Hélder Filipe Gouveia Gregório

Licenciado em Engenharia Informática

ClearPhoto - Augmented Photography

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Orientador : Nuno Manuel Robalo Correia,
Prof. Catedrático, Universidade Nova de Lisboa

Júri:

Presidente:	Prof. Doutor Luís Manuel Marques da Costa Caires
Arguente:	Prof. Doutor Maria Teresa Caeiro Chambel
Vogal:	Prof. Doutor Nuno Manuel Robalo Correia

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2014

ClearPhoto - Augmented Photography

Aos meus pais e namorada

Acknowledgements

The contributions of several people and entities have made this thesis possible and I would like to extend my appreciation especially to the following.

First and foremost, I would like to express my gratitude to my advisor Prof. Nuno Correia for the opportunity of letting me work and explore this project, and for his guidance and support.

Furthermore, I would like to thank the Departamento de Informática da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (DI - FCT/UNL) for being my second home during these five years and financially support me during the second semester and giving me the opportunity to gain experience as a teaching assistant.

Special thanks to my family, who have supported me through this hard path, in particular my parents and girlfriend, for their immense patience, encouragement and endless love.

To my colleagues, with who I shared the same space, adversities and laughs, day in and day out through these years, and who I am thankful for taking some time off to be my test subjects and answer my survey for evaluating this thesis.

Abstract

The widespread use of mobile devices has made known to the general public new areas that were hitherto confined to specialized devices. In general, the smartphone came to give all users the ability to execute multiple tasks, and among them, take photographs using the integrated cameras.

Although these devices are continuously receiving improved cameras, their manufacturers do not take advantage of their full potential, since the operating systems normally offer simple APIs and applications for shooting. Therefore, taking advantage of this environment for mobile devices, we find ourselves in the best scenario to develop applications that help the user obtaining a good result when shooting.

In an attempt to provide a set of techniques and tools more applied to the task, this dissertation presents, as a contribution, a set of tools for mobile devices that provides information in real-time on the composition of the scene before capturing an image.

Thus, the proposed solution gives support to a user while capturing a scene with a mobile device. The user will be able to receive multiple suggestions on the composition of the scene, which will be based on rules of photography or other useful tools for photographers. The tools include horizon detection and graphical visualization of the color palette presented on the scenario being photographed. These tools were evaluated regarding the mobile device implementation and how users assess their usefulness.

Keywords: Photography, cameras, computational aesthetics, image capture, image quality, mobile devices.

Resumo

A massificação de dispositivos móveis, deu a conhecer ao público em geral novas áreas que estavam confinadas a dispositivos especializados. De uma forma geral, o *smartphone* veio dar a todos os seus utilizadores a capacidade de realizar múltiplas tarefas, e entre elas, fotografar com recurso a câmaras integradas.

Embora estes dispositivos venham com câmaras cada vez melhores, o seu potencial não é totalmente aproveitado uma vez que os sistemas operativos oferecem APIs simplificadas e aplicações fotográficas com capacidades reduzidas ao utilizador. Aproveitando este ambiente propício ao crescimento do computador de bolso, encontramo-nos no melhor cenário para desenvolver aplicações que ajudem o utilizador a obter um bom resultado.

Com o objectivo de fornecer um conjunto de técnicas e ferramentas aplicadas à tarefa em questão, surge esta dissertação, que apresenta como contribuição um conjunto de técnicas para dispositivos móveis capaz de fornecer informações em tempo real sobre o cenário a ser capturado.

Assim, a solução proposta visa o desenvolvimento de uma ferramenta que dá suporte a um utilizador durante a captura de uma cena com um dispositivo móvel. O utilizador será capaz de receber várias sugestões sobre a composição da cena, baseadas em regras de fotografia ou outras ferramentas úteis para os fotógrafos. Esta ferramenta inclui detecção do horizonte e visualização gráfica de uma paleta de cores presentes no cenário. A avaliação foi feita sobre a implementação num dispositivo móvel e sobre a forma como os utilizadores avaliam a sua utilidade.

Palavras-chave: Fotografia, câmaras, estética, captura de imagem, qualidade de imagem, dispositivos móveis.

Contents

List of Figures

List of Tables

1

Introduction

Decades before photography was created La Roche (1729 - 1778) described, in his imaginary tale *Giphantie* the possibility to permanently capture images from nature, on a canvas which had been coated with a sticky substance. Following La Roche prediction, Thomas Wedgwood succeeded in capturing the first silhouettes temporarily, culminating in the first successful picture by Joseph Niépce in 1826 [LR87].

Since that time, photography has evolved from revealing pictures in photographic paper to its digitization. The development of digital cameras and its commercialization through the last 20 years enabled photographers to explore and master new techniques. Aided by the invention of photo editing software and the evolution of the industry, there was a mass popularization of multi-function mobile systems with the capability of taking high quality photos. Taking advantage of these systems and creating software that can facilitate a photographer's job or improve the learning conditions of such a task, is the next logical step to take.

It was predicted that by the end of 2013, 1.4 billion smartphones would be in use, where one in every five people in a world population of 7 billion would own one [Leo13]. While these handheld devices might not have cameras so powerful as the latest digital single-lens reflex (DSLR) cameras, manufacturers are taking a different approach by creating lenses for these devices [Bol13], making them a reliable tool for high quality photography.

This is the perfect scenario for developing applications that explore the world of photography with smartphones and take full advantage of these devices, in an attempt to reduce the gap between amateur photography and professional photography.

1.1 Problem Description and Objectives

Digital photography is tightly related to computational photography. Although the concept is increasingly being adopted, it refers broadly to sensing strategies and algorithmic techniques that enhance or extend the capabilities of digital photography [Sze12], creating a new kind of images that cannot be captured with a traditional camera [Pul+09].

Taking full advantage of multi purpose handheld devices, applications centred in obtaining the best aesthetic results, is something that is not yet well explored. Of the many devices that make part of our daily life, the smartphone might be the most widely disseminated one. Although, for many professionals it might not replace high-end cameras, we can not deny the fact that many smartphone owners use the embedded camera and have taken photography in a different perspective since the device popularization. A major problem is that the manufacturers do not take advantage of the embedded camera capabilities on their default mobile operating systems, due to the lack of control offered through their APIs.

There is a lot of work done in terms of improving a photo by using editing software (e.g, Adobe Photoshop), but the main purpose of these tools is to edit the result after a photo session. For instance, if we imagine that a photographer is trying to take a photo of a mountain scenery. Unless the individual is experienced and takes many photographs of the same scenery in different angles and different focal lengths, for an amateur, the more common option will be to photograph with the mountain centred in the viewfinder, since this is the subject. Completely unaware of the aesthetic difference between a mountain centred and a dislocated one, this kind of photograph would be impossible to edit without reducing its size, or relocate the subject including some degree of distortion.

Even though the APIs lack of support, using this type of systems to reduce the gap in knowledge between professional and amateur photographers, and enriching the users experience by offering options beyond the standard ones is the approach considered in this dissertation.

Therefore, the objective of this thesis is to present a set of tools and techniques where the user can get a better understanding of the scenery and provide ways to obtain a photograph with a better aesthetic result.

1.2 Presented Solution

The aim is to provide a more enriching experience for the smartphone owner that uses the camera application frequently. The final purpose is to obtain the best aesthetic results by offering a capture system able to interpret a scene, in a semi-automatic way.

Before capturing an image, the user will be able to access a set of tools that show information about the scenery through simple visual cues.

This application includes a set of tools to draw grids and suggestions that will aid when taking a photo. While the grids will help in the correct placement of the subject that

is being photographed, the suggestions will consist in a visual cue or a rating calculated from the scene that is being captured. Being colour an important part of an image, some of these suggestions are related to the colourfulness of the scenery, where the user can get a better understanding of the colours being used and which colours are complementary through simplified histograms or color wheels.

Another type of suggestions are related to composition of the image. Some features have been implemented to help a user understand what is the subject in the scenery through object segmentation, the detection of prominent lines and detection of the horizon line for sceneries of landscape or seascape.

The technology used to develop this solution is a smartphone with an Android operating system. Currently we are using the Samsung Galaxy Note with Android 4.1. The system was developed in Java and C++ using both Software Development Kit (SDK) [Goob] and Native Development Kit (NDK) [Gooa] available for Android. Along with the NDK, the OpenCV (Open Source Computer Vision) [Its] library was used for image processing.

1.3 Contributions

The main contributions of this thesis are:

1. **Photography tool for mobile devices**: Introduces a novel approach, incorporating the knowledge of how to obtain pleasant aesthetic results. Since not all users with mobile devices have enough *know-how* when taking photographs, the main objective of this dissertation is to make functionalities and knowledge in photo composition available to the most casual photographer. This results in a thinner gap between amateur and professional photographers;

2. **Library for image processing**: Contribute with a modular library with a set of calls for later use in other applications;

3. **Guidelines for computational photography analysis**: The application will provide guidelines to create the computational equivalent to some of the techniques used in photography, exploring the technological advancements of image processing in mobile devices. The purpose is not to just make them computationally possible but also inspire the implementation of new and improved methods for the field of computational photography in this type of devices.

1.4 Document Organization

This document is structured in five chapters: introduction, related work, system description and functionalities, results and evaluation, and conclusions and future work. The first chapter, **Introduction**, presents an overview of the dissertation, where several issues

are addressed such as context, problem description, proposed solution and the expected contributions. The second chapter, **Related Work**, is dedicated to systems related to this thesis whose features or techniques are relevant to our solution. This chapter focuses on fundamental concepts and processing techniques related to photography, applications for image capturing and processing, systems relevant for aesthetic assessment and computer vision algorithms. The third chapter, **System Description and Functionalities**, describes the implemented solution, first by defining the concept and architecture, and then presenting the functionalities in detail, as well as the technologies used. The fourth chapter, **Results and Evaluation**, describes our testing methods and analyses the results obtained from evaluating our solution. The last chapter, **Conclusions and Future Work**, critiques and comments the work developed in this thesis and possible improvements outlined as future work.

2

Related Work

This chapter will describe concepts, systems, techniques and algorithms related to the theme proposed by this thesis. This chapter is divided into five sections. In the first section we explain some of the fundamental concepts that are directly related to the camera and its properties. In the second section, we enumerate techniques used to produce high quality images obtained from photography. Since there are many mobile applications related to image capturing and processing, the third section presents a summary of some of those applications during and after the capture. The fourth section, refers to the state of the art related to computational aesthetics and applications of related concepts for image evaluation and classification. The last chapter, describes algorithms for extraction of features in an image.

2.1 Fundamental Concepts of Photography

In the world of photography, there are many technical concepts and properties that are fundamental. It is the photographer's job to use those concepts and properties in order to explore creativity and capture the moment with the best possible result. For a better understanding of this proposal, a brief description of some concepts and properties are enumerated in the following sections.

2.1.1 Light

Probably the most fundamental element in photography, capturing the light reflected by the objects is the core in photography. The various colours of the light spectrum are reflected and recorded by the camera sensor, defining an image in a raw format with all

the chrominance information.

In photography, it is necessary to understand light, since there are multiple types of light [San10]. Natural light is easily interpreted as something that emits its own light, and not just reflects it (e.g., the Sun). This light can vary with the seasons, weather conditions and through out the day. Although the source is the same, depending on the season and the time of day, the angle at which the light falls on the subject may vary. Another aspect is the amount of light available that can also vary with the time of day and weather conditions. Another type of light, is the artificial light, which can be generated by the photographer. It can be manipulated at free will including the number of sources, direction and colour of the light.

2.1.2 Exposure

Exposure is the amount of light that reaches the camera sensor and is controlled by choosing the shutter-speed, aperture of the lens and ISO value, although ISO doesn't necessarily affect the amount of light that goes through [Kam12; San10]. All of these variables are independent and the same result can be obtained by different permutations. The correct combination can have an important part on the end result. Each of these variables will be described later.

2.1.3 Shutter

Shutter is a mechanic or electronic component that allows the light to pass for determined period of time, and reach the light-sensitive electronic sensor to capture a permanent image of the scene. The velocity the shutter takes to perform an action is called shutter-speed and can vary from milliseconds to seconds, depending on the technique the photographer intends to use to capture the scenery. Lower shutter-speeds allow to create long exposure images, while faster shutter-speeds tend to avoid shaken or blurred images, allowing perfectly sharp images of objects or people in movement [San10].

2.1.4 Aperture

Aperture is the hole that controls the amount of light that passes and reaches the sensor [Kam12; San10]. It appears represented in a value of f/, which represent a ratio between the aperture and the focal length, and can be called of *f-stop*. The higher the value, the smaller the aperture value is, and this value will depend on what the subject is and what the photographer wants to maintain sharp in the photo. For example, if the aperture is wide open, then the f/ will be smaller and will result in an image sharpen around what the lens is focusing on and blurred on everything else affecting the depth of field.

2.1.5 Depth of Field

Depth of field represents the portion of the image in front of and behind the focused plan that comes with obvious clarity [Kam12; San10]. This effect can vary with the lens aperture (Section 2.1.4). The larger the lens aperture, the smaller the depth of field which will result in a larger f/ value and a greater amount of light that passes through. The difference between a shallow depth of field and a wider depth of field can be viewed in Figure 2.1. The depth of field can be used in a creative manner, leaving to the photographer's criteria, the amount of sharpness she wants from the nearest object to the farthest object.

(a) (b)

Figure 2.1: Difference between a shallow depth of field (a) and a wider depth of field (b) [Kam12].

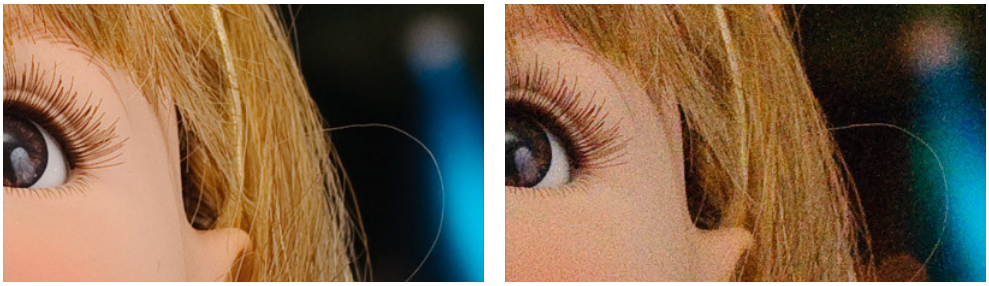
2.1.6 ISO

This is the measure that defines the camera sensor sensitivity to the light [Kam12]. Digital cameras tend to behave better in low light conditions with higher ISO values, which means that for higher ISO values, the camera's sensor becomes more sensible to light rays. In digital cameras and mobile devices, the sensitivity can be adjusted if necessary. However, increasing the camera's sensitivity to the light might ruin a photograph by introducing some digital noise in the image, as shown in Figure 2.2[1]. To reduce this negative effect on the image, the use of high ISO values can be compensated with fast shutter speeds and low aperture values.

2.1.7 White Balance

It is a known fact that the human eye is more sensible to light variations than colour variations, therefore, when we see an object reflecting light, our brain instantly interprets the colour. This means that in areas of different brightness, our eyes adapt and interpret the same colour, although, to the camera they are not equal. Since cameras are not capable of simulating the human brain, that is why white balance is used in photography, in order to match the captured ambience light to what our brain would read [Kam12]. Figure

[1] http://photographylife.com/what-is-iso-in-photography



(a) (b)

Figure 2.2: Difference between an image with ISO value of 200 (a) and 3200 (b).

2.3 illustrates various examples of the same image with different tonalities that can be corrected adjusting the white balance.

(a) Too cold (b) Well balanced (c) Too warm

Figure 2.3: Three examples of white balance applied to a photograph [Kam12].

2.2 Processing Techniques for Photography

To render seascapes showing both the sky and the sea it was a hard task before digital photography, due to luminosity range being too extreme. Photographers overcame these difficulties by exploring concepts such as the ones described in Section 2.1. Exploring such concepts led to a development of new techniques, resulting in photographies with different properties. This chapter will describe some of these techniques and how they can be achieved.

2.2.1 Long-Exposure Photography

Long-exposure (or time-exposure) photography exists since the popularization of photography. In the beginning, a person was obligated to stand completely still in front of a camera so that the final result would be as sharp as possible. With this premise, long-exposure photography is a technique which involves taking a picture with a long shutter-speed [Kam12]. This way the camera sensor will record more light while the

shutter is open. With such long speeds the sensor cannot record moving objects, resulting in perfectly sharp capture of stationary objects and blurring or obscuring of moving elements. This technique is more successful under low light conditions due to the time that the sensor is exposed to light, but this can be suppressed by using special filters for the lenses. By taking so long to close the shutter, the sensor keeps absorbing light creating a brighter photograph, producing a near daytime effect. This technique made easier for professionals to photograph at night, and gave form to new types of photography such as light painting, where a person with a light source can draw paths in the air. Being more sensitive to light, while the shutter is open, the sensor records all the paths drawn resulting in an image where the paths form a continuous line and the person or object moving the light source is obscured, as shown in Figure 2.4[2]. Long-exposure can also be simulated by manually blurring specific areas of a photo, using image editing software.

(a) (b)

Figure 2.4: Examples of Long-Exposure Photography [Kam12].

2.2.2 High Dynamic Range Imaging

Although there is a big improvement in the technologies related to photography, cameras still have a problem of not being able to perceive colours the same way as the human eye. Due to the inability of digital cameras to correctly perceive a scene luminosity, it is possible to incorrectly record colours and lose information in lighter or darker areas accordingly to the exposure. High Dynamic Range Imaging is based on a capture that can represent more accurately a range of intensity levels found in scenes, compensating this problem. In photography, this technique can be achieved by taking multiple Standard Dynamic Range (SDR) photographs of the same scenario with different exposure values that can vary depending on the device. After taking all the samples, the process consists in combining all the raw data of over-exposed and under-exposed areas in one image. By doing this, the image will result in a photograph with a broader tonal range, as shown in Figure 2.5[3].

[2]Source: (a) `http://www.hongkiat.com/blog/light-painting-artworks/`, (b) `http://www.flickr.com/photos/awfulsara/35403447`

[3]Source: `http://www.flickr.com/photos/wetworkphotography/7437783578`

Although cameras already have enough computational power to perform these techniques, Debevec and Malik [DM08] also proposed a method of recovering high dynamic range radiance in photographs taken with conventional image equipment. As expected, multiple photographs are taken with different values of exposure. Using these photographs as samples, they recover the response function of the imaging process using the assumption of reciprocity, which can be defined by a sensor response to the total exposure (i.e. $itensity \times time$ controlled by the aperture and shutter-speed (Section 2.1)). Having obtained a response function, the luminosity value of each pixel is computed using all the available exposures, in which its value is closer to the middle of the response function.

Another approach [VJ11] involves three cameras, side by side, in the same optical axis. Each of these cameras takes a photograph with different exposure values, taking a first picture underexposed, a second picture with the normal exposure, and a third picture overexposed. These three images are overlapped and merged, reconstructing information lost in overexposed and underexposed areas.

(a) Non HDR (b) HDR

Figure 2.5: Example of a picture taken with (a) Standard Dynamic Range versus (b) the same picture with High Dynamic Range.

2.2.3 Panoramic Photography

Panoramic photography is a technique that creates an image with an enlarged field of view which approximates or exceeds the human eye (160º by 75º). Among specialized methods and devices, one of interest is the use of Catadioptric cameras and lenses. These cameras are based on a system of lenses and curved mirrors that allow a field of view of 360º over a single viewpoint, bypassing the need of horizontal panning as it occurs with other methods. Since it uses mirrors and lenses, the light rays bend preventing any kind of distortion or chromatic aberration. Without the need of computation, another advantage is the use of these cameras for video shooting of 360º panoramas. There are on the market some add-on lenses for mobiles devices that make this technique possible, such as GoPano micro (Figure 2.6(a)).

There are also methods to generate panoramas by stitching multiple horizontal images through software. Brown and Lowe [BL07] described an algorithm that would generate a graph to recognize individual panoramas by finding all pairwise image overlaps using feature-based methods. After finding all images with matching features, it would readjust the rotation of the images to generate a panoramic image. This method would be insensitive to the ordering, orientation, scale and illumination of the input images.

Szeliski and Shum [SS97] describe methods to create full view panoramic mosaics. First they describe a method of generation of cylindrical panoramas with a sequence of images taken by a camera mounted on a levelled tripod. This algorithm consists in estimating consecutive horizontal and vertical translations for each image. To recover the translational motion, the incremental translation is estimated by maximizing the corresponding points between them. It would be possible to convert an image to 2D spherical or cylindrical coordinates for a known tilting angle but it would not minimize the error between two images, therefore, this method can only handle the simple case of pure panning motion.

Secondly, Szeliski and Shum [SS97] introduced an algorithm that does not need a set of pure horizontal images. Instead, as long as there is no strong motion between sampled images, there are no constraints on how images are taken. This makes photographs taken by handheld devices without a tripod a reliable source for creating panoramas. According to Szeliski and Shum, the center point of the sampled images can be described in 3D by a set of matrices which correspond to the image plane translation, the focal length scaling and a 3D rotation matrix. After estimating the mean focal length of the images and rotation matrix, they can stitch the images in a 3D dimensional space. Since it is made by stitching multiple images, the final product presents distortions at the north pole. This is because of a necessary warp to cylindrical or spherical coordinates (Figure 2.6(b)) to have a full view of the panorama without using a specialized viewer.

(a) (b)

Figure 2.6: Image of attachable lens for iPhone, GoPano micro (a), and panorama made by the second algorithm described at Szeliski and Shum [SS97] with distortion at the north pole.

2.3 Image Capture and Processing Applications

Since the first attempts to capture a scenery, to its popularization in the XIX century, the world of photography has suffered improvements, that still shock many professionals in the business. Since the upgrade of analogue cameras to the digital world, the use of negatives and dark rooms to new techniques like HDR (High Dynamic Range) imaging, the current market has been increasingly overwhelmed by mobile devices and their ability to easily dethrone today's digital cameras. Proof of this fact is the wide range of applications related to photography available in mobile devices application stores like Play Store and App Store, some with a more professional objective than others. Throughout this chapter it will be discussed some of those applications for image capturing and processing. Along with these applications, research that has been done in this field will also be discussed.

2.3.1 Image Capture Applications

The advancements in mobile devices created a new type of market. Due to this virtual markets available for any Android or iOS user, the number of mobile applications are constantly increasing. Camera related applications are no exception to this rule. In both markets there are many applications fully capable of capturing images that were designed for social networks or include special features. We will start by presenting the default applications in both Android and iOS systems, and other applications found on both of those markets, ending with a discussion comparing all of them.

2.3.1.1 Android and iOS Native Applications

By default, the newest mobile operating systems already have an incorporated application to take photos. For example, on iOS the default application is rather simple. It has very few customization options for a user that has more knowledge in the area, although it is possible to record videos and choose between full screen photos or photos with a squared format, using one of the two cameras available, with or without flash. Besides these, the iOS native application offers a shortcut to access the device's gallery.

On the other hand, Android's native application is a flexible alternative. It offers access to more advanced functionalities in the likeness of today's digital cameras. Some of these options include changing ISO and exposure values, white-balancing, contrasts, and choose the resolution of the final product. It also enables the user to choose the correct capture mode for the moment, e.g., sports mode, indoor or portrait. Android's application adds meta-data tags to the image which may include GPS location and renames the file according to that location. It becomes more user friendly, when it displays a grid on the screen. This grid serves as a guiledine so the user can position the object in the frame. Despite all the options, one of the most useful features is the anti-shake system that applies corrections onto an image to compensate the user's movements.

2.3.1.2 Photoshop Express (iOS)

The tool developed by Adobe [Ado] has a shooting mode with some extra features in comparison with the native application. Having a preview of the image taken is an interesting feature to be used in a more professional context, allowing the user to decide if it is a usable photo before saving it in the gallery. Although in most of the available applications the zoom feature is already a given, in Photoshop Express it can be controlled by an horizontal slider. The fact that it is always visible, the user understands how to make zoom more easily compared with default applications, where the zoom can be done by performing a pinching action on the screen. The pinching action might not be very clear for someone using the application for the first time, therefore, an horizontal slider as the one presented in Photoshop Express might be a good alternative.

2.3.1.3 Photosynth (iOS)

Photosynth, developed by Microsoft [Mic], was created to support a social network centred in creating and sharing panoramic photos. The social features will not be described since they are not the main topic of this thesis. Regarding the image capturing abilities, this application allows the user to create a software generated panoramic image. The device displays a 3-dimensional spherical space that rotates with the user's movement. As soon as the capture starts, Photosynth automatically captures the initial scene and all the adjacent scenes while the user is rotating. After the capture, this application identifies specific features in one photograph and matches them with others previously taken. Photographs are then paired by analysing the position of their matching features. To visualize the panoramic image, the stitched images are displayed in a 3-dimensional spherical space similar to the one presented on the capture display, with the particularity that the user must scroll to see the final result. Outside the application, when previewing the image in the gallery, it presents some deformations due to spherical transformations applied to the sampled photos.

2.3.1.4 Camera FV-5 (Android)

Camera FV-5 [Vaz] is one of the most complete applications for photography in the Play Store. Although it has a screen with many options and information, it is what most resembles to a digital camera display. It offers full control over exposure, ISO and white balance. Exposure can be manually selected by the user or, alternatively, she can choose between modes that automatically determine exposure values based on specific regions of the image displayed in viewfinder. Multiple focusing modes are available, that allow macros, setting the focus to infinity or tapping the display and selecting the object to focus. More related to camera utilities, multiple flash modes are available including a flash mode that fixes red eyes on photos, and other shooting utilities that include a shooting timer, image stabilization and burst mode. For a more inexperienced user, default programs with pre-defined exposure settings can be used. The most interesting trait are

the indicators in the viewfinder that display values of exposure time, aperture, ISO, battery remaining and how many photos are in buffer (Figure 2.7(b)). Camera FV-5 allows control over the available parameters, recreating some photographic techniques. Due to hardware issues, these recreations are the result of software emulation and not from lens adjustments thus reflecting in the quality of image taken.

(a) (b)

Figure 2.7: Camera FV5 interface (a) and indicators (b) of aperture, exposure time, ISO, etc [Vaz].

2.3.1.5 SketchCam

SketchCam [LM07] is a research project that uses a different approach towards mobile devices in photography. With a touch screen, it enables children to capture images by sketching the area of interest on the display. Using this approach, it allows the user to become more selective towards the scenario in front of her. It enables creativity in a way that the user may be able to create different frames for the picture that is being taken. After selecting the point of interest in the view display, it creates an object that can be used for future collages. This may help teaching the basic concepts of composition and photo editing by using a different display.

2.3.1.6 Frankencamera

Although there are many mobile devices with capabilities to take photos, most of them do not take full advantage of the imaging hardware and offer a highly simplified API. The programmer cannot control the camera exposure time or retrieval of raw sensor data. Motivated by these problems, Frankencamera [Ada+10] is an open-source architecture with a custom-built camera based on Linux and gives full control of the hardware to the programmer through C++ language. This architecture consists in an application processor, a set of photographic devices such as flashes and lenses, and one or more image sensors, each with a specialized image processor, forming a tightly coupled pipeline to coordinate all elements. All sensors, devices and parameters that describe the capture and post-processing of a single output image, can be programmed through its API allowing a mechanism to precisely manipulate the hardware state over time.

Being a custom made platform, it brings some advantages towards closed platforms. One of these advantages is the ability to take long-exposure photos without a tripod.

Using an embedded gyroscope, the camera will stream full-resolution raw frames that will be stored, only if their gyroscope tags indicate low motion when the frame was taken. Another useful application is the creation of panoramic photos with extended dynamic range. In most devices, the user has to take various individual photographs and stitch them together on a computer, but with this system it is possible to individually set the exposure time of each shot creating a panorama with extended dynamic range and previewing the result instantly.

2.3.1.7 Camera51

Camera51 is another recent application for the Android operating system which does not only set exposure and white balance automatically, but also aims to help optimize how a shot is framed which has similarities with the objective of this thesis. This application analyzes the scene looking for objects, faces, shapes, lines and other criteria, and suggests the best framing based on composition rules, such as the "Rule of Thirds". It also provides the option to manually select up to three objects in the scene by tapping on them, which will also be used to determine the best framing, focus and exposure. Since all the suggestions are calculated automatically the application has a simple interface (Figure 2.8) which only allows to change between the rear and frontal camera, and change the flash options. It is not clear what kind of algorithms are being used to calculate the best placement. Besides the final result, there is not any kind of visual hint if the best placement is being calculated based on the scenarios composition or colour information.

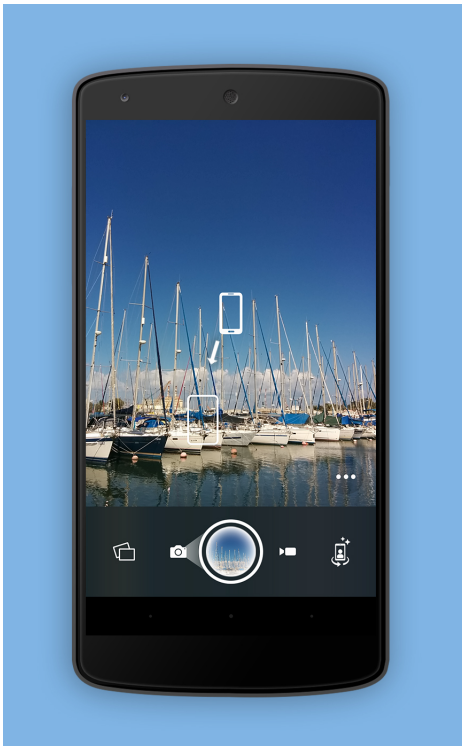


Figure 2.8: Screenshot of Camera51 interface showing the best framing suggestion.

2.3.1.8 Discussion

All commercial applications and research projects share the most basic features that should come embedded in any system capable of taking photos. These features include access to a gallery, control over flash, control between frontal or rear camera, an auxiliary grid and control over zoom. Android applications, comparatively to iOS, offer more control over the device's hardware, such as shooting mode, resolution and image quality, aperture and ISO values. Allowing almost full control of the hardware to the user, is a very important feature that must be taken in consideration when developing an application to take photos. Given this fact, Android became a more reliable platform for users that want to use their mobile device for something more than casual photos.

With some interesting features, Camera FV5 is one of those applications for amateur photographers that presents a similar interface to a digital camera. It enables the possibility of adjusting some photographic parameters and introduces the emulation of photography techniques. Interesting features that should be noted on Photosynth is the way the application handles the creation and preview of panoramas, where it detects and stitches in real-time, a sequence of consecutive photos by matching features on a 3-dimensional spherical space.

As research projects, SketchCam and FrankenCamera can go beyond what is available on standard systems. Although designed for kids, Sketchcam presents a system with a very different way to interact with the user in how she takes a photo. Selecting the point of interest by sketching a continuous path and giving form to different shapes of frames in a display with a live video feed, can be handy when a user only wants to emphasize a region or object in the viewfinder. Frankencamera allows computational photography to go a step further. It is the perfect example of what is possible by taking full advantage of a device capabilities. It allows to take long-exposure photos using the available gyroscope proving that better photos can be taken using available information from multiple sensors.

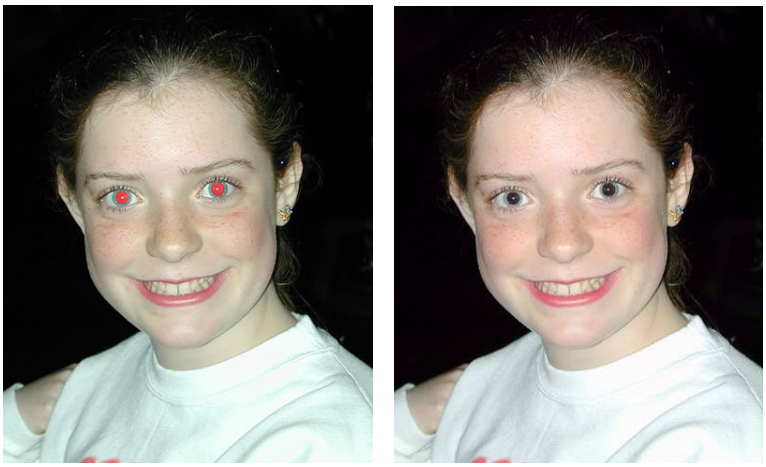
2.3.2 Image Processing Applications

Image processing encompasses the task of altering images, whether they are digital photographs, traditional photochemical photographs, or illustrations. During recent years, image retouching of photographs has gained an important role in the industry and more recently, image editing has became available to anyone. Chosen by some as primary cameras, mobile devices with embedded cameras also have a role to play on this subject. Research has shown that users often do little editing of photos after the shot has been taken [Bre+12], and for that reason mobile devices now offer applications for both capturing and editing images. This section will describe some applications that apply some image editing principles and tools.

2.3.2.1 Android and iOS Native Applications

iOS already brings functionalities for image editing and since it is a system supported by all Apple devices, automatically all these devices have the same basic tools that can be used in photographs. These basic tools include image rotating and image cropping where a user can define which part to crop and select the proportion of the rectangle where the crop will be applied. Since iOS does not have an option to remove the red-eye effect while shooting, a user can later remove it by editing the photo and selecting the eye affected. More related to image effects, iOS offers a set of filters that can be applied to a photo including an option of auto enhance, where it readjusts the images white balance and automatically detects and removes the red-eye effect (Figure 2.9[4]).

Android, on the other hand, comparing with previous versions, has a full set of tools available for image processing. Besides sharing the same options as iOS, it has a a group of advanced adjustments that can be applied to a photo [Sha13]. With these, a user can readjust exposure levels, contrast, hue, etc. In cases where only a specific part of a photo needs to be adjusted, Android allows local fine-tuning where a user applies corrections to multiple selected areas. At any point within the editor, a user can drag down from the top to view the original photo and save the edited image specifying the desired size and quality.



(a) (b)

Figure 2.9: Image (a) before and (b) after applying the auto enhancing tool. It is possible to see that the red-eye remover only darkens the red area and does not take in account the real colour of the eye.

2.3.2.2 PixelNote

PixelNote [Lin+13] is an iPad application for photo editing that works through a multimodal interface combining voice recognition and direct user interaction to manipulate images. It uses natural language to express how to modify an image and sketching to

[4]Source: `http://www.shortcourses.com/images/b4ch6/eerie.jpg`

localize these changes to specific regions. Using the two inputs, a user can select and tag an object with a voice command that can be used for future identification through voice recognition.

The speech recognition technology converts the voice into character strings that PixelNote can process. First the strings pass through a local speech recognition engine that is trained for a specific set of words selected from a user study. When the system encounters words out of the expected vocabulary, the recorded voice data is sent to a remote speech recognition server. When all else fails, PixelNote shows a gallery with options that may be appropriate . Thus, this fallback system allows the user to also learn the vocabulary of the system while editing the image successfully.

2.3.2.3 Discussion

iOS and Android default applications represent two extremes of what is possible in terms of image editing on a mobile device. Other applications such as Photoshop Express (Android and iOS), Photo Editor (Android) or Camera Awesome (iOS) have similar functionalities as Android's default application. Being relatively advanced in photo editing, Android introduces the notion of local fine-tuning to an image where a user can apply different corrections to localized areas of an image. As a research project, PixelNote explores different ways of interacting with mobile devices. Using voice recognition with sketching on localized areas, PixelNote enables object selection and tagging, and correction of specific areas in an image by recognizing specific voice commands. This project shows that is possible to extend an application capabilities for photo editing, exploring different types of interaction and reducing the difficulties of such a task on a small and portable screen.

Both multimodal interfaces and localized corrections, are features that should be taken in consideration when thinking of how to apply effects and interact with small screens as the ones in mobile devices.

2.4 Image Evaluation

To understand aesthetic problems, Hoenig [Hoe05]1 described and formalized a set of theorems and components that could provide a measurable basis for aesthetics. According to the author, in 1933, George David Birkhoff came up with a formula that encapsulated his insights into a aesthetic value, described by $M = Order/Complexity$. This represents the reward one gets, by experiencing orderliness while putting effort in focusing details, giving higher aesthetic value to beauty over complexity. Birkhoff's concept sparked interest of computer scientists in aesthetics, creating the term computational aesthetics. This term is described as a set of computational methods that can make applicable aesthetic decisions in a similar fashion as humans can.

2.4.1 Composition Rules

To obtain aesthetic results, photographers follow certain rules of composition that are the result of past artistic development. These rules are now considered as rules of thumb and serve as guidelines. Following these guidelines helps obtaining pleasant results but they are not absolute. Professional photographers also criticize them and defend that one must know when to break them. Being such a subjective topic, there is no recipe for a good composition and photographers always have the final call when taking a photo. This section describes some of the rules that capture the viewers attention and can be used in computational aesthetics.

2.4.1.1 Colour Balance

Although a good photograph is dependent on the subject that is being captured, colour has a major impact in creating a certain mood and empathy with the viewer.

It is rare for a colour to be isolated in a photo shoot, and depending on the colour palette, a different relation between them will be established. These relations can create similar or different emotions that can be explored in a composition [San10]. A relation created by two complementary colours gives a sensation of balance, but if both colours have different luminosity values, the less luminous colour must be present in a greater amount comparing to its complement (Figure 2.10). To evoke a mood and arouse emotions, each colour has its own meaning that can be interpreted in different ways by different cultures. For the western civilization, yellow symbolizes cheerfulness, joy and optimism, but for the eastern civilization, it is related to the imperial kingdom and symbolizes something sacred. On the other hand, in Egypt, it is a colour for mourning.

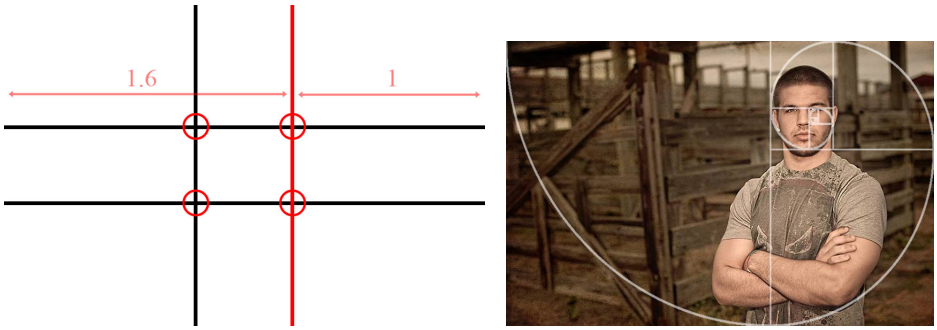
Figure 2.10: Image of two complementary colours balanced in a frame [San10].

2.4.1.2 Rule of the Golden Section and Golden Spiral

There are rules used by many photographers, artists, and architects, considered to obtain very appealing results. The golden section rule is based on the golden ratio. This value can be achieved from a division between two consecutive numbers in a Fibonacci sequence. For example, defining the sequence [8,13,21] as a subsequence of the original Fibonacci, dividing 13 by 8, and 21 by 13 will result in a ratio, that in the limit will be equal to the golden number (i.e. ≈ 1.6180339). This golden number is what defines a golden section [San10].

This section, which is believed to be aesthetically pleasing, consists of a group of rectangles in which the ratio of the longer side to the shorter is equal to the golden ratio. It is possible to draw a logarithmic spiral whose growth factor is equal to this ratio, called the golden spiral, which converges to the smallest rectangle in the section.

Using this golden ratio, one can form a grid dividing the golden section in 1.6 to 1 parts. Applying this division to a golden section, we obtain a grid as in Figure 2.11(a), where the intersection of the lines indicate imaginary points where the main subject should be located. From this point onward, these imaginary points will be treated as power points.



(a) (b)

Figure 2.11: Guidelines formed from dividing the (a) golden section in 1.6 to 1 parts and from (b) the golden spiral.

2.4.1.3 Rule of Thirds

The rule of thirds is based on the golden section [San10]. The rule of thirds consists in dividing the rectangle in nine equal parts. The scene is divided in thirds both horizontally and vertically with power points at the intersections. Being a derivation of the golden section, the fundamental concept of where the object should be, remains the same. The main subject should be positioned in one of the power points and along the the lines, as shown in Figure 2.12.

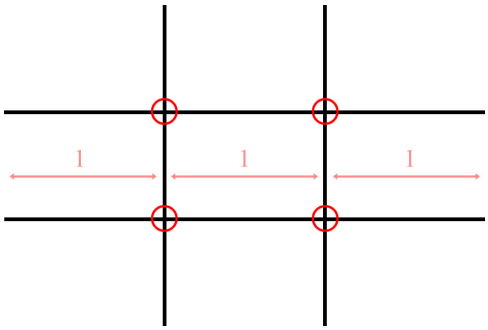


Figure 2.12: Guidelines of the rule of thirds with nine equal rectangles and respective power points at the intersection between horizontal an vertical guidelines.

2.4.1.4 Triangles and Golden Triangles Rule

The most common shape of composition in a portrait is that of a triangle, imagining a portrait with the head being the peak and the width of the body being the base [Cle04]. This enhances the subject and boosts the composition.

The golden triangles rule uses a group of triangles that follow the proportions described by the golden section. Using a golden section, we draw a diagonal line between two corners of the rectangle and connect a perpendicular line to each of the remaining corners. In some cases, this can be simplified to only one perpendicular, having only one power point in the intersection with the diagonal line and a suggestive region in the frame to place the elements [San10].

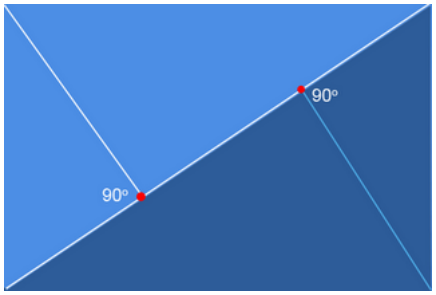


Figure 2.13: Guidelines obtained by the golden triangles rule.

2.4.1.5 Usage of leading lines

Lines can be used implicitly by creating an imaginary line between two subjects in a picture, or explicitly, like the edges of a building. In the perspective of Kamps [Kam12] vertical, horizontal, diagonal or curved lines can be formed of just about anything and have the purpose of leading the viewer to a specific area, giving emphasis to the subject being photographed (Figure 2.14[5]).

Horizontal lines are easier to interpret and give a sensation of stability and safety.

[5]Source: `http://goo.gl/7m0mSS`

Vertical lines can delimit the begin and the end of a scene, and work as an enforcement for horizontal lines. Diagonal lines are responsible for creating perspective in a photo. If the photo does not have a specific subject to photograph, diagonals can direct a viewer to outside of the frame, but on the other hand, the viewer eyes can be imprisoned by using straight angles. Curved lines can have a number of curves, and for that reason, they can give a sensation of movement. Although, depending on the subject and depth of field, the same curves might have different results. It is important to refer that these interpretations can depend on the viewer.

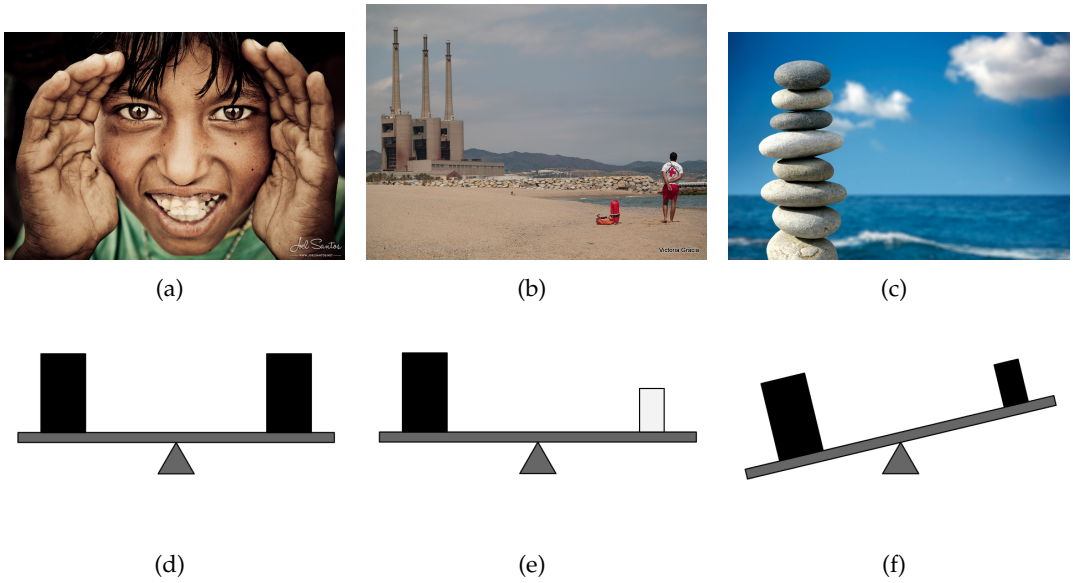
(a) (b)

Figure 2.14: Examples of a curved line (a) that redirects the viewer's eye to the maple tree, and vertical lines (b) that redirects to the subject in the photo.

2.4.1.6 Balance of elements

Unconsciously, the human mind evaluates a photo and checks if there exists any balance or unbalance between the elements [San10]. To judge the balance between the elements of a composition, we must imagine a frame divided by two and a scale that will measure the weight of the left side with the right side (Figure 2.15[6]). When the scale is perfectly balanced which is very common in symmetrical images, it means that both sides have the same weight visually, creating what is called static balance. In dynamic balance, it is possible to create a balance in the image between two imbalanced sides. This can be achieved if one of the sides has a larger element and the remaining side has a smaller but brighter object. The scale does not have to be perfectly balanced and a strong composition can be created with unbalanced sceneries. This way, the viewers attention will lie over the same side, empowering the subject in the frame.

[6]Source: (a) http://joelsantos.net/, (b) http://www.flickr.com/photos/victoriagracia/3869143989/sizes/o/, (c) http://photographyjunction.wordpress.com/2012/07/19/balancing/



(a) (b) (c)

(d) (e) (f)

Figure 2.15: Example images of element balancing and corresponding schematic. (a-d) Static balance normally associated with symmetry. (b-e) Dynamic balance where the left side has a larger subject that is being balanced by a smaller but brighter subject on the right side. (c-f) Unbalanced picture where the subject is positioned on the left side, leaving the right side with a negative space.

2.4.2 Evaluation of Aesthetic Features

Some authors have used some of the rules described in Section 2.4 as a basis for classifier training or score attribution on extracted features. Rules like Rule of Thirds (Section 2.4.1.3) and spatial distribution of a subject have already been topics of research.

Bhattacharya, Sukthankar, and Shah [BSS10] attempted to associate a users' notions of aesthetics by formulating photographic quality assessment measures in a machine learning context. One of these measures consists in the relative foreground position which is defined as the normalized Euclidean distance between the foregrounds center of mass and one of the four *power-points*, that although it works for images with single-subject compositions, it is not viable for landscape or seascape scenarios.

Liu et al. [Liu+10] use a similar approach to generate a score. Instead of just using the Rule of Thirds, the authors also use a saliency map and the prominent lines in a photo. The final score is calculated by Eq. 2.1, where E_{point} represents the sum of the mass of each salient region multiplied by its minimum distance to a *power-point* in the rule of thirds, and E_{line} represents the sum of each saliency of value of a prominent line multiplied by the minimum distance to a rule of thirds line. γ_{point} and γ_{line} are weights given to each of the components. The author that defined the line based in the Rule of Thirds is a better predictor than its point-based counter part, so the weights in Eq. 2.1 are $\gamma_{point} = 1/3$ and $\gamma_{line} = 2/3$.

$$S_{RT} = \gamma_{point} * E_{point} + \gamma_{line} * E_{line} \tag{2.1}$$

Although it is more oriented for assessing human portraits, Khan and Vogel [KV12] present an approach that explores a photos' spatial composition, by computing a score given the location of a face centroid in a specific template (Figure 2.16). This template gives higher scores on lighter areas, and good scores on blur locations around those lighter areas.

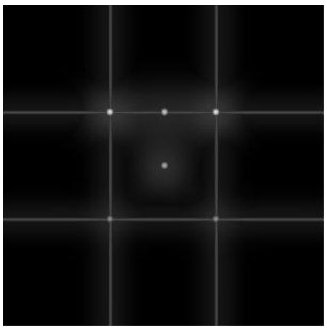


Figure 2.16: Spatial composition template used in [KV12].

Extracting features related to colour have also been explored when trying to identify a photograph with a good aesthetic score.

Khan and Vogel [KV12] proposes a set of features extracted from an area where a face is recognized. This set of features include the illumination of a face by calculating the absolute difference between mean Value (in HSV) of left and right side of face bounding box; the background contrast, calculating absolute difference between mean value of face bounding box and image without face bounding box; and brightness of an image by calculating its mean value.

Other more elaborated methods were presented by Luo, Wang, and Tang [LWT11] exploring the lightning and color arrangement in a photograph. After identifying the subject area, the author explores the colourfulness, clarity contrast and lightning contrast between the subject area and the background.

2.4.3 Practical Application of Aesthetic Evaluation Systems

As mentioned in Section 2.4, computational aesthetics can be described as aesthetic decisions made by computational methods. Increasing aesthetic awareness, researchers developed systems and algorithms that extract and evaluate features of an image based in rules such as the ones described in Section 2.4.1.

Defining aesthetics as a "concern with beauty and art and understanding of beautiful thing", Datta et al. [Dat+06] described a system capable of extracting visual properties and automatically tell the difference between aesthetically pleasing and displeasing images. Based on data extracted from an on-line photo sharing community, a set of images and associated aesthetic ratings given by the community were used to train a classifier.

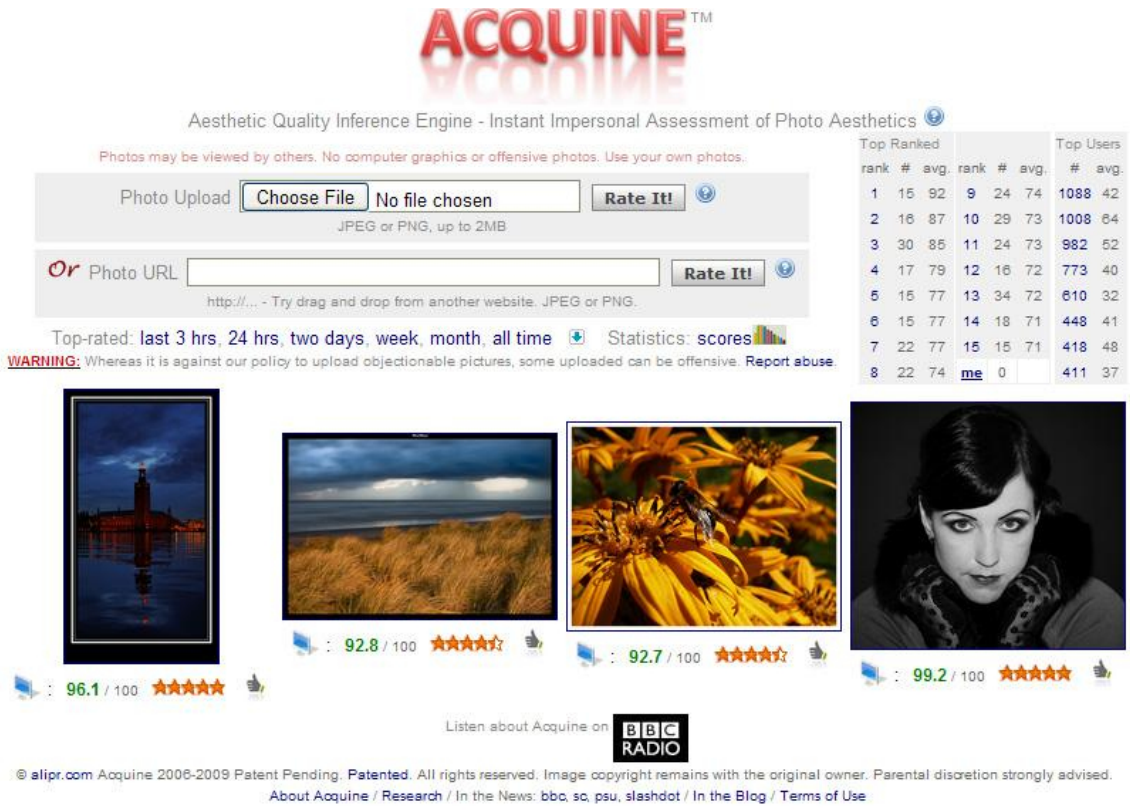
From all image samples, two-dimensional matrices for each of the color components

are obtained. A total of 56 candidate features related to the image properties (e.g. exposure, hue, saturation, size and aspect ratio) and composition (e.g. rule of thirds, use of texture and shape convexity) are extracted from those matrices. These features were chosen to study patterns that could lead to higher or lower aesthetic ratings. By using a segmentation method based on clustering, information relevant to some features was extracted from objects within the photographs. From all the candidate features, 15 visual features were selected and establishing a significant correlation between the visual properties of photographic images and their aesthetics ratings given by the community. The selected features would later be used by the classifier to attribute a rating to an image. Later, a publicly accessible system called *ACQUINE* [DW10] was developed. A user could upload their photographs and have them rated automatically for aesthetic quality. Compromising on a subset of the features previously presented, *ACQUINE* was able to generate quick responses through a simple interface (Figure 2.17(a)) that kept the underlying classifier hidden. A user would then submit an image and wait for the classifiers prediction of aesthetic value. The user could also give a rating on a 7-star scale (Figure 2.17(b)) (similar to Photo.net's rating scale) that would be stored for future validation and improvements on the classifier.

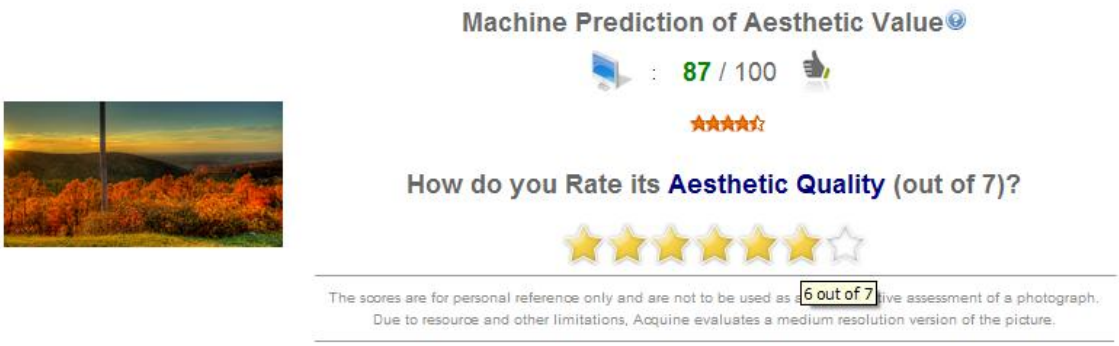
Following the same methodology, Yeh et al. [Yeh+10] described a ranking system. This system listed 1000 ranked photographs ordered from the highest rank to the lowest. The score of each photograph was considered as a linear combination of each feature and its corresponding optimal weighting factor, that was found after the extraction of features. However, since the optimal weights might not combine with the users preferences, it allows them to combine personal taste with a trained model, and rearrange the ordered list of ranked photographs. The weighting adjustments can be feature-based where the user can personally select the weight of each feature (Figure 2.18(a)), or an example-based approach (Figure 2.18(b)), where the user selects a photograph of their liking and the system updates the weighting based on the example chosen. The features chosen to extract are quite similar to the ones used in [Dat+06], but introduce an interesting composition rule. They extract the subject region and assess the simplicity of a photograph by the colour distribution of the remaining region that corresponds to the background.

2.5 Computer Vision Algorithms

Feature detection and matching are an essential component of many computer vision applications. Feature detection refers to information extraction from an interesting part of the image that can be later used for matching and finding a correlation between other images. In this chapter we will describe different types of features that can be used to determine different relations between images.

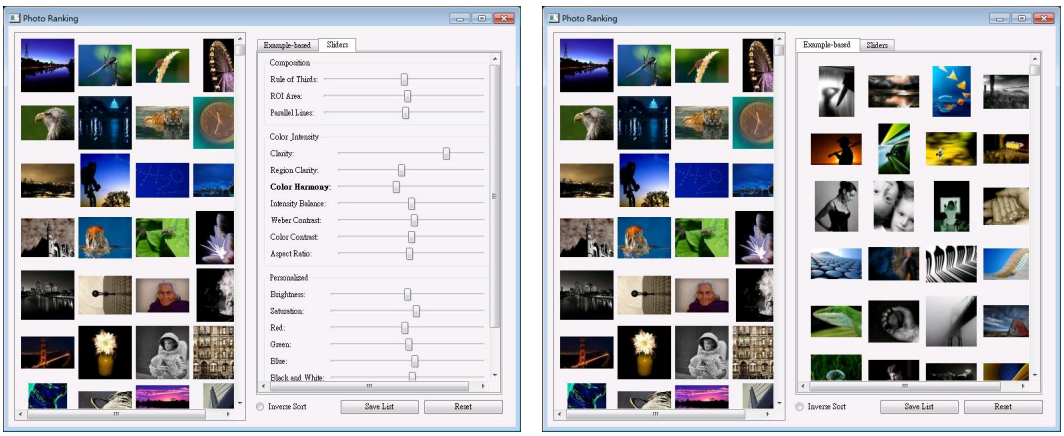


(a)



(b)

Figure 2.17: (a) The frontpage of the ACQUINE system, where the users could upload photos. The list of top users on the right side and photos with high ACQUINE aesthetics scores randomly selected at the bottom. (b) Screenshot of the ratings page after a photo upload. [DW10]



(a) (b)

Figure 2.18: (a) Re-ranking photographs by adjusting the feature weighting and (b) by selecting a few photographs as example [Yeh+10].

2.5.1 Keypoint Detection

Keypoint detection is based on specific locations in the images that can be easily distinguished. These local features are often described by the appearance of patches of pixels surrounding the point location [Sze11]. There are several feature detectors of this type. In this section, we will describe three: SIFT [Low99], SURF [BTVG06], and FAST [RD06].

2.5.1.1 SIFT

Scale Invariant Feature Transform (SIFT) [Low99], is an algorithm used for object recognition. This algorithm uses a set of features that are invariant to image scaling, translation, and rotation, and partially invariant to illumination changes, noise and minor variations in the viewpoint. The algorithm starts by identifying stable points that remain invariant to scale transformations. After extracting the first keypoints, it eliminates the ones that have low contrast and are badly localized alongside an edge, narrowing the total number of keypoints, ensuring that the keypoints are more stable for matching and recognition. Dominant orientations are assigned to each keypoint based on local image gradient direction. A descriptor vector for each of the remaining keypoints is computed in the final stage.

2.5.1.2 SURF

Speeded Up Robust Features (SURF) [BTVG06] is an algorithm that detects points of interest that are scale and rotation invariant while claiming to be faster and more robust than SIFT (Section 2.5.1.1). This algorithm uses a Hessian matrix with the convolution of Gaussian functions for feature detection. The SURF descriptor reproduces the orientation based on a circular region around a point of interest and constructs a square region aligned to the selected orientation extracting the descriptor from it.

2.5.1.3 FAST

Features from Accelerated Segment Test (FAST) [RD06] is a fast algorithm capable of detecting feature points in real-time frame-rate applications. FAST is a corner detector algorithm that detects a candidate point and tests if it is a corner through its adjacent points inside a perimeter of 16 pixels. Although it is faster than both SIFT (Section 2.5.1.1) and SURF (Section 2.5.1.2), FAST is not as robust. The reduced ability to average out noise is why the results are not as good as the other algorithms, but can be used to process image features in real-time.

2.5.2 Colour Features

Colour also provides valuable information for object description and matching. However, there can be large variations in lighting and viewing conditions, complicating the description of images. Therefore, the properties of colour features extracted must be invariant. SIFT (Section 2.5.1.1) has proven to be a very robust and precise feature descriptor, but properties such as light color changes have no effect because the image is converted to gray-scale [SGS08]. Abdel-Hakim and Farag [AHF06] extends this algorithm by proposing Coloured SIFT (CSIFT), building the SIFT descriptor in a colour invariant space.

2.5.3 Texture Features

Texture is considered an important component of human visual perception. Texture can be defined by its coarseness, contrast and direction, and has properties such as periodicity and scale [HR04]. Manjunath and Ma [MM96] presented a method for extraction of texture information for browsing and retrieval of large image data. This method uses a Gabor function that results in the mathematical representation of a sinusoidal wave. This function is then applied multiple times over the image with different parameters, changing both the scale and orientation of the sinusoidal. Working as a filter, the remaining pixels are then eligible as candidate features. The values of scale and orientation applied on the Gabor function to obtain each candidate feature are then used as its feature descriptors.

2.5.4 Discussion

Of the three algorithms described for keypoint feature detection, SIFT is the most robust one. SIFT provides better results and SURF and FAST are faster, sacrificing its robustness. For the context of this dissertation, FAST might be the most appropriate as it claims to be able of detecting features on real-time frame-rate applications.

Because color features are often computed around specific salient points, they cannot be evaluated independently. For a more discriminative power, one must have in consideration the invariance properties of color features. In this section we described methods

that use different color spaces to explore such invariances.

Texture can be defined by its coarseness, contrast and direction, and has properties such as periodicity and scale. In this section we mentioned Gabor functions as one of the methods used to extract texture information from an image.

3

System Description and Features

This chapter describes the most relevant aspects related to the development of the system such as the concept behind the application, the architecture and general structure. It includes a detailed explanation about the functionalities implemented with the proposed technologies and the preliminary user interfaces.

3.1 System Description

Photo aesthetic quality assessment aims to classify the photographs into high or low quality automatically. Tong et al. [Ton+05] attempted to classify photographs taken by professionals or casual users using low-level features derived from computer vision techniques. As already mentioned in Section 2.4.2, techniques for training classifiers to automatically assess the quality of an image, have already been devised. Since mobile devices are constantly used in photography nowadays, it is only logical that improving photography in such devices is the next step to take.

3.1.1 Concept

The development of this set of features started by thinking of useful information that could be extracted from a real-time camera feed. The purpose of extracting this information is to give a user a different insight of the photo that is being taken and help to attain better results without resorting to any photo manipulation techniques or tools. It results in an application that demonstrates a set of tools and techniques that process real-time images and responds with visual cues overlaid on the processed data.

These visual cues have as an objective, to identify key elements in an image, such as

colour information. Colours have already been identified as an important factor in a photograph (Section 2.4.1.1). Therefore, understanding the colour composition and pureness of the colours is useful for a user to study the composition and rearrange the colours before even taking the photo. The same applies when talking about the composition of the elements. As an example, identifying the main object or detecting faces may be useful for testing various compositions in an attempt to find the best one.

With this idea in mind, we gathered a number of features that we thought to give useful information when taking a photograph. Some of these features are related to the colours being displayed. We devised a couple of histograms to visualize the distribution amount of each colour, as well as a saturation detector for detecting when the scenario has colours with a low level of pureness. Other features more related to the composition of the photo were implemented, such as face detection and object detection through segmentation, detection of the horizon line when photographing a seascape or landscape, detection of main lines which have an important role when guiding the viewer in a photo, detection of the line of symmetry and testing the balance between the elements in an image. With the main purpose of obtaining the best aesthetic results without any image editing, we also implemented a couple of metrics regarding the simplicity of an image and number of colours used. Later, these features will be described in detail and discussed regarding the way the result is displayed and the general behaviour of the algorithms.

3.1.2 Architecture

Figure 3.1 illustrates the architecture chosen for this project which is based on mechanisms offered by the Android operating system. The system starts by receiving data from the device's camera, and pass it to the *CameraViewer*. This object, besides controlling which feature is enabled, also controls the camera and serves as a gate since it receives all input data and reliably redistributes to the objects controlling each enabled feature. At this stage, it passes through various *wrapper* objects. Each object is responsible for wrapping the data and send them to the Java Native Interface (*JNI*) where all algorithms are run with the help of *OpenCV*. All the results are then sent back to the object responsible for each feature. The last stage is the Presentation stage where all results of the algorithms are post-processed and the visual information is updated and shown to the user through the viewfinder.

Each feature has its wrapper as well as presentation stage. The Presentation stage was implemented only in Android and all the visual cues work as an overlay over the current camera's live-feed, so that the user can have an immediate response to what she is seeing. Figure 3.2 illustrates a class diagram with the dependencies of each feature, were all that is drawn is an overlay and has its own implementation. Each feature must extend the *Overlay* class, therefore it can be drawn in the viewfinder since the *Overlay* itself extends the default Android class *View*.

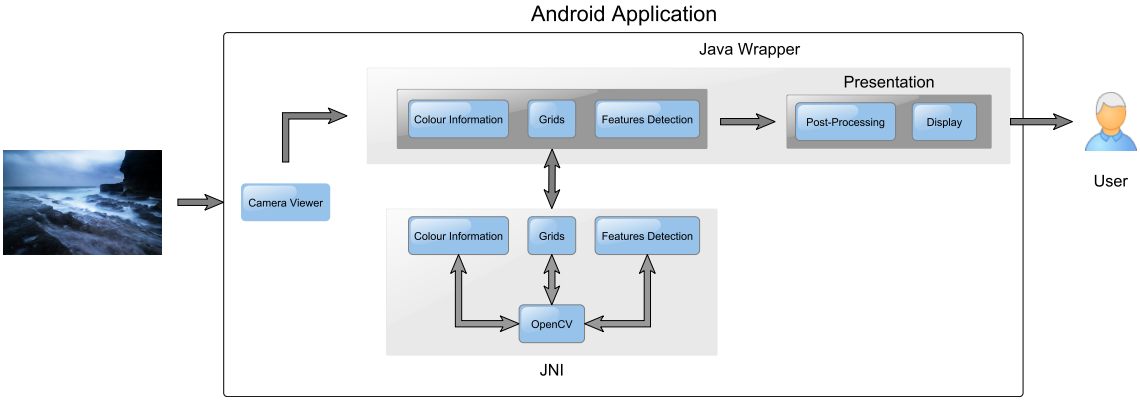


Figure 3.1: Architecture of the system.

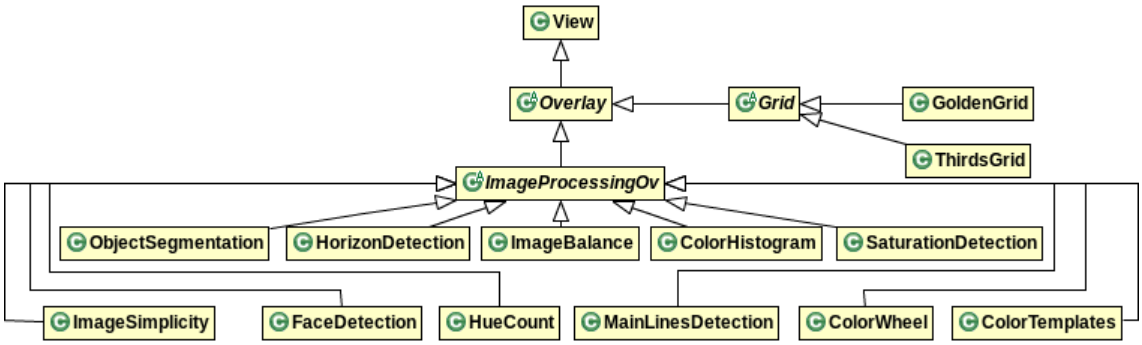


Figure 3.2: Class diagram of the Presentation stage in Figure 3.1.

Still in the presentation stage, to alternate between features we devised a simple interface coupling similar features. Figure 3.3 shows the interface, which consists of a list of the implemented features. To use a feature, the user simply has to tap on a checkbox to start the algorithm. This interface also allows to use multiple features at the same time. Such a simple interface was chosen to keep the viewfinder cleaner and due to the lack of icons that accurately represent what each feature is supposed to do.

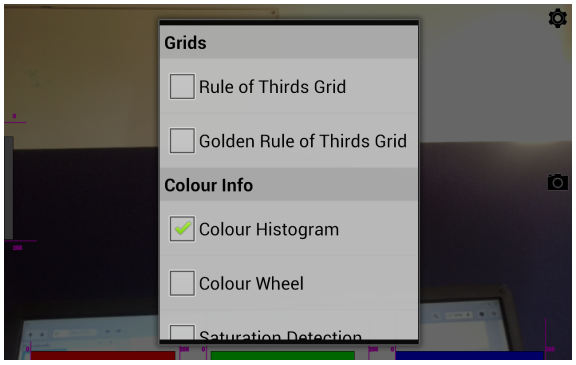


Figure 3.3: Menu to select each feature.

The JNI stage is composed by all the algorithms implemented. This stage is where the Android's Java Virtual Machine calls native applications which can be programmed

for the hardware, or libraries that were written in other languages such as *C* or *C++*. In the context of this thesis, we created our own native library in *C++* complemented by the library *OpenCV* which, by itself, already includes several hundreds of computer vision algorithms. To avoid management of memory addresses between the JNI stage and Android code we used static classes instead of objects. Therefore our native library has a static class for each feature, in resemblance to the class diagram in Figure 3.2.

3.2 Features

This section describes in detail the functionalities and experiments implemented in the proposed system. The set of functionalities implemented were considered to be more interesting to explore in a real-time environment than the usual set of features already presented in regular camera applications or digital cameras.

3.2.1 Colour Histograms and Average Saturation

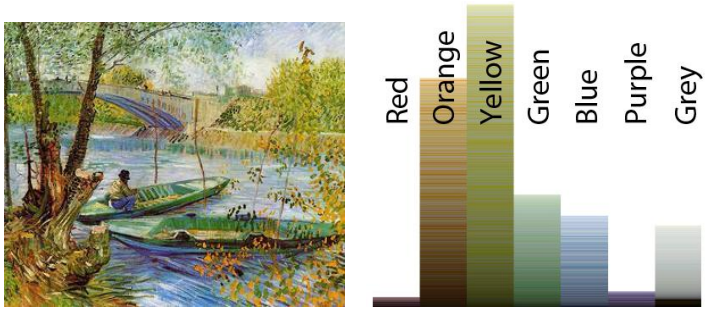
Colour histograms have been used for a long time in image processing and photography to measure the distribution of colours. A colour histogram represents the number of pixels in each of a fixed list of colour ranges, that span the image's colour space, the set of all possible colours.

These can be built in any kind of colour space and have multiple dimensions depending on the number of measurements taken, although it is normally used to count the number of pixels for three-dimensional colour spaces.

The visualization of such information becomes really important when colour is one of the elements in which photography is most focused on. Bertin [Ber83] stated that in the field of information visualization, colour has historically been used as one of the primary visual variables being considered as one of the fundamental building blocks of visualizations today.

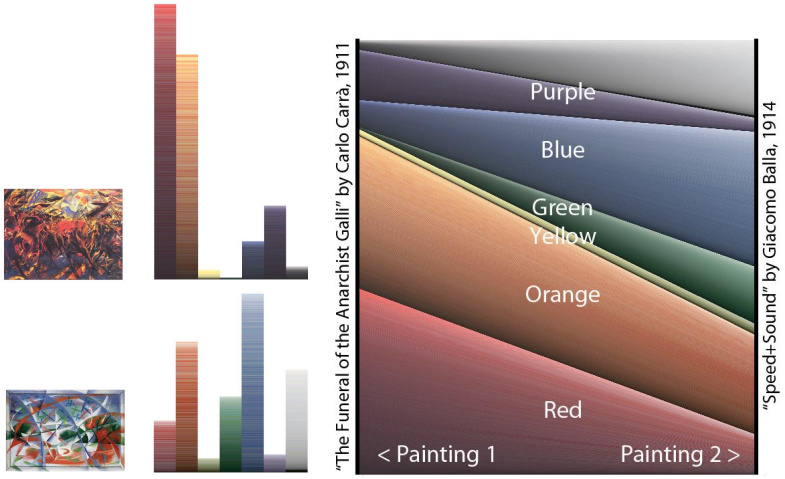
Exploring the importance of colour, Haber, Lynch, and Carpendale [HLC11] presented a method where colour information is interpreted with a variation of a stacked line graph, which give the viewer a sense of the colours and the amount of each colour that make up either an image, or a series of images. The author uses mainly the *hue* channel in the *HSB* colour space. When using one image, it uses a six-colour labelled histogram to define an image as shown in Figure 3.4. However, even though the histogram is simplified to a six colour range, it is difficult to compare the histograms generated by two different images. Figure 3.5 shows the application of a labelled stack line graph when comparing two images.

Another important property is colour saturation which is the pureness of a colour relative to its own brightness. The saturation of a colour is determined by a combination of light intensity and how much it is distributed across the spectrum of different wavelengths. The purest (most saturated) colour is achieved by using just one wavelength at a



(a) (b)

Figure 3.4: "Fishing in Spring, the Pont de Clichy" by Vincent van Gogh (a) and the corresponding labelled linear histogram representation.



(a) (b)

Figure 3.5: Linear histograms representations of "The Funeral of the Anarchist Galli" by Carlo Carrà (top) and "Speed+Sound" by Giacomo Balla (bottom) (a) with a labelled stacked line graph representation comparing the amount of each colour in both paintings (b) [HLC11].

high intensity, such as in laser light. If the intensity drops, then as a result the saturation drops.

Colour saturation influences the vividness of an image as a desaturated image is said to be dull, less colourful or washed out but can also make the impression of being softer.

3.2.1.1 Algorithm Description

As described previously, colour is obviously a very important part in an image. With that being said, we implemented some simple algorithms that compute the amount of colour and saturation that exists in an image.

It was already said that an histogram is an important way to visualize the amount of each colour represented on the scenario. We also computed histograms to give this information. For our first implementation, we generated four histograms for each channel

of the *RGB* colour space and a grayscale version of the input image. Each frame to be processed comes in a packed format and is then converted to a gray image and to a *RGB* image. We then create a full-range histogram for each of the channels.

Knowing the amount of each colour in each one of the channels we can then obtain an interval that contains all the relevant colours. We considered as relevant colours, colours that have an amount of at least 5% of the maximum amount found in the histogram. Given the colour range for each channel, we also calculate the amount of pixels that are near the limits. This amount is calculated by applying a Gaussian function where the peak is the limit in the colour range. This can be expressed by the equation

$$\sum_{i\in[0,50]}^{i} hist(i) + hist(i) * \frac{1}{0.7\sqrt{2\pi}} e^{-i}, \tag{3.1}$$

$$\sum_{j\in[205,255]}^{j} hist(j) + hist(j) * \frac{1}{0.7\sqrt{2\pi}} e^{-j} \tag{3.2}$$

where the Gaussian function has the parameters $\sigma = 0.7, \sigma^2 = 0.5$ and $\mu = 0$. i and j are the colour value in the histogram and $hist(i)/hist(j)$ is the amount of colour on the frame. The result from equations 3.1 and 3.2 are then used to represent visually the amount of pixels near each limit.

In our second implementation we created another histogram but in this case we converted the input image to the *HSV* colour space and performed an histogram over the *Hue* channel. Due to the chosen method to visualize, the *Hue* spectrum was quantized into six bins, which correspond to the primary colours red, green and blue, and its consequent secondary colours. When processing the input image, a vector with a total length equal to the number of bins will maintain the amount of pixels belonging to it. Each bin covered a total of thirty colours in the *Hue* spectrum. The amount percentage for each bin is then calculated and displayed to the user.

As mentioned in the previous section, the colourfulness of a scenario is always important considering that it is what makes an image more vivid. Due to that fact, we also implemented a low saturation detector. This detector works by averaging the *Saturation* channel in the *HSV* colour space and compare its value to a threshold. If the value of saturation of an image is below a threshold of 50, then a suggestion is shown meaning that the use of a monochromatic filter might be useful in that situation. This threshold was defined by comparing the average saturation of 1000 images labelled as low-saturation images by the Flickr community.

3.2.1.2 Interface Display

In the previous section we described the algorithms for a saturation detector and two ways to calculate the colours presented in an image. For the later, it becomes even more important to correctly visualize that information.

For our first implementation which involved the calculation of three histograms based in the *RGB* colour space and one with the image converted to gray scale, we tried to represent this information by using less space than a conventional histogram and slightly simplifying the amount of information. As shown in Figure 3.6, our histograms are visualized by a bar with a dynamic size within certain limits which define the total range of colours within each channel. The purpose of this representation is for the user to have a perception of the range of colours being used in each channel. The size of the bar is dynamic since it is defined by the first and last relevant colours found in a channel.

As mentioned before, this representation also shows the amount of pixels found near the boundaries of the colour range of each channel. This is represented by a line that grows vertically depending on the amount calculated by the equations 3.1 and 3.2.

(a) (b)

Figure 3.6: Visual cue chosen for the first implementation of the histogram being applied to a real-time scenario (a) and a reconstruction of the histogram made for the green channel that indicates de colour range in use and the amount of pixels near the limits (b).

For the second implementation of the histogram, we chose to represent the *Hue* concentration of each bin through a hexagon that resembles a colour wheel divided into six colours. Each section of the hexagon is filled with the percentage amount of the respective colour presented in the input image, as shown in the Figure 3.7.

(a) (b)

Figure 3.7: Visual cue chosen for the second implementation of the histogram being applied to a real-time scenario (a) and the resulting histogram isolated (b).

For the saturation detection, we display a thumbnail with the real-time image converted into grayscale on the bottom-right corner of the viewfinder. This is an indicator that the usage of a monochromatic filter might be useful in that situation (Figure 3.8).

Figure 3.8: Frame detected as containing low saturation. Indicator is shown on the bottom right corner, which is the input frame converted into grayscale and in a reduced scale.

3.2.1.3 Discussion

After implementing these features, we could conclude a couple of things. Starting by our first implementation of a histogram, the first noticeable problem is the fact that the histogram is made for the *RGB* colour space. Although it is the most used colour space in image manipulation applications and photography systems, the conversion of real-world colours to *RGB* is not direct.

With the representation we created a one-dimensional histogram that omitted the amount of each colour and showed the range of colours being used in the image. This range is determined by the first and last colours found as relevant with the purpose to show the spread of tones across a channel. As a general rule, in most cases, a well balanced shot has a nice spread of tones which peak somewhere around the middle and taper off around the edges. Ignoring the peak around the middle and stabilization around the edges, we think this representation successfully serves the purpose giving an idea of the colours being used. The problem with this representation is that it does not contemplate any flaws in the middle of the histogram. This means that if it shows that a range of colours is being used, within that range, there is a possibility that some of the colours were not found in the image. Figure 3.9 illustrates this example. Considering that the blue channel of an image as the histogram shown in Figure 3.9(a), there is a valley between two peaks that contains colours that were not used. Figure 3.9(b) displays our representation based on that histogram, where we can see that the concavity in the original histogram and the colours that were not used in that concavity, are shown as a part of the range of colours detected.

Our second implementation uses an hexagon in resemblance of a colour wheel, it is easier to convert real-world colours to the *HSV* colour space. Although it is easier to perceive the colours being used, it does not have into account the saturation or brightness

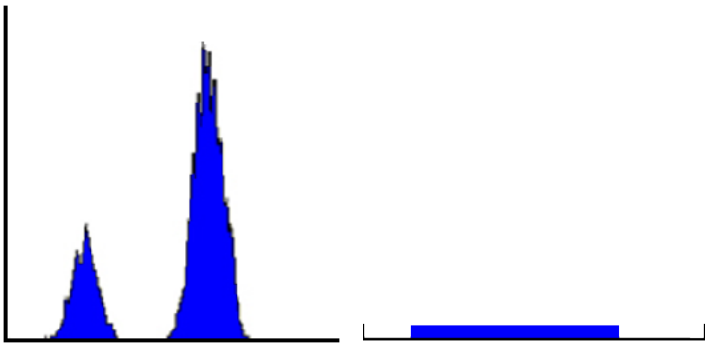


Figure 3.9: Illustrative figure of the problem found in our histogram representation (b) compared to the corresponding regular histogram (a).

of each colour. The main problem in this representation is the amount of percentages displayed. As it is, each part of the hexagon will be fully filled when all the *Hue* values in the converted image are within the same range, which results in a scaling problem as this rarely happens. Each processed frame normally contains at least a small portion of all the colours presented in the wheel, meaning that with the representation chosen, small amounts of each section will be filled making them almost irrelevant. Even with this problem, the main purpose of this representation is to visualize the colours that have a higher presence in a scenario and what complementary colour should be used to provide a colour balance (Section 2.4.1.1).

As for our saturation detection method, it is not very complex, however the threshold chosen to classify an input frame as low saturated or not, was selected by averaging the saturation of 1000 images labelled as *low saturation* by the Flickr's community. This being said the threshold was calculated over a set of static images which may have been digitally manipulated. This means that in a real-time scenario this threshold might not be the most appropriate as it is difficult to find scenarios to test and calculate a more adequate threshold. It would be necessary to find a large amount of real scenarios with low saturated colours and manually filter them.

3.2.2 Colour Templates and Hue Counting

As said before, much of what viewers perceive and feel from a photo is through colours (Section 2.4.1.1). Although colour perception is culture-related and depends on the context, colour science studies show that the influence on human emotions from a certain color or color combination is usually stable under different cultural backgrounds [Man07]. Professionals follow certain rules of color composition and scene composition to produce aesthetically pleasing photographs. For example, photographers focus on artistic color combination and properly put colour accents to create unique compositions and to invoke certain feelings among the viewers of their artworks.

Luo and Tang explored the relation between colours [LT08] in a feature to measure

the color harmony of a photo in terms of learning the color combinations (coexistence of two colours in the photo) from a training dataset, to later determine if a photo has a high or low quality. For each photo in the dataset, a 50-bin histogram is generated for each channel in the *HSV* colour space. The value of the color combination between hue i and hue j is defined as $H_{hue}(i) + H_{hue}(j)$, with a similar definition for the saturation and value channel. A histogram of high (low) quality training photos is generated by the formulas

$$H_{high,hue}(i,j) = Avg(H_{high,hue}(i) + H_{high,hue}(j)), \tag{3.3}$$

$$H_{low,hue}(i,j) = Avg(H_{low,hue}(i) + H_{low,hue}(j)), \tag{3.4}$$

with a similar process for combinations of $H_{high,sat}(i,j)$, $H_{low,sat}(i,j)$, $H_{high,sat}(i,j)$ and $H_{low,sat}(i,j)$. To measure whether a photo is high or low quality based on the its colour combinations, the classifier is trained with the formula

$$f_h = Hue_s * Sat_s * Bri_s, \tag{3.5}$$

where $Hue_s = \frac{Hue_{high}}{Hue_{low}}$, $Sat_s = \frac{Sat_{high}}{Sat_{low}}$ and $Val_s = \frac{Val_{high}}{Val_{low}}$. Although in [LT08], the purpose of this feature is to train a classifier to identify the quality of a photo, it can also be used to query a database and obtain a set of images similar to the one being evaluated. Since our purpose with this thesis is to give useful information in real-time to the photographer, we explored the colour harmonization using the *Hue* channel. Although we opted to not use any classifiers, we also implemented a score to each frame based on its hue count.

3.2.2.1 Algorithm Description

As mentioned before, we implemented a simple scoring system for each input frame and an algorithm that detects the harmonic colour template being used on an input frame.

As for our scoring method, we implemented a feature described by Ke, Tang, and Jing [KTJ06], where the purpose is to measure the colour simplicity. One of the main differences between professional photos and snapshots is that photos taken professionally are more colourful and vibrant depending on its brightness and saturation levels, while using a less number of unique hues. The first step in hue count is the conversion of the input frame to the *HSV* colour space. After the colour space conversion, the *Hue* channel is filtered by selecting pixels that have a brightness value in the range $[0.15, 0.85]$ and saturation higher than 0.2 because the hue calculation would be inaccurate if otherwise. The next step is to calculate a 20-bin histogram H with the hue values that passed the filter. After computing the histogram we select the number of bins N that satisfy the equation

$$N = \{i | H(i) > \alpha m\}, \tag{3.6}$$

where m is the maximum value of the histogram and α controls the noise sensitivity of the hue count. The author used $\alpha = 0.05$ as it produced good results for their training set. The final score is calculated by the equation

$$score = 20 - N. \tag{3.7}$$

Our harmonic colour template detector is based in a method described in [CO+06]. In the original article, the author uses a predetermined set of templates such as the ones in Figure 3.10, which can also be perceived as the colour distribution of the input image. The harmonic templates may consist of shades of the same colour (type i, V, T), complementary colours (type I, Y, X) or more complex combinations (type L). Although not used, the N-type template is applied to grayscale images. In our implementation we focused on templates for slight variations of the same colour and for complementary colours.

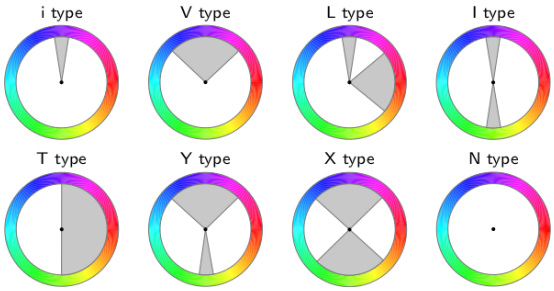


Figure 3.10: Harmonic templates on the hue wheel. Colours that fall into gray areas are considered to be harmonic. Size and rotation of the gray areas may vary.

The original article uses a hue histogram and tries to match the histogram with each of the templates in its various rotations. This method would prove to be computationally heavy on a mobile device in a real-time scenario, therefore, we opted to simplify the histogram and the template calculation.

We started by converting the input image into the *HSV* colour space and manipulate the *Hue* channel. For our implementation, we used a wheel divided into twelve regions instead of using a full-colour wheel which enabled us to create a 12-bin histogram of quantized hues ordered by amount. The purpose of having an ordered histogram, is to obtain the first two main hues present in the image. Knowing the two main colours and its bins, we can map each bin into one region of the 360 degree wheel. This mapping is what will allow us to determine if a image fits into a monochromatic template or a complementary template, by calculating the degree difference between the bins of the two main colours. If the difference between the two main colours is between 0º and 90º we classified the frame as monochromatic, or as complementary if the difference is between 150º and 210º.

After classifying an image as having a monochromatic or complementary template,

starting in the most relevant bins, we verify if the adjacent bins have an occupation percentage in the image of at least 40% of the starting bin. This will allow us to know the size of the template. Starting in a bin which is mapped into one region in the twelve-region colour wheel, we defined that the maximum size for a monochromatic template or one of the colours in a complementary template, is of three regions.

Since the scoring described before is based on a 20-bin histogram, we decided to modify it and perform the scoring based on a 12-bin histogram. This way, both the scoring and the template detection are in accordance with one another.

3.2.2.2 Interface Display

For this feature, we chose to show the colour wheel divided into twelve regions. This would prove to simplify both the implementation of the algorithm and display of the templates. Similarly to the original templates (Figure 3.10) the gray areas indicate the main colours present in the image and its neighbours that were considered relevant. As for the scoring method, we took the most simple approach and decided to show the scoring result since both these methods are in accordance with one another.

3.2.2.3 Discussion

There are no major problems with each of these methods. However the most relevant issue, might be the size of templates. As mentioned before, starting in a bin we verify if the adjacent bins occupy a region that is at least larger than 40% of the starting point, where the threshold was chosen experimentally with a dataset of static photographic images. The problem with choosing a threshold with a given dataset, is that the threshold might not work perfectly in a real-time scenario since the chance of getting snapshots while receiving real-time information is way higher then getting a frame that is considered to have aesthetic value.

3.2.3 Face Detection and Composition Guidelines

Since long, the detection and recognition of faces are areas of interest in the domain of computer vision. Such technology has been used in biometric identification, video-conferences and query systems. Mobile devices are also able to find and recognize a face. Android provides a simple way to identify the faces in a bitmap image, with each face containing all the basic location information. Besides faces, current Android cameras already offer an API for detection of eyes, mouth and most recent versions are able to recognize faces to unlock such devices.

The fact that we can easily detect faces, enables us to explore some rules used in photography. One of these rules is the Rule of Thirds explained in Section 2.4.1.3 which states that for an image to be visually interesting, the main focus of the image needs to lie along one of the lines marked in thirds, or the intersection points between those lines [Kam12].

Other rule is the Rule of Odds which states that images are more visually appealing when there is an odd number of subjects. For example, if we are going to place more than one person in a photograph, we should use 3 or 5 or 7 [Kam12]. Studies have shown that people are actually more at ease when viewing images with an odd number of subjects. This rule can also be boosted by using triangles, since a triangle is one of the strongest compositional shapes, as it can add a sense of visual unity. In essence, a triangle is a closed curve incorporating at least one diagonal. Since the curve is closed, it won't lead the eye outside of the frame. A single triangle in the middle of the frame can lead to a somewhat static composition, but triangular composition can be found in many famous works of art.

By combining these rules with a facial detection system, we devised a feature that could be helpful in respecting these essential guidelines in photography composition.

3.2.3.1 Algorithm Description

For this feature, we tried to mix the rules previously stated with face detection in a real-time environment. Although it can be used separately, we implemented the two simple grids that follow the Rule of Thirds (Section 2.4.1.3) and the Rule of the Golden Section (Section 2.4.1.2) that uses the golden ratio.

As far as implementing these grids, there was no difficulty since it was only needed to obtain the device screen size and generate four lines for each rule, where these lines would divide the screen in 9 equal parts for the Rule of Thirds or in sections with a proportion of 1.6:1 as shown in Figure 2.11(a).

These guidelines would then be useful to test different compositions with faces detected. As for detection of faces, we opted by using the mechanisms available in *OpenCV* instead of using the ones available in the Android API. This is due to the fact that our objective was of creating a library to run only in JNI. If we opted by using the Android API, then it would become specific of the application instead of being portable code for other applications. With the *OpenCV* library we used the Haar feature-based classifier to detect faces [VJ01]. This method needs a large set of positive and negative images to train the classifier. This train is based on the extraction of Haar features, as the ones shown in Figure 3.11. After training a classifier, the implemented algorithm uses the concept of Cascade of Classifiers where a total of 6000 features are divided by groups of features, and for a small window in an image the first group is tested. If that region is a non-face region, it is forever discarded and passes to the next group of features that can then concentrate on testing the remaining regions where there is a probability of finding a face. The regions where the faces are found, are then returned and used by us.

Having detected the faces position and the lines for the grids, we can then calculate the distance between the center of a face region and the nearest intersection line. The intersection with the smallest distance, is the most viable placement for the subject in accordance to the Rule of Thirds.

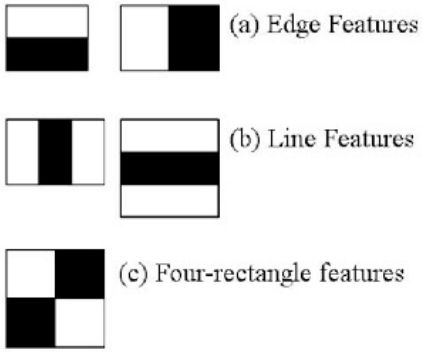


Figure 3.11: Type of convolution kernels used to extract features [Its].

3.2.3.2 Interface Display

We started by describing the Rule of Thirds and Rule of Golden Section that are viewed as a simple grid over the viewfinder. We respected the proportions established by both rules, as illustrated by Figure 3.12(a) and Figure 3.12(b). As for the detection of faces, we took the most common approach. Since we know the position of the regions that contain faces, we opted to draw a bounding rectangle around that regions (Figure 3.12(c), 3.12(c)).

After detecting faces we test to see if there are three faces detected. This is so that we can apply the concept of the Rule of Odds with the detected faces. If there are three faces then we can also try and boost the composition by linking their positions and form a triangle. We display the connection between the faces by drawing lines between them.

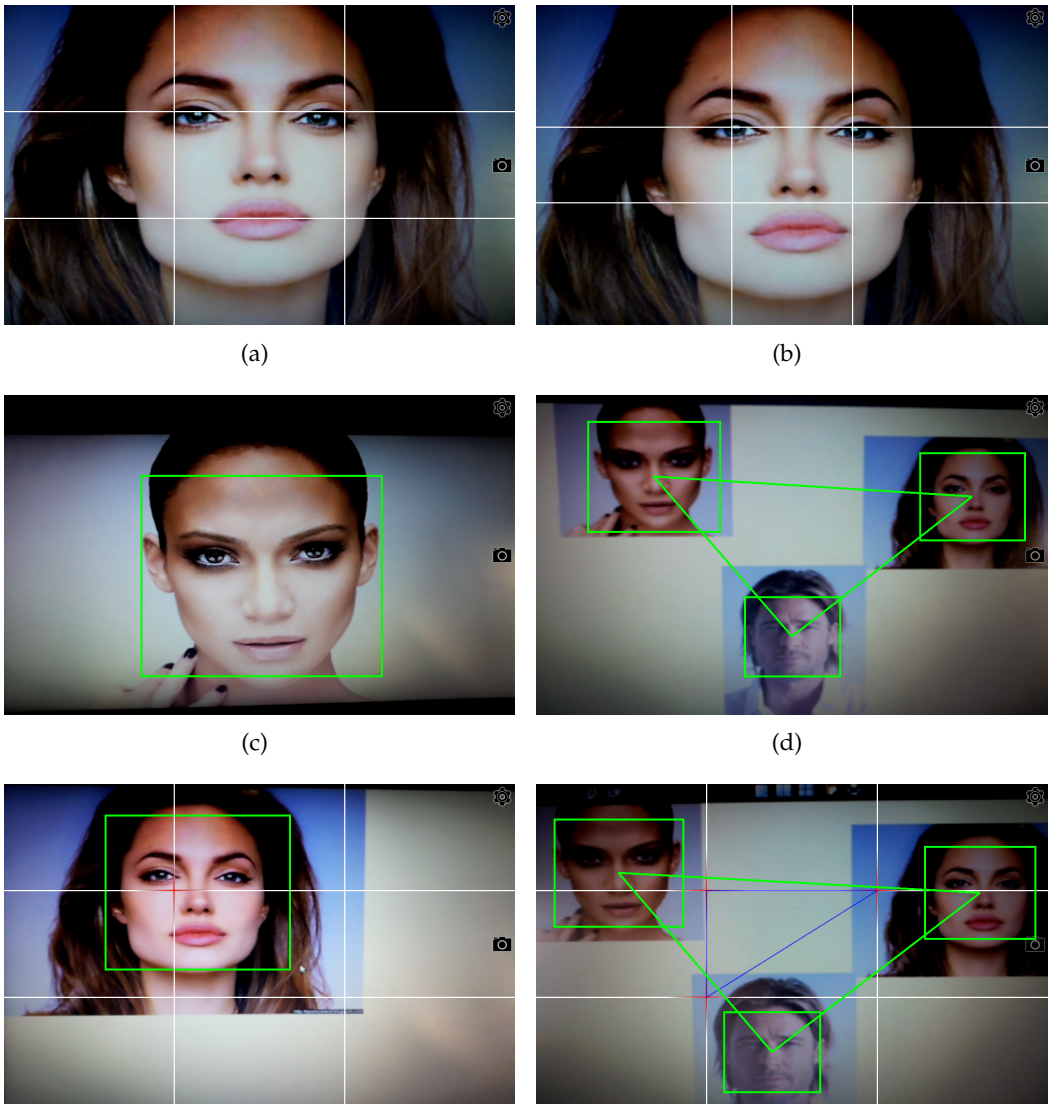
Figure 3.12(d) illustrates this triangle formed by the detected faces as well as the suggestions. Each of the intersection points on both grids highlights when it is a suggested point for a face. To complement the triangle composition when three faces are detected, we also suggest a triangular composition with the nearest intersection points of the grid in cyan blue as shown in Figure 3.12(e) and Figure 3.12(f).

3.2.3.3 Discussion

Due to the simple implementation, there are not any particular problems. The face detection might be the most prone to problems part. As referred before, the face detection uses a trained classifier and we used a default classifier for frontal face recognition provided by the *OpenCV* library. Therefore, this classifier will not detect faces in profile.

Although the user can choose which feature to display at a time, when they are all activated, this many visual cues are quite distracting and occupy a lot of space in the interface.

However, as a proof of concept these visual cues do work but it is important to keep in mind that the positions suggested, are simple suggestions, as it is not a recipe for obtaining aesthetic results. The user must always have the final word for the placement of the detected faces, on the importance of having odd number of elements and on the



(a) (b)

(c) (d)

(e) (f)

Figure 3.12: Multiple visual cues used to implement these features, such as helping grid (a,b), face detection (c,d) and a mixture of both (e,f), tested when detecting one or three faces.

exploration of triangular compositions.

3.2.4 Object Segmentation

Unless it is a photography of a seascape or a landscape, it is normal for a photo to have a relevant subject. With this in mind, segmentation of a subject in multiple scenarios might be useful to attain a better composition.

Segmentation of an object in a photo, is a topic that has already been widely researched. One of the simplest ways of extracting information about the object in a real-time scenario is to have information about its background beforehand and subtract it.

Yang et al. [Yan+04] describe a system for security cameras, able to recognize and track a moving object. This is possible once the system is collecting information about the background as time passes. Using this information, the system can subtract any object that is considered strange, and start the tracking. This method has the advantage of being fast and discards the possibility of training a classifier for object identification. Butler, Sridharan, and Bove Jr describes a similar approach [BSBJ03], where the key is to learn the background and generate a model of it by representing each pixel in the frame by a group of clusters, where these clusters are ordered by the likelihood of modelling the background. Incoming pixels are matched against the corresponding cluster group and classified as part of the background. In our case, these methods have an obvious problem. Normally, the subject is already placed when the user wants to take a photo, therefore, the subject would be confused as a background. The user would have to point the camera before placing the subject to initiate a process of extracting information and learn what is the background. When talking about a camera in a mobile device or any digital camera, this method is nearly impractical, as the regular user tends to move the arms quite easily ruining the learning process started before. Thus, being an unreliable method for object segmentation.

For our use case, we envision a method for generic object segmentation that could be used in real-time. Since it is designed to be generic, we could not train a classifier to recognize multiple objects, or use edges information for comparison, as it would need multiple photos for data collection, and the segmentation would be limited to a few objects.

3.2.4.1 Algorithm Description

Colour is a very important feature in a photo [Kam12; San10]. With this in mind, when photographing, the main subject should cause visual impact due to its contrast relative to the background or other elements in the scenery. For this, we used a slightly simplified version of the *Histogram Based Contrast* (HC) method described by Cheng et al. [Che+11] for extraction of salient regions in a photo that evaluates global contrast differences, calculating saliency values for image pixels using colour statistics. The saliency of a pixel is

defined using its colour contrast to all other pixels in the image. This can be described as,

$$S(I_k) = \sum_{\forall I_i \in I} D(I_k, I_i), \tag{3.8}$$

where $D(I_k, I_i)$ is the colour distance between saliency value I_k and pixel value I_i in the input image in the *L*a*b** colour space. Since this is computationally expensive, Cheng et al., reduces the number of colours needed to consider, by quantizing each colour channel to have 12 different values, reducing the number of colours to $12^3 = 1728$. Since a natural image covers only a small portion of the full colour space, the less occurring colours are ignored, ensuring that the most occurring ones, cover at least 95% of the image pixels. The remaining 5% are replaced by the closest colours in the histogram. Since this quantization might introduce artifacts, the author performs a smoothing procedure over each saliency value. Each saliency value is replaced by the weighted average of the $n/4$ neighbours where n is the number of colours that fill 95% of the image pixels.

After obtaining a saliency map, the map is turned into a binary segmentation mask using a fixed threshold. Finally GrabCut [RKB04] will be used repeatedly to refine the segmentation result initially obtained by the binary segmentation mask.

3.2.4.2 Implementation Details

To use this algorithm in a real-time scenario in a mobile device, our implementation had to be simplified. GrabCut is too heavy to run smoothly on a mobile device, and for that reason, we had to discard its usage sacrificing the refinement of the segmentation for speed. After obtaining the saliency map described in [Che+11], we give a label to each pixel depending on its value. These labels would be used in GrabCut to define which pixels are considered foreground, probable foreground, background, or probable background. In our case, we used the labels to create masks with the areas defined as probable foreground and probable background. A pixel is classified as probable foreground if its saliency value was larger or equal to 200 and probable background if it was between 20 and 200. We defined experimentally the thresholds using images from the dataset used to test the original algorithm.

After obtaining a mask with each pixel labelled, we created a mask with all pixels that were considered as probable foreground and calculated the center of mass for the resulting mask. To remove artifacts that might exist from using an incorrect threshold value, we generate a bounding rectangle that starts at the calculated center of mass and expands in every direction. Each edge will continue expanding while each row or column that passes has at least a count of 25 pixels belonging to the segmentation mask, and stops when 50 rows or columns have been covered with a count of pixels below the previously stated. These thresholds were found experimentally and seemed reasonable for the test images.

The obtained bounding rectangle and mask with areas considered probable foreground is a good estimate of what was the main object in the scenario. The result of the segmentation was a section of the mask containing the probable foreground pixels, cropped in the area defined by the bounding rectangle. To fill in the gaps in the segmentation mask, we merged the probable foreground pixels with the probable background ones and cropped with the bounding rectangle.

3.2.4.3 Interface Display

After generating the mask we chose to simply show a semi-transparent mask over the live feed obtained from the camera, as shown in Figure 3.13. This would be a good indicator of what is the subject in the scenario and if it had any colour striking features relatively to the rest of the elements.

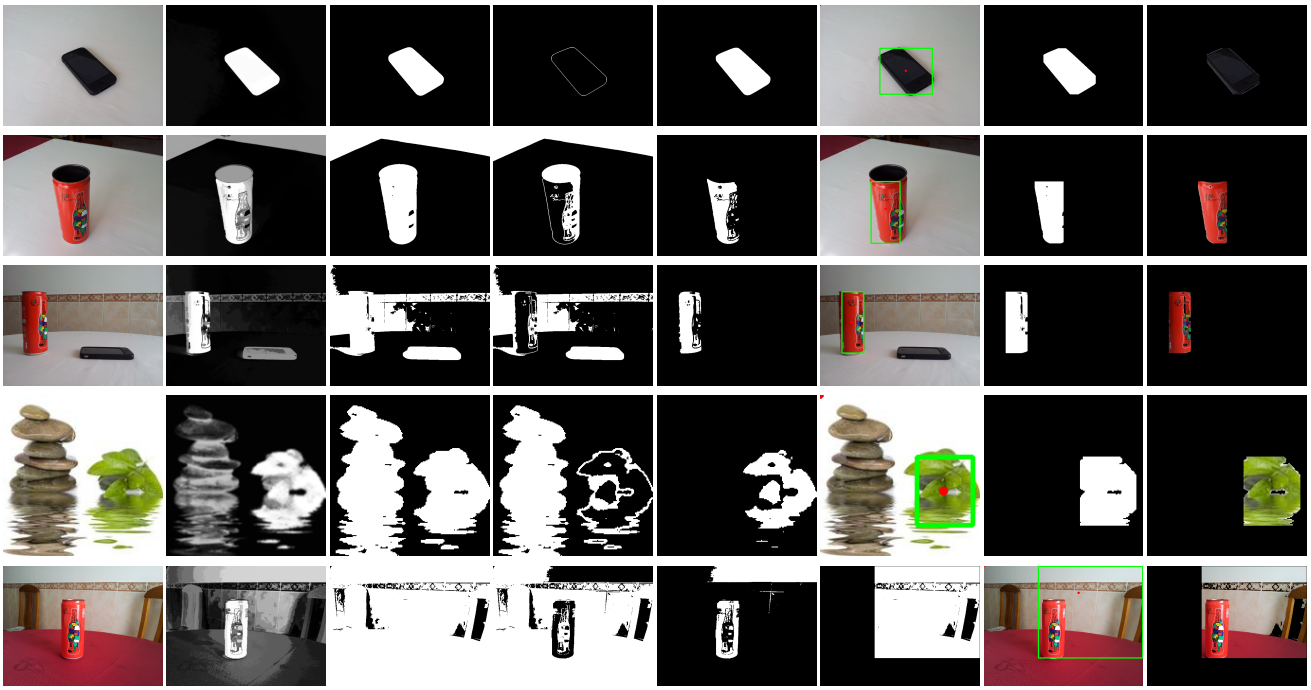
Figure 3.13: Example of object segmentation interface. The resulting mask of the algorithm is then displayed as a green overlay in the camera live feed.

This could also be used together with a guideline such as Rule of Thirds, described in Section 3.2.3, to reposition the object and change the composition.

3.2.4.4 Discussion

Being a simplified version of the original algorithm, it does not work perfectly. Since it depends on the global contrast of a subjects colours, this segmentation might not work if the background is not plain or simple. Figure 3.14 show the multiple stages taken to segment an object in an image and its final result. As it is possible to observe in this figure, the input image shows a singular object in a plain background with a successful segmentation. In the second image, it is possible to confirm that, although the object is detected, the foreground threshold is not sensitive to lightning changes, as the darkest side in the can is considered as being a probable background.

This algorithm also might not work when trying to segment two objects. Example of this is the third input image in Figure 3.14, where we can verify by the saliency map, that



a) b) c) d) e) f) g) h)

Figure 3.14: Steps taken when segmenting an object. a) Input image. b) Saliency map generated by the algorithm in [Che+11]. c) Binary mask. d) Probable background pixels. e) Probable foreground pixels. f) Bounding rectangle and center of mass point. g) Cropped mask containing the probable foreground area filled in with probable background pixels. h) Segmentation applied to the input image.

the can is obviously more salient than the other object, therefore the second object is classified as probable background. Another example of this is the fourth image. With a plain background and two objects, the object with more vivid colours is the one considered as foreground, leaving the remaining object as part of the foreground.

The fifth image shows an example were the lightning affects the outcome of the algorithm. In this case, the can and a part of the wall are considered as foreground. The solution to this would be to use a higher threshold when defining the foreground pixels, as the wall is visibly less salient than the can in the saliency map. Since it is detecting the wall as a part of the foreground, it causes the center of mass to be slightly dislocated and generates a larger bounding rectangle. When merging the probable foreground pixels with the probable background ones and cropping with the bounding rectangle, it segments a big portion of the image that its not relevant.

This object segmentation method was also used in other features implemented for this thesis.

3.2.5 Image Simplicity

To reduce the distraction caused by the objects in the background, photographers usually try to use a simple background. With that perspective in mind, the simplicity of a photo

was considered a good feature to analyse. The easiest form to obtain simplicity is to place the subject against a neutral background like a scenery or the sky. Backgrounds can be entirely neutral, like a solid backdrop or a cloudless sky; or they can complement the image, like a starfish on the sand.

Since snapshots often have cluttered backgrounds, it is expected to have edges uniformly distributed in the image. On the other hand, in many professional photos the subject is well defined, in focus, and high frequency edges are found confined to certain regions in the image.

3.2.5.1 Algorithm Description

We experimented three algorithms to deal with this problem. The first algorithm that we tested was presented by Luo and Tang in [LT08], where the author explored the simplicity of a photo through its colour distribution. For a photo, we quantize each channel in the RGB colour space into 16 values, creating a histogram with 4096 bins, which gives the counts of quantized colours present in the image. After creating the histogram, we calculate the maximum count (h_{max}) in the histogram. The simplicity of a photograph is then defined as:

$$Simplicity = \left(\frac{\|S\|}{4096}\right) * 100 \tag{3.9}$$

where S is the number of bins in the histogram that have a pixel count equal or above γh_{max}. γ is 0.01 as chosen by the author in the original article. As described in [LT08], the simplicity factor for high quality photos falls in $[0\%, 1.5\%]$, and low quality photos in $[0.5\%, 5\%]$.

For our second experiment, we implemented an algorithm described by Kao, Wu, and Liu [KWL]. Although very simple, the intuition behind this algorithm is that a very complex image is likely to contain a large amount of edges. This feature is described as:

$$B_{complexity} = \frac{n_e}{n_T}, \tag{3.10}$$

where n_e is the number of pixels that are edge and n_T is the total number of pixels in the image.

In our third experiment we implemented an algorithm described by Ke, Tang, and Jing [KTJ06], where the idea was to compute the spatial distribution of the high frequency edges. In professional photographs, the subject is normally well defined and in focus, meaning that high frequency edges will be placed in a smaller area. We started by implementing a 3×3 Laplacian matrix with $\alpha = 0.2$ as follows:

$$\begin{bmatrix} 0.2 & 0.8 & 0.2 \\ 0.8 & -4 & 0.8 \\ 0.2 & 0.8 & 0.8 \end{bmatrix}. \tag{3.11}$$

This matrix is then applied to the image and we take its absolute value to ignore

the direction of the gradients. Since this algorithm is to be applied on coloured images in real-time, we split the channels of each frame and perform this computation on each channel, taking the mean across all the channels in the end. This will create a Laplacian image that will be resized to 100×100 and normalized to values between 0 and 1. This will help when calculating the amount of area the edges occupy. It is expected for well defined objects as the ones used in high quality photos to produce a smaller bounding box, on the other hand cluttered images, are expected to have the opposite effect. The area of the bounding box is calculated by projecting the Laplacian image L onto the x and y axis independently so that:

$$P_x(i) = \sum_y L(i, y), \tag{3.12}$$

$$P_y(j) = \sum_x L(x, j). \tag{3.13}$$

This results in two vectors, one for the x axis and another one for the y, that will have a length correspondent to the image width and height, were each position in the vector will have the amount of pixels considered as an edge in that column or row.

After the projection of the Laplacian image onto the x and y axis, we find the position with the largest count of edges for each vector. The position with the maximum count will be considered the peak and we calculate the width w_x and w_y for each vector, that contains 98% of the mass of the projections P_x and P_y from that peak. The area of the bounding box containing a high density of pixels is then defined by $w_x w_y$ and the quality measure for the image is then $1 - w_x w_y$.

3.2.5.2 Interface Display

In our application we tried to understand how these three algorithms would perform under a scenario where this feature would be used in real-time. For that we displayed on the screen the scores obtained from each method (Figure 3.15(a), 3.15(b)) where, for the first algorithm [KWL] and the third algorithm [KTJ06], the score is between 0 and 1, and greater values indicate a simplicity in the scenario. For the algorithm presented in [LT08], we can obtain any value between 0% and 100% but the author defined two ranges of values where we could fit images considered simple and complex. For input frames that fit into $[0\%, 1.5\%]$ and $[0.5\%, 5\%]$ are considered as simple and complex, respectively. However, there is a range between $[0.5\%, 1.5\%]$ where it can be either one.

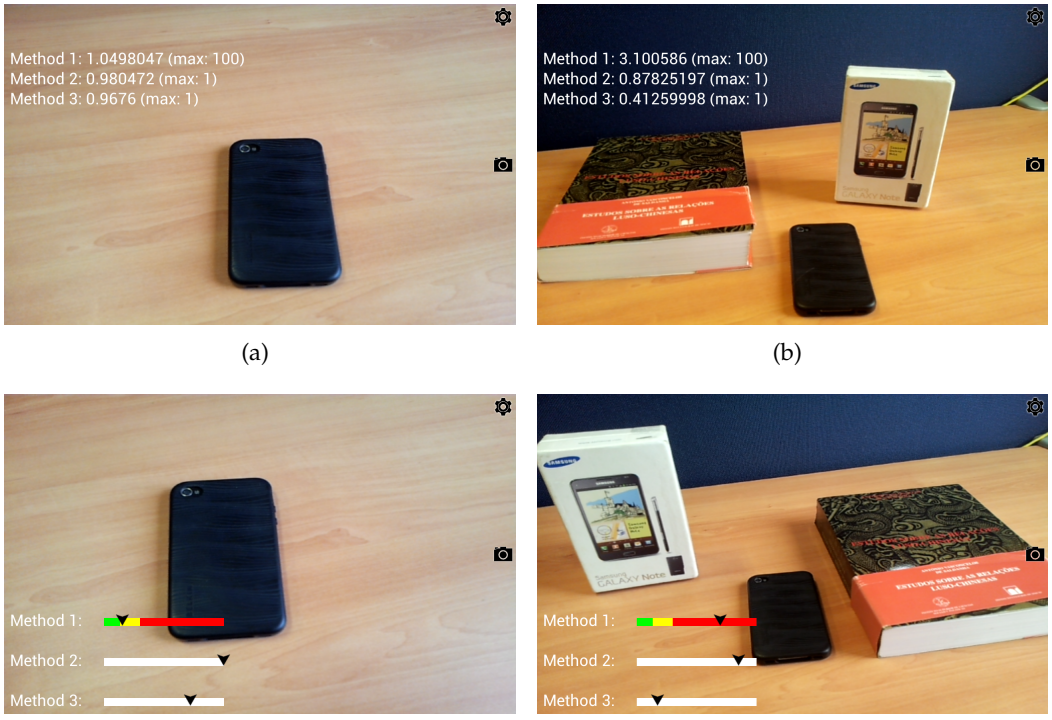
Alternatively, these scores could be replaced by a bar that represents the full scale of each algorithm with a marker indicating the current score of the frame (Figure 3.15(c), 3.15(b)). In the case of the first method [LT08] we have three ranges, that were used to divide the bar into three parts equally proportional to each range. They are then differentiated with a simple colouring label. If the indicator is in the green or red area, the

scenario is considered simple or complex, respectively. If it is a case where it can be considered both, it is represented by the yellow area as seen in Figure 3.15(c). This bar does not represent the full scale, as it only goes to 5% since it is the highest value considered in the authors range.

3.2.5.3 Discussion

Between the three methods implemented to detect the simplicity of an image, the method described by Ke, Tang, and Jing [KTJ06] has proven to be the most effective as shown in Figure 3.15. We cannot really compare with the method described by Luo and Tang [LT08], since it uses colour informations instead of edges. However it shows good results when comparing the scores obtained from a frame with a single object (Figure 3.15(c)) with a frame that has more objects and is, therefore, more complex (Figure 3.15(b)).

For the two remaining methods, the third method described by Ke, Tang, and Jing [KTJ06] presents better results. Both are based on the number of edges in the image, however the method described in [KTJ06] has into account the spatial distribution of the edges, which make it more sensible when new elements are added to the image. Therefore, it presents more reliable results. The scores obtained by the method described by Kao, Wu, and Liu [KWL] did not vary much regardless of the simplicity of the scenario.



(a) (b)

(c) (d)

Figure 3.15: Examples of visual cues given to the user. We experimented showing the results of the three implemented methods (a, b), versus using a graphical representation such as a fixed scale with an indicator showing the current score (c, d).

Although not implemented, a slight variation of this algorithm would be to segment

the subjects in the scenario and perform the simplicity calculation over the background, since professional photographers also suggest to use plain backgrounds [San10]. Snapshots are considered to have a more cluttered background comparing to professional photos, performing this computation only on segmented backgrounds would result in more accurate scores for the second and third method since the edges of the main subjects would be ignored, and therefore, it would be easier to detects edges on the background. This would also improve the score of the first method since it would then be possible to ignore the colours of the object. Ignoring the colours of the object, we could then obtain a more precise score by calculating the histogram over the background since cluttered background are likely to have more colours than simple background chosen by professionals.

3.2.6 Main Line Detection

Lines can have an important role in a photograph, as described in Section 2.4.1.5. These lines can have multiple interpretations, depending if they are horizontal, vertical, diagonal or curved. They can give a sensation of stability and safety, movement or delimit the begin and the end of a scene.

A popular method for edge detection is the use of Canny Edge Detector [Can86]. This method applies several convolution filters to each pixel with the goal of finding the pixels where the intensity variation is high [Nób13]. The algorithm starts by applying a Gaussian filter which will blur the image reducing the noise and extra edges that might be detected. After the Gaussian filter, it applies two Sobel kernels to find gradients in the horizontal and vertical direction such as:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{3.14}$$

Finally the gradient of a pixel is calculated by

$$S_p = \sqrt{S_x^2 + S_y^2}, \tag{3.15}$$

and the pixel will be accepted as an edge if it is above an upper threshold, or between the upper and lower threshold but connected to a pixel that is above the upper threshold.

Another method of line detection used is the Hough Transformation [IK88]. Lines can be represented in the Cartesian space by the equation

$$y = mx + b. \tag{3.16}$$

Any line can be represented by the equation 3.16 and therefore, it can be manipulated

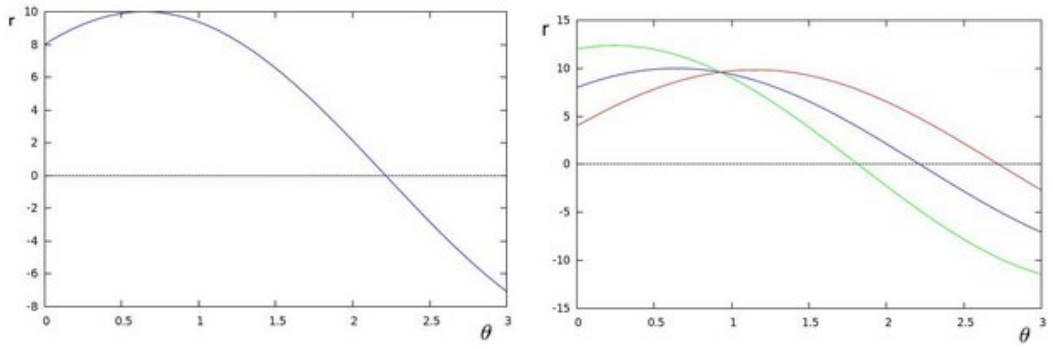
to other coordinates such as Polar Coordinates, where its general equation would be

$$y = \left(-\frac{cos\theta}{sin\theta}\right) x + \left(\frac{\rho}{sin\theta}\right), \tag{3.17}$$

and each point on a plane is determined by a distance r from a fixed point and an angle θ from a fixed direction. The Hough Transform consists in a two-dimensional space where each line is represented by a tuple (θ, ρ) and therefore all lines that pass through a point (x_0, y_0) can be represented in the Hough Space by the equation

$$\rho = x_0 cos\theta + y_0 sin\theta. \tag{3.18}$$

In Figure 3.16(a) we can observe the sinusoid obtained by plotting the family of lines that goes through a point (x_0, y_0), in the plane $\theta\rho$. By generating the same plot with a family of lines that pass through a point (x, y) we can find a line by finding the number of intersections between the sinusoid curves has shown in Figure 3.16(b). The more curves intersecting means that the line represented by that intersection has more points [Its].



(a) (b)

Figure 3.16: a) Sinusoid formed by family of lines that pass through $x_0 = 8$ and $y_0 = 6$ in plane θr b) Plot of three sinusoids that pass through the points $x_0 = 8, y_0 = 6, x_1 = 9, y_1 = 4, x_2 = 12, y_2 = 3$ with an intersection point in $(0.925.9.6)$. This intersection point with parameters (θ, ρ) defines the line in which $(x_0, y_0), (x_1, y_1)$ and (x_2, y_2) lay [Its].

The combination of the Canny Edge detector and the representation of a line in the Hough Space will be useful in the implementation of this functionality.

3.2.6.1 Algorithm Description

In our algorithm we start by applying the Canny Edge detector to each frame. This will result in a binary image with only the edges B_e of the frame. In order to get the coordinates with a large count of intersections, we must first create an accumulator with a length equal to the Hough space were the coordinates will be represented. This length will correspond to $180 * 2\rho$, where θ will have values between $[0, 180]$ and ρ will be between $[-\rho, \rho]$.

After the initialization of the accumulator, we scan each pixel of B_e searching for a pixel that is edge. When found, the coordinates for this pixel will be used to generate a family of lines in the Polar Coordinates system through the equation 3.18. By varying the θ value in this equation, we obtain its corresponding ρ and increment the count of intersections in the coordinates (θ, ρ).

After filling the accumulator, we select the pairs of (θ, ρ) with a number of intersections above a given dynamic threshold. To be selected as a line candidate, each pair (θ, ρ) has to have a number of intersections above the threshold and be a local maximum. This is verified by comparing its value with the value of its neighbours in a range of 9×9.

When a line is selected as a main line we convert its coordinates (θ, ρ) to the Cartesian Coordinate system. This will result in vector with pairs of points that will later be used to draw a line in the interface, considered as an important line. The steps taken through this algorithm can be visualized in Figure 3.17.

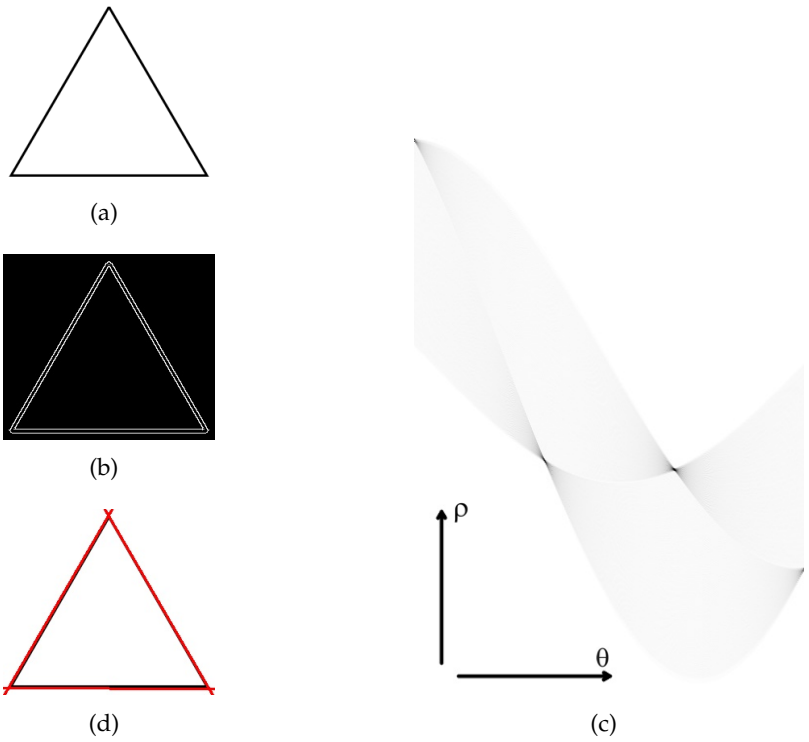


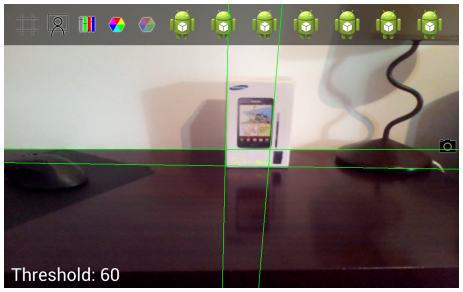
Figure 3.17: a) Source image. b) Canny edge detection method applied to the source image. c) Visual representation of the accumulator. Darker areas represent a larger number of intersections in those (θ, ρ) coordinates. d) Lines drawn in red that represent the main lines detected in the source image.

3.2.6.2 Interface Display

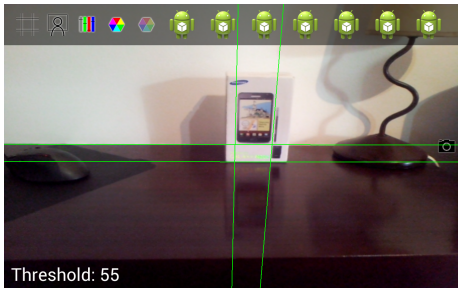
For its interface, we choose to simply draw the lines over the real feed of the camera. The photographer can then use the main lines detected to choose a better angle or rearrange the composition, so that the final viewer has a guidance when looking at the final product.

Figure 3.18 demonstrates the result of main lines detection applied in a real-time scenario, drawing the prominent lines directly in the viewfinder. In this feature we added controls to increase or decrease the threshold value, that is also shown on the lower left corner.

(a)



(b)



(c)

Figure 3.18: Main lines detection interface with threshold of 130 (a), 60 (b) and 65 (c).

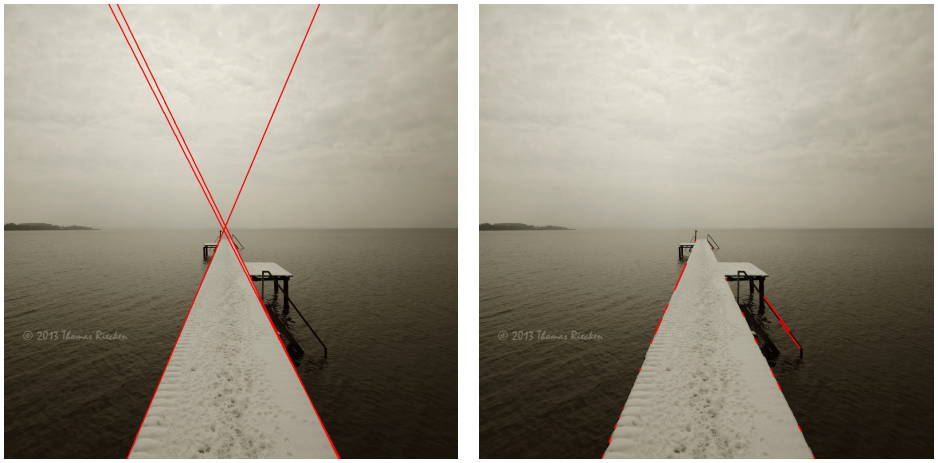
Since a photo can be highly textured, adding control over the number of lines that appear on screen, decreases the probability of having too many lines drawn and can improve the frame-per-second rate at which those lines are shown.

3.2.6.3 Discussion

This algorithm can be quite slow and ineffective if the source image has too many details or textures. A solution to obtain a speed increase in the algorithm, was to establish a limit of a maximum of 15 lines that can be detected at any time for any threshold.

As previously mentioned, this algorithm draws lines across the viewfinder which is a problem of the method used. As an alternative, the method we implemented could be replaced by a line detection method using the progressive probabilistic Hough Transform [MGK00], as this presents line segments instead of lines that go across the full height or width of the frame. Although it claims to be faster, it was not chosen as it would be necessary to define even more thresholds that could not be easily controlled by the user. In this case we would have to define the minimum line length and the maximum line gap between points of the same line, which would be very difficult to determine experimentally and hard to control in a mobile device.

In Figure 3.19 we can verify the difference between both methods on the same input image with the same parameters (minimum line length and maximum line gap set to default). In Figure 3.19(a) we can verify that it can give the photographer a a more reliable insight of were the lines converge. On the contrary, Hough Transform with progressive probabilistic detection in Figure 3.19(b) only shows small scattered segments that make it hard to understand the point of convergence. A solution could be to try and create a more relevant line by joining multiple segments but it would only add computational effort to a device that already struggles dealing with real-time image processing.



(a) (b)

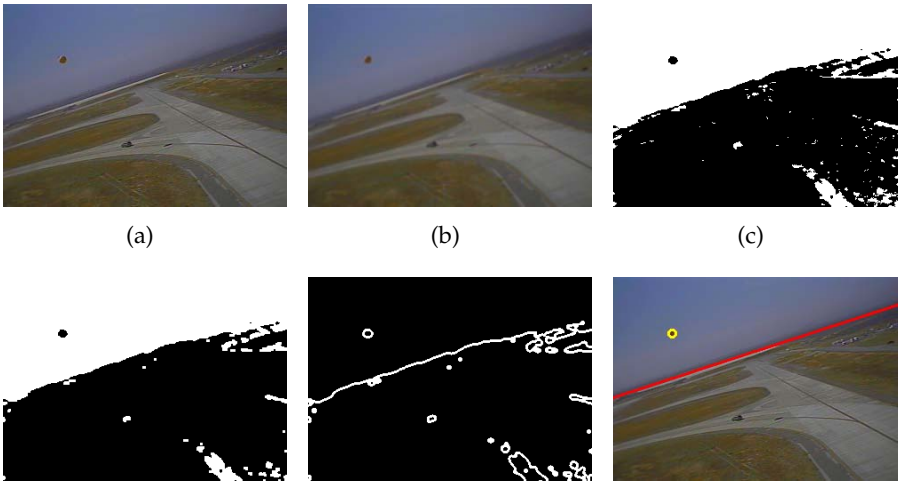
Figure 3.19: Main line detection with regular Hough Transform (a) and with progressive probabilistic Hough Transform (b).

3.2.7 Horizon Detection

Horizon detection in still images or video sequences contributes to applications like automatic correction of image tilt and image quality enhancement. In seascapes and landscapes, which are genres of photography, a feature that detects the horizon line in real-time is useful for a correct placement of the line and indicate a tilt correction on the mobile device. At the pixel level, sky detection can be used for content-based image manipulation, like picture quality improvement using colour enhancement and noise reduction, or as background detection for 3D depth-map generation [Zaf+06].

Sky detection research has also proven useful for object detection for small unmanned vehicles [MSH05]. McGee, Sengupta, and Hedrick [MSH05] presented a system for sky segmentation and horizon detection based on an image colour and texture properties. For sky segmentation the author used a support vector machine (SVM) for classification of sky and non-sky pixels in the colour space YCrCb, after smoothing the image with a Gaussian filter to reduce the effects of noise. After the SVM divides the sky from non-sky pixels (Figure 3.20(c)), a binary image will be generated that will then suffer successive erosions and dilations to remove unwanted pixels that were considered sky pixels, as

shown in Figure 3.20(d). After removing small sections of misclassified pixels, to find the borders between sky and non-sky regions, it smooths the binary image and classifies all pixels with value near 0.5 as boundary pixels. After performing the edge detection, the horizon detection is then applied using the Hough Transformation method as explained in Section 3.2.6. With the the horizon line, any areas considered as non-sky regions are then considered as an obstacle. Figure 3.20 shows all the steps taken in this algorithm.



(a) (b) (c)

(d) (e) (f)

Figure 3.20: a) Input image. b) Smoothed image. c) Binary segmentation. d) Result from erosion and dilation. e) Border between sky and non-sky areas. f) Horizon and obstacle found.

3.2.7.1 Algorithm Description

For our application, we adapted part of an algorithm described by Zafarifar, Weda, et al. [ZW+08] that uses both colour and edge features to detect the horizon lines. This algorithm explores the physical phenomenon of colour de-saturation and brightness increase along zenith-to-horizon direction to calculate the position and angle of the horizon, that are then refined using edge detection techniques.

This implementation starts by calculating a sky probability map that represents the probability of each pixel belonging to the sky. This probability assumes that clear sky has some properties:

1. Pixels in the top area of the image, have a higher probability of being sky-pixels,
2. They have a certain range of colour, when limited to day-light condition,
3. Pixels that represent sky, contain low texture,
4. The speed of change in colour values in horizontal and vertical directions is limited,
5. There is a luminance increase and chrominance decrease along the zenith-to-horizon direction.

Taking into account these properties, the probability of a pixel being a sky-pixel (P_{sky}), can be calculated through a pixel colour, texture, position and gradient features by the expression

$$P_{sky} = P_{colour} * P_{texture} * P_{position} * P_{gradient}. \tag{3.19}$$

In our implementation we simplified the algorithm by ignoring the gradient information and consequently the sky luminance and chrominance variations, which resulted in the expression

$$P_{sky} = P_{colour} * P_{texture} * P_{position}, \tag{3.20}$$

described by Herman and Bellers [HB03]. As previously stated, the areas that have a higher probability of being related to the sky are conventionally at the top of the image. With that in mind, the probability $P_{position}$ can be calculated as

$$P_{position} = e^{-\left(\frac{L}{\#lines}\right)^2}, \tag{3.21}$$

where L is y for a pixel in the position (x, y), and $\#lines$ is the total height of the image.

The colour probability distribution, is calculated to each one of the channels in *YUV* colour space with the expression

$$P_{colour} = e^{-\left[\left(\frac{y-y_0}{\sigma_y}\right)^2+\left(\frac{u-u_0}{\sigma_u}\right)^2+\left(\frac{v-v_0}{\sigma_v}\right)^2\right]}, \tag{3.22}$$

where y,u and v are the values in each channel of the pixel being processed. The rest of the parameters are determined empirically by the author, as:

$$y_0 = 210, \sigma_y = 130; u_0 = 150, \sigma_u = 40; v_0 = 100, \sigma_v = 40. \tag{3.23}$$

The probability function for texture of a pixel is calculated as

$$P_{texture} = e^{-0.2*(tt)^2}, \tag{3.24}$$

where tt is the absolute difference of luminance values of a current pixel and the following one in the same line.

After calculating the probability of each pixel with the equation 3.21, we can then segment the probability map and create a binary image using a threshold set as 60, meaning if the pixel has a probability greater than 60% then it is considered as sky in the binary image.

After segmenting the sky, we apply the Hough Transform to the resulting mask and store the values of (ρ, θ). These values will be used on the second part of the algorithm that uses edge detection methods.

The second part of the algorithm is where we mix both colour and edge detection methods to find the horizon line. From the first part we stored the (ρ_{CD}, θ_{CD}) values

obtained by applying the Hough Transformation to the binary image containing the areas that were considered sky. The purpose of this is to use these parameters that indicate the position and angle of a probable horizon line.

For the detection using edges, we start by applying the Canny edge detection method to the luminance component of the image, and since we are only interested in the most prominent straight edges, we choose a large value for the low pass filter. This will smooth the detailed edges yielding in a binary image that contains the significant edges. We then apply the Hough Transformation to the edge map which will result in a matrix representing all the lines in the Hough Space similar to Figure 3.17(c).

Since we previously stored the position and angle of the horizon line calculated by the colour edge detection method, we proceed to multiply a weighting factor in the form of a 2D Gaussian function to the result of the Hough Transform in the edge detector with the parameters (ρ_{CD}, θ_{CD}) obtained previously, as its center and find the pair (ρ_{ED}, θ_{ED}) with the highest value in the post-processed Hough map.

Now with the pairs (ρ_{CD}, θ_{CD}) and (ρ_{ED}, θ_{ED}) that represent a probable horizon line in each of the methods, we define the horizon line as an average of the ones obtained from the previous methods.

3.2.7.2 Interface Display

For this feature we chose the simplest way to draw a horizon line which was to draw the line in the viewfinder. To eliminate the chance of displaying erroneous results, we took advantage of the Android sensors. Using information given by the mobile device relative to its accelerometer and magnetic field we were able to filter the results. With this information we can detect the orientation of the mobile device and compare it with the line slope. If the slope of the line is vertical/horizontal and the orientation of the mobile device is horizontal/vertical, the line is then excluded and not drawn. Figure 3.21 has an illustrative example of the horizon line based on the edge detector and color detector previously described, as well as the result obtain by merging the two.

3.2.7.3 Discussion

Although this method can perform under the right conditions, it has a high fail rate in detecting the correct horizon line. Previously we stated that the colour detector was not fully implemented. Originally the algorithm has into account the physical phenomenon of colour de-saturation and brightness increase along zenith-to-horizon direction, which involved extracting the gradient features as expressed in equation 3.19. This was ignored in an attempt to minimize the computational load on the mobile device. The extraction of gradient features would also give useful information about the angle and position of the horizon along with a confidence metric on these parameters. We obtained an angle and position for the horizon line on the colour detector, however we cannot obtain the

Figure 3.21: Exemple of horizon detection where the green and blue line are the result of the edge and color detector respectively. The red line represents the result of combining both methods.

confidence metric. Without this confidence metric, the edge detector has a high probability of failure in cases where the image as an object with well defined edges, therefore, vertical objects might ruin the horizon detection in images with obstructed horizons.

The algorithm has an overall poor performance which is aggravated by the fact that it is used in real-time. Being used in real-time the mobile device is susceptible to minor shakes by the users movement. Since it is slow, while the frame is being processed, the image has already changed and the results displayed belong to frames from the past. This could be surpassed with mobile devices with higher computational capabilities or even processing. For this feature, we believe that using external processing server-based would be advantageous as it could reduce the detection time by at least half of the time that it currently takes.

Another problem in this algorithm is the fact that it was made to work with images that have a clear blue sky with a certain tone to fit in the thresholds defined. To detect other skies, we would have to detect the average sky colour in real-time, and redefine the threshold based on that information, although this would not be necessarily easy since the thresholds were determined based on a series of images.

3.2.8 Image Balance

Many of the photography rules of composition relate to the idea of balance. Ideally, we want our images to be balanced. By "balanced", we mean that no single area of the image draws our eye so much that we get stuck there. A balanced image feels pleasing to the eye, and not asymmetric in any way or has multiple elements with its visual weight evenly distributed. Every element in the composition carries a certain amount of visual weight. To keep the image balanced, it is necessary to compensate for each element with

a counter-weight through colours, different levels of contrast, or different subject positions. As described in Section 2.4.1.6, we can easily perceive the balance of an image by imagining a dividing line through the middle and compare the weight of both sides as it can be balanced, unbalanced or have any kind of visual tension between the elements.

3.2.8.1 Algorithm Description

We did not follow any previous algorithm as we experimented multiple ways of trying to find the elements in a picture and explore its size. We explain the algorithm for our experiments and the algorithm chosen for this feature.

First experiment

For our first experiment we tried to take a simple approach. The first step was to obtain the left and right part of an image creating two independent regions. Since we were starting by experimenting a way of detecting symmetry, an easy way to achieve that would be by finding the edges of both regions using a Laplacian kernel, and calculate its center of mass. For the center of mass to be close of one another in both regions, one of the sides had to be flipped in the horizontal. After finding the centers of mass we tried to see if the image was symmetric by calculating the Euclidean Distance between both centers of mass.

For a second attempt, instead of just dividing the image in left and right side, we divided each of the areas into six sub-areas and calculated the center of mass of each one of those sub-areas. To see if an image was symmetric we opted to count the number of edges for each subregion and sum the absolute difference between the edge count of overlapping subregions on both left and right side of the image.

Second experiment

At this point we started to understand that it was important to properly get the symmetry axis in an image. In this attempt we started by converting the input frame to the *HSV* colour space and use the hue channel to perform colour clustering. We used the implementation of *kmeans* in OpenCV [Its] to segment the image into two clusters in a naive attempt to segment the object in the image. After the clustering, we find the center of mass of each cluster. We then assume that the perpendicular line between the line segment formed with those two points, is the symmetry axis in the input frame. After finding the symmetry axis we calculate the amount of each cluster on each side of the symmetry line. We tried to understand if a image was symmetric by comparing the amounts of each cluster in both sides of the symmetry line.

Final algorithm

In the final algorithm, we used a simplified implementation of a method used for real-time object tracking symmetry [LK06], as this would give us our symmetry line. To find the symmetry, it starts by converting the input image into an edge

image using Canny edge filter with a lower threshold of 182 and upper threshold of 350, where the pixel is accepted as an edge if the gradient value is higher than the upper threshold, it is rejected if the pixel gradient is below the lower threshold or it is accepted if the gradient value is between both thresholds and is connected to a pixel that is above the upper threshold. These threshold values nearly satisfy the recommendation of a upper:lower ratio of 2:1 [Its]. Each pair of edge pixels, i and j, votes for a particular ρ and θ in the Hough transform accumulator. In order to reject noisy edge pixels and to favour votes made by edge pixels with symmetric image gradients, a Gaussian weight function is used so that the weight of any pixel (i, j) with the parameters (ρ, θ) in the Hough space, is maximized when gradient orientations are symmetric about their mid point. This means that for each pair of (ρ, θ) in the Hough map, its value will be voted by the following equation:

$$H(\rho, \theta) = \sum_{i,j \in \Gamma(\rho,\theta)} W(i, j), \tag{3.25}$$

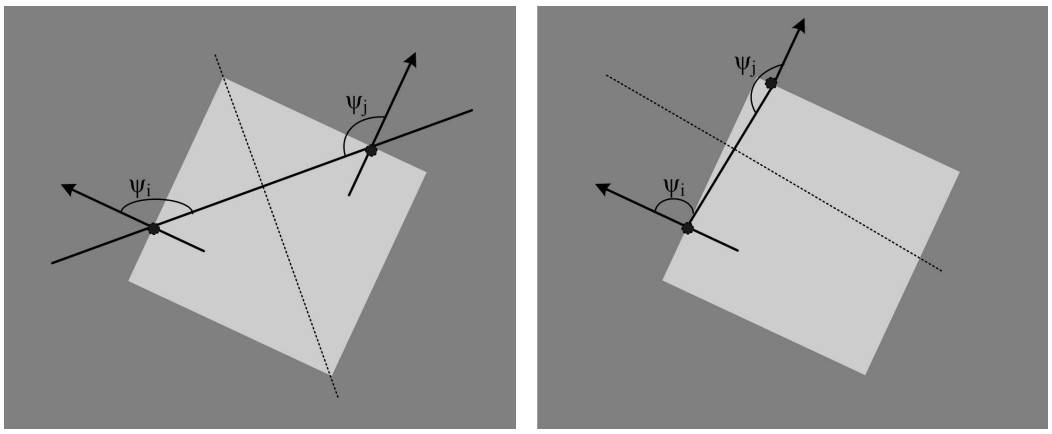
where $\Gamma(\rho, \theta) = \{(i, j)|\rho(i, j) = \rho, \theta(i, j) = \theta\}$. This means that for a pair (ρ, θ), its value will be the sum of the weighting function of all pixels (i, j) with the same parameters (ρ, θ) in the Hough space. The weighting function is described as:

$$W(i, j) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(\psi_i - \psi_j)^2}{2\sigma^2}}, \tag{3.26}$$

where ψ_i and ψ_j are the image gradient angles and σ defines the strictness of the gradient symmetry criteria. The relation between the image gradient angles and the level of symmetry is illustrated in Figure 3.22. Horizontal and vertical Sobel filters are applied to determine the image gradients and using the absolute magnitude of angles we verify that $|\psi_i - \psi_j| = 0$ which means that there is a symmetry. On the other end, if $|\psi_i - \psi_j| \approx \frac{\pi}{2}$ means that there is no symmetry and therefore, it will receive a very low weight by the equation 3.26. Figures 3.22(a) and 3.22(b) illustrate these statements.

After the voting process, a total of three peaks in the Hough map are selected. The peaks in the Hough accumulator are identified using the Non-Maxima suppression, which locates the highest value above a threshold in the Hough map, selects it as its peak and suppresses all its neighbours to zero. In our case, the suppression neighbourhood of each peak is $\frac{1}{20}\rho_{max}$ and $\frac{1}{20}\theta_{max}$. To obtain the lines formed from the three peaks selected, we average the median point of each line segment and slopes. The median point will give us a point in the frame that we know for sure that belongs to the symmetry line and together with the slope, we can calculate which of the frame borders are intersected by the symmetry line.

The next step to take after understanding where is the symmetry line in the input frame, is to calculate the weight of the main elements on each side of the symmetry



(a) (b)

Figure 3.22: Angles used in weight calculations. Arrows represent the image gradient direction at the edge pixels. [LK06]

line. Knowing the symmetry line we then create two masks that will be used to differentiate the left from the right side of the line. We combine these masks with the object segmentation method (Section 3.2.4) obtaining the most salient elements in each one of the sides.

The final step is to calculate the percentage of amount occupied by the elements on each side of the symmetry line. By calculating the difference between the occupation amount on each side we can then classify as balanced if this difference is more or equal to 70% or unbalanced if the difference is less or equal to 30%. If the difference does not contemplate any of these cases, it is considered that the image has a kind of visual tension.

3.2.8.2 Interface Display

Since the purpose of this application is to have simple visual cues, we choose to show the user the line of symmetry being detected at the current frame and in which of the categories it belongs, considering the balance detected. As shown in Figure 3.23, on the upper-left corner, there is a list with the three categories of balance. Given difference percentage of occupation amount between the two sides of the symmetry line, one of this categories is highlighted informing the user if the current scenario is balanced, unbalanced or has any kind of visual tension.

3.2.8.3 Discussion

Although it can give an approximate result from what is expected, this feature has some problems regarding the chosen algorithms. Originally the algorithm described by Li and

Figure 3.23: Example of the visual cue to inform the user about the scenario balance.

Kleeman [LK06] was intended to detect and track objects in real-time based on its symmetric features. Thus, if we imagine a plain background with a simple object, the symmetry line will be calculated according to its edges, so the line will go across the object. This proves that even if the scenario is completely one-sided, it can be considered as balanced. This would depend on the symmetry axis idealized by the user, which is not the case. This can also raise problems when there is more than one object in the frame, since the lines formed by the chosen peaks, can all be related to the same object. We then added a threshold that gradually increases on each iteration when finding the peaks ensuring that the maximum threshold when finding a peak is

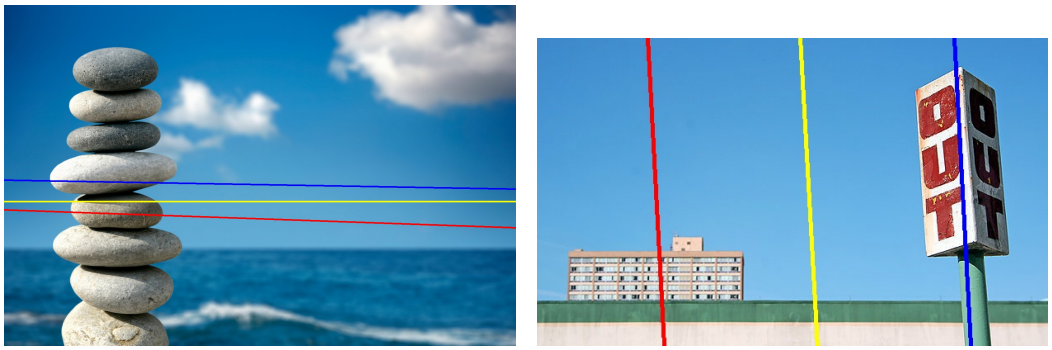
$$max_{threshold} = max_{threshold} - (max_{threshold} * (0.25 * n_{peaks})), \tag{3.27}$$

where $max_{threshold}$ starts with the maximum value found on the Hough map and gradually decreases with the number of peaks found (n_{peaks}).

Figure 3.24 illustrates the problem stated. In Figure 3.24(a) is visible the first problem discussed, where the image is obviously unbalanced to anyone who sees the picture but since there are no edges on the right side, the symmetry axis is calculated only with the edges found on the left side. In this case the result is acceptable as it is possible to also draw a horizontal symmetry axis making it perfectly balanced but its counterintuitive.

In a scenario with two objects, it becomes possible to select a symmetry axis for each object and calculate an averaged symmetry line with the two previously found. Figure 3.24(b) illustrates the solution we opted for the second problem. Using a dynamic maximum threshold ($max_{threshold}$), we were able to select a symmetry axis for each object and average them, calculating a correct symmetry axis.

However, this solution has some difficulty in classifying unbalanced scenarios with two objects were one is obviously larger than the other. Figure 3.25 illustrates an example were the same scenario with two objects of different sizes, are classified differently. This is due to the fact that if one object is obviously larger than the other, the symmetry axis is defined based on the larger object most relevant edges, therefore, the edges from the smallest object are not taken into account.



(a) (b)

Figure 3.24: Examples of an image obviously unbalanced that is considered as balanced due to the symmetry line calculated (a) and the averaging of two symmetry lines detected on two different objects (b). The blue and red lines represent two of the peaks found in the Hough map and the yellow line the resulting of averaging them both.

(a) (b)

Figure 3.25: Examples of an unbalanced image with objects of different sizes where one is classified as unbalanced (a) and the other as balanced (b), as a result of the calculated symmetry axis.

Besides the problem in the discovery of the symmetry axis, the object segmentation also has the problems that were discussed in Section 3.2.4.4 where what we consider an object in the scenario, might not have enough colour contrast relatively to the background.

The thresholds used in this algorithm were chosen experimentally with samples of images considered symmetric or unbalanced, which means that the values might not perform well on various compositions in a real-time case scenario.

3.3 Discussion

Through out this section we described the functionalities and the algorithm implemented for each one, how we show the information to the user and discuss problems and advantages that they might have.

Since colour is an important part of an image, we described some features that are related to the colour properties of an input frame. These features include two different representations of an histogram, one with a bar for each channel and grayscale image that indicate de range of colours being used as well as the indication of quantity of pixels near the edges of the scale, and the other with a representation similar to a colour wheel that indicates the amount of each primary and secondary colour in the *Hue* spectrum. Still related to colour properties, since highly colourful images affect the viewer and grab their attention, we implemented a saturation detector. When the colours of a scenario are low on saturation it displays a suggestion indicating that a monochromatic filter might be useful.

A rule followed by professional photographers is to use a small number of colours and combine them wisely to trigger an emotion in the viewer. With that in mind we implemented a scoring system that is given to the input frame, based on the number of colours detected and a display of the template being used, as it can be a monochromatic or complementary template [CO+06].

More related to the composition of a photography, we also implemented features based in detecting elements from a real-time feed. The most common detector when talking about photography is the face detector [Its]. Implemented in a wide variety of systems now available, we took a little step further in mixing this concept with rules related to photography, such as the Rule of Thirds (Section 2.4.1.3) and triangular compositions [San10]. Although not implemented, these rules could also be used with our implementation of object segmentation based on saliency maps [Che+11]. The object segmentation, even if dependent on the colour contrast of the object in comparison to the rest of the background, gives quite accurate results with simple backgrounds.

Besides the simplicity of the background, in photography, simpler photos are considered to have good aesthetic properties [Kam12]. Therefore, we experimented a couple of metrics that rate the simplicity of a scenario based on the edges detected. We obtained the best result with the our third metric that is not only based on the number of edges

detected, but also as into account the spatial distribution of those edges [KTJ06].

Still related to the composition, the balance of an image (Section 2.4.1.6) has an important role on the visual impact onto the viewer. With that in mind we implemented a feature to evaluate the balance in the scenario. This is based on a detected symmetry line and the weight of the objects on each side of the symmetry.

The lines that can be found in images are very powerful elements that with a little practice can add dynamic impact to a photograph in terms of mood as well as how they guide a viewer in a photo (Section 2.4.1.5). Varying between vertical, horizontal, diagonal or curved lines, we focused on detecting dominant straight lines without taking into account their inclination, to help the user find and work with of those important lines. Still related with lines, the horizon line is is usually the most important when taking a photography of a seascape of landscape. Although not perfect, our implementation of a horizon detector [ZW+08] can still be useful when repositioning this line according to the Rule of Thirds (Section 2.4.1.3).

4

Results and Evaluation

To assess the results of our tools, we took various approaches as for testing. In this chapter we describe the methods, results of the tests and inquiries on a group of users.

4.1 Algorithm Comparison

Since we decided to use a modified version of some algorithms to apply them in a real-time environment, it made sense to compare our results with the ones obtained from the original algorithms.

One the algorithms we used that was partially implemented, was the object segmentation (Section 3.2.4). For this algorithm we opted for not using Grabcut [RKB04] to refine the results since the saliency map already gave a good indication of what was the object in a scenario. To compare with the original algorithm, we used the same dataset as the author and compared the resulting segmentation masks.

From a subset of 500 images, we ran our implementation and could conclude that for almost all the images, a significant part of the main subject in the photo was always detected. Since we decided not to use Grabcut, our results could not be refined and therefore our segmentation masks would have a lot of unfilled areas that were still considered as part of the subject. In other cases, the saliency map that we implemented completely failed. Some of the reasons for this to fail was the lack of contrast of the subject in comparison to the background, as the background might have a higher contrast comparing to the rest of the image. We could also verify that another thing that highly influences the results is the brightness, as bright areas were more likely to be chosen as part of the subject.

As referred in Section 3.2.4, our segmentation mask was based on pixels that were

considered as foreground while all other were background. In an attempt to fill the areas that were not considered foreground and still belonged to the subject, we considered all pixels as foreground since they were inside of a bounding area. We also did a comparison of this method with the original algorithm using the same subset of images. For the tested images, it fulfilled its purpose. However, in most images it ruins the results because many background elements are then added to the mask. Some of the results for the methods tested can be seen and compared in Annex B.

Another algorithm we tested, was the colour template detection. For this feature we implemented a simpler algorithm instead of simplifying the original one. However, Cohen-Or et al. [CO+06] did not describe any methods of evaluation for this algorithm. We then used a couple of images that showed results and compared the results of our own algorithm. In Annex C it is possible to observe some images used by the author and compare the results obtained from the original algorithm with our results. From what we could conclude, our results were not too far off from what was expected and still give a good indication if the scenario is using a monochromatic or complementary template, and what are the most salient colours.

The last algorithm we compared was the horizon detection algorithm. We gathered 32 images from Flickr that were labelled with the tags *seascape*, *landscape* and *horizon*, as these were likely to have a clear horizon line and include some with an obstructed horizon. The result was evaluated by comparing the position and angle of the detected horizon line with a manually-annotated horizon line. For each image, the position of the horizon line was calculated by averaging the vertical position of the two points forming the line, and used the same two points to calculate its slope then converted to an angle. In Annex D are the results obtained from the images tested. We could verify that in the images labelled as *seascape* and *landscape* the deviation of the horizon position was relatively low with only an average relative error of 9.5% and 8.2%, respectively. On the contrary, images labelled as *horizon*, had an average relative error of 55%, due to being images more prominent to edges. However, for the angles, there is a high deviation in most of the values. This is because the values of the angles are very close to zero, thus the difference between the calculated value and the original one is almost neglectible but has high impact in the relative error for being a division of two very small numbers. In most cases, the angle difference between the real horizon line and the detected horizon line is small and does not really affect the judgement of where the horizon line is.

4.2 Algorithm Execution Time

Having the purpose of displaying information in real-time, we considered that the execution times of each algorithm should be taken into account. Slow algorithms are less likely to be useful in a real-time scenario and might even ruin the experience of the user. With that in mind, we did performance tests over each feature to understand its reliability in real-time processing.

4.2.1 Testing tool

These times were taken by generating log files that contain trace information to analyse. We used the Android's *Debug* class to call the methods *startMethodTracing()* and *stopMethodTracing()*, which start and stop logging of trace information to disk. This option is very precise since we can specify blocks of code from where we want to generate tracing data. For all tests we started logging in the moment that the application receives a frame from the device's camera, until the processing is done and the view from the current feature is refreshed [Goob].

After generating the log files, we used a tool called *dmtracedump* to generate a graphical call-stack diagram. Figure 4.1 illustrates an example of the generated diagram from a trace file. In this tree diagram each call is represented as a node, and shows the call flow (from parent node to child node) using arrows. For each node, *dmtracedump* shows <*ref*> *callname* (<*inc-ms*>, <*exc-ms*>,<*numcalls*>), where

- <*ref*> - Call reference number, as used in trace logs,
- <*inc-ms*> - Inclusive elapsed time (milliseconds spent in method, including all child methods),
- <*exc-ms*> - Exclusive elapsed time (milliseconds spent in method, not including any child methods),
- <*numcalls*> - Number of calls.

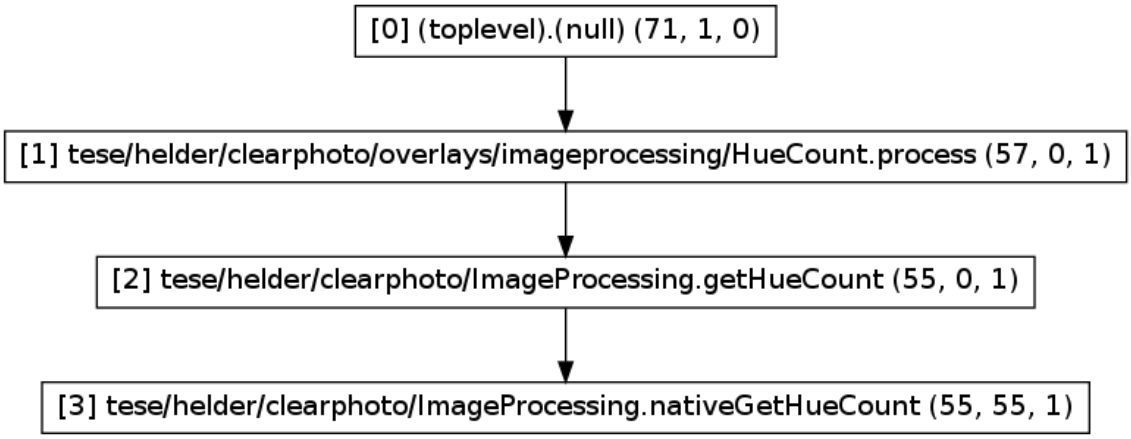


Figure 4.1: Graphical diagram generated from a trace log file.

Using these diagrams, we were able to get the amount of milliseconds spent in a method (<*exc-ms*>) and any method called by it (<*inc-ms*>). With the trace log file we can obtain the total time it took to perform the computation from the start to finish. For each feature we took 10 samples and calculated the average time and standard deviation for the total amount of time it took to perform the trace and the time spent in the method

that processed the data, ignoring the lowest and highest value. This way we expect to obtain relevant information about the time spent processing the frame and refreshing the view of each feature.

4.2.2 Results

All the times taken are in Annex A represented by a table with the time spent processing and the average of total time spent for each sample. Some of these features were tested under different conditions or parameters to understand the difference in performance. For example, the saturation detection (Table A.1) was tested in scenarios where the detector successfully indicated low saturation and in scenarios where it failed. We could conclude by the samples taken that the detector performs better when it fails to detect a low saturation environment. This is due to the extra computational effort that the device has to make to convert the frame into grayscale and display it on the viewfinder. However the algorithm is not that slow and it still performs quite well, disregarding the visual cue that can easily be replaced by a more appropriate one.

Another feature that we experimented in different scenarios was the face detection (Table A.6). We experimented detecting one or three faces as well as using the suggestions for improving the composition. As we can conclude from the results, there were not any significant changes from each one the scenarios. As expected, the most costly computation was the detection of a face in a frame. In comparison, the calculation of the suggestion to improve the composition is neglectable.

The detection of main lines is dependent of a threshold, therefore we sampled the algorithm with different thresholds. The threshold values tested were 160, 100 and 40 as lower thresholds detect more lines. As we can conclude from Table A.9, for all thresholds the computational effort was similar regardless of the value.

We also implemented three algorithms to calculate the simplicity on an image, therefore, the performance of each one was tested individually (Table A.8). We could conclude that the fastest method was the second method described by Kao, Wu, and Liu [KWL]. However, as discussed in Section 3.2.5.3, this is the least reliable method comparing to the other two. On the other hand, the first method [LT08] and the third method [KTJ06] tested are approximately 50ms slower, which makes it a good option.

From the remaining features, the ones that performed better were the histograms calculation, detection of templates and scoring based on the *Hue* channel. For the sampled tests, all these features performed in 50-150ms which can be translated into 6-20 frames-per-second, where the colour histogram in the *Hue* channel and colour template detection had the slowest times.

On the other hand, the slowest features were the object segmentation, the image balance and the detection of the horizon line, with average times of approximately 240ms, 600ms and 2700ms, respectively. Being more complex algorithms, these results were expected, however, detecting the image balance and the horizon line, are algorithms too

demanding to be used in a real-time scenario. Object segmentation and the horizon line detection are algorithms that give important visual feedback since they work as an overlay over the real frame, however, both algorithms perform poorly.

These tests were performed in a Samsung Galaxy Note with Android 4.1. We made sure that each feature was doing what was expected of it for at least 30 seconds, so that the results would be the most reliable possible. After that, we extracted the log files from the mobile device and extracted the execution time from a total of 10 runs for each feature.

4.3 Users Testing

The evaluation of this proof-of-concept tool was made by conducting tests with 11 users. The test was composed of a set of basic tasks and a questionnaire. In some cases, it led to a discussion with more experienced test subjects. The tasks were focused on getting an assessment about the utility and overall satisfaction on the functionalities implemented and tried to understand if they are useful in a real-time scenario. The questionnaires consisted in a total of 47 questions using the Likert scale (1-*strongly disagree* to 5-*strongly agree*) and an area for suggestions and comments about what was experienced. During the evaluation tests, a Samsung Galaxy Note was used for the application.

4.3.1 Participants

As mentioned before, we did tests to a total of 11 subjects of both genders, with 64% males and 36% females. The ages varied through 22 and 24, being the predominant age 23 with 50% of the users. To understand what kind of background the users had, we realized a couple of questions to assess their knowledge in photography. We could conclude that 10 (91%) of the test subjects take photographs, but 7 (64%) claimed to have some knowledge in the area and classified their level of knowledge as amateur. When asked about what kind of devices they use to take photographs, the majority answered camera phones (75%). Some still admitted to use still cameras (27%), instant cameras (18%) and DSLR cameras (18%). The frequency of use of each device can be seen in Figure F.1 (Annex F).

4.3.2 Questionnaire

For the questionnaires, we asked the users to test all the functionalities and answer a total of 47 questions using the Likert scale. The users started by experimenting the features related to colour information.

When asked about the *RGB* histogram, we could conclude that the tests subjects considered the feature useful (Figure F.2(d)). When asked if it gave a good idea of the range of colours being used and if the purpose of the dynamic bar was clear, we obtained a positive feedback (Figure F.2(a) and F.2(c)). However, we observed during the tests that

the purpose of the line indicating the amount of pixels near the boundaries was not well understood and that could be verified by the results ($Mo = 3$) in Figure F.2(b).

Regarding to the histogram implementation using the *Hue* channel, the results were positive. The users strongly agreed that the number of colours in the spectrum was adequate (Figure F.3(a)) and that in this representation, it would be easy to see the most used colours and its complementary to balance the image (Figure F.3(b)). However, when asked about its usefulness and if it gave a good idea of the colours being used, the average of responses was *agree* (Figure F.3(c) and F.3(d)). We were expecting *strongly agree*, as we though it would be much useful in comparison to regular histograms.

We could observe that many test subjects had difficulty in understand the saturation detection. Although they mostly answered *agree* when asked if it indicated the usage of a monochromatic filter and if the feature was useful (Figure F.4(a) and F.4(b)), we could detect that there were difficulties in understanding the concept of the feature and how to try it out. We already discussed in Section 3.2.1.3 that it would be difficult to find real-world scenarios to test this feature, and that could be verified during the testing stage.

About the colour template detector, the average of votes went to *agree* when asked about its usefulness and if the difference between scenarios with a monochromatic or complementary colour schemes was easy to understand (Figure F.5(c) and F.5(d)). However, we had low results when we asked if the colour scale was adequate, with many of the users answering *neutral* (Figure F.5(a)).

When asked about if the hue scoring feature combined well with the colour template detector and if it would reflect the simplicity of the scenario (Figure F.6(b) and F.6(c)), users almost unanimously answered *agree*. Since it was considered to reflect the simplicity, the users also *agreed* that it was useful (Figure F.6(d)) which was the main concern in this feature. However, most of the test subjects responded *neutral* when asked if the scale used was adequate (Figure F.6(a)).

Face detection is something that is already implemented in multiple photographing devices nowadays. We obtained very positive answers when we asked about the usefulness of using face detection with grids, exploring the triangular composition and the rule of odds (Figure F.7(a), F.7(b), F.7(c) and F.7(d)). Overall, the test subjects mostly choose *strongly agree* when asked if this feature was useful (Figure F.7(f)). We considered this feature to have the problem of showing too much information, however, many of the test subjects disagree as we can see in Figure F.7(e).

For the horizon detection, most users *agreed* that the detector gave the correct result and it was useful (Figure F.8(a) and F.8(c)). Being used in a real-time environment, we considered the performance to be an important part of these set of tools. When asked about if it performed well in real-time, the test subjects confirmed that this feature has poor performance and is to slow to be used in this conditions (Figure F.8(b)).

When asked about the detection of prominent lines, the results were pretty average. The test subjects *agreed* to be useful, the resulting lines corresponded to the leading lines in the scenario and that the visual cue would facilitate its usage in a composition (Figure

F.9).

Compared to all the other features, the object segmentation received the lowest scores. The test subjects responded *neutral*, when asked about its usefulness and if the result was accurate (Figure F.10(a) and F.10(b)). These answers were somewhat expected since the algorithm would need specific conditions to work well, such as having an object with a salient colour over a a simple background.

For the image simplicity feature, the test subjects gave the same scoring as the other features regarding its usefulness (Figure F.11(f)). When asked if bars or numbers were a good visual indicator, the users considered the bars to be better (Figure F.11(a) and F.11(b)). Since we implemented three different algorithms, we also asked about its results. The users considered the first implementation [LT08] to have the most accurate results (Figure F.11(c)). The other two algorithms [KWL; KTJ06] received the same score even though we considered the algorithm by Ke, Tang, and Jing [KTJ06] had the best results of all three implementations (Figure F.11(d) and F.11(e)).

In the end the user had to test the image balance feature. We had to explain to some test subjects what consisted this feature, as it was not obvious to everyone. The questionnaires results show that the users mostly *agreed* when asked if the symmetry axis was accurate, if the result was the expected and if it was useful (Figure F.12).

Overall we received positive feedback in most of the features. Although not perfect, some tests subjects considered the tools useful, specially the detector of prominent lines, the image simplicity detector and the histogram using the *Hue* channel. We also received a couple of critiques. Although, the interface was not our main concern, some users suggested that it should give more hints on what each feature actually does. For someone who does not have enough knowledge in photography, they might not take much benefit of these tools, and for that to happen it would be needed to implement an interface able to give useful hints about each one. The algorithms were considered to be too slow which caused the users to be impatient. Many of the times, we could verify that the user did not know if it had to wait for a result. It was also suggested to give a feedback while the algorithm is being run. One of the users suggested to use this features after the photo was taken to analyse and improve the photo, becoming useful in the learning of how to take photographs.

5

Conclusions and Future Work

The following chapter presents a brief analysis concerning the work accomplished with this dissertation, as well as what we intend to improve and implement in future applications.

5.1 Conclusion

This dissertation introduces a set a features based on rules used by professional photographers, applied in a real-time scenario when photographing. This was developed as a proof-of-concept for a smartphone.

After a thorough research, we could conclude that colour and composition were important properties to attain a better aesthetic result in photography. This solution provides means for gathering information about the scenario being photographed using these properties. Using these tools, the user is now able to understand what are the most used colours, if the scenario is using monochromatic or complementary tones, and the pureness of the colours being used. With this, the user can try and balance the colours.

The same goes for composition, as this thesis offers tools to detect important elements such as the horizon line, the subject of a scenario and detect its simplicity. As a helping tool to obtain an aesthetic result, the user can then take advantage of the suggestions and reorganize the composition of the scenario.

In our testing stage, we compared our modified versions of algorithms with the original ones, we tested the performance of each feature and realized user evaluation to assess its usefulness and accuracy. From the tests realized, we could conclude that these algorithms can be useful but there is still much room for improvement. A more user oriented interface, faster algorithms and better devices, would be needed.

It is important to refer that the final decision about the composition and colours is always of the final user. Although these features try to give a suggestion based on a set of rules that are considered to give aesthetic results, it is important to know when to ignore the suggestions and break the rules.

A limitation of this tool is the device in which it runs. Some of the features implemented are computationally heavy which makes them not perfect to be used in a real-time scenario and loses accuracy. This was confirmed by the performance tests that we realized.

There are still many improvements and functionalities that could be implemented, which is the subject of the following section.

5.2 Future Work

In the future, some of the features in our solution must be improved since running these algorithms in real-time is a heavy task for a mobile device. Including an external processing unit to the architecture, should be an option to take into consideration, as it could be a way to reduce the processing times.

Since the purpose of this thesis is to give some type of information about the aesthetics of a photo, in the future, we plan to implement a classification system to rate the photo after it was taken as it would encourage the user to try and understand the flaws in the scenario and correct them.

In the end, this set of tools would be fully integrated in a photographic application along with the a set of utilities for photo manipulation to be used after the capture. This utilities would include tools such as rotate, crop, apply a set of available filters, and auto white-balance that could be applied to specific parts of the image. If a more capable manipulation software was needed, after the capture, the user could annotate directly on the photo which areas should be corrected and what correction to apply. In this case, a subset of handwritten words that related to image corrections could be detected and applied directly. A more user oriented interface with hints explaining each feature would also be necessary to give the final user an idea of what to expect from it.

Finally, we believe that our solution contributed to this field and could be further improved and so the writing of a paper, or papers, would follow the delivery of this document.

Bibliography

[AHF06] A. E. Abdel-Hakim and A. A. Farag. "CSIFT: A SIFT descriptor with color invariant characteristics". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2006, pp. 1978–1983.

[Ada+10] A. Adams et al. "The Frankencamera: An Experimental Platform for Computational Photography". In: *ACM Trans. Graph.* 29.4 (July 2010), 29:1–29:12.

[Ado] Adobe. *Adobe Photoshop Express*. Accessed: January 2014. URL: `http://www.adoberevel.com/apps/photoshopexpress`.

[BTVG06] H. Bay, T. Tuytelaars, and L. Van Gool. "Surf: Speeded up robust features". In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[Ber83] J. Bertin. "Semiology of graphics: Diagrams, networks, maps (WJ Berg, Trans.)" In: *Madison, WI: The University of Wisconsin Press, Ltd* (1983).

[BSS10] S. Bhattacharya, R. Sukthankar, and M. Shah. "A framework for photo-quality assessment and enhancement based on visual aesthetics". In: *Proceedings of the international conference on Multimedia*. ACM. 2010, pp. 271–280.

[Bol13] S. Bolton. *Is your smartphone replacing your camera?* Accessed: January 2014. 2013. URL: `http://www.torontosun.com/2013/10/25/is-your-smartphone-replacing-your-camera`.

[Bre+12] S. Brewster et al. "Rethinking camera user interfaces". In: *Digital Photography VIII* 8299 (2012).

[BL07] M. Brown and D. G. Lowe. "Automatic panoramic image stitching using invariant features". In: *International Journal of Computer Vision* 74.1 (2007), pp. 59–73.

[BSBJ03] D. Butler, S. Sridharan, and V. M. Bove Jr. "Real-time adaptive background segmentation". In: *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*. Vol. 3. IEEE. 2003, pp. III–349.

[Can86] J. Canny. "A computational approach to edge detection". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 6 (1986), pp. 679–698.

[Che+11] M.-M. Cheng et al. "Global contrast based salient region detection". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 409–416.

[Cle04] M. Cleghorn. *Portrait Photography: Secrets of Posing & Lighting*. Lark Books, 2004.

[CO+06] D. Cohen-Or et al. "Color harmonization". In: *ACM Transactions on Graphics (TOG)* 25.3 (2006), pp. 624–630.

[DW10] R. Datta and J. Z. Wang. "ACQUINE: aesthetic quality inference engine-real-time automatic rating of photo aesthetics". In: *Proceedings of the international conference on Multimedia information retrieval*. ACM. 2010, pp. 421–424.

[Dat+06] R. Datta et al. "Studying aesthetics in photographic images using a computational approach". In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 288–301.

[DM08] P. E. Debevec and J. Malik. "Recovering high dynamic range radiance maps from photographs". In: *ACM SIGGRAPH 2008 classes*. ACM. 2008, p. 31.

[Gooa] Google. *Android NDK Documentation*. Accessed: January 2014. URL: `http://developer.android.com/tools/sdk/ndk/index.html`.

[Goob] Google. *Android SDK Documentation*. Accessed: January 2014. URL: `http://developer.android.com/sdk/index.html`.

[HLC11] J. Haber, S. Lynch, and S. Carpendale. "ColourVis: Exploring colour usage in paintings over time". In: *Proceedings of the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*. ACM. 2011, pp. 105–112.

[HB03] S. Herman and E. Bellers. *Adaptive segmentation of television images*. US Patent App. 10/538,338. 2003.

[Hoe05] F. Hoenig. "Defining computational aesthetics". In: *Proceedings of the First Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*. Eurographics Association. 2005, pp. 13–18.

[HR04] P. Howarth and S. Rüger. "Evaluation of texture features for content-based image retrieval". In: *Image and Video Retrieval*. Springer, 2004, pp. 326–334.

[IK88] J. Illingworth and J. Kittler. "A survey of the Hough transform". In: *Computer vision, graphics, and image processing* 44.1 (1988), pp. 87–116.

[Its] Itseez. *OpenCV*. Accessed: January 2014. URL: `http://opencv.org/`.

[Kam12] H. J. Kamps. *The Rules of Photography and When to Break Them*. CRC Press, 2012.

[KWL] C.-T. Kao, H.-F. Wu, and Y.-T. Liu. "Automatic Aesthetic Photo-Rating System". In: ().

[KTJ06] Y. Ke, X. Tang, and F. Jing. "The design of high-level features for photo quality assessment". In: *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2006, pp. 419–426.

[KV12] S. S. Khan and D. Vogel. "Evaluating visual aesthetics in photographic portraiture". In: *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*. Eurographics Association. 2012, pp. 55–62.

[LM07] J.-B. Labrune and W. Mackay. "Sketchcam: creative photography for children". In: *Proceedings of the 6th international conference on Interaction design and children*. ACM. 2007, pp. 153–156.

[LR87] J.-C. Lemagny and A. Rouillé. *A History of Photography*. Vol. 11. Cambridge University Press New York, 1987.

[Leo13] H. Leonard. *There Will Soon Be One Smartphone For Every Five People In The World*. Accessed: January 2014. 2013. URL: `http://www.businessinsider.com/15-billion-smartphones-in-the-world-22013-2`.

[LK06] W. H. Li and L. Kleeman. "Real time object tracking using reflectional symmetry and motion". In: *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*. IEEE. 2006, pp. 2798–2803.

[Lin+13] J. Linder et al. "PixelTone: a multimodal interface for image editing". In: *CHI'13 Extended Abstracts on Human Factors in Computing Systems*. ACM. 2013, pp. 2829–2830.

[Liu+10] L. Liu et al. "Optimizing photo composition". In: *Computer Graphics Forum*. Vol. 29. 2. Wiley Online Library. 2010, pp. 469–478.

[Low99] D. G. Lowe. "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[LWT11] W. Luo, X. Wang, and X. Tang. "Content-based photo quality assessment". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2206–2213.

[LT08] Y. Luo and X. Tang. "Photo and video quality evaluation: Focusing on the subject". In: *Computer Vision–ECCV 2008*. Springer, 2008, pp. 386–399.

[Man07] B. Manav. "Color-emotion associations and color preferences: A case study for residences". In: *Color Research & Application* 32.2 (2007), pp. 144–150.

[MM96] B. S. Manjunath and W.-Y. Ma. "Texture features for browsing and retrieval of image data". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 18.8 (1996), pp. 837–842.

[MGK00] J. Matas, C. Galambos, and J. Kittler. "Robust detection of lines using the progressive probabilistic hough transform". In: *Computer Vision and Image Understanding* 78.1 (2000), pp. 119–137.

[MSH05] T. G. McGee, R. Sengupta, and K. Hedrick. "Obstacle detection for small autonomous aircraft using sky segmentation". In: *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE. 2005, pp. 4679–4684.

[Mic] Microsoft. *Microsoft Photosynth*. Accessed: January 2014. URL: http://photosynth.net/.

[Nób13] R. P. d. S. Nóbrega. "Interactive acquisition of spatial information from images for multimedia applications". MA thesis. Faculdade de Ciências e Tecnologia, 2013.

[Pul+09] K. Pulli et al. "Mobile visual computing". In: *Ubiquitous Virtual Reality, 2009. ISUVR'09. International Symposium on*. IEEE. 2009, pp. 3–6.

[RD06] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection". In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 430–443.

[RKB04] C. Rother, V. Kolmogorov, and A. Blake. "Grabcut: Interactive foreground extraction using iterated graph cuts". In: *ACM Transactions on Graphics (TOG)*. Vol. 23. 3. ACM. 2004, pp. 309–314.

[SGS08] K. E. van de Sande, T. Gevers, and C. G. Snoek. "A comparison of color features for visual concept classification". In: *Proceedings of the 2008 international conference on Content-based image and video retrieval*. ACM. 2008, pp. 141–150.

[San10] J. Santos. *Fotografia: Luz, Exposição, Composição, Equipamento*. Edições Centro Atlântico, 2010.

[Sha13] V. Sharon. *Get started with Android 4.4 KitKat's advanced photo editor*. Accessed: January 2014. 2013. URL: http://howto.cnet.com/8301-11310_39-57611362-285/get-started-with-android-4.4-kitkats-advanced-photo-editor/.

[Sze11] R. Szeliski. *Computer vision: algorithms and applications*. Springer, 2011.

[Sze12] R. Szeliski. "Open platforms for computational photography: technical perspective." In: *Commun. ACM* 55.11 (2012), p. 89. URL: http://dblp.uni-trier.de/db/journals/cacm/cacm55.html#Szeliski12.

[SS97] R. Szeliski and H.-Y. Shum. "Creating full view panoramic image mosaics and environment maps". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 1997, pp. 251–258.

[Ton+05] H. Tong et al. "Classification of digital photos taken by photographers or home users". In: *Advances in Multimedia Information Processing-PCM 2004*. Springer, 2005, pp. 198–205.

[VJ11] A Vavilin and K.-H. Jo. "Fast HDR image generation from multi-exposed multiple-view LDR images". In: *Visual Information Processing (EUVIP), 2011 3rd European Workshop on*. IEEE. 2011, pp. 105–110.

[Vaz] F. G. Vazquez. *Camera FV-5*. Accessed: January 2014. URL: `http://www.camerafv5.com`.

[VJ01] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2001, pp. I–511.

[Yan+04] T. Yang et al. "Real-time and accurate segmentation of moving objects in dynamic scene". In: *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*. ACM. 2004, pp. 136–143.

[Yeh+10] C.-H. Yeh et al. "Personalized photograph ranking and selection system". In: *Proceedings of the international conference on Multimedia*. ACM. 2010, pp. 211–220.

[ZW+08] B. Zafarifar, H. Weda, et al. "Horizon detection based on sky-color and edge features". In: *Electronic Imaging 2008*. International Society for Optics and Photonics. 2008, pp. 682220–682220.

[Zaf+06] B. Zafarifar et al. "Blue sky detection for picture quality enhancement". In: *Advanced Concepts for Intelligent Vision Systems*. Springer. 2006, pp. 522–532.

Algorithms Execution Times

	Execution Time (ms)	
#	Success	Fail
1	52	117
2	30	105
3	35	115
4	31	139
5	40	119
6	58	111
7	50	116
8	30	115
9	16	117
10	32	178
Average (ms)	37.40	118.63
Standard deviation (ms)	8.98	8.55
	Total Time (ms)	
Average	75.92	147.77
Standard deviation	16.55	29.78

Table A.1: Total time spent processing a frame and displaying the result, and execution times of the saturation detection algorithm in cases where the scenario is/isn't saturated.

#	Execution Time (ms)
1	21
2	46
3	32
4	43
5	64
6	42
7	28
8	31
9	65
10	37
Average (ms)	40.38
Standard deviation (ms)	11.48
	Total Time (ms)
Average	85.60
Standard deviation	27.84

Table A.2: Total time spent processing a frame and displaying the result, and execution time of calculating the colour histograms using the *RGB* channels and grayscale.

#	Execution Time (ms)
1	68
2	109
3	147
4	95
5	71
6	83
7	60
8	114
9	77
10	106
Average (ms)	98.13
Standard deviation (ms)	27.38
	Total Time (ms)
Average	152.17
Standard deviation	20.63

Table A.3: Total time spent processing a frame and displaying the result, and execution time of calculating the histograms using the hue channel.

#	Execution Time (ms)
1	67
2	117
3	100
4	76
5	60
6	86
7	61
8	54
9	74
10	109
Average (ms)	77.63
Standard deviation (ms)	21.98
	Total Time (ms)
Average	128.28
Standard deviation	24.26

Table A.4: Total time spent processing a frame and displaying the result, and execution time of calculating the colour templates.

#	Execution Time (ms)
1	66
2	55
3	41
4	67
5	54
6	67
7	57
8	41
9	53
10	61
Average (ms)	55.38
Standard deviation (ms)	10.14
	Total Time (ms)
Average	56.75
Standard deviation	8.29

Table A.5: Total time spent processing a frame and displaying the result, and execution time of calculating the score based on the number of colours being used from the Hue channel.

	Execution Time (ms)			
#	**1 Face**	**3 Faces**	**1 Face w/ Comp.**	**3 Faces w/ Comp.**
1	72	48	52	63
2	64	88	34	99
3	41	30	55	50
4	72	212	33	53
5	47	89	39	59
6	41	30	54	52
7	73	33	50	40
8	63	33	31	66
9	70	31	60	56
10	52	67	60	42
Average (ms)	60.13	52.38	47.13	55.13
Standard deviation (ms)	11.99	25.50	10.32	7.64
			Total Time (ms)	
Average	93.24	85.99	78.60	92.99
Standard deviation	14.78	31.23	9.79	11.95

Table A.6: Total time spent processing a frame and displaying the result, and execution time of finding one or three faces while using or not the composition rules.

#	**Execution Time (ms)**
1	193
2	187
3	257
4	209
5	198
6	191
7	184
8	197
9	193
10	109
Average (ms)	194
Standard deviation (ms)	7.65
	Total Time (ms)
Average	239.52
Standard deviation	8.18

Table A.7: Total time spent processing a frame and displaying the result, and execution time of calculating the saliency map and segmenting the object.

	Execution Time (ms)		
#	**Method 1 [LT08]**	**Method 2 [KWL]**	**Method 3 [KTJ06]**
1	125	139	250
2	110	82	191
3	168	113	216
4	95	83	175
5	118	108	162
6	144	79	148
7	94	102	147
8	116	142	167
9	154	104	148
10	174	119	139
Average (ms)	128.75	106.25	169.25
Standard deviation (ms)	24.48	18.65	24.34
	Total Time (ms)		
Average	172.29	143.14	221.28
Standard deviation	33.59	32.33	37.62

Table A.8: Total time spent processing a frame and displaying the result, and execution time of calculating the simplicity with each one of the tested methods.

	Execution Time (ms)		
#	T **= 40**	T **= 100**	T **= 160**
1	354	283	489
2	374	313	384
3	478	314	382
4	387	406	376
5	381	321	472
6	366	299	389
7	349	303	404
8	373	281	411
9	412	332	432
10	380	320	363
Average (ms)	378.38	310.63	406.25
Standard deviation (ms)	16.94	15.24	32.31
	Total Time (ms)		
Average	434.18	356.56	455.28
Standard deviation	50.29	25.10	43.41

Table A.9: Total time spent processing a frame and displaying the result, and execution time of finding the most relevant lines in a scenario with different thresholds (T).

#	Execution Time (ms)
1	1721
2	2848
3	2874
4	2983
5	2897
6	2731
7	2562
8	1704
9	1707
10	1729
Average (ms)	2383.63
Standard deviation (ms)	560.30
	Total Time (ms)
Average	2659.29
Standard deviation	485.30

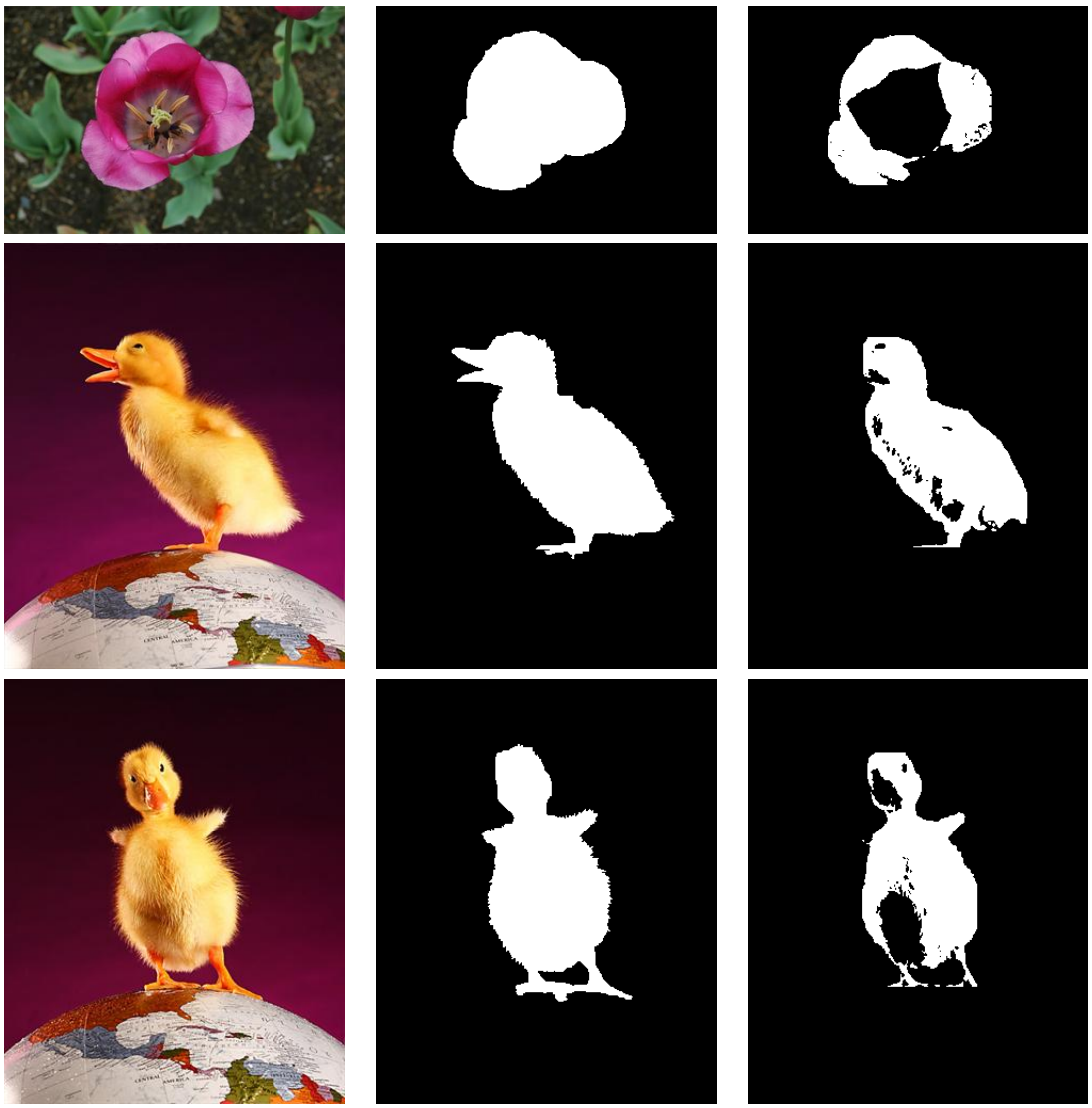
Table A.10: Total time spent processing a frame and displaying the result, and execution time of finding the horizon line.

#	Execution Time (ms)
1	361
2	332
3	336
4	357
5	327
6	331
7	296
8	349
9	364
10	296
Average (ms)	336.13
Standard deviation (ms)	20.55
	Total Time (ms)
Average	593.68
Standard deviation	111.35

Table A.11: Total time spent processing a frame and displaying the result, and execution time of calculating the image balance.

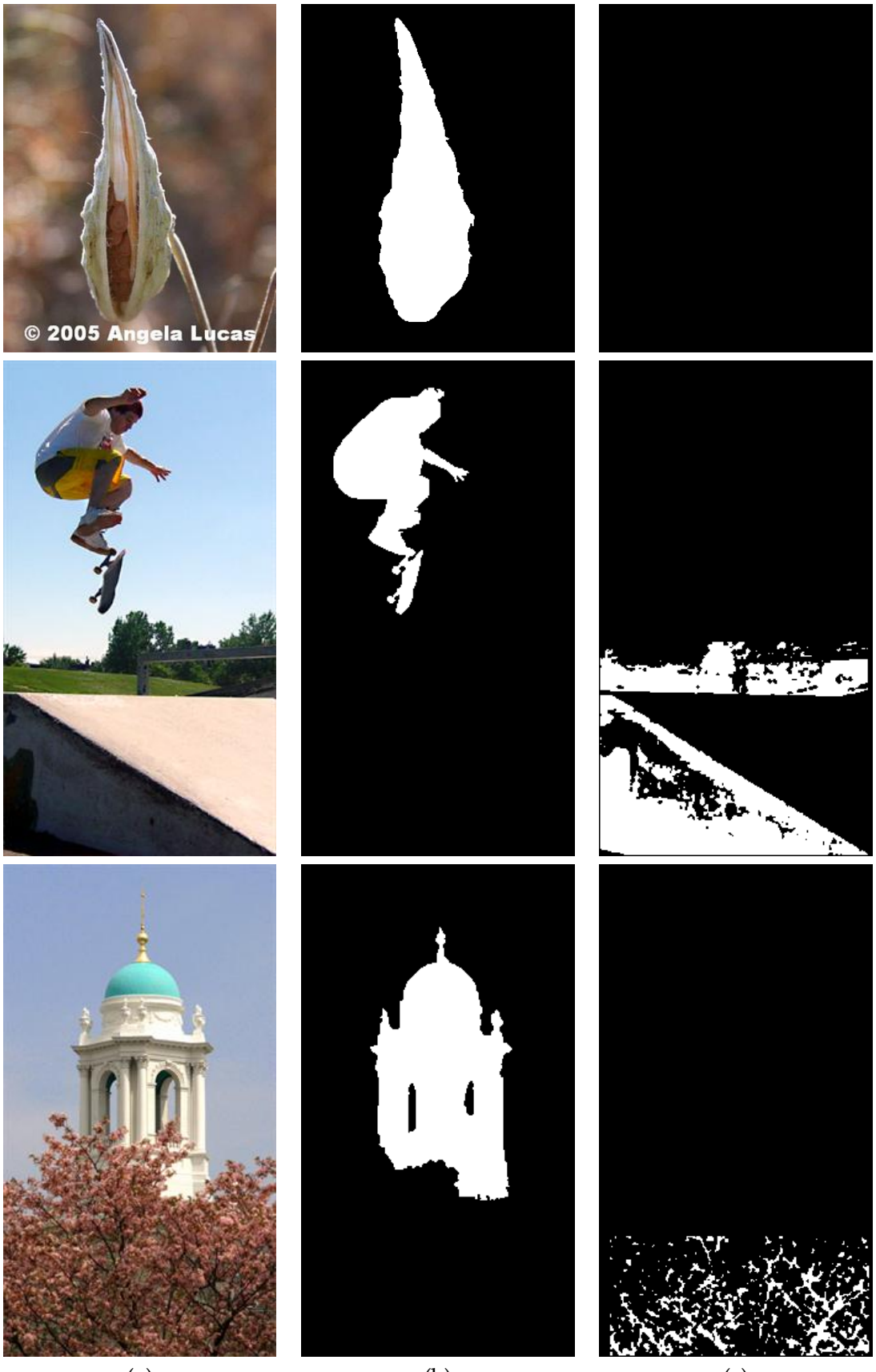
B

Object Segmentation Results Comparison



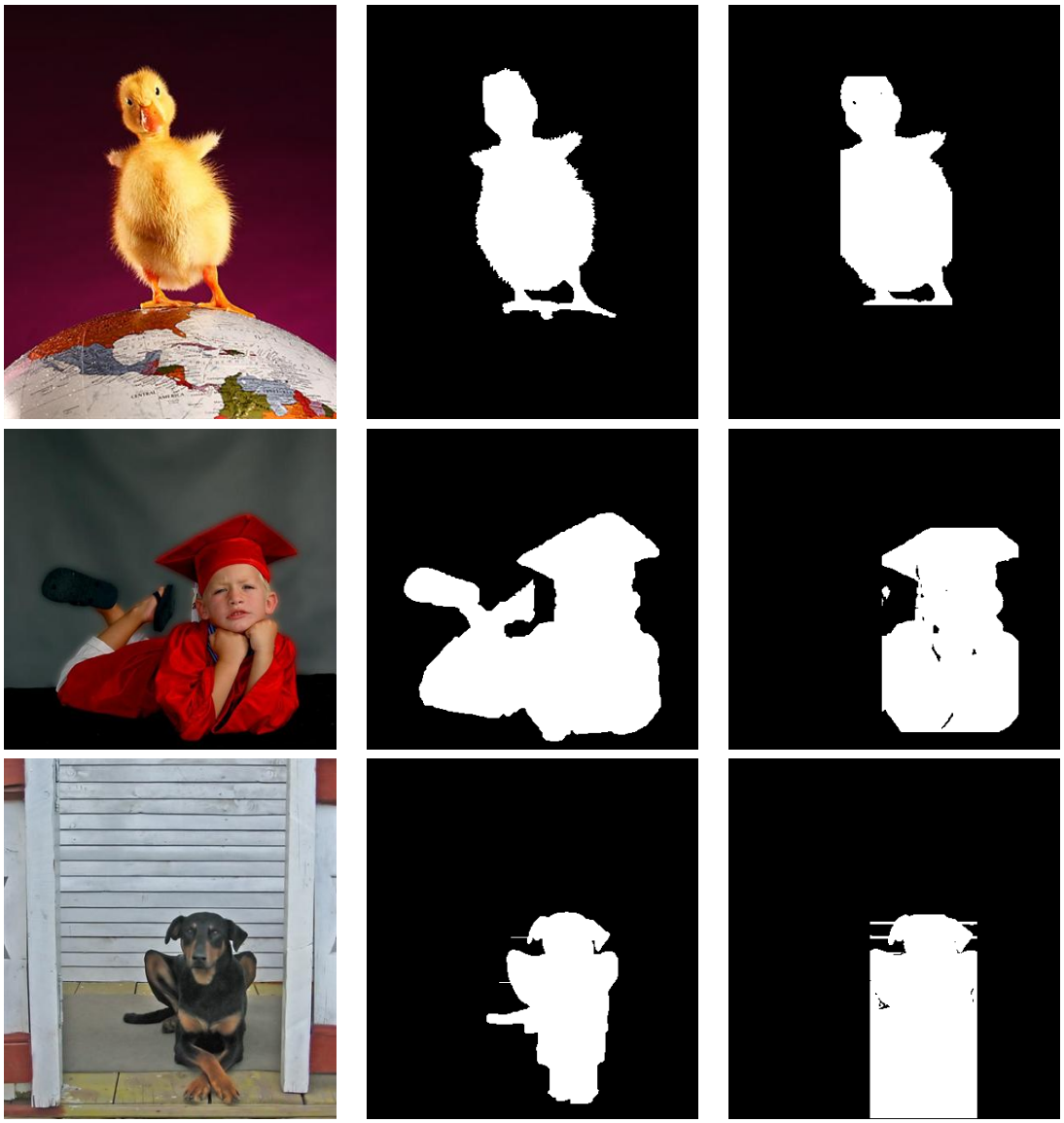
(a) (b) (c)

Figure B.1: Example of correct object segmentation of the implemented algorithm (c) only using the pixels considered as foreground in comparison to the original algorithm (b) [Che+11].



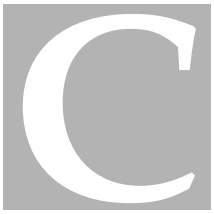
(a) (b) (c)

Figure B.2: Example of failed object segmentation of the implemented algorithm (c) only using the pixels considered as foreground in comparison to the original algorithm (b) [Che+11].



(a) (b) (c)

Figure B.3: Example of correct object segmentation of the implemented algorithm (c) using the pixels considered as foreground and background in comparison to the original algorithm (b) [Che+11].



Colour Template Detection Results Comparison

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

Figure C.1: Source images (a, d, g, j) and templates detected by the original algorithm implemented by Cohen-Or et al. [CO+06] (b, e, h, k) in comparison to the templates detected by our algorithm (c, f, i, l).

D

Horizon Detection Results

LANDSCAPe	Real		Algorithm		Absolute Error		Relative Error	
	Position	**Angle**	**Position**	**Angle**	**Position**	**Angle**	**Position**	**Angle**
#1	336.5	0.00098	309	-0.00781	27.5	0.00879	8.17%	899.98%
#2	305.5	0.03222	283	0.14258	22.5	0.11037	7.36%	342.59%
#3	151.5	-0.01269	151	-0.03417	0.5	0.02147	0.33%	169.14%
#4	152.5	0.00488	155	-0.05317	2.5	0.05806	1.64%	1188.98%
#5	169.5	0.00488	165	0.00098	4.5	0.00391	2.65%	80.00%
#6	726	0.07858	853	0.00130	127	0.07728	17.49%	98.34%
#7	170.5	0.00293	165	-0.06923	5.5	0.07215	3.23%	2462.89%
#8	220.5	-0.06728	233	-0.07991	12.5	0.01263	5.67%	18.77%
#9	166.5	0.00684	155	0.00879	11.5	0.00195	6.91%	28.57%
#10	168.5	0.02246	168	0.00879	0.5	0.01367	0.30%	60.86%
#11	263	-0.00195	236	-0.01660	27	0.01465	10.27%	749.92%
#12	337.5	0.01465	331	0.00879	6.5	0.00586	1.93%	40.00%
#13	440	-0.00195	325	-0.91620	115	0.91424	26.14%	46809.28%
#14	361	0.02343	449	0.50979	88	0.48636	24.38%	2075.50%
#15	161.5	0.03612	204	0.03514	42.5	0.00098	26.32%	2.70%
						Average	9.52%	3668.50%

Table D.1: Detection results of the implemented algorithm for each of the test images labelled as *landscape*. Absolute error indicates the absolute difference between each parameter and the manually-annotated horizon line. Relative error is the absolute error normalized to the corresponding parameter of the manually-annotated horizon line.

SEASCAPE	Real		Algorithm		Absolute Error		Relative Error	
	Position	Angle	Position	Angle	Position	Angle	Position	Angle
#1	313.5	0.00488	323	0.01758	9.5	0.01269	3.03%	259.97%
#2	210.5	0.06534	231	0.00098	20.5	0.06436	9.74%	98.51%
#3	354.5	0.01660	371	0.02636	16.5	0.00976	4.65%	58.80%
#4	429.5	0.02441	295	0.03563	134.5	0.01122	31.32%	45.97%
#5	165	0.03332	167	0.06048	2	0.02716	1.21%	81.51%
#6	276	0.02343	286	0.00879	10	0.01464	3.62%	62.49%
#7	173	0.02148	167	0.06048	6	0.03900	3.47%	181.56%
#8	258.5	0.00098	236	-0.01660	22.5	0.01758	8.70%	1799.85%
						Average:	8.22%	323.58%

Table D.2: Detection results of the implemented algorithm for each of the test images labelled as *seacape*. Absolute error indicates the absolute difference between each parameter and the manually-annotated horizon line. Relative error is the absolute error normalized to the corresponding parameter of the manually-annotated horizon line.

HORIZON	Real		Algorithm		Absolute Error		Relative Error	
	Position	Angle	Position	Angle	Position	Angle	Position	Angle
#1	465.5	-0.00684	209	0.03514	256.5	0.04198	55.10%	614.08%
#2	563.5	-0.02636	427	-0.01660	136.5	0.00976	24.22%	37.03%
#3	156.5	-0.00488	472	0.08718	315.5	0.09206	201.60%	1885.48%
#4	427	0.02343	184	0.03563	243	0.01220	56.91%	52.05%
#5	171.5	0.00098	373	0.21955	201.5	0.21857	117.49%	22381.74%
#6	304	-0.00586	254	0.46365	50	0.46951	16.45%	8013.01%
#7	464	0.00391	437	-0.03465	27	0.03856	5.82%	987.15%
#8	451	0.00596	452	0.02683	1	0.02087	0.22%	349.90%
#9	147	0.00977	222	0.00098	75	0.00879	51.02%	90.00%
						Average:	59.22%	4224.54%

Table D.3: Detection results of the implemented algorithm for each of the test images labelled as *horizon*. Absolute error indicates the absolute difference between each parameter and the manually-annotated horizon line. Relative error is the absolute error normalized to the corresponding parameter of the manually-annotated horizon line.

Users Questionnaires

User Data

1. **Gender?** M □ F □
2. **Age?** _____

User's Past Experience

3. **Do you take photographs?** Yes □ No □
4. **Do you have any knowledge in photography?** Yes □ No □
5. **If yes, how do you classify your knowledge in photography?** Professional □ Amateur □
6. **If yes, what kind of device do you normally use to take photographs?**
   - □ Still cameras Never □—□—□—□—□ Regularly
   - □ Instant cameras Never □—□—□—□—□ Regularly
   - □ DSLR cameras Never □—□—□—□—□ Regularly
   - □ Camera phone Never □—□—□—□—□ Regularly
   - □ Other: _____ Never □—□—□—□—□ Regularly

About this questionnaire

RGB Histograms

7. **The purpose of the dynamic bar is clear**
   Strongly disagree □—□—□—□—□ Strongly agree
8. **The purpose of the growing line is clear**
   Strongly disagree □—□—□—□—□ Strongly agree
9. **Gives a good idea of the range of colours being used**
   Strongly disagree □—□—□—□—□ Strongly agree
10. **This tool is useful in a real scenario**
    Strongly disagree □—□—□—□—□ Strongly agree

Hue Colour Histogram

11. **The number of colours in the spectrum is adequate**
    Strongly disagree □—□—□—□—□ Strongly agree
12. **Can easily see the most used colour and which complementary colour should be used to balance the image**
    Strongly disagree □—□—□—□—□ Strongly agree
13. **Gives a good idea of the range of colours being used**
    Strongly disagree □—□—□—□—□ Strongly agree
14. **This tool is useful in a real scenario**
    Strongly disagree □—□—□—□—□ Strongly agree

Saturation Detection

15. **The visual cue for this feature encourages the usage of a monochromatic filter**
    Strongly disagree □—□—□—□—□ Strongly agree
16. **This tool is useful in a real scenario**
    Strongly disagree □—□—□—□—□ Strongly agree

Colour Template Detector

17. **The colour scale is adequate**

Strongly disagree □—□—□—□—□ Strongly agree

18. **This feature foes along well with the hue scoring feature**

Strongly disagree □—□—□—□—□ Strongly agree

19. **It is easy to understand the difference between scenarios with a monochromatic or complementary colour schemes**

Strongly disagree □—□—□—□—□ Strongly agree

20. **This tool is useful in a real scenario**

Strongly disagree □—□—□—□—□ Strongly agree

Hue Count Score

21. **The scoring scale is adequate**

Strongly disagree □—□—□—□—□ Strongly agree

22. **This feature goes along well with the colour template detection**

Strongly disagree □—□—□—□—□ Strongly agree

23. **The scoring can reflect the simplicity of a scenario**

Strongly disagree □—□—□—□—□ Strongly agree

24. **This tool is useful in a real scenario**

Strongly disagree □—□—□—□—□ Strongly agree

Face Detection

25. **This feature is useful when used with rule of thirds or golden rule of thirds**

Strongly disagree □—□—□—□—□ Strongly agree

26. **The connection between three faces is useful to explore the rule of odds**

Strongly disagree □—□—□—□—□ Strongly agree

27. **The connections between three faces is useful to explore the triangular composition**

Strongly disagree □—□—□—□—□ Strongly agree

28. **The suggestive placements are useful**

Strongly disagree □—□—□—□—□ Strongly agree

29. **When used with a grid, it is too much visual information**

Strongly disagree □—□—□—□—□ Strongly agree

30. **This tool is useful in a real scenario**

Strongly disagree □—□—□—□—□ Strongly agree

Horizon Detection

31. **The resulting line gives the expected horizon line**

Strongly disagree □—□—□—□—□ Strongly agree

32. **This feature performs well in real-time**

Strongly disagree □—□—□—□—□ Strongly agree

33. **This tool is useful in a real scenario**

Strongly disagree □—□—□—□—□ Strongly agree

Main Lines Detection

34. **The resulting lines correspond to the leading lines in the scenario**

Strongly disagree □—□—□—□—□ Strongly agree

35. **The visual cue facilitates the usage of leading lines in a composition**

Strongly disagree □—□—□—□—□ Strongly agree

36. **This tool is useful in a real scenario**

Strongly disagree □—□—□—□—□ Strongly agree

Generic Object Segmentation

37. The object segmentation is accurate

Strongly disagree □—□—□—□—□ Strongly agree

38. This tool is useful in a real scenario

Strongly disagree □—□—□—□—□ Strongly agree

Image Simplicity Detection

39. Bars are a good visual indicator for this feature

Strongly disagree □—□—□—□—□ Strongly agree

40. Numbers are a good visual indicator for this feature

Strongly disagree □—□—□—□—□ Strongly agree

41. The first method displays a good result for simplicity of the scenario

Strongly disagree □—□—□—□—□ Strongly agree

42. The second method displays a good result for simplicity of the scenario

Strongly disagree □—□—□—□—□ Strongly agree

43. The third method displays a good result for simplicity of the scenario

Strongly disagree □—□—□—□—□ Strongly agree

44. This tool is useful in a real scenario

Strongly disagree □—□—□—□—□ Strongly agree

Image Balance Detection

45. The symmetry axis is accurate

Strongly disagree □—□—□—□—□ Strongly agree

46. The visual cue displays a correct result from what was expected

Strongly disagree □—□—□—□—□ Strongly agree

47. This tool is useful in a real scenario

Strongly disagree □—□—□—□—□ Strongly agree

Suggestions and Comments

48. Any comments and suggestions are appreciated.

_____

_____

_____

_____

_____

_____

Users Questionnaire Results

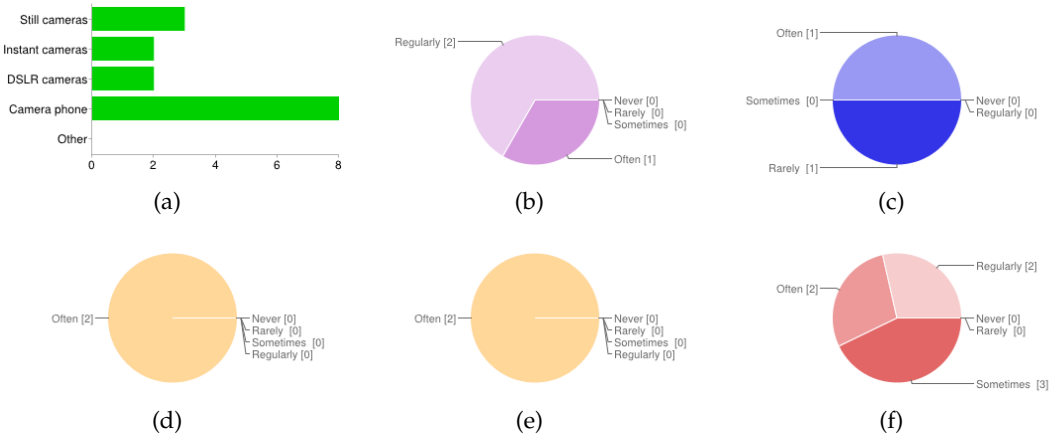
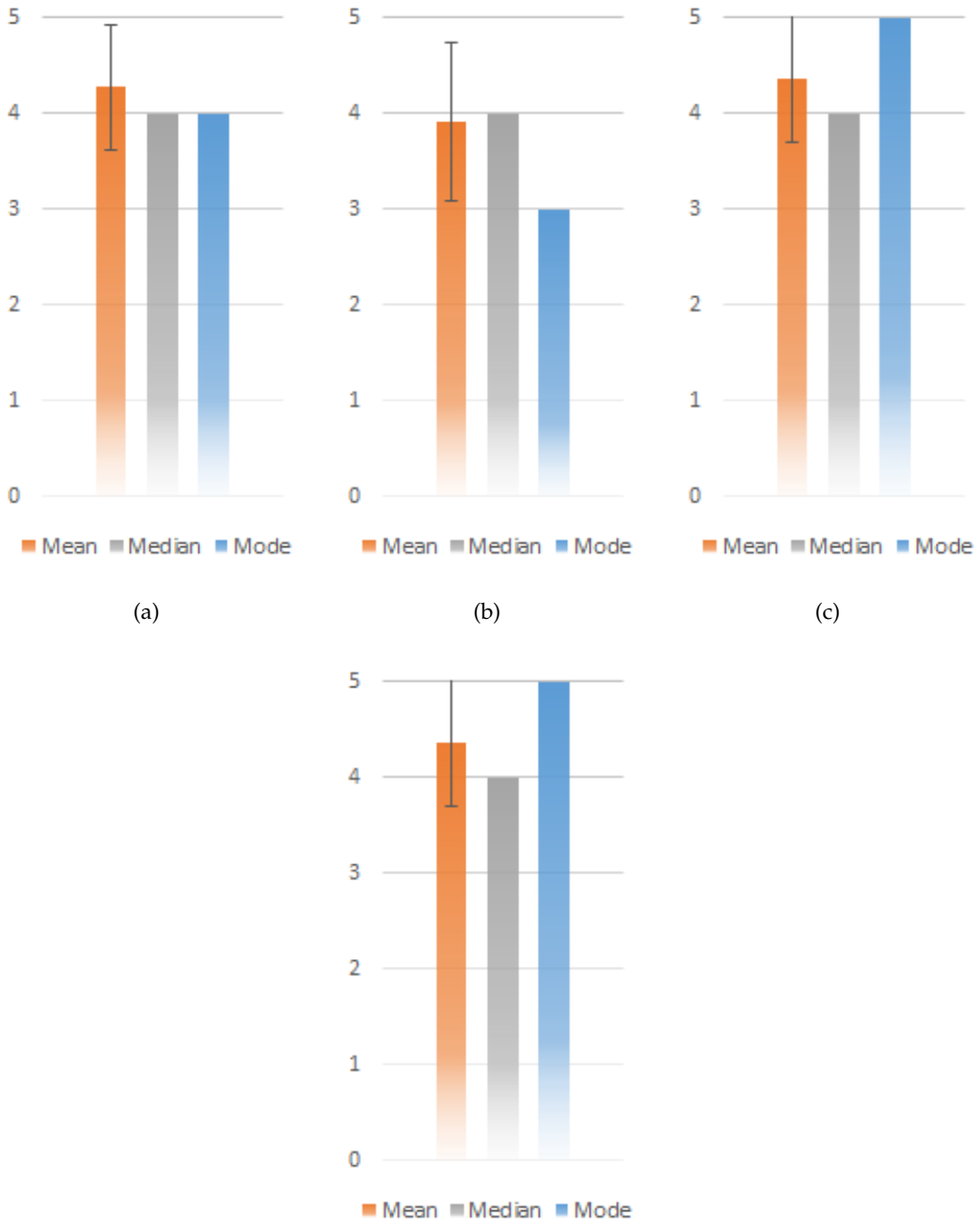


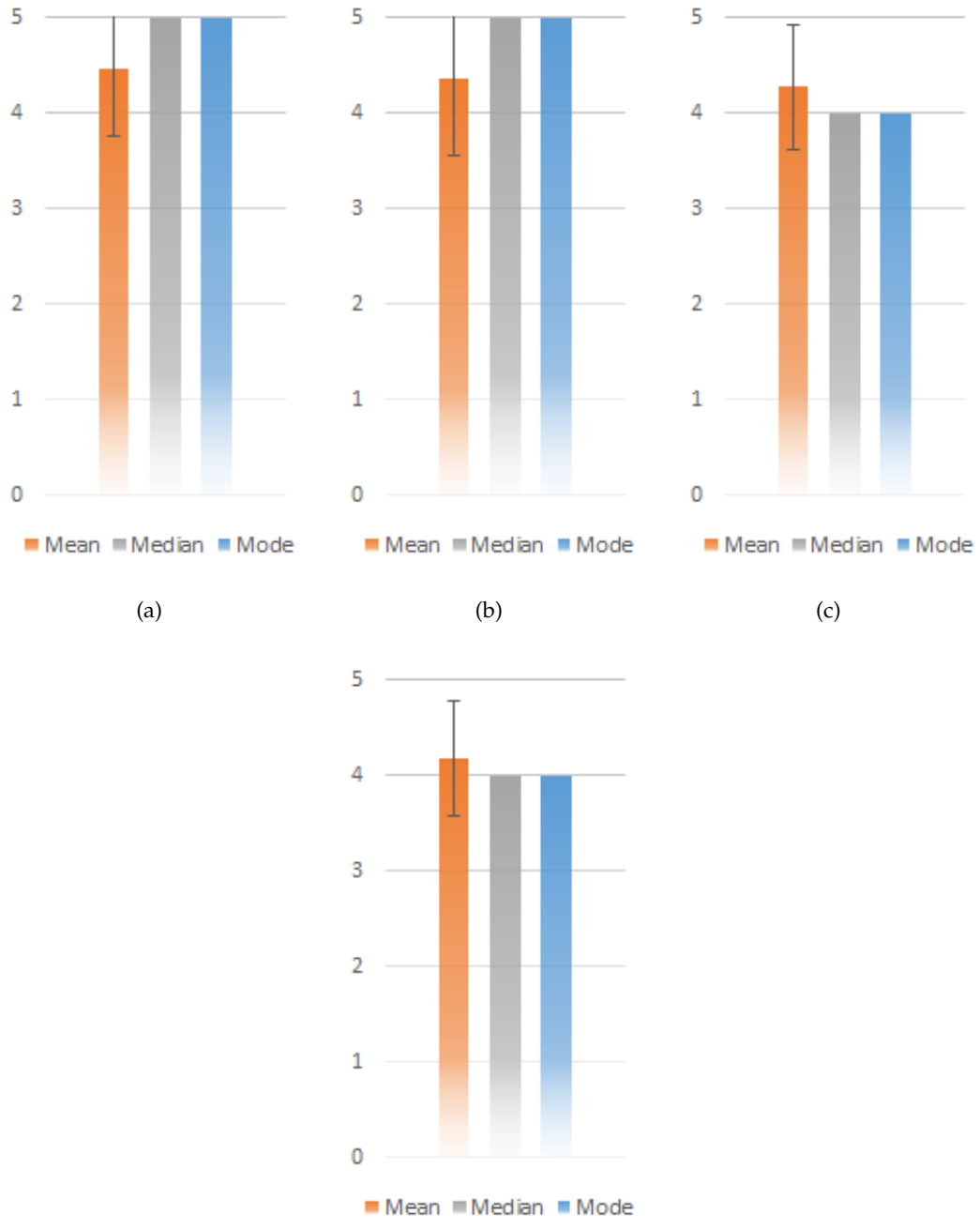
Figure F.1: Results to the question Q6.



(a) (b) (c)

(d)

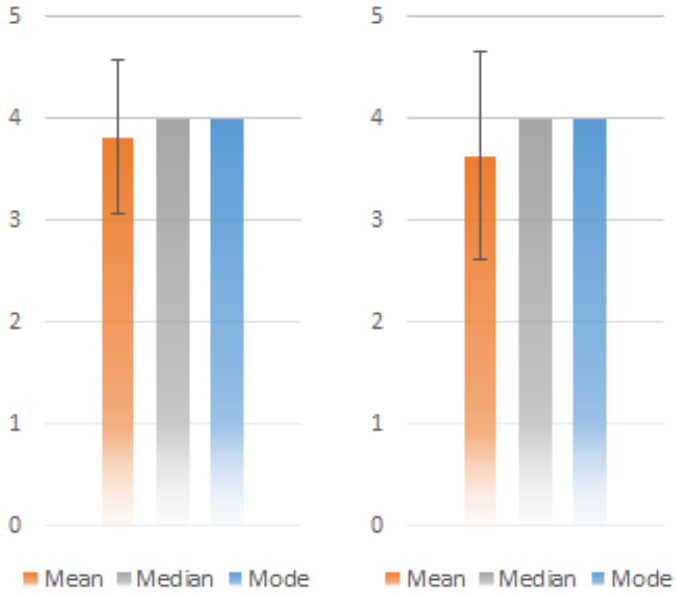
Figure F.2: Results to the question Q7 (a), Q8 (b), Q9 (c), Q10 (d).



(a) (b) (c)

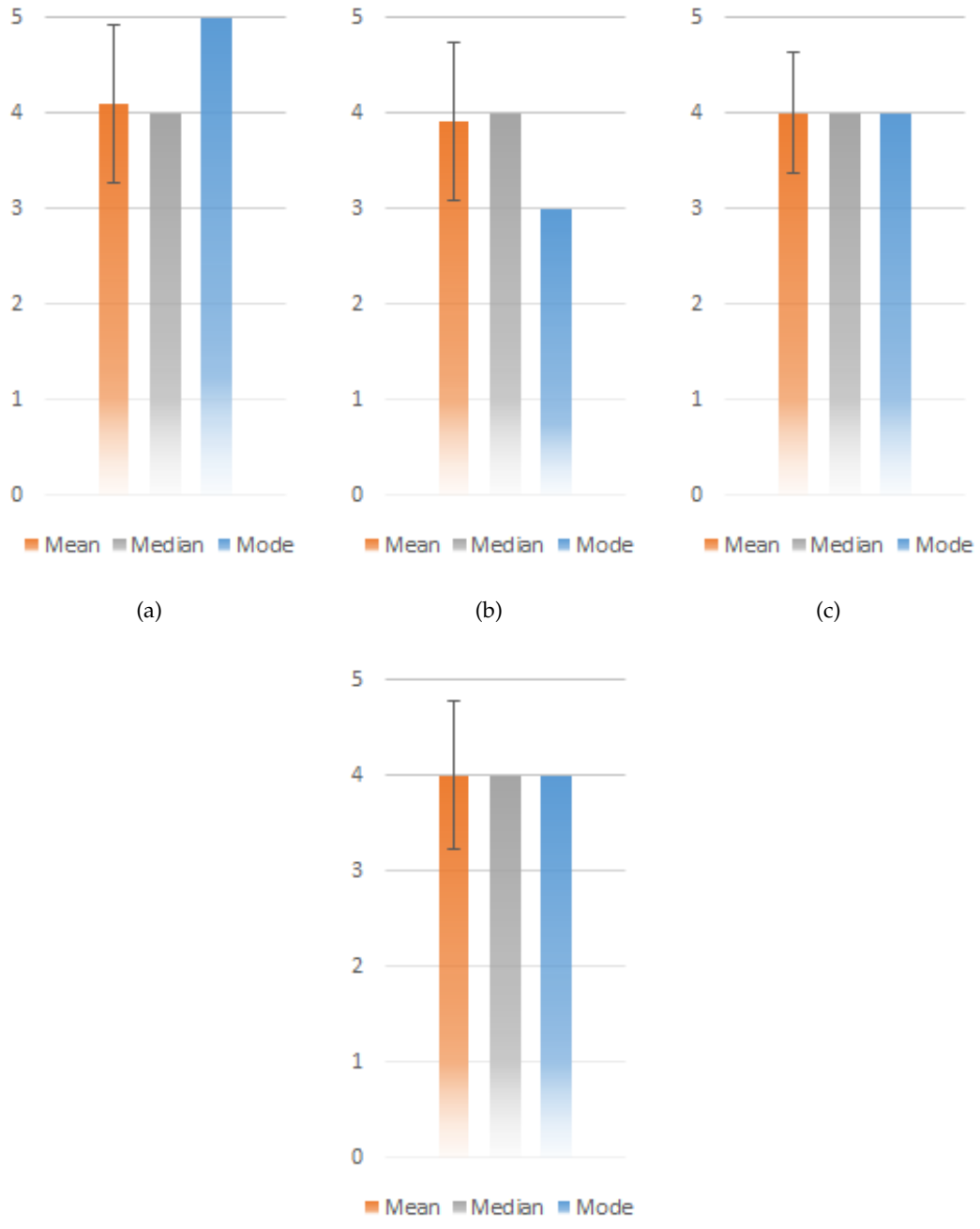
(d)

Figure F.3: Results to the question Q11 (a), Q12 (b), Q13 (c), Q14 (d).



(a) (b)

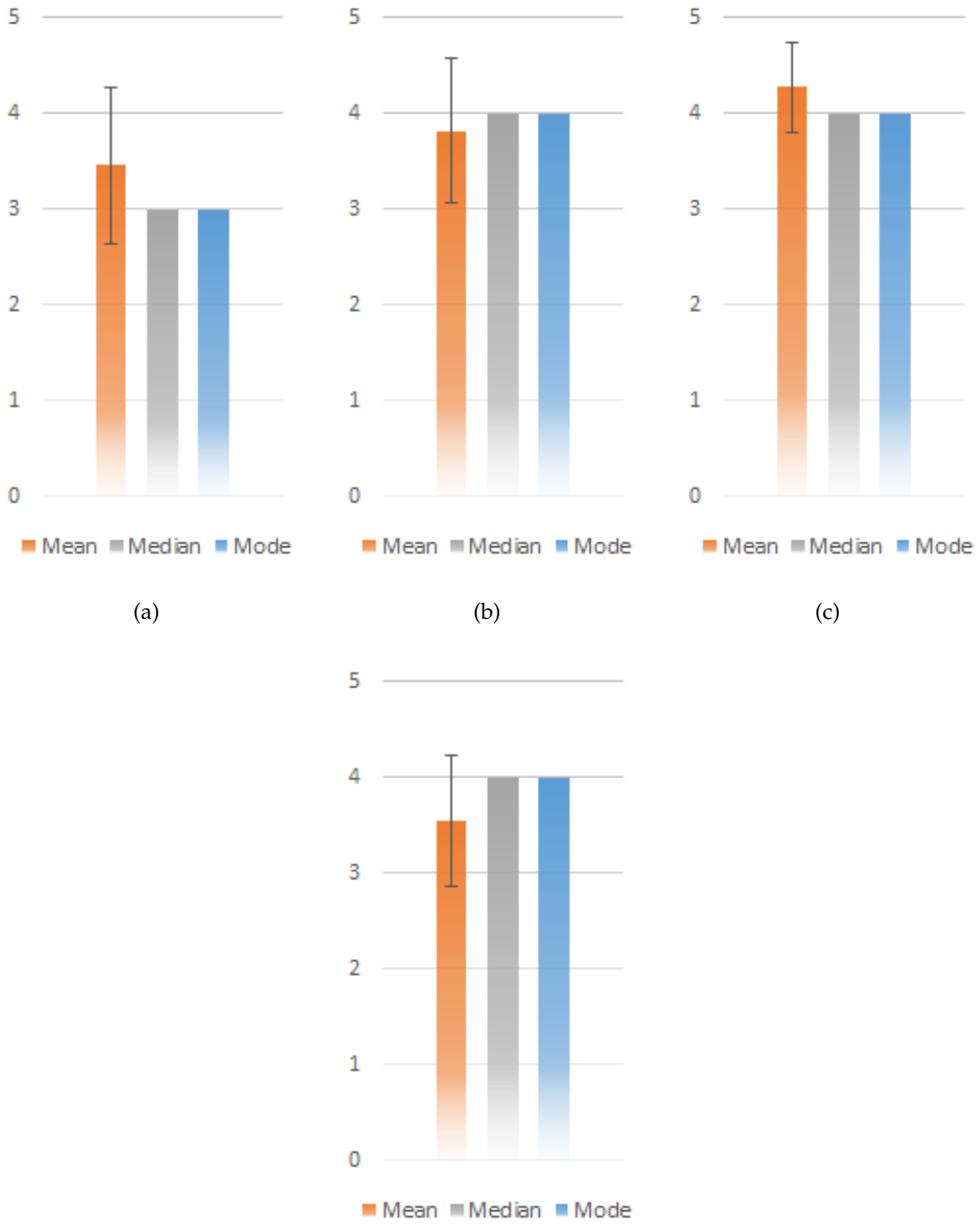
Figure F.4: Results to the question Q15 (a), Q16 (b).



(a) (b) (c)

(d)

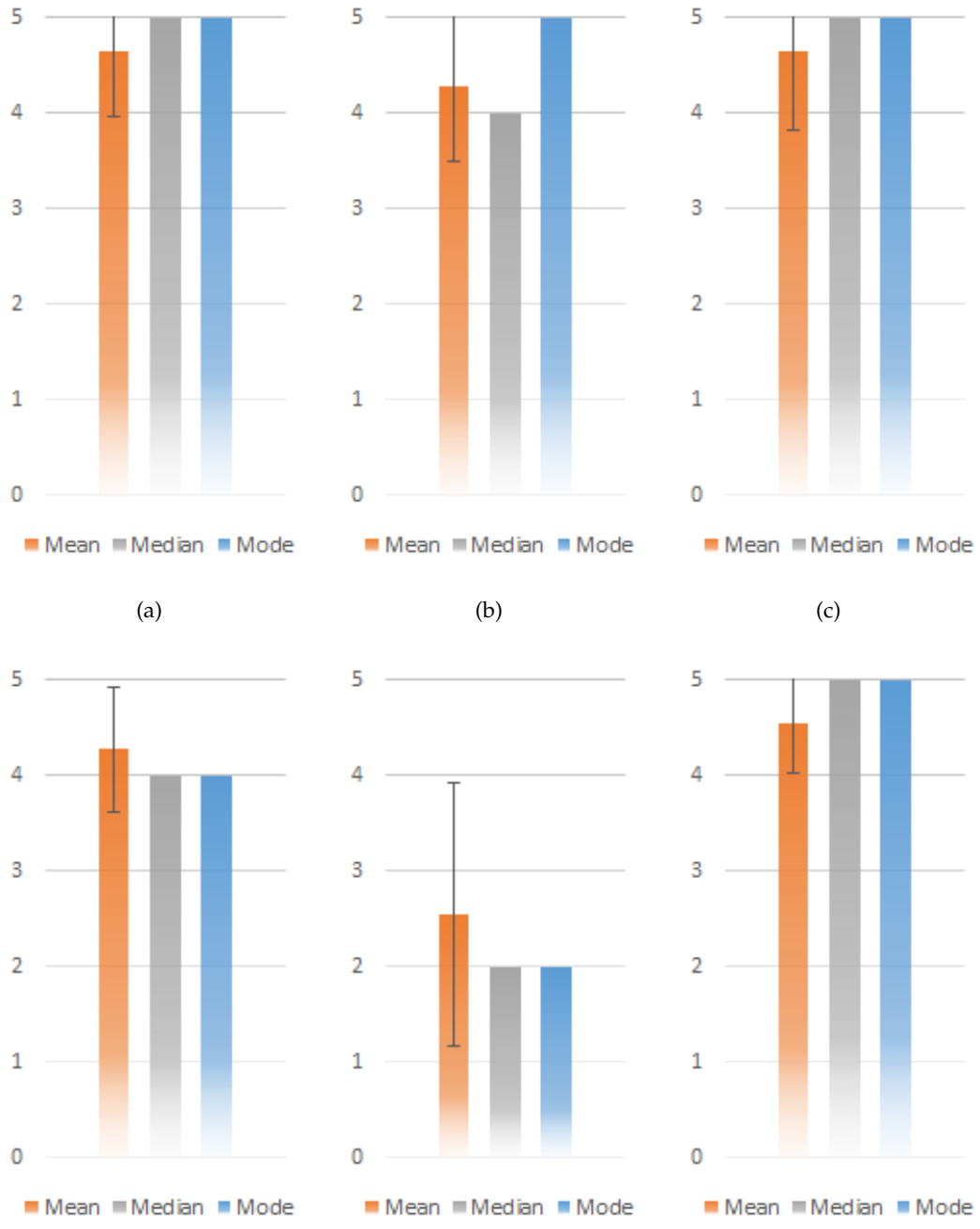
Figure F.5: Results to the question Q17 (a), Q18 (b), Q19 (c), Q20 (d).



(a) (b) (c)

(d)

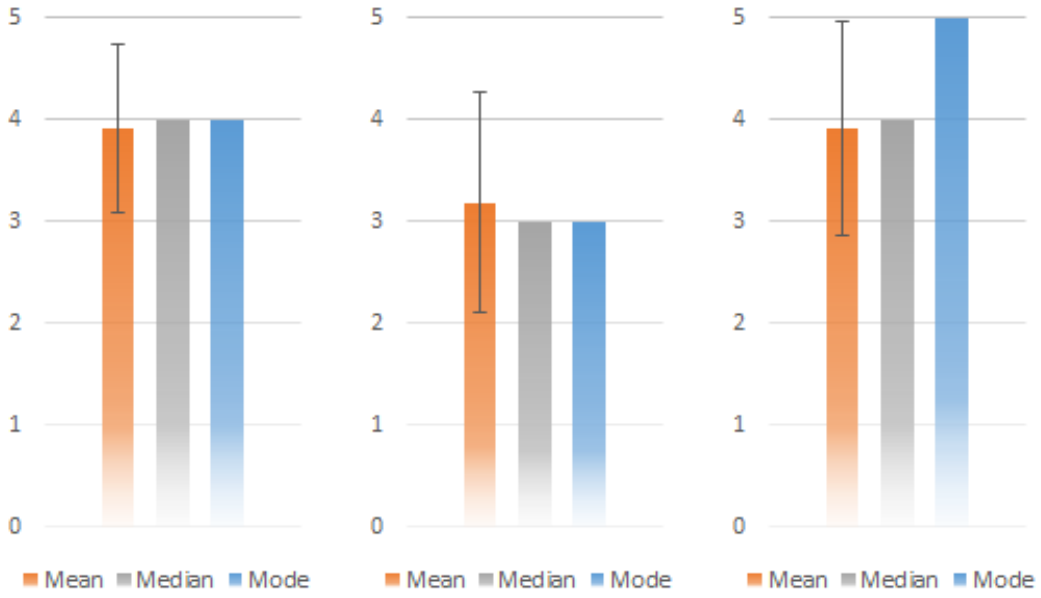
Figure F.6: Results to the question Q21 (a), Q22 (b), Q23 (c), Q24 (d).



(a) (b) (c)

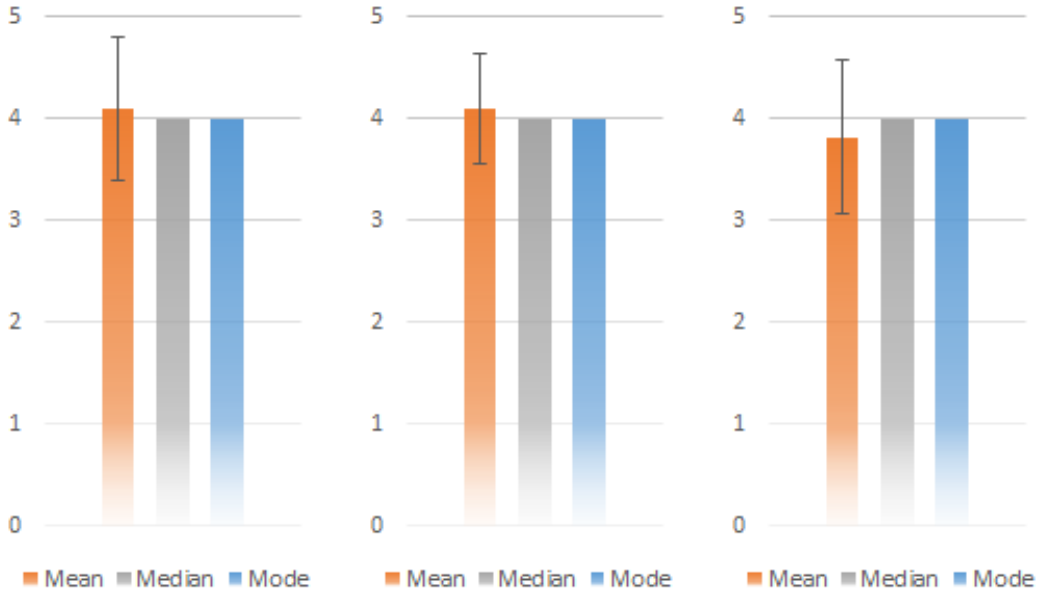
(d) (e) (f)

Figure F.7: Results to the question Q25 (a), Q26 (b), Q27 (c), Q28 (d), Q29 (e), Q30 (f).



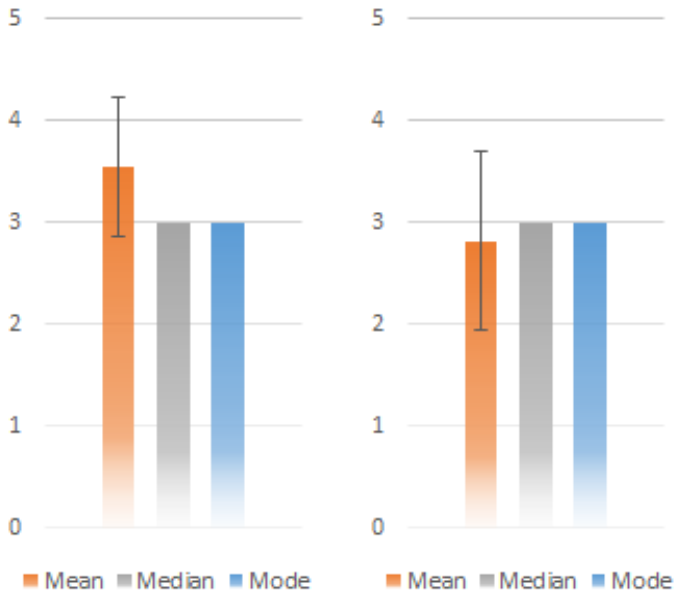
(a) (b) (c)

Figure F.8: Results to the question Q31 (a), Q32 (b), Q33 (c).



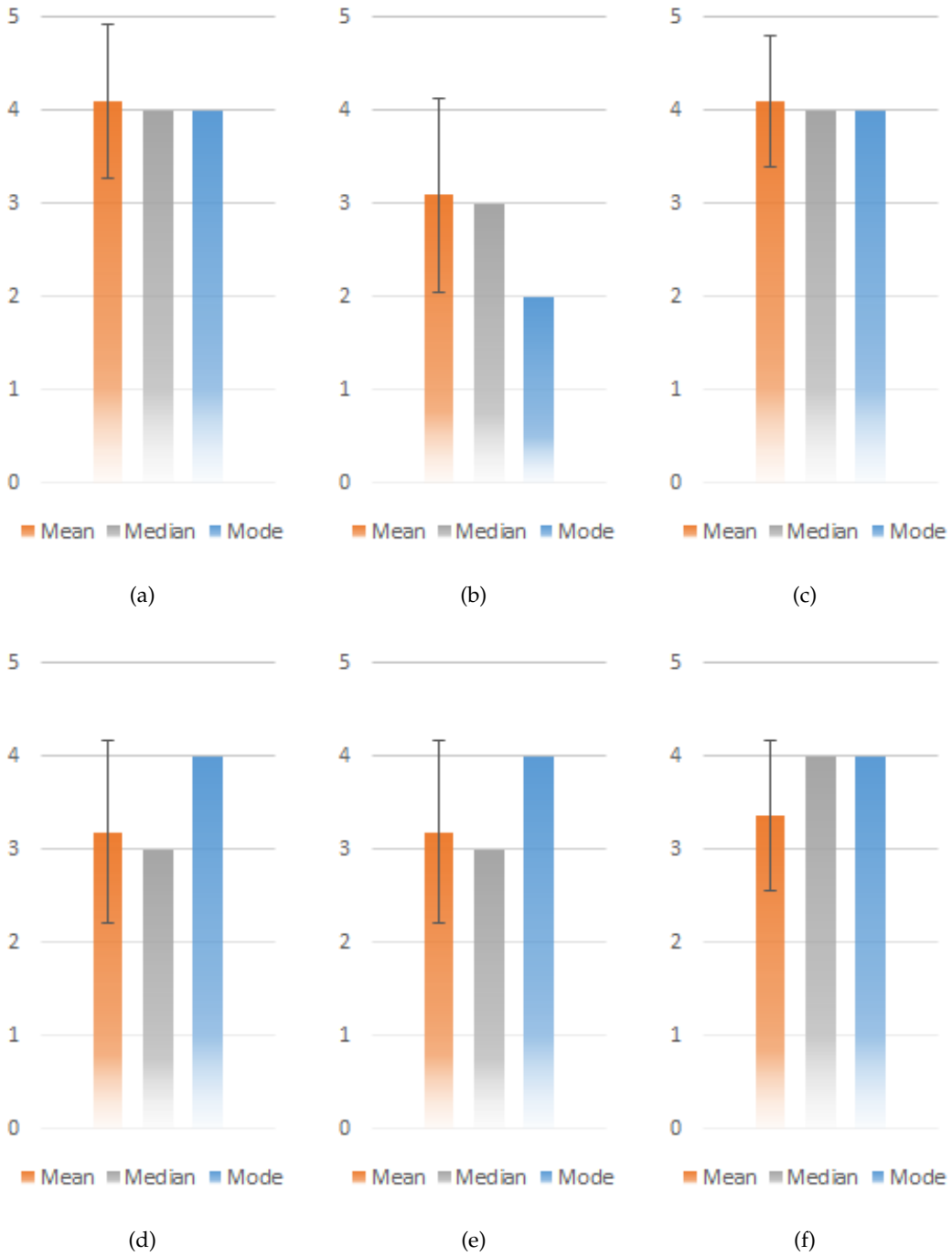
(a) (b) (c)

Figure F.9: Results to the question Q34 (a), Q35 (b), Q36 (c).



(a) (b)

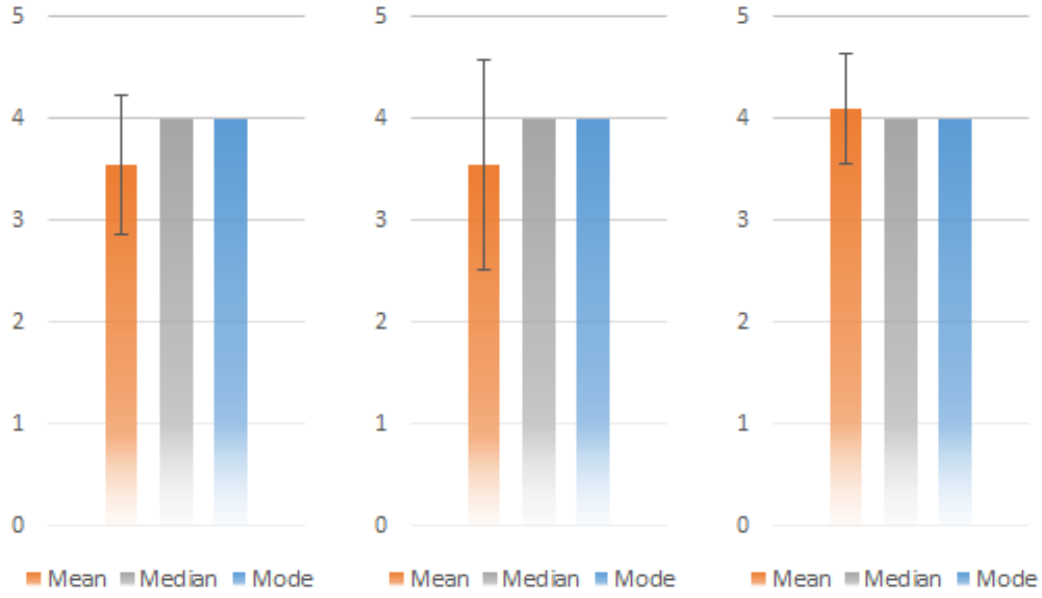
Figure F.10: Results to the question Q37 (a), Q38 (b).



(a) (b) (c)

(d) (e) (f)

Figure F.11: Results to the question Q39 (a), Q40 (b), Q41 (c), Q42 (d), Q43 (e), Q44 (f).



(a) (b) (c)

Figure F.12: Results to the question Q45 (a), Q46 (b), Q47 (c).