exhaustive nearest neighborhood methods to get faster matching of key points. by Krishna Paudel.



Figure 14 Represents Image Stitching Application in X-ray, of stitching three images, By [12].

4. Main components of image stitching

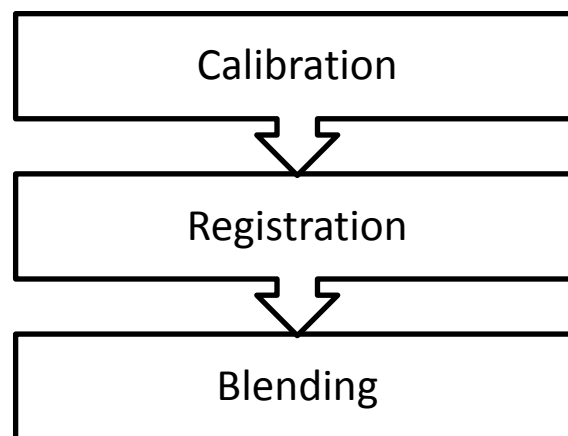


Figure 15 Image stitching main steps: Calibration, Registration, Blending.

There are three main components in image stitching: calibration, image registration and blending. camera calibration aims to estimate the extrinsic and intrinsic camera parameters. registration is the process of saving and comparing the images to find the translations that can help in aligning of the images. and then images will be blended into a single image.

4.1. Calibration

This component aims to minimize the differences between an ideal lens model and the camera-lens combination that was used. These differences result from optical defects such as distortions and exposure. Extrinsic camera parameters define the location and the orientation of the camera reference frame to a known world reference frame. Intrinsic camera parameters link the pixel coordinates of an image point with the corresponding coordinates in the camera reference frame. This process uses a basic camera model called *pinhole*, this model is used when modeling a standard camera in computer graphics. it is a major simplification because real cameras are more complicated. For further information, check this paper [13].

Figure 16 shows the basic structure of pinhole camera model, a real word point P_c assumed to be in the camera coordinate system that has its origin at O_c (the optical center), the projected point p lies on the sensor plane which determined by the 3D origin C_s . S_x and S_y are the scaling factors and Z_c defines the so-called optical axis.

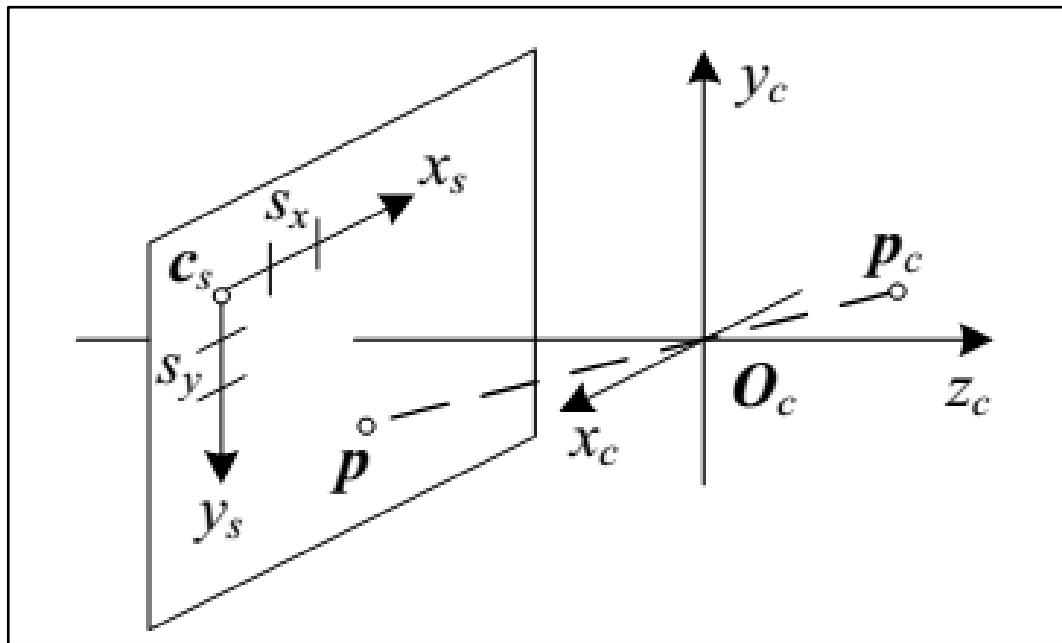


Figure 16 Basic pinhole camera model. Extracted from [13].

4.2. Registration (Alignment)

The purpose of image registration is to create geometric correspondence between the registered images by comparing and then aligning them (also known as Global alignment), using distinct features, and estimating the transformation matrices from them. That way image blending is processed to join the images smoothly.

4.3. Blending

When creating a panoramic image, its input images will overlap each other this causes stitching marks to be strongly visible. Blending is applied across the stitching region to hide its marks and resolve the stitching errors. There are many ways of blending, like feathering and Gaussian pyramid... For further study [13].



Figure 17 the image before blending. Extracted from [13].



Figure 18 panorama mosaic after applying a blending method called multi-band blending. Extracted from [13].

5. Image stitching techniques

5.1. Direct techniques

Direct techniques compare pixel intensities of the images to stitch. Their main advantage is that they minimize the sum of absolute differences between the pixels of overlapping areas. The main disadvantage of direct techniques is that they have a limited range of convergence.

5.2. Feature-based techniques

In feature-based techniques stitching is done by detecting and matching feature points in an image pair, features like points, lines, edges, corners or any other shapes. There are many feature detector techniques like Scale-Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF). We will to see how SIFT and SURF works and what differences exist between them.

5.2.1. Scale-Invariant Feature Transform (SIFT)

SIFT is an algorithm that was published by David Lowe in 1999 [14], its main task is to detect and describe local features in images. Features that are geometrically invariant to scaling and rotation transformations. They are also partially invariant to change in illumination. Providing a large number of features that can be extracted from any image with an efficient algorithm. In addition, features are very distinctive and different from each other, which allows a single feature to be correctly matched with high probability against a large database of features. However, there are four major stages to generate the set of features.

a. Scale-space extrema detection: It searches over all scales and image locations using a Difference-Of-Gaussian function to identify potential interest points (Extrema) that is whether minima or maxima point. Scale and location can be calculated by the following function:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Equation 1 Applying Gaussian filter (G) to an Image (I), with a scale value (σ).

* is the convolution operator,

σ is the scale value,

$G(x, y, \sigma)$ is a variable-scale Gaussian,

$L(x, y, \sigma)$ is the image after applying Gaussian filter.

$I(x, y)$ is the input image.

Then locating the scale-space extrema using Difference-Of-Gaussian function, by computing the difference between two images, one with scale k times the other:

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

Equation 2 Difference Of Gaussian of two images, one with scale of k -times the other.

To detect the local maxima and minima of $D(x, y, \sigma)$ each point is compared with its 8 neighbors at the same scale, and its 9 neighbors up and down one scale. If this value is the minimum or maximum of all these points, then this point is an extrema.

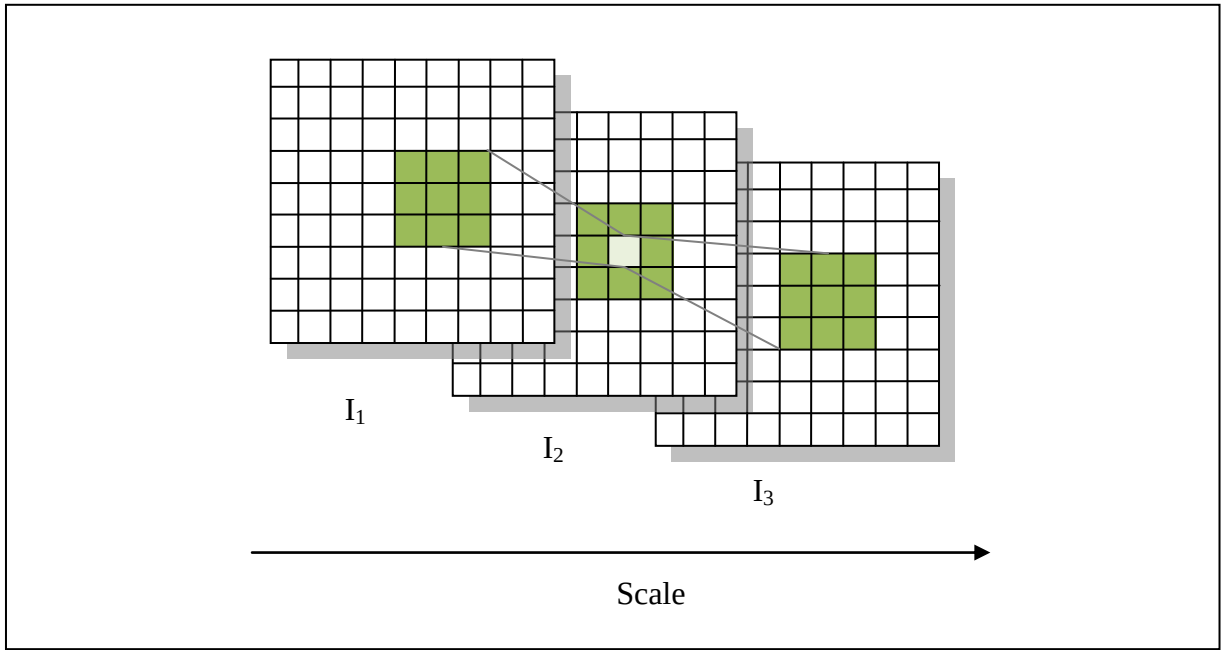


Figure 19 Represent the comparison between a point and its 8 neighbors in the same scale and its 9 neighbors of up and down scale, to detect the maxima or minima, I_1 is the downscale and I_3 is the upscale

This stage is the most consuming, because it applies many Gaussian filters with different variances.

b. Keypoint localization: Consists of eliminating the unstable keypoint candidates that have been detected in the previous stage. And keeping the well localized keypoints. Finding the location of the extremum $\hat{x}$ can be achieved by calculating the Laplacian:

$$\hat{x} = \frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x}$$

Equation 3 location of the extremum point, achieved by the laplacian.

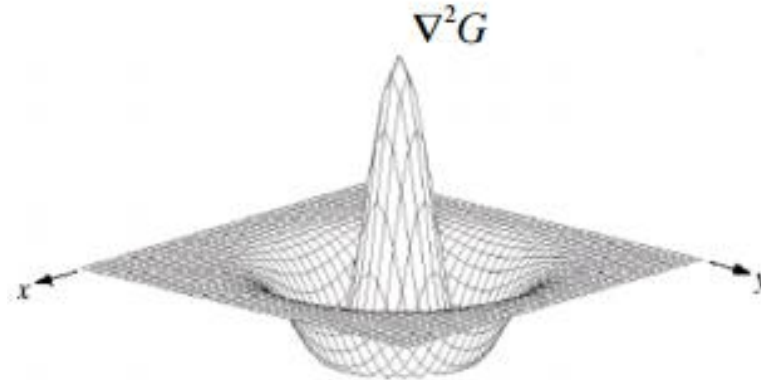


Figure 20 Laplacian of a Gaussian. Extracted from [15].

c. **Orientation assignment:** Assigning a consistent orientation to each keypoint based on local image properties so the keypoint descriptor can be represented later. The gradient magnitude $m(x,y)$ and the orientation $\theta(x,y)$ is pre-computed using pixel differences, which means that these calculations for the gradient are done for every pixel in neighboring region around the keypoint in the Gaussian-blurred image L .

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

Equation 4 Gradient magnitude.

$$\theta(x,y) = \tan((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Equation 5 Orientation equation.

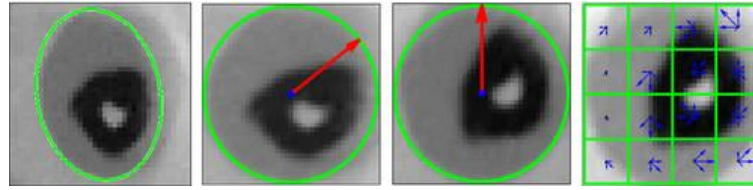


Figure 21 Represents SIFT orientation vectors, image extracted from [16].

d. **Keypoint descriptor:** Using the local gradient data to create keypoint descriptors. The gradient information need to be rotated to line up with the orientation of the keypoint and then to be weighted by a Gaussian with variance of $1.5 * \text{keypoint scale}$. Keypoint descriptors typically use a set of 16 histograms, aligned in a 4×4 grid, each with 8 orientation bins, one for each of the main compass directions and one for each of the mid-points of these directions. the results in a feature vector containing $4 * 4 * 8 = 128$ elements. Explained in the figure below.

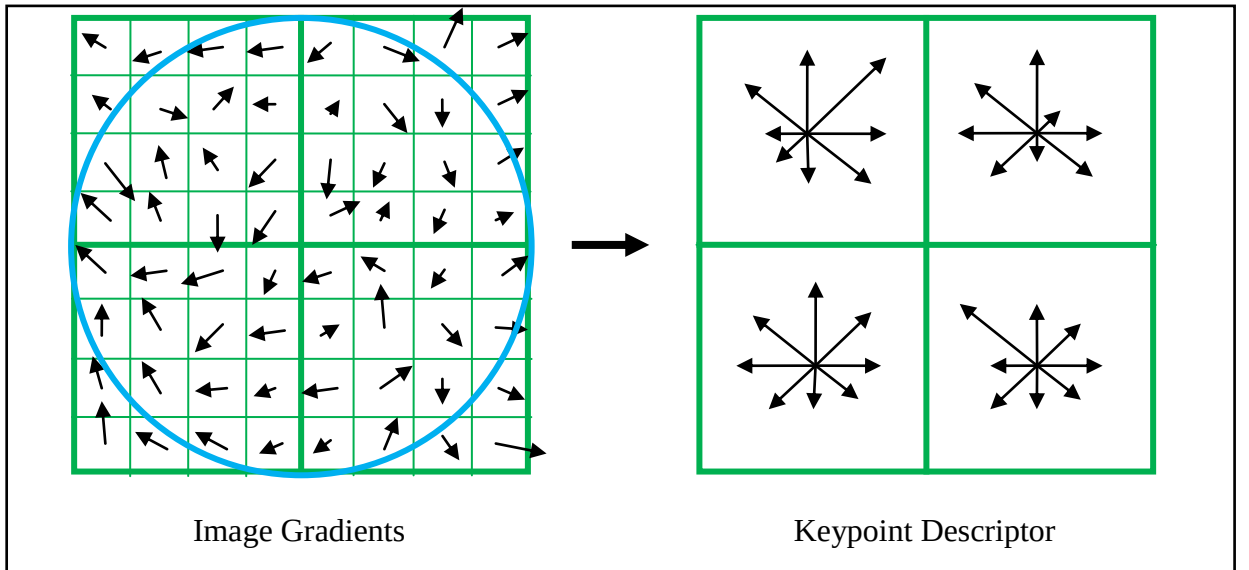


Figure 22 Generating Image descriptors.

5.2.2. Speeded Up Robust Features (SURF)

SURF method was first presented by Herbert Bay at the 2006 European Conference on Computer Vision (ECCV). it's a local feature detector and descriptor that is inspired from SIFT. [17]

The SURF method is based on the same principles and steps of SIFT approach, but it uses 64 dimensions and utilizes a different scheme with the aim of being faster and more accurate. It can be divided into 3 steps:

a. Detection: identifying interesting features (interest points) that have to be done robustly. SIFT uses a cascading filtering approach. Where the Difference of Gaussians, DoG, is calculated on progressively downscaled images. SURF uses a Hessian based blob (a square-shaped filter) detector to find interest points. In that order the integral image must be computed.

$$I_{\Sigma}(x) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j)$$

Equation 6 Integral image.

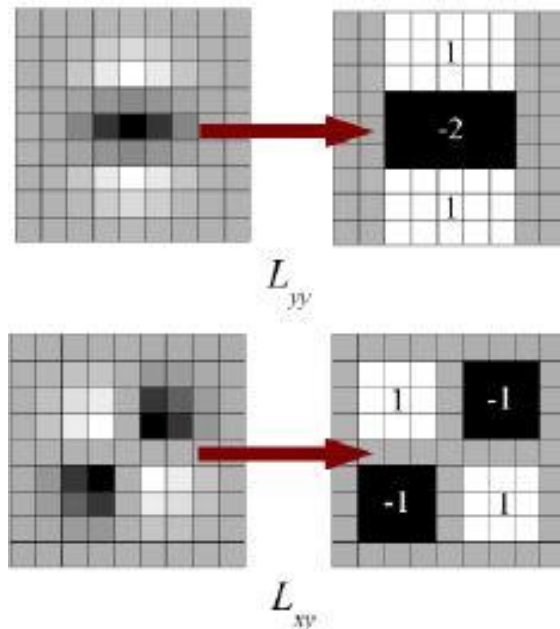


Figure 23 the integral images (the sum of all pixels that are used later in Hessian-matrix) before and after applying Hessian detector's filter (square-shaped).

SURF uses the determinant of the Hessian Matrix for selecting the scale, In a given point $x(x,y)$ and scale σ :

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix}$$

Equation 7 Hessian-matrix that is combining the four integral images: xx, xy, xy, yy.

$$Det(H) = D_{xx}D_{yy} - (wD_{xy})^2$$

Equation 8 Determinant of Hessian-matrix.

Where $L_{xx}(x, \sigma)$ etc. are the second-order derivatives of the grayscale image.

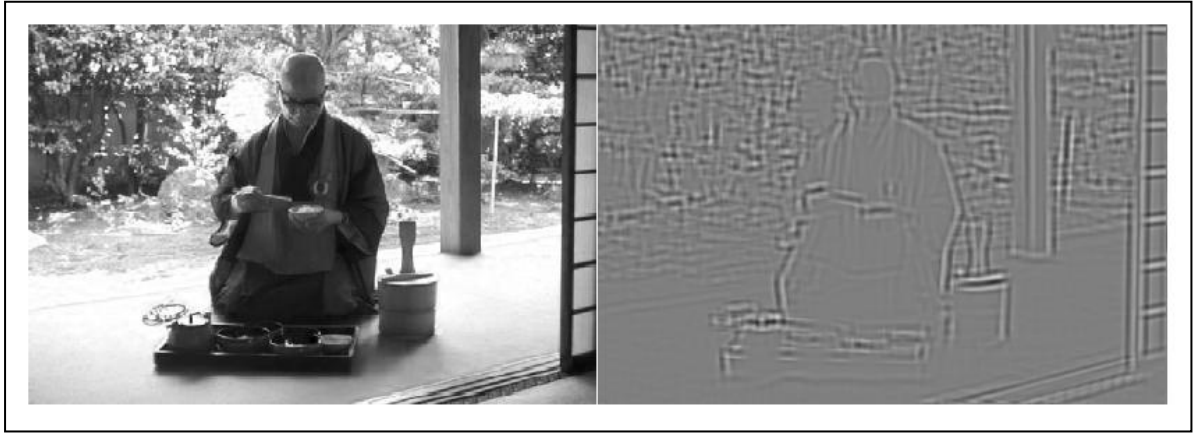


Figure 24 Grayscale image in the left and the image after applying box filter. Image extracted from [18].

For orientation assignment, SURF uses wavelet responses in horizontal and vertical direction for a neighborhood of size $6s$, then they are plotted in a space as given in the below figure. The dominant orientation is estimated by calculating the sum of all responses within a sliding window with an angle of 60 degrees.

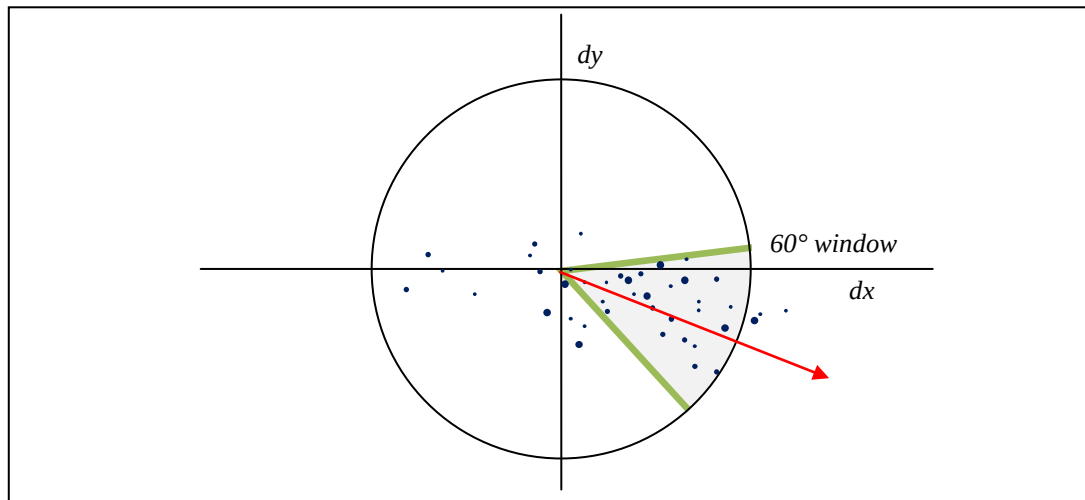


Figure 25 Determining Orientation in SURF.

b. Description: giving a unique and robust description for every interest point regardless of scale and rotation. It is done by describing the intensity distribution of the pixels within the neighborhood of the point of interest.



Figure 26 Example of image and SURF features detected. Extracted from [19].

c. Matching: Determining which objects an image contains based on predetermined interest points and by comparing the descriptors obtained from different images.

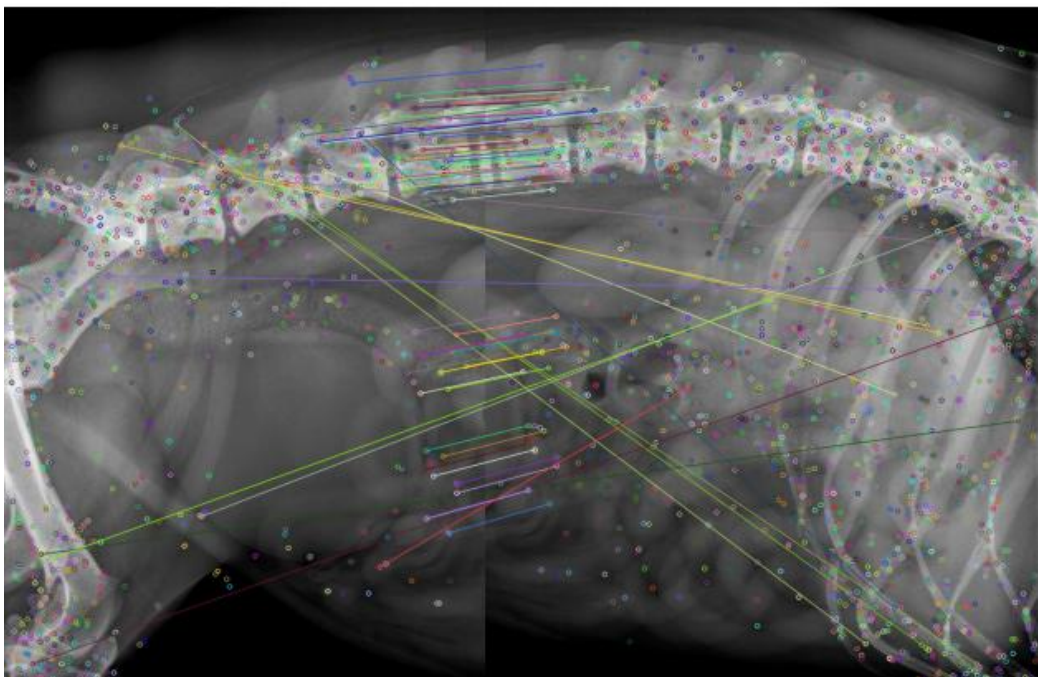


Figure 27 Matching two images. Extracted from [12].

6. Comparison

	SIFT	SURF
Scale Space	DOG convolved with difference size images	Difference size box filter convolved with integral image
Keypoints detection	Local extrema detection > Non maximum suppression > Eliminate edge responses with Hessian matrix	Determine the potential keypoints with Hessian matrix and Non-maximum suppression
Orientation	Uses image gradient magnitudes and orientations are sampled around the keypoint location, using the scale of the keypoint to select the level of Gaussian blur for the image	A sliding orientation window of size 60° , detects the dominant orientation of the Gaussian weighted wavelet responses at every sample point within a circular neighborhood around the interest point
Descriptor	The keypoint descriptor allows for significant shift in gradient positions by creating orientation histograms over 4×4 sample regions.	An oriented quadratic grid with 4×4 square sub-regions laid over the interest point. For each square the wavelet responses are computed from 5×5 samples. For each field we collect the sums dx , $ dx $, dy , $ dy $.

Table 1 An overview comparison between SIFT and SURF. Extracted from [20].

For a performance comparison between SIFT and SURF the reader can refer to [20] that they tested with this image and custom properties.



Figure 28 The Test Image of [17], size of the image is 800×640 pixel.

6.1. Performance Comparison

SURF is obviously a faster method with good results, shown in Detector Performance Comparison below. SURF is faster due to Fast-Hessian Performances and because it uses box-filter and determining orientation method (sliding window).

Method	Time (msec)
SIFT	1036
SURF	354

Table 2 Represents comparison of processing time between SIFT and SURF. Extracted from [20].

6.2. Detector Performance Comparison

Fast-Hessian that is used by SURF is almost as good as Difference-Of-Gaussian, and 5 times faster.

Detector	Number of points	Computing time (msec)
Fast-Hessian	1418	120
Difference-Of-Gaussian	1520	650

Table 3 Comparison between Fast-Hessian and Difference-Of-Gaussian detectors. Extracted from [20].

7. Conclusion

According to several Image stitching algorithm comparisons, SURF is the closest practical algorithm available, it's more distinctive and more compact leading to significant improvements in matching accuracy and speed. SURF has good performance when it comes to rotation and illumination invariance, which will be needed in panoramic image stitching. In the next chapter we are going to see our application implementation and design of image stitching application using SURF method.

CHAPTER 3:

DESIGN AND IMPLEMENTATION

1. Introduction

The goal of this chapter is to show the design and implementation of an android application that uses image stitching to create panoramic images. And taking profit the maximum of android system functionalities and smartphone features like (sensors, notifications, vibrations ...), to take the very best pictures by guiding the user and doing calculation and pre-detection and some other preparations.

2. Tools and software used

Through our project development process, we have used several software tools to implement, design and test our application.

2.1. Android SDK

The Android Software Development Kit contains the necessary tools to create, compile and package Android applications. Most of these tools are command line based. The primary way to develop Android applications is based on the Java programming language.

2.2. Android NDK

NDK or Native Development Kit is a set of tools that allows developers to implement a parts of the application using languages such as C/C++. It basically builds native libraries (.so files) from the application's native source code. We used this to implement our two native algorithms in C++: Matching and Stitching, that are based on OpenCV (a library of visual domain algorithms and resources) that we are going to see next.

2.3. Android Studio

It's an Android focused IDE. Android studio contains all the Android SDK tools to design, test, debug and profile any android app. It is similar to Eclipse with the ADT plug-in, but there are many features available in Android Studio that can increase development productivity. features like:

- Powerful code editing
- Rich Layout Editor
- Gradle-based build support
- Template-based wizards

- Lint tool analysis (it's a static code analysis that checks android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility and internationalization).

2.3.1. Gradle

Gradle is a free Java build tool for project automation, similar to Ant and Maven. It is used by software developers working in Java for compiling code, resolving dependencies, packaging, unit testing, code execution and deployment. It's a very flexible general purpose build tool, switchable, build-by-convention frameworks, very powerful support for multi-project builds, very powerful dependency management. Extracted from [21].

```
buildscript {
    repositories {
        jcenter()
        //build scripts goes here ...
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.4.0'
    }
}

allprojects {
    repositories {
        jcenter()
        //build scripts for all projects goes here ...
    }
}
```

Figure 29 A simple code of Gradle configurations, that it shows a simple gradle configuration file that is responsible to define build configurations.

2.4. OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source library that includes several hundreds of computer vision algorithms. It was started by Intel in 1999 and the first release came out in 2000. It supports a wide variety of programming languages like C++, Python, Java...etc. OpenCV algorithms are expanding day-by-day. OpenCV provides the following modules:

2.4.1. Core A compact module defining basic data structures, including multi-dimensional array Mat and basic function used by all other modules.

2.4.2. Imgproc an image processing module that includes linear and non-linear image filtering, geometrical transformations (resize, warping, generic remapping), color space conversion... and so on.

2.4.3. Video a video analysis module that includes motion estimation, background subtraction and object tracking algorithms.

2.4.4. Calib3d basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation... and so on.

2.4.5. Features2d a module for feature detectors, descriptors and descriptors matchers.

2.4.6. Objdetect detection of objects and instances of the predefined classes like for example faces, eyes, people, cars... etc.

2.4.7. Highgui an easy to use interface to video capturing, image and video codecs, as well as simple UI capabilities.

OpenCV is only a translation of mathematical methods that were studied by experts and professors into algorithms, it is similar to Matlab that can be used as a programming library (it doesn't provide a full algorithm of making panorama, but it can be used to do that).

We use OpenCV's native interface to gain the maximum performance and because android provides more java memory heap for native libraries (that is executed by NDK) unlike in java part, we need every memory space available due to much image processing.

2.5. ArgoUML

ArgoUML is an UML diagramming software, that is written in Java and released under the open source Eclipse Public License. One of the best UML diagramming tools. It was originally developed at UC Irvine by Jason E. Robbins, it's now an open source project hosted by Tigris.org.

3. Generating Panorama Process

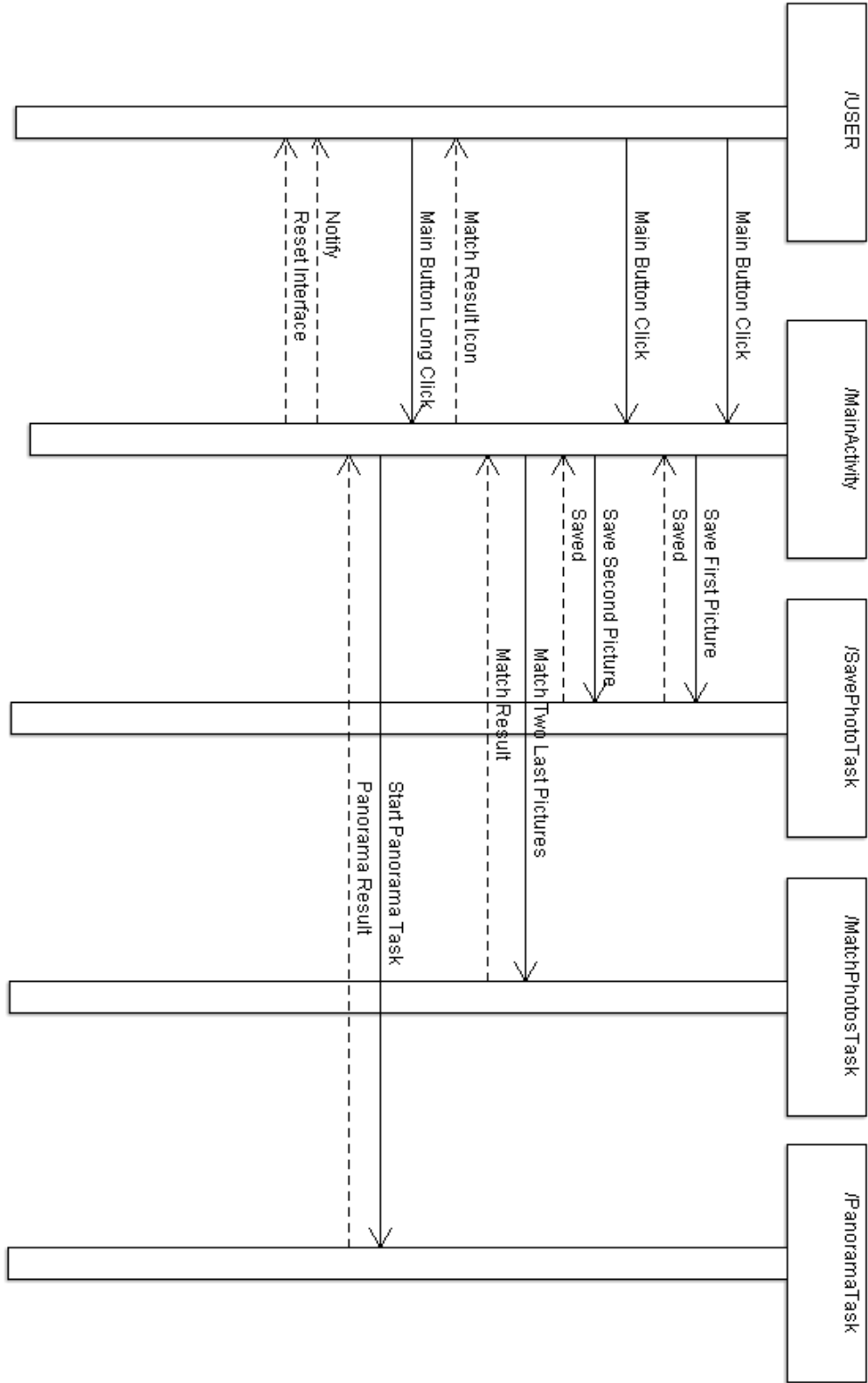


Figure 30 Generating Panorama Process, this diagram represents between 4 classes of our application and the user.

First the user interacts with our application's interface, to take the panorama pictures he uses the main button, every single click saves a picture. The first picture is saved without a matching verification process being done, then the user takes a second, third and so on..., every time he takes an image after the first one, the matching task is called to verify the correspondence between every two following images and it shows at the interface the verification result. Then, when user long clicks on the main button to start a panorama task, this task is explained in the state-transition diagram figure up, then the panorama result is returned from the stitching process, and our activity notifies the user using the notification central of android and resets the interface to take another panorama image.

4. Panorama Image Processing

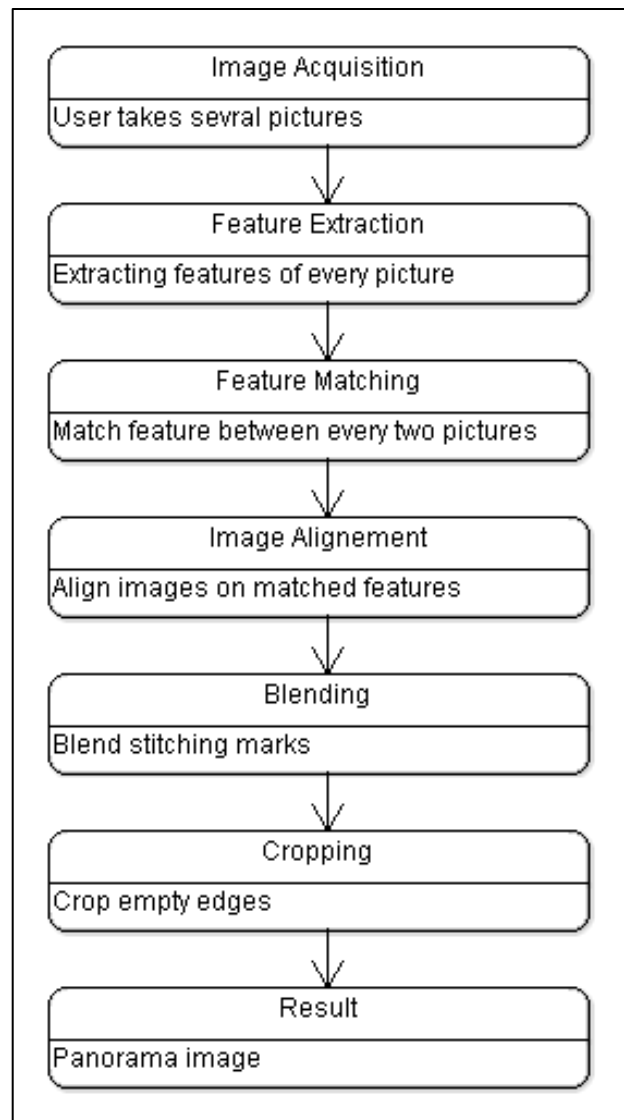


Figure 31 An state-transition diagram of our panorama image processing.

4.1. Image acquisition

In this step, user takes panorama pictures, with caring of guiding, these images which will be used later to generate the panoramic picture.

4.2. Feature extraction

After the user long clicks on the main button, stitching process starts, this step is the first step in image stitching, where features being extracted from the token images in the previous step.

4.3. Feature matching

This step uses the extracted features from the previous step, and matches these features in every two images, to assure correspondence between the images.

4.4. Image alignment

This step where images begin to be aligned with each other according to the matched features. This is aligning, positioning, scaling, rotating and warping the images to generate the unfinished panorama picture.

4.5. Blending

The previous step leaves stitching marks in panorama image, this uses several methods to blend these stitching marks that the image will look original.

4.6. Cropping

After image stitching is completely finished the generated image have a black empty edges due to warping the images in cylindrical or spherical shapes, this step crops these edges and leaves only the safe panorama area.

5. UML Classes Diagram

Our application's UML Classes diagram, of three main classes, Main Activity Class which represents all the functionalities, and prepare Static Class which prepares the static part before the application runs, and the Preview Class that deals with camera and drawings in interface.

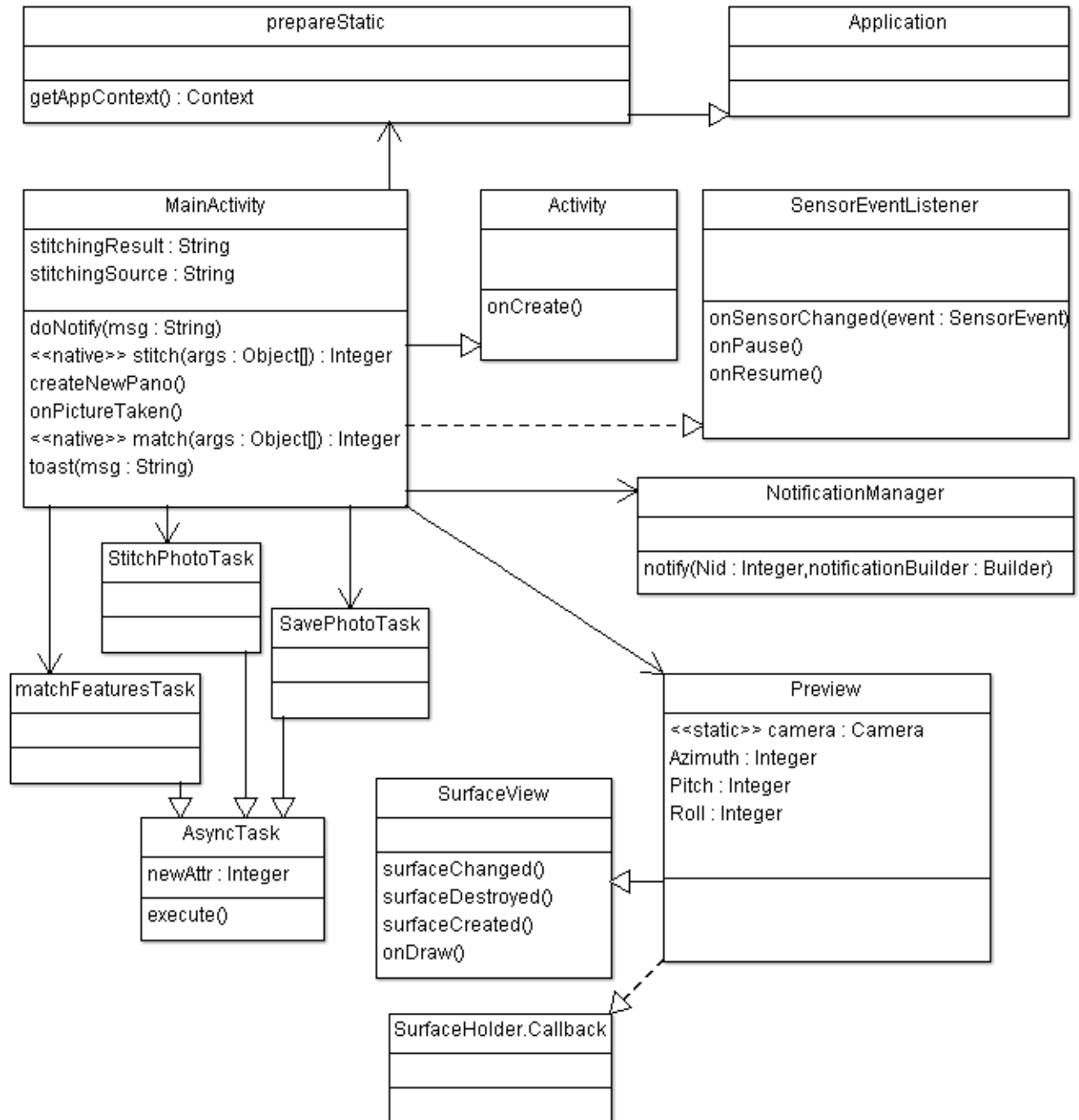


Figure 32 Our main UML classes diagram.

5.1. Main Activity Class

This is the main class of our application that deals with other classes, it mainly responsible for the application process activity (extends Activity class). This class also deals with sensors (implements SensorEventListener). Also this class is responsible for saving, stitching and matching verification tasks.

5.2. Preview Class

This class deals with the camera by SurfaceView class (it extends this class) that it connects to the camera and setups its parameters and links the surface view with camera buffer. And deals also with the shown graphics like the guiding lines and the tilt guiding arrows and other icons.

5.3. Prepare Static Class

This class prepares the static libraries (stitching and matching verification process within OpenCV library), and links the main activity with these libraries by passing the application context, it is necessary to do in android when using these libraries. It is called first in our application.

6. Application Interfaces

The main interface of the application is shown in Figure 33 below, and the configuration interface is shown in Figure 34.

6.1. Main Interface

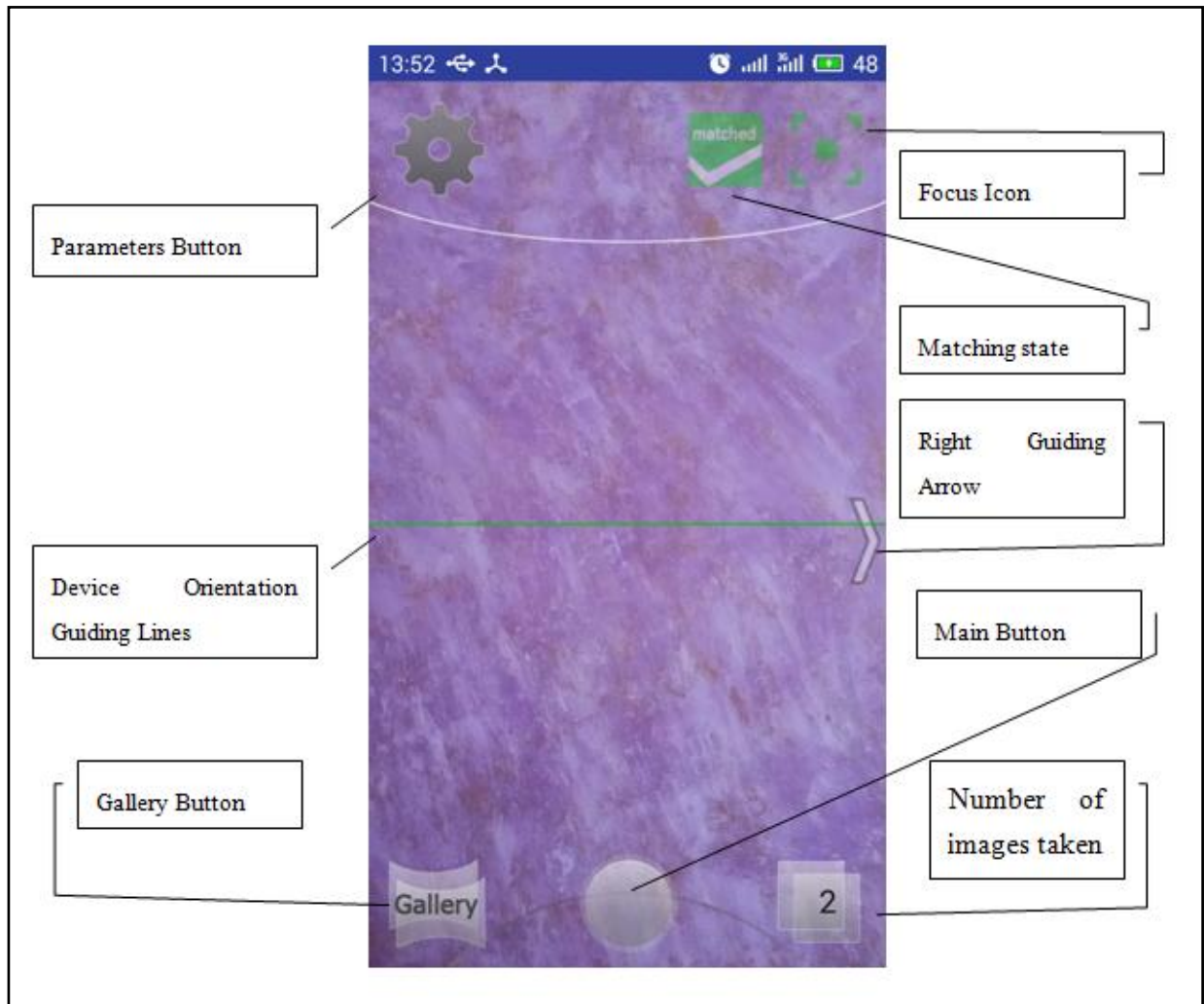


Figure 33 Our application's main interface.

In this main interface we have:

- Main button for taking pictures and starting the panorama process, the Gallery button for showing previous panoramic pictures, Parameters button for showing the configurations interface.
- Focus icon to represent focus state, Matching verification icon to represent matching state, and the number of images taken in the current panorama, and the guiding arrows and lines.

6.2. Configurations Interface

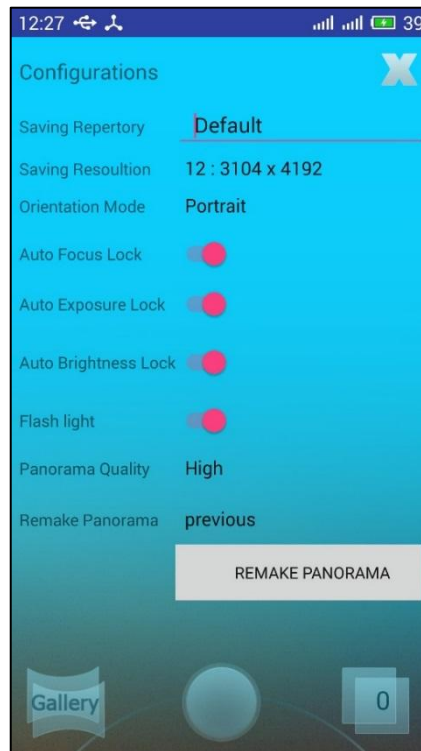


Figure 34 Interface of configurations.

In this configuration interface we have:

- Saving repertory: where user chooses where to save the panorama project images and that he can remake the panorama later
- Saving resolution: is resolution of the input images
- Orientation mode: whether in portrait or landscape
- Auto Focus lock: to lock camera's auto focus feature, that we will explain later
- Auto Exposure lock: to lock the auto exposure of camera through taking the panorama pictures.
- Auto Brightness lock: to lock brightness auto changement of the camera through taking the panorama pictures.
- Flash light: is to enable or disable the camera's flash light for taking the panorama images.
- Panorama Quality: the quality of image stitching that we will talk about later in this chapter and the next one.
- Remake panorama: is an option to remake a previous panorama using different configurations.

7. Functionalities

7.1. Main Button

Clicking the main button once takes a picture. Holding the button starts image stitching to make the panorama if there are two or more pictures in current panorama shooting cycle.



Figure 35 Main Button.

7.2. Android device orientation angle guiding lines

Two lines are drawn in the screen based on sensor values. Their goal is to guide user to take picture with the right angles. One line for roll angle and the other for pitch angle.

7.3. Camera Autofocus state

An icon in the top right of application's main screen that indicates auto focus state. Its color becomes red when autofocus feature is still processing and then turns green when ready. This helps user to wait for autofocus calculations and then take the picture with the right focus.

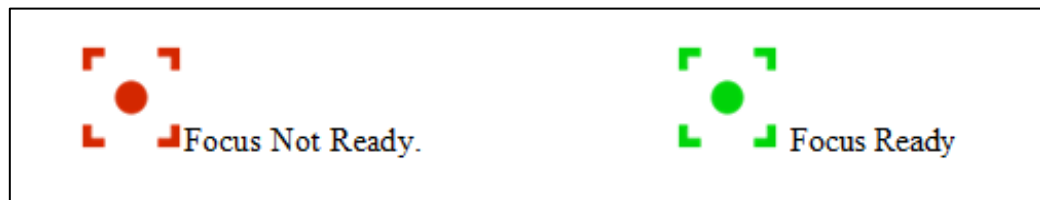


Figure 36 Focus states, Not Ready: when Camera Focus is still processing, Ready: when Camera Focus processing is done.

7.4. Camera Autofocus Lock

A Parameter that allows the user to set the autofocus lock on or off. its functionality is to automatically lock focus after taking the first panoramic picture, to avoid focus differences between panoramic parts before stitching. The goal is to ensure no object will change the focus of panorama parts.

7.5. Taken panorama pictures

A number of pictures already taken by user, appear to the right of the main button.



Figure 37 Represents Number of taken pictures, that will be stitched to make the panorama picture.

7.6. Notifies user on panorama has created

By profiting Android notification center, our app notifies the user when the panorama picture has finished processing, and provides a direct button to open the result image.

7.7. Tilt Guiding

Using the phone's Magnetometer sensor, we use the current angle in comparison to the north pole (not accurate to the real north pole, but useful) direction. After we apply a low pass filter to avoid any disturbance in received values. Pictures are stored with their angles so that the user can be guided in taking the next pictures.



Figure 38 Represents Tilt Guiding Arrows.

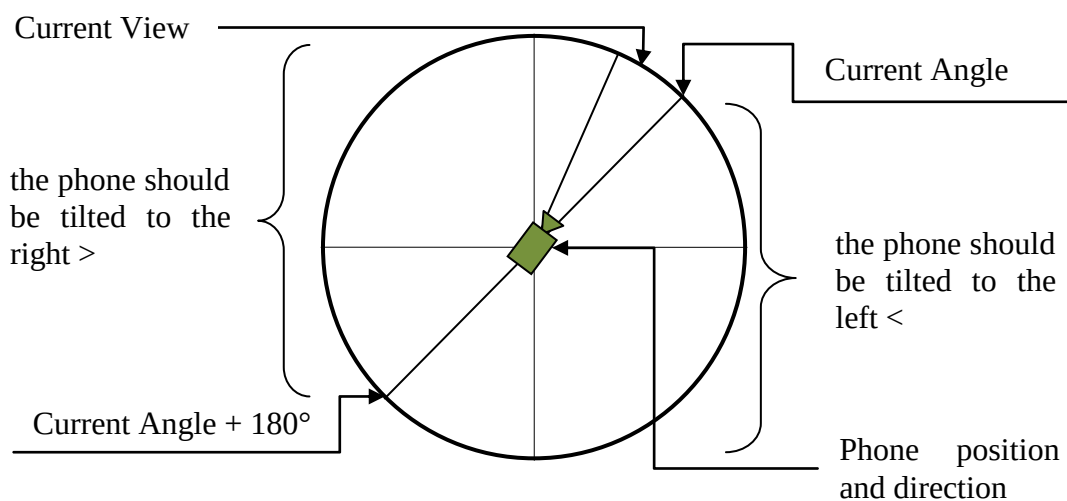


Figure 39 Represents where the phone should be tilted to.

7.8. Matching verification

We do feature matching between the current taken picture and the previous one, to ensure that pictures that following each other have enough correspondence to do image stitching later. The user is shown whether a red or green icon. If the two pictures have no correspondences, the user is informed to retake the last picture.



Figure 40 Two icons to inform the user about the matching result.

7.9. Gallery

To have access to the location of where the panoramic images are stored, we have put a gallery button for the user.



Figure 41 Gallery Button. The user can touch it to access panoramic images.

7.10. Parameters

The application configuration for modifying various parameters like resolution, stitching quality, folder name...etc. Unlike many other applications.

7.10.1. Input Images Resolution

The user can select image resolution using this menu item.

7.10.2. Panorama quality

A parameter to choose panorama quality level of four choices: High, Intel Fast Panorama Parameters, Medium and Low.

These different qualities are a preset stitching parameters, for example Low is faster but with a lower quality, Intel FPP extracted from [22], and High will take much longer time but a better quality, these sets of parameters will be used in the stitching algorithm to generate the panoramic image depending on them.

7.10.3. Autofocus Lock feature

Users can activate or deactivate autofocus lock, before taking the panorama pictures.

7.10.4. Ability to re-compute a previous panorama with different configurations

Users can choose any previous panorama, and re-compute the panorama using new configurations.

8. Dealing with Device Sensors

A sensor is a device that measures a physical quantity and converts it into a signal. Smartphone sensors are electronic chipsets that detect and gather data for various usage purposes.

Most Android-powered devices have built-in sensors that measure motion, orientation, and various environmental conditions.

In our application we used two sensors: Accelerometer and Magnetometer.

8.1. Accelerometer

The accelerometer is a sensor that measures change in velocity in axial dimensions (typically 2 or 3 axes). We have used the accelerometer sensor to calculate the roll and pitch angles. The goal is to guide user while holding the device to take the pictures.

Roll is the rotation angle around the y axis of the device, and pitch is rotation angle around the x axis, as shown in the figure below.

8.2. Magnetometer

Magnetometer is a sensor that detects the Earth's magnetic field. compass applications use this sensor to point at the planet's north pole.

We Used Magnetometer (digital compass) to define an approximate north pole direction, to use this direction angle (called azimuth angle) between 0° and 360° to calculate and guide user whether to tilt the Smartphone to the left or to the right. Notice that in this guiding to calculate azimuth angle we must incorporate this sensor with the Accelerometer to help correct the raw magnetic measurements using its tilt information.

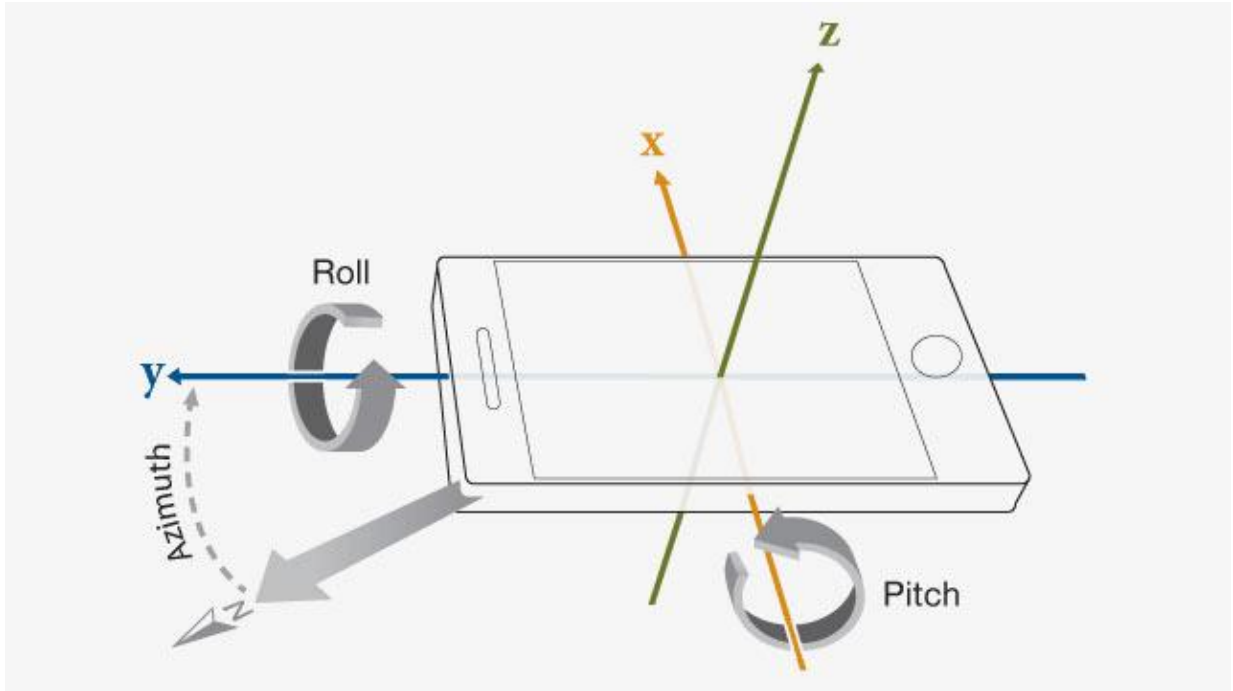


Figure 42 Shows device orientation angles and tilting angle (Azimuth).

8.3. Low Pass filter

A filter is any medium through which signal can pass. A low-pass filter is a filter that passes signals with a frequency lower than a certain frequency, and rejects high frequencies. This filter is used to filter any noise in sensors signal, resulting in a smother signal. [23].

$$y(x) = \alpha * f(x - 1) + (1 - \alpha) * f(x)$$

Equation 9 Mathematical low pass filter euqation, $f(x)$ is the recived frequency, α is the filter size that $0 \leq \alpha \leq 1$.

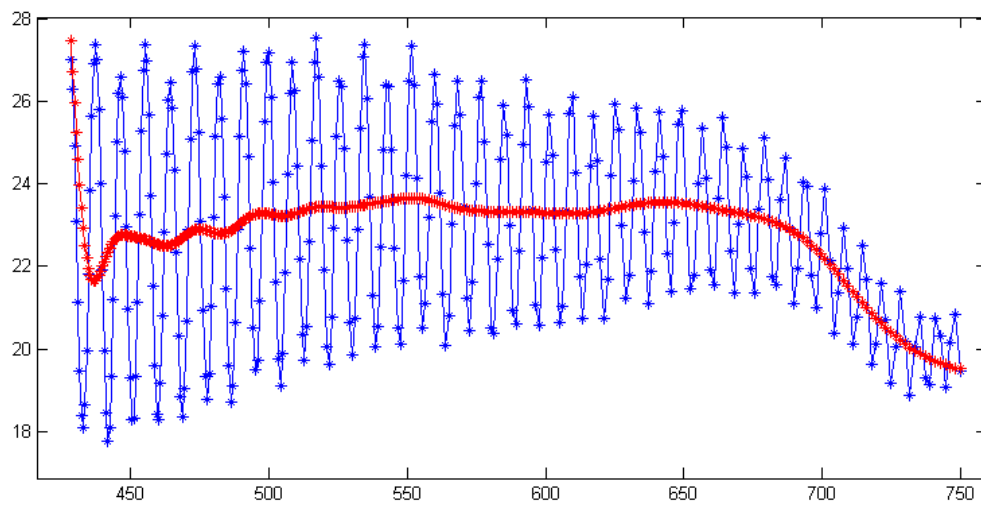


Figure 43 A noisy random signal (blue) and its low pass filter (shown in red).

9. Implementation

We have used Android Studio in our implementation.

9.1. Dealing with Sensors

(onSensorChange) function is called whenever changes occur in sensors data, so we detect which sensor is calling back and then receive its values then we calculate the rotation matrix, and remap coordinating system for better values, then we extract the rotation matrix that is used later.

```
public void onSensorChanged(SensorEvent event) {  
    // getting magnetometer's data  
    if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {  
        if (!mLastMagnetometerSet) {  
            System.arraycopy(event.values, 0, mLastMagn, 0, event.values.length);  
        } else {  
            for(int i = 0; i < 3; i++) {  
                // applying low pass filter for sensor's data  
                mLastMagn[i] = sensor_alpha * mLastMagn[i] + (1.0f - sensor_alpha) * event.values[i];  
            }  
        }  
        mLastMagnetometerSet = true;  
    }  
    // getting accelerometer's data  
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {  
        System.arraycopy(event.values, 0, mLastAccelerometer, 0, event.values.length);  
        mLastAccelerometerSet = true;  
    }  
    // calculation orientation matrix using both data from magnetometer and accelerometer  
    if (mLastAccelerometerSet && mLastMagnetometerSet) {  
        SensorManager.getRotationMatrix(mR, null, mLastAccelerometer, mLastMagn);  
        SensorManager.remapCoordinateSystem(mR, SensorManager.AXIS_X, SensorManager.AXIS_Z, moutR);  
        SensorManager.getOrientation(moutR, mOrientation);  
    }  
}
```

9.2. Guiding Angle

After We receive sensors data we do some calculation to deal with interface and guide user whether to orient or to tilt the device, we save the previous image angle then we compare this angle with the current angle within the rotation value (which is how much the user must rotate the phone to get the next picture), then we detect the aimed direction whether right or left, after that we guide the user to tilt to the right direction using these previous informations. This guiding is explained in the previous pages.

```
if( lastAngle.isSet ) {  
    // Setting current direction  
    if( direction == None ){  
        if( currentAngle > lastAngle + rotationValue ){  
            direction = Right ;  
        }else{  
            if (currentAngle < lastAngle - rotationValue){  
                direction = Left;  
            }  
        }  
    }  
    if( direction == Right ) {  
        if (!((currentAngle > lastAngle + minValue) && (currentAngle < lastAngle + maxValue))) {  
            if ((currentAngle < lastv + minValue) || (currentAngle > lastAngle + minValue + 180)) {  
                canvas.drawBitmap(rightTilt);  
            } else {  
                canvas.drawBitmap(leftTilt);  
            }  
        }  
    }else{  
        // Same work for Left direction  
    }  
}
```

9.3. Dealing with android AsyncTask

AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads. We have used this class with three main process: saving the picture, stitching process (panorama generating) and matching verification process.

```
class (Task Name) extends AsyncTask<parametres > {  
    @Override  
    protected Type onPreExecute() {  
        // called before task starts  
    }  
  
    @Override  
    protected Type doInBackground(Input) {  
        // task work here  
    }  
  
    @Override  
    protected Type onPostExecute(Input) {  
        // called after the task is done  
    }  
}
```

We have used AsyncTask to deal with Stitching and Matching algorithms, without affecting the interface processing.

10.Power points

- Our Application provides useful panorama parameters, that other application doesn't. These parameters could be helpful to users where it separates between a normal user all the way to a professional photographer. This Last one would wait for a minute or two to get a better result.
- There is a lack of panorama applications in Google Play Store. And most Android device's default Panorama camera doesn't provide any configurations, or much guidance. And we think our application will be very successful.

- Autofocus lock gives good results, which we are going to see in the next chapter.
- taking pictures manually allows user to avoid any appearing objects (for example: a person walks into the shooting scene), and makes it easy and comfort for the user in taking pictures, especially with guiding.
- Our application results can be used later in Virtual Reality applications.

11.Conclusion

In this chapter, we have seen our application's design and implementation, design of UML, interfaces, classes and implementation of code, functionalities and calculations.

CHAPTER 4:

TESTS AND RESULTS

In this chapter, we are going to see our application performances and tests.

1. Device Used (Meizu M2 Note)

It's an average Android device by the new company Meizu, released in 2015, it has an Android OS v5.1, an Octa-core 1.3 GHz Cortex-A53 CPU, Mali-T720MP3 GPU, 2 GB of RAM and 16 GB of internal storage. Its main camera is 13 Mega Pixels resolution, that supports auto focus, face detection, HDR and has a dual tone flash.



Figure 44 Represent the user phone in our tests, Meizu M2 Note.

2. Device Performance

Performance Table, we tested the same dataset in some other phones, to compare where to put our test case Meizu M2 Note in performances (performances are related to this device's performance).

Smartphone						Performance
NAME	CPU	GPU	RAM	CAMERA	Android	
HTC ONE M8	Quad-core 2.3GHz	Adreno 330	2GB	Dual 4MP (20MP)	5.1	1.5
MEIZU M2 NOTE	Octa-core 1.3GHz	Mali-T720 MP2	2GB	13MP	5.1	1
SAMSUNG GALAXY A3	Quad-core 1.2GHz	Adreno 306	1.5GB	8MP	5.0	0.9

Table 4 Represents our main test device comparison with other Smartphone

3. Resolution and Time test

Figure 45 shows the time taken to process a three pictures panorama of a different resolution every time, to generate a panorama in High panorama quality mode (please note that even the resolution of input images in these testes varies from low to high, the panorama quality is set to high all the time), to figure out the relation between resolution and our application panorama generating performances.

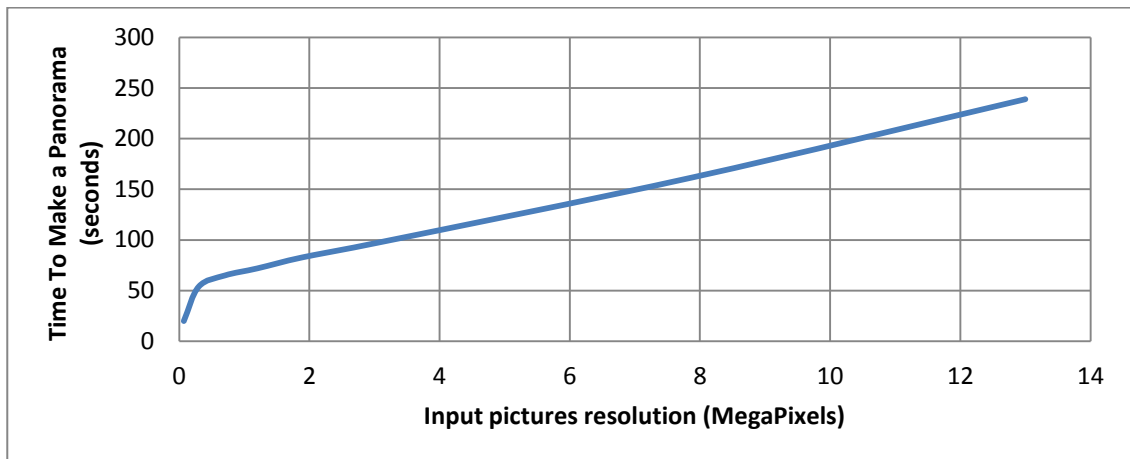


Figure 45 A Graph of the relation between Pictures Resolution and Time to make a panorama.

We note that at 0.5 MP resolution the graph takes a curve into a slow ascending line.

4. Resolution Error Rate

The same Previous test, we tried to extract error rate in the results (notice error extraction is only what We see, there is no tool to extract errors precisely). Error rate is low due to High Quality stitching and blending which has cost some time.

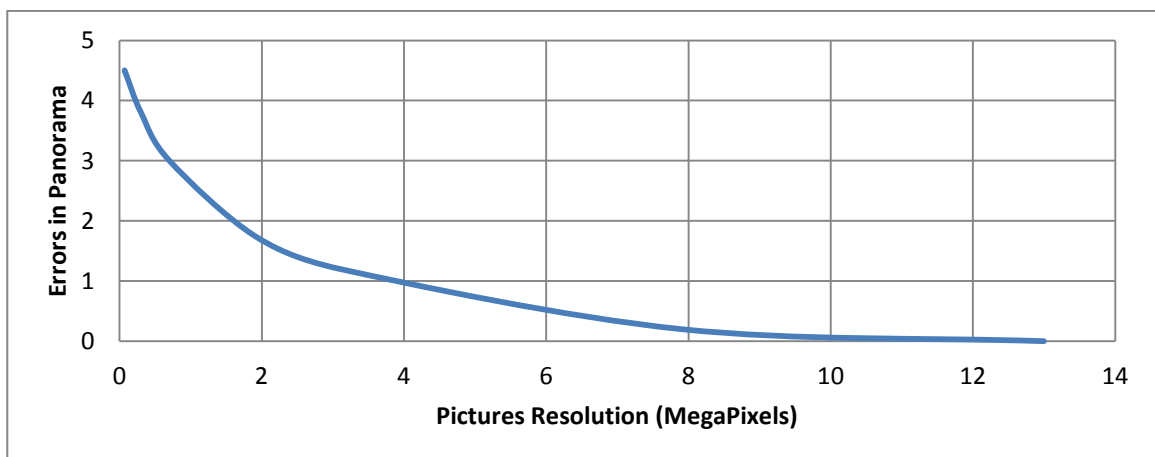


Figure 46 A Graph of Picture Resolution Error Rate (High Quality Panorama Processing).



Figure 47 Represents the detected errors.



Figure 48 How the is the correct image stitching, Extracted from the most better image resolution 13MP and Higher Processing.

5. Panorama Quality Test

Panorama quality are a set of parameters that is used when stitching the images. most important are: Resolution for image registration step (Work mega pixels), Confidence for feature matching step, it presents how much features matched should be according to all extracted features (match confidence), Resolution for stitching step (Seam mega pixels).

Our Panorama Qualities parameters like following:

- **High Quality:** 0.2 MP for Work, 0.2 MP Seam, 0.5 Match Confidence.
- **Medium Quality:** 0.05 MP for Work, 0.05 MP Seam, 0.3 Match Confidence.
- **Low Quality:** 0.05 MP for Work, 0.05 MP Seam, 0.1 Match Confidence.
- **Intel Fast Panorama Parameters:** 0.08 MP for Work, 0.08 MP Seam, 0.1 Match Confidence.

We did another 3 images panorama, with different resolutions and different Panorama qualities, then tested Performance in time.

Panorama Quality Resolution	High Quality	Intel FPP [22]	Medium Quality	Low Quality
Very High	211s	195s	171s	166s
High	144s	130s	109s	106s
Medium	79s	59s	49s	47s
Low	45s	36s	14s	12s
Very Low	16s	X	X	X

Table 5 Represents our tests data of Panorama Quality and Resolution within performance time

The fields marked by X, which is not possible to do image stitching due to lower resolution input images and even lower work resolution. It's not possible to extract features out of very low resolution images.

According to these results the best Panorama quality mode is medium, because it cost less processing time and provides a better result. Also, because Low mode is not that much faster and it give poor image quality.

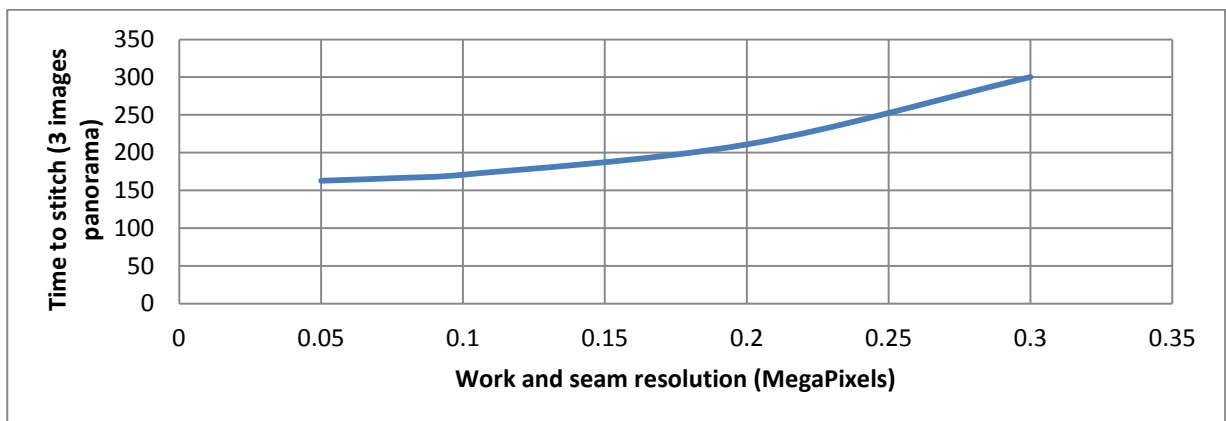


Figure 49 Graph represents work resolution (of the stitching process) and the time to stitch images.

6. Autofocus lock Test

We tested our application's feature autofocus lock, by taking two panoramic picture of a specific scene that had an object somewhere in front, this object will cause focus change resulting to different(anomaly) focus input pictures that doesn't have the same focus range like the other input images, this will cause difficulties in image stitching process and will result errors.



Figure 50 A picture of the computer science department, before we added autofocus lock feature. Notice on the left side of the picture it's a bit blurry.

7. Auto brightness and exposure lock Test

The camera's auto-exposure and auto-brightness also affects the panorama result, especially on environments that different in colors and brightness, this why we lock brightness and exposure on the first image taken. You can notice the left side of the result in Figure 54 and its caption explains more.

8. Manual picture taking vs. automatic picture taking

Other panorama camera applications depend on automatic picture taking, where user's main job is to tilt the phone. In our application we support manual picture taking, for user comfort, we let the user take the picture when they are ready. With the guiding feature, the user can have feedback regarding the alignment of the individual pictures. Once two consecutive pictures are taken, a matching process is started to check if they can be combined together. Manual picture taking doesn't avoids many of the problems associated with automatic picture taking such as appearing objects (for example, a person comes across the camera or an object like a car and when user shakes the phone or tries to hold down the phone for a minute).

9. Resolution Comparison

The application can produce panoramic pictures with high resolutions, that it maximum can get to the sum of input picture resolution, for example the input images in this device can get to 13 mega pixels, the resulting image of 3 of these images will be around 30 mega pixels (A very large image, that calculated: 3 images $\times$ (13MP - 3 correspondence areas size) = 30). Other panorama applications do not allow to change the resolution and it's limited to a specific resolution (for example: 0.9 MP, like seen on Samsung Galaxy A3) so the resolution of the output image will be less than the camera total resolution (around 2.5 MP for Samsung Galaxy A3). Our application takes a lot of time to produce an image, but with a good quality panoramic image.

10. Panorama Camera Applications comparison

Application Feature	Our Application	Meizu M2 Note Application	SAMSUNG Galaxy A3 Application	HTC ONE M8 Application
Input Image Resolution	<u>From Lowest Resolution to same Camera Resolution (13MP in our test case)</u>	1.8 MP	0.9 MP	1.3 MP
Ability to change resolution	<u>Yes</u>	No	No	No
Performance Time	Varies from 15 second to several minutes (in our test case)	<u>Several seconds < 5</u>	Several seconds < 10	Several seconds < 10
Guiding lines (sensor data)	<u>Yes (Roll and pitch)</u>	Yes (Roll only)	No	<u>Yes (Roll and pitch)</u>
Result Quality	<u>Depends on the selected resolution and processing quality</u>	Fair (sum of 1.8MP)	Fair (sum of 0.9 MP)	Good
Configurations	<u>Yes</u>	No	No	No
Tilt Guiding	<u>Yes (Advanced)</u>	Yes	Yes	Yes
Flash light	<u>Yes</u>	No	No	No

Table 6 A comparision between our application and others

These applications are not limited to a developing cause, but the manufacturer chooses always performance for a better user experience, but it is not always the best choice for quality. We give user the freedom to choose between performance and quality.

11. Device Angle effects

Device orientation angle is very important that can affect image stitching process performance and quality. In our case, panorama images should be aligned with the same angle. This will shrink down the size of picture's safe space and cause more stitching errors. This is the main reason why professional panoramic cameras have supports.



Figure 51 effect of camera angle(tilt) on image stitching. The red rectangle represents the safe area of the picture, and the circle shows how much the camera was curved to the front.



Figure 52 More guided panorama image, notice the comers are straight which clarifies that the device was in a good orientation.

12. Some results



Figure 53 A result of the application, panorama image of science of computer faculty. Taken May 2016.



Figure 54 Another result, taken in the university. Taken May 2016.

13. Future goals

For future work, the work presented in this document can be improved by:

- Incorporating sensor's received data into the image alignment and warping process.
- dividing the image stitching process, and start doing image stitching while the user is taking pictures, two by two input images, and then re-do the process for the resulting images.

- Making the guiding features even more dynamic.
- Incorporate camera's HDR to make HDR panorama pictures.

GENERAL CONCLUSION

GENERAL CONCLUSION

In this work, we have gone through the steps of creating an Android application that uses smartphone cameras to produce panoramic images. The key idea is to extract features from common areas of consecutive images to create an image with a wider field of view. The process of combining two consecutive images is called image stitching. We have studied two common feature extraction techniques SIFT and SURF. In our application we have chosen to use SURF features in the process of image stitching, because they are computationally cheaper. Moreover, we have found that SURF features to be better for the purpose of image stitching.

We have also shown the design and implementation of our application using the Android OS SDK and NDK. We have tested our application on different data sets using a variety of handsets with different hardware characteristics. The data sets comprise pictures taken under different conditions such as indoors vs. outdoors, lighting, and the presence moving objects. We have found that our application produces panoramic images with good quality under most of these conditions. However, there are limitations which are mainly due to computation power of the smartphones and the single-threaded implementation of our application. Typically, the time taken by the stitching process grows quickly as the size of the input images grows as well. In general, we foresee that this issue is going to be mitigated as the hardware performance gets better. We also expect that a multi-threaded implementation can help in reducing considerably the processing time taken by the stitching process.

Our implementation provides a convenient user experience through a friendly interface. Typically, the smartphone sensors are used to provide feedback regarding the angle of the camera while taking pictures. The idea is to give a hint to the user to stay as aligned as possible with respect to the previously taken pictures. Additionally, manual picture taking is used throughout the whole process, to reduce the artifacts due to an automatic picture taking mode (for example, dynamic scenes and different lighting conditions). We also gave the user control over the working of the application by making it possible to configure the parameters used in image stitching.

Future work includes exploiting the multi-cores of the on-board processors to speed up the stitching time. Another future work can be using better feature matching techniques to create better panoramic pictures in a shorter time.

REFERENCES

Bibliography:

- [1] N. McCarthy, "Digital Cameras Are Getting Ditched For Smartphones," Shutterfly, 8 Dec 2014. [Online]. Available: <https://www.statista.com/chart/3036/digital-cameras-are-getting-ditched-for-smartphones/>.
- [2] tutorialspoint.com. [Online]. Available: http://www.tutorialspoint.com/android/android_architecture.htm.
- [3] "netmarketshare," Net Applications.com, April 2016. [Online]. Available: <https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1>. [Accessed mai 2016].
- [4] E. Locardi. [Online]. Available: <http://www.blamethemonkey.com/hdr-photography-panorama-tutorial>.
- [5] D. Knight, "Top Panorama Problems and How to Solve Them," 2008. [Online]. Available: <http://www.theperfectpanorama.com/articles/problem-4-stitching.html>.
- [6] "Gobi Tripod BUNDLE," bushman-panoramic, [Online]. Available: <http://bushman-panoramic.com/bundles/gobi-bundle.html>. [Accessed 2016].
- [7] "FACEBOOK SURROUND 360 CAMERA," MMXV UNCRATE, LLC, [Online]. Available: <http://uncrate.com/stuff/facebook-surround-360-camera/>.
- [8] E. HUTCHINGS, "Unlock A Smartphone Just By Looking At It," psfk.com, 2012. [Online]. Available: <http://www.psfk.com/2012/07/face-recognition-iphone-lock.html>. [Accessed 2016].
- [9] L. Bedigian, "5 Of The Coolest Gesture Control Devices," Benzinga, 2014. [Online]. Available: <http://www.benzinga.com/top-stories/14/09/4880572/5-of-the-coolest-gesture-control-devices>. [Accessed 2016].
- [10] "Register an Aerial Photograph to a Digital Orthophoto," mathworks, [Online]. Available: <http://www.mathworks.com/help/images/register-an-aerial-photograph-to-a-digital-orthophoto.html>. [Accessed 2016].
- [11] D. Yining and Z. Tong, "Generating Panorama Photos," Hewlett-Packard Labs, 1501 Page Mill Road, Palo Alto, CA 94304.
- [12] P. Krishna, Stitching of X-ray Images, Box 536 751 21 Uppsala: Uppsala University, November 2012.
- [13] F. M. Schaukowitsch, "An interactive application for camera calibration,image stitching and projective operations," Vienna, 2013.
- [14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91-110, 2004.

- [15] [Online]. Available: <http://www.di.ubi.pt/~agomes/cvm/teoricas/05-edgedetection.pdf>.
- [16] A. SEDAGHAT and H. EBADI, "VERY HIGH RESOLUTION IMAGE MATCHING BASEDON LOCAL FEATURES AND K-MEANS CLUSTERING," no. 30, p. 166–186, June 2015.
- [17] H. Bay, T. Tuytelaars and L. V. Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346-359, 2008.
- [18] E. Oyallon and J. Rabin, "An Analysis of the SURF Method," Image Processing On Line.
- [19] "Introduction to SURF (Speeded-Up Robust Features)," OpenCV, [Online]. Available: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html.
- [20] L. Juan and O. Gwun, "A Comparison of SIFT, PCA-SIFT and SURF," *International Journal of Image Processing (IJIP) Volume(3), Issue(4)*.
- [21] A. Olić, "Introduction to Gradle," [Online]. Available: <https://semaphoreci.com/community/tutorials/introduction-to-gradle>. [Accessed 2016].
- [22] "Fast Panorama Stitching," Intel Corporation, 2014. [Online]. Available: <https://software.intel.com/en-us/file/fast-panorama-stitchingpdf>.
- [23] "Low-pass filter," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Low-pass_filter.
- [24] E. Adel, M. Elmogy and H. Elbakry, "Image Stitching based on Feature Extraction," *International Journal of Computer Applications Volume 99– No.6, August 2014*.
- [25] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant," *IJCV*, 2007.
- [26] P. Mayani, "Introduction to Android Studio," October 2013. [Online]. Available: <https://dzone.com/articles/introduction-android-studio>. [Accessed 2016].

Abstract

In this document, we have discussed the subject of creating an application for smartphones on Android system, that can generate panoramic images (photos with a wide angle of view). We have done some experiments and researches on algorithms and methods of image stitching, and calculation required to do the work. We studied these methods with considering the limitations of smartphone performance. The results of our work are panoramic images with high quality, which consumes some time to be produced and other normal or low quality panoramas that does not require too much time, and this is by providing multiple settings for the user to choose from, whether it was just an ordinary user or a professional photographer.

ملخص

في هذه المذكرة ناقشنا موضوع إنشاء تطبيق للهواتف الذكية على نظام الاندرويد يقوم بعمل صور بانورامية (ذات زاوية رؤية واسعة)، حيث قمنا ببعض التجارب والبحوث حول الخوارزميات وطرق تخطيط الصور والحسابات المطلوبة للقيام بهذا العمل، ثم درسنا ذلك باعتبار محدودية الهواتف الذكية في الأداء، وقمنا بإنتاج صور بانورامية بجودة عالية التي تستهلك الكثير من الوقت ليتم انتاجها وأيضا صور اخرى متوسطة التي لا تأخذ وقت كبيراً، وهذا بتوفير عدة إعدادات للمستخدم ان يختار منها سواء كان مجرد مستخدم عادي أو مصور محترف.

Résumé

Dans ce mémoire de master, nous avons discuté le sujet de la création d'une application pour les Smartphones sur le système Android qui peut générer une des images panoramiques (photos avec un grand angle de vue), Nous avons étudié des algorithmes et des méthodes de couture d'images et le calcul nécessaire pour faire le travail. alors, nous avons étudié avec tenant compte des limites de Smartphone dans la performance, nous avons atteint pour réaliser des images panoramiques de haute qualité, qui consomme beaucoup de temps à produire et à d'autres panoramas de qualité normale, et ceci est en fournissant de multiples paramètres pour l'utilisateur de choisir, si il était juste un utilisateur ordinaire ou un photographe professionnel.